Satış fiyatı .....: 171.500 Lira
KDV .....: 1.715 Lira

KDV'li SATIŞ FİYATI ..... 240000 Lira

TOPTAN SATIŞ
tanbul Devlet Kitapları Müdürlüğü, Adana, Ankara, Burdur, Elazığ, Erzurum, İzmir, Samsun, Sivas, Trabzon , Van ve Zonguldak Bölge Şeflikleri.

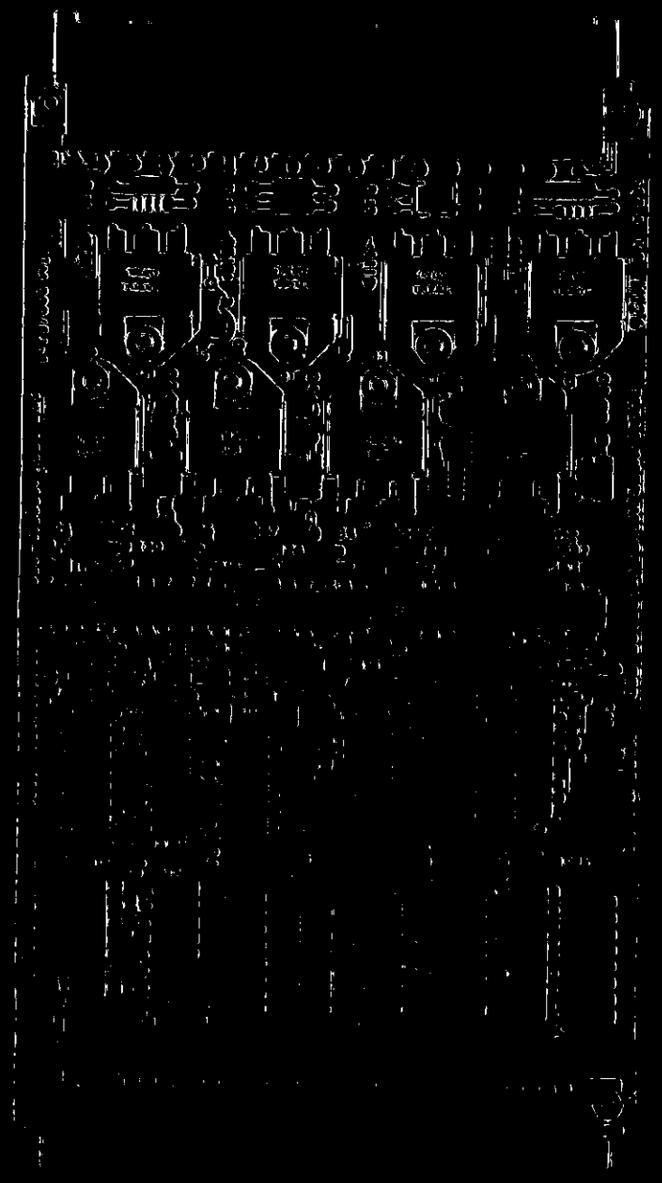
PERAKENDE SATIŞ
Millî Eğitim Yayınevleri ve Bakanlık yayınları satıcısı kitapçılar.

Endüstriyel Okullar İçin **Mikroişlemci - Tabanlı Sistemler - Kitap IV**

ENDÜSTRİYEL OKULLAR İÇİN

MİKROİŞLEMCİ TABANLI SİSTEMLER

KİTAP IV



M.E.B.

18.02.97

2696 57

ENDÜSTRİYEL OKULLAR İÇİN

Mikroişlemci - Tabanlı Sistemler

Kitap IV

A. Potton

Evren Ofset A.Ş. Web Ofset Tesisleri ANKARA - 1994

Millî Eğitim Bakanlığı Yayınları ..... : 2696
Yardımcı ve Kaynak Kitaplar Dizisi ..... : 57

ISBN 975 - 11 - 0867 - 5

Hükûmetimiz ile Dünya Bankası arasında imzalanan Endüstriyel Okullar Projesi çerçevesinde hazırlanan "Mikroişlemci Tabanlı Sistemler Kitap IV" Millî Eğitim Bakanlığı, Talim ve Terbiye Kurulu Başkanlığının 18/05/1994 gün ve 4026 sayılı kararı ile kaynak kitap olarak uygun bulunmuş ve 10.000 adet bastırılmıştır.

Çeviri - Dizgi - Mizampaj : Üniversal Dil Hizmetleri ve Yayıncılık A.Ş.
Çevirmen : Dr. Abidin SÖNMEZ
Editör : Prof. Dr. Sezgin ALSAN
Baskı Hazırlık - Baskı - Cilt : Evren Ofset Basım Sanayii ve Ticaret A.Ş.

© Yayın hakkı: Technician Education Council
Türkçe yayın hakkı Millî Eğitim Bakanlığına aittir. 1994

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak,
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl...
Hakkıdır, Hakk'a tapan, milletimin istiklâl!

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
"Medeniyet!" dediğin tek dişi kalmış canavar?

Arkadaş! Yurduma alçakları uğratma, sakın.
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın...
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri "toprak!"diyerek geçme, tanı :
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı :
Verme, dünyaları alsan da, bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki fedâ?
Şühedâ fışkıracak toprağı sıksan, şühedâ!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüdâ.

Ruhumun senden, İlâhi, şudur ancak emeli :
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar-ki şahâdetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder-varsa-taşım,
Her cerîhamdan, İlâhi, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerred gibi yerden nâ'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyen sana yok, ırkıma yok izmihlâl:
Hakkıdır, hür yaşamış, bayrağımın hürriyet;
Hakkıdır, Hakk'a tapan, milletimin istiklâl!

Mehmet Âkif ERSOY

ATATÜRK'ÜN
GENÇLİĞE HİTABESİ

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk cumhuriyetini, ilelebet, muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin, en kıymetli hazinendir. İstikbalde dahi, seni, bu hazineden, mahrum etmek isteyecek, dahilî ve haricî, bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok nâmüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın, bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dahilinde, iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlilerin siyasî emelleriyle tevhid edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi, vazifen; Türk istiklâl ve cumhuriyetini kurtarmaktır! Muhtaç olduğun kudret, damarlarındaki asîl kanda, mevcuttur!

K. Atatürk

Bilgi çağına girerken bütün ülkelerin üzerinde önemle durdukları ve giderek daha fazla kaynak ayırdıkları sektör eğitimdir. Bilim ve teknolojideki gelişmelere paralel olarak eğitimde kaliteyi yükseltmek, gençlerimize ileri sanayi toplumunun gerektirdiği bilgi, beceri ve davranışları kazandırmak Millî Eğitimimizin temel amaçlarından biridir.

Ülkemizde; ekonomik, sosyal ve kültürel alanlarda olduğu gibi, sanayi alanında da önemli gelişmeler olmaktadır. Nitelikli insangücü ihtiyacının giderek arttığı ülkemizde meslekî ve teknik eğitim büyük önem kazanmaktadır.

Bu alandaki ihtiyacı karşılayabilmek için; çağdaş bilim ve teknolojik metodları bilen, yorumlayan, kullanan, geliştiren ve alanındaki yeniliklere uyum sağlayan, üretken teknik insangücünün yetiştirilmesi gerekmektedir. Bu konuda, teknik öğretim kurumlarımıza büyük iş düşmektedir.

Bu kurumlarımızdaki öğrencilerin iyi yetişmeleri için devletimiz her türlü desteği sağlamakta ve Hükûmetimiz ile Dünya Bankası arasında imzalanan İkraz Anlaşmasıyla yürütülen Endüstriyel Okullar Projesiyle bu okullarımız, çağdaş eğitim imkanlarına kavuşturulmaktadır. Bu okullarımızda çeşitli meslek alanlarında ihtiyaç duyulan 42 adet yabancı teknik ders kitabının tercüme haklarının satın alınması, basım ve dağıtımlarının yapılarak öğrenci ve öğretmenlerimizin istifadesine sunulması, bu proje kapsamında yürütülen faaliyetlerden biridir.

Eğitim ve kültür düzeyleri yüksek, gelişen teknolojiye uyum sağlayabilen toplumlar, geleceğin dünyasının şekillenmesinde önemli rol oynayacaklardır.

Bu ve benzeri çalışmaların ülkemiz için yararlı olmasını diliyorum.

Nevzat AYAZ
Millî Eğitim Bakanı

SUNUŞ

Varlıklarını sürdürmek isteyen toplumlar, kalkınmanın gerektirdiği sayıda nitelikli insangücünü yetiştirmek için eğitime değer vermek ve ona bilimsel ve teknolojik bir nitelik kazandırmak mecburiyetindedirler.

Eğitim, Cumhuriyetin kuruluşundan beri ülkemizde yenileşme aracı olarak görülmüştür. Bugün Eğitim sistemimiz, bilim çağına girilen dünyamızda, toplumumuzun büyüyen ve çeşitlenen ihtiyaçlarına cevap vermede bir takım problemlerle karşı karşıyadır.

Eğitimle ilgili problemlerin çözümünde, yeni yöntemler, teknikler ve araçlar geliştirmek için araştırmalar yapmak, ayrıca daha önce yapılmış araştırmalar sonucu geliştirilen bilgi ve teknolojiyi ülkemize getirmek zorundayız.

Eğitime ayrılacak finansman kaynaklarının sınırlı olması, ülkemizi, genel bütçe dışındaki imkanlardan faydalanmaya zorlamaktadır. Devletimiz bu imkanları araştırmış, mesleki ve teknik öğretim kurumlarımızın bilim ve teknolojide meydana gelen gelişmelere paralel olarak modernleştirilmesi için Uluslararası İmar ve Kalkınma Bankası (Dünya Bankası - IBRD) ile yapılan ikraz Anlaşmasıyla Endüstriyel Okullar Projesi uygulamaya konulmuştur.

Bu projenin amaçları; Endüstriyel Okulların yeni teknoloji ürünü makina ve teçhizatla donatılarak yenilenmesi, çeşitli meslek alanlarında müfredat programlarının geliştirilmesi, burslar ve yurt dışından danışman temin edilmesi yoluyla öğretmenlerimizin eğitilmesi ve çeşitli meslek alanlarında ders kitaplarının tercüme ve yayın haklarının satın alınarak Eğitim Sistemimize kazandırılmasıdır.

Proje ile belirlenen hedeflere büyük ölçüde ulaşılmıştır. Projenin amaçlarından biri olan çeşitli meslek alanlarında (Hidrolik - Pnömatik, Soğutma ve İklimlendirme, CNC, Döküm, Elektronik, Bilgisayar, PLC ve Metal İşleri) teknik ders kitapları, uzmanlardan kurulu komisyonlarca seçilmiş ve tercüme edilerek yayımlanmıştır.

Büyük kaynak ve emek harcayarak Eğitim Sistemimize kazandırdığımız kitapların öğretmen ve öğrencilerimize faydalı olmasını dilerim.

Salih ÇELİK
Projeler Koordinasyon
Kurulu Başkanı

Yazarın Ön sözü

Bu kitap, Hutchinson tarafından Teknik Eğitim Kurulu (TEK) adına yayımlanan mikroelektronik/mikroişlemci serisindeki kitaplardan biridir. Bu serideki kitaplar, Teknik Eğitim Kurulu programlarıyla bağlantılı kitaplarla kullanılmak üzere hazırlanmıştır.

1978 Haziranında, İngiltere Başbakanı, mikroişlemcilerin kullanıma girmesinin belirli sanayi dallarındaki istihdam modeli üzerinde yaratacağı etkiden duyduğu kaygıyı dile getirmiştir. Bundan, Sanayi Bakanlığı ve Ulusal Girişim Dairesi aracılığıyla, mikroişlemci teknolojisini kullanmayı ve geliştirmeyi özendirecek bir girişim doğmuştur.

Mikroelektronik araç ve gereçlerin araştırılması, geliştirilmesi ve üretimi için personel eğitimi ve öğretimi ile bunların diğer sanayi dallarına uygulanması da, böyle bir geliştirme programının önemli bir yönü olarak görülmüştür. 1979'da, teknik eğitim ünitelerinin (ünite, bir öğrenci tarafından ulaşılacak hedeflerin tanımıdır) ve bunlara bağlı eğitim paketlerinin geliştirilmesi için Teknik Eğitim Kurulu tarafından bir proje oluşturulmuştur. Bu proje için gereken para Sanayi Bakanlığı tarafından sağlanmakta ve proje bakanlık namına National Computing Centre Ltd. tarafından yönetilmektedir.

TEK, mikroelektronik üreticileri ve kullanıcıları olarak sanayicileri ve eğitimcileri içeren bir komite kurmuş; buna ek olarak, çok sayıda kişi ve kuruluşa danışılmıştır. Mikroelektronik cihazlarla ilgili tasarım, üretim ve servis alanlarında çalışan teknikerler ve teknik mühendisler için programa ilişkin kitaplar geliştirilmiştir. Yazılan beş kitap şunlardır :

Mikroelektronik Sistemler	Kitap 1
Mikroelektronik Sistemler	Kitap 2
Mikroelektronik Sistemler	Kitap 3
Mikroişlemci-Tabanlı Sistemler	Kitap 4
Mikroişlemci-Tabanlı Sistemler	Kitap 5

Aynı zamanda, mikroelektronik cihazların uygulama alanları ve potansiyelleri ile ilgili genel bir bilgi sahibi olmak isteyen teknikerler için de kitaplar hazırlanmıştır:

Mikroişlemcilerin Değerlendirilmesi	Kitap 3
Mikroişlemci Temelleri	Kitap 4

Bu aşamayı, öğrenim paketlerinin üç yazım ekibi tarafından geliştirilmesi izledi. Bu ekiplerde temel sorumluluğu üstlenen kişiler şunlardır:

Mikroelektronik Sistemler 1, 2, 3	- P. Cooke
Mikroişlemci-tabanlı Sistemler 4	- A. Potton
Mikroişlemci-tabanlı Sistemler 5	- M.J. Morse
Mikroişlemcilerin Değerlendirilmesi	- G. Martin
Mikroişlemci Temelleri	- G. Martin

Kitapların temel özelliklerinin belirlenmesi aşamasında N. Bonnett, proje sorumluluğunu, R. Bertie de onun yardımcılığını yürütmüştür. Bay Bonnett, kitabın yazımı aşamasında da danışman olarak görev yapmaya devam etmiştir. Projenin yöneticisi W-Bolton, yardımcısı da K. Snape'dir.

Kendi Kendine Öğrenim

Kendi kendine öğrenime yardımcı olmak üzere, her bölüm içinde sorulara yer verilmiştir. Sorular, her bölümün sonunda bulunmakta ve bunlara ilişkin referanslar da, kendi kendine öğrenimde en uygun konumu gösterecek biçimde, ilgili metnin kenarında (örneğin S1.2) yer almaktadır. Her sorunun yanıtı kitabın sonunda verilmiştir.
Bu nedenle, bu dizideki kitaplar, hem kendi kendine öğrenimde hem de sınıf ortamında öğretim durumunda kullanılmak amacıyla geliştirilmiştir.

İçindekiler

Giriş

1 Çeviriciler ve Assembly dili 15
1.1 Bilgisayar dili 15
1.2 Çeviriciler 16
1.3 Onaltılı sayıların gösterilmesi 16
1.4 Çeşitli Çeviriciler 17
1.5 Komut kısaltması çevirisi 19
1.6 Assembly listelemesi 21
1.7 Sembolik adresler 21
1.8 Açıklamalar 24
1.9 Veri sabitlerinin gösterilmesi 25
1.10 Sözde-işlemler 27
1.11 Sözdizimi kuralları ve hata mesajları 29
1.12 Mutlak programlar 32
1.13 Yeri değiştirilebilir programlar 34
1.14 Yerleştiriciler ve yeniden-yerleştirme yükleyicileri 35
1.15 Bağlayıcılar ve bağlama yükleyicileri 35
1.16 Çeviriciler için çalışma uygulamaları 37
Kaynaklar ve bibliyografi 39
Sorular 40

2 Mikroişlemci seçimi 41
2.1 Bir mikroişlemciyi nasıl seçeriz ? 41
2.2 İşlem gücü 42
2.3 Bazı gerçek mikroişlemciler 43
2.4 Sözcük uzunluğu 51
2.5 Mikroişlemci mimarisi 53
2.6 Komut takımları 55
2.7 Adresleme 58
2.8 Bir programlama problemi 63
2.9 Değerlendirme Testleri 78
Kaynaklar ve bibliyografi 78
Sorular 79

3 Mikroişlemci seçiminde diğer hususlar 81
3.1 Mikroişlemci G/Ç olanakları 81
3.2 G/Ç işlemlerinin senkronizasyonu 88
3.3 Kesmeler 91
3.4 Mikroişlemci enerji gereksinimleri 104
Sorular 107

4 Mikroişlemci destek birimleri 109
4.1 Giriş 109
4.2 Tamponlama gereksinimi 112
4.3 MİB yonga takımı 117
4.4 Adres kod çözücüleri 122
4.5 G/Ç birimleri 124
4.6 Bellek birimleri 134
4.7 Sayıcı-zamanlayıcılar 142
4.8 8080'li en küçük sistem 144
Kaynaklar ve bibliyografi 144
Sorular 144

5 Tipik mikroişlemci sistemleri 147
5.1 Özel ve genel-amaçlı sistemler 147
5.2 Mikroişlemci denetimli bir çamaşır makinesi 149
5.3 Kelime işlemci 152
5.4 Mikroişlemci geliştirme sistemi 157
Sorular 160

6 Mikroişlemci sistem planlaması 162
6.1 Tasarım ve geliştirme 162
6.2 Geliştirme aşamaları 163
6.3 Sistem özelliği 163
6.4 Maliyet-verimlilik ölçümleri 167
6.5 Sistem akış diyagramı 167
6.6 Başlangıç aşamasındaki donanım kararları 168
6.7 Donanım ve yazılım bölüşümü 170
6.8 Çözümlü örnek 175
Sorular 177

7 Mikroişlemci geliştirme sistemleri 179
7.1 Donanım ve yazılım geliştirme 179
7.2 Geliştirme sistemi düzenlemesi 180

7.3 Destek yazılım 181
7.4 Çapraz-yazılım 190
7.5 Donanım destek olanakları 191
Kaynaklar ve bibliyografi 194
Sorular 194

8 Yazılım tasarımı yöntemleri 195
8.1 İyi yazılım tasarımı yöntemlerinin önemi 195
8.2 Program özellikleri 196
8.3 ~~Programın testi ve~~ bakımı 197
8.4 Büyük ve küçük programlar 198
8.5 Modüler programlar 199
8.6 Bölümlendirme 202
8.7 Yukarıdan-aşağıya yöntemler 205
8.8 Yapısal programlar 206
8.9 İçiçe yapılar 216
Kaynaklar ve bibliyografi 217
Sorular 218

9 Donanım geliştirme 220
9.1 Donanım geliştirme 220
9.2 Prototip donanımın yapımı ve testi 220
9.3 Prototip yapımında adım adım yaklaşımı 222
9.4 Donanım ve yazılım/firmware'ın bir araya getirilmesi 226
9.5 Test ve hata ayıklamanın önemi 227
Sorular 228

10 Mikroişlemci sistem tasarım çalışması 229
10.1 Tasarım/geliştirme programı 229
10.2 Sistem özellikleri 230
10.3 Sistem akış diyagramı 231
10.4 Maliyet-verimlilik kriterlerinin formüllenc' mesi 231
10.5 Donanım birimlerinin seçimi 234
10.6 Sistem donanım düzenlemesi 238
10.7 G/Ç port tasarımı 240
10.8 Bellek tasarımı 240
10.9 Kesme kullanımı 242
10.10 Yazılım tasarımı 244
10.11 Modüller arasında veri aktarımı 246
10.12 Modüler yapı 247

10.13 Prototip donanım yapımı, test ve hata ayıklama 250
10.14 Son test 255
Sorular 255

11 Programlanabilir belleğin pratik yönleri 257
11.1 Mikroişlemci sistem geliştirmede EPROM'un rolü 257
11.2 Programlanabilir bellek birimlerinin seçimi 258
11.3 Kalıcı monolitik bellek birimleri 260
11.4 EPROM programlama 265
11.5 ROM simülasyonu 270
Kaynaklar ve bibliyografi 272
Sorular 272

12 Mikroişlemci hata-bulma 274
12.1 Hata-bulma cihazları 274
12.2 Hatalar ne zaman ortaya çıkar? 275
12.3 Tipik hatalar 275
12.4 Görsel inceleme 276
12.5 Multimetre 277
12.6 Mantık probu 278
12.7 Osiloskop 279
12.8 Tekrarlamalı test dalga biçimlerinin üretilmesi 281
12.9 Test ve self-test programları 284
12.10 Serbest çalışan test 286
12.11 Yazılım hataları 293
Kaynaklar ve bibliyografi 295
Sorular 295

13 Özelleştirilmiş test birimleri 297
13.1 Giriş 297
13.2 Mantık analizörü 297
13.3 İşaret analizörleri 308
Kaynaklar ve bibliyografi 314
Sorular 315

Ekler

A: Intel 8080 merkezi işlem birimi 317
B: Zilog Z80 339
C: Motorola 6800 368
D: Gelişmiş Mikro Birimler AmZ8002 16-bitlik mikroişlemci 386
E: NATIONAL yarıiletken tek-yongalı N-kanallı mikrodenetleyiciler 409
F: Intel 8080A CPU için zamanlayıcı üreteci ve sürücüsü 428
G: Intel 8080A CPU için sistem denetleyici ve yol sürücüsü 431
H: Signetikler yüksek hızlı hex üç durumlu tamponlar ve tersleyiciler 433
I: Sekizli saydam mandal 435
J: Motorola MC6820 çevresel arabirim adaptörü 437
K: Intel 8251A programlanabilir iletişim arabirimi 447
L: Intel 2114 1,024 X 4-bit statik RAM 449
M: Texas Instruments dinamik RAM 451
N: Intel 2716 16K UVEPROM 462
O: Mostek Z80 sayıcı/zamanlayıcı devresi 464
P: Zilog Z6132 yarı-statik RAM 465
Q: Texas Instruments kod çözücü/tekilleyiciler 467
R: ASCII kodları 469

Soruların cevapları 471
İndeks 487
Terimler Sözlüğü 489

Eşim Dorothy'ye, yalnızca notları ustaca daktilo ettiği için değil, kitabın hazırlık dönemi sırasında gösterdiği sabır için de teşekkürü bir borç bilirim.

Bournemouth, Kasım 1982 A. POTTON

Giriş

Amaçları TEC, Mikroişlemci-tabanlı Sistemler U80/674'te belirtilen bu kitap, bir mikroişlemcinin yapısı ve komut takımı hakkında bazı temel bilgilere sahip olan okuyucular için hazırlanmıştır. Kitabın içeriği, bu temel bilginin üzerine, mikroişlemcilerin kendi özellikleri ve sistemler içinde ne şekilde işlev gördükleri hakkında daha geniş ve daha derin bir anlayışın oluşturulmasını sağlayacak şekilde düzenlenmiştir.

Kitabın ana bölümlerinden birinde, mikroişlemci-tabanlı sistem geliştirme ve tasarım yöntemleri tanımlanmaktadır. Özellikle, hem donanım hem de yazılım için, yerleşmiş biçimsel tasarım uygulamalarının kullanımı ve planlanması konuları üzerinde durulmuştur. Yazılım tasarımında yukarıdan-aşağıya yapısal yöntemler kullanılmıştır. Program örnekleri çoğu kez, 8080, Z80 ve M6800 mikroişlemcileri için üç değişik biçimde verilmiştir. Donanım ve yazılımın tasarımı ile, hata ayıklama işlemine yardımcı olmada özel amaçlı mikroişlemci geliştirme sistemlerinin rolü belli bir ayrıntı düzeyinde araştırılmıştır.

Okuyucunun bilgi dağarcığını genişletmek amacıyla, bir dizi farklı mikroişlemcinin özellikleri karşılaştırılmış ve aradaki farklılıklar açıklanmıştır. Örnek olarak verilen birimler, 4bitlik tek-yongalı basit bir mikrobilgisayar ile 16-bitlik güçlü yeni bir mikroişlemci arasında değişen birimlerdir. Ayrıca bir çok uygun destek biriminin özellikleri de tanımlanmıştır. Geliştirilen bu karşılaştırma işlemleri mikroişlemci sistemlerine de uygulanmıştır.

Kitabın daha ileriki bölümleri, mikroişlemci sistemlerinde arızaların yerlerinin belirlenmesi konusundaki ilkeleri tanıtmaktadır. Osiloskop ve multimetre gibi alışılmış araçlar kullanarak arızanın yerinin belirlenmesi yöntemleri açıklanmıştır. Fakat (sözü edilen) bu bölümün büyük bir kısmı, mikroişlemci geliştirme sistemleri, mantık analizörü ve işaret analizörü gibi özel amaçlı cihazların kullanımıyla ilgilidir. Bu bölüm ayrıca EPROMların tanımı, mikroişlemci sistemlerinde kullanımları ve EPROM programlama yöntemlerini de içermektedir.

Temel olarak gerçek yarı iletken elemanları konu aldığından kitabın Ek bölümlerinde bir dizi eleman çeşidi için üretici bilgilerine de kısaca yer verilmiştir. Sayfa sayısının ve fiyatın belli bir düzeyi aşmaması için bu kısım zorunlu olarak sınırlı tutulmuştur; okuyucuya, ilgili üreticiden ürünle ilişkili tüm bilgileri kapsayan kaynakları edinmesini önermekteyiz.

Bölüm 1 Çeviriciler ve assembly dili

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde;*

1 *Amaç programın kaynak programdan derlenmesi işlemini tanımlayabilmeli,*
2 *Bir düzenleyicinin işlevinin, assembly dilinde kaynak program hazırlamak olduğunu söyleyebilmeli,*
3 *Assembly listelemesinin sağladığı bilgileri yorumlayabilmeli,*
4 *Sözde-işlemlerin işlevini belirtebilmeli,*
5 *Sözde-işlemler ve çalıştırılabilir komutlar arasındaki temel farkı tanımlayabilmeli,*
6 *Yeri değiştirilebilir bir programla mutlak programın birbirinden farklı olduğunu belirtebilmeli,*
7 *Bir yerleştirici/yeniden yerleştiricinin işlevini tanımlayabilmeli,*
8 *Bir bağlayıcının işlevini tanımlayabilmeli,*
9 *Bağlanmış programlardaki genel değişkenlere olan gereksinimi açıklayabilmelisiniz.*

1.1 Bilgisayar dili

Günümüzde, her kesimden sürekli artan sayıda insan, bilgisayar programlamayı öğrenmektedir; örneğin ülkemizde bilgisayar programlama öğretimi, şimdi bir çok ilk ve orta öğretim kurumlarında da başlamış durumdadır. Bilgi işleme işinin tipine bağlı olarak öğrencilere başlangıçta, iki farklı yaklaşımdan birini temel alarak programlama yapmaları öğretilecektir. Bunlar aşağıdadır:

1 BASIC, FORTRAN, PASCAL vb. yüksek-düzey dillerde program yazma. Bu diller, programcının zamanını kullanması açısından oldukça verimlidirler. Çünkü oldukça karmaşık işleri gerçekleştirebilecek programlar, bu dillerle nispeten daha çabuk bir şekilde üretilebilirler. Öğrencinin -bu dillerde program yazarken kullandığı- sistemin içerisinde gerçekleşen işlemleri bilmesine gerekmez. Yüksek-düzeyli diller, genel amaçlı bilgi işlem çalışmalarının büyük çoğunluğu için belirgin farkla en iyi yaklaşım olmaktadır.
2 Makine-kodu biçiminde program yazma. Bu yaklaşımla, basit işleri gerçekleştirecek programları bile geliştirmek oldukça uzun zaman alabilir. Bu nedenle programcının zamanını kullanması açısından verimli değillerdir. Makine-kodlu programlar yazımında başarılı olmak için programcı, kullanmakta olduğu bil-

gisayarın iç yapısı ve temel komut takımı hakkında iyi bir bilgiye sahip olmalıdır. Bu nedenle mikroişlemci içeren sistem veya devrelerin tasarımında çalışabilecek kişiler doğal olarak makine-kodu programlama yöntemlerini öğreneceklerdir. Bu metinde okuyucuların makine-kodu programlaması ile bir ölçüde tanışık oldukları varsayılmıştır.

Makine-kodu ve assembly dilindeki programların genel amaçlı bilgi işlem çalışmalarında geliştirilmeleri çok çok uzun zaman alsa da, bunlar bilgisayar devre ve sistem tasarımcıları için hayati öneme sahiptirler. Bunun nedeni makine-kodu komutlarının programcıya, devre ve sistem donanımı üzerinde, yüksek düzey dillerin deyimlerinden çok daha kısa sürede ve doğrudan kontrol olanağını vermesidir.

Bu konunun amacı, belirli bir mikroişlemci için assembly dilinde programlama el kitabı işlevi görmek değildir. Bunun yerine okuyucuya, kullandığı bilgisayar sistemiyle birlikte verilen el kitabına başvurması önerilmektedir. Buradaki amaç, çevirici programları ve assembly dilinin bazı önemli özellikleri hakkında genel bir temel kazandırmaktır. Assembly dilinde komut kısaltması kaynak kodu, mikroişlemci operasyonlarını makine-kodu düzeyinde tanımlamak için genellikle en uygun araç olduğundan ve daha sonraki konular içerisinde bu amaçla kullanıldığından, bunlara kitabın başlangıcında yer verilmiştir. Biçimsel programlama yöntemlerinin daha esaslı şekilde ele alınması, 8. bölüme kadar ertelenmiştir.

1.2 Çeviriciler

Bir mikroişlemciye ilişkin makine kodunda programlar yazarken, programlar önce; MOV, LDA vb. gibi standart komut kısaltması biçiminde yazılır. Ardından programcı, mikroişlemci veri tablolarından çeşitli komut kodlarını arar. Bu şekilde yazılan programlar, en basit bilgisayar sistemi kullanılsa bile belleğe yüklenebilirler.

Genellikle onaltılı formatta yazılan komut kısaltmasının, komut kodu biçimine dönüştürme işlemine elde derleme denir. Basit mikrobilgisayarların çoğunluğu onaltılı kodun doğrudan klavyeden girilmesine imkan tanır, klavye onaltılı kodu ikili biçime dönüştürerek belleğe saklar. Bu şekilde çalışan mikrobilgisayarlara, Motorola MEK6800D2, Nascom 1, Kim, ve Intel SDK80 ile SDK85 örnek olarak verilebilir. Bir çok okuyucunun buna benzer bir sistemin nasıl kullanıldığı konusunda deneyimi olduğuna kuşku yoktur.

1.3 Onaltılı sayıların gösterilmesi

Bir sayısal bilgisayardaki komut takımları ve sayılar esasen ikili biçimdedir, çünkü, klasik sayısal elektronik cihazlar, veriyi birler ve sıfırlar biçiminde kullanırlar. İkili sayıları tanımlamak için uygun bir yöntem, aşağıda gösterildiği gibi onaltılı gösterilme biçimidir:

1100	1001	0001	1111	İkili
C	9	1	F	Onaltılı

Okuyucunun onaltılı sayıların bu şekilde kullanımı hakkında bilgi sahibi olduğu varsayılmıştır.

Bu konu içinde tanımlanan farklı çeviriciler, kullanıcının onaltılı sayıları çeşitli şekillerde tanımlamasını gerektirir. Herhangi bir çevirici programının kullandığı standardı benimsemek yerine aşağıda verilen genel kural dikkate alınmıştır:

Başka şekilde belirtilmedikçe tüm adresler ve makine kodları onaltılı biçimdedir.

Onaltılı sayılarla birlikte hiç bir özel ek kullanılmamıştır. Farklı anlamaların ortaya çıkabileceği yerlerde sayılar, tipleri aşağıda gösterildiği gibi rakamlardan sonra parantez içine yazılarak kuşkuya yer bırakmayacak biçimde tanımlanmıştır:

10 (ondalık)
103 (onaltılı)
1101 (ikili)

1.4 Çeşitli çeviriciler

Programlama tekniklerinin geliştirilmesine ilişkin temel aracın assembly dili olduğu kitaplarda, konunun bir mikroişlemci ve bir çevirici ile sınırlanması yaygın bir kuraldır. Bunun, konu bütünlüğünü sağlaması ve okuyucunun çelişkiye düşmemesine neden olması gibi önemli sonuçları bulunmaktadır. Ancak bu kitabın amaçlarından biri de, okuyucunun farklı mikroişlemci tipleri hakkındaki bilgisini artırmak ve diğerlerinin yanında sağlıklı karşılaştırma yöntemlerinin yerleşmesini sağlamaktır. Dolayısıyla bu bilgi artırma amacına yönelik olarak, metin boyunca çeşitli mikroişlemcileri kapsayan, farklı bir çok mikrobilgisayar sistemleri kullanılarak geliştirilmiş programlar sunulmuştur.

Genelde programlama örneklerinin çoğunluğu üç değişik mikroişlemci koduyla verilecektir: 8080, M6800 ve Z80. Zaman zaman diğer mikroişlemciler için yazılmış program kodlarına da yer verilmiştir. Program örneklerinin hazırlanmasında aşağıdaki mikroişlemci sistemleri kullanılmıştır (program listelemesinin son biçimi Futuredata sistemi kullanılarak hazırlanmıştır):

1 8080, Z80 ve M6800 mikroişlemcileri için çevirici desteği sağlayan GR Futuredata genel amaçlı mikroişlemci geliştirme sistemi
2 Standart CP/M 8080 çeviricisi ASM'yi destekleyen North Star Horizon.
3 Z80 'ZEAP' çeviriciyi destekleyen Nascom 2
4 6800 çeviricis' ASMB'yi destekleyen SWTP 6800 sistemi
5 Z8002 çeviricisini destekleyen Zilog sistemi

Çeşitli sistemlerde bulunan olanakların gösterilmesine ek olarak değişik mikroişlemciler için program kodlarının verilmesinin, metni, mümkün olan en geniş okuyucu kitlesinin kendi kullandıkları sistemlere uyarlayabilmesini mümkün kılacağı ümit edilmiştir. Program örneklerinin büyük çoğunluğu Futuredata sistemi kullanılarak hazırlanmıştır. Genel verimliliğinin yanı sıra bu sistem, M6800, 8080 ve Z80 çeviricileri destekler ve assembly listelemesinde format, sayı gösterilmesi vb. konularda tutarlılık için bazı kriterler sağlamaktadır. Değişik sistemler tarafından sunulan çeviriciler arasındaki farklılıklar, izleyen bölümlerde çeşitli genel özellikler tanımlanırken belirtilecektir.

İzleyen bölümde, belirli bir mikroişlemci için üreticisinin komut kısaltması kullanılarak yazılan programlar, kaynak program olarak adlandırılacaktır. Çevirici, bu kaynak programı makine-kodu biçimine çevirmektedir. Bu işlem sonucunda oluşan makine koduna amaç program adı verilir. Kaynak programı amaç programa dönüştürme işlemine ise derleme denir. Makine-kodu ve veri onaltılı formatta gösterilebilir.

Bir kısaltıcı kaynak program ve bunun makine-kodu amaç programı Şekil 1.1'de gösterilmektedir.

Amaç program (onaltılı) — *Kaynak program*

```
3E  04      MVI  A,X'04'
06  3F      MVI  B,X'3F'
87          ADD  A
32  00 01   STA  X'0100'
```

Şekil 1.1

1.5 Komut kısaltması çevirisi

Sorulara ilişkin, Önsöz'deki nota bakınız

En temel düzeyde bir çevirici programı, komut kısaltması makine-koduna dönüştürmek için bir çevirmen gibi işlev görür. Örnek olarak 8080 mikroişlemcisinin aşağıdaki komut

S1.2 anımsatıcısını ele alalım,

```
INX B
```

Bu komutun kodu
00000100 (ikili) = 04'tür.

Programın çalıştırılabilmesi için önce programın diğer komut kodlarıyla birlikte bu kodun da, bellekte doğru bir yere yerleştirilmesi gerekir. Program, komut kodları belleğe yerleştirildikten sonra çalıştırılabilir. Dolayısıyla 8080 anımsatıcılarına ilişkin çeviricilerin işlevlerinden biri de INX B'yi 04'e dönüştürmektir.

Sayısal bilgisayarlar veriyi ikili biçimde işlerler. Bu nedenle komut kısaltması çevirebilmek için bunların bilgisayara ikili biçimde girilmesi gerekir. Bu işlem normalde, mikroişlemci geliştirme sistemlerinin konsol aygıtları tarafından yapılmaktadır. Tipik bir konsol, daktilodakine benzer bir klavye ile bağlantılı olan bir video görüntüleme terminalinden (VDT) oluşur. Bir tuşa basıldığında o tuşun üzerindeki harf ya da rakama karşılık gelen kod bilgisayara iletilir. O anda çalışmakta olan bir giriş programı karakter kodunu belleğe saklar.

Sayı, harf, noktalama işaretleri vb. göstermek için çeşitli uluslararası onaylanmış kodlar bulunmaktadır. Bunlar arasında en yaygın olanlardan birisi Ek R'de verilen Bilgi Değişimi için Amerikan Standart Kodu (American Standard Code for Information Interchange) 'dur.

ASCII kod karakterlerini VDT klavyesinden alan program normalde, aldığı karakterleri görüntülemek üzere VDT ekranına yansıtacaktır. Bu, kullanıcıya, yararlı bir kontrol olanağı verir. Ayrıca program, girilen metin karakterlerindeki yanlışları kolayca düzeltilebilmesine de olanak sağlayabilir. Bu

programlara, düzenleyici adı verilir. Düzenleyiciler Bölüm
S1.3 7.3'te tartışılmıştır.

Belli bir düzenleyici programın, karakter kodlarını bellekte 1000 konumundan başlayarak sakladığını varsayalım. Programın aşağıdaki iki satırı girildiğinde belleğin bu bölgesinin içerdiği değerler Tablo 1.1'de gösterilmektedir:

```
MVI     A,03
ADD     B
```

Boşluk ve kontrol fonksiyonlarının, satır başı ve satır atlatma karakter kodlarının da kapsandığına dikkat ediniz.

S1.1 Bu şekilde bütün bir kaynak programın karakterleri girildikten sonra artık çevirici programı çalıştırabiliriz. Çevirici kodlanmış programı içeren bellek bölgesini, komut anımsatıcılarını simgeleyen metin karakterlerine ilişkin kombinasyonları arayarak tarar. ADD B gibi bir kombinasyonu tanıdığında çevirici, bunun makine komut koduna karşılık gelen bayt 80'i oluşturur.

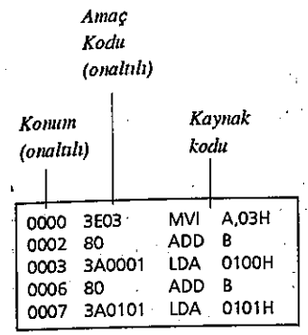
Buraya kadar anlatılanlardan, belli bir mikroişlemciye özgü assembly dilinin, komut takımları aynı olmadıkça bir başka mikroişlemci için kullanılamayacağı açık bir şekilde anlaşılmış olmalıdır.

Çevirici dilleri, uyumluluğun açıkça görülebildiği durumlarda bile mikroişlemcilerde birbirinin yerine kullanılamayabilirler. Örneğin 8080 mikroişlemcisinin komut takımı, Z80 komut takımının bir alt kümesidir. Bu nedenle, bir Z80 çeviricisinin, 8080 için makine-kodu programları oluşturulmasında, ancak, komutların 8080'e uygun olanlarla sınırlı tutulması koşuluyla kullanılabileceği düşünülebilir. Aslında 8080 (Intel) ve Z80 (Zilog)'in üreticileri özdeş komutlar için bile farklı anımsatıcılar kullanmışlardır. Dolayısıyla 04 makine kodu komutu 8080 için INX B, ve Z80 için INC BC şeklinde tanımlanmıştır. 8080 programlarında Z80 çeviricisi kullanmak için programcının, programlarını Z80 komut kısaltması biçimine dönüştürmesi gerekecektir.

Tablo 1.1

Karakter	*Adres*	*İçerik (ASCII)*
M	1000	4D
V	1001	56
I	1002	49
SP	1003	20
A	1004	41
,	1005	2C
0	1006	30
3	1007	33
CR	1008	0D
LF	1009	0A
A	100A	41
D	100B	44
D	100C	44
SP	100D	20
B	100E	42

Komut kısaltması çevirisi işlemiyle ilgili yukarıda verilen tanımlama, bir ölçüde basitleştirilmiş bir tanımlamadır. Komutların bir çoğu yalnızca işlem kodlarından oluşmaz, aynı zamanda işlenenleri de içerir. Örneğin 8080 mikroişlemcisine ilişkin MVI A,03 komutu, 03'ün A kaydedicisine yüklenmesine neden olur. Bu komut kodu iki baytlık bir alan kaplar. Birinci bayt işkodu, 3E ikincisi ise 03 işlenenidir. Bu örnekteki çevirici tarafından gerçekleştirilen işlemler arasında, 03 ASCII kod karakterlerinin ilgili ikili sayıya dönüştürülmesi de bulunmaktadır. Etkin bir assembly dili programcısı olmak için bir çeviricinin iç çalışmasını çok ayrıntılı biçimde bilmenin gerekli olmayışı konunun iyi bir yönüdür.



Şekil 1.2 Assembly listelemenin içerdiği bilgiler

1.6 Assembly listelemesi

Bir makine-kodu amaç programı komut kısaltması kaynak programdan çevrilmesi sırasında çevirici, bir assembly listesi oluşturur. Bu, esas olarak, makine kodu ile kaynak programın her bir anımsatıcı komut tarafından üretilen kopyasıdır. Şekil 1.2'de bir 8080 assembly dili programı listesi gösterilmektedir. Listelemenin içerdiği bilgileri gösteren eklemeler yapılmıştır.

Assembly listelemesi programın belgelerle ispatlanması için değerli bir elemandır ve programdaki hataların ayıklamasında hayati işlev görür. Listelemenin ek bir avantajı da, Bölüm 1.8'de tartışıldığı gibi açıklamalara yer verilmiş olmasıdır. Belgeleme amaçları için bir mikroişlemci geliştirme sisteminde, normalde, program listelerinin basılı kopyalarını alabilmek amacıyla bir yazıcı bulunur.

1.7 Sembolik adresler

Makine kodunda programlar yazarken programcının her bir komut kodunun sonuçta saklanacağı yerlerin takibini yapması gerekir. Çünkü bazı komut kodları diğer komut kodlarının saklandığı bellek bölgelerine bağımlıdır.

Şekil 1.3'te gösterilen program listelemesini ele alalım. Programın sonundaki JNZ koşullu atlama komutunun 0103 şeklinde bir işleneni vardır, bu işlenen atlama koşulunun sağlanması

```
-- 8080/85 ASSEMBLER V01 ---------------------------------------- PAGE 1
                                                                  FUTUREDATA

0000                        ORG     X '0100'
0100    210012              LXI     H, X '1200'
0103    7E                  MOV     A, M
0104    C605                ADI     5
0106    77                  MOV     M, A
0107    23                  INX     H
0108    05                  DCR     B
0109    C20301              JNZ     X '0103'
010C                        END
```

S1.4 *Şekil 1.3*

```
-- 8080/85 ASSEMBLER V01 ---------------------------------------- PAGE 1
                                                                  FUTUREDATA

0000                        ORG     X '0100'
0100    210012              LXI     H, X '1200'
0103    060A                MVI     B, 10
0105    110001              LXI     D, X '0100'
0108    7E                  MOV     A, M
0109    C605                ADI     5
010B    77                  MOV     M, A
010C    E1                  POP     H
010D    23                  INX     H
010E    05                  DCR     B
010F    C20801              JNZ     X '0108'
0112                        END
```

Şekil 1.4

durumunda çalıştırılacak bir sonraki komutun adresidir. Bu nedenle bir bütün olarak tüm komutun kod sözcüğü, halen JNZ komutunun varış adresi olan 0103 konumundaki MOV A,M komutunun bulunduğu yere bağlıdır.

Şimdi varsayalım ki programda, başlangıca yakın bir yerde araya ek komutların konulmasını gerektiren bir hata bulunmuş olsun. Sözgelimi, programın ilk komutunun önceden olduğu gibi 0100 konumunda bulunmasını ve JNZ işleminin varış yerinin, yine önceki gibi MOV A,M olmasını arzu ediyoruz. Bu durumda JNZ komutunun işleneni Şekil 1.4'teki listelemede gösterildiği gibi 0108 olarak değiş-

```
-- 8080/85 ASSEMBLER V01 ---------------------------------------- PAGE 1
                                                                  FUTUREDATA

0000                        ORG     X '0100'
0100    210012              LXI     H, X '1200'
0103    7E          TEKRAR  MOV     A, M
0104    C605                ADI     5
0106    77                  MOV     M, A
0107    23                  INX     H
0108    05                  DCR     B
0109    C20301              JNZ     TEKRAR
010C                        END
```

Şekil 1.5


```
-- 8080/85 ASSEMBLER V01 ---------------------------------------- PAGE 1
                                                                  FUTUREDATA

0000                        ORG     X '0100'
0100    210012              LXI     H, X '1200'
0103    060A                MVI     B, 10
0105    110001              LXI     D, X '0100'
0108    7E          TEKRAR  MOV     A, M
0109    C605                ADI     5
010B    77                  MOV     M, A
010C    E1                  POP     H
010D    23                  INX     H
010E    05                  DCR     B
010F    C20801              JNZ     TEKRAR
0112                        END
```

Şekil 1.6

tirilmelidir. Bu nedenle araya komut ekleme veya komut çıkarma şeklinde yapılan program değişiklikleri dikkatli bir şekilde yapılmalıdır. Sayısal adresler kullanılırken atlama veya dallanma komutlarının varış adreslerinde gereken her değişiklik, ayrıntılı bir şekilde kontrol edilmelidir. Bu sorun hem mutlak hem de bağıl atlama ve dallanmalar için söz konusudur.

Çeviriciler, kaynak program satırlarının başına etiket konulmasına olanak verirler. Etiket daha sonra sembolik bir adres oluşturur, aynı satırdaki çevrilen komut kodu daha sonra bu adreste saklanır. Bu sembolik adres daha sonra diğer komutlarda işlenen olarak kullanılabilir. Şekil 1.5 ve 1.6'da, Şekil 1.3 ve 1.4 ile aynı programlar gösterilmiştir, ancak bunlarda JNZ komutunun işleneni olarak bir sembolik adres kullanılmıştır. Çeviricinin, atlama varış adresi için otomatik olarak doğru sayısal değeri araya eklediği görülmektedir.

Sembolik adres, bir komut kodu içersin ya da içermesin her hangi bir bellek konumunu tanımlamak için kullanılabilir. Bu uygun bir kullanımdır, çünkü sembolik adres kullanımı, veri saklama alanlarına sembolik referans verilmesine izin verir. Bu olanağın tam olarak açıklanabilmesi sözde-işlemler hakkında bir miktar bilgiyi gerektirdiği için, konu, Bölüm 1.10'a kadar ertelenmiştir.

Eğer, etiketler ve sembolik adresler dikkatle seçilirse assembly listelemesinin 'okunabilirliğinin' arttırılmasına yardımcı olabilirler. Örneğin Şekil 1.5 ve 1.6'da JNZ komutunun varış adresi elbette bir döngü komutunun başlangıcıdır. TEKRAR sembolik adresinin seçilmesi, okuyucuya, bir döngünün bulunduğunu hatırlatmada yardımcı olur. Önemsiz gibi görünebilen bu durum aslında önemlidir. Assembly dilindeki programlarda temel sorunlardan biri de, dikkatli bir şekilde belirtilmedikçe ve belgelenmedikçe, programı yazanın dışındaki kişiler tarafından anlaşılmalarının çok zor olmasıdır. Sembolik adreslerin yaratıcı bir şekilde seçilmesi iyi bir program belgeleme özelliklerinden biridir.

```
-- 8080/85 ASSEMBLER V01 ----------------------------------------------- PAGE 1
                                                                          FUTUREDATA

0000                          ORG     X '0100'
0100   210012                LXI     H, X '1200'
0103   222801                SHLD    ISRT1
0106   210013                LXI     H, X '1300'
0109   222A01                SHLD    ISRT2
010C   060A                  MVI     B, 10
010E   2A2801     TEKRAR     LHLD    ISRT1
0111   7E                    MOV     A, M
0112   C605                  ADI     5
0114   2A2A01                LHLD    ISRT2
0117   77                    MOV     M, A
0118   23                    INX     H
0119   222A01                SHLD    ISRT2
011C   2A2801                LHLD    ISRT1
011F   23                    INX     H
0120   222801                SHLD    ISRT1
0123   05                    DCR     B
0124   C20E01                JNZ     TEKRAR
0127   00                    NOP
0128              ISRT1      DS      2
012A              ISRT2      DS      2
012C                         END
```

Şekil 1.7 Açıklama içermeyen program

Bu konunun içinde yer alan listelerin büyük bir bölümünün oluşturulmasında kullanılan 8080 çeviricisi için, bir etiketteki karakter sayısı onaltıyı geçmemelidir. Karakterler alfasayısal (harf veya rakam) olabilirler fakat ilk karakter daima bir harf olmalıdır.

1.8 Açıklamalar

Çeviriciler, programcının kaynak programda uygun noktalara açıklamalar eklemesine olanak tanırlar; bu, assembly listelerini daha okunabilir kılmak için bir başka yardımcı özelliktir. Şekil 1.8, Şekil 1.7'deki programla aynı kodlara sahip olan fakat açıklamaların eklendiği bir programın listesini göstermektedir. Eğer açıklamalar dikkatle yazılırsa, programın kuruluş mantığını tanımlayabilir ve aynı zamanda programın gerçekleştirdiği işlemler hakkında bir 'tekrarlanan rehber' sunarlar.

Şekil 1.8'de, örnekte kullanılan 8080 çeviricisinde, komut kısaltmalarını içeren bir satırda, işlenenin ardından açıklama bölümleri girme olanağı olduğu görülmektedir. Açıklamalar, kaynak programda, bir satırın tamamını da kaplayacak uzunlukta olabilir. Ancak bu durumda satır, bir yıldız işaretiyle başlamalıdır. Açıklama tanımlanmalarının ayrıntıları kullanılan çeviriciye bağlı olarak değişir. Örneğin bazı çeviricilerde açıklamalardan önce noktalı virgül konulması gerekir.

Açıklamalar yalnızca assembly listesini etkilerler. Çevrilmiş makine kodu programı etkilemezler.


```
----- 8080/85 ASSEMBLER V01 ------------------------------------- PAGE 1
                                                                  FUTUREDATA -------

                * ***********************************************  *
                *  BU PROGRAM BİR LİSTEDEKİ HER SAYIYA 5 (ONDALIK) EKLER . SONUÇLAR  *
                *  YENİ BİR LİSTEDE SAKLANIR. ESKİ LİSTE 1200'DEN BAŞLAR VE ON GİRİŞ  *
                *  İÇERİR. YENİ LİSTE 1300'DEN BAŞLAMAKTADIR                         *
                *                                YAPI                                 *
                *                                ****                                 *
                *  ISARETCI1 = X'1000'                                                *
                *  ISARETCI2 = X'1300'                                                *
                *  DONGU = 10                                                         *
                *  DO UNTIL DONGU = 0                                                 *
                *      OLUNCAYA KADAR İŞLEME DEVAM ET                                 *
                *      ISARETCI1 TARAFINDAN GÖSTERİLEN SAYIYI AL                      *
                *      SAYI = SAYI + 5                                                *
                *      SAYIYI ISARETCI2'NİN GÖSTERDİĞİ ADRESTE SAKLA                  *
                *      ISARETCI2 = ISARETCI2 + 1                                      *
                *      ISARETCI1 = ISARETCI1 + 1                                      *
                *      DONGU = DONGU -1                                               *
                *  END DO                                                             *
                *                                                                     *
                  ***********************************************

0000                          ORG     X '0100'
0100   210012                LXI     H, X '1200'
0103   222801                SHLD    PTR1          ISARETCI1 = X '1200'
0106   210013                LXI     H, X '1300'
0109   222A01                SHLD    PTR2          ISARETCI2 = X '1300'
010C   060A                  MVI     B, 10         HL'DEKİ ISARETCI1'İ AL
010E   2A2801     TEKRAR     LHLD    PTR1          GÖSTERİLEN SAYIYI AL
0111   7E                    MOV     A, M          SAYI = SAYI + 5
0112   C605                  ADI     5             HL'DEKİ ISARETCI2'Yİ AL
0114   2A2A01                LHLD    PTR2          SAYIYI GÖSTERİLEN ADRESTE
0117   77                    MOV     M, A          SAKLA
0118   23                    INX     H             ISARETCI2 = ISARETCI2 + 1
0119   222A01                SHLD    PTR2          ISARETCI2'Yİ BELLEĞE YAZ
011C   2A2801                LHLD    PTR1          HL'DEKİ ISARETCI1'İ AL
011F   23                    INX     H             ISARETCI1 = ISARETCI1 + 1
0120   222801                SHLD    PTR1          ISARETCI1'İ BELLEĞE YAZ
0123   05                    DCR     B             DONGU = DONGU -1
0124   C20E01                JNZ     TEKRAR        İŞ SONU?
0127   00                    NOP
                *
0128              PTR1       DS      2             ISARETCI1
012A              PTR2       DS      2             ISARETCI2
012C                         END
```

Şekil 1.8

1.9 Veri sabitlerinin gösterilmesi

Bölüm 1.3'te, bütün metin boyunca program kodu veya adresleri tanımlayan sayıların, tersi belirtilmedikçe onaltılı olarak kabul edildiğini söylemiştik. Dolayısıyla onaltılı sayıları ayırdetmek için başka her hangi bir özel format kullanılmamıştır. Ancak assembly kaynak programların çoğunluğunda, onaltılı sayıların, ondalık sayılardan ayrılması gerekir. Program listelemesi oluşturmak için burada kullanılan tüm çeviricilerde, kaynak programda, fazladan hiç bir tanımlama içermeyen bir sayı ondalık olarak kabul edilmiştir. Tablo 1.2'de, onaltılı sayıların farklı çeviricilerde nasıl tanımlanmaları gerektiği gösterilmektedir.

Tablo 1.2

Çevirici	Ondalık sayı	Onaltılı sayı
Futuredata 8080	nn	X'nn'
Futuredata Z80	nn	X'nn'
Futuredata 6800	nn	X'nn'
CP/M 8080 ASM	nn	nnH
Nascom Z80 ZEAP	nn	nnH
SWTP 6800 ASMB	nn	$nn
Zilog Z8000 PLZ/ASM	nn	%nn

Tablo 1.3

Çevirici	Komut
Futuredata 8080	MVI A,X'F3'
Futuredata Z80	LD A,X'F3'
Futuredata 6800	LDAA #X'F3'
CP/M 8080 ASM	MVI A,0F3H
Nascom Z80 ZEAP	LD A,0F3H
SWTP 6800 ASMB	LDAA #$F3

Farklı çeviriciler için kaynak kodu komutlarının bir listesi Tablo 1.3'te verilmiştir. Komutların her biri söz konusu mikroişlemcinin A kaydedicisinin F3 ile yüklenmesine neden olur. CP/M8080 ASM ve Nascom Z80 ZEAP örneklerinde onaltılı sayıların ilk karakterlerinin harf olmamasına dikkat ediniz. Bu nedenle F3, 0F3H şeklinde yazılmalıdır. Ayrıca SWTP 6800 ASMB ve Futuredata 6800 örneklerinde '#' karakterinin onaltılı sayı formatının bir parçası olmadığına, ancak dolaysız adresleme modunun tanımlandığını gösterdiğine de dikkat ediniz.

Bazı çeviriciler uygun karakterler ekleyerek programcının, sayıları ikili ve sekizli biçimlerde tanımlamasına da olanak verirler. Örneğin CP/M 8080 çeviriciler Şekil 1.9'daki programda gösterildiği gibi ikili ve sekizli sayıları kabul eder. Sayıya eklenen 'B' harfi, sayıyı ikili olarak tanımlarken 'O' harfi sekizli olduğunu gösterir. Ondalık sayılara eklenen 'D' harfinin yazılıp yazılmaması isteğe bağlıdır. Bu çeviricide onaltılı sayıların bir sayısal karakterle başlaması gerektiğine dikkat ediniz. Bu nedenle F3, 0F3 şeklinde tanımlanmalıdır.


```
0000  0EB3      MVI  C, 10110011B
0002  061F      MVI  B, 037O
0004  018303    LXI  B, 899D
0007  3AC23F    LDA  3FC2H
000A  3E41      MVI  A, 'A'
000C  21335A    LXI  H, '3Z'
```

Şekil 1.9 CP/M 8080 çevirici kullanarak sabitlerin gösterilmesi

Şekil 1.9'un son iki satırı ASCII kodlu karakterlerin kaynak programda tırnak içine alınarak kullanılışlarını göstermektedir. Örneğin MVI A,'A' komutunu ele alalım, görüyoruz ki bu komut iki baytlık 3E41 koduna çevrilmektedir. Burada 3E, MVI için işkodu ve 41 de A'nın ASCII-kodu biçimidir. Çeviricinin metin dizilerini ASCII kodu biçimine dönüştürme yeteneği, yazıcıdan ya da VDT'den mesaj görüntüleyen programların yazımında çok yararlıdır.

1.10 Sözde (Sahte)-işlemler

Bir programın çevrilmesi sırasında çeviricinin, geçerli komut kısaltmalarını ararken kaynak programın kodlu karakterlerini taradığı anımsanacaktır. Sözde-işlemler, mikroişlemci komut takımına dahil olmayan ancak, çevirici tarafından tanınan komut kısaltmalarıdır. Anımsatıcılar, komut takımına mensup olmadıklarından, sözde-işlemler, çalıştırılabilir komutlara çevrilemezler. Bunlar programcının, çe-
S1.5, viriciye, çevirme işlemi sırasında kullanılmak üzere bilgi aktarmasına yarayan
1.6 araçlardır.

Şekil 1.10, ilki ORG olan sözde-işlemler içeren bir program için assembly listesidir. Bu sözde-işlem iki kez kullanılmıştır. Birincisinde kaynak programda aşağıdaki satırda geçmektedir:

```
ORG     X'1000'
```

S1.8 Program çevrildiğinde, çevirici bu satırı, programın bir sonraki çalıştırılabilir komutu için makine kodunun 1000 konumundan başlayarak saklanacağı anlamında algılayacaktır. Böylece daha sonraki satırlarda yer alan MVI B,50 anımsatıcısı 06,32 şeklinde iki baytlık kod şeklinde çevrilecektir, bunun sonucu olarak 1000 ve 1001 konumlarında saklanacağı kabul edilir. Bu makine kodu komutların aslında bu noktada bellekte saklanmadıklarını vurgulamak gerekir. Çevirici, bellek konumunun önemli olduğu komut adres işlenenlerini çevirirken basitçe bu bilgiyi kullanır. Şekil 1.10'daki JNZ komutu, bunun tipik bir örneğidir. İlk komutun 1000 konumunda saklanacağı gerçeğinden yola çıkarak çevirici SONRGIR (NEXTIN)

```
..... 8080/85 ASSEMBLER V01 ..........................  PAGE  1
                                                        FUTUREDATA .........

                 * ****************************************** *
                 * BU PROGRAM 1F PORT'U ARACILIĞIYLA ELLİ SAYI ALIR VE BU SAYILARI ASCII *
                 * KODUNA ÇEVİRİR.                                                       *
                 * KOD KARAKTERLERİ 1F00'DA BAŞLAYAN BİR LİSTEDE SAKLANIR                *
                 * DURUM SÖZCÜĞÜ, 1E PORTU ARACILIĞIYLA GİRİŞTİR                         *
                 * SAYI KAYNAĞI DURUM SÖZCÜĞÜ - 0'DA HAZIR DEĞİLDİR                      *
                 *                          YAPI                                         *
                 *                          ****                                         *
                 * DONGU=50(ONDALIK)                                                     *
                 * ISARETCI = X '1F00'                                                   *
                 * DO UNTIL DÖNGU = 0                                                    *
                 * STATUS = HAZIR OLUNCAYA DEK İŞLEM YAP                                 *
                 * DURUM SÖZCÜĞÜNÜ TARA                                                  *
                 * İŞ SONU                                                               *
                 * ISARETCI1 TARAFINDAN GÖSTERİLEN SAYIYI AL                             *
                 * SAYIYI GİR                                                            *
                 * ASCII'YE DÖNÜŞTÜR                                                     *
                 * GÖSTERİLEN ADRESTE SAKLA                                              *
                 * ISARETCI = ISARETCI + 1                                               *
                 * DONGU = DONGU -1                                                      *
                 * END DO                                                                *
                 * ****************************************** *

0000                          ORG     X ' 1000 '
1000  0632                    MVI     B, 50                    DONGU = 50
1002  21001F                  LXI     H, LISTE                 ISARETCI = LISTE = H
1005  DB1E      SONRGIR       IN      X ' 1E '                 DURUM GİRİŞİ
1007  CA0510                  JZ      SONRGIR
100A  DB1F                    IN      X ' 1F '                 SAYI GİRİŞİ
100C  0E30                    MVI     C, X ' 30 '
100E  B1                      ORA     C                        ASCII'YE DÖNÜŞTÜR
100F  77                      MOV     M, A                     LISTE'DE SAKLA
1010  23                      INX     H                        ISARETCI = ISARETCI + 1
1011  05                      DCR     B                        DONGU = DONGU -1
1012  C20510                  JNZ     SONRGIR

1015                          ORG     X ' 1F00 '
1F00            LISTE         DS      50                       50(ONDALIK) YER AYIR
1F32                          END
```

Şekil 1.10

etiketli satır için üç baytlık kodun 1005, 1006 ve 1007'de saklanması gerektiğini belirler. Bu nedenle JNZ SONRGIR (JNZ NEXTIN) komutunu, 1005 işleneni ile birlikte çevirir.

Sözde-işlemler, programın son iki satırında da yer almaktadır. DS sözde-işlemi, bellekte bir alanı tanımlar, bu alanın ilk konumuna bir sembolik adres aracılığıyla başvuru yapılabilir. Şekil 1.10'un son iki satırındaki ORG ve DS sözde-işlemleri 1F00'dan başlayan elli bellek konumu tanımlamaktadır. İlk konuma, yani 1F00'a, program içinde LISTE sembolik adresi kullanılarak gönderme yapılabilir.

LISTE sembolik adresinin, doğrudan adresleme kullanılarak A kaydedici içeriğini 1F00 konumunda saklamak için kullanılabileceğine dikkat ediniz.

Böylece aşağıdaki komut geçerli olacaktır:

```
STA LISTE
```

LISTE ayrıca, dolaysız adresleme modunda da bir işlenen olarak kullanılabilir. Aşağıdaki komut çalıştırıldığında, kaydedici çifti H,L'yi, LISTE'in değeri ile yani 1F00'la yükler:

```
LHLD LISTE
```

S1.7 Sözde-işlemler çeviriciden çeviriciye farklılık gösterirler. Buradaki amaç bunların işlevlerini bir kaç örnek yardımıyla genel olarak tanımlamaktır. Okuyucuya, mevcut sözde-işlemler hakkında ayrıntılı bilgi edinmek için kullandığı çeviricinin el kitabına başvurmasını öneririz. Kitabın ileriki bölümlerinde önemli sözde-işlemler kullanıldığında bunların işlevleri ilk geçtikleri yerde anlatılacaktır.

1.11 Sözdizimi kuralları ve hata mesajları

Bu bölümde bir assembly dili kaynak programının formatı tartışılmıştır. Çeviriciler gündeme geldiğinde ilk olarak, kaynak kodun satır format veya düzeni için kurallar saptanır. Eğer bu kurallara uyulmazsa çevirici, istenen makine-kodu komutlarını çevirmeyecektir.

Şu anda, bu konudaki bir çok örnekte kullanılan Futuredata çeviricisinin satır formatına aşina durumdasınız. Kaynak kodu satırı dört alana ayrılmıştır. Bunlar etiket, işkodu, işlenen ve açıklama alanlarıdır. Bu alanların belirtildiği tipik bir program satırı aşağıdadır:

Etiket	*Opkod*	*İşlenen*	*Açıklama*
(Label)	*(Opcode)*	*(Operand)*	*(Comment)*
BASLA	OUT	35H	ETİKETİN KOPYASINI AL

Burada verilen birçok örnekten de görüleceği gibi, kaynak programın her satırında bu alanların hepsinin doldurulmuş olması koşulu yoktur. Ancak alanlar doldurulurken belirli bazı kurallara uyulması gerekir. Kuralların biri, alanların daima doğru sırada olmasıdır. Bu sözdizimi kuralı çeviriciye ilişkin kullanıcı el kitabında açıklanmıştır. Sözdizimi kuralları, farklı çeviriciler için genel hatlarıyla birbirine benzer fakat ayrıntıları değişir.

Çevirme işlemi süresince çevirici program, kaynak programı, sözdizimi hataları

```
* ERROR DEMONSTRATION
(HATA GÖSTERİMİ)
BASLA   MV      B,A
*
        MVI     A, X '31FD'
*
        OUT     B, X, '35'
*
BASLA   OUT     X, '35'
*
        LDA     VERI
*
        JP      BASLA
*
        END
```

Şekil 1.11

yönünden kontrol eder. Bu türde hataların bulunduğu satırlar assembly listesinde belirtilir. Genellikle hatanın tipi ile ilgili bazı işaretler de verilir.

Şekil 1.11, Futuredata çeviricisi için yazılan ve içinde, özellikle yapılmış hatalar bulunan bir 8080 assembly dili programını göstermektedir. Şekil 1.12'de assembly listelemesi görülmektedir. Görüldüğü gibi sözdizimi hataları açık bir şekilde belirtilmiştir. Şekil 1.13'te ise aynı programın CPM/M8080 çeviricisi ASM için yazılmış şekli bulunmaktadır. Burada da yine hatalar belirtilmiştir fakat bu belirtme burada çok daha az karmaşık şekilde yapılmıştır. Listelemede hatalı komutun karşısındaki bir harf hatanın bulunduğu yeri tanımlamaktadır. Sözdizimi hatasının tipini göstermek için, harf, assembly dili el kitabındaki hata tipleri listesinde bir girişi ifade eder. Dolayısıyla listelemenin başında ikinci komut için yer alan V harfi bir veri hatasının kodudur. MVI A komutu için 31FD işlenenli uygun değildir çünkü bu komut için iki onaltılı basamağı aşmayan bir işlenen gerekmektedir.

Okuyucu, Şekil 1.11 ve 1.13'ü karşılaştırırsa, iki çevirici arasında başka küçük farklılıklar da bulacaktır. Her iki çevirici de

```
                                                        PAGE 1
--- 8085/85 ASSEMBLER V01  ------------------  FUTUREDATA -

                    * ERROR DEMONSTRATION  (HATA GÖSTERİMİ)
0000                BASLA     MV        B, A
** DUPLICATE SYMBOL   ** (SEMBOL TEKRARI)
** INVALID OPCODE   ** (GEÇERSİZ İŞKODU)
                    *
0000  3EFD                    MVI       A, X '31FD'
                    *
0002  D300                    OUT       B, X, '35'
** INVALID OPERAND   ** (GEÇERSİZ İŞLENEN)
** UNDEFINED SYMBOL   ** (TANIMSIZ SEMBOL)
                    *
0004  D335                    OUT       X, '35'
** DUPLICATE SYMBOL   ** (SEMBOL TEKRARI)
                    *
0006  3A0000                  LDA       VERI
** UNDEFINED SYMBOL   ** (TANIMSIZ SEMBOL)
0009  F20400                  JP        BASLA
                    *
000C                          END
```

Şekil 1.12


```
                    ;         ERROR DEMONSTRATION
                    ;
L                   BASLA     MV        B,A
                    ;
V0000   3EFD                  MVI       A, 31FDH
                    ;
S0002   D300                  OUT       B, 35H
                    ;
0004    D335        BASLA     OUT       35H
                    ;
U0006   3A0000                LDA       VERI
                    ;
0009    F20400                JP        BASLA
```

Şekil 1.13

kaynak programda açıklama yapılmasına olanak tanır. Bu açıklamalar çevirme sırasında dikkate alınmazlar fakat assembly listelerinin anlaşılmasında önemli yararlar sağlarlar. İki çevirici arasındaki farklılık, açıklamaların tanımlanma biçimindedir. Futuredata çeviricisinde tamamı açıklama içeren satırlar, yıldız işaretiyle başlamalıdır, oysa CP/M 8080 ASM'de, Bölüm 1.8'de anlatıldığı gibi bunun için bir noktalı virgül gerekmektedir. Futuredata çeviricisinde bir komutun açıklama alanındaki açıklamanın başına her hangi bir tanımlayıcı karakter koyulması gerekmezken, CP/M 8080 ASM'de yine noktalı virgüle ihtiyaç vardır. Okuyucu bu küçük farklılıkları ayırdetmede zorluk çekmemelidir.

```
                                                        PAGE  1
----- 6800/02 ASSEMBLER V01 ------------------------  FUTUREDATA ---------

                  * ******************************************  *
                  *                                            *
                  * M6800 'Z' KARAKTERİNİN LİSTEDE ARANMASI    *
                  * BU PROGRAM 1000 - 17FF BELLEK KONUMLARINDA 'Z' NİN ASCII *
                  *                                            *
                  * KODUNUN İLK GÖRÜNDÜĞÜ YER İÇİN ARAMA YAPAR *
                  *                                            *
                  * DÖNGÜDEN ÇIKMA DURUMUNDA 'Z'NİN BULUNDUĞU ADRES 2000'DE *
                  *                                            *
                  * SAKLANIR                                   *
                  *                                            *
                  * 'Z' BULUNAMAZSA (DÖNGÜDEN) ÇIKMA DURUMUNDA 2000,2001 'İN İÇERİĞİ *
                  *                                            *
                  * 1800'DÜR                                   *
                  *                                            *
                  * CIK                                        *
                  *                                            *
                  * ******************************************  *
0000                          ORG       X ' 3000 '
3000  CE1000      ARAMA       LDX       #X ' 1000 '            İNDEKS KAYDEDİCİSİNİ HAZIRLA
3003  A600        ARAMA1      LDAA      0, X                   BİR SONRAKİ KARAKTERİ AL
3005  815A                    CMPA      # ' Z '               BU KARAKTER 'Z' Mİ?
3007  2706                    BEQ       ARAMA2                 EVETSE DALLAN
3009  08                      INX                              İSARETCİ'Yİ ARTTIR
300A  8C1800                  CPX       #X ' 1800 '            LİSTENİN SONU MU?
300D  26F4                    BNE       ARAMA1                 DEĞİLSE YENİDEN YAP
300F  FF2000      ARAMA2      STX       X ' 2000 '             ADRESİ SAKLA
3012  39                      RTS                              ÇIK
3013                          END
```

Şekil 1.14 M6800 mutlak program

1.12 Mutlak programlar

Şekil 1.14'te gösterilen program listelemesi, M6800 için bir mutlak program örneğidir. Bu tip bir program yazarken programcı, programın çalıştırılması sırasında komutların saklandığı alanları program içinde tanımlar. Bu tanımlama normalde bir ORG ya da benzer bir sözde-işlem aracılığıyla yapılır.

Mutlak programlar basit olma avantajına sahiptirler ve program çok büyük boyutlu olmadığı sürece yeterlidir.

Modern programlama yöntemleri büyük programları modüler biçimde yazmayı tercih etmektedir. Bu anlayış, programlama çalışmasını daha küçük bir kaç alt-göreve bölmeyi gerektirmektedir. Daha sonra alt-görevlerden her birine, kendi içinde bir programlama işi olarak yaklaşılmaktadır. Alt programlar bağımsız program modülleri olarak yazılmakta ve hata ayıklama işleminden geçirilmektedir. Bu aşamada eğer gerekiyorsa, alt-programların kendileri de başka altprogramlara bölünebilirler. Tüm alt programlar yazıldıktan, hataları ayıklandıktan ve tümüyle test edildikten sonra bütün bir programı oluşturacak şekilde birleştirilmelidir.

Birbirinden ayrı alt-programları daha büyük bir program oluşturmak için bir araya getirmek bazı sorunlar doğurur. İdealde alt-programlar, çalıştırma öncesinde bellekte kapladıkları alan açısından ekonomik fakat üstüste binmeyi önleyen bir şekilde saklanmalıdır. Bu, her alt-programın kapladığı bellek alanının dikkatli bir şekilde izlenmesi ve ORG sözde-işlemlerinin uygun şekilde kullanılmaları ile gerçekleştirilebilir.

Şekil 1.15 ve 1.16 7D çıkış portundan on karakter alan genel bir programlama görevine ilişkin alt-programların listelemelerini göstermektedir. Çıkış işlemi 7E durum port adresinden alınan durum bilgisi ile anlaşma yöntemini kullanmaktadır. Durum sözcüğü sıfır ise çevre biriminin veri için hazır olmadığı kabul edilmektedir. Okuyucunun bu aşamada programı ayrıntılı bir şekilde anlaması gerekmemektedir.

Listeleri Şekil 1.15 ve 1.16'da gösterilen altyordamlar, ayrı ayrı hata ayıklama işleminden geçirilebilir ve denenebilirler. OPMSG yordamında hata ayıklama ve test etme işlemleri OPCH'nin yerine geçen bir kukla altyordamının kullanımını gerektirir. Bu kukla altyordam basitçe tek bir RET komutundan oluşabilir. Bu geliştirme aşamasında programcı, -altyordamın programın kendi içinde bulunması koşuluyla- programı, bellekte herhangi uygun bir alana yerleştirebilmek için ORG sözde-işlemini kullanabilir. Doğru yerleştirme konusunda dikkatli olunması, yalnızca, iki ya da daha çok altyordam veya başka programların bellekte bulunması


```
                                                          PAGE  1
..... 8080/85 ASSEMBLER V01 ..........................  FUTUREDATA .........
                                  OPMSG
                 * ****************************************  *
                 *  BU ALTYORDAM X'7'D PORTU ARACILIĞIYLA ON VERİ BAYTI UZUNLUĞUNDA  *
                 *  BİR MESAJI GETİRİR VERİ BELLEKTE X'2000' KONUMUDAN BAŞLAYARAK    *
                 *  SAKLANMAKTADIR                                                   *
                 *                          YAPI                                     *
                 *                          ****                                     *
                 *  DONGU=10                                                         *
                 *  SONRADRES = X '2000'                                             *
                 *  DO UNTIL DONGU = 0                                               *
                 *          SONRADRES'TEN BAYT ÇIKIŞI YAP                            *
                 *          SONRADRES = SONRADRES + 1                                *
                 *          DONGU = DONGU -1                                         *
                 *  END DO                                                           *
                 * ****************************************  *

0000                          ORG     X '1009'
1009  060A       OPMSG        MVI     B, 10          DONGU = 10
100B  210020                  LXI     H, VERI        İŞARETÇİYİ KUR
100E  CD0010     SONRADRES    CALL    X '1000'       OPCH'YI ÇAĞIR
1011  23                      INX     H              İŞARETÇİYİ ARTTIR
1012  05                      DCR     B              DONGU = DONGU - 1
1013  C20E10                  JNZ     SONRADRES
1016  C9                      RET
                 *
1017                          ORG     X '2000'
2000             VERI         DS      10
200A                          END
```

Şekil 1.15 Mutlak altyordam

```
                                                          PAGE  1
..... 8080/85 ASSEMBLER V01 ..........................  FUTUREDATA .........
                                  OPCH
                 * ****************************************  *
                 *  BU ALTYORDAM X'7D' PORTUNA BİR BAYT YERLEŞTİRİR DURUM SÖZCÜĞÜ     *
                 *  X'7E' PORTUNDAN OKUNUR BİRİM HAZIR OLMADIĞI ZAMAN DURUM = 0 DIR   *
                 *  VERİ BAYTININ ADRESİ GİRİŞTE HL'DE OLMALIDIR                       *
                 *                          YAPI                                       *
                 *                          ****                                       *
                 *  DO UNTIL DURUM = HAZIR                                             *
                 *          DURUM SÖZCÜĞÜNÜ GİR                                        *
                 *  END DO                                                             *
                 *  VERİ BAYTININ ÇIKIŞINI YAP                                         *
                 * ****************************************  *

0000                          ORG     X '1000'       DURUM SÖZCÜĞÜNÜ AL
1000  DB7E       OPCH         IN      X, '7E'
1002  CA0010                  JZ      OPCH           VERİ BAYTINI AL
1005  7E                      MOV     A, M           VERİ BAYTININ ÇIKIŞINI YAP
1006  D37D                    OUT     X '7D'
1008  C9                      RET
1009                          END
```

Şekil 1.16 Mutlak altyordam

durumunda gereklidir. Bu noktada programcı, programların bellekte ekonomik bir şekilde, üstüste binmeden paketlenmelerini sağlayabilmek için ORG sözde-işlemlerini, yeniden gözden geçirmelidir.

Programları, her ne kadar modüler biçimde yazmak mümkünse de, mutlak assembly kullanılması halinde bile, program modüllerinin sayısı ve büyüklüğü arttıkça ortaya çıkan uyumsuzlukların derecesi artacaktır. Gerçekte zorluk, mutlak assembly ile, makine-kodu amaç programının bellekteki yerleşiminin, programcı tarafından kaynak programda bir ORG sözde-işlemiyle tanımlanmasının gerekli olmasından kaynaklanmaktadır.

1.13 Yeri değiştirilebilir programlar

Bölüm 1.12'de tartışılan mutlak programlarla ilgili sorunlardan kaçınabilmek amacına yönelik çeviriciler vardır, bunlar programcının konuma bağımlı olmayan kaynak programlar yazabilmesine olanak tanırlar. Yeri değiştirilebilir programlarla, programcı, kaynak programda makine-kodu amaç programın sonuçta yerleştirileceği konumu tanımlamaz. Dolayısıyla ORG sözde-işlemi kullanılmaz. Çevirme işlemi, amaç program sanki 0000 konumundan başlayarak saklanacakmış gibi gerçekleşir. Programın, yürütülmesinden önce gerçekte nerede saklanacağı konusunun incelenmesi ilerki kısımlara bırakılmıştır.

Şekil 1.17'de Bölüm 1.12'de anlatılan OPCH yordamı için bir listeleme görülmektedir. Ancak Şekil 1.17'deki program, "yeri değiştirilebilir bir program" içindir. Herhangi bir ORG sözdeişlemi dahil edilmemiştir ve listeleme, amaç kodunu, 0000 konumundan başlayarak saklanacakmış gibi göstermektedir. Şekil 1.18'deki listeleme OPMSG altyordamı içindir ve kısmen yeri değiştirilebilir biçimdedir. Çalıştırılabilir komut kodunun yeri değiştirilebilir, fakat veri saklama bölgesi olan VERI, program sonundaki ORG ve DS sözde-işlemleri yardımıyla 2000 şeklinde mutlak olarak tanımlanmıştır. Yeri değiştirilebilir çeviriciler normal olarak yeri değiştirilebilir bir program içinde mutlak kod bölümlerine izin verirler.

Yeri değiştirilebilir bir programın çevrilmesinde, komutların sembolik adresleri programın ilk komutunun adresine bağlı olarak tanımlanır. Bu nedenle, Şekil 1.18'deki OPMSG listelemesinde SONRKAR etiketli satırdan çevrilen komutun adresi 0005 olarak belirlenmiştir. Mutlak adres bu olmayacaktır fakat basitçe programın çalıştırılabilir ilk komutu olan MVI B,10'un koduyla arasındaki mesafeyi göstermektedir.

S1.9

1.14 Yerleştiriciler ve yeniden yerleştirme yükleyicileri

Çevrilmiş bir yeniden yerleştirilebilir programın çalıştırılabilmesi için, önce, program makine kodu yükleyici programın bir biçiminde belleğe yüklenmiş olmalıdır. Bu noktada, program kodunun bellekte nerede bulunacağına ilişkin karar verilmesi gerekir. Bu belirlendikten sonra program kodundaki konuma bağlı tüm işlenenler uygun şekilde değiştirilmelidir.

Şekil 1.17'deki OPCH yordamının konumunu 5000 olarak değiştirmek istediğimizi varsayalım. Bu durumda JZ OPCH komutunun CA0000 olan kodunu CA0050 olarak değiştirmek söz konusu olacaktır, çünkü koşullu atlamanın varış adresi artık 5000'dir. Bu da, tabii ki, programın çalıştırılabilir ilk komutu olan IN komutunun ilk baytının adresidir. Yeri değiştirilebilir bir programın kodunu belli bir konumda mutlak kod oluşturmak için değiştiren programlar bulunmaktadır.

Bu programlara, yerleştiriciler adı verilir. Yeniden yerleştirici yükleyiciler aynı işlevi görürler, fakat buna ek olarak yeniden yerleştirilmiş kod, bellekte tanımlanan yere yüklenir.

S1.10

1.15 Bağlayıcılar ve bağlama yükleyicileri

Bölüm 1.12'de iki ya da daha çok program modülünü daha büyük tek bir program oluşturmak üzere birleştirmede ortaya çıkan sorunları tartıştık. Yeri değiştirilebilir biçimde program modülleri yazmak, sorunların bir çoğunu giderecektir. Bazı

```
                                                          PAGE  1
----- 8080/85 ASSEMBLER V01 -------------------------  FUTUREDATA ---------

                                   OPCH
            * ***************************************** *
            *  BU ALTYORDAM X'7D' PORTUNA BİR BAYT YERLEŞTİRİR DURUM SÖZCÜĞÜ  *
            *  X'7E' PORTUNDAN OKUNUR BİRİM HAZIR OLMADIĞI ZAMAN DURUM = 0 DIR *
            *                                                                 *
            *                                YAPI                             *
            *                                ****                             *
            *  DO UNTIL DURUM = READY                                         *
            *          DURUM SÖZCÜĞÜNÜ GİR                                    *
            *  END DO                                                         *
            *  VERİ BAYTININ ÇIKIŞINI YAP                                     *
            * ***************************************** *

0000  DB7E    OPCH      IN      X, '7E'      DURUM SÖZCÜĞÜNÜ AL
0002  CA0000            JZ      OPCH
0005  7E                MOV     A, M         VERİ BAYTINI AL
0006  D37D              OUT     X '7D'       VERİ BAYTININ ÇIKIŞINI YAP
0008  C9                RET
0009                    END
```

Şekil 1.17 Yeri değiştirilebilir altyordam

çeviriciler program modüllerinin bağlanması için ek kolaylıklar sağlayan özel sözde-işlemler de tanımlarlar. Şekil 1.18 de görülen GLBL anımsatıcısı bunlara bir örnektir.

GLBL sözde-işlemi bir program modülünde, o modül içinde doğrudan tanımı yapılmayan sembolik adreslere referans verilmesine izin verir. Sembolik adres, ilk modülün daha sonra bağlanacağı tamamen ayrı bir başka program modülünde tanımlanabilir. Bir program modülünde tanımlanan ve başka modüllerde görülen sembolik adreslerin global semboller oldukları söylenir. Şekil 1.18'de yer alan OPMSG listelemesinde GLBL OPCH satırı OPCH'yi bir global sembol olarak adlandırmaktadır, bu, OPCH'nin bir başka program modülünde tanımlandığı anlamına gelir. Böylece programcıya, OPCH normal şekilde OPMSG içinde tanımlanmamış olsa bile, kaynak program modülüne, CALL OPCH gibi komutları ekleyebilme olanağı vermiş olur. OPCH'nin, bir global sembol olduğu belirtilmemiş olsaydı, CALL OPCH komutu assembly listelemesi içinde tanımlanmamış bir değişken olarak gösterilecekti. Şekil 1.19'da, tam olarak bu durum gösterilmektedir.

```
- - - - 8080/85 ASSEMBLER V01 - - - - - - - - - - - - - - - - - - - - - - - - - - PAGE 1
                                                                  FUTUREDATA - - - - - - - - -

                                        OPMSG
                 * ****************************************** *
                 *                                            *
                 * BU ALTYORDAM X'7D' PORTU ARACILIĞIYLA ON VERİ BAYTI UZUNLUĞUNDA *
                 * BİR MESAJI GETİRİR VERİ BELLEKTE X'2000' KONUMUNDAN BAŞLAYARAK *
                 * SAKLANMAKTADIR                             *
                 *                                            *
                 *                        YAPI                *
                 *                        ****                *
                 * DONGU=10                                   *
                 * SONRADRES = X '2000'                       *
                 * DO UNTIL DONGU = 0                         *
                 *          SONRADRES'TEN BAYT ÇIKIŞI YAP     *
                 *          SONRADRES = SONRADRES + 1         *
                 *          DONGU = DONGU -1                  *
                 * END DO                                     *
                 *                                            *
                   ******************************************

                          GLBL        OPCH                    DIŞ REFERANS
                 *
0000  060A       OPMSG    MVI         B, 10                   DONGU = 10
0002  210020              LXI         H, VERI                 İŞARETÇİYİ KUR
0005  CD0000     SONRKAR  CALL        OPCH                    OPCH'Yİ ÇAĞIR
0008  23                  INX         H                       İŞARETÇİYİ ARTTIR
0009  05                  DCR         B                       DONGU = DONGU - 1
000A  C20500              JNZ         SONRKAR
000D  C9                  RET
                 *
00OE                      ORG         X '2000'
2000             VERI     DS          10
200A                      END
```

Şekil 1.18 Global simge içeren yeri değiştirilebilir altyordam

Global semboller sıklıkla, kullanıldıkları program modülü içinde doğrudan tanımlanmadıkları için bu sembolleri içeren işlenenler çevirici tarafından oluşturulamazlar. Şekil 1.18'de, çeviricinin, çevirme işlemi sırasında sembolik olarak OPCH şeklinde referans yapıldığı adresin mutlak ya da bağıl değerini hiç bir şekilde bilemeyeceği için, CALL OPCH komutu işlenenini 0000 olarak çevirdiği görülmektedir. Bu bilgi, OPCH modülünün yükleneceği konum belli oluncaya kadar mevcut olmayacaktır. Bölüm 1.14'te anlatılan yerleştiriciler ve yer değiştirici yükleyiciler bu sorunun çözümü için uygun değildir.

Bağlayıcı olarak bilinen programlar, program modüllerinin birbirine bağlanması için gerekli işlemleri gerçekleştirirler. Bu yolla, çalışabilmesi için bağlayıcıyla bağlanmış bulunan grubun ilk modülünün saklanacağı adres verilmiş olmalıdır. Bağlayıcı, bağlanacak modülleri tarar, global sembollerin mutlak değerlerini saptar ve bu değerleri, global sembollerden yararlanan komutların işlenen kodu olarak ekler.

1.16 Çeviriciler için çalışma uygulamaları

Düşük maliyetli manyetik disketler yaygın şekilde kullanıma girinceye kadar assembly dilinde program geliştirme çalışmaları çoğunlukla, ikincil saklama ortamı olarak delikli kağıt bantların kullanımıyla gerçekleştirilmekteydi. Bu yöntemde, düzenleme, çevirme, bağlama ve yükleme işlemleri çok yavaştı. Bu, esasen işlemin her aşamasında kağıt bandın delinmesi ve okunması için harcanan zamandan kaynaklanmaktaydı. Her ne kadar günümüzde delinmiş kağıt bant çok seyrek kullanılıyorsa da, assembly dilinde program geliştirilmesine, yedek saklama ortamı olarak ses kaseti kullanımına destek veren bir kaç sistem bulunabilir. Profesyonel assembly programcılarının büyük çoğunluğu disktabanlı bir bilgisayar sistemi kullanabilir durumdadır. Bu bölümde yedek saklama alanının sabit disk ya da disket olduğu varsayılmıştır.

Manyetik disk-tabanlı saklama ortamı içeren bir sistemin özelliklerinden biri de normal olarak bir disk-işletim sistemine (DOS) sahip olmasıdır. DOS, programların ve verinin RAM ile disk arasındaki aktarımını düzenleyen bir programdır. Kullanıcı, sistem konsolunun klavyesi aracılığıyla komutlar girerek DOS ile iletişim kurar. Kullanıcı açısından DOS'un en önemli özelliği, veri ve programların diskte isimlerle tanımlanan dosyalar içinde saklanıyor olmasıdır. Bu bölümde klasik Disk İşletim Sistemlerinden birinin kullanıldığı varsayılmıştır.

Assembly dili programlaması çalışmasının ilk aşaması, bir düzenleyici program kullanarak kaynak programın hazırlanmasıdır. Düzenleyici, VDT konsol klavyesi

```
                                                                    PAGE  1
----- 8080/85 ASSEMBLER V01 ------------------------------------ FUTUREDATA ---------

                                          OPMSG
                  * ***************************************** *
                  *                                           *
                  * BU ALTYORDAM X'7D' PORTU ARACILIĞIYLA ON VERİ BAYTI UZUNLUĞUNDA *
                  * BİR MESAJ ÇIKARIR VERİ BELLEKTE X'2000' KONUMUNDAN BAŞLAYARAK  *
                  * SAKLANMAKTADIR                                                  *
                  *                                YAPI                             *
                  *                                ****                             *
                  *                                                                 *
                  * DONGU=10                                                        *
                  * SONRADRES = X '2000'                                            *
                  * DO UNTIL DONGU = 0                                              *
                  *          SONRADRES'TEN BAYT ÇIKIŞI YAP                          *
                  *          SONRADRES = SONRADRES + 1                              *
                  *          DONGU = DONGU -1                                       *
                  * END DO                                                          *
                  * ***************************************** *

0000  060A      OPMSG     MVI     B, 10          DONGU = 10
0002  210020              LXI     H, VERI        İŞARETÇİYİ KUR
0005  CD0000    SONRKAR   CALL    OPCH           OPCH'Yİ ÇAĞIR
** UNDEFINED SYMBOL**
0008  23                  INX     H              İŞARETÇİYİ ARTTIR
0009  05                  DCR     B              DONGU = DONGU - 1
000A  C20500              JNZ     SONRKAR
000D  C9                  RET
                  *
00OE                      ORG     X '2000'
2000              VERI    DS      10
200A                      END
```

Şekil 1.19 Global simgelerin belirtilmemesininin sonucu olarak tanımlanmamış simge.

aracılığıyla girilen kaynak program metni üzerinde kolayca işlem yapmaya olanak verir. Kaynak program istenen biçime geldiğinde disk üzerinde kaynak dosyası olarak saklanabilir. Düzenleyici işlemlerinin ayrıntıları sistemden sisteme oldukça farklılıklar gösterir.

Kaynak program dosyası hazırlandıktan sonra çevirici, bu dosya üzerinde işlem yaparak daha sonra diskte amaç dosya olarak saklanacak olan amaç programı oluşturur. Çevirici, disk üzerinde, genelde tek bir komut aracılığıyla RAM'e aktarılarak yürütülebilir şekilde saklanır. Bu komut içinde kaynak dosyanın adının bulunması gerekebilir. Buna ilişkin tipik bir DOS komutu aşağıdadır:

ASM DOSYA

Burada DOSYA ifadesi, programcı tarafından kaynak programı içeren dosyaya verilen addır. Böyle bir komut, oturmuş bir sistem olan CP/M işletim sistemi tarafından kullanılmaktadır. Bazı çeviricilerin benimsediği diğer bir komut ise basitçe çeviriciyi çağırır. Çevirici, daha sonra kaynak dosyanın adını soran bir

mesajı konsola gönderir. Kullanıcı, klavye aracılığıyla dosya adını girer. Futuredata evrensel mikroişlemci geliştirme sistemi bu şekilde çalışmaktadır.

Çevirme işlemi başlamadan önce kullanıcı belli bazı seçenekler arasından seçim yapabilir. Bu seçeneklerden birisi assembly listesini, konsolde görüntülenmesine ek olarak bir yazıcıya yönlendirebilme seçeneğidir. Bir başka seçenek ise assembly listesini, hata içeren satırların dışındaki satırları içerecek şekilde bastırmaktır. Bu, uzun bir kaynak programdan sözdizimi hatalarının ayıklanması aşamasında özellikle yararlıdır, çünkü böylece her bir çevirme işlemi sırasında alınan basımın süresi kısalır. Yine, assembly uygulamalarının ayrıntıları ve kullanıcı tarafından tanımlanma şekilleri, kullanılmakta olan çeviriciye bağlıdır.

Çevirici amaç programı diske sakladıktan sonra yapılacak olan işlemler, kullanılan çevirici tipine bağlı olacaktır. Basit bir mutlak çevirici ile makine kodu amaç programını, DOS LOAD komutu kullanarak RAM'e aktarmak mümkün olabilir. Ardından amaç program çalıştırılabilir. Eğer program yeri değiştirilebilir özellikte ise, bir yerleştirici ya da yeniden adresleme yükleyicisi kullanılmalıdır. Yerleştirici veya bağlama yükleyicisi çağrıldığında kullanıcı, amaç programın RAM'de saklanacağı adresi girecektir.

Program modüllerinin bağlanabilmesi olanağı tanıyan çeviriciler, çalışmalarının ayrıntıları bakımından bir çok farklılıklar gösterirler. Belki de en yaygın olanlar amaç program modüllerinin bağlanmasını kolaylaştıran çeviricilerdir. Kaynak program modülleri ayrı ayrı amaç dosyalara çevrilirler. Amaç dosyaların içerikleri, daha sonra, tek bir amaç kod yükleme dosyası yaratan bir bağlayıcı program kullanılarak bağlanırlar. Bir yeniden adresleme yükleyicisi daha sonra yükleme dosyasının içeriğini RAM'e transfer eder. Kaynak kod modüllerinin bağlanmasının
S1.12 mümkün olduğu alternatif yöntemler de bulunmaktadır. Daha sonra, yeniden adresleme yükleyicisi kullanılarak birleşik kaynak programa çevrilir.

Kaynakça ve bibliyografi

1.1 L.A. Leventhal, 6800 Assembly Language Programming, Osborne and Associates Inc, USA (1978).

1.2 A.R. Miller, 8080/Z80 Assembly Language, John Wiley (1981).

Sorular

1.1 Bir kaynak program, çevrilmeden önceki haliyle RAM'de hangi biçimde saklanır ?

1.2 Bir çeviricinin temel işlevi nedir ?

1.3 Kaynak program metni oluşturmada programcıya yardımcı olan bir program adı söyleyiniz.

1.4 Şekil 1.4'te gösterilen listelemede çeşitli bilgi tiplerini adlarıyla tanımlayınız.

1.5 Bir assembly dili programda sözde-işlemlerin genel işlevlerini belirtiniz.

1.6 Sözde-işlemlerin çalıştırılabilir komutlardan ne gibi farklılıklarının bulunduğunu açıklayınız.

1.7 Şekil 1.18'deki sözde-işlemleri ne olduklarını belirtiniz.

1.8 Bir mutlak programın ilk komutunun adresini tanımlayan bir sözdeişleme ilişkin anımsatıcıyı söyleyiniz.

1.9 Mutlak ve yeri değiştirilebilir assembly dili programların arasındaki temel farklılığı söyleyiniz.

1.10 Bir yerleştirici/yer değiştiricinin işlevini tanımlayınız.

1.11 Bir bağlayıcının işlevini tanımlayınız.

1.12 Şekil 8.12, 8.13 veya 8.14'te gösterilen programlardan birini, giriş yapabildiğiniz bir çevirici ile uyumlu olacak şekilde değiştiriniz. Kaynak metni hazırlamak için bir düzenleyici kullanınız. Programı çevirin, bağlayın ve belleğe yükleyin. Programı sistem monitör/hata ayıklayıcısı yardımıyla çalıştırınız.

Bölüm 2 Mikroişlemci Seçimi

Bu konunun amaçları: *Bu konuyu bitirdiğinizde;*

1 *Üretici veri sayfalarının yardımıyla bir mikroişlemcinin aşağıdaki önemli fiziksel özelliklerinden bazılarını analiz edebilmeli,*
   (a) *Sözcük uzunluğu/veri yolu uzunluğu*
   (b) *Tipik komutların işlem hızları*
   (c) *Komut takımı büyüklüğü ve olanakları*
   (d) *Adresleme modlarının çok yönlülüğü*
   (e) *Adres yolunun uzunluğu*
2 *Farklı mikroişlemcileri karşılaştırmak için yapılan değerlendirme testlerinin kullanımını açıklayabilmesiniz.*

2.1 Bir mikroişlemciyi nasıl seçeriz ?

Bu konunun genel amacı okuyucuya, belli bir uygulamada kullanmak üzere bir mikroişlemci seçerken önemli olan fiziksel özelliklerin bir bölümüne ilişkin fikir vermektir. Bu bölümün yazıldığı tarihte ondan fazla üretici, ellinin üzerinde farklı mikroişlemci üretmektedir. Diğer üreticiler, daha tanınmış cihazlar için "ikinci kaynak" durumundadır. Günümüzde, çok fazla sayıda farklı mikroişlemcinin mevcut olması, tasarım mühendisinin her projenin başlangıcında ilk olarak oturup, teknik esaslar çerçevesinde belli bir uygulama için en uygun cihazı seçmesi gerektiği sonucunu ortaya çıkarmaktadır. Aslında bu, sık görülen bir durum değildir.

Mikroişlemci seçimi genellikle, tamamı teknik olmayan etkenler tarafından belirlenir. Bu etkenlerin en başta gelenlerinden biri tasarımcının ya da tasarım ekibinin daha önceki deneyimidir. Eğer bu deneyim ağırlıklı olarak M6800 cihaz ailesi ile ilişkili ise o zaman aynı seri cihazları kullanmayı sürdürme eğilimi söz konusu olacaktır. Bu, anlaşılır bir yaklaşımdır çünkü değişikliği gerekli kılan nedenler çok güçlü değilse, geçmiş deneyimin geliştirilmesi ve yabancı olunan bir mikroişlemcinin kendine özgü niteliklerinin yol açabileceği sorunların önlenmesinin daha doğru olacağı açıktır. Sonuç olarak deneyimli bir tasarım ekibi, daha uygun görünen başka cihazların bulunması durumunda bile, belli bir mikroişlemcide ısrar edebilir. Tasarımcının veya tasarım ekibinin çalışmakta olduğu firma ya da kuruluşun geçmişteki deneyimi de, mikroişlemci seçimini etkileyecektir. Donanım ve yazılım

tasarımına yardımcı olan geliştirme sistemleri Bölüm 7'de tartışılmıştır. Bunlar pahalı cihazlardır ve çoğunlukla yalnız bir mikroişlemci tipine destek verirler. 8080 mikroişlemcisini destekleyen cihazlara yoğun şekilde yatırım yapmış olan bir şirketin, M6800 kullanmayı arzu eden bir mühendise, gerekçeleri güçlü olmadıkça sıcak bakması pek de olası değildir.

Bununla birlikte, mühendis için, bir mikroişlemcinin belli bir uygulamadaki avantaj ve dezavantajlarını kavramanın önemli olduğunu kabul etmek gerekir. Bu, en azından, belli bir mikroişlemcinin açıkça uygun olmadığı bir uygulamada kullanılmasını önleyecektir. Ancak her şeyin ötesinde büyük çoğunlukla, donanım ve yazılım tasarımcılarının beceri ve deneyimleri, mikroişlemci seçiminden çok daha önemli bir unsurdur. Başka bir deyişle, bir projenin başarılı ya da başarısız olmasını en çok etkileyen, çoğunlukla, mikroişlemci seçiminden çok mikroişlemcinin kullanım biçimidir.

2.2 İşlem gücü

Bölüm 2.1'de gördüğümüz gibi, bir sistemde kullanılan mikroişlemcinin seçimi hem teknik hem de teknik-dışı olmak üzere bir çok faktörün etkisi altındadır. Fakat bu seçimde temel bir gereklilikle karşılaşmış bulunuyoruz: sistem, istenen işlemleri, verilen süre içerisinde gerçekleştirmek için yeterli işlem gücüne sahip olmalıdır. Mühendisin, belli bir mikroişlemcinin gerekli bilgi işleme gücüne sahip olup olmadığını değerlendirebilecek durumda olmasının önemli olduğu açıktır. Ne yazık ki işlem gücü, örneğin elektrik gücü gibi çok net bir şekilde belirlenebilecek bir özellik değildir.

Ancak farklı mikroişlemcilerin işlem güçleri arasında anlamlı bazı karşılaştırmalar yapmak mümkündür. Örneğin iki farklı mikroişlemcinin performansını, bellekte, bir alandan diğerine 100 baytlık veri aktaran bir programın yürütülmesi için harcadıkları süreyi bularak karşılaştırabiliriz. Eğer mikroişlemcilerden biri, söz konusu işi diğerinden iki kat daha hızlı yapıyorsa, bu durum, tasarlanan uygulama bellekte sık sık ve hızlı şekilde veri blokları taşıması gerektiriyorsa kuşkusuz önem taşımaktadır.

Okuyucu bu gibi tek tek karşılaştırmaları çok fazla göz önünde bulundurmaktan kaçınmalıdır. Bir işlem tipini çok verimli bir şekilde gerçekleştirebilen bir mikroişlemci, diğer durumlarda daha yetersiz kalabilir. Yukarıda tanımlanan, bellek içinde veri hareketini çok hızlı şekilde yapan işlemci aritmetik işlemler gerektiğinde çok yavaş olabilir. Bu konunun geri kalan bölümünün büyük kısmı, farklı mikroişlemcilerin işlem güçlerinin nasıl karşılaştırılacağı ile ilgilidir.

Her ne kadar bazı durumlarda işlem gücü sorunu, mikroişlemci seçiminde kritik bir faktör olabilmekteyse de diğer bazı durumlarda değildir. Bunun nedeni, mevcut geliştirme desteği gibi teknik-dışı faktörlerin, belirli bir uygulama için gerekenden çok daha fazla işlem gücüne sahip bir mikroişlemcinin seçimine neden olmasıdır. Bu türde "aşırılıklar", bilgi işlem uygulamaları oldukça yaygındır. Bundan bir kaç yıl önce gereğinden çok daha fazla bilgi işleme gücüne sahip bir işlemcinin kullanılması kötü bir tasarım olarak görülürdü. Geçmişte, işlem gücü maliyetle doğrudan ilişkiliydi. Her ne kadar işlem gücü ile maliyet arasındaki bağlantı hala mevcut olsa da mikroişlemciler için bu çok daha az düzeyde doğrudan bir ilişkidir. Özellikle merkezi işlem birimi, yani mikroişlemcinin kendisi, çoğu kez bir sistemin en düşük maliyetteki parçalarından biridir. Bu nedenle mühendisin temel sorunu, güç fazlalığı söz konusu ise bunu fazla önemsemeden yeterli işlem gücünün her zaman mevcut olduğunu garanti etmektir.

Tartışmamızın buraya kadar ki bölümünde bir sistemde yalnızca bir mikroişlemcinin kullanılmasından söz ettik. Bu da yine, merkezi işlem biriminin genel sistem maliyeti içinde önemli bir paya sahip olduğu zamanlardaki alışılagelmiş yaklaşımı yansıtmaktadır. Merkezi işlem biriminin şimdiki durumda sistemin en ucuz parçalarından biri olduğu anımsanırsa, bir sistemde daha yüksek işlem gücüne sahip bir cihaz yerine iki ya da daha çok mikroişlemci kullanılması son derece makul bir durum olarak karşımıza çıkar ve bu da genelde arzu edilir. Bu, mikroişlemcilerin düşük maliyetinin bilgisayar sistem tasarımına olan yaklaşımı nasıl değiştirdiğine ilişkin örneklerden biridir.

2.3 Bazı gerçek mikroişlemciler

Mikroişlemcileri karşılaştırırken önemli olan özellikleri göstermek amacıyla bazı gerçek cihazları örnek olarak alacağız. Bu bölümde bu cihazlar tanıtılmakta ve bunların mikrobilgisayar sistemleriyle ne şekilde bağlantılı oldukları anlatılmaktadır. Tanımlanan mikroişlemciler Tablo 2.1'de sıralanmıştır, bunlara ait veri sayfaları, Ek A-E'de bulunmaktadır.

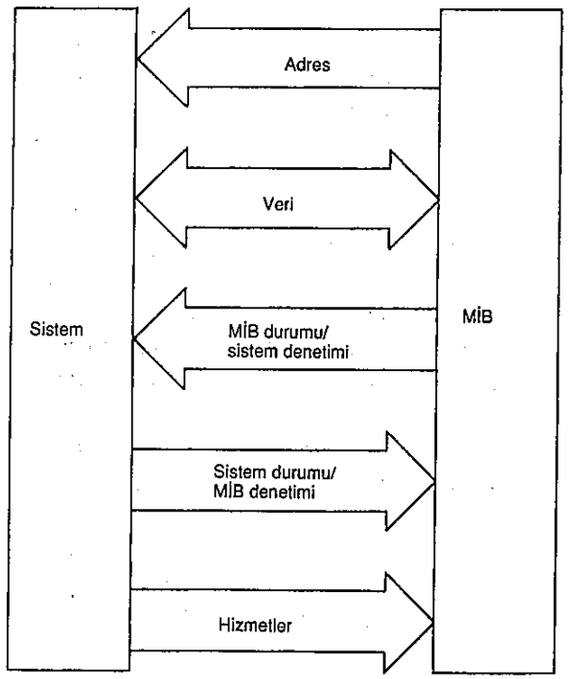
Bu mikroişlemcilerden COP420L, diğerlerinden daha farklı bir şekilde ele alınmalıdır. 8080, M6800, Z80, ve Z8000'i çok uygun şekilde merkezi işlem birimleri veya MİB'ler şeklinde tanımlamak mümkündür. Bu, bir bilgisayar sistemini bu mikroişlemcilerden biri ile birlikte çalıştırabilmek için, dış dünyayla iletişim kurmak üzere bellek, G/Ç arabirim devreleri ve çevresel cihazları eklememizin gerektiği anlamına gelmektedir. Ancak COP420L, biraz daha değişik tipte bir cihaza örnektir. Bellek ve G/Ç arabirim devreleri MİB ile birlikte yonga içine toplanmıştır. Bir bilgisayar sistemini bir COP420L ile birleştirmek için yalnızca bir güç kaynağı

ve gereksinim duyabileceğimiz çevresel cihazlar gereklidir. COP420L gibi cihaz-
S2.1 lara, tek-yongalı mikrodenetleyiciler ya da tek-yongalı mikrobilgisayarlar denir.

Tablo 2.1

Mikroişlemci	*Başlıca üreticisi*	*Sözcük uzunluğu*
COP420L	National	4
8080	Intel	8
M6800	Motorola	8
Z80	Zilog	8
Z8000	Zilog	16

Yalnızca MİB içeren mikroişlemciler esnek ve genişletilebilir özelliktedir. Genel amaçlı çok sayıda bilgi işlem işinde kullanılabilirler. Diğer yandan tek-yongalı mikrodenetleyicilerle birleşik sistemler daha sınırlı bir tanımı olan düzenlemeye sahiptirler. Tek-yongalı mikrodenetleyiciler normalde belli bazı değişmez işlerle özel amaçlı bilgi işlem görevlerinde kullanılır. Genel amaçlı ve özel amaçlı bilgisayarlar arasındaki farklılıklar Bölüm 5'te ayrıntılı olarak incelenmiştir.



Şekil 2.1 MİB ve sistem arasında bilgi alışverişi

Tek-yongalı mikrodenetleyiciler özel durumunu bir tarafa bırakırsak; bir mikrobilgisayar sisteminin en sık gereksinim duyduğu şeyleri tanımlamak mümkündür. MİB yongası ile diğer sistem bileşenleri, örneğin G/Ç devreleri arasındaki bilgi alışverişine olan gereksinim bunlara bir örnektir. Şekil 2.1'de MİB ile sistem arasındaki çeşitli bağlantılar, işlevsel açıdan birbirinden ayrılmıştır. Çeşitli MİB-sistem arabağlantı hatlarının kesin yapı ve işlevi bir mikroişlemciyi diğerlerinden ayıran özelliklerden biridir. Şekil 2.1'i, tanımladığımız mikroişlemcilerin özel niteliklerini göstermek için bir model olarak kullanacağız.

Veri yolu "*n*" sayıda paralel hattan oluşmaktadır, buradaki "*n*", mikroişlemci sözcük uzunluğuna eşittir. Bu yol, komut kodu ve hesaplanabilir işlenenleri sistem belleği ile mikroişlemci arasında taşır. Ayrıca G/Ç devreleri ile mikroişlemci arasında da veri aktarımını sağlar. Adres yolu, bellekte, bellek okuma ve yazma işlemleri sırasında verinin konumunu tanımlayan adresleri taşır. Adres yolu ayrıca, bir giriş veya çıkış işlemi sırasında belirli bir portu tanımlayan bir adresi de taşıyabilir.

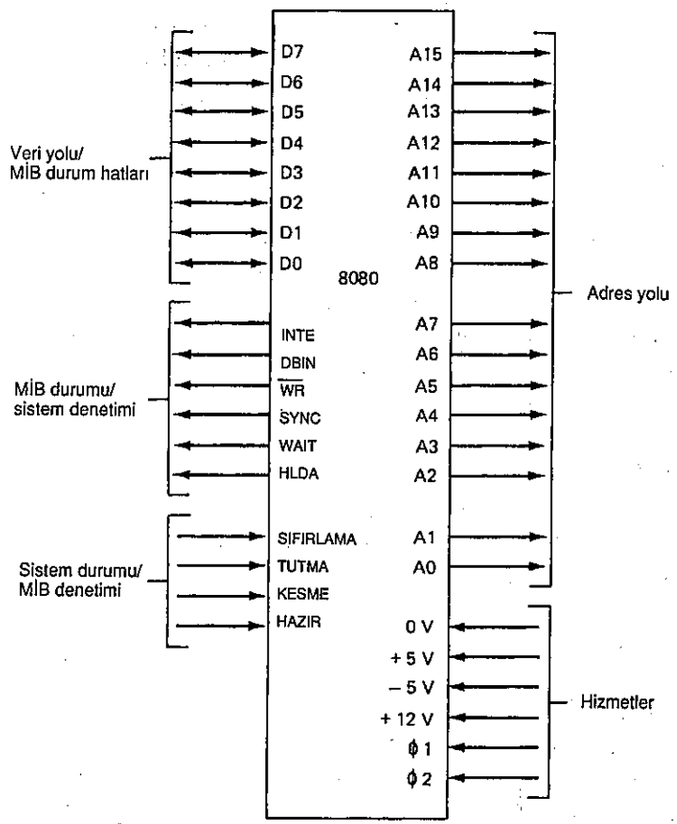
'MİB durumu/sistem denetimi' şeklinde etiketlenmiş hatlar mikroişlemcinin iç faaliyetine ilişkin bilgi sağlayan sinyalleri taşırlar. Bunun tipik bir örneği M6800 cihazının R/$\overline{W}$ hattıdır. Bu hat, M6800 bellek okuma işlemi yaparken yüksek, yazma işlemi söz konusu olduğunda ise alçak düzeydedir. Dolayısıyla, okuma ve yazma işlemlerinin gerçekleştirildiğini sisteme bildiren bir durum hattı olarak kabul edilebilir. Bunun yanında R/$\overline{W}$ hattına sistem açısından bakıldığında, hat ne zaman yüksekse belleğin, adres konumunun içeriğini adres yoluna yerleştirmesinin ne denli önemli olduğu açıkça anlaşılır. Bu gerçekleşmezse okuma işlemi başarılı olmayacaktır. Dolayısıyla sistem için R/$\overline{W}$, belli bir işlevi başlatan bir komut hattı veya denetim hattı gibi kabul edilebilir.

8080 Mikroişlemcisinin MİB durumu/sistem denetimi işlevini tamamlayan INT gibi hatları bulunmaktadır. Sistem açısından INT, mikroişlemciyi, örneğin yazıcı gibi çevresel bir cihazın herhangi bir iş için hazır olduğu yolunda bilgilendiren bir durum hattı gibi kabul edilebilir. Ancak mikroişlemci açısından INT, tartışmasız, normal program çalışmasını askıya alan ve denetimi, araya giren bir hizmet
S2.2 yordamına aktaran bir denetim hattıdır.

Son olarak Şekil 2.1'de 'Hizmetler' şeklinde belirtilen işlevler gelmektedir. Bu başlık altına, farklı mikroişlemciler arasında önemli ölçüde değişiklikler gösteren doğru akım (d.c.) besleme gereksinimlerini dahil ediyoruz. Bir çok mikroişlemci ayrıca, bir ya da daha çok sayıda dış alternatif akım (a.c.) saat dalgabiçim kaynaklarına da gerek duyarlar ve bunlar, hizmetler sınıfına girerler.

Şekil 2.1, MİB'i sistemin diğer bölümlerine bağlayan bütün hatların her birinin yalnızca bir işlevi olduğunu ve bu nedenle her bir hattın farklı olduğunu ortaya koymaktadır. Pratikte, bu her zaman böyle değildir. Bunun nedeni mikroişlemci tasarımcılarının, üretim maliyetlerini azaltacağı için mikroişlemci üzerindeki pin sayısını en az düzeyde tutmak istemelerinde yatmaktadır. Sonuç, istenen işlevleri sağlayacak sayıda pinin bulunmamasıdır. Bu sorunu aşmak için bazı sinyaller çoğullanmıştır. Bu, bazı hatların işlem saykılının farklı aşamaları boyunca farklı
S2.4 işlevleri gerçekleştirdikleri anlamına gelmektedir.

Sinyal çoğullanmasının tipik bir örneği, 8080 mikroişlemcisinin, ana işlevi veri yolu olan sekiz hattıdır. Her bir komut saykılının ilk iki saat aralığı süresince bir periyot için, bu hatlar bir veri baytı yerine sekiz MİB durumu/sistem denetimi sinyali taşırlar. Bu, her ne kadar 8080 paketinde fazladan sekiz pin kullanılmasını



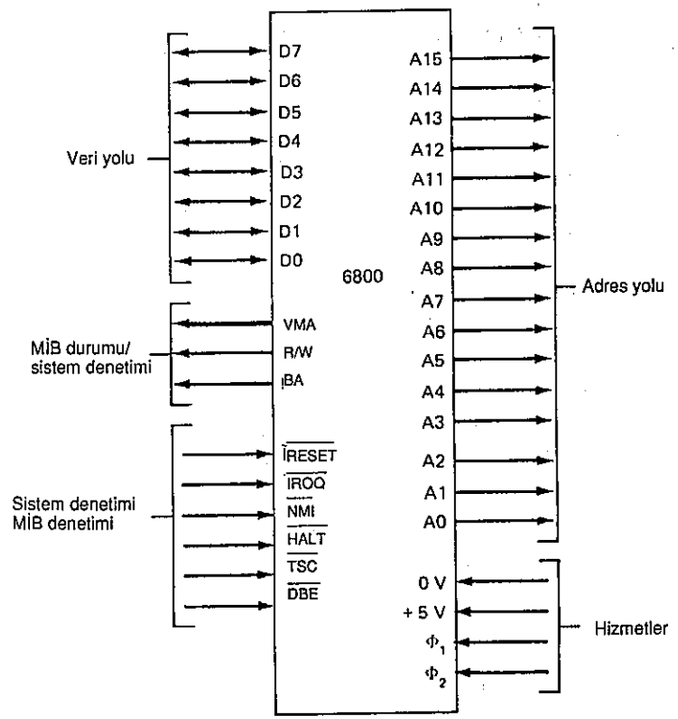
Şekil 2.2 8080 pinlerinin işlevleri

sağlarsa da, bariz donanım sorunları ortaya çıkar. Veri veya durum sinyallerini oluştukları anda 'yakalamak' için mikroişlemciye özel harici kaydediciler gerekir.

8080

8080, ilk 8-bitlik mikroişlemcilerden biri olmasına karşın bu kitabın yazıldığı dönemde en çok kullanılan mikroişlemcilerden biriydi. Şekil 2.2'de 8080'in pinleri, Ek A'da olduğu gibi doğrudan fiziksel yapısı şeklinde değil işlevsel gruplar olarak tanımlanmaktadır. İlk 8-bitlik mikroişlemcilerden biri olarak 8080, kullanıcı için daha sonra geliştirilen cihazlardan biraz daha az uygundur. Örneğin 8080'in en az üç doğru akım kaynağını gerektirir. Ayrıca 8080, iki fazlı bir saat sinyaline de gerek duyar.

Bu bölümde daha önce söylediğimiz gibi 8080 çoğullanmış bir veri yoluna sahiptir. Her işlem saykılı için D0-D7 hatları, mikroişlemci içindeki faaliyetlere ilişkin sistemin diğer bölümlerini bilgilendiren durum sinyallerini taşır. Mikroişlemci, bu



Şekil 2.3 M6800 pinlerinin işlevleri

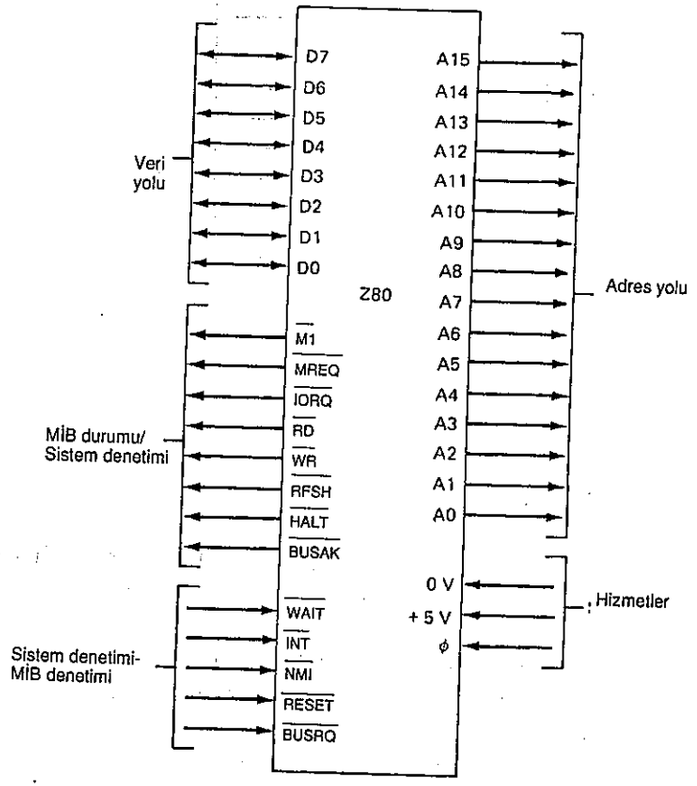
durum sinyallerinin D0-D7 hatlarında bulunduğu süre boyunca, SYNC hattını yüksek düzeyde tutar. D0-D7'de veri bulunduğu zaman DBIN hattı yüksektir. Bu iki sinyal dış devreler tarafından durum sinyalleri ve veri baytlarını yakalamak için kullanılabilirler. Özellikle bu işlevi gerçekleştirmek için özel amaçlı entegre devreler üretilmiştir.

S2.5

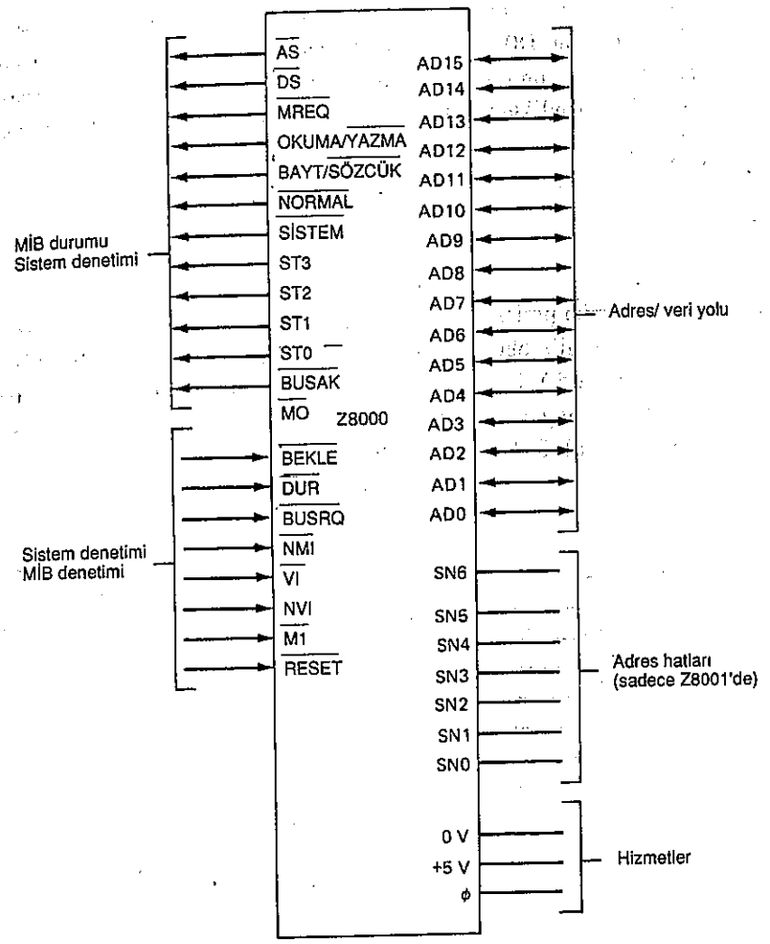
M6800

Şekil 2.3 M6800'ün pinleri üzerindeki sinyaller, işlevsel gruplar şeklinde tanımlanmaktadır. 8080 gibi 6800 de iki-fazlı bir saat sinyali gerektirir fakat 8080'den farklı olarak tek bir 5 V luk güç kaynağına gereksinim duyar. M6800'de herhangi bir pin üzerindeki sinyallerden hiç biri çoğullanmaz. Durum ve denetim sinyallerinin tam işlevleri Ek C'de anlatılmıştır.

4



Şekil 2.4 Z80 pin işlevleri



Şekil 2.5 Z8000 pin işlevleri

Z80

Her ne kadar Z80 ve 8080 mikroişlemcilerinin bir çok ortak komutu bulunmaktaysa da, Şekil 2.2 ve Şekil 2.4'ün karşılaştırılması iki cihazın bilgi işlem sisteminin geri kalan kısmına bağlanma şekillerinin çok farklı olması gerektiğini göstermektedir. Denetim ve durum hatlarının tam işlevleri Ek B'de anlatılmıştır. Z80 ve 8080 arasındaki ana donanım farklılığı, Z80'de, veri yolundaki sinyallerde hiç bir çoğullamanın oluşmamasıdır. Ayrıca 8080'den farklı olarak, yalnızca 5 V'luk tek bir güç kaynağının gerekli olduğu görülecektir. Hem 8080 hem de 6800 den farklı olarak Z80, yalnızca tek-fazlı bir saaat sinyaline gereksinim duyar.

Z8000

Z8000 başlığı tek bir mikroişlemciyi değil bir mikroişlemci ailesini tanımlamaktadır. Bu ailedeki mikroişlemciler burada ele alınan mikroişlemcilerin büyük farkla en güçlüleridir. Bu grubun üyelerinden ikisi Z8001 ve Z8002'dir. Şekil 2.5'te Z8001 ve Z8002 için pin bağlantıları işlevsel gruplar olarak görülmektedir. Z8002 daha geniş olarak Ek D'de anlatılmıştır.

Z8000 ailesine mensup mikroişlemciler esas olarak 16-bitlik verilerle işlem yaparlar. Bu, veri yolu uzunluğunun 16-hat olduğu anlamına gelir. Bu nedenle Z8000 paketindeki pin sayısını makul sınırlar içinde tutmak için tasarımcı, veri ve adres sinyallerinin çoğullanması yolunu seçmiştir. Dolayısıyla AD0-AD15 olarak adlandırılan hatlar işlemlerin bir döneminde adres, diğer bir döneminde ise veri taşır.

Z8001 ve Z8002 arasındaki temel farkın adres yolunun sayı ya da genişliğinde yattığı Şekil 2.5'ten de anlaşılmaktadır. Z8001, Z8002'ye oranla, fazladan yedi tane daha hatta sahiptir. Bu hatlar genişletilmiş adres yolunun en değerlikli kısmını oluşturmaktadır. Bu yüzden Z8002, 8080, 6800, ve Z80'de olduğu gibi, doğrudan 2^{16}=65.536 bellek konumunu adresleyebilir. Z8001, bu fazladan adres hatlarıyla 2^{23}=8.388.608 konuma erişim yapabilir. Belleğin büyük bir bölümüne erişim yapabilme özelliği, Bölüm 2.8'de anlatıldığı gibi bazı uygulamalar için çok önemli
S2.6 olabilir.

COP420L

Bu bölümde anlatılan diğer mikroişlemcilerden farklı olarak, COP420L, kullanıcıya, veri ya da adres yoluna erişime yapmaya izin vermez. Çünkü bellek ve G/Ç devreleri mikroişlemci yongası içinde bulunmaktadır. MİB-sistem bağlantıları işlevsel gruplara bölünmüş olduğundan, bu cihaz söz konusu olduğu sürece konumuzla ilgisi olmayacaktır.

Ek E'de COP420L'nin tek-yongalı mikrodenetleyiciler ailesinin bir üyesi olduğu görülmektedir. Bu ailenin üyeleri, bellek, G/Ç olanakları ve işlem hızı konularında farklılık gösterirler. COP420L'nin, 1K x 8 ROM ve 64K x 4 RAM içerdiği Ek E'de belirtilmiştir. ROM maske-programlı tiptedir yani program ve veriler, birimin üretimi sırasında yerleştirilmiştir. Bu özellikler 5. Bölüm'de daha geniş olarak tartışılmıştır.

2.4 Sözcük uzunluğu

Bir mikroişlemcinin sözcük uzunluğu, aritmetik, mantık, veri aktarımı ve G/Ç komutlarının yürütülmesi sırasında işlenen verinin bit sayısını tanımlar. Mikroişlemcinin içinde veri, çeşitli işlev birimleri arasında paralel bir veri yolu üzerinden aktarılır. Bu ana veri yolundaki paralel hat sayısı, mikroişlemcilerin çoğunluğunda sözcük uzunluğuna eşittir. Yine, G/Ç hatlarının sayısı ve iç çalışma kaydedicilerinin çoğunluğunun uzunluğu da normalde sözcük uzunluğuna eşittir.

Bu kitap yazıldığı sırada mevcut mikroişlemcilerin büyük bir kısmında sözcük uzunluğu 8 bitti. 32, 16, 12, 4 ve hatta 1bitlik sözcük uzunluğuna sahip mikroişlemciler de vardır. Özellikleri en anlaşılır şekilde ortaya konmuş olan mikroişlemci belki de Motorola M6800dır. Bu mikroişlemci A ve B olarak adlandırılan iki ana çalışma kaydedicisine sahiptir. Bunların her biri 8-bitlik veri saklar. Veriyi bellek ve çalışma kaydedicileri arasında taşıyan ana veri yolu 8-bit genişliğindedir, yani 8 hattan oluşur. Bu veri yolu aynı zamanda G/Ç işlemleri için de kullanılır. Bir M6800 sisteminde her bir bellek konumu 8-bitlik veri saklar.

6800 mikroişlemcisi klasik bir 8-bitlik makine olarak adlandırılır, bunun nedeni belki de, bir bilgisayar MİB'in iç yapısıyla ilgili alışılagelmiş fikirlere, diğer mikroişlemcilerin çoğunluğundan daha fazla uygun olduğu anlamına gelmektedir. Ancak, veri işleme fonksiyonlarının büyük çoğunluğunu 8 bit ölçüsüyle tanımlayan böylesine sade ve net bir mimari ve komut takımıyla bile, diğer veri parçaları mümkün olabilmektedir. 6800'ün iç yapısında bir kaç tane 16-bitlik kaydedici mevcuttur. Hepsi de programcının erişimine izin veren indeks, yığın işaretçisi ve program sayıcı kaydedicileri bunlar arasındadır. Dolayısıyla komut takımları içerisinde, bu kaydedicilerde saklanan verinin değişmesine neden olan komutlar vardır. Böylece bir 8-bitlik mikroişlemcide, 16-bitlik veri sözcükleri üzerinde işlem
S2.8 yapan, 16-bitlik kaydedicilerle bazı komutlar elde etmiş olmaktayız.

Eğer daha ayrıntılı incelersek komut takımının, her bir veri sözcüğünün bitleri üzerinde işlem yapan ve dolayısıyla bir-bitten oluşmuş veri birimlerini işleyen komutlar içerdiğini görürüz. Bunların anlatılmasındaki amaç, halen mevcut olan hiç bir mikroişlemcinin yalnızca bir sözcük uzunluğu ile çalışmadığını vurgulamaktır. Böyle bir cihazın mevcut olması pek olası değilse de, her halde mevcut olsaydı kullanılması çok zor olacaktı. Aslında eğilim ters yöndedir. En son olarak geliştirilen mikroişlemci aileleri farklı veri parçaları üzerinde işlem yapma yetenekleri nedeniyle ekstra esneklik ve bilgi işlem gücü kazanmaktadır.

Eğer M6800 mikroişlemcisi klasik bilgisayar mimarisinin bir örneği olarak kabul

edilirse, Z8002'i de yeni kuşak mikroişlemciler için bir örnek olarak alabiliriz. Ek D'de görüldüğü gibi Z8002, bir 16-bitlik mikroişlemci olarak adlandırılmaktadır; yani işlenecek veri birimlerinin çoğunluğu 16 bittir. Bu cihazın iç yapısında çeşitli biçimlerde tanımlanabilen birkaç genel amaçlı çalışma kaydedicisi bulunur. En kısa ve yaygın tanımlama şöyledir: Z8002, her biri 16-bitlik veri saklayabilen 16 adet genel-amaçlı çalışma kaydedicisi içerir. Bunlar R0-R15 olarak tanımlanırlar. Z8002'nin komut takımında bu kaydedicilerde bulunan 16-bitlik veri sözcüklerini taşıyan ve işleyen komutlar bulunur. Ayrıca verileri, 8-bitlik sözcükler biçiminde taşıyan ve işleyen komutlar da vardır. Bu komutlar için, R0-R7 adlı sekiz kaydediciden her biri, 8-bitlik iki kaydedici gibi ele alınırlar. Dolayısıyla programcının 8-bitlik 16 adet genel-amaçlı kaydediciye etkin bir şekilde erişim hakkı bulunmaktadır. Sözcük uzunluğu esnekliği bu kadarla kalmaz. 32bitlik sözcük formunda veriyi işleyen komutlar da mevcuttur. Bu komutlar için R0-R15 adlı onaltı kaydedici, programcıya 32-bitlik 8 adet genel-amaçlı kaydediciye etkin bir şekilde erişim hakkı vermek için, çiftler halinde birleştirilmiştir.

Daha yeni bazı mikroişlemcilerde esneklik o kadar gelişmiştir ki karakteristik bir sözcük uzunluğu tanımlamak zordur. Bunlardan ikisi Motorola 6809 ve Intel 8088'dir. Bunlar çoğu kez 8-bitlik cihazlar olarak adlandırılır ve veri bu mikroişlemcilerde tabii ki 8-bitlik birimler halinde alınır ve çıkarılır. Ancak her ikisinde de iç veri aktarımları ve veri işlemelerinin, 16-bitlik bir işlemcinin karakteristiği gibi değerlendirilmeleri daha doğru olabilir. Bu özelliğin M6809 ile ilişkili yönleri 2.6 nolu kaynakta daha ayrıntılı olarak incelenmiştir.

Sözcük uzunluğu tanımlamasının bazı inceliklerini anlattıktan sonra şimdi de bu parametrenin, bir mikroişlemcinin belli bir uygulama için ne denli uygun olduğuna ilişkin etkilerini ele alalım. İlk bakışta sözcük uzunluğu ve işlem gücü arasındaki ilişki açıktır. Eğer bir mikroişlemci, bir komutun yürütülmesi sırasında, verilen bir işlemi 8-bitlik bir işlenen üzerinde gerçekleştirebiliyorsa, veriyi, işlemi yalnızca 4-bitlik bir işlenen üzerinde yapabilen ve diğer bütün özellikleri eşit olan bir başka cihazdan iki kat daha hızlı işliyor demektir. Bir sistem tasarım çalışması sırasında işlem gücü yetersizliği kendini gösterdiğinde, olası çözümlerden birisi, daha büyük sözcük uzunluğu olan bir mikroişlemciye geçmektir. Fakat bu şekilde elde edilecek daha yüksek veri işleme hızı bazı uygulamalar için kullanışlı olmayabilir.

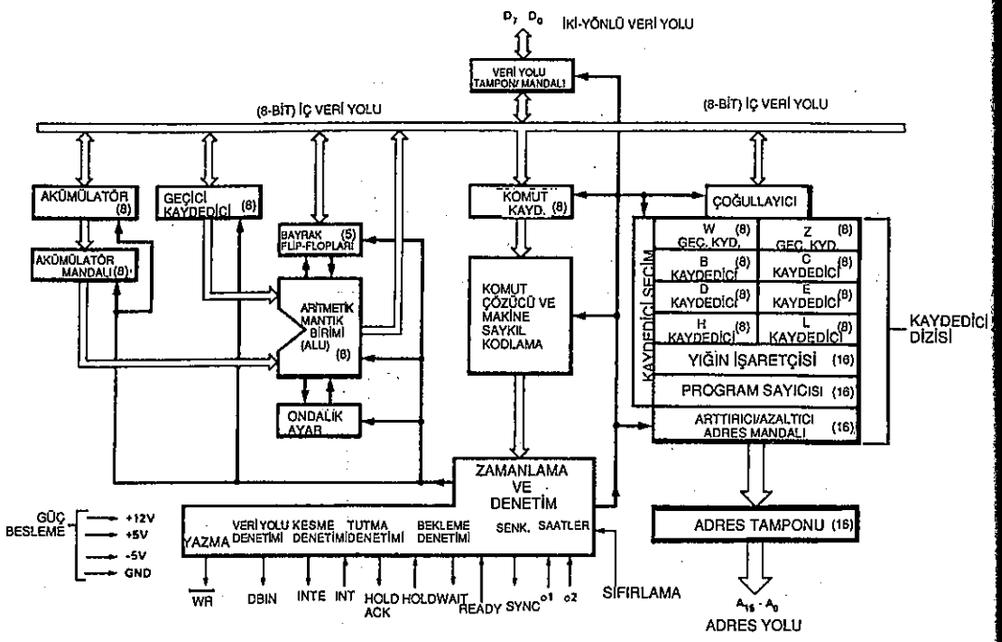
Bellekte saklanan bir 4-bitlik sayı listesindeki en büyük girişin bulunması işleminin yapılmasının sık sık gerektiği bir uygulamayı ele alalım. Piyasada yaygın olarak bulunan 4-bitlik mikroişlemcilerden herhangi bir tanesi, uygun bir programla birlikte bu işlevi yerine getirebilir. Fakat varsayalım ki piyasada mevcut olan 4-bitlik cihazlardan hiç birisi bu sıralama işlemini istenen kısa bir sürede gerçekleştiremiyor. Bu durumda, sistem tasarımcısının seçebileği çeşitli seçenekler bulunmaktadır. Bu seçeneklerden biri, 8-bitlik bir mikroişlemci kullanarak daha yüksek bir işlem gücü elde etmektir. 8-bitlik kapasiteden yararlanmak için iki 4-bitlik sayıyı 8 bit uzunlukta bir sözcüğe dönüştürmeliyiz. Her bir 4-bitlik sayıyı, 8-bitlik sözcüğün en düşük değerlikli yarısı olarak saklamak, mikroişlemcinin her defasında 4-bitlik veri işlediği anlamına gelir, bu da bize bir yarar sağlamaz. Burada beklenmedik bir güçlük, sayıların iki tanesinin 8-bitlik bir sözcük olarak toplanması durumunda, sıralama programının karmaşıklığının artmasıdır. Bu durum hemen hemen daima, bu şekilde kullanılan 8bitlik cihazın teorik avantajını ortadan kaldıracaktır. Aslında 8-bitlik bir mikroişlemci, bu tekniği kullanarak arama yapabilmek için, diğer tüm özellikleri eşdeğerde olan bir 4-bitlik cihazdan daha uzun süre harcayacaktır. Kısacası böyle bir durumda, 8-bitlik mikroişlemciden daha büyük işlem gücü elde edilmez.

Pratik hayatta karşılaşılan çoğu durumda, karşılaştırma yapmak bu kadar kolay değildir. Pratikte karmaşıklığı yaratan şey, 4-bitlik mikroişlemciden 8-bitlik mikroişlemciye geçerek çok daha güçlü bir komut takımına erişiyor olmamızdır. Bunun ne anlama geldiğini görmek için Eklerde açıklanan COP420L'nin komut takımıyla her hangi bir 8-bitlik mikroişlemcinin komut takımını karşılaştırınız. 8-bitlik cihazın daha güçlü olan komut takımı, işlem hızının artırılmasında, sözcük uzunluğu kadar, hatta ondan daha da önemli olabilir. Bu konu Bölüm 2.6'da daha ayrıntılı olarak anlatılmıştır.

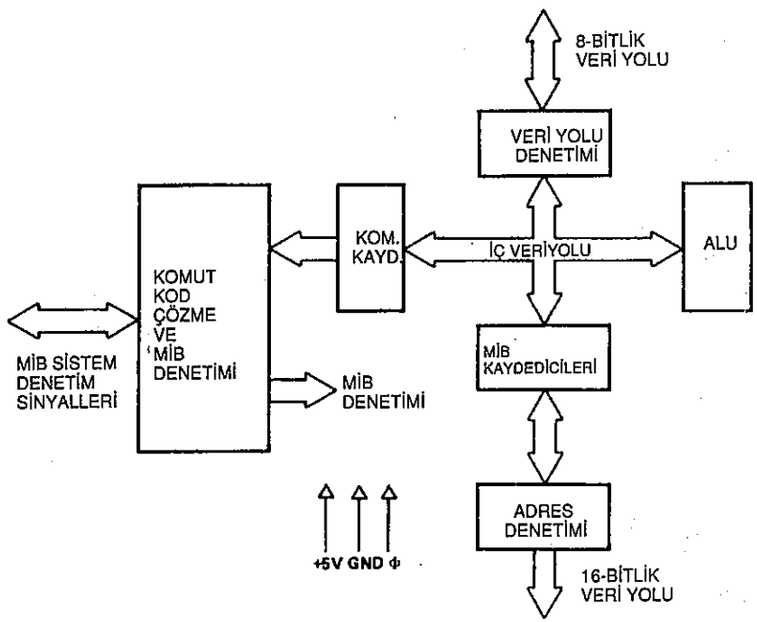
2.5 Mikroişlemci mimarisi

Bir mikroişlemcinin iç yapısından, ya da herhangi bir bilgisayarın merkezi işlem biriminden söz ederken genellikle mimari terimi kullanılır. Bir mikroişlemci olan entegre devre içinde bileşen birimlerin karmaşık bir topluluğu bulunur: kaydediciler, sayıcılar, aritmetik devreler, mantıksal devreler, bellek elemanları vb. Mimari tanımı içerisine, çoğu zaman yalnızca, programcı ve uygulama mühendisinin doğrudan ilgi alanına giren bileşenler dahil edilir. Bu bileşenler kaydediciler ve, içerik ya da işlevleri bir komutun yürütülmesi sonucu doğrudan değiştirilebilen diğer birimlerdir. Tanımı, bu şekilde basitleştirildiğinde bir mikroişlemciye, programcının modeli denir.

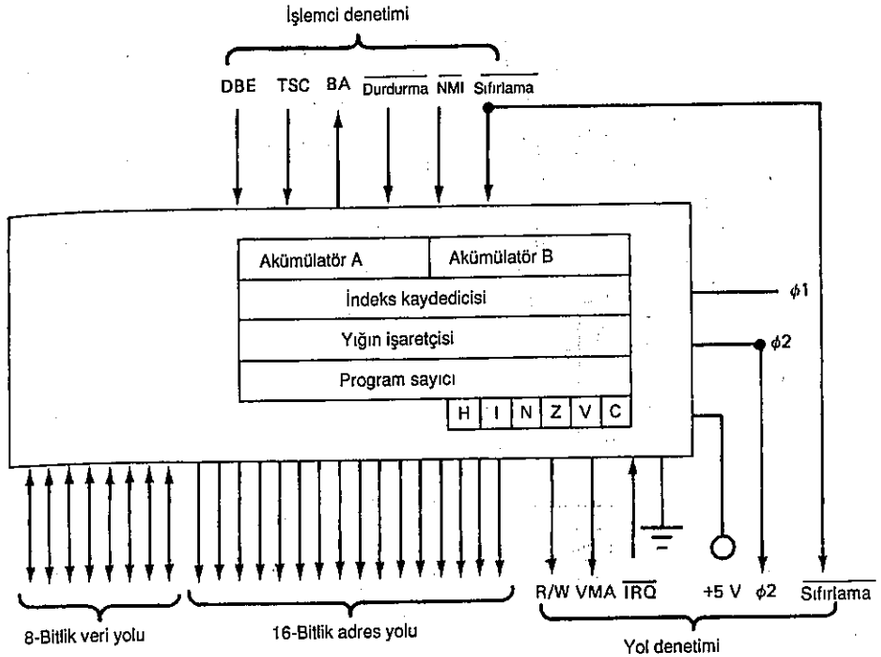
Üreticilerin veri sayfaları mikroişlemci mimarisinin tanımını içerirler. Ayrıntılara ne ölçüde girildiği, üreticiye bağlıdır. Şekil 2.6'da, 8080 mikroişlemcisinin mimarisini gösterilmektedir ve Intel veri sayfalarından alınmıştır. Şekil 2.7 ile 2.8'de ise,
S2.9 Z80 ve M6800 mikroişlemcileri için programcı modellerini göstermektedir ve
2.10 Zilog ve Motorola veri sayfalarından alınmışlardır.



Şekil 2.6 8080 MİB işlevsel blok diagramı



Şekil 2.7 Z80 MİB işlevsel blok diagramı



Şekil 2.8 M6800 Giriş/Çıkış hatları ve programcı modeli

2.6 Komut takımları

Genel

Farklı mikroişlemciler birbiriyle karşılaştırılırken en çok zorluk yaratan konu belki de bunların komut takımlarının karşılaştırılmasıdır. Bunun nedeni her mikroişlemci tipinin kendine özgü üstünlük ve zayıflıklarının bulunması, ve bunların bir başka cihazınkilerle doğrudan karşılaştırılmasının mümkün olamamasıdır. Ortak özellikler elbette olabilir. Tüm mikroişlemcilerde, çalışma kaydedicilerinden biri veya birkaçı ile bellek arasında veri aktarımına izin veren komutlar vardır. Aritmetik ve mantık işlemlerini gerçekleştiren komutlar da, G/Ç komutları gibi, bir başka belirgin ortak özelliktir. Ancak ileride göreceğimiz gibi, örneğin, görünüşte benzer olan aritmetik işlemlerin bile farklı mikroişlemci tiplerinde doğrudan karşılaştırılması mümkün olmayabilir.

Komut takımları konusunun kısaca incelenmesi bile bu konunun işlemci yapısından ayrı tutulamayacağını hemen gösterecektir. Bir mikroişlemcinin yapısı ve komut takımı, onu bir bütün olarak tanımlayan iki unsurdur, bu unsurlar eşit düzeyde

önemlidir ve birbirini tamamlayıcı niteliktedir. Komutlar, yalnızca, cihaz yapısı bir bölümünde, örneğin bir kaydedicide saklanan bilgilerin işlenmesiyle ilişk oldukları zaman tanımlanabilirler. Dolayısıyla 8080 komut takımındaki ADD komutu, sistem yapısı içinde A ve B kaydedicilerinin bulunduğunu gösterir çün komutun, A ve B kaydedicilerinin içeriklerinin toplanmasına ve sonucun A kayd dicisinde kalmasına neden olacağını belirtmektedir. ADD W gibi bir kom sözgelimi 8080 için anlamsız olacaktır, çünkü cihaz, W olarak adlandırılan kaydedici içermemektedir. Buradan çıkarılabilecek bir sonuç; bir çok kayded içeren karmaşık yapılı bir mikroişlemcinin, yalnızca bir ya da iki kaydedic bulunan basit bir cihazdan daha kapsamlı bir komut takımına sahip olmasın beklenebileceğidir. Daima olmasa da bu genellikle doğrudur, çünkü programcı kullanabileceği bir kaydedicinin bulunması, ilgili bir ya da daha çok komut bulunması anlamına gelir. Bir komut takımı büyüklüğünün mikroişlemci yapısı karmaşıklığıyla ne denli ilişkili olduğunu görmek için, Ek C ve D'de bulun M6800 ve Z8002 mikroişlemcilerinin komut takımlarını ve yapılarını inceleyin

Farklı mikroişlemcilerde belirli bir işlem grubuna ilişkin bulunan komutla sayısını karşılaştırırken, her üreticinin komutları farklı biçimde tanımlayabileceğ göz önünde bulundurmak gerekir. Örnek olarak, Mantıksal AND (VE) işlem gerçekleştirmek için bulunan komut sayısını alalım. Önce Ek C'de yer alan M680 komut takımına bakınız. Burada, işlemi yapmak için AND A ve AND B biçimin iki komutun bulunduğu açıkça görülmektedir. Diğer taraftan Ek A'daki 8080 kom takımı, aynı amaç için, üç komut sıralamaktadır: ANA r, ANA M ve ANI. Anc ANA r komutu, en az yedi komut için genel bir tanımlama olarak kabul edilebil çünkü komut kısaltmasındaki 'r', A, B, C, D, E, H veya L kaydedicilerind herhangi biri anlamına gelebilir. Bu komutların kısaltma biçimleri aşağıdadır:

```
ANA   A
ANA   B
ANA   C
ANA   D
ANA   E
ANA   H
ANA   L
```

Kaydedici- ve bellek-yönelimli mikroişlemciler

Bir mikroişlemcinin komut takımı, aynı zamanda o mikroişlemcinin tasarım felsefesinin esaslarını da ortaya koyar. Ek C'de yer alan M6800'ın komut takımı yeniden bakınız. Bu mikroişlemci çoğu kez bellek-yönelimli olarak tanımlan

çünkü iki işleneni olan mantıksal, aritmetik ve diğer işlemlerin çoğunluğu, bu işlenenlerden birinin bellekte saklanmasını gerektirir. Bunun tipik örnekleri anımsatıcı biçimindeki iki çıkarma komutudur:

```
SUBA   XXXX
SUBB   XXXX
```

Burada xxxx dört basamaklı onaltılı bir adrestir. Bu komutlar çalıştırıldıklarında, adreslenmiş bellek konumunun içeriğini duruma göre A veya B kaydedicisinden çıkarırlar. Örneğin aşağıdaki komut onaltılı 1000 adresinin içeriğini B kaydedicisi içeriğinden çıkarır ve sonucu B kaydedicisinde bırakır:

Komut kısaltması	*Makine kodu*
SUBB X '1000'	F0 10 00

Alternatif bir düşünce sistemi de 8080 ve Z80 mikroişlemci aileleri için uyarlanmıştır. Bu cihazlar kaydedici-yönelimli olarak kabul edilirler. İki işleneni bulunan komutların büyük bölümünde, bunların her ikisinin de çalışma kaydedicilerinde kalmaları gerekir. Bu nedenle M6800'in, SUBB komutunun doğrudan bir eşdeğeri yoktur. Çıkarmaya ilişkin 8080 komutlarının tümü, bir işlenenin A kaydedicisinde olmasını, diğerinin ise A, B, C, D, E, H veya L çalışma kaydedicilerinden birinde bulunmasını gerektirir. M6800'ün SUBA komutunun etkisini tam olarak gerçekleştirmek için bir kaç ayrı işleme gerek vardır, çünkü, çıkarma işleminin gerçekleşebilmesinden önce belirtilen bellek konumu içeriğinin B kayedicisine aktarılması gereklidir. Sonuç olarak aşağıdaki iki aşamalı işlem söz konusu olur:

1 Bellek konumunun içeriğini B'ye aktar.
2 B kaydedicisi içeriğini A kaydedicisinden çıkar.

Ancak, 8080, bellekten A kaydedicisine yalnızca tek-baytlık veri aktarımına izin verdiği için yeni bir sorun ortaya çıkar. Bu durumda, A kaydedicisi, başlangıçta işlenenlerden birini içerir. Aşağıdaki uygulama bu sorunu ortadan kaldırır:

1 A kaydedicisi içeriğini, C kaydedicisinde sakla
2 Bellek konumu içeriğini A'ya yükle
3 A kaydedicisi içeriğini, B kaydedicisine aktar
4 C kaydedicisi içeriğini, A kaydedicisine aktar
5 B kaydedicisi içeriğini, A kaydedicisinden çıkar

Bu işlemi gerçekleştiren bir program, Şekil 2.9'de gösterilmektedir.

```
MOV  C, A        A'yı C'ye kaydet
LD   A, 1000H    Sayıyı al
MOV  B, A        A'yı B'ye aktar
MOV  A, C        C'yi A'ya aktar
SUB  B           B'yi A'dan çıkar
```

Şekil 2.9 8080 çıkarma programı

Bu örnekten, 8080'de çıkarma işleminin, M6800'dan daha az kullanışlı olduğu sonucuna varabiliriz. Aslında çıkarılacak en doğru sonuç, 8080'i çok uygunsuz, verimsiz bir şekilde kullanmaya çalıştığımızdır. M6800'ün SUBA komutunu etkisini 8080 ile tam olarak kopyalamaya çalışmak yararsız bir çabadır. Bunu yerine programcı, 8080'in kaydedici-yönelimli mimarisini kullanan programlar yazmalıdır. Herhangi bir aritmetik veya mantıksal komutlar dizisi çalıştırılmadan önce, ilgili sayılar, çalışmakta olan A, B, C, D, E, H ve L kaydedicilerine aktarılmalıdır, bu sayılar burada, 8080 aritmetik veya mantıksal komutlarının işlenenler olarak kullanılmak üzere uygun şekilde yerleştirilirler. 8080'in kaydedici-yönelimli komut takımı, programcının erişebileceği yedi çalışma kaydedicisi içeren mimarisine yansımıştır; M6800'da iki kaydedici bulunur. İki işlemcideki kaydedicilerin işlevleri de bir çok yönden farklı olduğu için, bu karşılaştırmanın çok özenli şekilde
S2.11 yapılması gerekir.

2.7 Adresleme

Adresleme modları

Bir kaç istisna dışında, bir mikroişlemcinin veya herhangi bir bilgisayarın komutları aşağıdaki işlevleri tanımlar:

1 Yükleme, saklama, kaydırma ya da toplama gibi bir işlemi
2 Üzerinde işlem yapılacak veriyi

Bir komutun adresleme modu, verinin tanımlanma biçimini belirler. Geniş bir adresleme modu grubu, programcı için kapsamlı bir komut takımı kadar önemlidir. Bu bölümde mikroişlemci programcılarının kullanımı için en çok bulunan adresleme modları anlatılacaktır. Üreticilerin çoğu, çeşitli adresleme modları için ayrı isimleri kullanır. Ancak bazı farklılıklar bulunabilmektedir, bu farklılıkların bir kısmından burada söz edilecektir. Tam anlamıyla tanıdık olunmayan bir mikroişlemciyle karşılaşıldığında, en akıllıca yol, daima, çeşitli adresleme modlarının kesin tanımlaması için üretici veri sayfalarının incelenmesidir.

Örtük adresleme

Komutta, bir işlem kodu ya da işkodu (opkod) ile, veriyi veya verinin bulunduğu yeri tanımlayan adres olmak üzere iki kısmın bulunmasını gerektiren diğer adresleme modlarının aksine, örtük adreslemede, işlenen olarak kullanılacak olan verinin yeri, işlem kodunun içinde belirtilir. Örtük adresleme modu için tipik bir örnek, M6800 mikroişlemcisindeki ABA komutudur. ABA komutu çalıştırıldığında, A ve B kaydedicilerinin içerikleri toplanır ve işlemin sonucu A'da kalır. 8-bitlik mikroişlemcilerde örtük adreslemeli komutlar çoğu kez tek bitlik bir kodla tanımlanırlar çünkü işlenen(ler)i belirtmek için ek bilgiye gerek yoktur. 8-bitlik mikroişlemcilerin bazı komutları bu adresleme modundadır. Aşağıda bazı örnekler verilmiştir:

1 8080 XCHG; kaydedici çiftleri D,E ve H,L'nin içeriklerini karşılıklı değiştir anlamındadır.
2 Z80 LD SP,IY; yığın işaretçisi SP kaydedicisini, IY kaydedicisi içeriği ile yükle anlamına gelir.

16-bitlik mikroişlemcilerde örtük adreslemenin tanımı bir parça farklılık gösterebilir. Bunun temel nedeni, bu cihazların 8-bitlik mikroişlemcilere göre çok daha karmaşık bir mimariye sahip olmalarıdır, bunlardaki kaydedici sayısı 8-bitlik mikroişlemcilerden çok daha fazladır. Şekil 2.7 ve Ek D'de bulunan Z80 ve Z8000 programcı modellerini birbiriyle karşılaştırınız. Z8000, genel amaçlı onaltı kaydedicisi sahiptir, buna karşılık Z80'de yedi kaydedici vardır. Ayrıca Z8000'in komut takımının düzgün olduğu söylenir. Bu, diğer özelliklerine ek olarak, belli bir kaydedicinin içeriğini işlenen olarak kullanan bir komutun, başka herhangi bir kaydediciyle de çalışabileceği anlamına gelir. NEG R3 anımsatıcısı ile tanımlanan ve 3 kaydedicisi içeriğinin ikiye tümleyeninin negatif değeriyle değiştiren bir komut bulunmaktadır. İşkodunun, verinin bulunduğu konumu belirtmesi ölçüsünü kullanarak, bunun bir örtük adresleme örneği olduğunu söylemek mümkündür. Ancak NEG R3 komutu NEG xx şeklindeki onaltı komuttan oluşan grubun yalnızca bir tanesidir, buradaki xx, onaltı çalışma kaydedicisinden (R0-R15) herhangi birini belirtebilir. Bu nedenle örtük adresleme terimi bu durum için kaydedici adreslemesi şeklinde değiştirilebilir. Kaydedici adreslemesi terimini bazı yazarlar, aynı zamanda, Z80 ve 8080 mikroişlemci ailelerinde, örtük adresleme yerine de kullanmaktadır. Z8000 üretici veri sayfalarında kullanılan örtük adresleme terimi, burada yapılan tanımlamadan ve diğer üreticilerin tanımlamalarından tamamen farklı şekilde açıklanmaktadır, bu durum karmaşıklığı arttırmaktadır. 2.2 no'lu kaynağın 3-32 sayfalarında bu adresleme modu hakkında daha ayrıntılı bilgiler mevcuttur.

Dolaysız adresleme

Dolaysız adresleme modu, işkodu ile işleneni oluşturan veriden meydana gelen, iki-parçalı bir komutu gerektirir. 8-bitlik bir mikroişlemcide işlenen, komuta bağlı olarak bir ya da iki bayt yer işgal edecektir. Bu adresleme modunun, bir baytlık işleneni bulunan bir örneği aşağıdadır:

```
MVI   X'F0'      kod 3E F0
```

Bu, A'ya F0'ı yükleyen bir 8080 komutudur. Aşağıda gösterilen komut Z80'e aittir ve H, L, kaydedici çiftini 16-bitlik 10F0 işleneni ile yükler:

```
LD    HL, X'10F0'      kod B6 F0 10
```

Doğrudan adresleme

Bu adresleme modu ile komut, açık bir şekilde iki parçaya ayrılmıştır: işlenen ve adres. Bu mod ile dolaysız adresleme arasındaki farka dikkat ediniz. Dolaysız adreslemeyle veri, komutun bir parçası olarak verilir. Doğrudan adreslemede ise veri adresi komutun bir parçasıdır. 16-hatlık adres yolu olan mikroişlemcilerde bu adres 16 bitten oluşacaktır. Bu nedenle 8080, Z80 veya M6800 gibi tipik 8-bitlik mikroişlemcilerde doğrudan adresleme modu içeren bir komut üç bayttan oluşacaktır, tek- baytlık bir işkodu ve iki baytlık bir adres. Buna bir örnek, bellekte 2F00 konumunun içeriğini A kaydedicisine yükleyen Z80 komutudur:

```
LD    A, X'2F00'      kod 3A 00 2F
```

Bu durum şematik olarak Şekil 2.10'da gösterilmiştir. 1000 konumunda bulunan komutun çalıştırılmasının sonucunda A, 1F ile yüklenecektir.

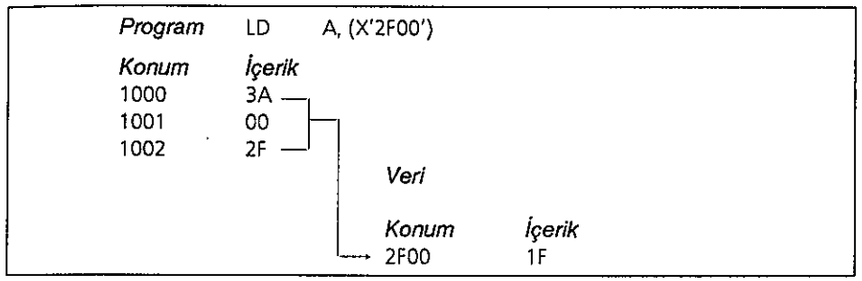
M6800'a ilişkin kaynaklarda doğrudan adresleme tanımında önemli bir fark bulunmaktadır. M6800 üretici veri sayfalarında, doğrudan adreslemenin iki tipi anlatılmaktadır. Birinci tipte, komut sözcüğünün, yalnızca, adresin en düşük değerlikli olan baytını içermesi gerekir. Komut çalıştırıldığında adresin en düşük değerlikli baytının sıfır olduğu kabul edilir. Bu adresleme modunu kullanan bir komut örneği aşağıdadır:

```
LDA   A, X'F3'      kod 96 F3
```

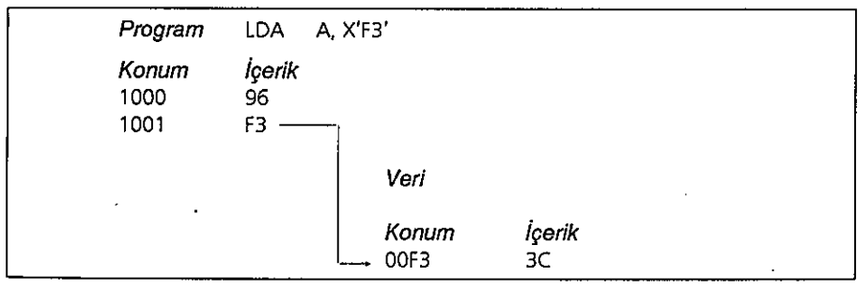

Komut, çalıştırıldığında, 00F3 bellek konumu içeriğini A kaydedicisine yükler. Bu adresleme modu, doğrudan adreslemenin daha yaygın biçimine göre, daha hızlı yürütme ve daha az bellek kullanımı avantajlarına sahiptir, ancak, yalnızca 0000'dan 00FF'ye kadar olan aralıktaki adreslere erişilebilmesi bir dezavantajdır. Normalde, bu moddan, sık sık kullanılan ve belleğin alt sıralı kısımda saklanan işlenenler için yararlanılır. Şekil 2.11'de doğrudan adreslemenin bu değişik tipi gösterilmektedir, çalıştırıldığında A kaydedicisini 3C ile yükler.

Doğrudan adreslemenin daha yaygın biçimi, M6800 belgelerinde genişletilmiş adresleme olarak adlandırılır. Bu mod, iki-baytlık adres gerektirir ve 0000 'dan FFFF'ye tüm adres alanına erişime izin verir. Bir örneği aşağıdadır:

```
LDA   A, X'10F3'      kod B6 10 F3
```



Şekil 2.10 2F00 konumunun doğrudan adreslenmesi



Şekil 2.11 M6800'de değiştirilmiş doğrudan adresleme

Kaydedici dolaylı adresleme

Bu adresleme modu 8080, 8085, Z80 ve Z8000 aileleri ve başka mikroişlemciler tarafından desteklenmektedir, M6800 ailesi bu modu farklı bir biçimde kullanır.

Dolaylı bir adres kullanan komutta komut sözcüğü, adres işlenenini içeren b kaydedici ya da kaydedici çiftini tanımlar. Kaydedici çiftine çoğu kez adr işaretçisi denir. Örneğin aşağıda gösterilen Z8000 komutu, R8 kaydedicisini, R kaydedicisinde bulunan bir adresten yükler:

LD R8, qr1 kod 2F 18

Burada QR1, kaydedici dolaylı adreslemesini belirtmek için kullanılmıştır. Z80 için standart bir assembly dili formatıdır. Dolaylı adresleme kullanan bir komu yürütmeden önce, işaretçi kaydedici(ler)in doğru adreslerle yüklenmesini sağlay bilmek, elbette ki programcının sorumluluğundadır. Kaydedici dolaylı adreslem sinin bu örneği Şekil 2.12'de şematik olarak verilmiştir. Kaydedici dola adreslemesinin, bu defa 8080 mikroişlemcisi için, bir başka örneği aşağıda gös rilmiştir. Komut, H,L kaydedici çiftinde bulunan adresin içeriğini, A kaydedicisi eklemektedir:

ADD M kod 86

İndeksli adresleme

İndeksli adresleme, kaydedici dolaylı modunun, dolaylı adres işaretçisine bir ofs eklenmesine izin veren genişletilmiş bir biçimi gibi kabul edilebilir. Eğer bir komu indeksli adreslemeyi belirtiyorsa, komut kodu bir ofset içerecektir. Komut çalış rıldığında, işlemci, bu ofseti bir indeks kaydedicisinin içeriğine ekleyerek sonuç yeni bir adres üretir. İndeksli adresleme modu 8080/8085 cihaz ailesi tarafınd desteklenmemektedir, fakat 6800, Z80 ve Z8000 işlemcileriyle birlikte kullanıl bilir.

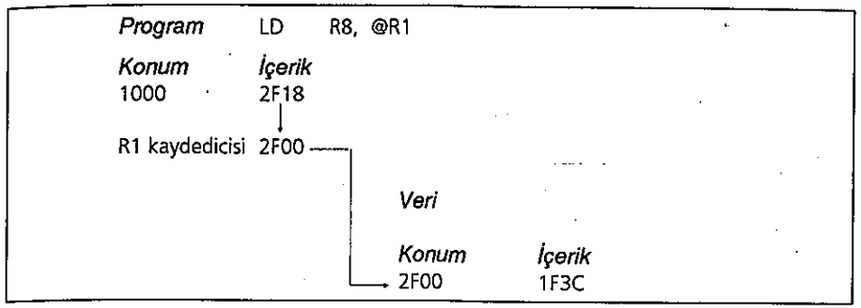
M6800 mimarisi, Şekil 2.8'deki programcı modelinde gösterildiği gibi, tek b indeks kaydedicisi içerir. İndeks kaydedicisinin doğru 'taban adresi' ile yüklenme sini ve böylece ofset eklendiğinde istenen veri adresinin bulunmasını sağlama programcının sorumluluğundadır. Diyelim ki bu yapıldı ve 16bitlik X inde kaydedicisinde sözgelimi 2000 bulunuyor. O zaman, indeksli adreslemeli b M6800 bellek referans komutu için çevirici formatı: S2.12

2.13

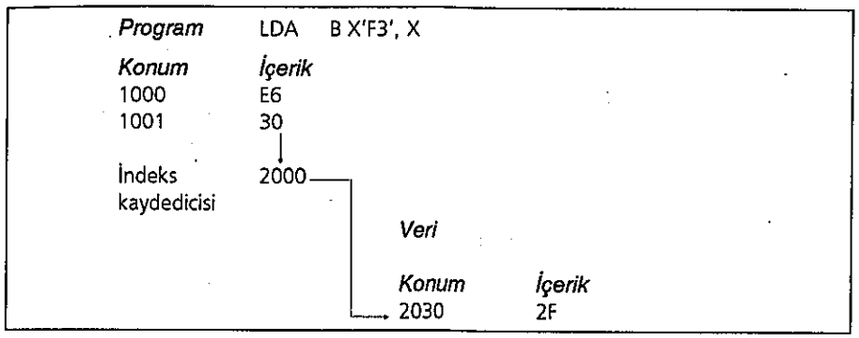
LDA B X'30', X olacaktır.

Burada, X'30', Futuredata çeviricisinde, genelde olduğu gibi, bir onaltılı sayı göstermektedir. İkinci 'X' ise indeksli adreslemeyi ifade eder. Dolayısıyla sonuç elde edilecek adresi belirleyebilmek için, daha önceden 2000'e kurulan inde

kaydedicisi içeriğine 30 ekleriz. Bu nedenle adres, 2030 olur ve B kaydedicisi, yürütmeden sonra bu konumun veya 2F'nin içeriği ile yüklenir. Bu durum, Şekil 2.13'de diyagram olarak gösterilmiştir.



Şekil 2.12 Z8000'de, kaydedici dolaylı adresleme

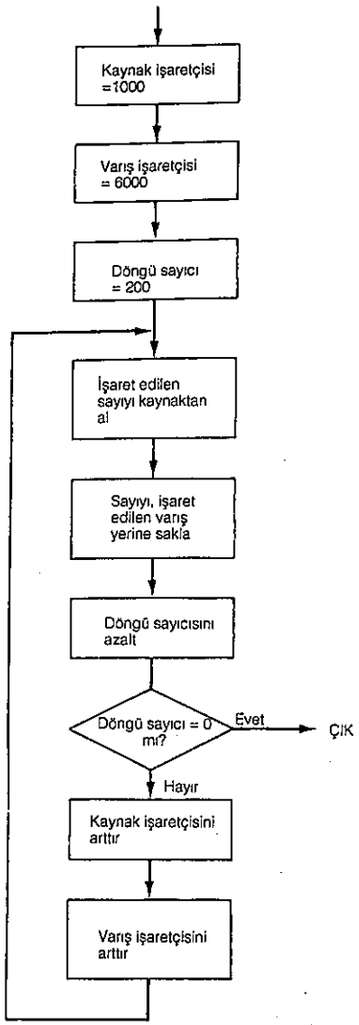


Şekil 2.13 M6800'de indeksli adresleme

İndeksli adresleme modu, verimli program uygulamaları için çok güçlü bir araçtır. Daha sonra geliştirilen mikroişlemciler, bu olanağı geliştiren bir mimariye sahiptir. Örneğin Z80, şekil 2.7'de gösterildiği gibi IX ve IY şeklinde iki indeks kaydedicisi içermektedir. Bu yapının daha da geliştirilmiş şekli Z8000 tarafından ortaya konmaktadır, burada, genel-amaçlı kaydedicilerden en az onbeş tanesi, aynı zamanda adres indekslemesi için de kullanılabilmektedir. Z8000'in kompleks mimarisi, bu bölümde verdiğimiz 'klasik' tanımlamalara uymayan birkaç adresleme modunun kullanımına izin vermektedir.

2.8 Bir programlama problemi

Bu bölümün başlangıcında her işlemcinin kendine özgü üstünlük ve zayıflıklarının bulunduğunu söylemiştik. Bunu göstermek için, üç farklı mikroişlemci tarafından



Şekil 2.14 Veri aktarım programı akış diyagramı

gerçekleştirilen tipik bir işlemi inceleyeceğiz. İşlemciler, h de 8-bitlik cihazlar olan 8080, Z80 ve M6800 mikroişlem ridir. Görev ise, 2000(onaltılı) baytlık bir veri bloğunu, ka bellek yeri 1000-2FFF'den, varış yerine 6000-7FFF'ye ak maktır. Üç mikroişlemciyi karşılaştırmak için kullanacağı ölçütler şunlar olacaktır: Yürütme hızı, gerekli makine ko nun bayt sayısı, ve, standart bir algoritmanın ne kadar ko gerçekleştirilebildiği.

Şekil 2.14, kullanılacak algoritma için akış diyagramını termektedir. Yöntem, iki adres işaretçisi kullanan standart yöntemdir, bu iki adres işaretçisinden birisi, bir sonraki ad veri baytının elde edildiği kaynak yerinde, diğer işaretçiyi bir sonraki adresi, veri baytının saklanacağı varış yeri tutmaktadır. Doğru sayıda baytın aktarılmış olduğunu sa yabilmek için bir döngü sayıcı kullanılmıştır. Her bir aktarımı tamamlandıktan sonra döngü sayacı azaltılır ve s değerine ulaşıp ulaşmadığı kontrol edilir. Eğer sıfır de görülmezse kaynak ve varış işaretçileri arttırılır ve bir sonr veri aktarımı gerçekleştirilir. İşlem, döngü sayıcısı sıfıra ul tığı zaman durur.

Şekil 2.14'teki akış diyagramında, programın, bir kullanı hazırlama, ve bir de veri aktarımı kısmı olmak üzere parçaya ayrıldığı hemen görülmektedir. Kullanıma hazırla bölümünde kaynak işaretçisi, varış yeri işaretçisi ve dö sayıcı, başlangıç değerlerini almaktadır. Bu bölüm yalnızca kez çalıştırıldığı için harcanacak zaman, programın top yürütme zamanının küçük bir kısmıdır, oysa veri aktar bölümü 2000 (onaltılı) kere çalıştırılmaktadır.

8080 çözümü

Öncelikle, bu işlemde, bir 8080 mikroişlemci programcısı karşı karşıya kaldığı probleme bakalım. Makine kodu ko kısaltmaları biçiminde uygun bir program aşağıda göste mektedir:


```
BASLA   LXI  H, X'1000,
        LXI  D, X'6000,
        LXI  B, X'2000,   Kullanıma hazırlama tamamlandı

TEKRAR  MOV  A, M
        STAX D            Bayt aktarma tamamlandı

        DCX  B
        MVI  A, 0
        CMP  C
        JNZ  SAYI
        CMP  B
        JZ   BITIS        Döngü sayıcısı azaltıldı ve sıfır kontrolü yapıldı

SAYI    INX  X
        INX  D            İşaretçiler arttırıldı

        JMP  TEKRAR
BITIS   RET
```

Bu program için bir assembly listelemesi Şekil 2.15'te verilmiştir. 8080 ile adres işaretçisi tanımlanmasının tek yolu, dolaylı adresleme kullanmaktır. Bu nedenle kullanıma hazırlama işlemi, kaynak işaretçisini, H,L kaydedici çiftine, kaynak yerindeki ilk adres olan 1000'i yükleyerek yapılır. Varış yerindeki ilk adres 6000'dir ve bu sayı D,E kaydedici çiftine yüklenmektedir. B,C kaydedici çifti, aktarılacak baytların onaltılı sayısı olan 2000 ile yüklenmektedir ve bir döngü sayıcı gibi işlev görmektedir.

Veri aktarma döngüsündeki MOV A,M komutu, A kaydedicisine, adresi H,L kaydedici çiftinde bulunan bellek konumunun içeriğini yükler. Bu, dolaylı adresleme için bir örnektir. Daha sonraki STAX D komutu, veri baytını, adresi D,E kaydedici çiftinde bulunan, bellek konumunda, A'ya saklar - yine dolaylı adresleme. Bu yolla, kaynaktan varış yerine bir baytın aktarılması tamamlanmış olmaktadır.

Bu noktada döngü sayıcı azaltılmalı ve sıfıra ulaşıp ulaşmadığı kontrol edilmelidir. Bu işlem, program içindeki koşullu atlama komutu, JZ tarafından yapılmaktadır. JZ komutu, eğer Z (sıfır) bayrağı kurulmuşsa atlamayı gerçekleştirir. Tek bir kaydedici, aktarılacak bayt sayısı olan 2000'i tutamayacağından, burada B,C kaydedici çifti 16-bitlik bir döngü sayacı gibi kullanılmaktadır. 8080 komut takımında, bu kaydedici çiftinin içeriğini azaltan DCX B şeklinde bir komut bulunmaktadır. Fakat bu, sınırlı bir komuttur, çalışmasının durum bayraklarına bir etkisi yoktur.

```
                                                                    PAGE  1
----- 8080/85 ASSEMBLER V01 ------------------------------------ FUTUREDATA ---------

                  * ****************************************** *
                  *  8080 VERİ BLOĞU AKTARMA PROGRAMI          *
                  *  BU PROGRAM 1000-2FFF'DEN 6000-7FFF'YE 2000(ONALTILI) BAYTLIK  *
                  *  BİR BLOK AKTARIR                          *
                  * ****************************************** *

0000  210010      BASLA     LXI     H, X'1000,      KAYNAK İŞARETÇİSİ
0003  110060                LXI     D, X'6000,      VARIŞ YERİ İŞARETÇİSİ
0006  010020                LXI     B, X'2000,      DÖNGÜ SAYACI
                  *KULLANIMA HAZIRLAMA TAMAMLANDI
                  *
0009  7E          TEKRAR    MOV     A, M            İŞARETLENEN SÖZCÜĞÜ AL
000A  12                    STAX    D               VARIŞ YERİNE SAKLA
000B  0B                    DCX     B               DÖNGÜ SAYICISINI AZALT
000C  3E00                  MVI     A, 0
000E  B9                    CMP     C               EDDB-0 MI?
000F  C21600                JNZ     SAYI            HAYIRSA ATLA
0012  B8                    CMP     B               EDB-0 MI?
0013  CA1B00                JZ      BITIS           İŞ SONU
0016  23          SAYI      INX     H               KAYNAK İŞARETÇİSİNİ ARTTIR
0017  13                    INX     D               VARIŞ YERİ İŞARETÇİSİNİ ARTTIR
0018  C30900                JMP     TEKRAR          TEKRARLA
001B  C9          BITIS     RET
001C                        END
```

Şekil 2.15 8080 veri bloğu aktarma programı

Bu sınırlama çok önemlidir, çünkü JZ 'sıfırsa atla' komutunun etkisi, bir önc komut tarafından Z bayrağının kurulmasına bağlıdır.

Bu zorluğun üstesinden gelmek için A kaydedicisi sıfırla yüklenir ve, C ve kaydedicileri ile sırayla karşılaştırılır. Karşılaştırma komutu CMP, DCX'in aksi Z bayrağını etkiler ve bu nedenle etkin bir koşullu atlama, bu komutu izleyebil Döngü sayıcı azaltılması işlemi sıfırla sonuçlanmadığı zaman, adres işaret kaydedici çiftleri D,E ve H,L arttırılır ve döngünün başında, koşulsuz bir atla gerçekleşir. Program bir altyordam biçiminde yazılmıştır ve çıkış, altyordam dönüş komutu, RET aracılığıyla olmaktadır.

8080 komut takımının, bu programlama görevi için uygunluğunu gösteren bir kan programın yürütümü için harcadığı süredir. Bunu belirlemek için, programda çeşitli komutların çalıştırılmasında gerekli olan saat saykıl sayıları listelenmiş Ek A'nın bir bölümü olarak verilen 8080 komut takımı özeti, takımdaki her komutun çalıştırılması için gerekli olan saat saykıl sayısını göstermektedir. Bur dan, tüm programın yürütülmesi için gerekli olan saat saykıllarının toplam say hesaplanabilir. Yürütme zamanı, saat saykılları ve saat periyodu tarafından sabi lenmiştir. Farklı mikroişlemciler arasında bir çeşit anlamlı karşılaştırma yapı çabasıyla, her cihazın standart versiyonunun maksimum saat frekansını esas alac

ğız. Bugün, mikroişlemcilerin çoğunun, standart versiyonlarına ek olarak, yüksek-hızlı versiyonlar bulunmaktadır.

8080 programının kullanıma hazırlama bölümündeki üç LXI komutunun her biri 10 saat saykılı gerektirir ve bu, programın bu kısmı için toplam 30 saykıl anlamına gelir. Aslında kullanıma hazırlama bölümünün, programın yürütme zamanına katkısı minimum düzeydedir ve yalnızca bütünlük için burada belirtilmiştir. Her bir turda veri aktarma döngüsü, yani çalışan komutların sırası aynı olmayacaktır. Bunun nedeni, koşullu atlama komutlarının bazen atlamayı başlatması, bazan başlatmamasıdır. Bir koşullu atlamanın çalıştırılması için geçen süre de, atlamanın gerçekte oluşup oluşmadığına bağlı olarak değişir. Veri aktarma döngüsünün komutları içerisinde üç olası rota bulunmaktadır. Bunlar aşağıda A, B ve C olarak gösterilmiş ve saat saykılları belirtilmiştir. Atlamanın oluşmasına neden olan koşullu atlama komutları, yıldızla işaretlenmişlerdir.

	Komut			*Saat saykılı*
A rotası	TEKRAR	MOV	A, M	7
		STAX	D	7
		DCX	B	5
		MVI	A, 0	7
		CMP	C	4
		* JNZ	SAYI	10
	SAYI	INX	X	5
		INX	D	5
		JMP	TEKRAR	10
				60
B rotası	TEKRAR	MOV	A, M	7
		STAX	D	7
		DCX	B	5
		MVI	A, 0	7
		CMP	C	4
		JNZ	SAYI	7
		CMP	B	4
		JZ	BITIS	7
		INX	X	5
		INX	D	5
		JMP	TEKRAR	10
				68

```
C rotası    TEKRAR   MOV   A, M     7
                     STAX  D        7
                     DCX   B        5
                     MVI   A, 0     7
                     CMP   C        4
                     JNZ   SAYI     7
                     CMP   B        4
                     *JZ   BITIS    10
            BITIS    RET            10
                                    61
```

Yukarıdaki saat saykıllarının tümünün ondalık biçimde olduklarına dikkat edin

Döngü turlarının toplam sayısı, şimdi, 2000 veya 8.192 (ondalık)'dir. Bunlard C, döngüyü sona erdirdiği için yalnızca bir kez oluşur. B, döngü sayısı 0100, 020 0300-1F00, yani 1F ya da 31(ondalık) defa olduğu zaman ortaya çıkacaktır. Sa saykıllarının toplam sayısı, N:

(1 x 61) + (31 x 68) + (8.160 x 60)
= 491.769 (ondalık)'dur.

Buna, kullanıma hazırlama bölümü için alınan 30 saykıl eklenmelidir, bu 491.799(ondalık) genel toplamını verir. 8080 için minimum saat periyodu 32 nsn'dir. Dolayısıyla yürütme süresi:

516.464 x 320 nsn = 157.37 msn olur.

M6800 çözümü

M6800 mikroişlemcisi ile veri aktarımını gerçekleştiren bir program, aşağı makine kodu biçiminde gösterilmiş, Şekil 2.16'da da tam bir assembly listeleme verilmiştir.

```
BASLA    LDX    #X'1000,
         STX    KISARETCI
         LDX    #X'6000,
         STX    VISARETCI
         LDX    #X'2000,
         STX    DSAYISI        Kullanıma hazırlama tamamlandı
```



```
TEKRAR   LDX    KISARETCI
         LDAA   0, X
         LDX    VISARETCI
         STAA   0, X           Bayt aktarma tamamlandı

         LDX    DSAYISI
         DEX
         BEQ    BITIS          Döngü sayıcı azaltıldı ve sıfır
         STX    DSAYISI        kontrolü yapıldı

         LDX    KISARETCI
         INX
         STX    KISARETCI
         LDX    VISARETCI
         INX
         STX    VISARETCI      İşaretçiler arttırıldı

         BRA    TEKRAR
BITIS    RTS
```

M6800, dolaylı adresleme modunun kullanılmasına izin vermez. Bu durumda, adresin işaret edilmesi, indeksli adresleme modu kullanılarak gerçekleştirilir. İndeksli adresleme modunu kullanan bellek referanslı bir komutta, komut kodunun bir parçası olarak 8-bitlik bir adres ofseti bulunur. Komut çalıştırıldığında, işlemci, 8-bitlik ofseti 16-bitlik indeks kaydedicisi X'in içeriğine ekleyerek yeni bir adres oluşturur. İndeksli adreslemeyle yükleme ve saklama komutlarının formatı aşağıda verilmiştir:

```
LDAA    d, X
STAA    d, X
```

Burada 'd' offset, ya da yer değiştirmedir. Şekil 2.16'daki programın buna uyan komutları şöyledir:

```
LDAA    0, X
STAA    0, X
```

Şekil 2.16'da gösterilen programda, indeks kaydedicisi, aslında, önceki örnekte bulunan 8080 kaydedici çiftleri H,L ve D,E ile aynı rolü oynamaktadır, çünkü ofset her durumda sıfırdır.

```
                                                          PAGE  1
..... M6800 ASSEMBLER V01 ..........................  FUTUREDATA .........

                 * ****************************************** *
                 *                                            *
                 * M6800 VERİ BLOĞU AKTARMA PROGRAMI          *
                 * BU PROGRAM 1000-2FFF'DEN 6000-7FFF'YE 2000(ONALTILI)
                 * BAYTLIK BİR BLOK AKTARIR                   *
                 *                                            *
                 * ****************************************** *

0000  CE1000     BASLA     LDX    #X'1000'
0003  FF0037               STX    KISARETCI        KAYNAK İŞARETÇİSİ
0006  CE6000               LDX    #X'6000'
0009  FF0035               STX    VISARETCI        VARIŞ YERİ İŞARETÇİSİ
000C  CE2000               LDX    #X'2000'
000F  FF0039               STX    NSAYISI          DÖNGÜ SAYICISI
                 * KULLANIMA HAZIRLAMA TAMAMLANDI
                 *
0012  FE0037     TEKRAR    LDX    KISARETCI        KAYNAK İŞARETÇİSİNİ AL
0015  A600                 LDAA   0, X             İŞARETLENEN SÖZCÜĞÜ AL
0017  FE0035               LDX    VISARETCI        VARIŞ YERİ İŞARETÇİSİNİ AL
001A  A700                 STAA   0, X             VARIŞ YERİNE SAKLA
001C  FE0039               LDX    DSAYISI          DÖNGÜ SAYICISINI AL
001F  09                   DEX                     DÖNGÜ SAYICISINI AZALT
0020  2719                 BEQ    BITIS            İŞLEM TAMAM MI?
0022  FF0039               STX    DSAYISI          DÖNGÜ SAYICISINI SAKLA
0025  FE0037               LDX    VISARETCI
0028  08                   INX                     KAYNAK İŞARETÇİSİNİ ARTTIR
0029  FF0037               STX    KISARETCI        KAYNAK İŞARETÇİSİNİ SAKLA
002C  FF0035               LDX    VISARETCI
002F  08                   INX                     VARIŞ YERİ İŞARETÇİSİNİ ARTTIR
0030  FF0035               STX    VISARETCI        VARIŞ YERİ İŞARETÇİSİNİ SAKLA
0033  20DD                 BRA    TEKRAR           TEKRARLA
                 *
0035              VISARETCI DS     2                KAYNAK İŞARETÇİSİ
0037              KISARETCI DS     2                DÖNGÜ SAYICISI
0039              DSAYISI   DS     2
                 *
003B  39          BITIS     RTS
003C                        END
```

Şekil 2.16 M6800 veri bloğu aktarma programı 1

Burada ele aldığımız örnekteki gibi programlama problemleri için, M6800 il indeksli adreslemenin kullanımında iki sınırlama vardır. Birincisi, adres ofsetini maksimum değeri FF'dir, bu da ofset kolaylığını, bu uygulama için değersiz hal getirir. Bu konu, adresleme modlarını kapsayan Bölüm 2.7'de daha ayrıntılı şekild tartışılmıştır. İkinci sınırlama ise M6800 mimarisinin, adres indekslemesinde kulla nılabilecek yalnızca bir kaydedici içermesidir. Şekil 2.14'te gösterilen algoritma kaynak ve varış adreslerinin her biri için birer tane olmak üzere iki adres işaretçis gerektirmektedir. Bir başka problem de, Bölüm 2.6'da anlatıldığı gibi 16- bitlik b kaydedicisin gerekli olmasında yatmaktadır. M6800'de programcı, yalnızca 16-bi lik iki kaydediciye erişebilmektedir. Bunlar yığın işaretçisi S, ve indeks kaydedicis X'tir. Yığın işaretçisi bir döngü sayıcı gibi kullanılabilir. Yığın işaretçisi ayrıca kesme hizmeti sırasında altyordam dönüş adreslerini ve işlemci durum bilgisini saklamada önemli bir role sahiptir. Bu açıdan, başka amaçlar için kullanılırken, çok

dikkatli olunması gereklidir. Şekil 2.16'da gösterilen program, X kaydedicisini görevlendirerek, yığın işaretçisinin bir döngü sayıcı gibi kullanılmasını engellemiştir.

X kaydedicisi bu programda, üç ayrı işlev gerçekleştirmektedir. Bunlar:

a Bir kaynak adres işaretçisi
b Bir varış yeri adres işaretçisi
c Bir döngü sayıcı

Bu üç işlevi yerine getirmek için, kaynak adresi, varış yeri adresi ve döngü sayacı olmak üzere üç değişkenin, mevcut değerleri bellekte saklanır. Bunlar daha sonra, gerektiğinde X kaydedicisine yüklenir ve değişiklikten sonra belleğe geri alınırlar.

Programın kullanıma hazırlama bölümünde, X kaydedicisi, ilk kaynak adres olan 1000 ile yüklenir. Bu 16-bitlik değer daha sonra iki bellek konumunda saklanır, bu bellek konumlarından ilki KISARETCI sembolik adresi ile tanımlanmıştır. Tek STX komutu bunu gerçekleştirmektedir. Varış yeri işaretçisi ve döngü sayacının kullanıma hazırlanması da, daha sonra benzer şekilde yapılır.

Veri aktarma döngüsü Şekil 2.14'te gösterilen akış diyagramını aynen izler. Assembly listelemesindeki açıklamalar, programa, kendinden açıklamalı bir özellik kazandırmaktadır. Kaynak adresi, varış yeri adresi ve döngü sayacının yürürlükteki değerlerinin, program tarafından kullanılmalarından önce, bellekten X'e yüklendiklerine dikkat ediniz.

Her ne kadar Şekil 2.16'da verilen program, Şekil 2.14'teki akış diyagramının açıkladığı algoritmayı yerine getirirse de, bunun aslında M6800 için biraz verimsiz bir program olduğu ileri sürülebilir. Komut takımının incelenmesi, programın daha hızlı çalışmasını ve bellekte daha az yer kaplamasını sağlayan basit bir değişikliği ortaya koyacaktır. Değiştirilmiş program aşağıdadır:

```
BASLA    LDX    #X'1000,
         STX    KISARETCI
         LDX    #X'6000,
         STX    VISARETCI      Kullanıma hazırlama tamamlandı

TEKRAR   LDX    KISARETCI
         LDAA   0, X
         INX
```

```
        STX     KISARETCI
        LDX     VISARETCI
        STAA    0, X          Bayt aktarma tamamlandı

        INX
        CPX     #X'8000'
        BEQ     BITIS
        STX     VISARETCI     İşaretçiler arttırıldı

        BRA     TEKRAR
BITIS   RTS
```

X'le karşılaştır komutu CPX'in kullanımıyla, ayrıca bir döngü sayıcının kullanımına gerek kalmadığı görülmektedir. Veri aktarma döngüsü sırasında, varış yeri adres işaretçisi, X kaydedicisinde arttırıldıktan sonra, X'in içeriği 8000 ile karşılaştırılır. Bu, varış yeri adresinin sınır değeridir. Eğer bir eşleşme bulunursa, veri aktarma tamamlanmıştır ve koşul kodu kaydedicisinin (bayrak) Z biti 1'e ayarlanır. Sıfır komutu BEQ'da bulunan bir sonraki koşullu dallanma, programın altyordam dönüşü RTS aracılığıyla, döngüden çıkmasına neden olur.

Bu değişik program Şekil 2.14'teki akış diyagramına tam olarak uymaz. Değiştirilmiş bir akış diyagramı Şekil 2.17'de gösterilmektedir.

Deneyimli bir M6800 programcısı, CPX komutunun bu gibi durumlardaki kullanımı hakkında elbette bilgi sahibidir ve algoritmayı başlangıçtan itibaren bunu göz önünde bulundurarak tasarlayacaktır. Bu örnek, komut takımı hakkında ayrıntılı bilgi sahibi bulunmanın, programcıya program yapısını, kullanılan mikroişlemcinin güçlü yönlerinden yararlanarak tasarlama olanağını nasıl verdiğini gösteren, iyi bir örnektir.

Yürütme zamanını hesaplarken, programın daha verimli bir versiyonunu kullanacağız. Kullanıma hazırlama komutlarının gerektirdiği saat saykılları sayısı (ondalık biçimde) aşağıda gösterilmektedir:

Komut		*Saat saykılı*
LDX	#X'1000'	3
STX	KISARETCI	6
LDX	#X'6000'	3
STX	VISARETCI	6
		18


```
                                                          PAGE  1
M6800/2 ASSEMBLER V01 ..........................        FUTUREDATA .........

        * ***************************************** *
        *  M6800 VERİ BLOĞU AKTARMA (DEĞİŞTİRİLMİŞ) PROGRAMI          *
        *  BU PROGRAM 1000-2FFF'DEN 6000-7FFF'YE 2000(ONALTILI)        *
        *  BAYTLIK BİR BLOK AKTARIR                                    *
        * ***************************************** *

0000 CE1000   BASLA    LDX    #X'1000'
0003 FF0027            STX    KISARETCI        KAYNAK İŞARETÇİSİ
0006 CE6000            LDX    #X'6000'
0009 FF0025            STX    VISARETCI        VARIŞ YERİ İŞARETÇİSİ
     * KULLANIMA HAZIRLAMA TAMAMLANDI
     *
000C FE0027   TEKRAR   LDX    KISARETCI        KAYNAK İŞARETÇİSİNİ AL
000F A600              LDAA   0, X             İŞARETLENEN SÖZCÜĞÜ AL
0011 08                INX                     KAYNAK İŞARETÇİSİN ARTTIR
0012 FF0027            STX    KISARETCI        KAYNAK İŞARETÇİSİNİ SAKLA
0015 FE0025            LDX    VISARETCI        VARIŞ İŞARETÇİSİNİ AL
0018 A700              STAA   0, X             SÖZCÜĞÜ VARIŞ YERİNE SAKLA
001A 08                INX                     KAYNAK İŞARETÇİSİNİ ARTTIR
001B 8C8000            CPX    #X '8000'        TAMAMLANDI MI ?
001E 2709              BEQ    BITIS
0020 FF0025            STX    VISARETCI        VISARETICI'YI SAKLA
0023 20E7              BRA    TEKRAR           TEKRARLA
     *
0025           VISARETCI DS   2                VARIŞ YERİ İŞARETÇİSİ
0027           KISARETCI DS   2                KAYNAK İŞARETÇİSİ
     *
0029 39        BITIS  RTS
002A                  END
```

Şekil 2.17 M6800 veri bloğu aktarma programı 2

Veri aktarma döngüsünde A ve B olarak gösterilen iki alternatif rota bulunmaktadır.

	Komut			*Saat saykılı*
A rotası	TEKRAR	LDX	KISARETCI	5
		LDAA	0, X	5
		INX		4
		STX	KISARETCI	6
		LDX	VISARETCI	5
		STAA	0, X	6
		INX		4
		CPX	#X '8000'	3
		BEQ	BITIS	4
		STX	VISARETCI	6
		BRA	TEKRAR	4
	BITIS	RTS		52

B rotası	TEKRAR	LDX	KISARETCI	5
		LDAA	0, X	5
		INX		4
		STX	KISARETCI	6
		LDX	VISARETCI	5
		STAA	0, X	6
		INX		4
		CPX	#X '8000'	3
		*BEQ	BITIS	4
	BITIS	RTS		5
				47

Program yürütülmesi sırasında kullanıma hazırlama komutları bir defa, B bir d ve A ise 8,191(ondalık) defa çalıştırılır. Böylece toplam saat saykılı sayısı N,

18 x 47 x (8.191 x 52) = 425.997 (ondalık) olur.

Standart M6800 için minimum saat periyodu 1.000 nsn'dir. Dolayısıyla yürüt zamanı;

425.997 x 1.000 nsn = 426 msn olarak bulunur.

Z80 çözümü

Z80 ile, 8080'e çok benzeyen bir kaydedici-yönelimli mimariye dönmekteyiz. Z komut takımındaki komutlar, büyük çoğunlukla 8080 komutlarının aynısıdır. A rıca çok sayıda ek komutlar da vardır. Bu bilgiler ışığında, Z80 programının 80 programına benzemesini bekleyebiliriz. Şekil 2.18'deki listeleme ve aşağıda gö terilen program bunun doğru olmadığını ortaya koymaktadır.

BASLA	LD	DE, X '6000'	
	LD	HL, X '1000'	
	LD	BC, X '2000'	*Kullanıma hazırlama tamamland*
	LDIR		
	RET		

8080 ve Z80 programları arasındaki fark, Z80 programında LDIR komutunu kullanılmasında yatmaktadır. Bu komut, 8080 komut takımında bulunmayan güçlü

bir veri aktarma komutudur. LDIR komutu çalıştırıldığında, H,L kaydedici çiftinin işaret ettiği adresin içeriği, D,E kaydedici çifti tarafından işaretlenen adrese aktarılır. Daha sonra H,L ve D,E kaydedici çiftlerinin içerikleri arttırılır ve B,C kaydedici çiftinin içeriği azaltılır. Bu işlem sonunda B,C'de sıfır bulunuyorsa, bir sonraki komut çalıştırılır, bulunmuyorsa, veri aktarma, işaretçilerin yeni değerleriyle tekrarlanır.

LDIR'ın yukarıdaki tanımlamasından, tek başına bu komutun, Şekil 2.14'teki akış diyagramında bulunan veri aktarma bölümünü tam olarak gerçekleştirdiği görülmektedir.

```
                                                        PAGE  1
... Z80 ASSEMBLER V01  ------------------------------  FUTUREDATA ---------
                 * ***********************************************  *
                 *                                                  *
                 * Z80 VERİ BLOĞU AKTARMA (DEĞİŞTİRİLMİŞ) PROGRAMI  *
                 * BU PROGRAM 1000-2FFF'DEN 6000-7FFF'YE 2000(ONALTILI) *
                 * BAYTLIK BİR BLOK AKTARIR                         *
                 *                                                  *
                 * ***********************************************  *
0000  110060     BASLA     LD      DE, X '6000'      VARIŞ YERİ İŞARETÇİSİ
0003  210010               LD      HL, X '1000'      KAYNAK İŞARETÇİSİ
0006  010020               LD      BC, X '2000'      DÖNGÜ SAYACI
                 * KULLANIMA HAZIRLAMA TAMAMLANDI
                 *
0009  EDB0                 LDIR                      BLOKU AKTAR
000B  C9                   RET
000C                       END
```

Şekil 2.18 Z80 veri bloğu aktarma programı

Bu programın yürütme zamanını hesaplamak kolaydır. Kullanıma hazırlama için gereken 16-bitlik üç yükleme komutunun her biri, yürütülmeleri için 10 saat saykılına ihtiyaç duyar. Programın geri kalan bölümü için yalnızca iki rota mümkündür. Bunlar aşağıda gösterilmektedir (yine, ondalık biçimde verilen saat saykılları ile birlikte):

	Komut	*Saat saykılı*
A rotası	LDIR	16
		16
B rotası	LDIR	21
	RET	10
		31

Yordamı sona erdirdiği için B, yalnızca bir kez çalıştırılır; kullanıma hazırla komutları da yalnızca bir kez çalıştırılır. A ise 8.191 (ondalık) defa çalıştırı Böylece toplam saat saykılı sayısı N,

30 + 31 + (8.191 x 16) = 131.103 (ondalık) olur.

Z80 için minimum saat periyodu 400 nsn'dir. Dolayısıyla yürütme süresi;

131.103 x 400 nsn = 52.44 msn olarak bulunur.

Bazı sonuçlar

Yukarıdaki bölümlerde, belirli bir programlama görevine olan yaklaşımın, mikroişlemcinin farklı komut takımları ve mimarileri tarafından nasıl etkilendiği inceledik. Ele alınan örnek, oldukça sık karşılaşılan bir örnektir. Veri blok dizilerinin işlenmesi, bir çok farklı program tipinde sıkça gerekli olmaktadır. mikroişlemcinin, bir dizi işleme uygulaması için, etkinlik ve uygunluğunu araştı mış bulunuyoruz.

Belli bir programlama görevi için, üç mikroişlemcinin komut takımlarının gücü ve uygunluğunu değerlendirirken, program yürütme süresi ve belleği doldurm düzeyleri, yararlı yol göstericilerdir. Burada yaptığımız alıştırmanın sonuçla Tablo 2.2'de gösterilmektedir:

Tablo 2.2

İşlemci	*Yürütme süresi*	*Program belleği*
8080	165 msn	25 bayt
M6800	426 msn	42 bayt
Z80	52 msn	12 bayt

Programlarda yapılan, 'ince ayar' niteliğindeki küçük değişikliklerin, yürütme sürelerini ve bellek doldurma düzeylerini sınırlı ölçüde azaltmaları olasıdır. Örne ğin 8080 programında, koşullu atlama JZ'nin yerine, RET'in izlediği koşullu altyordam dönüşü RZ'nin kullanılması, her bir döngü turunda, bir saat saykılı kada tasarruf sağlayacaktır. Bu küçük gelişme, sonuçlarla ilgili yorumlarımızı belirgin ölçüde değiştirmeyecektir.

Buradaki sonuçlara bakıldığında, 2000 baytlık bir veri bloğunun, bellekteki 1000 kaynak yerinden, 6000'deki varış yerine aktarılmasında, Z80'in, 8080'den daha

kısa bir yürütme süresi sağladığı ve programın daha küçük bir bellek alanını kapladığı açıkça görülmektedir. 8080 de, M6800'e göre, benzer şekilde, daha avantajlıdır. Ancak tek bir programlama göreviyle ilişkili sonuçlar, otomatik olarak, Z80'in, M6800'den daha üstün olan 8080'e karşı genel anlamda bir üstünlüğe sahip olduğu şeklinde genişletilmemelidir. Biz üç mikroişlemcinin komut takımlarının yalnızca bir yönünü araştırdık. Başka bilgi işlem görevleri kolaylıkla, farklı sonuçlar ortaya çıkarabilir.

Amacımız, bir mikroişlemci tipinin diğerine karşı üstünlüğüyle ilgili herhangi bir noktayı kanıtlamaktan çok, karşılaştırma yöntemlerini göstermek ve mikroişlemci performansını belirlemede komut takımının önemini vurgulamaktır.

Fakat üç mikroişlemci ile ilişkili belirli bazı noktalar açıklığa kavuşmuştur. Bunlardan belki de en önemli olanı, özel dizi işleme komutlarının, bu tipte bir işlemi gerçekleştiren programlar üzerinde sahip olabileceği etkidir.

Bu alıştırmada Z80'nin gösterdiği iyi performans, LDIR komutunun varlığından kaynaklanmaktadır. Okuyucu, Ek B'de bulunan Z80 komut takımını incelemeli ve veri dizisinin işlenmesine ilişkin diğer komutlara dikkat etmelidir.

Bu alıştırmada ayrıca 8080'in, 16-bitlik işlenenlerin kullanımına ilişkin bir sınırlaması da ortaya çıkmıştır. Her ne kadar program bir defada 8-bitlik verinin aktarılması ile ilgili ise de, programda işlenen olarak kullanılan değişkenlerin bir çoğu, 16-bitlik sayılardır. Adreslerin kendileri ve aktarılacak veri baytı sayısı bunlara dahildir. 8080'in, DCX B gibi 16-bitlik azaltma komutlarının Z bayrağını etkileyememesi, bu çerçevede yetersizlik olarak değerlendirilmelidir.

Bu alıştırmada M6800 ile karşılaşılan temel sorun, adres işaretçisi olarak kullanılabilen, 16-bitlik, yalnızca bir kaydedici içeren mimari olmuştur. Gördüğümüz gibi, veri aktarma akış diyagramını uygun şekilde gerçekleştirebilmek için en az iki kaydedici gereklidir. Ancak M6800, birisi adres işaretçisi kaydedici X'te yer alan 16-bitlik iki işlenen arasında, mantıksal karşılaştırma işlemi yapmamıza izin verir. Şekil 2.17'deki programda bu, ayrı bir döngü sayacına olan ihtiyacı ortadan kaldırmak için kullanılmıştır. Bir başka avantaj ise, alıştırmada kullanılamamış olsa da, indeksli adresleme modunun bulunmasıdır.

Her ne kadar bu bölümde dikkatimizi, komut takımının, yürütme süresi ve programın belleği doldurma düzeyi üzerine olan etkilerine yoğulaştırdıysak da, diğer etkenler genelde daha önemlidir. Programcılar, açık, iyi yapılandırılmış, izlenmesi ve değiştirilmesi kolay olan program yazmanın öneminin, giderek daha çok farkına

varmaktadır. Komut takımının, assembly dilinde, belirtilen bu özellikleri taşı programlar yazmayı destekleme yeteneği, dikkate alınacak önemli bir konu Program yapısı konusu, Bölüm 6'da daha ayrıntılı olarak tartışılmıştır.

Daha karmaşık bilgi işlem görevleri için, günümüzde genel olarak, assembly ya da makine kodu yerine, PASCAL, COBOL, FORTRAN ve ADA gibi yüks düzeyli dillerin kullanılmasıyla, programların daha kolay ve daha ekonomik şekil geliştirilebileceği kabul edilmektedir. M6800 ve Z8000 gibi, daha yakın geçmi kullanıma sunulan mikroişlemcilerden bazıları, yüksek-düzeyli dillerin destekle mesine özellikle önem veren komut takımlarına sahiptir.

Komut takımlarının bu yönlerinin daha ayrıntılı şekilde incelenmesi, bu kitab amacı dışındadır, fakat okuyucu, gelecekte bu konunun giderek artan bir öne sahip olacağını bilmelidir. 2.1 nolu kaynak, komut takımlarının bu özelliklerin iyi bir tanıtımını yapmaktadır.

2.9 Değerlendirme testleri

Bölüm 2.8'de farklı mikroişlemcilerin, basit bir bilgi işlem görevinin uygulanmas yoluyla nasıl karşılaştırılabileceklerini gördük. Verilen örnekte, işlem performan sının yalnızca, bellekte veri bloklarının taşınması ile ilgili yönü araştırılmıştır Herşeyi içine alan, anlamlı bir işlem gücü sağlamak için, farklı tipte çeşitli hesap lamaları içeren bir dizi benzer alıştırmayı da yapmak gerekir.

Belli bir uygulama için en uygun mikroişlemciyi seçerken, aralarında seçim yapı lacak mikroişlemcilerin her biri için, özel test programlarının tasarlanması, yakla şımlardan birini oluşturmaktadır. Bu programlar, söz konusu uygulamada mikroişlemciden beklenen ana görevleri kapsar. Bundan sonra, mikroişlemcinin verimliliği, yürütme zamanı ve programın belleği doldurma düzeyi parametrelerine göre değerlendirilebilir. 'Değerlendirme testleri' (Bench marks) terimi çoğu kez bu tür testler için kullanılır. Değerlendirme testlerinden ayrıca, mikroişlemci sistem
S2.14 performansının tüm yönlerini karşılaştırmak için de yararlanılır.

Kaynaklar ve bibliyografi

2.1 S. Davis, '16-bit microprocessors - special report', Electronic Design News, pp: 71-85 (5 August 1979).

2.2 Z80 - Assembly Language Programming Manual, Zilog (1978).

2.3 W. Barden, The Z80 Microcomputer Handbook, Howard W. Sams & Co., Inc. (1978)

2.4 M6800 Microprocessor Programming Manual, Motorola Semiconductor Products Inc. (1975).

2.5 L.A. Leventhal, A. Osborne and C. Collins, Z8000 Assembly Language Programming, Osborne/Mc Graw-Hill (1980).

2.6 M. Gooze, 'How a 16-bit microprocessor makes it in an 8-bit world (6809 development)', Electronics, Vol. 52(20), pp: 122-5 (27 September 1979).

Sorular

2.1 COP420L cihazının, 8080, Z80 ve M6800 gibi mikroişlemcilerden kavramsal olarak ne farkı vardır ?

2.2 Ek B'yi kılavuz gibi kullanarak, Z80 $\overline{\text{HALT}}$ hattının, bir durum hattı olarak mı, yoksa bir denetim hattı olarak mı kabul edilebileceğini;
(a) Mikroişlemci
(b) Mikroişlemcinin birlikte bulunduğu sistem açısından belirtiniz.

2.3 Aşağıdaki mikroişlemcilerin gerektirdiği saat fazlarının sayısını sıralayınız.
(a) 8080
(b) Z80
(c) 6800
(d) Z8002

2.4 Bazı mikroişlemci yonga tasarımcılarının, neden 'durum'u diğer sinyallerle çoğullamayı tercih ettiklerini açıklayınız.

2.5 Çoğullanmış durum sinyalleri özelliğine sahip iki mikroişlemci örneği veriniz.

2.6 Z8001 mikroişlemcisi adres yolunun, Z8002'nin adres yolundan farkları nelerdir ?

2.7 Aşağıdaki mikroişlemciler tarafından adreslenebilen maksimum bellek konumu sayısı nedir ?
(a) 8080
(b) Z80
(c) Z8001

2.8 Veriyi 16-bitlik sözcük biçiminde işleyen 8080 komutlarına iki örnek veriniz.

2.9 Mikroişlemci mimarisi ile ne anlatılmak istendiğini açıklayınız.

2.10 Bildiğiniz herhangi bir mikroişlemcinin mimarisini tanımlayan bir diyagram çiziniz.

2.11 Bellek-yönelimli ve kaydedici-yönelimli mikroişlemciler arasındaki farkı açıklayınız.

2.12 Aşağıdaki terimlerin normal tanımlarını söyleyiniz.
(a) Örtük adresleme
(b) Doğrudan adresleme
(c) Dolaysız adresleme
(d) Dolaylı kaydedici adreslemesi
(e) İndeksli adresleme

2.13 Kaydedici dolaylı ve indeksli adresleme modları, iki önemli adresleme modudur. Aşağıdaki mikroişlemcilerde bunlardan hangisi yoktur?
(a) 8080
(b) M6800

2.14 Değerlendirme testi ile ne denmek istendiğini açıklayınız.

2.15 Ek A, B ve C'de verilen komut takımlarını kullanarak, 8080, Z80 ve M6800 için, iki adresin içeriğini toplayan ve sonucu üçüncü bir adreste bırakan programlar yazınız. Her bir programın saklanması için gereken bayt sayısını karşılaştırınız. M6800, 8080 ve Z80'nin, sırasıyla 1, 1 ve 2 MHz'lik saat frekanslarıyla çalıştıklarını varsayarak, üç programın yürütme hızını karşılaştırınız.

Bölüm 3 Mikroişlemci seçiminde diğer hususlar

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde, mikroişlemcilerin aşağıdaki önemli özelliklerini, üretici veri sayfalarının yardımıyla analiz edebilir duruma gelmiş olmalısınız: (i) olanaklar; (ii) güç gereksinimleri.*

3.1 Mikroişlemci G/Ç olanakları

Her bilgi işlem sistemi, verileri, MİB'den bir başka yere, ve bir başka yerden MİB'e aktarmak için gerekli olanakları içermelidir. Çünkü MİB'in temel işlevi, istenen bir çıkış formunun oluşturulabilmesi için, kendisine sunulan veriyi işlemektir. Giriş; bir klavye, dönüştürücü, anahtar ya da çok çeşitli bilgi kaynaklarının herhangi birinden yapılabilir. Çıkış, görüntülenmesi ya da yazdırılması gereken rakamsal sonuçlardan oluşabilir. Diğer yandan çıkış, elektronik veya elektromekanik cihazları denetleyen ve gözle görünmeyen sinyaller olabilir.

Bu nedenle bilgi işlem sistemleri dünyasına genel olarak baktığımızda, bir MİB'i bağlamak istediğimiz dış bir cihazın tipi çok fazla çeşitlilik gösterir. En basit aygıt, tek bir hat bağlantısı gerektirir. Bunlar arasında gösterge lambaları, röleler ve anahtarlar sayılabilir. Lamba ve röle gibi elemanlar için, MİB, tek bir çıkış hattının mantıksal durumunu değiştirmelidir. Çıkış birimi önemli miktarda güç kullanıyorsa, bu çıkış sinyalinin önemli ölçüde yükseltilmesi gerekebilir. Anahtar gibi birimler, anahtarın kapalı ya da açık durumda olmasına bağlı olarak yüksek veya alçak bir MİB giriş sinyali üretmek için, basit bir devreye bağlanırlar.

Daha karmaşık G/Ç birimleri, yani yazıcılar, sayısal göstergeler, klavyeler ve ADC'ler, bilgiyi kodlanmış biçimde alır veya verirler. Örneğin bir yazıcı, her biri sözgelimi 8-bitlik olan kod karakterlerini kabul eder. Kod karakterlerinin her biri, bir harfi, sayıyı ya da noktalama işaretini temsil eder. Ek R'de, bu biçimdeki tipik bir kod gösterilmektedir. Bir ADC, analog girişin büyüklüğünü temsil eden bir kod karakteri oluşturur. Dolayısıyla buna benzer karmaşık birimlerde G/Ç işlemleri, bilgiyi, her biri birkaç veri biti içeren karakterler ya da sözcükler biçiminde aktarırlar.

Bu noktada seri ve paralel veri aktarma arasındaki önemli bir farklılık açıklanmalıdır. N-bitlik bir veri karakterinin iki nokta arasında paralel olarak aktarılması, bir

toprak dönüşü ile birlikte n sayıda bağlantı hattını gerektirir. Karakterin her biti ayrı bir hat üzerinden taşınır. Diğer yandan seri veri aktarma, toprak dönüşü ile birlikte yalnızca tek bir hat bağlantısını kulanır. Seri modda veri bitleri, tek hat üzerinden birer birer aktarılır. Her veri aktarımı yönteminin kendine özgü avantaj ve dezavantajları vardır, bunlar Bölüm 4'te tartışılmıştır. Hem paralel hem de seri veri aktarma, mikrobilgisayar sistemlerinde, verinin, çeşitli tiplerdeki çevresel birimlerden başka bir yere ve başka bir yerden çevresel birimlere aktarılması amacıyla yaygın şekilde kullanılmaktadır. Mikroişlemci tasarımcıları, G/Ç olanaklarının sağlanması konusunda, üç olası yaklaşımdan birini uygulama eğilimindedir. Bunlar aşağıdaki gibi açıklanabilir:

1 Tek-yongalı mikrobilgisayar yaklaşımı
2 G/Ç port yaklaşımı
3 Bellek-haritalı G/Ç yaklaşımı

Elbette her mikroişlemci üreticisi, kendi ürünleri için kabul edilmiş bulunan yaklaşımın büyük farkla en iyi yöntem olduğu konusunda ikna edici iddialar ortaya atmaktadırlar. Her zaman olduğu gibi her yaklaşımın kendine özgü avantaj ve dezavantajları vardır.

Tek-yongalı mikrobilgisayar G/Ç

Tek-yongalı bir mikrobilgisayar ile, özel G/Ç arabirim devre düzenlemelerine ya hiç gerek kalmaz ya da minimum düzeyde ihtiyaç duyulabilir. Ek E'de anlatılan COP420L, bunun güzel bir örneğidir. Adı geçen ekten de görüleceği gibi bu işlemcide, farklı tipte toplam 23 G/Ç hattı bulunmaktadır. Bunlar, mikroişlemci ile dış dünya arasında seri ve paralel veri aktarmayı mümkün kılmaktadır. Bu yaklaşımda, ortaya çıkması muhtemel olan zorluğun, çok yönlülük eksikliği olduğu açıktır. G/Ç olanakları tümüyle mikroişlemci üreticisi tarafından belirlendiğinden, kullanıcının özel G/Ç gereksinimlerinin karşılanmadığı bir durumun ortaya çıkacağını düşünmek zor olmayacaktır. Aynı şekilde, çok basit bir sistemde bulunan G/Ç olanakları, kullanıcının gereksinimlerinden çok daha fazlasını sağlayacak düzeyde olabilir. Bunu, MİB-tipinde mikroişlemcilerinin kullanıcılarının durumu ile karşılaştırınız. Burada kullanıcı, G/Ç olanaklarını, sistem gerekliliklerine tam olarak uyarlamaya çalışır. Tek-yongalı yaklaşımıyla benzer esnekliği sağlamak, her uygulama için özel olarak tasarlanmış bir mikroişlemci gerektirecektir.

Tek-yongalı mikrobilgisayar üreticileri, esneklik sorununun üstesinden gelmek için, iki yaklaşımı benimsemişlerdir. Birinci yaklaşım, birbirinden izole edilmiş tek tek birimler yerine mikroişlemci aileleri imal etmektir. Bu mikroişlemci aileleri,

mimari ve komut takımları açısından büyük oranda benzer özelliklere sahip birimlerden oluşur. Aileye mensup her bir mikroişlemci, mevcut bellek kapasitesi ve bazen de G/Ç olanakları açısından farklılık gösterebilir. Böylece kullanıcı, ailenin içinden, kendi gereksinimlerine en çok uyan birimi seçer.

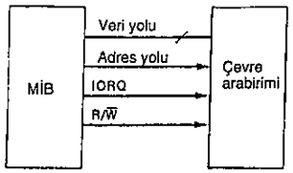
Bazı üreticiler esneklik sorununa, tek-yongalı bir mikrobilgisayar ailesinin her üyesi için belirli G/Ç seçenekleri sağlayarak, ikinci bir şekilde yaklaşmaktadır. COP420L, bu yaklaşıma bir örnektir. G/Ç hatlarının özelliklerine ilişkin çeşitli seçenekler Ek E'de liste halinde verilmiştir. Örneğin bu seçeneklerden birinin, bir LED nümerik göstergenin parçalarını etkinleştirecek düzeyde akım sağlayabilen G/Ç hatları grubuna ilişkin olduğunu görüyoruz. Dolayısıyla mikrobilgisayar, nümerik veriyi, bu hatların, doğrudan bir LED nümerik göstergesine bağlanmasıyla görüntüleyebilecektir. Ek olarak herhangi bir devre düzeneği gerekmemektedir.

G/Ç port yaklaşımı

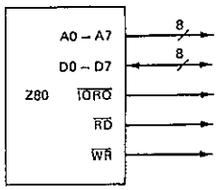
Mikroişlemcilerle gerçekleştirilen G/Ç işlemlerine en yaygın yaklaşım, giriş ve çıkış verisinin üzerinden geçtiği portların sağlanmasıdır. Bu olanağın temel özelliklerinden birisi, komut repertuarında özel-amaçlı komutların bulunmasıdır. Farklı mikroişlemciler bu komutları farklı biçimlerde belirtmektedir, fakat biz, bunun için genel-amaçlı IN ve OUT anımsatıcılarını kullanacağız. Bir IN veya OUT komutu çalıştırıldığında MİB, adres yoluna bir port adresi yerleştirir. Veri daha sonra, veri yolundan alınır ya da verilir. Bu noktada, çevresel birimin veriyi kabul etme ya da veri oluşturma için hazır durumda olmasını sağlamak, çevresel arabirim devresinin (PIC-Peripheral Interface Circuit) sorumluluğundadır.

Mikroişlemci sistemlerinde G/Ç verisi, normalde, bir başka yerden çevresel birimlere veya çevresel birimlerden bir başka yere, veri yolu aracılığıyla taşınır. Bazı mikrobilgisayarlar giriş ve çıkış verisi için ayrı ayrı yollara sahiptir, ancak bu lüksten yararlanmak, mikroişlemci paketinde sınırlı sayıda pin bulunması nedeniyle mümkün olmaz. Bu nedenle sistem veri yolu, bellekten bir başka yere veya bir başka yerden belleğe taşınan veriye ek olarak, G/Ç verisini de taşır. Her ne kadar esas amaç veriyi bir başka yerden çevresel birimlere veya çevresel birimlerden bir başka yere taşımak ise de, G/Ç işlemlerinin çoğunluğu, durum ve denetim bilgilerinin alışverişini de kapsar. Çünkü G/Ç işlemlerinin çoğunluğunda, çevresel birimin faaliyetlerinin, dahili MİB işlemleriyle senkronize edilmesi gereklidir. Bu konu, Bölüm 3.2'de ayrıntılarıyla gözden geçirilmiştir.

Bir giriş ya da çıkış işleminin başarıyla gerçekleştirilebilmesi için MİB, çevresel birimleri, G/Ç işleminin yapılmakta olduğu hakkında bir şekilde bilgilendirebilme-



Şekil 3.1 MİB ve çevresel arabirim arasındaki bağlantılar



Şekil 3.2 G/Ç hatları

lidir. Bunun tipik bir yolu, MİB'nden IORQ şeklinde adlandırılacağımız bir hat üzerine bir sinyalin çıkmasıdır. MİB ayrıca giriş veya çıkış işlemine hangi çevresel birimin katılacağını belirlemelidir. MİB, bunu, söz konusu çevresel birim için ayrılmış bulunan port adresini adres yoluna yerleştirerek yapar.

Sistem veri yolu iki yönlü olduğundan, MİB'in veri akış yönünü belirlemesi gereklidir. Bu yalnızca G/Ç işlemleri için değil, aynı zamanda, bellekten okuma ve belleğe yazma işlemleri için de geçerlidir. Veri yönünün belirtilmesi önemlidir, çünkü veri yoluyla ilişkili tamponların, verinin uygun yönde girmesine izin verebilmeleri sağlanmalıdır. Veri yönü göstergesi olarak R/$\overline{W}$ sinyal tanımlamasını kullanacağız.

G/Ç hatlarının genel bir görünümü Şekil 3.1'de gösterilmiştir. Burada G/Ç portlarının başarılı şekilde kullanılabilmeleri için gerekli olan sinyaller de belirtilmektedir. Bu, G/Ç port yaklaşımından yola çıkan her mikroişlemcinin, özel olarak, örneğin IORQ gibi bir sinyale ayrılmış bir hatta sahip olduğu anlamına gelmez. Ancak, MİB durum bilgisinden IORQ'nun işlevini görecek bir sinyalin oluşturulmasının mümkün olduğu anlamına gelir. İleride göreceğimiz gibi, bu bölümde ele alınan üç mikroişlemciden yalnızca Z80, özel olarak IORQ işlevine ayrılmış bir hatta sahiptir.

Z80 mikroişlemcisi G/Ç port yaklaşımına tam olarak bir örnek oluşturur. Şekil 3.2'de G/Ç işlemlerine katılan hatlar gösterilmektedir. Bir IN veya OUT komutu çalıştırıldığında IORQ hattı alçak düzeye iner. Port adresi, adres yolunun en düşük değerlikli 8 hattında görünür ve veri 8 veri yolu hattından alınır ya da verilir. Çeşitli sinyallerin zamanlamaları ve G/Ç komutlarının sınırları Ek B'de bulunabilir.

Z80 gibi 8080 de G/Ç yaklaşımını kullanmaktadır. Ancak bu durumda, arabağlantı kurma işlemi, özellikle IORQ sinyalini taşıyan bir hat bulunmadığı için karmaşık hale gelmektedir. Bunun yerine durum sinyalleri, her komut çevriminin bir bölümü süresince veri yolu hatlarında ortaya çıkar. Tablo 3.1 bu durum sinyallerinin işlevini göstermektedir. Durum sinyallerinden yararlanmak için, durum baytı, veri yolunda görüldüğü zaman yakalanarak bir kaydedici içine alınabilir. 8080 SYNC sinyalini oluşturur, bu sinyal bu durumda, Bölüm 4.3'te anlatıldığı gibi kaydedici mandallaması amacıyla kullanılabilir. Ek A'da, 8080 G/Ç işlemlerinin zamanlamasıyla ilgili daha fazla bilgi bulunmaktadır. Durum baytının nasıl yakalandığını gösteren devre, Bölüm 4.3'te verilmiştir.

Tablo 3.1 8080 durum baytı

Sembol	*Veri yolu biti*	*İşlevi*
INTA	D0	Kesme isteği için alındı sinyali
$\overline{WO}$	D1	Belleğe yazma ya da çıkış işlemi (yüksek olduğu zaman bellek okunmasını veya giriş işlemini gösterir)
STACK	D2	Adres yolunun yığın adresini taşıdığını gösterir
HALTA	D3	HALT komutu için alındı sinyali
OUT	D4	Çıkış işlemi
M1	D5	MİB'in komut getirme fazında olduğunu gösterir
INP	D6	Giriş işlemi
MEMR	D7	Bellekten okuma işlemi

Z8000 tarafından da bir G/Ç port yaklaşımının değişik bir uygulaması sağlanmıştır. 8080 gibi, Z8000'de de G/Ç komutunun ne zaman çalıştırıldığını gösteren özel olarak tanımlanmış bir IORQ hattı yoktur. Bunun yerine durum bilgisi, her bir komut saykılının belli zamanlarında, ST0, ST1, ST2 ve ST3 hatlarında ortaya çıkan bir kod ile oluşturulmaktadır. Tablo 3.2'de, farklı işlemler için ayrılmış kodlar gösterilmektedir. Bu tablodan, 0010 veya 0011 kodunun bir G/Ç işleminin gerçekleşmiş olduğunu gösterdiği anlaşılmaktadır. 8080'in durum baytı gibi, Z8000 durum kodu, ortaya çıktığı anda yakalanarak bir kaydedicinin içine konulabilir.

Z8000 ile G/Ç işlemlerinde bir başka sorun daha ortaya çıkar. Bölüm 2.2'den hatırlanacağı gibi, 16-bitlik veri formatı ve entegre devre paketinin pin sayısını sınırlama isteği, veri ve adreslerin çoğullanmasını gerektirir. Dolayısıyla bir Z8000 IN veya OUT komutu çalıştırıldığı zaman, komut saykılının bir kısmında, çoğullanmış yol, 16-bitlik port adresini taşır. Aslında tüm bir getirme-çalıştırma döngüsünü ele alırsak, çoğullanmış 16-hat yolunda, değişik zamanlarda, dört farklı tipte bilgi kendini gösterir. Bunlar;

1 Komutun adresi. Bu, komut getirme fazında, yol üzerinde görünür.
2 Komut saykılı
3 G/Ç birimine ayrılmış port adresi

4 Çevresel birimlerden başka bir yere ve başka bir yerden çevresel birim aktarılan veri

Tablo 3.2 Z8000 durum kodu

Kod				*İşlem tipi*
ST_0	ST_1	ST_2	ST_3	
0	0	0	0	Dahili işlem
0	0	0	1	Bellek tazeleme
0	0	1	0	Normal G/Ç işlemi
0	0	1	1	Özel G/Ç işlemi
0	1	0	0	Ayrılmış (rezerve)
0	1	0	1	Maskelenemez kesme alındısı
0	1	1	0	Vektörlenmemiş kesme alındısı
0	1	1	1	Vektörlenmiş kesme alındısı
1	0	0	0	İşlenen için bellek işlemi
1	0	0	1	Yığın için bellek işlemi
1	0	1	0	Ayrılmış
1	0	1	1	Ayrılmış
1	1	0	0	İkinci getirme için bellek işlemi
1	1	0	1	Birinci getirme için bellek işlemi
1	1	1	0	Ayrılmış
1	1	1	1	Ayrılmış

8080'deki gibi, sinyal çoğullanması, MİB'in dış devre yapısında ek sorunlara yol açar. Z8000'in, bir ölçüde normal dışı olan diğer bir özelliği, birbirine benzeyen G/Ç komut takımı içermesidir. Bundan dolayı da "normal" ve "özel" G/Ç işlemleri söz konusu olur. G/Ç işlemleri sırasında Z8000, yalnızca 8bitlik port adreslerine izin veren 8080 ve Z80'den farklı olarak, bir de 16-bitlik port adresi oluşturur. Dolayısıyla Z8000, iki G/Ç komut takımıyla toplam olarak 2^{16}=65.536 normal port ve eşit sayıda özel portu hizmete sunabilir. Z80 ve 8080 8-bitlik port adresleri ve tek bir G/Ç işlem takımıyla sınırlı oldukları için, bu mikroişlemcilerde kullanılabilen maksimum port sayısı 2^8=256'dır.

S3.10

Bellek-haritalı G/Ç

G/Ç olanaklarının direkt alternatifleri arasında en önde geleni, bellek-haritalı G/Ç sistemidir. Yalnızca bellek-haritalı G/Ç olanağı sunan mikroişlemci ailelerinden en önemlileri, M6800 ve 6502'dir. Teknik, mikroişlemcinin kendisinin tasarımı

giderek daha basitleştirme eğilimi göstermekte, fakat, çevresel arabirimler için gerekli olan dış devre düzeneğini daha karmaşık hale getirmektedir.

Yalnızca bellek-haritalı G/Ç olanağı sunan mikroişlemcilerin özelliklerinden biri, komut takımlarında özel amaçlı giriş ve çıkış komutlarının bulunmamasıdır. M6800'in Ek C'de bulunan komut takımını inceleyiniz. 8080 ve Z80'deki IN ve OUT komutlarının eşdeğerlerinin bulunmadığını göreceksiniz. M6800 durum hatlarının tanımları da incelenirse, IORQ gibi G/Ç işlemlerine özel herhangi bir sinyalin olmadığı da görülecektir. İlk bakışta M6800'in, veri giriş ve çıkışı için herhangi bir olanağının bulunmadığı görülebilir.

G/Ç olanaklarının bulunmayışı, bellek-haritalı G/Ç olanağı kullanan mikroişlemcilerin, giriş işlemleri ve bellek okuma işlemleri arasında herhangi bir ayrım yapmamaları gerçeğinden kaynaklanmaktadır. Benzer şekilde, veri çıkış işlemleri, aynen belleğe yazma işlemleri gibi ele alınırlar. M6800 ile ana belleğin arasında bağlantı kurmak amacıyla, durum bilgisini taşımak için, WMA, R/W ve 2 hatları kullanılır. Aynı hatlar, çevresel arabirim devreleri tarafından, G/Ç işlemleri sırasında, kullanılırlar. Bunun önemli bir sonucu, çevresel arabirim devrelerinin, MİB'e, aynen ana bellek hücreleri gibi davranacak şekilde tasarlanmaları gerektiğidir.

G/Ç ve bellek okuma/yazma işlemleri arasındaki bu benzerlik, donanımla sınırlı değildir. M6800'de veri, uygun konumun adresiyle birlikte, ana bellekten, LDAA veya LDAB komutları tarafından okunur. Veri giriş işlemleri için aynı komutlar kullanılır. Benzer şekilde, STAA ve STAB, hem belleğe yazma, hem de veri çıkışını gerçekleştirmek için kullanılırlar. Herhangi bir bellek referans komutunun, aynı zamanda G/Ç verisini işlemek için de kullanılabilmesi, bellek-haritalı sistemlerin özelliğidir.

Bellek-haritalı G/Ç kullanan sistemlerin arabirim devrelerinin tasarımını basitleştirmek için, özel-amaçlı entegre devreler bulunmaktadır. Bu devrelerden bir bölümü, M6800 ve 6502 mikroişlemci aileleriyle uyumludur. Paralel G/Ç işlemlere ilişkin arabirim devreleri, çevresel arabirim adaptörleri ya da PIA'lar şeklinde adlandırılırlar. Bunları Bölüm 4'te daha ayrıntılı olarak anlatacağız. G/Ç port ve bellek-haritalı sistemlerin donanım gereksinimlerini karşılaştırırken şöyle ifade söylenebilir: Örneğin Z80 mikroişlemcisi için bir paralel arabirim çoğu kez, basit bir yapıya sahiptir ve birkaç kapı ile birlikte bir 8-bitlik mandaldan oluşur, buna karşılık M6800'e ilişkin benzer bir arabirimde bir PIA bulunması daha muhtemeldir. Yine de, M6800 PIA'larından daha seyrek kullanılsalar da, Z80 ile uyumlu olan özel-amaçlı G/Ç arabirim entegre devreleri mevcuttur.

Bellek-haritalı G/Ç'ın, sistem adresi ve veri yollarına erişime izin veren he
bir mikroişlemcide uygulanabileceği, dikkatli bir okuyucunun gözünden kaçı
caktır. Bellek-haritalı G/Ç, 8080, Z80 ve Z8000 ile kullanılabilir. Fakat G/Ç
olanakları da mevcutsa, yaygın şekilde kullanılmaz. Bellek-haritalı G/Ç kull
dığında, G/Ç amaçları için ayrılan adreslerin ana bellek ile aynı adres al
paylaştığı unutulmamalıdır. Bir başka deyişle, bir G/Ç birimi için ayrılmış bir
S3.7 bellek konumunu için veya bir bellek konumu için ayrılmış bulunan bir ad
3.8 bir G/Ç birimi için kullanılamaz. Bu husus, mikroişlemci sisteminin ta
3.11 aşamasında bellek bölümleme konusuna gelindiğinde dikkate alınmalıdır.

3.2 G/Ç işlemlerinin senkronizasyonu

Bir bilgi işlem sisteminde, MİB ve çevresel birimler, normalde senkron (eşzam
olarak çalışırlar. Dolayısıyla, sözgelimi bir çıkış yazıcısının çalışma hızını d
dan doğruya MİB'in hızına bağlı olarak düzenleyen herhangi bir mekani
yoktur. MİB, her zaman için yazıcıdan çok daha hızlı çalıştığından, sistemde sü
eşzamanlı olarak çalışması arzu edilmeyecektir. Doğaldır ki, bilgisayarın, yazı
ilgili olmayan işlemlerle uğraşırken, eşzamanlı olarak çalışılması, sistemin ka
edilemeyecek ölçüde derecede yavaşlamasına neden olacaktır.

Eğer, MİB ve çevresel birimler arasında sürekli ve tam bir senkroizasyo
olmasının arzu edilmeyen bir durum olduğunu kabul ettiğimizde, karşımıza,
aktarma istendiğinde, sistemin bu iki birimi arasında olması gereken senkroniz
yon problemi çıkacaktır. Bu durum da, yine, bir yazıcı örneği üzerinde açıklanabil
Tek bir karakter kodu alan ve ona karşılık gelen karakteri basan çok basit
yazıcıyı ele alalım. Günümüzde yazıcıların çoğu, bu "her defada bir karakt
yaklaşımını kullanmamaktadır, fakat, biraz daha karmaşık tiplerde de bile senk
nizasyon problemleri mevcuttur.

Kendi örneğimizdeki yazıcının, maksimum 100 karakter/sn (cps) hızda yazabil
ğini varsayacağız. Burada bir senkronizasyon sorunu söz konusudur, çünkü, h
hangi bir MİB, karakter kodlarını yazıcıya, yazıcının basma hızından çok daha h
bir şekilde iletecektir. Örnek olarak 8080'i alırsak, aşağıdaki komutların çalışm
sıyla, "A" karakterinin kodu, port adresi FF'ye yerleştirilecektir.

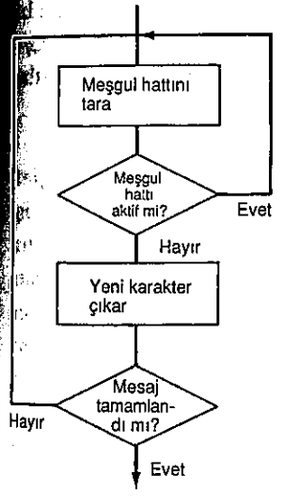
```
MVI  A, X'41'    ; A'yı, A'nın ASCII koduyla yükle
OUT  X'FF'       ; FF portuna çıkar
```

Bu iki komutun yürütülmesi, eğer 8080, 1 MHz'lik bir saat frekansında çalışıyorsa

toplam olarak 17 saat saykılı ya da 17 mikrosaniye sürecektir. Kolayca görülebileceği gibi, maksimum basım hızı 100 cps ise, karakter kodlarının yazıcıya, her 17 mikrosaniyede bir tane olacak şekilde gönderilmesi çok yararlı olmayacaktır. Senkronizasyon sorununun en önemli noktası budur.

Yukarıda anlatılan senkronizasyon problemini çözmenin iki yolu vardır. Bu yolların her ikisinde de, MİB ve çevresel birim arasında iki-yönlü bir iletişim kullanılır. Bunlardan ilkine meşgul tarama ya da onay diyeceğiz, bu yöntemde çevresel birimin, MİB'den veri almaya ya da MİB'e veri göndermeye hazır olmadığı zaman aktif hale geçen, bir "meşgul" durum sinyali üretmesi gereklidir. Dolayısıyla yukarıdaki örnekte anlatılan yazıcıda, bir karakter yazılırken aktif hale geçen bir meşgul hattı bulunacaktır. Yazma tamamlandığında meşgul hattı pasif duruma geçer, bu da yazıcının yeni bir karakteri kabul edebileceğini gösterir.

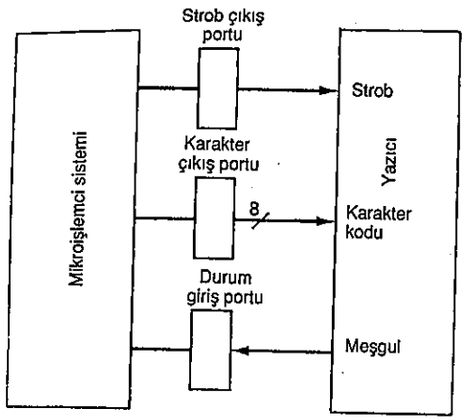
Onay senkronizasyonu tekniği uygulandığında MİB, bir G/Ç işlemini yapmadan önce, meşgul hattının durumunu kontrol eder. Şekil 3.3, uygulanan yöntemin bir akış diyagramını göstermektedir. Karakteri yazıcıya aktaran veri çıkış portuna ek olarak, meşgul sinyalini taşımak için bir de giriş portu hattı kullanılmaktadır. Şekil 3.4'te bunun genel bir şematik diyagramı görülmektedir. Şekil 3.3'te ayrıca bir de strob hattı gösterilmiştir. Bu hat yazıcıya, yeni bir karakter kodunun veri hatlarında mevcut olduğunu bildirmek üzere mikroişlemci tarafından üretilen bir darbe taşır.



Şekil 3.3 Yazıcı, anlaşma akış diyagramı

Kolay bir yöntem olan G/Ç onay yönteminde, MİB'in, yararlı işlemler yapmak yerine, meşgul hattının durumunu kontrol etmek için uzunca bir zaman harcaması gibi bir problem söz konusudur. Yukarıda anlatılan yazıcıyı göz önüne alırsak, her 10 msn'nin yalnızca 17 mikrosaniyesi çıkış verisi için ayrılmıştır. Zamanın geri kalan kısmı, mikroişlemci tarafından, sürekli olarak, meşgul hattının durumunu kontrol etmek için kullanılmaktadır.

Çoğu kez, yukarıda sözü edilen "boşa harcanmış" zaman süresince yapılabilecek herhangi yararlı bir işlem yoksa, bunun



Şekil 3.4 Yazıcı arabirim hatlarının şematik diyagramı

önemi yoktur. Böyle durumlarda onay yöntemi hemen hemen her zaman al
natif yöntemelerden daha basit ve daha ekonomik şekilde uygulanabilen bir tekn
tir.

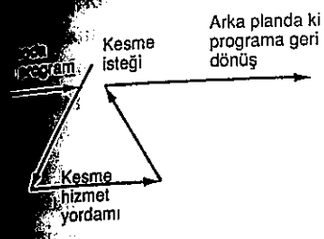
Basit onay (handshaking) yöntemiyle senkronizasyon, yalnızca, meşgul durum
nun kontrol edilmesine harcanan zaman süresince başka işlerin yapılmasına ol
gereksinimin zorlayıcı olduğu durumlarda, bir mecburiyet olmaktadır. Bu duru
özellikle, MİB ile bağlantılı çalışan birkaç giriş ve çıkış biriminin bulunduğu b
sistem için söz konusudur. Böyle bir durumda, program tarafından başlatılan G/
elsıkışma uygulamasını bırakıp, çevresel birimlerin kendileri tarafından başlatıla
giriş ve çıkış işlemlerine geçmek gerekir.

Bir çevresel birimin MİB'in dikkatini kendine çevirmesi ve bir G/Ç işlemi başla
ması için, MİB'in çalışmasının kesilebilmesi gerekir. Kesme olanakları, mikroiş
lemciden mikroişlemciye büyük farklılıklar gösterir. Ancak temel ilke ortaktı
çevresel birim, halen çalışmakta olan programın askıya alınması, ve ardından
kesmeyi yapan birimden veri alan ya da bu birime veri getiren bir programa atlama
yapılmasıyla sonuçlanan bir olaylar dizisini başlatabilir. Yukarıda anlattığımız
yazıcıya tekrar dönersek, meşgul hattı bu durumda MİB tarafından taranmayacak
ancak bunun yerine, her bir yazma işlemi sonunda bir kesme yaratmak için
S3.1 kullanılabilecektir. MİB, yapmakta olduğu işi askıya alacak, yazıcıya yeni bir
3.2 karakter kodu gönderecek ve daha sonra da önceki işine dönecektir.

3.3 Kesmeler

Kesmeler temelde, MİB'in içinde çalışmakta olan programın düzgün bir şekilde askıya alınabilmesi ve denetimin, kesme hizmet yordamı adı verilen bir başka programa aktarılmasını sağlarlar. Yürürlükteki program o şekilde askıya alınır ki, daha sonraki bir aşamada, mümkün olan minimum kesintiyle ile yeniden bu programa dönülebilir. Kesme, işlemci kesme hattı etkinleştirildiğinde başlar. M6800, Z80 ve Z8000 mikroişlemcilerinde bunun için, kesme hattı alçak duruma getirilir. 8080 mikroişlemcisinde kesme hattı, yüksekte-etkindir.

Kesmenin, gerçek-zamanlı çalışma şekli hakkında bilgi edinebilmek için, bir binanın içindeki ortamı izlemek ve kontrol etmek amacıyla kullanılan bir mikroişlemciyi ele alalım. Mikroişlemcinin yaptığı iş, her odadaki sıcaklık transdüserinin sayısallaştırılmış (digitized) çıkışlarını kontrol etmeyi ve ısı girişini, sıcaklık sabit kalacak şekilde ayarlamayla ilgili, yordam tarzında ya da arka plandaki görev şeklinde bir iştir. Her oda sırayla gözden geçirilir. Sıcaklık dönüştürücüsüne ek olarak her oda, çıkışı, herhangi bir duman görüldüğünde bir kesmeyi başlatan bir duman dedektörü ile donatılmıştır. Dolayısıyla, bir kesme ortaya çıktığında denetim, arka plandaki programdan, yağmurlama sistemini ve sesli alarmı çalıştıran bir programa aktarılır. Bunlar yerine getirildikten sonra bir sıfırlama anahtarı mikroişlemcinin arka plandaki programa dönmesini sağlar. Şekil 3.5'te program akışı sembolik olarak gösterilmektedir.



Şekil 3.5 Kesme cevabı

Bu örnek, kesmelere çok çeşitli kaynakların neden olabildiği durumu açıklamaktadır. Bir kesme durumunda işlemci, doğru odadaki yağmurlama vanalarını açmak için kesmenin kaynağını belirlemeye yarayacak bir takım araçlara sahip olmalıdır. Bu ve diğer etkenlerin kesme olanaklarındaki önemini açıklamak için birkaç tanımlamaya gerek vardır.

Maskelenebilen ve maskelenemeyen kesmeler Bazen bir kesme olanağını "kapatabilmenin" avantajları vardır. Maskelenebilen bir kesme hattı, programda uygun bir yere özel bir komut eklenmesiyle bize bu imkanı verir. Maskelenemeyen kesmeler

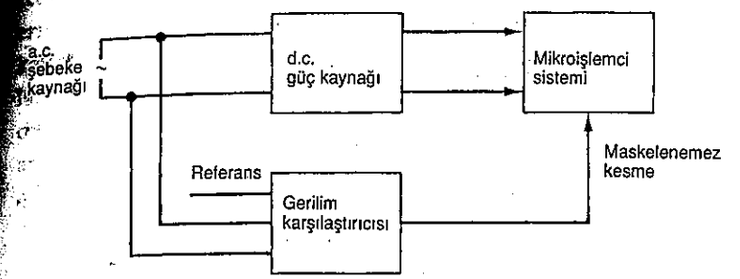
asla kapatılamazlar.

Vektörlenmiş ve vektörlenmemiş kesmeler Çok sayıda kesme durumunda, g
likle, kesmenin kaynağını belirleyebilmek önemlidir. Vektörlenmiş bir k
sisteminde kesmeyi oluşturan birim, aynı zamanda, MİB'e, tanımlayıcı bir k
da adres sağlar. Bu, MİB'in, uygun kesme hizmeti yordamına mümkün olan en
zamanda girmesine olanak tanır. Vektörlenmemiş kesmelerde MİB, kesmeyi o
çıkaran birimi tesbit edinceye kadar kesme üretmesi muhtemel her birimi t
S3.5 teker taramalıdır.

Kesme kullanan birçok mikroişlemcide, daha önce söylediğimiz gibi, zaman za
kesme olanağını kapatmak veya yetkisizleştirmek uygun olur. Ancak öyle durum
da vardır ki, kesmenin yetkisizleştirilmesi oldukça tehlikeli olabilir. Bunlar, işle
cinin hemen haberdar olması gereken son derece önemli olaylara ilişkin kesmel
dir. Mikroişlemcinin kontrolündeki bir kimyasal işlem sisteminde, fabrika
infilak etmek üzere olduğunu belirten bir kesme, buna iyi bir örnek olarak alınabil
Bu gibi çok önemli olaylar, maskelenemeyen kesmeleri oluşturmada kullanılır
Maskelenemeyen kesme oluşturmak için kuvvetli gerekçeler bulunmadıkça, mü
kün olan her yerde maskelenebilen kesmeler kullanılmalıdır.

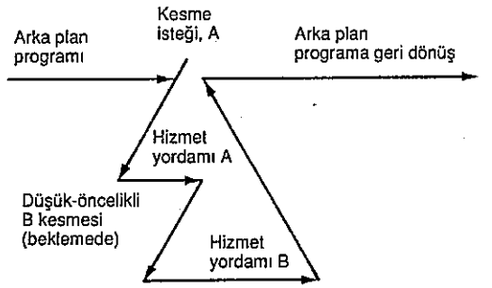
Maskelenemeyen kesmeler, bilgisayar sistemlerinin güç kaynağı arızalarında
korunmasında özellikle önemlidir. Güç arızası, şebeke geriliminin düşmesi
neden olduğunda, devrenin güç kaynağı içindeki saklama kondansatörleri, düş
gerilimi gerekli minimum düzeyin üzerinde tutarak sınırlı bir süre için mikroişlem
sistemi besler. Saklama kondansatörünün yeterli ölçüde büyük olması koşuluyl
bu, mikroişlemcinin kontrolündeki sistemin istenen sıra içinde kapatılmasına yet
cek kadar zaman kazandırabilir. Mikroişlemci, kapatma işlemini gerçekleştirme
için, enerji, belli bir kritik değerin altına düşünceye kadar komutları çalıştırma
sürdürür. Doğru sırada gerçekleşmeyen kapatma işlemi, büyük makineler ve
kimyasal fabrikalar gibi sistemlerde önemli güvenlik riskleri ortaya çıkarabil
Yukarıda anlatılan şekilde bir enerji arızası kesme olanağının uygulanması oldukç
kolaydır. Şekil 3.6, bir gerilim karşılaştırıcının ana güç girişini nasıl izlediğin
göstermektedir. Bu gerilim, referans düzeyin altına düştüğünde maskelenemeyen
bir kesme oluşturulur.

Enerji arızası kesmeleriyle, bazı kesmelerin diğerlerinden daha önemli olabilece
ğini anlatmış olduk. Bir sistemde birkaç kesme kaynağı bulunduğunda, bunu
kesme oluşturan her bir birime bir öncelik sıra numarası vermek şeklinde genişle
tebiliriz. Çoklu kesme durumunda, çakışan hizmet talepleri sorunu ortaya çıkar. İk
S3.3 kesmenin tam olarak aynı anda oluşması pek olası değildir fakat, bir kaynaktan
3.4



Şekil 3.6 Enerji arızası kesmesi

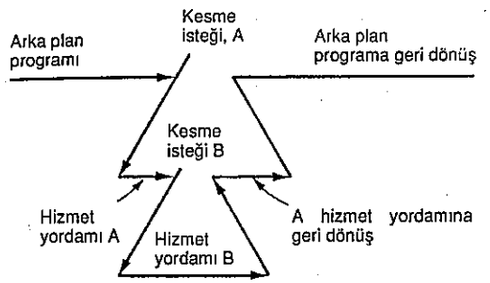
gelen kesme için gerekenin yapılmakta olduğu anda başka bir kaynaktan yeni bir kesme gelebilir. Önceliklerin belirlenmesi, yeni kesmenin, öncelik durumuna göre ya o anda hizmete girmesine ya da ertelenmesine olanak sağlar.



Şekil 3.7 Yüksek öncelikli bir kesmeye hizmet verilirken ortaya çıkan daha düşük öncelikli bir kesme

Şekil 3.7'de, daha düşük öncelikli kesmenin, daha yüksek öncelikli bir kesme yordamı sırasında ortaya çıkması durumunda program akışını göstermektedir. Sistem, yüksek öncelikli kesme yordamı tamamlanıncaya kadar, düşük öncelikli kesme için herhangi bir işlem yapmaz. Bir başka alternatif durum Şekil 3.8'de görülmektedir. Burada, düşük öncelikli kesme yordamı, daha yüksek öncelikli bir kesmenin ortaya çıkmasıyla askıya alınmıştır.

Kesmelere önceliklerin belirlenmesi (maskelenemeyen kesmelerin özel durumu hariç), normalde mikroişlemcilerin kendi içinde mevcut olan bir olanak değildir. Bazı uygulamalarda gerekli bir özelliktir ve bu nedenle de, bir dış devre düzenlemesiyle mikroişlemciye sağlanması gerekir. Bazı mikroişlemci aileleri, içinde, kesme öncelik belirleme sistemi sağlayan birimler bulunmaktadır.



Şekil 3.8 Düşük öncelikli bir kesmeye hizmet verilirken ortaya çıkan daha yüksek öncelikli bir kesme

M6800 kesme olanakları

M6800, burada anlatılan mikroişlemciler arasında, doğrudan kesme olanakla sahip olanıdır. Ek C'deki bacak tanımlarında maskelenebilen ve maskelenemey kesme hatları gösterilmektedir. Bu hatlar, sırasıyla $\overline{IRQ}$ ve $\overline{NMI}$ kısaltmaslarıl belirtilmiştir ve alçakta-etkindirler. Maskelenebilir kesmeyi iş yapabilir durı getirmek için bir "kesme maskesini sıfırla" komutu CLI çalıştırılmalıdır. "Kes maskesi kur" komutu, SEI, çalıştırıldığında da kesme yetkisiz duruma gelir. komutlar aynı zamanda, koşul kodu ya da bayrak kaydedicisindeki kesme bitin kurulması veya sıfırlanmasına neden olurlar. Bu bitin durumu, maskelenemey kesmenin yetkili hale getirilip getirilmediğini tespit etmek için kullanılabilir.

İşlemci tarafından bir programın yürütülmekte olduğunu ve kesme maskesin kurulmadığını varsayarsak, $\overline{IRQ}$ alçak düzeye çekildiğinde, aşağıdaki olaylar diz meydana gelir:

1 Yürürlükteki komutun yürütülmesi tamamlanır. Bundan sonra gelen kesmel etkisiz hale getirilir.
2 İndeks kaydedicisi, program sayıcısı, A kaydedicisi, B kaydedicisi ve koş kodu kaydedicisi içerikleri yığın içinde saklanır.
3 Kesme vektöründe bulunan adrese bir program atlaması gerçekleşir.

Kesme vektörü, kesme hizmet yordamının ilk komutunun adresini tutmak iç kullanılan, özel olarak ayrılmış bellek konumu çifti FFF8 ve FFF9'dur. Kesm oluşmadan önce kesme vektörünün doğru adresi içermesini sağlamak, programc nın sorumluluğundadır. Eğer hizmet yordamı 1000'de başlıyorsa programc FFF8'in 10, FFF9'un da 00 ile yüklü olmasını sağlamalıdır.

Kesme hizmet yordamının normalde, bir "kesmeden dönüş" komutu RTI ile sona ermesi gerekir. Bu komutun yürütümü, indeks kaydedicisi, program sayıcısı, A kaydedicisi, B kaydedicisi ve koşul kodu kaydedicisi içeriklerinin yığından çıkarılan veri ile yüklenmesini sağlar. Bu ise çalışan kaydedicilerin hepsinin, kesme ortaya çıktığında sahip oldukları değerleri yeniden almalarını sağlar. Dolayısıyla, arka plandaki programda, kesmeden önce hangi komutta kalındıysa o komuta bir atlama gerçekleşir. Kesmeler ayrıca, otomatik olarak da, yeniden yetkili hale döndürülebilirler.

M6800'ün maskelenemeyen bir kesmeye cevabı, aynen maskelenebilen kesmeye olduğu gibidir, fakat kesme vektörü farklıdır. Bu durumda kesme hizmet yordamının ilk komutunun adresi, FFFC ve FFFD adreslerinde saklanmalıdır. Bir kesme olarak tanımlanmamış olmasına karşın $\overline{SIFIRLAMA}$ (RESET) sinyali de aynı şekilde çalışır. $\overline{SIFIRLAMA}$ (RESET), alçak düzeye çekildiğinde, sıfırlama vektörünün FFFE ve FFFF'de saklanan bir adresine, program atlaması ortaya çıkar. Buradaki esas fark, çalışan kaydedicilerin içeriklerinin, maskelenebilen ve maskelenemeyen kesmelerdeki gibi yığında saklanmamasıdır.

Yazılım kesmesi sağlaması M6800'ün ilginç bir özelliğidir. M6800 komut takımında, "yazılım kesmesi" komutu SWI bulunur. Bu komut çalıştırıldığında işlemci aynen, bir kesme ortaya çıkmış gibi hareket eder. Bu durumda yazılım kesme vektörü FFFA, FFFB'dir. Yazılım kesmelerinin bir çok kullanımı bulunmaktadır, bunlardan birisi programın kolay bir şekilde sona erdirilmesidir. Okuyuculardan bir kısmı, SWI komutunu bu amaçla kullanan Motorola MEK6800D2 değerlendirme sistemini tanımaktadır. Yazılım kesme vektöründe sistem monitörünün (izleyicisinin) giriş noktası JBUG saklanır. Kullanıcı, programı, bir yazılım kesmesi komutu SWI aracılığıyla sona erdirir. Bu komut çalıştırıldığında sistem monitör programı devreye girer ve kullanıcının, programın istenen şekilde çalıştırıldığını kontrol etmesi için gerekli olanaklar sağlanır.

M6800'ün kesme olanaklarının temel özelliği basitliktir. Tek kaynaktan çıkan bir maskelenebilen ya da maskelenemeyen kesme, mikroişlemcinin harici bir devre düzenlemesi olmaksızın ya da küçük bir devre düzenlemesiyle gerçekleştirilebilir. Ancak işlemcinin çoklu kesme durumunda kesme vektörlemesi için bir yardım sağlamadığı da unutulmamalıdır. Bu tamamen, bir yazılım ve harici donanım kombinasyonuyla yapılmalıdır. Burada anlatılan diğer mikroişlemcilerde olduğu gibi M6800 de, çoklu kesme durumlarında herhangi bir dahili öncelik belirleme olanağı sağlamamaktadır.

8080 kesme olanakları

8080 ile kesme yönetimi 6800'dekinden farklı olarak, zevkli bir uğraştır. 8 maskelenebilir tek bir kesme olanağını destekler. Kesme yönetimi ile ilişkil sinyal, aşağıdaki gibi tanımlanırlar:

INT Yüksekte-etkin kesme istek girişi
INTE Yüksekte-etkin kesme etkinleştirme çıkışı
INTA Çoğullanmış veri/durum yolunda ortaya çıkan bir kesme alındı sinyal

8080 içinde INT kesme hattının durumu her komut saykılının son saat dar sırasında örneklenir. Eğer kesme devreye girmişse, bu, bir iç mandalın kurulma neden olur, fakat yürürlükteki komutla ilgili işlemler tamamlanana kadar işle devam edilir. Kesme mandalının kurulması, bir sonraki normal komutun yerinl kesme saykılının almasına neden olur. Ek A'da gösterilen kesme zamanla diyagramından, D0 veri/durum yol hattında, kesme saykılının T1 ve T2 aralıkl sırasında SYNC sinyaliyle eşzamanlı olarak INTA kesme alındı sinyalinin ort çıktığı görülmektedir. Kesmeyi oluşturan G/Ç biriminin arabirim devresi, a INTA sinyalini araştırmalı ve veri/durum yolunda T3 saat aralığı sırasında komut kodu ortaya çıkmasını zorlamalıdır. T3 sırasında eşzamanlı DBIN siny aktif hale gelir ve bu, çevresel arabirim devresi tarafından da kullanılabilir.

Kesme alındısı sırasında veri yoluna sıkıştırılan kod genellikle YENİDEN BA LAT (RESTART) komutu, yani RST içindir. 8080 kesme cevabında çok öne olduğundan bu komutu biraz ayrıntılı olarak anlatacağız. 8080 komut takımı se "yeniden başlat" komutu içerir, bunlar RST0-RST7'dir. Bu komutların her bi çalıştırıldığında program sayacı içeriğinin yığın içinde saklanmasına neden olu bir atlama gerçekleştirir. Tablo 3.3'te gösterildiği gibi, her albaştan komutu için tane olmak üzere sekiz atlama varış yeri tanımlanmıştır. Tablodan da anlaşılac gibi, eğer bir çevresel arabirim, kesme oluşturur ve kesme alındısına cevap ola da veri/durum yoluna, örneğin "DF" yüklerse, program sayacının içeriği yığın saklanacak ve 0018'e bir atlama gerçekleşecektir. 0018 konumuna kesme hizm yordamının ilk komutu veya daha sık karşılaşıldığı gibi belleğin bir başka bölg sinde bulunan bir yordama atlama komutu yerleştirilebilir. Dönüş adresi yığın saklandığından, "altyordam dönüş" komutu RET, tıpkı bir altyordam çalıştırılıyo muş gibi, ana programa dönülmesine izin verir.

Okuyucu 8080'in, 6800'den farklı şekilde, kesmeye verdiği yanıtın bir parça olarak, çalışan tüm kaydedicilerin içeriklerini otomatik şekilde saklamadığı farkedecektir. 8080 yalnızca, program sayıcının içeriğini saklar. Bu, kesme hizme

	Kod	Atlama varış yeri
RST0	C7	0000
RST1	CF	0008
RST2	D7	0010
RST3	DF	0018
RST4	E7	0020
RST5	EF	0028
RST6	F7	0030
RST7	FF	0038

yordamının ilk birkaç komutunun, kaydedicilerin içeriklerini yığın üzerine ya da bellekte özel olarak ayrılmış konumlara saklamakla ilgili olacağı anlamına gelir. Kaydediciler, hizmet yordamı sonunda ilk durumlarına getirilmelidir.

8080 bir kesme alındısı bildirdiğinde, çevresel arabirimin bir yeniden başlatma komutu sağlamasının gerekli olması, arabirim devresinin karmaşıklığını daha da arttırır. Ancak, örneğin 6800 ile karşılaştırıldığında 8080'de, sınırlı vektörlenmiş kesme sisteminin daha kolay gerçekleştirilebilmesi bir avantajdır. En çok sekiz kesme biriminin her biri için farklı bir yeniden başlatma komutu ayrılabilir. Bir arabirimin veri/durum yoluna yerleştirdiği kod, işlemciyi daha sonra uygun hizmet yordamına yönlendirecektir. İki kesme biriminin, farklı kodları aynı anda yola çıkarmaya çalışmalarını engellemek için, gerekli
S3.6 önlemler alınmalıdır.

Z80 kesme olanakları

M6800 ve 8080 ile karşılaştırıldığında Z80, çok daha kapsamlı kesme olanakları sunmaktadır. İki kesme hattı vardır. Bu hatlar, maskelenebilen ve maskelenemeyen kesmeler için sırasıyla $\overline{INT}$ ve $\overline{NMI}$ hatları şeklinde adlandırılır, her ikisi de yüksekte etkindir. $\overline{NMI}$ hattı alçak düzeye getirildiğinde aşağıdaki olaylar dizisi gelişir:

1 Yürürlükteki komutun yürütümü tamamlanır.
2 Program sayıcısının içeriği yığına itilir.
3 0066 komut konumuna atlama gerçekleştirilir.

Maskelenebilir kesmeye Z80 yanıtı üç olası modun birinde olabilir, 0, 1, veya 2. IM0, IM1 ve IM2 komutları çalıştırıldıklarında yanıt modunu sırasıyla 0, 1 ve 2'ye kurarlar. Bu nedenle, sistemi kullanıma hazırlama programına uygun komutu ekleyerek kesme cevap modunu doğru saptamak programcının sorumluluğundadır. Sıfırlama sonunda Z80, daha önceki durumu ne olursa olsun 0 modunda olacaktır.

Z80, 0 (sıfır) modundayken kesme yanıtı 8080'deki yanıta benzer. Kesme yapan birim, çalıştırılacak bir sonraki komut

için, veri yoluna kod çıkarmalıdır. Kolay anlatımla, 0 modunda 8080 arasındaki temel farklılık, Z80 veri yolunun, aynı zamanda, 8080 INT çoğullanmış veri sinyallerini de taşımamasıdır. 8080 gibi Z80 de, her saykılının son iki saat saykılı sırasında, maskelenebilen kesme hattı $\overline{INT}$ düzeyini örnekler. Komutun tamamlanmasından sonra, eğer $\overline{IRQ}$ alçak d bulunduysa işlemci bir kesme saykılı başlatır. Normal saykıla oranla kesme lının ayırıcı özelliği, $\overline{IORQ}$ ve $\overline{M1}$'in aynı zamanda etkinleşmesi yani alçak inmesidir. Kesme saykılı yalnızca bu durumda oluşur. Çevresel arabirim iç $\overline{IORQ}$ ve $\overline{M1}$'in alçak düzeye aynı anda inmiş olduklarının saptanması, 808 sinyaline eşdeğerdeki kesme alındısını ortaya çıkarır.

Mod 1, mod 0'a göre biraz daha basittir. Mod 1 durumunda maskelenebilen yanıtı, maskelenemeyen kesme yanıtına oldukça yakındır. Bu modda $\overline{IRQ}$ düzeye çekildiğinde, aşağıdaki olaylar dizisi ortaya çıkar:

1 Yürürlükteki komutun çalışması tamamlanır.
2 Program sayıcısının yürürlükteki içeriği yığında saklanır.
3 0038 konumuna atlama yapılması sağlanır.

0038 konumu maskelenemeyen kesme hizmet yordamının ilk komutunu, bir kodunu ya da hizmet yordamı çağrısını içerebilir. Maskelenemeyen kesme ile mod 1 arasındaki tek fark atlama komutunun varış adresindedir. Eğer kesme vektörlenmesi gerekmiyorsa, mod 1, en basit kesme arabirim devresinin k mına izin verir.

Vektörlenmiş kesme sistemi gerekli olduğunda mod 2, Z80'e, 8080 ve M kıyasla çok güçlü bir özellik kazandırır. Bu modu kullanmak için belleğe bir tablosu saklanmalıdır. Tüm kesme birimleri için hizmet yordamı adresl tabloda yer alır. Mod 2'deyken IRQ hattı alçak düzeye çekildiğinde aş olaylar dizisi başlar:

1 Yürürlükteki komutun çalışması tamamlanır.
2 İşlemci 16-bitlik bir adres işaretçisi oluşturur. En büyük değerlikli sekiz 1 kaydediciden alınır. En küçük değerlikli sekiz bit ise kesme birimi tara yerleştirildikleri veri yolundan alınır. Aslında kesme biriminin yalnızca sağlaması yeterlidir, çünkü işlemci, en küçük değerlikli biti her zaman 0 etmektedir.
3 Program sayıcısının yürürlükteki içeriği yığına itilir.
4 Bir atlama komutu çalıştırılır. Atlamanın varış adresi bellekten, 2'de oluş işaretçinin verdiği adresten kurtarılır (kurtarma yoluyla sağlanır-çev.).

Yukarıda anlatılanlar size çok karmaşık görünebilir. Fakat bunlara ilişkin bazı genel noktalar belirleyebiliriz. Vektörlenmiş kesmelerin hem donanım hem de yazılımda sorunu yol açmaları bunlardan biridir. Donanımla ilgili sorunlar kesme biriminin çevresel arabiriminde ortaya çıkar. Kesme alındısını oluşturan $\overline{M1}$ ve $\overline{IRQ}$'nun aynı anda alçak mantık düzeyinde olduklarını belirlemek ve vektör işaretçi adresinin en küçük değerlikli kısmını veri yoluna aktarmak için, devre gerekir. Yazılım sorunlarını bellekte, kesme hizmet yordamı arama tablosunun bulunmasını sağlamak programcının sorumluluğunda olduğu için ortaya çıkar. Kesme kaydedicisini, arama tablosu işaretçisinin en büyük değerlikli baytı ile yüklemek te programcının sorumluluğundadır. LDI,A komutu bu amaçla bulunmaktadır. Tüm Z80 kesmelerinde olduğu gibi, kesme yanıtı, çalışma kaydedicilerinin içeriklerini otomatik olarak saklama işlevini içermez, bu da programcının sorumluluğundadır.

Z8000 kesme cevabı

Burada ele alınanlar arasındaki en güçlü ve en yeni geliştirilen mikroişlemci olma özelliğine uygun olarak Z8000, en kapsamlı kesme olanaklarını da sunmaktadır. Bu olanakların tam anlamıyla açıklanabilmesi bir kitabın bir bölümünün bu konuya ayrılmasını gerektirecektir. Bu nedenle Z8000'in kesme olanaklarını yalnızca, diğer mikroişlemcilerle anlamlı bir karşılaştırma yapabilmeye yetecek düzeyde anlatacağız.

Z8000'in kesme yönetim biçimini anlatmak için, işlemci mimarisi ve komut takımına ilişkin bazı temel bilgilerin bulunması gerekir. Program durum alanı kavramının, kesme sistem yönetimi için çok büyük önemi vardır. Bu alan bellekte programcı tarafından korunan bir alandır ve Z80'in mod 2'de kullandığı kesme vektör tablosu ile bazı benzerlikleri bulunmaktadır. Ancak Z8000'de program durum alanı, kesme vektör adreslerine ek olarak başka bilgiler de saklar.

Z8000 program durum alanının konumu, programcı tarafından, sistemin kullanıma hazırlanması yordamı yazılırken belirlenir.

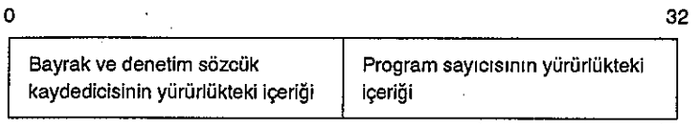
Yordam, aşağıda gösterildiği gibi, program durum alanı işaretçi kaydedicisi PSAP'ı yükleyen komutlar içerir:

```
LD      R2, #%1000
LDCTL   PSAP, R2
```

İlk komut 16-bitlik kaydedici R2'yi 1000 (onaltılı) ile yükler. Bu sayı daha sonra LDCTL komutu tarafından PSAP kaydedicisine aktarılır. Yukarıdaki iki komutun

çalıştırılması program durum alanının 1000 konumundan başlamasına neden

Z8000 kesmelerinin anlaşılabilmesindeki diğer önemli bir kavram, progra yürütümü sırasında herhangi bir anda tanımlanabilen ve Şekil 3.9'da göste işlemci durum sözcüğüdür. Bu, 32-bitlik bir sözcüktür, ilk 16-bitlik kısmı ba ve denetim sözcük kaydedicisi FCW'nun yürürlükteki içeriklerinden oluşur. Z8 FCW kaydedicisi, M6800'deki koşul kodu kaydedicisi CCR veya Z80/8080 ba kaydedicisiyle karşılaştırılabilir. FCW kaydedicisinin çeşitli bitlerinin işlevl D'de verilmiştir. İşlemci durum sözcüğünün ikinci 16-bitlik kısmı program say nın yürürlükteki içeriğinden oluşur.



Şekil 3.9 Z8000 program durum sözcüğü

Bölüm 2.3'ten hatırlanacağı gibi, Z8000 ailesi kitap yazıldığı sırada iki üye oluşmaktaydı, Z8001 ve Z8002. Kesme olanaklarını burada, daha basit olan Z80 ile inceleyeceğiz. Donanım kesmelerine ek olarak Z80, M6800'in SWI olanağı benzer şekilde yazılım kesmeleri de sağlamaktadır. Sağladığı bir diğer kesi olanağı ise yazılım tuzağı adı verilen kesme biçimidir. Z8002'nin tüm donanım kesme çeşitlerine, yazılımsal kesmeye ve yazılımsal tuzağa verdiği yanıtlar bir rine benzemektedir. Bu mikroişlemciyle kullanılabilir durumda olan kesme olana ları aşağıdaki listede özetlenmiştir:

1 Maskelenebilen, vektörlenmemiş donanım kesmesi. $\overline{NVI}$ hattı alçakta-etkind
2 Maskelenebilen, vektörlenmiş donanım kesmesi. $\overline{VI}$ hattı alçaktaetkindir.
3 Maskelenemeyen vektörlenmemiş donanım kesmesi. $\overline{NMI}$ hattı alçakta-etki dir.
4 "Sistem çağrı" komutu SC'nin yürütümü ile başlatılan yazılımsal kesme.
5 Z8002'nin izin vermediği durumlarda, belli bazı komutların yürütümleri iç girişimde bulunulmasıyla başlayan yazılımsal tuzaklar. Yazılımsal tuzak ve kesmeler, maskelenebilen donanım kesmelerinden daha yüksek önceliğe sahip tir.

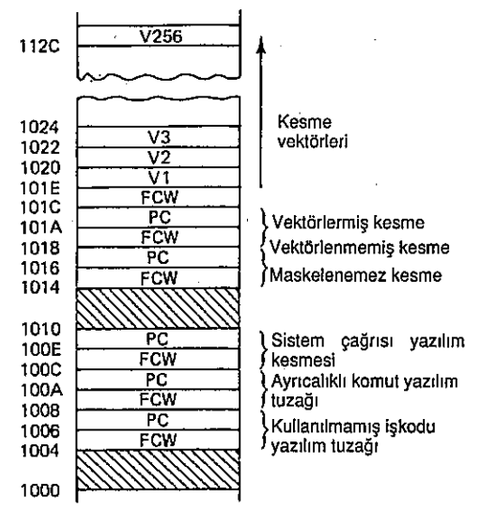
Z8000 kesme ya da tuzağı oluştuğunda, aşağıdaki olaylar dizisi ortaya çıkar:

a Yürürlükteki komutun çalışması tamamlanır.
b İşlemci bir kesme alındı saykılına girer. Bu saykıl sırasında kesme birim işlemciye, 16-bitlik bir kesme tanıtıcı sözcük göndermelidir.

c Kesme tanıtıcısı yığına itilir ve bunu, FCW kaydedicisi ve program sayıcısının içeriği olan işlemci durum sözcüğü izler.
d Program durum alanından, yeni bir işlemci durum sözcüğü oluşturulur ve FCW kaydedicisiyle program sayıcısına yüklenir. Bu, program sayıcının içerdiği adresteki komuta, otomatik bir atlamanın yapılmasına neden olur.

Program durum alanını programcı doldurduğu için, yukarıdaki olaylar dizisinden, program atlaması varış adresinin sistemin kullanıma hazırlanması sırasında belirlendiği anlaşılmaktadır. Çeşitli donanımsal ve yazılımsal kesme tiplerinin birbirinden ayırdedilmesi gerektiğinden, program durum alanlarının farklı bölgeleri bu değişik olay tipleri için ayrılmıştır. Program durum alanı işaretçi kaydedicisinin, yukarıda anlatıldığı gibi 1000'den başlatıldığının varsayılması sonucunda elde edilen program durum alanı haritası, Şekil 3.10'da gösterilmektedir. Haritayı yorumlarken, 16-bitlik bir mikroişlemci olmasına karşın Z8002'de belleğin, 8-bitlik baytlar biçiminde düzenlendiğini hatırlayınız. Dolayısıyla, örneğin 16-bitlik FCW kaydedicisi içeriğinin saklanabilmesi için, iki bellek konumu gerekmektedir.

Program durum alanının işlevini açıklığa kavuşturmak için özel bir örnek üzerinde duracağız. Bir Z8002 sisteminin program durum alanı işaretçisinin içeriği=1000 olacak biçiminde kullanıma hazırlandığını varsayalım. Bu hazırlama sırasında bellek konumları 1016 ve 1017, 0000 ile, 1018 ve 1019 konumları da F800 ile yüklenir. Şekil 3.10'dan, bu konumların, program durum alanının vektörlenmemiş kesmelerle ilişkili parçası olduğu anlaşılmaktadır.



Şekil 3.10 (PSAP)=1000 için Z8002 program durum alanı

Şimdi bir programın çalışmakta olduğunu ve Z8002'nin bu anda, 4bayt uzunlu komut kodunun 2000-2003 adreslerinde saklandığı şu komutu çalıştırdığını d nelim:

Komut kısaltması	*Kod*
LD R4, #%0FFF	21, 04, 0F, FF

Eğer $\overline{NVI}$ hattı, bu komutun yürütülmesi sırasında alçak düzeye çekilirse, vek lenmemiş kesme yanıtı başlar. Önce yürütülmekte olan komut tamamlanır. Z8 daha sonra kesme alındı saykılına girer. Bu durum, kesmeyi oluşturan çevr birim tarafından saptanabilir, çünkü dört Z8002 durum hattı ST3-ST0, Tablo 3.2 anlatıldığı gibi vektörlenmemiş kesme alındı kodu 0110'ı taşımaktadır. Du kodu saptanırken çevresel arabirim, çoğullanmış adres/veri yoluna 16-bitlik kesme tanıtıcısı yerleştirir. Maskelenemeyen kesme durumunda bu tanıtıcı işlevi kullanıcı tarafından belirlenebilir. Tanıtıcı, örneğin, kesmenin kaynağ adlandıran bir kod içerebilir. Ya da, kesme hizmet yordamı sırasında kullanılan sayısal veriden oluşabilir.

Kesme tanıtıcısı alındıktan sonra işlemci tarafından yığına aktarılır. İşlemci duru sözcüğü; FCW kaydedicisinin ve program sayıcısının yürürlükteki içeriklerind oluşmaktadır. Tabi program sayıcısının içeriği bu aşamada 2004'tür. İşlemci duru sözcüğü de yığın içinde saklanmaktadır. Ardından, 1016-1019 adreslerinden ye bir işlemci durum sözcüğü üretilir. Hatırlanacağı gibi bu adresler, kullanım hazırlama sırasında 0000,F800 ile yüklenmiştir. Dolayısıyla FCW kaydedicisi 000 ile, ve, program sayıcısı da F800 ile yüklenir. Bu nedenle F800 konumuna otomati olarak bir program atlaması gerçekleşir. Diğer çalışma kaydedicilerinin içerikler nin saklanmasının istenmesi durumunda bunun, F800'de başlayan kesme hizm yordamının başlangıç aşamasında yapılması gerektiğine dikkat ediniz.

Her ne kadar kullanıcı yukarıdaki örnekte, kesme tanıtıcısını dilediği biçimd kullanabilirse de, bu her zaman geçerli değildir. Diğer kesme tiplerinde tanıtıcı özel olarak belirtilmiş işlevlere sahiptir. Vektörlenmiş kesmede tanıtıcı, işlemciy kesme biriminin hizmet yordamına yönlendirmek için gerekli bilginin bir kısmını sağlar. PSAP'ın daha önce olduğu gibi 1000 ile başlatıldığı bir sistemde, kesme yanıtı işlem sırasının sonunda Z8002, program sayıcısına 101E+(2XVV)'yi yükler Bu terimde 101E, Şekil 2.12'de gösterildiği gibi program durum alanının vektör bölgesi başlangıcıdır. 8-bitlik VV sayısı kesme birimi tarafından, kesme tanıtıcısı nın en küçük değerlikli baytı olarak tanımlanır.

Vektörlenmiş kesme yanıtının bir örneği olarak, VI'yi alçak düzeye çeken ve

sonuçta ST3-ST0'da ortaya çıkan 0111(ikili) durum koduna, tanıtıcı sözcük 0006'yı adres/veri yoluna koyarak yanıt veren bir çevresel birimi ele alalım. İşlemci, tanıtıcı ve işlemci durum sözcüğünü yığında sakladıktan sonra, FCW'yu 1014 adresinde bulunan sayı ile, ve PC'yi 102A adresinde bulunan sayı ile yükler, çünkü:

VV = 06
2 x VV = 2 x 06 = 0C
101E + 0C = 102A'dır.

(*Not:* Tüm aritmetik işlemler onaltılı sisteme göredir.)

Bu bölüm, Z8000 kesme olanaklarına ilişkin bir kullanıcı kılavuzu değildir. Amacımız okuyucuya, bu olanakların gücü ve çok yönlülüğü ile ilgili bir fikir vermektir.

Kesme olanaklarının karşılaştırılması

Bir mikroişlemcinin seçiminde kesme olanaklarının önemli olup olmadığı, sistemdeki G/Ç birimlerinin sayı ve özelliklerine bağlıdır. Genel olarak, performans gereksinimleri, kesme ile başlatılan G/Ç kullanımını gerektirmedikçe, meşgul sinyalinin tekrar tekrar taranması işlemiyle birlikte basit anlaşma yöntemini uygulamak daha kolaydır. Tüm G/Ç birimlerini, mevcut zaman içerisinde bu şekilde hizmete sunmak mümkün olsaydı, kesme olanaklarının bir önemi olmayacaktı.

Mikroişlemcilerin kesme olanakları konusundaki çalışmamızda, bunların M6800'deki gibi çok basit ve Z8000'deki gibi yüksek düzeyde karmaşık kolaylıklar olabildiğini ve bu iki uç örnek arasında çeşitli düzeylerde çok çeşitli tiplerinin bulunduğunu gördük. Kesme olanaklarının karmaşık olması, çok sayıda G/Ç biriminin bulunduğu bir sistemde, her bir birimin isteğine hızlı bir şekilde yanıt verme gereksinimi ile birleştiği zaman, elde edilen yarar azalır. Böyle bir durumda verimli bir kesme vektörleme yöntemi önem kazanmaktadır.

Ele aldığımız mikroişlemcilerden M6800, en az karmaşık olan kesme sistemine sahiptir. Karmaşık olmaması donanım ve yazılım gereklilikleri açısından çok büyük avantaja sahiptir. Özellikle çevresel arabirim gereksinimleri çok basittir, çevresel birimin G/Ç işlemlerine ek olarak kesme hattını etkinleştirmek dışındaki görevleri çok azdır. Kesme yanıt sırasının kolaylığı, yazılım isteklerinin kolay olmasına neden olur, çalışma kaydedicilerinin saklanması bile otomatik olarak gerçekleştirilir.

8080, kesme işlemleri açısından, ele alınan mikroişlemciler arasında en az tatmin

edici olandır. Kuşkusuz, çevresel arabirimin bir sonraki komut kodunu yola çı
ması, diğer mikroişlemcilere kıyasla daha fazla donanım karmaşıklığına ne
olur. Ancak bu yaklaşımın M6800'e göre üstün yönü, kesme vektörleme işlem
kolayca uygulanabilmesidir.

Burada ele alınan 8-bitlik mikroişlemciler arasında Z80, kesme olanakları en g
olan mikroişlemcidir. Mod 0, 8080 kesme olanağıyla tam bir uyumluluk göst
Mod 1, M6800'ün kesme yanıtına birçok yönden benzeyen bir basit kesme y
sağlar. Mod 2, gerçek bir vektörlenmiş kesme avantajı sunar.

Z8000 mikroişlemcisi, çok kapsamlı donanımsal ve yazılımsal kesme ve yazılım
tuzak olanakları sunmaktadır. Z8000'in çok güçlü bir mikroişlemci oluştum
amacıyla geliştirildiği açıkça bellidir. Tasarımında ikinci bir disk saklama al
içeren ve çok daha büyük bilgi işlem sistemlerinde bulunan tipte işletim sistem
göz önüne alınmıştır. Buradaki gibi sistemlerde kullanılan teknikler, verimli kes
işlemlerinin çok önemli olduğu minibilgisayarlarda ve hatta anabilgisayar sist
lerinde kullanılan tekniklere dayanır.

3.4 Mikroişlemci enerji gereksinimleri

Düzenli bir şekilde çalışabilmesi için tüm mikroişlemciler, en azından belli
oranda sabit olan bir gerilimde, sabit ya da doğru akımla beslenmelidir. Ak
normalde, bir a.c. ana güç kaynağından, transformatör, doğrultucu, saklama k
dansatörü ya da bir gerilimi kararlı tutan devreler aracılığıyla alınır. Enerjinin pil
sağlanması da diğer bir alternatiftir.

Belli bir uygulama için mikroişlemci seçerken enerji kaynağı bazen dikkate al
mayan bir kriterdir. Fakat farklı mikroişlemcilerin gerektirdiği gerilim ve ak
kaynağı büyük farklılıklar gösterir. Bazı uygulamalar için gerilim ve akım kayn
mikroişlemcinin en önemli özelliği olabilir. Enerji kaynağının dikkatli bir şekil
hesaba katılması gerektiren uygulamalardan bir kısmı aşağıdadır:

1 Bileşen maliyetinin mümkün olan en düşük düzeyde olmasının gerektiği s
temler.
2 Her zaman ya da bazen pilden enerji alarak çalışması gereken sistemler.
3 Dar alanlarda ya da yüksek sıcaklıklarda çalışması gereken sistemler.

Belirli bir mikroişlemcinin gereksinimlerini tartışmadan önce, enerji kaynaklarını
tüm sayısal sistem elemanlarıyla ilişkili olan yönlerini ele alacağız. İlk ortak nok

az sayıda gerilim kaynağının tercih edilmesidir, bu görüşü kabul etmeyen tasarımcı sayısı herhalde çok azdır.

Yalnızca güç kaynağı gereksinimleri açısından bakıldığında, üç gerilim hattı gerektiren 8080, örneğin yalnızca bir +5 V'luk kaynağa gereksinim duyan M6800'den daha az tercih edilebilen bir mikroişlemcidir. Ancak, mikroişlemciyi, çok sayıda kaynak hattını gerektiren belirli bellek yongaları gibi sistem elemanlarıyla birleştirdiğimizi düşündüğümüzde, yalnızca bir kaynak gerilimi gerektirdiği için bir birimi seçmenin anlamsız olacağı unutulmamalıdır.

Herkesin kabul edeceği bir diğer görüş de, bir birimin akım gereksinimi ne kadar düşük olursa o kadar iyi bir birim olduğu görüşüdür. Bunun için çeşitli nedenler vardır. Enerji kaynağı devresinin maliyeti, çekilen akımla -az ya da çok- doğrusal olarak artar. Yüksek akım tüketimi, yüksek enerji çekilmesi demektir ve bu da daha sonra daha yüksek çalışma sıcaklıklarına neden olur. Güvenilirlik, sıcaklığın artmasıyla azaldığından, eğer sistemin yüksek akım tüketme özelliği varsa soğutucuların konulması gerekebilecektir.

Sayısal sistem birimlerinde yüksek miktarda akım tüketiminin, devrenin fiziksel yerleşim düzenini daha riskli hale getirdiği, fazla göz önünde bulundurulmamaktadır. Mikroişlemciler de dahil entegre sayısal devrelerin çoğunluğunda akım tüketimi sabit değildir. Entegre devre içindeki mantıksal kapıların her biri açılıp kapatıldığında akım gereksinimi ani değişmeler gösterir. Dolayısıyla sistem, güç kaynağından daha fazla akım çekilmesini gerektirebilir. Bu akımlar için sıklıkla nanosaniyeler düzeyinde ya da daha az düzeyde hızlı yükselme süreleri gereklidir, bu durum güç kaynağı devresini tasarlarken göz önünde bulundurulmalıdır. Okuyucu, biraz abartılı olmakla birlikte, 1 nsn yükselme süreli 100 mA genlikte darbe üretebilen bir devrenin tasarımını gözünde canlandırabilir.

Fourier analizi, bize, hızlı yükselme süreli darbe tipindeki dalgabiçimlerin, çok yüksek frekans bileşenleri içerdiğini söylemektedir. Bu nedenle, bir mikroişlemci devresindeki gerilim kaynağı hatlarının, çok yüksek frekanslı bileşenleri bulunan sinyalleri taşıyan iletim hatlarından oluştuğunu kabul etmeliyiz. Mikroişlemci sistemler için iyi tasarlanmış baskılı devre kartlarının VHF analog devrelerde kullanılan devre kartlarındakilerle benzer kurallara uymalarının nedeni budur. Bu kurallar arasında; yer düzleminde ve uygun boyutlarda iletkenlerin kullanılması ve güç kaynağı hatlarının düzenli aralıklarda ayrılması (dekuple edilmesi), bulunmaktadır. Birimlerin akım tüketimlerinin artmasıyla birlikte, bütün bu fiziksel yerleşim sorunları büyür.

Güç kaynağı gereksinimlerinin karşılaştırılması

Bir mikroişlemcinin tükettiği akım miktarı mikroişlemcinin imal edilişinde kullanılan yöntemlere bağlıdır. Kitabın yazıldığı dönemde, imalat teknikleri sürekli olarak geliştirilmekte, fakat mikroişlemci üretiminde iki yöntem ve bunların tipleri hakim durumdaydı. Bu yöntemler;

1 NMOS : N-kanal tümler metal oksit yarı-iletken
2 CMOS : tümler metal oksit yarı-iletken

Burada bu yöntemleri anlatmayı düşünmüyoruz. Yalnızca, bu yöntemler kullanılarak imal edilen mikroişlemcilerin özellikleri üzerinde duracağız.

Buraya kadar tanıtılan mikroişlemcilerin hepsi NMOS tipi birimlerdir. Bu mikroişlemciler 8080, Z80, M6800, Z8002 ve COP420'dir. Önce, biraz daha özelleştirilmiş bir birim olan COP420'yi hariç tutarak geri kalan genel amaçlı birimleri inceleyeceğiz. Bunların hepsi 0.5 - 1.5 W arası enerji tüketirler. Tipik kaynak gereksinimleri Z8002 için 300 mA'de 5 V ve M6800 için 200 mA'de 5 V dur. Bu değerler çok yüksek görünebilir fakat okuyucu orta ya da büyük sistemlerde mikroişlemci yongası güç gereksiniminin toplam güç ihtiyacının yalnızca küçük bir bölümü kadar olduğunu bilmelidir. Yüksekhızlı sayısal elemanlar, özellikle bellek yonga dizileri, iyi bilinen güç tüketicileridir. Toplam olarak 5 V, 10 A gibi bir güç tüketimi bir mikroişlemci sistemi için normaldir. Ayrıca transformatör ve saklama kondansatörü gibi güç kaynağı bileşenleri maliyetinin, mikroişlemci yongalarının maliyetinden kat kat daha fazla olabileceğini de belirtmeliyiz.

Yüksek ve düşük hız ve düşük işlem gücü olan mikroişlemci sistemlerini tasarlarken, normalde nispeten yüksek akım tüketimi tolere edilmelidir. Üretim ve bileşen maliyetlerinin düşük olmasının ya da pille çalışmanın istendiği diğer sistemlerde yüksek güç tüketimine izin veremeyiz. Bu durumda farklı bir mikroişlemcinin seçimi söz konusu olacaktır. Böyle bir seçim yaparken, sayısal elektronik bileşenlerin çoğunluğunda olduğu gibi, düşük güç tüketimini işlem hızının yüksek olmasına tercih etmeliyiz. Güç tüketimi az olan mikroişlemciler daha yavaş hızda çalışma eğilimindedir ve bu nedenle, her zaman olmasa da, güç tüketimi yüksek olan birimlere göre daha düşük düzeyde işlem gücü sunarlar.

COP420, orta düzeyde performansa sahip ve nispeten düşük güç gereksinimi olan mikroişlemcilere bir örnektir. Burada anlatılan diğer mikroişlemciler gibi COP420 da, NMOS teknolojisi kullanılarak imal edilmiştir, buna karşın, güç gereksinimi yalnızca 6 mA, 4.5-9.5 V'tur.

Düşük akım gereksinimlerine ek olarak kaynakta gerilimindeki büyük dalgalanmaların hoş görülebilmesi, çok ucuz güç kaynağı devrelerinin kullanılabilmesine izin vermektedir. Bu özelliklerinden dolayı nispeten küçük bir pil ile çalışabilir. Ancak pil ile sürekli bir çalışma sağlamak için, COP420L ile aynı aileden olan COP420C daha uygundur.

COP420L'ye benzer birçok yönü bulunan COP420C birimi, CMOS tekniği ile imal edilmiştir. Bu teknolojinin özelliklerinden birisi, güç tüketiminin çok düşük olmasıdır. COP420C, COP420L gibi aynı 16-mikrosaniyelik komut süresi ile çalışırken, yalnızca 600 μA, 5V gerektirir. Saat frekansının azaltılması, gereken gücü daha da azaltır. Minimum hızda çalışırken COP420C, 2,4V'ta 25 μA gibi çok düşük bir güç kaynağına gereksinim duyar. Bu durumda komut saykıl süresi 244 mikrosaniyedir.

Mikroişlemci imalatçıları, düşük güç gereksiniminin arzu edildiğini elbette bilmektedir. Bu kitabın yazıldığı dönemde, M6800 ve Z80'in özelliklerine benzer özelliklerde düşük güç gereksinimli CMOS mikroişlemcileri piyasaya sunulmuştur. Bu gibi gelişmelerin sürmesini beklemek doğrudur fakat mümkün olan en yüksek hız ve bilgi işleme gücü için yakın gelecekte de yüksek güç tüketiminin gereklerine katlanmayı sürdüreceğiz.

Sorular

3.1 Mikroişlemci G/Ç uygulamalarında "onay" ile ne anlatılmak istendiğini söyleyiniz.

3.2 Kesme ile başlatılan G/Ç'ın meşgul tarama yöntemine karşı başlıca üstünlükleri nelerdir?

3.3 Bir diyagram yardımıyla, düşük öncelikli bir kesmeye hizmet verilirken yüksek öncelikli bir kesme ortaya çıktığında, program akışının nasıl olacağını tanımlayınız.

3.4 Bir diyagram yardımıyla, daha yüksek öncelikli bir kesme hizmeti verilirken, düşük öncelikli bir kesmenin ortaya çıkması halinde, program akışını tanımlayınız.

3.5 Vektörlenmiş ve vektörlenmemiş kesmelerin arasındaki farkı açıklayınız.

3.6 Vektörlenmiş kesmelerin vektörlenmemiş kesmelere olan ana üstünlüğünü belirtiniz ve aşağıdaki mikroişlemcilerle bir kesme oluştuğunda ortaya çıkan

olaylar dizisini anlatınız:
(*a*) 8080
(*b*) M6800

3.7 Farklı mikroişlemcilerdeki G/Ç olanaklarına ilişkin üç alternatif yakl anlatınız.

3.8 G/Ç'ya port yaklaşımı ile bellek-haritalı G/Ç yaklaşımı arasındaki f açıklayınız. Bu G/Ç yöntemlerini kullanan mikroişlemcilere örnekler v niz.

3.9 Z80 mikroişlemci sisteminde bir çevresel arabirim devresi tarafından, G komutunun çalıştırıldığını bildirmek amacıyla kullanılan sinyal hangisi

3.10 Bir çevresel arabirimin Z8000 sisteminde durum sözcüğünü, bir G/Ç kom tunun çalıştırıldığını bildirmek amacıyla nasıl kullandığını açıklayınız.

3.11 Aşağıdaki mikroişlemcilere dayalı sistemlerde teorik olarak, adreslenebil cek maksimum G/Ç birim sayısı kaçtır?
(*a*) 8080
(*b*) Z80
(*c*) M6800
(*d*) Z8000

3.12 Maskelenebilen bir kesmeden çok maskelenemeyen bir kesmenin kullan mını gerektiren bir uygulamayı tanımlayınız.

3.13 Aşağıdaki birimlerin gerektirdiği bağımsız d.c. kaynak hatları sayısını b lirtiniz.
(*a*) 8080
(*b*) Z80
(*c*) M6800
(*d*) Z8000
(*e*) COP420L

3.14 Üretimde yaygın olarak kullanılan iki imalatçı yöntemiyle NMOS ve CMOS olarak tanımlanan birimler üretilmektedir. Bunlardan hangisi pil ile çalışan birimler için en uygunudur?

Bölüm 4 Mikroişlemci destek birimleri

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde; üretici veri sayfalarının yardımıyla, bir mikroişlemci sisteminde bulunan aşağıdaki işlevsel birimlerin özellik ve işlevlerini tanımlayabilmelisiniz:*

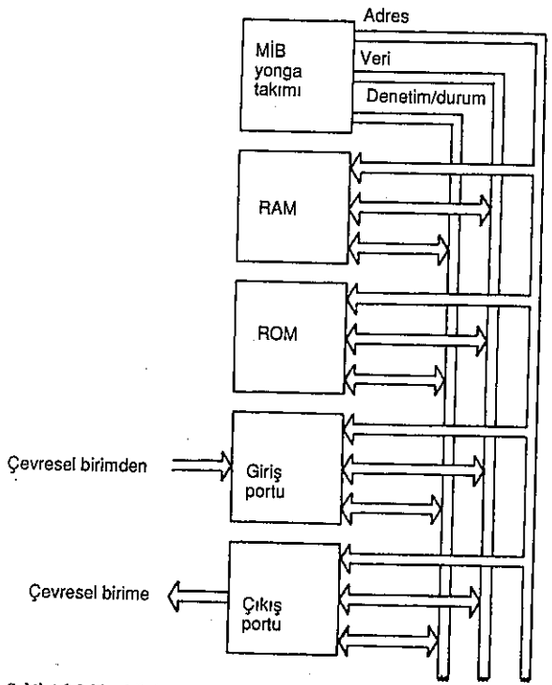
1 *MİB yonga takımı*
2 *Yol tamponları*
3 *Paralel G/Ç birimleri ve devreleri*
4 *Seri G/Ç birimleri*
5 *Adres kod çözücüleri*
6 *Yol tekilleyicileri*
7 *Saat devresi*
8 *Statik ve dinamik ana RAM*
9 *ROM ve EPROM.*

4.1 Giriş

Bölüm 2 ve 3'te, bazı mikroişlemcilerin, özel olarak hesaplama görevlerini verimli biçimde yapabilmeleri için minimum düzeyde ek devre düzenlemeleri gerektirecek şekilde tasarlandıklarını anlatmıştık. Bunlar, tek-yongalı mikrobilgisayar olarak adlandırılan birimlerdir. MİB-tipinde mikroişlemciler olarak tanımladığımız diğer birimler, mikrobilgisayar sisteminin yalnızca bir bölümünü oluştururlar. Tek-yongalı mikrobilgisayarlar, maliyet açısından avantajlıdırlar. MİBtipinde mikroişlemci ise genellikle daha yüksek bir işlem gücü avantajına sahiptir. Kullanıcının, belirli bir iş için gereken işlem becerisini sağlayabilecek destek devrelerini seçebilmesi ise, MİB-tipinde mikroişlemcilerde daha da önemli bir özelliktir. Dolayısıyla esneklik ve çok yönlülük, tek-yongalı sistemlerden çok daha fazladır.

MİB-tipindeki mikroişlemcileri destekleyen ek devre düzenlemelerine olan gereksinim, çok çeşitli özel-amaçlı entegre devrelerin üretimine yol açmıştır. Bu bölümde bu birimlerin bazılarını inceleyeceğiz. Bunlar, tamponlar ve mandallar gibi kavramsal olarak basit olan birimlerden, çok karmaşık seri ve paralel arabirim devrelerine kadar değişen bir çeşitlilik gösterirler. Bir kısmının karmaşıklık düzeyi, basit mikroişlemci yongalarının düzeyine eşittir.

MİB-tipinde mikroişlemci-tabanlı bir bilgisayar sistemi, Şekil 4.1'de blok diyag-



Şekil 4.1 Mikrobilgisayar yapısı

ram olarak gösterilmiştir. Sistem esas olarak, yol adı verilen paralel veri iletişim kanalları ile bağlanan MİB, RAM, ROM, G/Ç portları vb. işlev birimlerinden oluşmaktadır. Şekil 4.1'de adres yolu, veri yolu ve denetim/durum yolu gösterilmektedir. Çok genel ifade edilirse, veri yolunun, işlemler sırasında kullanılan komut kodlarını ve veriyi, MİB ve diğer işlevsel birimler arasında taşıdığını söyleyebiliriz. Adres yolu bir veri işlemine katılan işlevsel birimi tanımlayan adresi taşır.

Durum/denetim yolu tanımlaması daha az anlaşılır durumdadır. Mikroişlemciden başka bir yere veya başka bir yerden mikroişlemciye gelen giriş ve çıkış hatlarını içerir. Gerçek işlevi ve hat sayısı, sistem gereksinimlerine ve kullanılan mikroişlemciye göre değişir. Bu yoldaki hatların tipik örnekleri IORQ ve INT'dir. IORQ sinyali etkin olduğunda G/Ç portları tarafından, MİB'in bir G/Ç komutunu çalıştırmakta olduğunu belirtmek için kullanılır. INT, çevresel birimden, mikroişlemci içinde bir kesme dizisini başlatan sinyali taşır.

S4.1
4.2

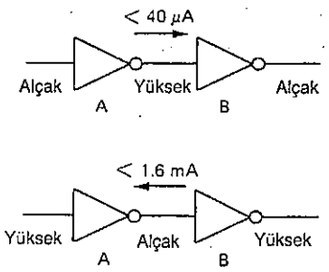
4.1 nolu kaynaktan alınan Şekil 4.2, çok daha az düzeyde genelleştirilmiş, fakat daha pratik olan bir sistem blok diyagramını göstermektedir. 16-bitlik Z8000

Şekil 4.2 Z8000 mikroişlemci sistemi

mikroişlemci-tabanlı bu sistemde işlevsel birimler, Şe 4.1'de gösterilen daha genel yapıdaki sisteme göre daha azı biçimde tanımlanmışlardır. Fakat yine de, sistem yol hatları veri, adres ve denetim/durum işlevlerine ayrılması oldu açık bir biçimde görülmektedir.

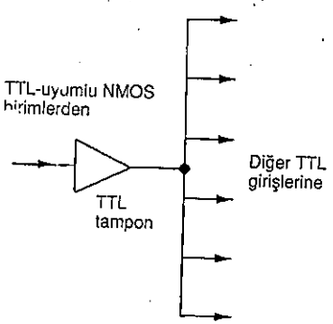
4.2 Tamponlama gereksinimi

Alan etkili transistör tabanlı entegre devrelerin yaygın biçim geliştirilmesinden önce sayısal sistemlerin büyük çoğunlu transistör-transistör mantık (TTL) teknolojisi kullanılarak mal edilmişlerdi. TTL mantık birimleri ailesi hala en ço bilinen birim ailesidir ve büyük farkla maksimum sayıda kul lanılabilir işlev sunan birimler topluluğudur. Bunun sonu olarak mikroişlemci sistem tasarımcıları TTL bileşenleri NMOS ya da belki CMOS teknolojisini kullanan mikroişlemc ailelerinin bileşenleriyle bir arada kullanmayı uygun bulmak tadır. Çok eski mikroişlemcilerde bu durum çok önemli soru lara yol açmıştır çünkü mikroişlemcinin mantıksal gerilim ve akım düzeyleri, TTL mantık düzeylerinden büyük oranda fark lıdır. İki birim tipi arasında bir arabirimin gerekli olduğu her durumda, akım ve gerilimi uygun şekilde dönüştüren özel amaçlı entegre devrenin yerleştirilmesi zorunluluğu doğmak taydı. Bu çeşit dönüştürücü devrelere kaydırma tamponları ad verilmektedir.



Şekil 4.3 TTL ile akım üretme ve çekme

Sözü edilen sorunlar, günümüzde bir ölçüde giderilmiştir. Mikroişlemci ailelerindeki entegre devrelerin çoğunluğu TTLuyumludur. Genellikle TTL birimlerine doğrudan doğru ya bağlanabilirler ve mantık sinyalleri alışverişini başarılı şekilde gerçekleştirirler. Ancak bu, bir NMOS mikroişlemcisinin akım/gerilim G/Ç özelliklerinin TTL bileşenlerinin özellikleriyle aynı olduğu anlamına gelmez. Aradaki fark temel olarak, birimlerin sağlayabildikleri çıkış akımı miktarındadır.



Şekil 4.4 Sürme kapasitesini arttırmak için TTL tamponu kullanılması

TTL mantık elemanları "akım çeken" şeklinde nitelendirilirler. Şekil 4.3, bir 704 altı tersleyici yonganın seri bağlanmış iki kapısını göstermektedir. A kapısının çıkışı yüksek olduğunda B kapısına, yalnızca, 40 μA'dan daha az, çok küçük bir akım

gider. Burada B kapısının etkin giriş direnci yüksektir. Eğer A kapısı çıkışını alçak düzeye çekeresek o zaman, akım B kapısı girişinden, A kapısı çıkışına doğru akar. Bu, A kapısının, B kapısından akım çektiği biçiminde tanımlanır. Standart TTL birimlerinda bu akım en çok 1.6 mA kadar olabilir. Gerçekte, doğru mantık düzeylerinin korunduğundan kesinlikle emin olmak için A kapısı, B kapısından, istenen en yüksek düzey olan 1.6 mA kadar akımı çekebilmelidir. Eğer Şekil 4.4'te görüldüğü gibi, birden daha fazla sayıda kapıdan akım çekilmesi gerekiyorsa, o zaman A kapısı, bağlanan her bir kapı için 1.6 mA kadar akımı çekebilecek durumda olmalıdır. Bu tipteki standart TTL kapıları en az 16 mA lik akımı çekebilir, bu da bir kapı çıkışının 10 girişi sürebilecek kapasitede olduğu anlamına gelir. Mikroişlemci sistemlerinde TTL'nin, düşük-güçlü Schotky ya da LS çeşidinin kullanılması daha yaygındır. Bu durumda maksimum düşükdüzeyli giriş akımı 0.36 mA'dir ve çıkışta çekilebilecek en yüksek akım 8 mA'dir, buradan her çıkış kapısının 8/0.36=22 girişi sürebileceği ortaya çıkmaktadır.

İncelendiğinde, tipik bir NMOS mikroişlemcisinin akım çekebilme düzeyinin normalde, TTL devrelerinden çok daha az olduğu görülür. Örnek olarak M6800'ü alırsak, veri ve adres hatları, birden fazla sayıda standart TTL kapısından akım çekemezler. Dolayısıyla eğer bu hatlardan, birden fazla TTL girişinin sürülmesi gerekirse, Şekil 4.4'te gösterildiği şekilde bir tampon kapı kullanılmalıdır.

Buraya kadar yalnızca, mantık düzeylerini TTL girişlerinde, gereken sınırlar arasında tutmak için, yeterli ölçüde akımın çekilmesi sorununu ele aldık. Daha küçük fakat eşit değerde önemli bir diğer sorun ise, gerektiğinde yol hatları üzerinde ortaya çıkması muhtemel darbeli dalgabiçimlerinin, uygun yükselme ve düşme zamanlarını ve gecikmelerini koruyabilme sorunudur. Bu sorun yalnızca, TTL'lerin, NMOS birimlerinden sürülmesi sırasında değil, iki NMOS biriminin bir yol aracılığıyla bağlanması sırasında da ortaya çıkmaktadır. Bu sorunun değerlendirilmesi için, bir birimin çıkışını bir diğer birimin girişine bağlayan iletkenin, süren birim için, kapasitansı büyük yük olduğunun anlaşılması gerekir. Çekilen birimin giriş empedansı da kısmen sığalı olarak görünecektir. Bu durumda çekilen birimin girişindeki gerilimi değiştirmek için, gerilim bir kondansatör üzerinden etkin biçimde değiştirilmektedir.

Q yüküyle yüklü bir kondansatörünün gerilimi

$V = Q/C$ ile hesaplanır.

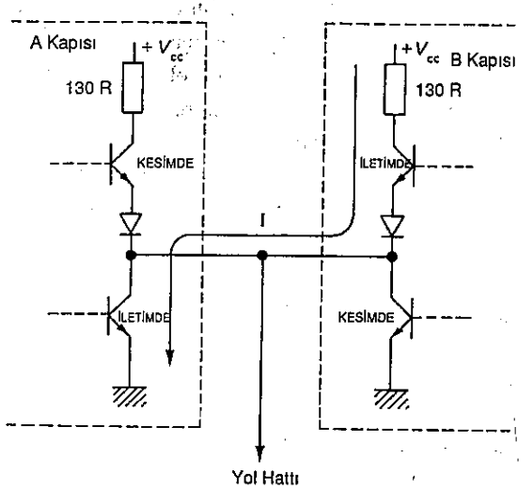
Gerilimin değiştirilebilmesi için yükün aktarılması gerekir. Dolayısıyla yükün aktarılabileceği maksimum yüksek hız, gerilim dalgabiçimlerinin en düşük yüksel-

me ve düşme zamanlarını ve gecikmesini belirleyecektir. Bu ise bir çıkış sürülebilecek giriş sayısını sınırlayan bir başka etkeni ortaya çıkarmaktadır, çünü ister TTL, isterse MOS olsun her bir giriş ek bir kapasitans oluşturmaktadır. Bu açıklanan statik ve dinamik sorunlar nedeniyle, M6800 hatları en çok 1 TTL, M6800 ailesinden 8 NMOS birimini sürebilir. Herhangi bir hattaki en yüksek kapasitif yüklenme 130pF'yi geçmemelidir, bu değere, ara bağlantı kablolarını kaçak kapasitansı ile yol hattına bağlanmış diğer birimlerin kapasitansı da dahil.

Okuyucu, yukarıda anlatılanlardan şu iki noktayı değerlendirebilmelidir: Birinci, en basit sistemler dışındaki tüm sistemlerde tampon birimler bir gereklilik. Donanım tasarımının genel bir kuralı, bir kuşku durumunda her zaman tamponlar sisteme eklenmesidir. İkinci nokta ise, kaçak kapasitansı minimum düzeye indir-
S4.5 iyi bir düzenlemenin, yüksek hızlı sayısal devreler için, yüksek-frekanslı analog
4.6 devrelerdeki kadar önemli olduğudur.

Üç-durumlu yol sürücüleri

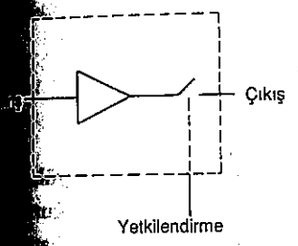
Yol içeren herhangi bir sistemde bir çok farklı birimin çıkış hattının, yol hatları aracılığıyla birbirine bağlanması kaçınılmaz bir gereklilikdir. Şekil 4.1 ve 4.2'de açıkça görülmektedir. Gösterilen sistemde ROM, RAM ve giriş portu, çeşitli zamanlarda, MİB'e aktarılmak üzere veri yoluna veri koymak isteyeceklerdir. Bu gereksinimin bir sonucu olarak birimlerin çıkış devrelerine belli bazı sınırlamalar konulmuştur. Sınırlamaların gerekçelerini anlamak için, iki standart TTL kapı çıkışı



Şekil 4.5 Totem direği çıkışlarının bağlantısı

şının bir yol hattına bağlanması halinde neler olacağını düşünelim. Standart TTL Şekil 4.5'te gösterilen totem direği çıkış devresini kullanır. Çıkış yüksek olduğunda totem direği'ndeki üstteki transistor iletimde, alttaki transistor ise kesimdedir. Çıkış alçak düzeyde iken bunun tersi bir durum ortaya çıkar. Şekil 4.4'de, İki kapının çıkışlarının birbirine bağlanması halinde nasıl bir sonuç elde edildiği görülmektedir. Totem Direği B'nin iletimdeki üst transistörü ile totem direği A'nın iletimdeki alt transistörü, Vcc ile toprak arasında düşük bir empedans gösterirler. Şaşırtıcıdır ki bu durum, çıkış devresine her zaman zarar vermez. Ancak kapılardan herhangi birinin düzenli biçimde çalışmasına engel olur.

Bir bilgi işlem sisteminin işlevsel birimlerinden sistem yolu üzerine veri yerleştirilmesi için normalde, "üç-durumlu (tristate) kapı" olarak bilinen özel çıkış devreleri kullanılır. "Tristate" terimi, National Semiconductor Corporation tarafından kullanılan tescilli bir marka adıdır ve telif hakkı bu kuruluşa aittir. Diğer imalatçılar aynı devre çeşidini adlandırmak için "three-state" terimini kullanabilmektedir.



Şekil 4.6 Üç-durumlu kapının işlevsel tanımlaması

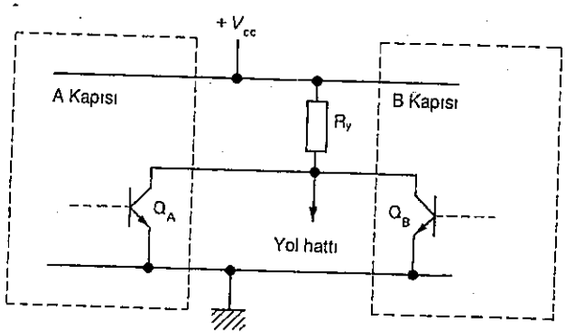
Üç-durumlu kapılarda normal kapı girişlerine ek olarak bir de "yetkilendirme girişi" vardır. Yetkilendirme hatlarının ikisinin de etkin olmadığı durumlarda, totem direği çıkış devresi her iki transistörü kesime sürer. Bu özellik çıkışın, hemen hemen açık bir devre gibi çok yüksek bir toprak direnci ile belirsiz bir gerilimde dalgalanmasına yol açar. Dolayısıyla devrenin üç olası çıkış durumu vardır - yüksek, alçak ve etkisiz. İşlevsel olarak devre, Şekil 4.6'da görülen, bir kapı ve anahtara benzer.

Normal totem direği çıkış devresi ile Şekil 4.6'daki üç-durumlu düzenleme arasındaki fark, herhangi bir anda yalnızca bir kapının yetkilenmesi koşuluyla, istediğimiz kadar sayıda üç-durumlu kapının çıkışını güvenle birbirine bağlayabilmemizdir. Üçdurumlu çıkışları olan, ve yol-yönelimli sistemlerde kullanılmak üzere özel olarak tasarlanmış bir dizi tampon kapı, Ek H'da gösterilmektedir. Bu konuda anlatılan mikroişlemci destek entegre devrelerinin çoğunluğunda dahili üç-durumlu
S4.7 çıkış devreleri bulunmaktadır.

Açık-kollektörlü yol sürücüleri

Üç durumlu tamponlar yerine bir başka seçenek, yol sürücü birimlerin açık-kollektör devrelerinin kullanılmasıdır. Üç-durumlu devreler bulunmad yaygın olarak açık kollektör tekniği kullanılmaktaydı. Ancak günümüzde cıların çoğunluğu üç-durumlu yaklaşımı tercih etmektedir.

Adından da anlaşıldığı gibi bir açık-kollektörlü kapı çıkış devresi, kollek direnci bulundurmayan tek bir transistördür. Kollektör yük direnci devreye dan bağlanmalıdır. Şekil 4.7'de, ortak bir yol hattına bağlanmış, iki açık-koll kapı çıkış devresi görülmektedir.

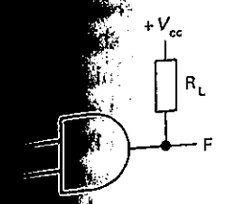


Şekil 4.7 Açık-kollektörlü kapılar

Yol hattı RL direnci aracılığıyla, her bir çıkış transistörü için kollektör y sağlayan Vcc'ye bağlanmıştır. Kapılardan herhangi birinin çıkışı yüksek oldu da çıkış transistörünün KESİM durumunda olduğunu biliyoruz. Kapılardan her gi birinin çıkışı alçak olduğunda ise çıkış transistörü İLETİM durum geçmektedir.

Şimdi Şekil 4.7'de, A kapısının çıkış transistörünün KESİM'de olduğunu düşü lim. Bu durumda, ortak çıkış hattı B kapısının çıkışını izler. Q_B, İLETİM olduğunda çıkış alçak, KESİM'de ise yüksek olacaktır. Açık kollektör yaklaşımı esas özelliklerinden birisi, çıkış transistörü KESİM durumundayken, pasif olan tü kapıların normalde yüksek çıkış durumunda olmalarının gerekmesidir. Ortak çı hattı daha sonra etkin kapının normal çıkışını izleyecektir. Doğal olarak, bir defa yalnızca bir kapının etkin olmasına izin verilmektedir.

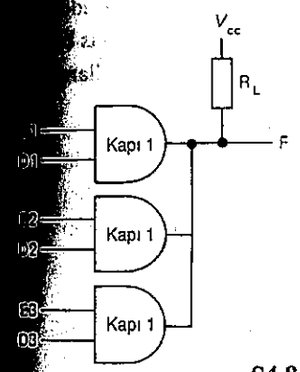
Açık-kollektörlü tamponlar için VE düzenlemesi uygundur. Şekil 4.8'de, doğrulu



Açık-kollektörlü VE kapısı
yük direnci

E	D	F
0	0	0
0	1	0
1	0	0
1	1	1

9 VE doğruluk tablosu



S4.8

Şekil 4.10 Açık-kollektörlü VE yol sürücüleri

tablosu Şekil 4.9'da verilen böyle bir kapı gösterilmektedir. TTL 7409 tipi entegre devre bu çeşit açıkkollektörlü kapılarından dört tane içerir. Doğruluk tablosundan, E girişi yüksek olduğunda, F çıkışının her zaman D girişini izlediği anlaşılmaktadır. Buradaki gibi tek bir kapı için, D girişi alçak olduğu zaman F daima alçaktır. Devrelerle ilgili adlandırmalarda bu, çıkış transistörünün KESİM durumunda olduğu anlamına gelir. Bu ise, çıkış hattını, bağlı olduğu başka bir kapı tarafından denetimi için serbest bırakır.

Şekil 4.10'da, tek bir yol hattını çeken üç açık-kollektör VE kapısı görülmektedir. E2 ve E3'ü alçak düzeyde tuttuğumuzu varsayalım. Bu 2 ve 3 kapılarının çıkış transistörlerini KESİM durumuna getirir, bu şekilde R_Y'den ya hiç akım geçmez ya da çok az geçer. Eğer E1 yüksekse yol üzerindeki mantık düzeyi her zaman D1'deki mantık düzeyini izler. Dolayısıyla kapılardaki E girişleri yetkilendirme hatlarıdır, bunlar belli bir kapının yolun denetimini almasına olanak tanırlar. Belirli bir zamanda yalnızca bir E girişinin yüksek olabilmesinin nedeni açıktır.

4.3 MİB yonga takımı

Bölüm 1'de 8080, Z80, 6800 ve Z8000 gibi birimleri MİB-tipinde mikroişlemciler olarak adlandırdık, bu adlandırma, bu birimlerin kendi başlarına, bir mikroişlemci sistemdeki MİB-'in işlevini görebilecek durunda oldukları anlamına gelir. Bu tam olarak doğru değildir. Belirtilen mikroişlemcilerin MİB işlevi gerçekleştirebilmesi için ek devre düzenlemesi desteğine gereksinimleri vardır. Her ne kadar bazen özel destek devre bileşenleri kullanılmaktaysa da söz konusu destek, genellikle özelleşmiş entegre devreler biçimindedir. Birlikte MİB işlevini gerçekleştiren birim ya da devre toplulukları için "MİB yonga takımı" terimini kullanacağız.

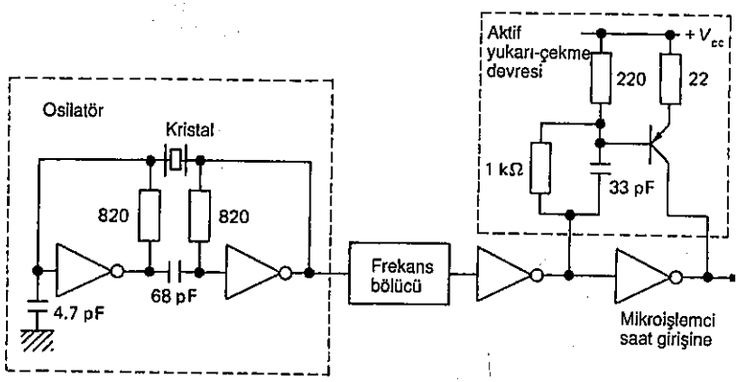
Saat gereksinimleri

Tüm mikroişlemciler, bir komutun getirilmesi ve yürütümü sırasında ortaya çıkan iç olayların zamanlamasını belirlemek için bir saat osilatörüne gerek duyar. Bazı eski mikroişlemci-

lerde frekansları aynı olan toplam dört saat dalgabiçimi gerekli olmuştu bunlar zamanla terkedilmiştir. Bugün dört-fazlı saat gereksinimi olarak tanı bu gereksinim sistem tasarımcılarının çoğunluğu tarafından kabul edile sorun olarak görülür. Ek A-E'de anlatılan mikroişlemcilerden 8080 ve iki-fazlı saat gerektirirler. Z80 ve Z8000 yalnızca bir saat dalgabiçimi, yani saata gerek duyarlar. Giderek artan şekilde yaygınlaşan bir saat olanağı COP420L'de bulunmaktadır, bu birimde saat osilatörü devresinin büyük b yonga içinde yer alır. Kullanıcı, birimin saat gereksinimini karşılamak için ca, uygun pinler arasına bir kristal veya rezonans devresi bağlamalıdır.

Z80 ve Z8000 gibi tek-fazlı saat gerektiren mikroişlemcilerde, ayrı bir saat devreleri bileşenlerinin kullanılması yaygındır çünkü devrenin, aktif birimle yalnızca TTL kapılarını kullanmasını sağlamak genellikle olanaksızdır. B nedeni totem direği çıkış devresi içeren standart TTL kapısının çok sınırlı bir kaynağı oluşturma özelliği olmasıdır, daha önce gördüğümüz gibi TTL kapı akım çekebilme yeteneği iyi düzeydedir. Açık-kollektörlü TTL kapılarında kaynağı oluşturma kapasitesi arttırılmıştır fakat, bu birimlerle bile, mikroişle lerin çoğunluğundaki dalgabiçimlerinin gerektirdiği yükselme süresi elde d zordur.

Şekil 4.11, bir transistörü standart TTL kapısı için aktif yukarı çekme gibi kull rak TTL kısıtlamalarının üstesinden nasıl gelindiğini göstermektedir. Göste devre, küçük değişikliklerle, Z80-tabanlı çeşitli mikroişlemci sistemlerinde elde etmek için kullanılmaktadır. Saat frekansı, kristal frekansı ile frekans bö oranı tarafından belirlenir. Uygun herhangi bir TTL, frekans bölücüsü kullanıl

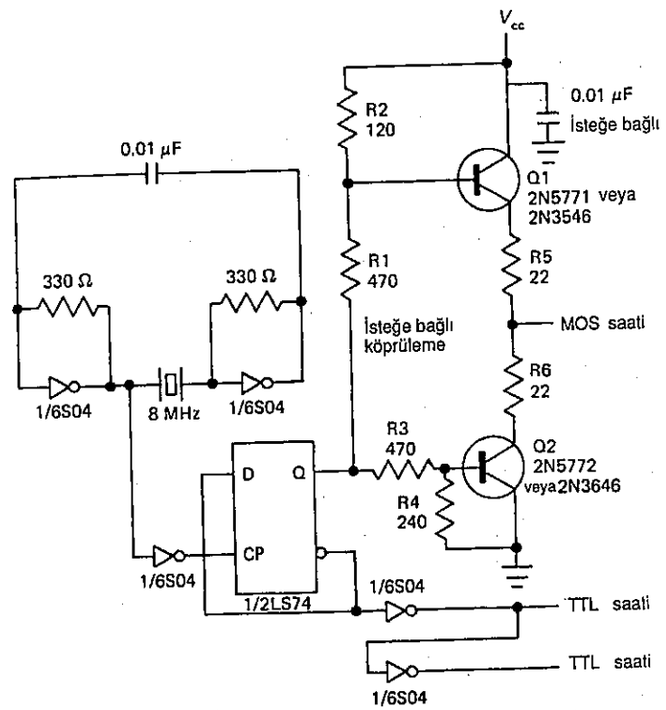


Şekil 4.11 Tek-fazlı saat üreteç devresi

lir. Bazı durumlarda osilatör frekansı, gerekli olan saatin frekansı ile aynıdır, böyle durumlarda frekans bölücü kullanılmayabilir. Aktif çekicide kullanılan transistör, saat dalgabiçiminin istenen yükselme süresini sağlayabilmelidir, en çok bilinen ve seçilen değer 2N3906'dır.

Z8000 mikroişlemcisinde kullanılması önerilen bir saat üreteç devresi Şekil 4.12'de görülmektedir (4.1 no.lu kaynak). Bu durumda saat dalgabiçim gereksinimleri bağımsız bir totem direği düzenleme sürücü devresi tarafından karşılanmaktadır.

Bir mikroişlemci tarafından saatın iki ya da daha fazla fazına gerek duyulursa, saat dalgabiçimlerinin üretilmesi için bir özel amaçlı entegre devre kullanılmasında hemen her zaman yarar vardır. Şekil 4.13'te 8080 MİB yonga takımının bir kısmını oluşturan bir 8224 saat üreteci gösterilmektedir. 8224, Ek F'de anlatılmıştır.



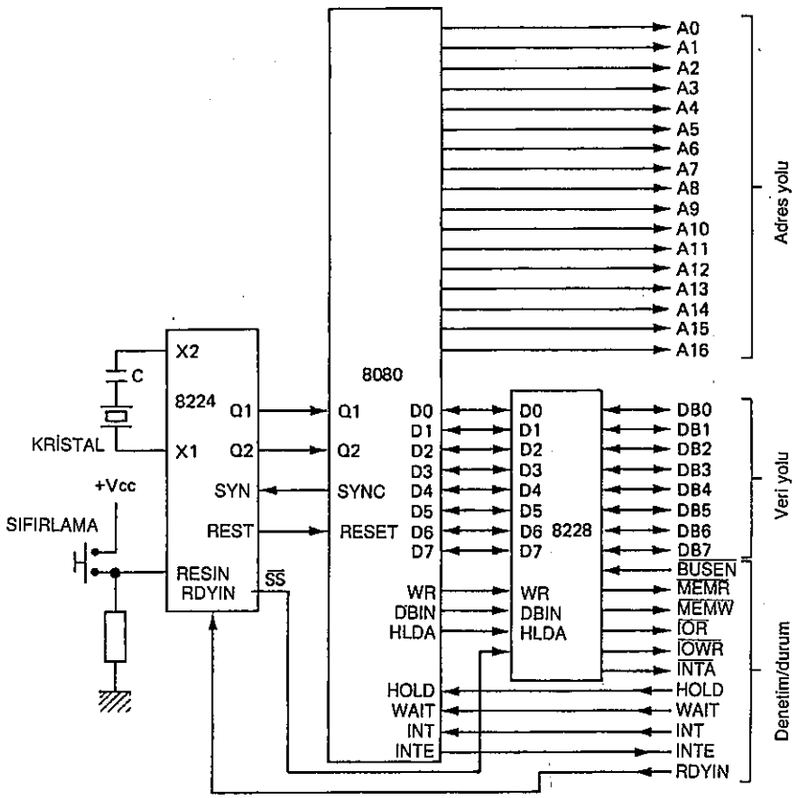
Şekil 4.12 Z8000 saat osilatörü ve sürücü devre

8224 içinde bir osilatör devresi ve 9-kademeli ferakns bölücü bulunur. Tipik bir sistemde 10pF'lik C değeri bulunan bir 18 MHz kristal, bir 2 MHz saat oluşturur.

Saatın her iki fazı da 8224 tarafından üretilmektedir. Ek olarak zamanlama iş-
lerini de gerçekleştirdiği için 8224, yalnızca bir saat üreteci değildir. 8080'in SYN
sinyali 8224 tarafından alınır ve, işlevi aşağıda açıklanan 8228 yol denetçisi iç
zamanlama strobu oluşturulmasında kullanılır. 8224'ün diğer işlevleri, sıfırla
işleviyle ve, 8080'in çok yavaş bellek birimleri ile yapılan okuma ve ya
işlemleri sırasında beklemesine olanak veren RDYIN sinyaliyle ilişkilidir.

Yol denetleyicileri ve tekilleyicileri

Burada örnek olarak alınan MİB-tipinde mikroişlemcilerden yalnızca Z80
M6800, adres ve/veya veri yol(lar)ı çoğullaması işlemini uygulamazlar. Bu
birimde, mikroişlemci bir saat üreteci ile birlikte MİB yonga takımını oluştu
Fakat 8080, durum ve veri sinyallerini bir 8-bitlik yolda çoğullar. Bunlar, bağım



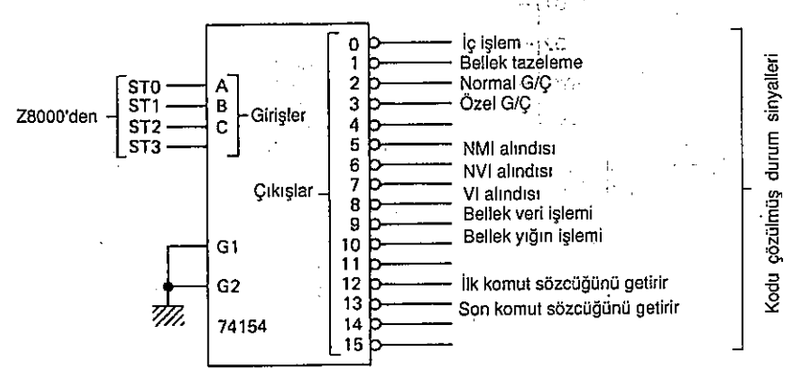
Şekil 4.13 8080 MİB yonga takımı

veri ve denetim/durum yolları elde etmek için Şekil 4.1'deki gibi tekillenmeli ya da birbirinden ayrılmalıdır. Bu tekilleme işlevini gerçekleştiren entegre devreler MİB yonga takımının bir parçasını oluşturur. Z8000'de adresler ve veri bir 16-hatlık yolda çoğullanırlar. Mikroişlemci, Şekil 4.1 deki gibi alışılagelmiş bir yapı içinde kullanılacaksa yine, tekilleme gereklidir.

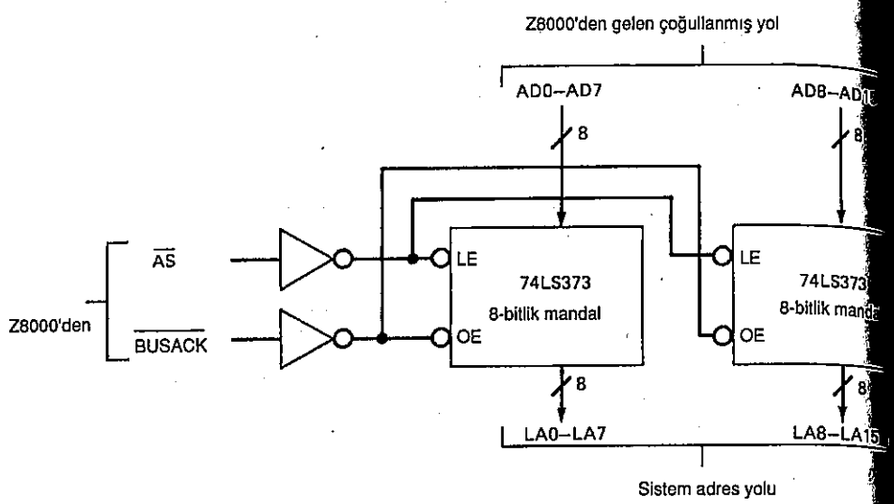
8080 MİB yonga takımı tam olarak Şekil 4.13'de gösterilmektedir. Burada 8228 yol denetleyicisi bağımsız veri ve denetim/durum yolları elde etmek için kullanılmıştır. 8228 Ek G'de anlatılmıştır. 8228 tarafından oluşturulan tipik durum hatları $\overline{MEMR}$ ve $\overline{IOWR}$'dir. Bellekten okuma yapılırken, $\overline{MEMR}$ sinyali etkindir. Bu sinyal, adresle birlikte, sistem bellek birimleri tarafından bellekten okuma işlemleri uygulandığı zaman kullanılır. Benzer biçimde $\overline{IOWR}$ da, bir çıkış portuna veri yazmak için OUT komutu çalıştırıldığı zaman etkindir. Tabii ki bu sinyal de çıkış port devresi tarafından kullanılmaktadır. Bazı denetim sinyalleri, HOLD, WAIT vb., çoğullanmazlar ve bu nedenle de 8080'e doğrudan ulaşabilirler.

Z8000 mikroişlemcisinde bağımsız denetim/durum yol hatlarının oluşturulması biraz daha farklı bir soruna yol açar. Bu mikroişlemci için durum bilgisinin, Tablo 3.2'de gösterildiği gibi, kodlanmış biçimde ST0-ST3 hatlarında taşındığı hatırlanacaktır. Orijinal durum sinyalleri üretmek için bu kod çözülmelidir. Şekil 4.14'de bir, 4 e 16 lık hat yapısındaki 74154 TTL kod çözücüsünün bu işlevi nasıl gerçekleştirdiği görülmektedir. Kod çözücülerle tanışık olmayan okuyucular bu noktada Kısım 4.4'e ve Ek Q'ya bakabilirler.

Z8000, çoğullanmış 16-hatlık adres/veri yolu kullandığından, aynı zamanda, açıklanan 8-bitlik mikroişlemcilerden de farklılık göstermektedir. Bu nedenle MİB



Şekil 4.14 Z8000 durum sinyali kod çözülmesi



Şekil 4.15

yonga takımının işlevlerinden biri adres ve veri hatlarını tekillemek veya ayı olmalıdır. Şekil 4.2'de gösterilen sistemde bu, 16-bitlik mandaldaki bir ad çoğullanmış yolda göründüğü zaman "yakalama" yoluyla yapılabilmek Z8000, özel olarak bu amaçla, bir zamanlama sinyalini, AS'yi üretir. Şekil 4.1 uygun bir düzenleme görülmektedir. Adres yoluna yapılan mandal çıkışlarını durumlu olduğuna dikkat ediniz. $\overline{\text{BUSACK}}$ sinyali, Z8000'in adres/veri denetimini bırakmak istediği her durumda etkin olmaktadır. Bu gerçekle mandalın üç-durumlu çıkışları yetkilerini yitirirler.

Şekil 4.2'de gösterilen sistemde veri yolu, çoğullanmış yolun basitçe tampoması ile elde edilir. Bu, adreslerle veri arasında olası bir çakışmaya yol açar nedenle bellek ve G/Ç portları o şekilde tasarlanmalıdır ki, bir adresin zaten m
S4.13 olduğu zaman aralığında, yola veri yerleştirmeye asla kalkışmamalıdırlar.

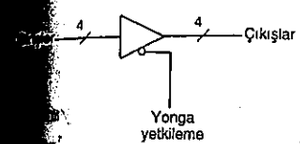
4.4 Adres kod çözücüleri

Mikroişlemci sistemlerinde kullanılmak amacıyla tasarlanmış birçok entegre renin daha yararlı olan özelliklerinden birisi de yonga seçim (CS) ya da y yetkilendirme (CE) diye adlandırılan, bir denetim hattı sağlanmasıdır. Yonga se hattı etkin değilken, sistem yollarıyla herhangi bir iletişim gerçekleşemez, durumda, sistem için, adı geçen birim yok gibi kabul edilir. Yonga seçim hatt

lunmayan basit birimlarde aynı etki, sistem veri yolunun sürücüsü olan üç-durumlu tamponların yetkilerinin kaldırılmasıyla elde edilebilir.

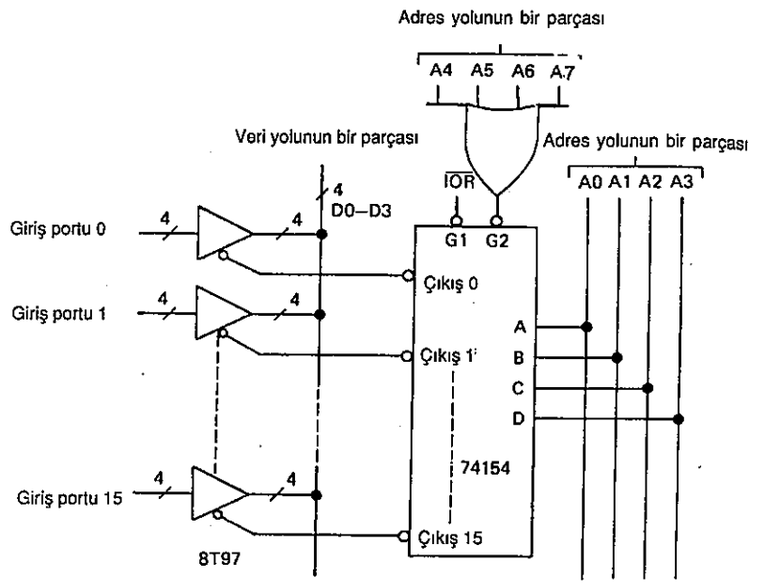
8080 veya Z80 gibi mikroişlemcilerde, G/Ç komutları bir port adresi tanımlarlar, bu adres mikroişlemci tarafından komut yürütümü sırasında adres yolunun en küçük değerlikli sekiz hattı üzerine konulur. Port adresini port devresini yetkilendirici bir sinyale dönüştürmek için bir adres kod çözücüsü gerekir. Ek Q da, 4-giriş/16-hat çıkışlı bir TTL kod çözücüsünü açıklayan bir veri sayfası bulunmaktadır. Bu 74154 birimi 4-bitlik bir kodu alarak kodun değerine göre 16 çıkıştan bir tanesini etkinleştirir.

74154'ün kullanımına bir örnek olarak, her biri 4 hattan oluşmuş 16 giriş portu gerektiren 8080-tabanlı bir mikroişlemci sistemini ele alalım. Her bir port için, Ek H'de anlatılan 8T97 tipinde basit bir üç-durumlu tampon kullanacağız. Her 8T97'nin altı kapısından yalnızca dördü kullanılacaktır. Ek H'den, 8T97 için üç-durumlu yetkilendirme hattının alçak-etkin olduğu anlaşılmaktadır. Doğruluk tablosunda bu, DIS4, yüksekte etkin yetkisizleme hattı olarak verilmiştir, bu tanımlama da aynı sonucu vermektedir. Yalnızca 1-4 kapılarını kullanarak, birimi tanımlamak için Şekil 4.16'daki sembolü kullanacağız.



4.16 8T97 tamponu 1-4 kapılarının sembolik olarak gösterilmesi

Şekil 4.17 bir IN işleminin yapılması sırasında belli bir portun seçim yöntemini göstermektedir. Yol yapısının, Şekil 4.13'teki 8080 MİB yonga takımı tarafından üretilene benzer olduğu kabul edilmiştir. Port adresleri 00-0F'dir. 74154'ün yetkilenmesi yalnızca; A3-A6'nın da alçak düzeyde olduğu sırada, $\overline{\text{IOR}}$ alçak düzeye çekilmesiyle, yani 8-bitlik adresin en büyük değerlikli dört biti 0 olması halinde gerçekleşen bir giriş işlemi sırasında mümkün olmaktadır. Eğer bu koşul sağlanmazsa, Şekil 4.17'deki portların hiç biri yetkilenmez, çünkü Ek Q'daki doğruluk tablosunda gösterildiği gibi 74154'ün tüm çıkışları yüksek düzeyde olacaktır. 00-0F port adresleri kullanılarak yapılan bir giriş işlemi sırasında, A4-A7 portlarının hepsi alçak düzeydedir, bu, G2'nin de düzeyini alçaltır; $\overline{\text{IOR}}$ alçak düzeydedir, bu da G1'i alçak düzeye çeker. Dolayısıyla 74154'ün



Şekil 4.17 Bir port adres kod çözücüsü olarak 74154

yetkisi ortadan kalkar. Birimin doğruluk tablosuna uygun olarak, 74154'ün yalnızca bir çıkışı alçak düzeyde olacak ve buna uyan üç-durumlu giriş portu kapısı yetkilenecektir, böylece sistem, veri yoluna verinin konulmasına izin verilecek.

G/Ç portlarının yetkilenmesi, adres kod çözücülerinin mikroişlemci sistemlerindeki uygulamalarından yalnızca biridir. Diğer uygulamalar arasında, Kısım 4.3'de anlatıldığı gibi, bir bellek dizisindeki bellek yongalarının yetkilenmesi ve denetim/durum sinyallerinin kodlarının çözülmesi sayılabilir.

S4.14

4.5 G/Ç birimleri

Kısım 4.3'te, MİB yonga takımının sistem tasarımcısına veri, adres ve denetim/durum yolları sağladığını gördük. Dış birimlerle mikroişlemci arasındaki iki yönlü veri aktarımı, veri yolu aracılığıyla yapılmaktadır. Veri işlemlerinin zamanlamasının doğru olmasını sağlayabilmek ve MİB'in, iletişim kurmak istediği dış birimi belirlemesine imkan tanımak için, özel-amaçlı G/Ç devreleri gerekmektedir. G/Ç devrelerinin yapısı ve karmaşıklığı, kullanılan mikroişlemciye ve veriyi alan ya da gönderen dış birime bağlı olacaktır. Tabii ki tek-yongalı mikrodenetleyicilerde G/Ç devreleri tek bit yonganın içinde bulunmaktadır. 8080, Z80, M6800,

Z8000 gibi diğer mikroişlemcilerde G/Ç devre konfigürasyonunun belirlenmesi sistem tasarımcısının işidir.

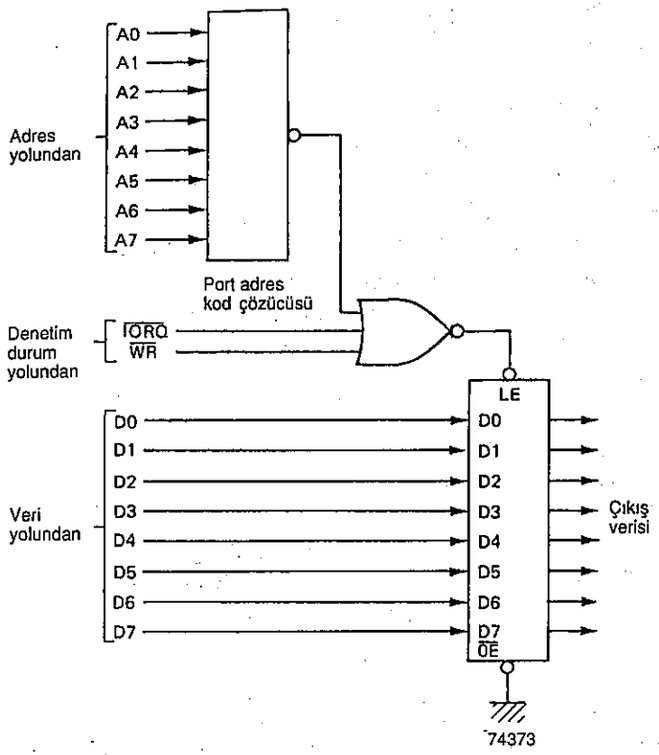
İlk olarak 8080, Z80 ve Z80000'i ele alacağız. Kısım 3.1'de, bu mikroişlemcilerin M6800'den, komut takımlarının veri giriş ve çıkış işlemleri için özel komutlar içermeleri nedeniyle farklı olduklarını söylemiştik. Bununla birlikte bulunan bir diğer özellik de, denetim/durum yolunda, G/Ç devrelerinin doğrudan kullanabilecekleri sinyalleri taşıyan hatların yer almasıdır. Bu hatlar, Z80'deki $\overline{IORQ}$ hattı örneğinde olduğu gibi, doğrudan mikroişlemci tarafından oluşturulabilirler. Bir başka seçenek de bunların, Şekil 4.13'teki 8080 sisteminde görülen 8228 gibi, yonga takımındaki diğer birimler tarafından sürülmeleridir. Bu birimler $\overline{IOR}$ ve $\overline{IOWR}$ sinyallerini sürerler. Benzer biçimde Z8000 sistemi, "normal" G/Ç ve özel G/Ç durum hatlarını sürmek için, Şekil 4.14'teki gibi bir kod çözücü kullanır.

G/Ç denetim/durum sinyallerinin kullanılabilmesi nedeniyle G/Ç devreleri çoğu kez oldukça basit olabilir. Bir giriş devresinin yalnızca bir üç-durumlu tampon takımından oluşabileceğini görmüş bulunuyoruz (Kısım 4.4). Şekil 4.17'de, 8080 $\overline{IOR}$ sinyalinin port adresinin kod çözücüsünü yetkilendirmek için nasıl kullanıldığı gösterilmiştir. Bu yetkilendirme daha sonra, uygun bir port adresiyle bir IN komutu çalıştırıldığı zaman, üç-durumlu kapıların yetkilenmesini sağlayacaktır.

Yukarıda giriş işlemleri için tanımlanan basit bir tampon kapı kullanımı, çıkış işlemleri için genellikle uygun değildir. Bunun nedeni, OUT komutu yürütümü sırasında, verinin, sistem veri yolunda yalnızca komut saykılının bir bölümü süresince var olmasıdır. Dolayısıyla herhangi bir çıkış devresi, çıkış verisini, yolda göründüğü zaman yakalamalıdır. Basit bir mandal bu amaç için uygun olabilir. Ek I'de uygun bir TTL sekizli mandal görülmektedir. Bilgi sayfası standart TTL 74373 devresinin F, ya da yüksek-hızlı türünü göstermektedir.

Z80 için bir çıkış devresi Şekil 4.18'de gösterilmiştir. LE alçak düzeyde olduğunda mandal saydamdır. Ek B deki, Z80 G/Ç zamanlama diyagramına bakıldığında, bir OUT komutu çalıştığı sırada $\overline{IORQ}$ ve $\overline{WR}$'nin alçak düzeyde olduğu görülecektir. Aynı anda, port adresi de adres yolundadır ve veri yolu, çıkış verisini taşımaktadır. Bu veri $\overline{IORQ}$ ve $\overline{WR}$ nin yüksek düzeye geçtiği ve böylece onu 74373 içine mandalladığı zaman içinde de yolda mevcut durumdadır.

74373 bazen, Şekil 4.17'deki 8T97'ye çok benzer şekilde bir giriş tamponu gibi kullanılır. Bu, LE alçak düzeyde iken mandallar şeffaflaştırılabildiği için mümkün olmaktadır. $\overline{OE}$, üç-durumlu denetim sağlamaktadır, bu nedenle devre, aynı bir sekizli üçdurumlu tampon gibi işlev görebilmektedir.

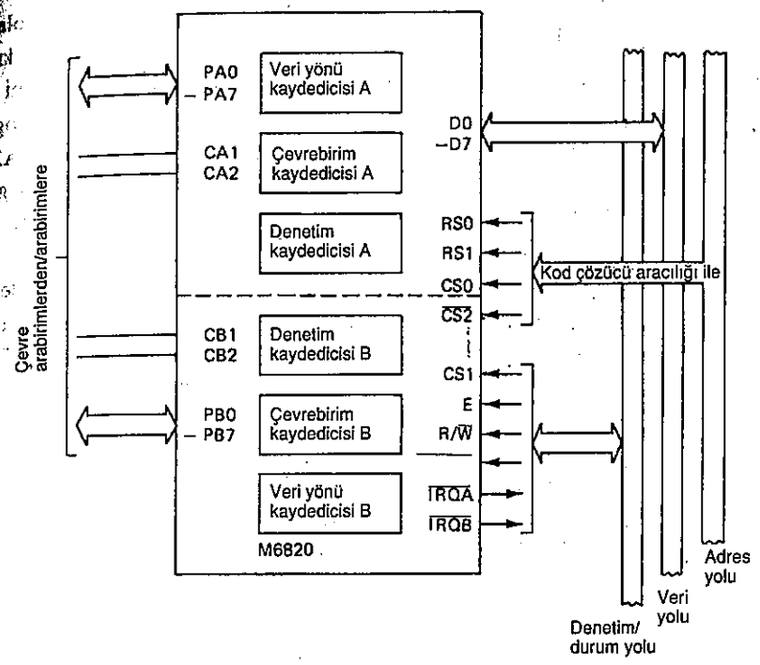


Şekil 4.18 Z80 sisteminde çıkış verisi mandalı durumunda 74373

Basit kaydedici ve tampon G/Ç devreleri G/Ç sorununa, düşük bileşen maliyetli çözüm sağlamaktadır. Bir diğer yaklaşım seçeneği ise, özel-amaçlı entegre devlerden yararlanmaktır. Günümüzde, mikroişlemci tiplerinin çoğunluğundaki G/Ç gereksinimlerini kapsayan çok çeşitli entegre devreler bulunmaktadır. Özel G/Ç entegre devreler kullanmanın hemen her zaman tercih edilmesinin söz konu olduğu iki durum aşağıdadır:

1 Mikroişlemci bellek-haritalı G/Ç kullandığında,
2 Seri veri G/Ç gerektiğinde

Birçok tasarımcı, basit tampon/kaydedici devrelerinin uygun olacağı durumlar
S4.15 bile özel G/Ç entegre devreler kullanmanın kolaylığını tercih etmektedir.



Şekil 4.19 M6820 nin iç yapısı ile dış bağlantılar

PIA'lar ve PIO'lar

Paralel G/Ç işlemlerinin uygulanmasında kullanılan entegre devreler çok çeşitli adlarla anılırlar, bunlardan en yaygın olanları çevresel arabirim adaptörü (PIA) ve paralel G/Ç (PIO) dir. Ek J'de tipik bir örnek, M6800 ailesinin bir üyesi olan M6820 PIA açıklanmaktadır. M6800, bellek-haritalı G/Ç kullanır, M6800 mikroişlemcisine dayalı hemen hemen bütün sistemlerde paralel G/Ç için M6820 gibi birimlerin kullanılması bu yüzdendir.

Özel-amaçlı G/Ç yongalarının çoğunluğu gibi, 6820 de programlanabilir özelliktedir ve çok çeşitli biçimlerde çalışabilme yeteneğindedir. Kullanıcı, M6820' yi, G/Ç işlemlerini gerçekleştirmeye başlamadan önce, gereken şekilde işlev görmesi için programlamalıdır. Bu çeşit diğer birimlerde yaygın olan uygulama, programlamanın, PIA içindeki kaydedicilere arzu edilen çalışma modunu tanımlayan kodların saklanması şeklinde yapılmasıdır. Bu işlem, M6800 mikroişlemcisi tarafından yürütülen bir programla yapılmakta ve buna, kullanıma hazırlama adı verilmektedir.

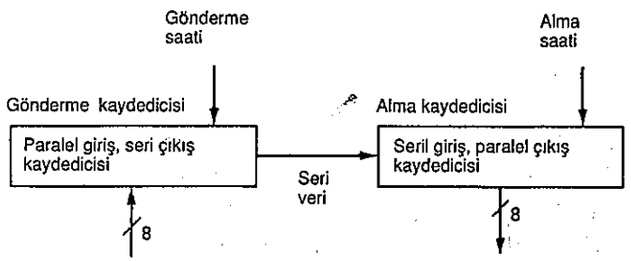
Şekil 4.19'da, M6820 PIA nın iç yapısı ile dış bağlantıları gösterilmiştir. Görüldü gibi birim, etkin biçimde, herhangi bir kombinasyonda giriş veya çıkış olabilen hat sunmaktadır. Hangi hatların giriş, hangi hatların çıkış olacağı, kullan hazırlama sırasında iki veri yönü kaydedicisine saklanan bir bit deseni ile s şekilde belirlenir. CA1 ve CB1 hatları, M6820 nin buna izin vermeye programl mış olması durumunda MİB'e iletilebilen, kesme sinyalleri taşıyabilir. CA2 ve C hatları da kesme sinyalleri taşıyabilir, fakat bunlar aynı zamanda çıkış gibi d
S4.16 görebilirler, bu da yine sistemin kullanıma nasıl hazırlandığına bağlıdır.

M6820, bellek-haritalı G/Ç için tasarlanmıştır ve bundan dolayı sistemdeki bel birimleriyle aynı denetim/durum hatlarını kullanır. M6800 MİB'i için M68 yükleme ve saklama komutlarını kullanarak üzerine yazılabilen ve üzerin okunabilen dört bellek konumudur. M6820 Ek J'de anlatılmıştır.

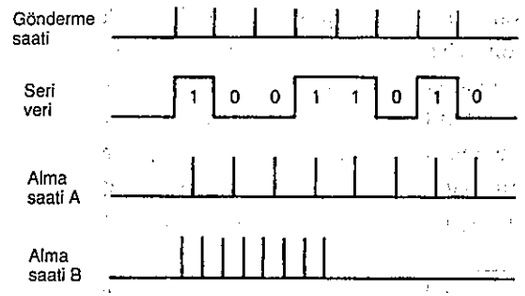
UART'lar ve USART'lar

Buraya kadar anlattığımız G/Ç tekniklerinin hepsi paralel veri aktarımı yöntem kullanmaktadır. 8-bitlik bir mikroişlemci için bu, bir G/Ç sözcüğünün 8 bit hepsinin aynı anda 8 bağlantı hattı aracılığıyla aktarımı demektir. Çevresel bir lerin bir çoğu, bunun aksine, veriyi seri biçimde yani bir kerede bir basam şeklinde, iletir. Bu durumda bağlantı bir toprak dönüşü ile birlikte bir kablo oluşur. Video görüntüleme terminalleri (VDT'ler), ve yazıcılar seri veri aktar kullanan birimlerin en sık karşılaşılan örnekleridir. Veri, mikroişlemci sist içerisinde sistem veri yolu boyunca paralel şekilde taşınır. Dolayısıyla, bir seri kaynağına veya tüketicisine arabirim gibi işlev gören bir G/Ç devresi, ve paralelden seriye sonra da yeniden paralele dönüştürmelidir.

Yukarıdaki kısa girişten sonra seri arabirim sorunu kolay gibi görünmektedir. B bir kaydırma kaydedicisi, verileri, paralelden seriye ve seriden paralele dönüşt bilir. Aslında özel-amaçlı entegre devrelerin kullanıma girmesinden önce bir



Şekil 4.20 Seri veri iletimi



Şekil 4.21 Bit eşzamanlaması sorunu

arabirimin uygulamaya katılması için gereken devre düzenlemeleri son derece önemsizdi. Bunun ana nedeni, seri veri aktarımı yapılırken, gönderen ve alan birimler arasındaki eşzamanlama sorunudur.

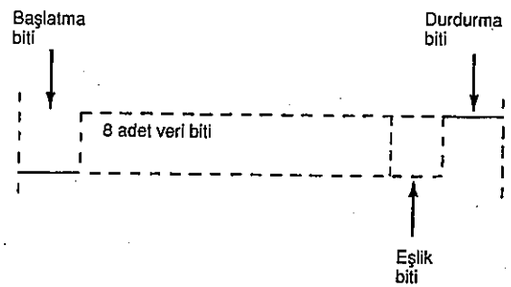
Böyle bir sorun bit eşzamanlaması ile ilişkilidir. Şekil 4.20 çok basit bir seri veri iletişim sistemini göstermektedir, burada 8-bitlik bir sözcük gönderme kaydedicisine paralel biçimde yüklenmektedir. Gönderme kaydedicisinin içeriği daha sonra, seri iletişim kanalına, gönderme saatinin belirlediği bir hızda birer birer kaydırılmaktadır. Basamaklar seri alma kaydedicisine alma saatinin belirlediği hızda kaydırılmaktadır. Alma kaydedicisi, alma saati darbesinin ortaya çıkış zamanlarında seri veri dalgabiçimlerini etkin biçimde örneklemektedir.

Şekil 4.21'de, 8-bitlik 10011010 veri sözcüğünün iletildiği durumu gösterilmektedir. Alma saati A gönderme saati ile aynı frekansa geldiği zaman, alma kaydedicisine doğru sözcük kaydırılır. Alma saati darbeleri gönderme saati darbelerine göre zaman içinde kayma gösterse de, olay yine aynı şekilde gerçekleşir. Alma saati B'de ise, dalgabiçimi, doğru frekansın iki katı kadar frekansla örneklenir, sonuçta alma kaydedicisinde kalan sözcük 11000011 dir.

Daha yüksek bir duyarlıkta eşit olan frekanslardaki alma ve gönderme saat osilatörleri tasarlamak zor olmadığından, Şekil 4.21'deki durumun bit eşzamanlaması sorununu abarttığı düşünülebilir. Fakat bu fazla yararlı bir çözüm değildir, çünkü, eğer seri veri sözcükleri peşpeşe iletilmekteyse, alma ve gönderme saatlerinin frekansları arasındaki herhangi bir farklılık, belli bir süre sonra, hatalara neden olacaktır. Gönderme saati osilatörünün, alma saati osilatörünün 999 darbe ürettiği bir zaman aralığında 1000 saat darbesi ürettiği bir durum buna örnek olarak gösterilebilir.

Veriyi seri olarak iletirken ortaya çıkan ikinci bir temel zamanlama sorunu sözcük eşzamanlamasıyla ilişkilidir. Şekil 4.20'deki basit sistemle bit eşzamanlı sorununun bir şekilde üstesinden geldiğimizi ve birbirini izleyen 8-bitlik sözcüklerin gönderme kaydedicisinden alma kaydedicisine seri olarak iletildiğini varsayalım. Veri dalgabiçimi sürekli bir bit akımından oluşacaktır. Bu anda alıcı tutmu yapar ve tek bir basamağı almazsa sistem kalıcı bir hata durumuna girer, çü alma kaydedicisi tarafından toplanan sözcükler, iletilen sözcüklerle aynı olm caktır. Bunun yerine, alınan sözcükler, iletilen bir sözcüğün son yedi basamağı bir sonraki sözcüğün ilk basamağından oluşacaktır.

Seri verinin eşzamanlamasına iki yaklaşım bulunmaktadır. Birincisi eşzam olmayan veri aktarımını kullanır. Bu yöntemin kullanılmasıyla veri sözcükl gönderme ve alma devreleri tarafından tanımlanan standart bir format kullanıl iletilirler. Şekil 4.22'de, uluslararası düzeyde kabul edilmiş RS232 formatı formunu gösterilmektedir. Görüldüğü gibi veri bitleri her zaman alçak düzeyde bir başlama biti ve her zaman yüksek düzeyde olan bir durdurma biti tarafı sonlanmıştır. Bir veri sözcüğünün başlangıcı bu nedenle, yüksek'ten-alçak'a ge tarafından belirlenmiştir. Başlangıç bitinden sonra 8 veri biti gelir, bunları da iste bağlı eşlik biti izler.

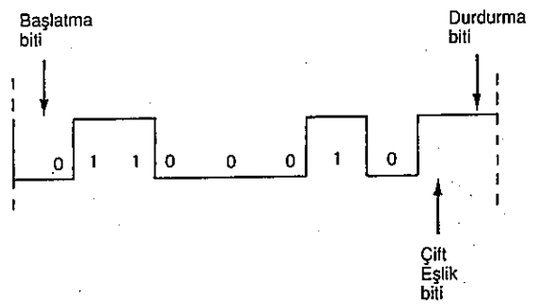


Şekil 4.22 RS232 eşzamanlı olmayan veri formatı

Eşlik biti, alan devrenin seri veride bir bozulma olup olmadığını denetleme aracılık eder. Eşlik denetlemeleri tek ve çift şeklinde tanımlanabilirler. Gönder ve alan devrelerde çift eşlik kontrolünün kullanıldığını varsayalım. Veri sözcüğ nün içindeki "1"ler göndericide sayılır. Daha sonra toplam "1" sayısını çift sa yapmak üzere sözcüğe, bir tane eşlik biti eklenir, aşağıda buna iki örnek verilmiş

Veri sözcüğü	*Eşlik biti*
11010011	1
01100110	0

İletilen sözcükte tek bir basamak bozulduysa eşlik biti, "1" i "0" a ya da "0" ı "1" e dönüştürmelidir, bu nedenle eşlik biti önemlidir. Bu dönüştürme sonucunda sözcükteki toplam "1" sayısına ya 1 eklenir ya da bu sayıdan 1 çıkarılır dolayısıyla bütün sözcükteki "1" lerin sayısı (veri+eşlik) çift bir sayıdan tek bir sayıya dönüşür. Bunu saptamak için alma devresi alınan sözcükteki birleri sayar. Aynı şekilde tek eşlik kontrolü uygulanır fakat burada eşlik biti toplam bir sayısını tek sayıya dönüştürür. Şekil 4.23'de çift eşlikli 46 veri sözcüğü için RS232 formatı gösterilmektedir.



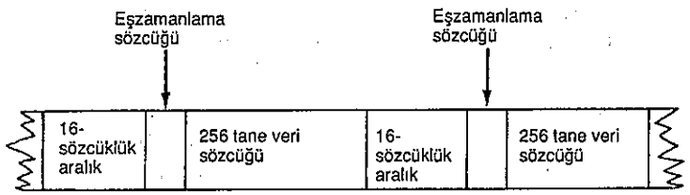
Şekil 4.23 Çift eşlikli 46 veri sözcüğü için RS232 biçimi

Bir asenkron seri veri iletim sisteminde, alıcı birim seri girişi tarayarak işe başlar ve ilk yüksek'ten-alçak'a geçişi bekler. Alınan sözcüğün örneklemesinin ortaya çıkacağı anları tesbit edebilmek için, bu geçişin ortaya çıkış zamanı bir zamanlama referansı olarak kullanılır. Bu işlem alıcıyı, her sözcüğün başında etkin şekilde eşzamanlı duruma getirir, eğer gönderme ve alma saat frekansları -en azından kabaca- eşitse, bit eşzamanlaması sorunu böylece giderilmiş olur.

Asenkron seri veri iletiminin uygulanması için, donanımın, Şekil 4.20'deki kaydırma kaydedicilerinden önemli ölçüde daha gelişmiş bir yapıda olması gerektiği açıktır. Seri veri iletişim sistemlerinde kullanılmak üzere özel-amaçlı entegre devreler bulunmaktadır, bunlar genel-amaçlı asenkron alıcı/vericiler (UART'lar) olarak bilinirler. Seri veri gönderici ve alıcıların işlevini gerçekleştiren özel-amaçlı G/Ç birimleri olarak kullanılabilirler. Bu bölümün ileriki kısımlarında söz konusu birimlerin özel bir örneği daha ayrıntılı olarak anlatılmıştır.

Asenkron seri veri iletişimine bir başka alternatif de eşzamanlı teknik tarafından sunulmaktadır. Bu yöntemde bit eşzamanlaması, veri dalgabiçimi boyunca bir saat dalgabiçimi gönderilerek elde edilmektedir. Dolayısıyla fiziksel iletişim bağlantısı, asenkron sisteme göre fazladan bir iletkeni gerektirmektedir. Alıcıda, saat, alınan bit akışı içindeki her bir basamağın yerini kesin olarak saptar.

Eşzamanlı seri iletişim bit eşzamanlaması sorununu ortadan kaldırır fakat sö eşzamanlaması sorununu çözümlemez. Bu sorun, iletilen sözcükler belirli biçime göre bloklar halinde düzenlenerek çözülür. Veri blokları önceden belirl bir eşzamanlama sözcüğü ile başlar ve bloklar birbirinden veri bulundurma fazlalık sözcüklerle ayrılmıştır. Şekil 4.24'de, bu çeşit tipik bir formatı gösteril tir. Eşzamanlı yöntemler, normalde, iletilecek veri miktarı büyük olduğu za ve/veya en yüksek iletim hızı gerektiği zaman kullanılırlar. Kullanılan blok form nedeniyle (Şekil 4.24'de gösterildiği gibi), eşzamanlı yöntemle her defasında ya da iki sözcük iletmek anlamlı olmayacaktır. Böyle bir durumda asenkron ile yöntemi çok daha uygundur.



Şekil 4.24 Tipik eşzamanlı seri veri biçimi

Eşzamanlı veri iletişimini uygulamak için gereken donanım, asenkron iletişim gerekli olandan farklıdır fakat daha az karmaşık değildir. Tahmin edilebileceği gib bu amaçla kullanılabilecek özel-amaçlı entegre devreler de bulunmaktadır. Çeşi adlarla anılırlar. Tipik bir örnek M6800 senkron seri veri adaptörü (SSDA-Synchr nous Serial Data Adapter) dür. Bu birim M6800 ailesindendir ve bu mikroişlemc nin bellek-haritalı G/Ç felsefesine uyar.

Bir seri arabirim birimi özel örneği, 8251, Ek K'da anlatılmaktadır. Bu birim genel-amaçlı eşzamanlı/asenkron alıcı/gönderici (USART-Universal Synchron us/Asynchronous Receiver/Transmitter) olarak bilinmektedir. Adından da anla lacağı gibi USART, hem senkron (eşzamanlı) ve hem de asenkron veri iletişi uygulamalarında kullanılabilir. USART'lar programlanabilen birimlerdir ve nedenle eşzamanlı ya da asenkron olarak isteğe göre, önceden kullanıma hazırla malıdırlar. Programlanan diğer entegre devrelerde olduğu gibi, kullanıma hazırla ma, USART'ın içindeki kaydedicilere uygun programlama kodlarını yüklenmesiyle gerçekleştirilir.

Bir vericiyi asenkron modda çalıştırırken, 8251, çeşitli seçenekler sunar. İletil sözcüğün içindeki veri bitlerinin sayısı 5 ile 8 arasında değişebilir. Şekil 4.22, 1 b aralıklı yüksek mantık düzeyi ile son bulan bir sözcüğü göstermektedir. 8251'd

kullanıcının isteğine bağlı olarak, 1 1/2 veya 2 bit aralıklara genişletilebilme olanağı vardır. Eşlik kontrolleri, kullanıcının isteğine göre uygulanır veya uygulanmaz.

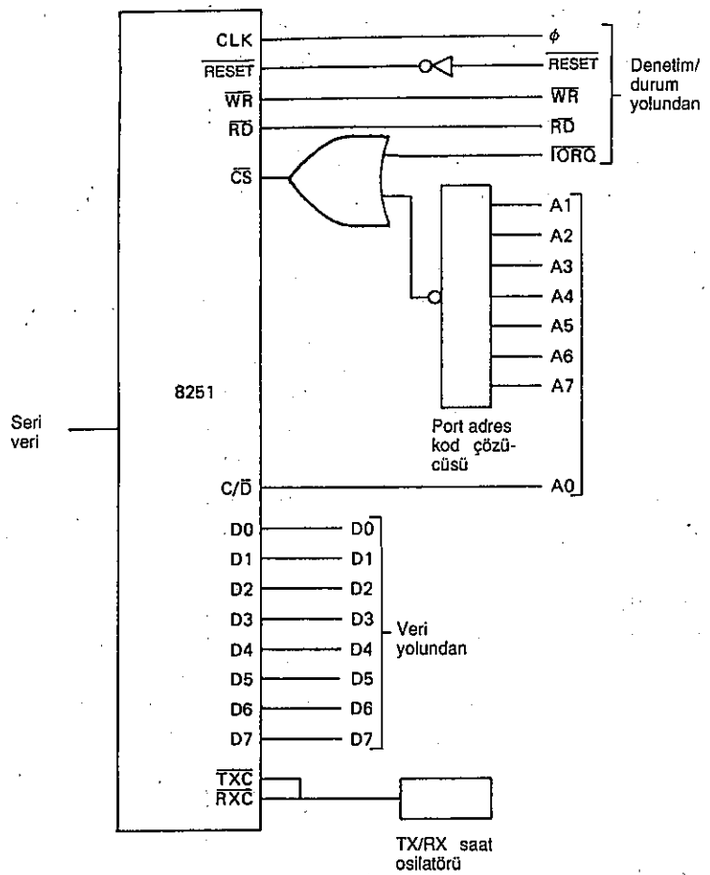
Bir asenkron alıcı olarak USART'ın, verici birimin sözcük biçimini tanıması için kullanıma hazırlanması gerektiği açıktır. USART, alınan sözcüğü paralel biçime dönüştürmek dışında ayrıca çeşitli kontroller de gerçekleştirir. Bunlar arasında:

1 Uygulanmışsa, eşlik kontrolü.
2 Çerçeve hata kontolü. USART, alınan her sözcükte, doğru durdurma bit(ler)inin beklenen yerde ortaya çıkıp çıkmadığını kontrol eder.
3 Ezme hatası sağlaması. Alıcı USART, mikroişlemciye sunduğu bir paralel sözcük birleştirir. Eğer mikroişlemci bu sözcüğü, bir sonraki paralel sözcük birleştirilmeden önce alamaz ise bir ezme hatası saptanır.

Asenkron moddaki uygulamada veri sözcüklerinin formatlanması USART tarafından otomatik olarak gerçekleştirilir. Programcı yalnızca, bu amaç için ayrılmış port aracılığıyla USART'a doğru veri sözcüğünün verilmesini sağlar. Başlatma biti, durdurma bit(ler)i ve bir eşlik biti eklenir ve formatlanmış sözcük daha sonra USART'tan seri biçimde çıkar. Alıcıda seri olarak alınır, başlatma biti, durdurma bit(ler)i ve bir eşlik biti ayrılır ve paralel bir veri sözcüğü alıcı sisteme sunulur. Bir durum hattı alıcı sistemi alınan sözcüğün hazır olduğu yönünde bilgilendirir. Bulunan hatalar, USART içindeki bir durum kaydedicisinde alıcı birim tarafından okunabilen bayrakların kurulmasına neden olur.

USART eşzamanlı modda çalışırken verinin biçimi, asenkron modda olduğundan çok daha fazla ölçüde programcının sorumluluğundadır. Eğer programcı USART-'ın, Şekil 4.24'de gösterilen 16 uzunluğunda ve tamamı-sıfır olan aralık oluşturucu bir sözcük akışı üretmesini isterse, program, mikroişlemciden USART'a, 16 uzunluğunda ve tamamı-sıfır olan bir sözcük akışı çıkaracak şekilde yazılmalıdır. Seçilen eşzamanlama sözcüğü ve onu izleyen veri sözcükleri benzer şekilde mikroişlemciden USART'a, doğru sırada gönderilmelidir.

Bir eşzamanlı alıcı olarak USART, "yakalama" modunda çalışabilir. Bu modda, alınan bit akışı belirlenen özel eşzamanlama sözcüğünü aramak için taranır. Sözcük bulunduğunda durum kaydedicisinde bir bayrak kurulur ya da bunun yerine, bir kesme oluşturulabilir. USART'ları, iki ardışık eşzamanlama sözcüğünü tanımayı gerektiren bir eşzamanlama şemasını uygulayacak biçimde kullanıma hazırlamak da mümkündür. Alınan veri sözcüklerinin alışılmış uygulama olan seriden-paralele dönüştürülmesine ek olarak bir alıcı USART, hata saptamak amacıyla eşlik kontrolü de yapabilir.



Şekil 4.25 Z80 yol hatları için USART 8251 arabirimi

Ek K'da, 8251 USART'ın, Şekil 4.13'de açıklanan standart yol yapısı özelliğinde bir 8080 sisteminde, bir G/Ç birimi gibi nasıl kullanıldığını anlatılmaktadır. 8251 USART, diğer mikroişlemci çeşitleriyle de kullanılabilir. Şekil 4.25, Z80 yol yapısına arabirimlenen 8251'i göstermektedir. Bazı uygulamalar, bu arabirim
S4.17 bağlantılarında küçük değişiklikleri gerektirebilir. Örneğin Şekil 4.25'de hiç kesme
4.21 kullanılmadığı anlaşılmaktadır, ancak bu tüm sistemlerde böyle olmayabilir.

4.6 Bellek birimleri

Bellek, her bilgi işlem sisteminde temel ve vazgeçilmez bir gerekliliktir. Büyüklük ve performans olarak uygun bir belleğin sağlanması bu nedenle, bilgisayar sistem tasarımcısının önemli bir görevidir. Bu bölümde, günümüzde mevcut olan, sistem

tasarımcısının kullanabileceği çeşitli bellek tiplerinin genel özelliklerini gözden geçireceğiz. Ayrıca belli bazı bellek birimlerini daha ayrıntılı olarak inceleyeceğiz.

Bilgisayar sistemleri tasarımcısının kullanabileceği bellek birimleri ve bellek teknoloji türleri her geçen gün artmaktadır, fakat bu bölümde özellikle ana bellekte veri saklama gereksinimine ağırlık verilecektir. Sistemlerin çoğunluğunda bu birincil bellek, çeşitli tiplerde tek entegre devrelerden oluşmaktadır. Yedek bellek özellikleri Bölüm 5'de anlatılmıştır.

Bellek tipleri

Ana bellek: Bazen yüksek-hızlı bellek olarak da adlandırılır. Bu adlandırma, bir mikrobilgisayar adresi ve veri yolu aracılığıyla doğrudan doğruya erişilebilir olan belleği anımsatmaktadır. Bir bellek referans komutunun yürütümü sırasında üzerinde okuma veya yazma işlemi yapılan bellek ana bellektir. Ana bellek ayrıca yürütülmekte olan programların komut kodlarını da saklar.

Yedek bellek: Saklama birimleri olarak da anılmaktadır. Mikrobilgisayar adresi ve veri yolu aracılığıyla doğrudan doğruya erişilebilir olmayan bellektir. Manyetik diskler ve bantlar yedek bellek ortamları için tipik örneklerdir. Yedek belleğe saklanan programların ve verinin, mikroişlemci tarafından kullanılabilmeleri için önce, ana belleğe aktarılmaları gerekir.

Rastgele-erişimli bellek (RAM): Okuma ve yazma işlemlerinin tüm konumlar için aynı sürede gerçekleştiği bir bellek türüdür. Tam olarak uygun olmasa da, RAM terimi aynı zamanda yaygın olarak, yalnızca okuma modunda çalışan ROM'un aksine hem okuma hem de yazma işlemlerinin gerçekleştirilebildiği ana belleği tanımlamak için de kullanılmaktadır.

Sıralı-erişimli bellek (SAM): Bir bellek konumunun fiziksel yerleşiminin veriyi okuma veya yazma işlemleri için harcanan süreyi etkilediği bellek türüdür. Belli bir adrese ulaşmak için, bu adrese gelininceye dek yer alan tüm adres konumlarının geçilmesi gerekir. Manyetik bant, disk ve kabarcıklı bellekler sıralı-erişimli bellek için tipik örneklerdir. Sıralı erişimli belleklerin ana belleklerde kullanımı çok çok azdır.

Sadece-okunabilir bellek (ROM): Günümüzde genellikle, üzerinde, üretim işlemleri sırasında kalıcı olarak verinin saklandığı yarıiletken birimleri anmak için kullanılır. Bazen bu terim, kullanıcının, kalıcı ya da yarı kalıcı olarak veri saklayabileceği birimler için de kullanılır (PROM, EPROM, ve EAROM'a bakınız).

Programlanır sadece-okunabilir bellek (PROM): Kullanıcının geridönüşüm olabilen bir işlem yardımıyla veri saklayabildiği yarıiletken bellek birimidir. çeşit birimlerin tipik bir örneğinde, veri saklamak için bir entegre devre iç isteğe bağlı olarak ayrılan eriyebilir-bağlantı özelliği bulunmaktadır. PROM zaman zaman, silinebilir birimleri tanımlamak için genel bir terim olarak kullanılmak (EPROM, ve EAROM'a bakınız).

Silinebilir programlanır sadece-okunabilir bellek (EPROM): Kullanıcının geri nüşümlü olarak veri saklayabildiği yarıiletken bir bellek birimidir. EPROM'un yaygın çeşitleri veriyi, bir entegre devre içindeki kondansatörlerde yükler biçim de saklamaktadır. Devreye bir süre için ultraviyole ışığı uygulanmasıyla bu y ortadan kaldırılabilir ve veri silinebilir. EPROM'lar bazen çoğunlukla-okunabi bellekler (RMM) diye de adlandırılırlar.

Elektriksel olarak değiştirilebilir sadece-okunabilir bellek (EAROM): Bazen ele triksel olarak silinebilir sadece-okunabilir bellek (EEROM) olarak da adlandır maktadır. Silme işleminin entegre devreye uygun bir gerilimin uygulanmasında etkilendiği EPROM çeşididir.

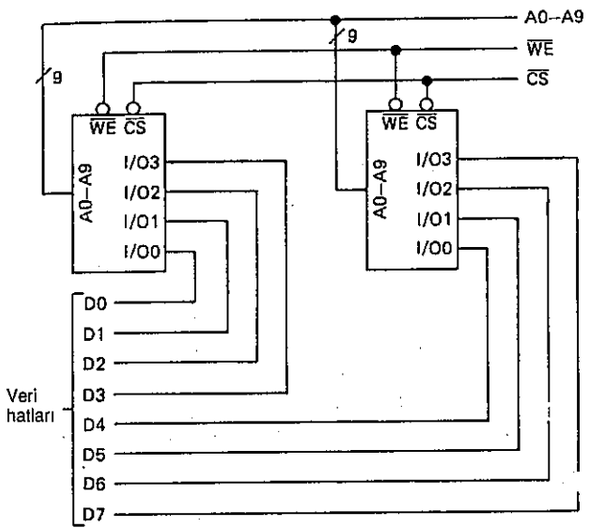
Erişim süresi ve bit maliyeti

Tasarımcılar için belleğin en önemli iki ölçütü, erişim süresi ve maliyetidir. Erişi süresi, bellek okuma sinyalinin etkin duruma gelmesi ile bellek çıkış hatlarında verinin kullanılabilir olması arasında geçen süredir. Mikroişlemcilerde bulun belleklerin, normalde, belirlenmiş bir en yüksek değerden daha kısa bir erişi süresine sahip olmaları gerekir. Bazı mikroişlemcilerde aynı zamanda bir beklem olanağı da bulunur, bu olanak, işletimin, yavaş bir bellekten veri beklenirken geçi olarak askıya alınmasına izin verir. 8080'in READY hattı ve Z80/Z8000'de WAIT hattı, bu amaçla kullanılan denetim girişleridir.

Bir bellek için ödenen fiyat genellikle, saklanan bilginin her bitinin maliyeti cinsinden dile getirilir. Hesaplama yaparken, bu fiyata yalnızca bellek birim(ler)i nin kendi maliyetlerinin dahil edilmediğinin hatırlanması gerekir. Monolitik (tek parça) entegre devrelerden oluşmuş bir bellek, baskılı devre kartlarının sağlanmasından d.c. güç kaynağına kadar çeşitli desteklere gereksinim duyar. Bu desteklerin maliyeti, entegre devre bellek bileşenlerinin maliyetini kolayca aşabilmektedir.

Statik RAM

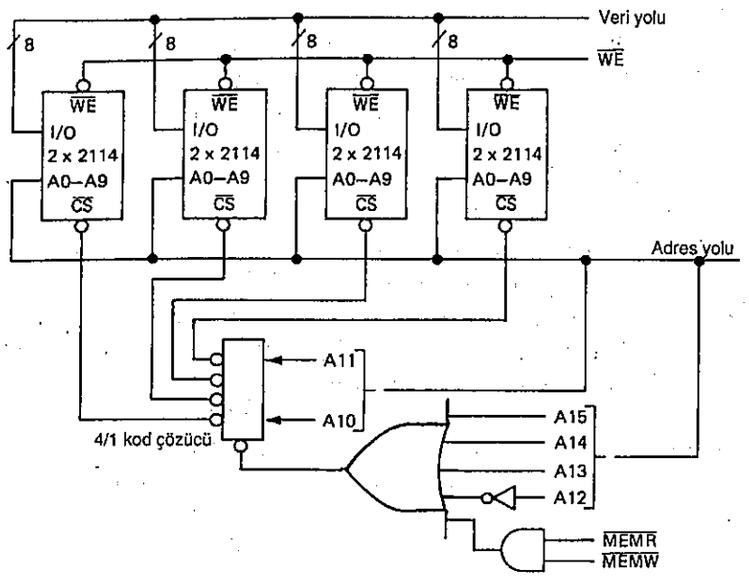
Bellek birimlerine ilişkin tartışmamıza statik RAM ile başlayacağız. Bu isim normalde bir monolitik (tek parça) entegre devre okuma/yazma birim çeşidi için kullanılır. Kitabın yazımı sırasında, bu tipteki entegre devrelerin çok popüler olanlarından birisi 2114'dü. Ek L'de bu yonganın veri sayfaları bulunmaktadır. Görüldüğü gibi her bir 2114, 4-bitlik 1.024 sözcük saklayabilmektedir. Dolayısıyla 8-bitlik bir mikroişlemci sisteminde 1 K'lık her bir bellek bloğu, 2114 tipinde iki tane entegre devreden oluşur, bunlardan birisi her sözcüğün en küçük değerlikli dört bitini, diğeri ise en büyük değerlikli dört bitini saklar. Tabii ki adres hatları iki yonga için de ortaktır. Yonga, $\overline{CS}$'nin alçak düzeye çekilmesiyle yetkilenir ve veri, adres konumlarından, $\overline{WE}$'nin durumuna bağlı olarak okunur ya da veya yazılır. 2114 tipte yongalardan oluşmuş, 1.024 X 8-bitlik bir bellek birimi Şekil 4.26'da görülmektedir. Daha büyük bellek birimleri bilinen yöntemlerle, Şekil 4.27'de gösterildiği gibi, bellek birimlerinin seçimi için üst-sıralı adres hatlarının kodları çözülerek oluşturulur.



Şekil 4.26 İki 2114 tarafından oluşturulan 1.024 X 8-bitlik bir bellek birimi.

Dinamik RAM

Yukarıda açıklanan, daha büyük bellek birimleri oluşturmak için bellek yongalarını birbirine bağlama yöntemi, dinamik bellek entegre devrelerinde de uygulanır. Ancak bazı sorunlar ortaya çıkar. Dinamik belleğin ilgi çekici özelliklerinden birisi,



Şekil 4.27 1000-13FF biçiminde adreslenen 1.024 X 8 statik bellek

tek bir entegre devre üzerine, statik birimlere göre daha fazla miktarda paketlene bilmesidir. Fakat bir entegre devrede bulunan bellek konumlarının sayısı arttıkça adres hatlarına ayrılan pin sayısı da buna paralel olarak artar. Bu durum, entegre devre üreticilerinin, her bir paketteki pin sayısını maliyet nedeniyle, olabilecek en düşük düzeyde tutma şeklindeki değişmeyen talepleriyle çelişmektedir. Adres hatlarının çoğullanması, bu sorunun giderek artan düzeyde benimsenen bir çözümüdür.

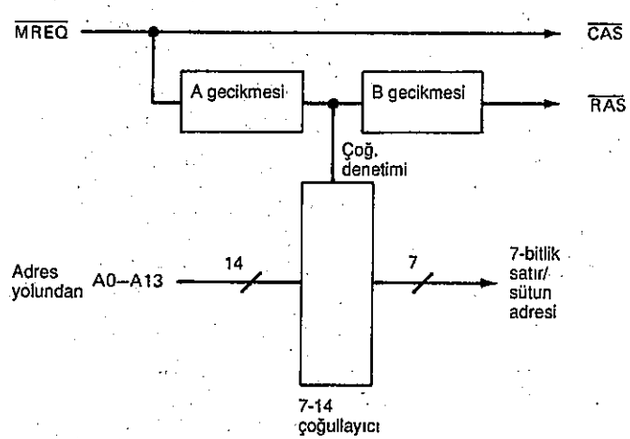
Ek M'de, çoğullanmış adres hatları içeren tipik bir dinamik bellek birimi için bilgi sayfası bulunmaktadır. 4116, 16.384 ayrı bellek konumu içermektedir. Bu nedenle belli bir adresi tanımlamak için, $16.384=2^{14}$ olduğundan, 14-bitlik bir adrese gerek vardır. 4116 içinde, entegre devre bellek birimlerinin çoğunluğunda olduğu gibi, bağımsız bellek hücreleri bir matris içinde düzenlenmişlerdir. Adres, hücrenin matris içindeki yerini belirtir ve bu nedenle, alışılageldiği gibi, bir satır adresi ve bir de sütun adresi olarak iki parçaya ayrılabilir.

4116 yı kullanmak için 7-bitlik satır ve sütun adresleri, 7 adres hattına farklı zamanlarda konulmalıdır. Adres hattına önce satır adresi yerleştirilir ve satır adres strobu $\overline{RAS}$ etkinleşir. 4116 yongaları içinde, $\overline{RAS}$, satır adresini bir kaydediciye

mandallar. Sütun adresi adres hatlarına konulur ve sütun adres strobu, $\overline{CAS}$, etkinleşir. Sütun adresinin mandallanması için uygun bir gecikme süresinin ardından, adreslenen bellek konumu üzerinde, $\overline{WRITE}$ hatının durumuna bağlı olarak, okuma veya yazma işlemi yapılır. Bu işlemlerin zamanlaması elbette çok büyük önem taşır, bu zamanlama Ek K'daki oku/değiştir/yaz çevrimine ilişkin zamanlama diyagramında gösterilmiştir. 4.5 nolu kaynağın 11. sayfasında, dinamik RAM'in çoğullanmış adreslemesi daha ayrıntılı olarak anlatılmıştır.

Satır ve sütun stroblarına olan gereksinimin bir sonucu olarak 4116, bir yonga seçim hattından vazgeçebilir, böylece, bir pin tasarruf edilmiş olur. Yonga seçimi, satır ve sütun strobları darbelerinin uygulanmasıyla etkin olarak gerçekleştirilir. Çoğullanmış adres hatlarının bir dezavantajı, $\overline{RAS}$ ve $\overline{CAS}$'ın doğru zamanda sağlanabilmesi için dış devre düzenlemesine gerek duyulmasıdır. Yonga seçimini gerçekleştiren bir devre, daha sonra, bu strobları seçilen yongaya ya da yongalara yönlendirir.

Çoğullanmış satır/sütun adreslemesi uygulayan bir belleğe gerekli olan denetim devresinin ilkeleri, Şekil 4.28'de gösterilmektedir. Devre çalışması bir Z80 $\overline{MREQ}$ sinyali tarafından başlatılmaktadır, fakat mikroişlemci sistemlerinin çoğunluğunda, bellek okuma/yazma işleminin gerektiğini belirten, aynı tipte bir durum hattı bulunur. Devre çalışmasını anlamak için okuyucunun, sırasıyla Ek B ve M'de verilen, Z80 ve 4116 okuma/yazma işlemleri zamanlama diyagramlarına bakması gerekir. Başlangıçta, A0-A6 adres bitlerini belleğin satır/sütun adres hatlarına çoğullayan $\overline{MREQ}$, yüksek konumdadır. $\overline{MREQ}$ alçak duruma geçtiğinde, bu,



Şekil 4.28 Çoğullanmış adres denetim devresi

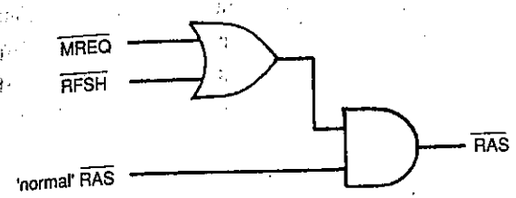
sütun adres strobunun elde edilmesini ve dolayısıyla da sütun adresinin A0-A6'y mandallanmasını sağlar. A erteleme periodundan sonra MUX denetim hattı alç düzeye çekilir, böylece A7-A13 adres bitleri satır/sütun adres hatlarına çoğullan Bir sonraki gecikme periyodu B'nin ardından $\overline{RAS}$ alçak düzeye iner, bunun sonu olarak satır adreslerini mandallar.

A ve *B* gecikme dönemlerini oluşturan devreler bu devrenin başarılı şekil çalışması için çok önemlidir. 4.5 nolu kaynağın I ve II. bölümlerinde çeşitli pra adres çoğullayıcı devreler bulunmaktadır.

Statik ve dinamik RAM arasındaki temel fark tazeleme gereksinimleriyle ilgili Dinamik RAM'lerde veriler, kondansatörler üzerindeki yük paketleri biçimin saklanmaktadır. Bu yük doğal olarak zaman içinde kaybolacağından, düzen aralıklarla yenilenmelidir. Tipik bir örnek olan 4116 için bu aralık, 2 msn aşmamalıdır. Tazeleme, her bir bellek konumuna sırayla erişilerek gerçekleştirili Çoğullanmış adreslemeli birimlerde bir satırın adreslenmesi o satırdaki tüm adre leri tazeleyecektir; bu ise işlemi basitleştirir ve hızlandırır. Uygulamada, belle için dışardan bir tazeleme sayıcı bulunmalıdır, bu sayıcı sayma işlemini yapark tüm adres satırlarını kapsayan sürekli bir çevrim içinde bulunur. Tazeleme sayıcın her durumunda $\overline{RAS}$ hattının etkinleştirilmesiyle bir tazeleme işlemi gerçekleştir lir.

Tazeleme işlemindeki temel uygulanabilme güçlüğü, tazeleme işleminin norm bellek okuma ve yazma işlemiyle birleştirilmesinin gerekli oluşundan kaynakla maktadır. Bu, iki yolla yapılabilir. Asenkron tazeleme, bir programın yürütüm sırasında, normal okuma/yazma işlemlerinin, tazeleme işlemi yapılabilmesi iç düzenli aralıklarla yetkisizlenmesiyle elde edilir. Birçok mikroişlemcide, bilg işlem uygulamasını bellek tazeleme işlemleri için askıya alan 8080 HOLD gib denetimler bulunmaktadır. Çalışması, belli bir birimin bilgi işlem çevrim zaman lamasına bağımlı olmadığından, çeşitli mikroişlemcilerde kullanılabilen standar belleklerin tasarımına izin vermesi, asenkron tazelemenin avantajıdır. Bellek taze lemesinin neden olduğu düzenli kesmelerin, program akışı zamanlamasını bozma ise bu tazeleme yönteminin dezavantajıdır.

Dinamik belleğin eşzamanlı tazelemesi, her bir komutun yürütümü sırasında mikroişlemci aktivitesiyle, tazeleme işlemini birbirine geçmeli olarak yapar. Tipik olarak tazeleme devresi, bir komutun yürütümü sırasında, MİB'den ya da MİB veri aktarımı işleminin yapılmadığı saat çevrimlerini saptamak için, mikroişlemci durum sinyallerini kullanır. M6800'ü örnek olarak alırsak, 1 saat periodu süresinc hiç bir veri aktarımı ortaya çıkmaz. Dolayısıyla 1 saat periodu sırasındaki herhang



Şekil 4.29 Z80 bellek tazeleme devresi

bir bellek tazeleme işlemi kullanıcıya bütünüyle şeffaf görünecektir. Dinamik belleğin M6800 ile arabirim bağlantısı kurması 4.2 nolu kaynağın 4-2.5 bölümünde tartışılmaktadır. Bir programın yürütümü sırasında, mikroişlemcinin normal çalışmasına karışmadığı için eşzamanlı tazelemenin ilgi çekici bir yöntem olduğu açıktır. Fakat bunun yanında belirli bir mikroişlemcinin özelliklerine bağımlı olması da bir dezavantajdır, bu, bir sistem için tasarlanan bir dinamik belleğin, farklı bir MİB'i olan bir başka sistemde kullanılamayacağı anlamına gelir.

Eşzamanlı tazeleme, Z80 ve Z8000 gibi bazı mikroişlemcilerle çok daha kolaylıkla uygulanmaktadır. Çünkü, tazeleme devrelerinin büyük bir bölümü, mikroişlemci yongasının içinde yer alır ve tazeleme işlemleri otomatik olarak, normal bilgi işlem çevrimi içinde geçmeli biçimde gerçekleştirilir. OPCODE komutu getirme zamanlaması diyagramından (Ek B), $\overline{RFSH}$ hattının, M1 veya $\overline{FETCH}$ çevriminin T3 ve T4 saat çevrimleri sırasında etkinleştiği görülmektedir. Aynı anda Z80, adres yoluna, Z80 yongası içinde yer alan bir tazeleme adres sayıcının çıkışını yerleştirir. Bu olanağı 4116-tabanlı bellek ile kullanmak için, "normal" RAS üreteç devresine yalnızca Şekil 4.29'da gösterilen devrenin eklenmesi gerekir.

Monolitik statik ve dinamik RAM birimlerinin açıklamasını, kısaca pratik konuları da tartışarak tamamlamak yararlı olacaktır. Statik RAM ile tasarım yapmak basitlik açısından çekicidir, bu basitliğin bedeli ise, bellek yongası sayısının artışı ve daha fazla enerji tüketimidir. Dinamik RAM'de daha az sayıda bellek yongası bulunur fakat tazeleme amaçları için dışsal devre düzenlemesi gerekli olabilir. Enerji tüketimi statik birimlara göre anlamlı ölçüde daha azdır. Büyük bellekler için fazladan eklenecek tazeleme devresi önemli bir sorun yaratmaz çünkü tüm bellek için yalnızca bir kere sağlanacak olan bir birimdir. Ancak şu da belirtilmelidir ki dinamik RAM, geçmişte, devre düzeni bakımından (genelde haklı nedenlerle) "yanıltıcı" olarak bilinirdi. Okuyucu, başarılı bir dinamik RAM oluşturmanın yalnızca birkaç entegre devreyi birleştirmek olmadığını unutmamalıdır. Düzgün yerleşim, yeterli güç kaynağı bağlantıları, düzenli aralıklarla köprüleme, birbirine komşu iletkenlerin birbirlerine etkide bulunmaların önlenmesi konularının hepsi

de çok önemlidir. Ayrıca, birçok dinamik RAM biriminin çok hatlı güç kaynağı gerektirdiği ve farklı kaynak gerilimlerinin, sıklıkla, belli aralıklarla açılıp kapatılmasının gerekli olduğu da unutulmamalıdır. Ek M'de görülebileceği gibi 4116'da durum böyledir.

Ek P, piyasaya yeni sunulan bir bellek yongasını göstermekte ve belleklerin gelecekte yöneleceği gelişme çizgilerinden birinin işaretini vermektedir. 6132 yarı-statik bir RAM'dir - tüm tazeleme devre düzenlemesi yonganın içinde olan dinamik RAM. Dolayısıyla hem statik, hem de dinamik RAM'in bir çok avantajını sunduğu söylenebilir. Yonga her ne kadar 32.768 bit saklarsa da, 4.096 8-bitlik sözcükler şeklinde düzenlenmiş olması, çoğullanmış adrese olan gereksinimi ortaya çıkarmaz. Bu birimin, düşük enerji tüketimi ve tek bir gerilim kaynağı gerektirmesi, diğer belirgin yönleridir.

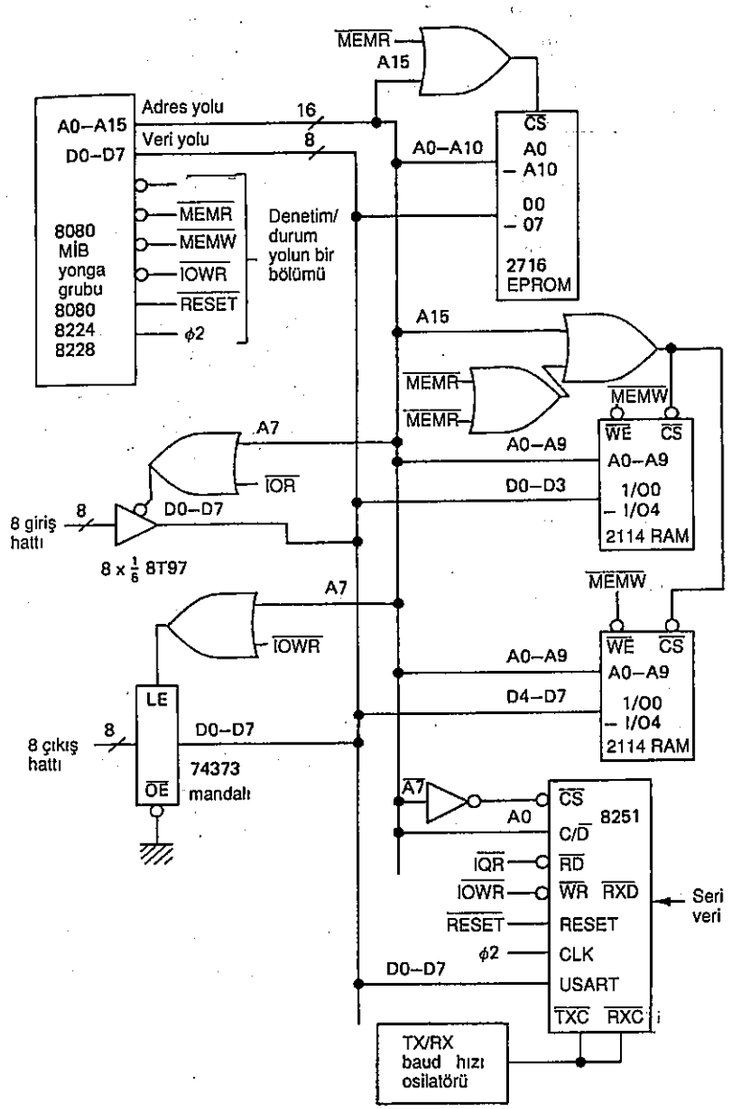
Çeşitli ROM, PROM ve EPROM tipleri ile bir mikroişlemci yolu arasında arabağlantı kurmak genellikle birkaç sorun ortaya çıkarır. Bunlar normalde statik birimlerdir ve dolayısıyla tazeleme sorunu oluşturmazlar. Ek N'de çok bilinen EPROM, 2716 açıklanmaktadır. $\overline{CS}$ hattı etkinleştirildiğinde 2716, adreslenen konumun içeriğini çıkış veri hatlarına yerleştirir. Okuyucu, 2716'nın iyi bir birim olduğuna, çalışmasının, daha önce anlattığımız RAM yongalarına göre daha yavaş olduğuna dikkat edilmelidir. Bu durum, bazı mikroişlemcilerdeki bekleme (wait) olanağının kullanılmasını gerektirebilir (Kısım 4.7'de anlatıldığı gibi). Doğal olarak, aynı anda
S4.22 başka bir sistem bileşeni veri yoluna veri yerleştirmeye çalışıyorsa, bir sadeceokunabilir birimin yetkilendirilmesi mümkün değildir.
4.25

4.7 Sayıcı-zamanlayıcılar

İleriki konularda sayma darbelerinin ve üretecinin zamanlama aralıklarının mikroişlemci sistemlerinde çok yaygın gereksinimler olduğunu göreceğiz. Özellikle eğer sistem, denetlenen sistemdeki olaylara hızlı bir biçimde yanıt vermesi gereken bir gerçekzamanlı denetleyicisi gibi işlev görmekteyse, bu gereksinimler önemli olmaktadır. Ek O'da, bir özel-amaçlı sayıcı devresi örneği, CTC görülmektedir.

Ek O'da gösterilen 3882 CTC, dört kanal içermektedir, bunlar giriş darbeleri sayıcıları olarak ya da sistem saatinden verilen zaman aralıklarını üreten zamanlayıcılar gibi çalıştırılabilirler. Daha önce anlatılan USART ve PIA gibi 3882 CTC de programlanabilir bir birimdir. Programlama ya da kullanıma hazırlama, CTC içine, gerekli işlem moduna uygun bit modelleri olan kaydediciler yükleyerek, bilinen şekilde gerçekleştirilir.

3882'nin ek bir avantajı, kesme üretebilme yeteneğidir. Z80 mikroişlemcisi ile kullanıldığında 3882, birden fazla CTC'yi bir papatya zinciri düzenlemesinde bağlayarak, bir kesme önceliği uygulama özelliğine de sahiptir.



Şekil 4.30 8080'li en küçük sistem

4.8 8080'li en küçük sistem

Şekil 4.30'de, bu bölüm içinde anlatılan birimlerin bir bölümünü kapsayan, 8080 tabanlı bir mikrobilgisayar sistemini gösterilmiştir. Sistem seri ve paralel G/Ç portları ile birlikte 2K'lık EPROM ve 1K'lık RAM içermektedir. Bellek ve G/Ç port adres alanının yalnızca küçük bir bölümü doldurduğu için, adres kod çözücü çok basit olabilmektedir. Örneğin 2716 EPROM'un $\overline{CS}$ hattı, $\overline{MEMR}$ ve A15 aynı anda beraberce alçak düzeye indiğinde enerjilenmektedir. Bu nedenle, A11-A14'teki mantık düzeyleri ne olursa olsun, bir EPROM konumunun içeriği okunacaktır. Böylece en alçak düzeydeki EPROM adresi 0000, 0800, 1000, 1800 vb., 7800 şeklinde adlandırılabilecektir. Bu bölgedeki diğer bellek elemanlarından hiçbiri ayrılmış adresler olmadığından bu durum son derece kabul edilebilir bir durumdur.

Şekil 4.30'da gösterilen sistemdeki RAM, 8000-FFFF adres alanında 1K blok olarak belirlenmiştir. Bu, 8000-83FF, 8400-87FF, 8800 8BFF vb. de olabilir. Benzer şekilde, 8-bitlik paralel çıkış portu 00, 01, 02, ..., 1F olarak adreslenebilir, bu giriş portu için de geçerlidir. Seri port 80, 81, ..., FF şeklinde adreslenebilir.

Kaynakça ve bibliyografi

4.1 P.Alfke, *A small Z8000 System*, Zilog application note (August 1979).
4.2 *M6800 Microprocessor Applications Manual*, Motorola.
4.3 W. Barden Jr, *The Z80 Microcomputer Handbook*, Howard W. Sams & Co. Inc. (1978).
4.4 *MCS-80 Users Manual*, Intel.
4.5 D. Bursky (Editor), *Memory Systems Design and Applications*, Hayden Book Company (1980).

Sorular

4.1 Bir mikroişlemci sistemde adres ve veri yollarının işlevlerini kısaca anlatınız.

4.2 Bir mikroişlemci sistemdeki tipik MİB denetim/durum hatlarına iki örnek veriniz.

4.3 TTL mantık kapılarına neden "akım çeken" dendiğini açıklayınız.

4.4 Tipik bir NMOS mikroişlemci çıkış hattının akım-çekme kapasitesi bir TTL kapısının aynı özelliğine göre nasıldır.

4.5 Hat kapasitansı ve giriş kapasitansının mikroişlemci sistemlerinde, darbelerin yükselme zamanlarını nasıl etkilediğini belirtiniz.

4.6 Tampon kullanımının mikroişlemci sistemlerde darbe yükselme zamanlarını nasıl arttırdığını açıklayınız.

4.7 Bir üç-durumlu tamponun olası çıkış durumlarını anlatınız.

4.8 Dört açık-kollektörlü ve iki-girişli VE kapısı çıkışlarının ortak bir yol hattına nasıl bağlandığını gösteren bir devreyi şematik olarak çiziniz.

4.9 Ek F'nin yardımıyla 8080 sisteminde, bir 8224 saat üretecinde kullanılan kristalin rezonans frekansını, gerekli saat frekansının 1 MHz olması halinde hesaplayınız.

4.10 8224'ün, 8080 mikroişlemci saatlerini üretmek dışındaki iki işlevini adlandırınız.

4.11 8228'in 8080-tabanlı mikroişlemci sistemindeki işlevini belirtiniz.

4.12 Z8000 mikroişlemci sisteminde ST0-ST3 hatlarından durum sinyallerinin nasıl türetildiğini gösteren bir devre şeması çiziniz.

4.13 Z8000 mikroişlemcisi tarafından üretilen AS sinyalinin işlevini açıklayınız.

4.14 Şekil 4.17'ye benzeyen, ve, 8T97 tamponlarıyla 74154 kod çözücüsünün, adresleri F0-FF olan 16 giriş portunu oluşturmak üzere nasıl kullanılabileceğini gösteren bir devre şeması çiziniz.

4.15 Yaklaşık daima, özel-amaçlı entegre devrelerin kullanımını gerektiren iki G/Ç işlem çeşidini anlatınız.

4.16 G/Ç işlemleri için bir M6820 PIA kullanıldığında giriş ve çıkış hatlarının nasıl tanımlandığını anlatınız.

4.17 UART ın işlevi nedir ?

4.18 Eşzamanlı ve asenkron seri veri iletişimi arasındaki farkı açıklayınız.

4.19 Bir dalgabiçim şeması yardımıyla RS232 seri veri formatını tanımlayınız.

4.20 Eşzamanlı ve asenkron seri veri iletişiminde sözcük ve bit eşzamanlaması nasıl elde edildiğini açıklayınız.

4.21 Asenkron olarak çalışırken 8251 USART'ın yaptığı hata denetimle anlatınız.

4.22 Statik RAM birimlerinin dinamik RAM birimlerine göre bir avantaj ve dezavantajını söyleyiniz.

4.23 Eşzamanlı ve asenkron dinamik RAM tazeleme işlemleri arasındaki fa açıklayınız.

4.24 Dinamik RAM kullanımında Z80 ve Z8000 mikroişlemcilerinin sahip old ğu özel avantaj nedir ?

4.25 Dinamik RAM yonga tasarımcılarının sıklıkla satır ve sütun adres çoğull masını tercih etmelerinin nedenlerini açıklayınız.

Bölüm 5 Tipik mikroişlemci sistemleri

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde; farklı mikroişlemci sistemlerini, aşağıdaki tipik uygulama örneklerinde*

a) Uygun eleman teknolojisi
b) Ortalama yonga sayısı
c) Toplam maliyet
d) Tahmini üretim hacmi

ölçütlerini göz önünde bulundurarak analiz edebilmelisiniz:

1 Küçük bir özel sistem örneği olarak bir çamaşır yıkama makinesi denetleyicisi,
2 Orta-boyutta yarı-özel sistem örneği olarak bir kelime işlemci,
3 Genel-amaçlı sistem örneği olarak bir mikroişlemci geliştirme sistemi

5.1 Özel ve genel-amaçlı sistemler

Bu bölümde, bir mikroişlemci-tabanlı sistemde gereken işlem miktar ve çeşitlerinin, mikroişlemci ve diğer sistem elemanlarının seçimini ne şekilde etkilediğini araştıracağız. Ayrıca sistem düzenlemesini de ele alacağız. Düzenleme terimi bir sistemin bileşenlerini ve bunların aralarındaki bağlantıların genel özelliklerini tanımlamak için kullanılır. Araştırmamız, büyük oranda özelleşmiş ya da bir başka deyişle özel sistemlerden, çok daha genel-amaçlı sistemlere kadar geniş bir çerçevede olacaktır. Özellikle bu bölümde anlatılanlar, bu sistemlerin genel doğası ile ilgilidir, ve, bilgisayar bir mikroişlemci olsun veya olmasın, bilgisayar-tabanlı sistemlerin çoğunluğu için geçerlidir. Bu nedenle "bilgisayar" sözcüğünü, mikroişlemciler, mini-bilgisayarlar ve daha büyük diğer makineler için genel bir ad olarak kullanmaktayız.

Alışılagelmiş tipte büyük bir bilgisayar, genel-amaçlı bilgi işlem sisteminin iyi bir örneğidir. Bu çeşit bir sistem, çok çeşitli bilgi işlem görevlerinin yerine getirilmesinde yaygın olarak gereken bir sistemdir. Bu görevler, kullanıcıların kendilerinin yazdıkları programların derlenmesi ve çevrilmesi işlemlerine ek olarak, iş, ticaret, bilim, istatistik vb. konularla ilişkili özel-amaçlı programların çalıştırılmasını kapsayabilir. Büyük, genel-amaçlı bilgi işlem sistemlerinin kullanıcıları, gerekli olan belli bir işlemi tanımlamak için, bilgisayar ile çok gelişmiş, ve çoğu zaman karmaşık bir yolla iletişim kurarlar.

Operatör iletişimi

Genel-amaçlı makinelerle karşılaştırıldığında özel bilgisayarlar bir konuda özelleşmişlerdir ve yalnızca çok sınırlı sayıda bilgi işlem görevlerini gerçekleştirebilirler. Bir ev gerecini denetleyen mikroişlemci buna örnektir. İşlevleri sınırlı olduğunda, kullanıcı belli bir bilgi işlem görevini tanımlarken doğal olarak sınırlı sayıda seçeneğe sahiptir. Dolayısıyla kullanıcı ile bilgisayar arasındaki iletişim düzeyi genel-amaçlı sistemlerdekinden çok daha düşüktür. Yalnızca belli bir bilgi işlem görevini gerçekleştirmek için tasarlanan bir bilgi işlem sisteminde kullanıcının müdahalesi açma-kapama anahtarını çalıştırmaktır. Diğer durumlarda ise ayrılan edilen işlevin seçimi basit bir ön panel anahtarı aracılığıyla yapılır. Hatta kullanıcının, sistemin bir bilgi işlem birimi içerdiğinin farkında olması bile gerekmez.

Çevresel birimler

Bir bilgisayar bir ya da birkaç bilgi işlem görevi için özel olarak ayrıldığında, çevresel birimlerin sayısı ve çeşitliliği, bu görevlerin ilişkili olduğu birimlerle sınırlıdır. Bu durum, bilgi işlem sisteminin gerektirdiği giriş ve çıkış olanaklarının çok kesin biçimde tanımlandığı anlamına gelir. Diğer taraftan bir genel-amaçlı bilgi işlem sisteminin, çok daha fazla esnek olması gerekebilir. Birçok farklı bilgi işlem işlevini gerçekleştirebilmek için daha çok sayıda ve çeşitte çevresel birim gerekli olabilir. Bilgi işlem sisteminin gereksinimleri değiştikçe çevresel birimler de zaman zaman değişebilir.

Program saklama

Özel ve genel-amaçlı bilgi işlem sistemleri arasındaki en belirgin fark belki de programın bellekte hangi yolla saklandığında yatmaktadır. Tanımıyla uyumlu olarak gerçek anlamda özel bir sistem, var olma amacı bütünüyle önceden belirlendiği için, daima aynı programı çalıştırır. Dolayısıyla program doğal olarak, sadece-okunur bellekte (ROM) saklanır. Bir genel-amaçlı bilgi işlem sisteminde ise, sistemin denetimi ile ilgili bazı temel programlar ROM'da saklanabilir, fakat diğer programlar, bir manyetik disk ya da manyetik bant belleği gibi kalıcı bir yedek saklama ortamında saklanırlar. Belli bir programın yürütümü gerektiğinde, bu program önce yedek saklama ortamından birincil rastgele-erişimli belleğe (RAM) aktarılır. Genel-amaçlı bilgi işlem sistemleri tarafından çalıştırılan programlar oldukça büyük olabildiği için, oldukça büyük boyutta bir ana RAM bellek, bu sistemlerin ortak özelliğidir. Özel sistemlerdeki RAM gereksinimi, bunun aksine, genellikle oldukça küçüktür, çünkü bir yığın alanı ve işlenmiş verinin geçici olarak saklandığı bir saklama ortamı ile sınırlanmıştır.

Sistem özellikleri

Özel ve genel-amaçlı bilgi işlem sistemlerinin ana özellikleri Tablo 5.1'de özetlenmiştir. Bu tablo, bir sistemi özel veya genel-amaçlı olarak kesin biçimde tanımlamak için bir yol gösterici gibi algılanmamalıdır. Gerçekte bir çok sistem, belli bazı örnekleri ele aldığımızda göreceğimiz gibi, kısmen özel ve kısmen de genel-amaçlı olarak kullanılabilmektedir.

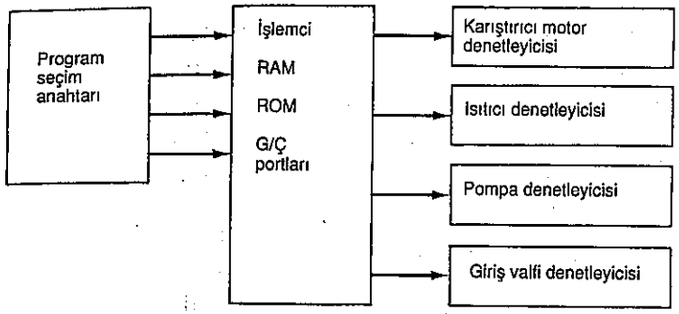
Tablo 5.1 *Bilgisayar sistemlerinin özellikleri*

Özel	*Genel-amaçlı*
Bir ya da çok az sayıda, belirli bilgi işlem görevi	Kullanıcı tarafından yazılan programlar da dahil işlem görevi
Program kalıcı biçimde ROM da saklanır.	Programlar yedek bellekte saklanır ve yürütüm öncesi RAM'a aktarılır.
RAM genellikle oldukça küçüktür.	RAM, nisbeten büyüktür.
G/Ç olanakları belirli ve sınırlıdır.	G/Ç olanakları esnektir.
Alt düzeyli kullanıcı iletişimi	Yüksek düzeyli kullanıcı iletişimi

5.2 Mikroişlemci-denetimli bir çamaşır makinesi

Bir çamaşır makinesi denetleyicisi, yüksek düzeyde özelleşmiş mikroişlemci sistemine örnektir. Tipik bir düzenleme, Şekil 5.1'de şematik olarak gösterilmiştir. Bu sistemde mikroişlemcinin işlevi, makinede yapılabilecek çeşitli yıkama programlarının gerçekleştirilmesi için denetim sinyalleri üretmekle sınırlandırılmıştır. Kullanıcı bir yıkama programı seçer ve daha sonra mikroişlemci, uygun zamanlarda pompa, karıştırıcı, ısıtıcı vb. birimleri harekete geçiren denetim sinyalleri dizisini üretir.

Şekil 5.1'den, ilk bakışta, çok sayıda giriş ve çıkış portlarına gereksinim olduğu sonucu çıkarılabilir. Oysa bu gereksinim oldukça düşük düzeydedir. Su pompası, su giriş valfı, karıştırıcı motor ve ısıtıcıyı denetleyen dört çıkış, bu birimlerin çalışmasını başlatan ve durduran mantık düzeylerini taşıyan hatlardır. Mantık düzeyleri, röle ya da daha karmaşık elektronik anahtarları devreye sokmak için kullanılır. Yüksekhızda dönmeli çalışmanın istenmesi durumunda beşinci bir hat



Şekil 5.1 Çamaşır makinesi denetleyicisi

gerekecektir. Isıtıcı denetiminin de yine, basit açma/kapama tipinde olması m temeldir, çünkü gereken düşük düzeydeki duyarlılık için karmaşık oransal sıcaklık denetim sistemine gerek bulunmamaktadır.

Ön paneldeki, anahtarlarla ilişkili giriş hatları sayısı, kullanıcıya sunulan yıka programının seçme düzeyine bağlıdır. Eğer 16 seçenekli bir program kullanılabi yorsa denetim anahtarlarının çıkışı 4-bitlik ikili bir koda dönüştürülebilir. Sıcak denetim sisteminin istenen kesinlik düzeyi yüksek değilse, karmaşık bir sıcak transdüserine (dönüştürücüsüne) gerek yoktur. Uygulanan tekniklerden birisi, s caklık önceden belirlenen değere ulaştığında kapanan ve böylece, mikroişlem giriş hatlarından birindeki mantık düzeyinde değişikliğe neden olan, bir ya da da fazla sayıda anahtar kullanmaktır.

Sistem özellikleri

Bu sistemdeki mikroişlemciden beklenen işlevleri incelediğimizde programı esnek olmayan ve aynı zamanda, basit bir program olduğunu görmekteyiz. Bu programın, orta büyüklükte bir ROM içinde saklanacağı anlamına gelmektedi ROM-yerleşik programın yürütümü sırasında yığın ve geçici saklama ortamı olar kullanılmak üzere ayrıca bir de çok küçük bir RAM gerekir. Herhangi bir karmaşı hesaplama gerekmediği ve çamaşır makineleri nispeten yavaş çalıştığı için, en yavaş mikroişlemcinin bile yeterli işlem gücünü sağlayabilmesi mümkündür. Gör düğümüz gibi mikroişlemcinin G/Ç gereksinimleri de fazla değildir. Bu tip denet leyicilerin çoğunluğunda sekiz giriş ve sekiz çıkış hattı yeterli hatta fazladır.

Üretim maliyetleri

Çamaşır makinesi gibi tüketim ürünlerinde kullanılan bir mikroişlemci sisteminin en önemli gerekliliği belki de düşük üretim maliyetidir. Ekonomik olarak tutulabilmesi için mikroişlemci-tabanlı sistem belirli yıkama programlarını gerçekleştirebilmeli ve üretim maliyeti, örneğin motorize anahtar esaslı bir denetleyiciden daha az olmalıdır. Tüketim ürünleri normalde, çok sayıda üretilirler, bu, geliştirme maliyetlerinin üretim maliyetlerinden daha az önemli olabileceği anlamına gelmektedir. Çünkü toplam geliştirme maliyeti çok sayıda birim arasında paylaştırılabilir, böylece sonuçta her birimin maliyete katkısı nispi olarak düşük düzeyde kalır. Üretim maliyetlerinin malın fiatına etkisi daha doğrudan bir etkidir ve daha fazladır.

Elektronik sistemlerin üretiminde maliyet, çoğunlukla, büyük oranda sistemdeki entegre devre sayısı tarafından belirlenir. Çünkü, baskılı devre kartı, enerji kaynağı büyüklüğü, montaj maliyeti gibi unsurların hepsi, entegre devre sayısına bağlıdır. Bu durum, her bir sistemin entegre devre sayısı ve üretim maliyeti arasında açıkça görülebilir bir ilişki demektir. Entegre devre sayısını olabilecek en az düzeyde tutma gereksinimi, çamaşır makinesi denetleyicisinde kullanılan mikroişlemcinin seçimini büyük oranda etkileyecektir.

Özet

Yukarıda anlatılan etkenler özetlenirse, en az sayıda entegre devre içeren performansı orta düzeyde bir mikroişlemci sistemin, çok güçlü bir seçenek oluşturduğu ortadadır. Bu bizi doğrudan doğruya, Bölüm 2'de anlatılan ve "tek-yongalı" olarak adlandırılan mikroişlemcilerden birini, örneğin COP420L'yi kullanmaya yöneltmektedir. Tek-yongalı birimler, burada anlatılan çamaşır makinesi denetleyicisi gibi uygulamalar için özel olarak tasarlandıklarından, ulaştığımız sonuç şaşırtıcı değildir. Bu birimler bir entegre devre paketi içinde, bir merkezi işlemci, ROM, RAM ve G/Ç portları içerirler. Programlar, entegre devre imalatı sırasında ROM'a saklanır ve çamaşır makinesi imalatçısı, bu nedenle, mikroişlemcilerin "sipariş" versiyonlarını satın alır. Tek-yongalı mikroişlemcileri satın alırken, müşterinin, kendi programını içerecek şekilde yapılan ROM değişikliğinin maliyetini kapsayan bir "maske" bedeli ödemesi gerekir. Bu ilk bedel bir defa ödendikten sonra, her bir mikroişlemcinin birim maliyeti genelde çok düşük olur. Kitabın yazımı sırasında bu bedel yaklaşık 500 ingiliz sterlininden başlamakta ve bu ödeme sonrasında her bir birimin maliyeti ise 2 ingiliz sterlininden daha az olabilmekteydi. Bu birim maliyetin gelecekte daha da azalması beklenebilir. 1979 yılında, burada anlatılan tipteki tek-yongalı mikroişlemcilerin dünyadaki pazarı 50 milyon birimden daha fazla olarak hesaplanmıştır.

S5.5

Aşağıdaki liste, kitabın yazımı sırasında mevcut olan çok fazla sayı ve çeşit tek-yongalı mikroişlemcilerden bir kısmını göstermektedir:

Rockwell PPS - 4/1 ailesi
Western Digital 1872/2272
Texas TMS1000 ailesi
AMI S2000/S2150/S2200/S2330
National COP400 ailesi
Oki MSM5840
NEC COM-42/43/44/45
NEC PD75XX

Listede, yalnızca, her defada 4 bit veri işleyen birimler yer almaktadır. Bu özel 8-bitlik tek-yongalı mikroişlemcilerle karşılaştırıldığında sınırlı bir performa ortaya çıkar, fakat maliyet daha düşüktür. Çamaşır makinesi denetleyicisi gibi uygulamalarda 4-bitlik bir mikroişlemci yeterli işlem gücünü, hatta daha da fa sını sağlar.

Üretim hacminin maske programlı bir tek-yongalı mikroişlemci kullanımını gerektirmediği uygulamalarda maliyeti düşüren başka seçenekler de vardır. Tek yonga içinde silinebilir sadeceokunur bellek bulunduran mikroişlemciler mevcut tur. Bunların maliyetleri, büyük miktarlarda satın alındıkları zaman maske programlı yongalardan önemli ölçüde daha yüksek olsa da, sistem tasarımcı tarafından programlanabilme avantajını getirirler. Az ya da orta miktarlarda, maske-programlı yongaların yüksek maskeleme bedelleri, programlanabilir birimi düşük maliyetli seçenekler haline getirmektedir.

5.3 Kelime işlemci

Kelime işlemciler, veriyi metin karakterleri biçiminde işlemek üzere tasarımlan bilgi işlem sistemleridir. Bu metin karakterleri alfabe harfleri, 0-9 gibi sayı noktalama işaretleri ve pound işareti, $, @ vb. özel karakterlerdir. İşletm daktilodaki klavyeye benzer bir klavye kullanarak metin karakterlerini girer. rakterler girildikçe bir video görüntüleme ekranında görüntülenirler. Metnin gi tamamlandıktan sonra işletmen, klavye aracılığıyla komutlar vererek metni çe biçimlerde işler. Fiziksel düzeni tatmin edici düzeye geldiğinde, klavyeden gi bir komut, metnin bir basılı kopya aygıtında basılmasına neden olur.

Şekil 5.2, bir kelime işlemciyle yapılabilecek metin işleme biçimlerine bir ö göstermektedir. Birinci paragrafta, işletmenin girdiği biçimiyle metnin bir böl

Bu, bir kelime işlemcinin, metin biçimindeki verinin işlenmesi için nasıl kullanılabileceğini gösteren bir örnektir. Birinci paragrafta, operatörün girdiği biçimiyle metnin bir bölümü görülmektedir. İkinci paragraf ise metni bir sağdan bloklama komutunun yürütümünden sonraki haliyle vermektedir. Kelime işlemcinin otomatik olarak paragraf yapısını koruduğuna dikkat ediniz.

Bu, bir kelime işlemcinin, metin biçimindeki verinin işlenmesi için nasıl kullanılabileceğini gösteren bir örnektir. Birinci paragrafta, işletmenin girdiği biçimiyle metnin bir bölümü görülmektedir. İkinci paragraf ise metni bir sağdan bloklama komutunun yürütümünden sonraki haliyle vermektedir. Kelime işlemcinin otomatik olarak paragraf yapısını koruduğuna dikkat ediniz.

Şekil 5.2

görülmektedir. İkinci paragraf ise aynı bölümü, bir İkinci paragraf ise metni bir "sağdan bloklama" komutunun yürütümünden sonraki haliyle vermektedir. Her bir satırda sözcük araları o şekilde ayarlanmıştır ki, satır sağ kenara kadar uzanmaktadır, böylece metnin tamamlanmış hali daha güzel bir görünüm kazanmaktadır.

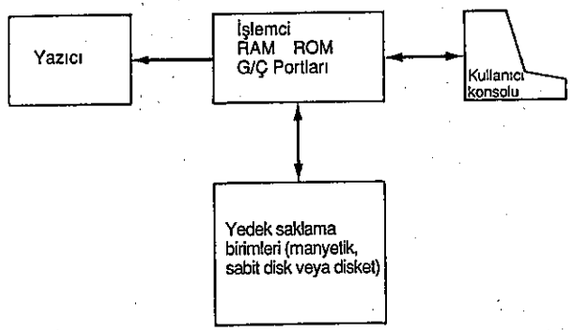
Operatör klavye aracılığıyla kodlanmış bir karakter girdiğinde, bu karakter, yaygın olarak metin arabelleği adı verilen bir bellek bölgesinde saklanır. Video görüntüleme birimi, o anda izlemekte olduğu tampon bölgesine saklanmış metni ekrana görüntüler. Dolayısıyla görüntülenen metnin işlenmesi, aslında, mikroişlemcinin, arabellek bölgesinde yer alan kodlanmış metin karakterlerini işlemesini kapsamaktadır. Yukarıda açıklanan "sağdan bloklama" komutu çalıştığında, metin karakterleri kodları, sözcükler arasına ek "boşluk" kodlarının yerleştirilmesiyle, metin arabelleğinde uygun şekilde taşınmaktadır.

İşletim özellikleri

Her ne kadar metin biçimindeki verileri işlemek gibi yalnızca bir işleve sahip olduğu söylenebilirse de kelime işlemci, Bölüm 5.1'de anlatılan çamaşır makinesi kadar özel bir sistem sayılmaz. Çünkü gerekli olan değişik işlemlerin sayısı, çamaşır makinesindeki işlem sayısından çok daha fazladır. Örneğin bir popüler kelime işlemci, işletmenin kullanabileceği 90'dan fazla sayıda komuta sahiptir. Bu durumda klavyeyi kullanan operatörün yaptığı denetim, çamaşır makinesi kullanıcısının basit anahtar işlemlerinden çok daha yüksek düzeydedir. Kelime işlemciyi, yarı-özel bilgi işlem sistemi şeklinde tanımlamak belki de en iyi tanımlama yoludur.

Bellek gereksinimleri

Şekil 5.3'de, bir kelime-işlem sisteminin ana bileşenlerini şematik olarak gö mektedir. Bu sistemi Bölüm 5.1'de incelenen denetimli çamaşır makin karşılaştırdığımızda, kelime işlemcideki RAM'ın çok daha büyük bir boyutta olduğu dikkatimizi çekebilir.



S5.7 *Şekil 5.3* Kelime işlemci

Harf, sayı, noktalama işareti vb. den oluşan her metin karakteri, 8-bitlik bi işgal eden bir kod ile temsil edilmekte ve normalde bir bellek konumunda sa maktadır. Bir tek sayfayı oluşturan metin karakterleri kolaylıkla 5.000 civa bellek konumunu gerektirebilir. Bu miktarda bir veriyi RAM'de saklamak ge olacaktır. Fakat yüzlerce sayfalık daha büyük dokümanlar için, kodlanmış m tamamını RAM'da tutmayı düşünmek gerçekçi olmaktan çıkar. Durum ne olsun, herhangi bir zamanda, metnin yalnızca sınırlı bir kısmı VDU'nun ekr görüntülenebilir. Kelime işlemci sistemlerinin çoğunluğunda, etkin bir yede lama ortamı, sıklıkla bir disket sistemi, gerekmektedir. Yedek disk saklama ğı, büyük miktarda verinin ana RAM bellekten ya da ana RAM belleğe hı biçimde aktarılmasına olanak tanır. Bu nedenle sınırlı miktarda, genellik sayfa, metin kodu ana RAM bellekte tutulur. Metnin başka bir sayfasının ing mesi gerektiğinde kod, yedek bellekten RAM'a aktarılır.

Ancak burada, yedek saklama ortamına aşırı düzeyde bağımlı olmanın k işlemcinin performansını olumsuz yönde etkileyebileceğine dikkat edilm Metnin, bir kısmı RAM'da, kalan kısmı ise bir diskette saklanan büyük bir p rafın baş tarafına bir eklemenin yapıldığı durumu ele alalım. Eklemenin yap yerden sonraki metnin tamamının yeniden düzenlenmesi gerekecektir. Bu yalnızca RAM'daki metinde gerçekleştirilebilen bir işlemi kapsar. Dolay

uygulama, RAM ile disket arasında çok sayıda metin bloğu aktarımını gerektire cektir, aktarım sayısı herhangi bir zamanda RAM'da tutulan metin bloğunun maksimum büyüklüğüne bağlı olacaktır. Eğer bu blok büyüklüğü çok küçükse, diskten veya diske metin aktarımı için harcanan zaman kullanıcı için kabul edilemez ölçüde fazla olabilir.

RAM'da ayrıca, klavye aracılığıyla verilen komutları yorumlayan ve metni istenen biçimde değiştiren program da saklanacaktır. Böyle bir programın kendisi, bellekte önemli ölçüde büyük bir alan kaplayacaktır. Yazılımın tamamlanması için bir disk işletim sistemi programı da gereklidir. Disk işletim programı ana ve yedek bellek arasında veri aktarımını düzenler. Normalde disk işletim programının en azından bir kısmı, sürekli olarak RAM'da bulunacaktır.

8080, Z80 veya 6800 gibi popüler 8-bitlik mikroişlemcilerden birisi kullanılarak tatmin edici, orta düzeyde yeterliliği olan bir kelime işlemci gerçekleştirilebilir. RAM büyüklüğü normalde 32-64K arasında olabilir. Bu 8-bitlik mikroişlemcilerde 16hatlık adres yolları bulunduğundan, doğrudan adreslenebilen bellek konumu sayısı en çok 64K olur.

Daha karmaşık kelime işlemciler için bu sınır uygun olmayacaktır, bu da bizi 8086, Z8000 veya 68000 ailesinden güçlü bir mikroişlemcinin kullanımına yöneltecektir, bu mikroişlemcilerin hepsi bellekte, 8-bitlik birimlerdekinden daha büyük alanları doğrudan adresleyebilmektedir. Bir bellek yönetimi destek birimi yardımıyla örneğin Z8001 bellekte 8 Mbayt'ı adresleyebilir, bu özellik hızlı cevaplama bakımından belirgin avantajlar sunabilir.

Çevresel birimler

Kelime işlemcinin gerçekleştirdiği işlemlerin açıklanmasından kolayca anlaşıldığı gibi G/Ç biriminin tipi önceden belirlenmiştir. İşlemciyle iletişim kurması gerekli olan işletmen bu amaçla normalde bir klavye giriş birimi kullanır. İşlemci ise kullanıcıyla, metin ve diğer mesajları, çıkış birimi olarak kullandığı bir ekranda görüntüleyerek iletişim kurar. Kelime işlemci son olarak genellikle bir basılı kopya çıkışı elde etmek için gerekir, bu durumda bir yazıcıya gerek vardır. Örneğin iş yazışmalarının gerektirdiği kalitede çıkış üreten yazıcılar pahalı araçlardır, bunların genel sistem maliyetine olan katkıları önemli ölçüde olabilir.

Geliştirme maliyetleri

Kelime işlemci üreten bir üretici, çok sayıda satış yapmayı arzu etse de üretim

hacminin, Bölüm 5.1'de anlatılan mikroişlemci-denetimli çamaşır ma hacmi ile aynı olması beklenmez. Bu nedenle geliştirme maliyetlerinin son satış fiyatına önemli düzeyde katkıda bulunması beklenebilir. Satış çamaşır makinesi satış fiyatın dan en azından birkaç kat daha yüksek muhtemeldir.

Basıldığı tarihte bir kelime işlemcinin maliyeti bir kaç bin ile birkaç onbin arasında değişmektedir. Bu satış fiyatının önemli bir oranı disk sürücüler vb. gibi elektronik olmayan parçalarla ilgilidir. Bir başka önemli bir oranı için programların geliştirilme maliyetiyle ilişkilidir. Yazılım geliştirme m rinin, toplam geliştirme maliyetinin % 50'den büyük bir kısmını oluşturma karşılaşılan bir durum değildir.

Bu tartışmadan, kelime işlemcinin son satış fiyatını etkileyen etkenlerin, m lemci-denetimli çamaşır makinesi ile ilgili etkenlerden farklı olduğu gö Her ne kadar iyi bir tasarım uygulamasında yonga sayısınının olabile durumda en aza indirilmesi gerekse de, kelime işlemcide bu konu, tasarım çok yüksek bir öncelikte değildir. Diğer taraftan kelimeişlemci yazılımı olmalı ve geliştirme maliyetleri bakımından olabilecek en düşük bir bulunmalıdır.

Yukarıdaki değerlendirme kitabın yazımı sırasında geçerli olan bir değerlen dir. Ancak, kelime işlemci gereçlerinin pazar potansiyellerinin çok büyük ol gösteren birçok belirti bulunmaktadır. Birçok insan klasik daktilonun kalkacağına ve yerini yazıcı eklenmiş güçlü, küçük kelime işlemcilerin al inanmaktadır. Bu gerçekleştiğinde kelime işlemci tasarımcılarının üzerind kılar, onları, çamaşır makinesi tasarımcısı ile aynı doğrultuda çaba gös yöneltecektir.

Yonga sayıları gözle görülür biçimde azalacak ve üretim maliyetlerinin üz geliştirme maliyetlerine göre çok daha fazla durulacaktır. Benzer baskılar kel -işlemci sisteminin mekanik ve elektromekanik parçalarının tasarımcıları geçerlidir. Bunun bir örneği olarak son birkaç yıl içinde satılan yazıcı maliyette belirgin düşüşle paralel olarak hızla artmıştır.

Bileşen sayısı

Tipik bir kelime-işlemci sistemindeki entegre devre sayısı, RAM büyüklü bağlı olarak 100 ile 300 arasında değişir. Bu düzeylerdeki ortalama yonga rakamları gelecekte azalma eğilimi gösterecektir. Bu durum, son birkaç yıl

geçerli olan, imalatçıların kelime işlemci gibi uygulamalar için özelleşmiş entegre devreler üretme yönündeki eğilimlerinin sonucudur. Sözü edilen özelleşmiş entegre devrelerden bir tanesi sıklıkla, daha genel tipteki çok sayıda devrenin yerini alabilir. Sürmekte olan bir başka eğilim ise, imalatçıların her bir entegre devre içine, giderek daha fazla sayıda bileşeni paketlemesi ve bu yolla belli bir işlevi gerçekleştirmek için gerekli olan entegre devre sayısını azaltmasıdır.

5.4 Mikroişlemci geliştirme sistemi

Mikroişlemci geliştirme sistemleri başlangıçta, mikroişlemci-tabanlı sistemler için donanım ve yazılım tasarımı yapan mühendislere yardımcı gereçler olarak oluşturulan birimlerdir. Örneğin mikroişlemci-denetimli yıkama projesi geliştiren bir ekip, haklı nedenlerle, kullandığı mikroişlemciyi destekleyen bir geliştirme sistemine erişebilmeyi isteyebilir. Sağlanan yazılım desteği, seçilen mikroişlemci için hızlı ve ekonomik program tasarımı ve hata ayıklama konularında yardımcı olmalıdır. Destek, daha sonraki konularda ayrıntılı olarak anlatılan "devreiçi emülasyon" biçiminde, donanım geliştirme için de sağlanabilir.

Bazı mikroişlemci geliştirme sistemleri "evrensel" olarak adlandırılmaktadır, pratikte bu, farklı birkaç mikroişlemciyi destekleyebildikleri anlamına gelmektedir. Genellikle mikroişlemcinin imalatçısı tarafından üretilen diğer bazı sistemler, yalnızca bir birimi destekler. Bu bölümde, evrensel tipleri anlatmayı daha sonraya bırakarak yalnızca bir mikroişlemciyi destekleyen mikroişlemci geliştirme sistemleri üzerinde yoğunlaşacağız.

Bellek gereksinimleri

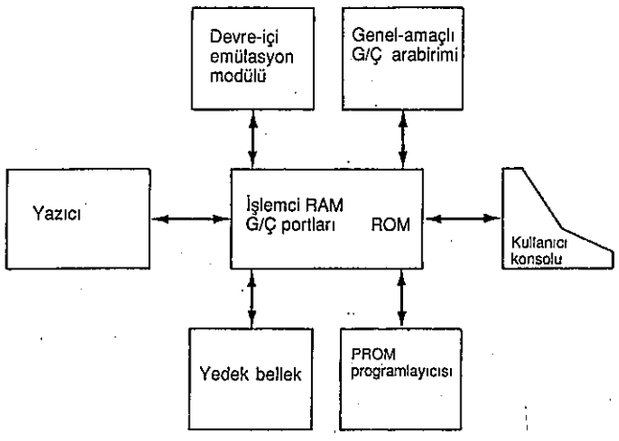
Yalnızca kendi metin-işleme ile ilişkili programı çalıştırması gereken kelime işlemcilerden farklı olarak, bir mikroişlemci geliştirme sisteminin, normalde, çok sayı ve çeşitte programı çalıştırabilmesi gerekir. Bu programlar, kullanıcı tarafından, geliştirilen sistem için oluşturulan programlara ek olarak yazılım destek paketlerini de içerecektir. Bu programların birçoğu çok büyüktür ve normalde RAM'a sığmazlar. Bu nedenle kapsamlı bir mikroişlemci geliştirme sisteminde, etkin ve hızlı-erişimli yedek bellek, bir gerekliliktir. Bölüm 5.3'de açıklanan kelime-işlem sisteminde olduğu gibi, yedek bellek, sistemlerin büyük bir çoğunluğunda bir manyetik disk saklama ortamıdır. Yedek saklama ortamlarının en popüler olanları 8 ya da 5 $^1/_4$ inç çapındaki disketlerdir. Böyle bir yedek belleğin bulunması, ana bellek ve disk arasında veri ve programların uygun şekilde aktarımını sağlayabilmek için bir disk işletim sistemini gerektirir. Disk işletim sistemi ayrıca, kulla-

nıcıya, programlarını ve veriyi adlandırılmış dosyalar şeklinde diskte saklayabilme rahatlığını verecektir.

Disk işletim sistemini ve mikroişlemci geliştirme sisteminin çeşitli diğer işlevleri desteklemek için büyük bir RAM gereklidir. Tipik geliştirme sistemleri günümüzde 32 ile 64K baytlık RAM ile çalışmaktadır. Mikroişlemci sistemleri için bu çok fazladır. Büyük bir anabilgisayar için bu büyüklükte bir RAM oldukça az sayılır.

Çevresel birimler

Şekil 5.4'de, tipik bir mikroişlemci geliştirme sisteminin bileşenlerini şematik olarak göstermektedir. Sistemde bir video görüntüleme terminali ya da konsol, kullanıcının merkezi işlemciyle iletişimine olanak veren bir klavye bulunmaktadır. Kullanıcı için, bir kelime işlemcinin sağladığı yazım kalitesi kadar yüksek olmasa bile bir kalite yeterli olsa bile, program listelemeleri vb. nin basılı kopyasını elde etmek için bir yazıcı gereklidir. Bu standart çevresel birimlera ek olarak, sistemin donanım ve yazılım geliştirme desteğindeki rolü daha özelleşmiş diğer çevresel birimler gerektirebilir. Bir PROM programcısı ve "devre-içi-emülatör" (ICE-In Circuit Emulation) yaygın olarak bir mikroişlemci geliştirme sisteminin parçasını oluştururlar. Bu alt-sistemlerin işlevi Bölüm 7'de anlatılmıştır.



Şekil 5.4 Mikroişlemci geliştirme sistemi

Eleman sayısı

Bir mikroişlemci geliştirilmesinde kullanılan entegre devrelerin toplam sayısı nispeten çok yüksek olacaktır, bu sayı belki de birkaç yüzü bulur. Buna çok fazla büyüklükteki RAM gereksiniminin katkısı vardır. Ek olarak, devre-içi emülasyon gibi çeşitli özel olanaklar da etkendir. PROM programlama ve genel-amaçlı seri/paralel arabirimler ek devre düzenlemesi gerektireceklerdir. Çok fazla entegre devre sayısı ve özel-amaçlı devre düzenlemesi eşliğinde sistemin nispeten yüksek geliştirme maliyeti sözkonusu olacaktır. Tüketim ürünleri ile ve orta düzeyde başarılı kelime işlemcilerle bile karşılaştırıldığında, mikroişlemci geliştirme sistemlerinin üretim miktarları nispeten daha düşüktür. Dolayısıyla bu tip sistemlerin yüksek geliştirme maliyetleri doğrudan doğruya satış fiyatına yansıtılmaktadır.

Destek yazılımı

Bir mikroişlemci geliştirme sisteminin yararlılığı, sistemle birlikte sağlanan destek yazılımın kalitesine yüksek oranda bağlıdır. Bu destek yazılım (Bölüm 8'de daha ayrıntılı olarak anlatılmıştır) en azından bir düzenleyici/çevirici/hata ayıklayıcı paketi içerecektir. Diğer olanaklar arasında, BASIC, PASCAL vb. gibi yüksek-düzey dillerden biri ya da daha fazlası için yorumlayıcılar veya derleyiciler bulunabilir. Devre-içi emülasyon veya PROM programlama gibi herhangi bir özel donanım niteliğini destekleyen programlar da sağlanacaktır.

Mikroişlemci geliştirme sistemi ile birlikte sağlanan yazılımın tamamı, ya da tamamı değilse bir kısmı, özel olarak sistem için yazıldığından, yazılım geliştirme maliyeti satış fiyatını belirgin oranda etkileyecektir. Bu da yine, nispeten küçük olan üretim sayısından kaynaklanır.

Sistem maliyetleri

Tipik bir mikroişlemci geliştirme sistemi, Zilog tarafından kendi mikroişlemcileri olan Z80'i desteklemek için üretilen sistemdir. Olanakları arasında 64 Kbayt RAM ve iki tane 8-inçlik disket sürücüsünden oluşan bir yedek saklama ortamı bulunmaktadır. Sistem, Z80 mikroişlemcinin devre-içi emülasyonuna olanak tanımakta ve ayrıca genel-amaçlı seri ve paralel giriş/çıkış arabirimleri bulundurmaktadır. Sistemle birlikte, çok çeşitli video görüntüleme terminalleri kullanılabilir.

Zilog sisteminin entegre devrelerini sayarsak merkezi işlemci, RAM ve yedek bellek denetleyicisinde bu sayının, yaklaşık 250 olduğu sonucuna varırız. Bu sayıya

devre-içi emülatör ve seri/paralel arabirimler de dahildir. Video görüntüle terminali ve yazıcıdaki entegre devreler bu sayı dışında tutulmuştur, bunl toplam sayısı, donanımın bu birimlerinin karmaşıklığına bağlı olarak 30 ve arasında değişebilir.

Kitabın yazımı sırasında bu tipte bir geliştirme sisteminin maliyeti yaklaşık ola 10.000 ingiliz sterlinidir. Genel olarak mikroişlemci geliştirme sistemleri, yalnı yazılım desteği ile birlikte olursa 5.000 sterlinden az bir maliyet ile, tam devre emülasyon yeteneği ve güçlü destek yazılımı bulunan ileri bir sistem olursa 30,0 sterlini aşan bir üst sınır arasında değişen maliyetler arasında değişebilmekted

Mikroişlemci seçimi

Burada anlatılan tipte bir mikroişlemci geliştirme sisteminin merkezi işleme normalde, sistemin desteklediği mikroişlemcidir. Dolayısıyla, örneğin Intel 80 mikroişlemcileri ile çalışan mühendisleri desteklemek için tasarlanan bir mikro lemci geliştirme sistemi, normalde bir 8080 işlemci içerecektir. Bir 8080 mikro lemcisi için, farklı tipte bilgisayar kullanarak, çapraz-çeviriciler çapraz-derleyiciler olarak adlandırılan programlar yardımıyla makine kodu ür mek mümkün olduğundan, bu çok kesin bir kural değildir. Hatta, 8080 için yazılı programları, farklı bir bilgisayarda, 8080 işlevlerinin aynen gerçekleştirilmesi sağlayabilen özel bir program kullanarak çalıştırmak ve hata ayıklaması yapı
S5.2 mümkündür. Ancak sistem, yalnızca tek bir mikroişlemciyi desteklediğinde, pro
5.4 ramların, hangi mikroişlemci için yazıldılarsa o mikroişlemciyi kullanarak çalış
5.8 rılması ve hata ayıklama işleminin yapılması, genellikle daha doğrudur.

Sorular

5.1 Aşağıdaki sistemlerin özel, yarı-özel ve genel-amaçlı gruplarından hangi ne girdiğini söyleyiniz.
- *(a)* Büyük bir firmada ücretlerin hesaplanmasında ve stok denetimin kullanılan bir bilgisayar.
- *(b)* Bir doktor tarafından kullanılan bilgisayar-yardımlı tanı sistemi.
- *(c)* Bir bilgisayar-denetimli satış noktası kasası
- *(d)* Eğlence yerlerindeki video oyunları

5.2 Aşağıdaki sistemler için, geliştirme ve üretim maliyetlerinden hangisini daha önemli olacağını, nedenleriyle birlikte söyleyiniz.
- *(a)* Yalnızca üçü tane imal edilecek büyük bir tezgah sistemini denetley özel bir mikroişlemci sistemi.

- *(b)* Basit BASIC programlarını çalıştırabilen bir kişisel el mikrobilgisayarı
- *(c)* Bir jet uçağı motoruna giden yakıt sistemini denetleyen ve 20 yıllık bir dönem içinde 5.000 adet satılması beklenen mikroişlemci.

5.3 Programların saklanma biçimi açısından bir genel-amaçlı sistem ile özel-amaçlı sistem arasındaki farklar nelerdir?

5.4 Tipik bir özel bilgi işlem sistemindeki entegre devre sayısı ile tipik bir genel-amaçlı sistemdeki entegre devre sayısını karşılaştırınız.

5.5 Tek-yongalı mikrobilgisayarlarla ilişkili olarak "maskeleme" maliyeti terimi ile ne anlatılmak istendiğini açıklayınız.

5.6 Verimli biçimde çalışması gerekiyorsa bir kelime işlemci için neden büyük bir RAM gerektiğini anlatınız.

5.7 Tipik bir kelime işlemcinin bileşen birimlerini gösteren bir blok diyagram çiziniz.

5.8 Geliştirme maliyetlerinin bir mikroişlemci sisteminin satış fiyatını nasıl etkilediğini açıklayınız:
- *(a)* Yalnızca birkaç sistem üretileceği zaman.
- *(b)* Çok sayıda üretileceği zaman.

Bölüm 6 Mikroişlemci sistem planlaması

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde;*

1 *Bir akış diyagramı yardımıyla bir sistem tasarım/geliştirme programının tipik aşamalarını gösterebilmeli,*
2 *Başarılı mikroişlemci sistem tasarımı için tam sistem özelliklerinin önemli bir ön koşul olduğunu kavramış olmalı,*
3 *Tipik bir durumda son-kullanıcı gereksinimlerinden yola çıkarak bir sistem özelliklerini oluşturabilmeli,*
4 *Tipik bir tasarım probleminin çözümü için iki ya da daha fazla başka seçenek olduğunda, aralarından en yüksek verimi, en düşük maliyetle sağlayan yöntemi seçebilmeli,*
5. *Sistem görevlerinin donanım ve yazılım arasında bölüştürülmesi işlemini analiz edebilmelisiniz.*

6.1 Tasarım ve geliştirme

Bu bölüm, bir ya da daha çok mikroişlemci bulunduran sistemler için programlar tasarlama ve geliştirme konularıyla ilgilidir. Dikkatli planlamanın, mühendislik tasarım ve geliştirme programlarının tümünde kesinlikle bulunması gerekir. Ancak bir sisteme herhangi bir türde bilgisayar katılması, diğer elektronik sistemlerde beklenenlerden daha fazla ve daha üst düzeyde özel sorunlarını birlikte getirecektir.

Mikroişlemci içermeyen elektronik bir sistemde performansı etkileyen, iki ana özellik vardır. Bunlardan birincisi elemanların -dirençler, kondansatörler, entegre devreler vb.seçimidir. İkinci özellik ise bu elamanların birbirine nasıl bağlanacağını belirleyen devredir. Bu özellikler, sistemin bir mikroişlemci içermesi durumunda da önemlerini aynı ölçüde korurlar. Ancak bu durumda sistemin çalışma biçimini etkileyen ek bir etken ortaya çıkar. Bu etken mikroişlemci tarafından çalıştırılan sistemde saklı programdır. Özel bir mikroişlemci sisteminde program sadece-okunur bellekte (ROM) saklanır, ve "firmware" olarak anılır. Daha çok yönlü, genel-amaçlı sistemlerde program, yürütülmesi gerektiğinde rastgele-erişimli belleğe (RAM) aktarılır. Bu noktadan sonra artık yazılım olarak adlandırılır. Bu bölümde yazılım terimini hem yazılım hem de firmware anlamında kullanacağız.

Alışılagelmiş elektronik sistemlerde devre düzenleme ve elemanların seçimi birbirinden bağımsızdır. Devre düzenlemesinin değiştirilmesi kolaylıkla, farklı bir eleman seçimini gerektirebilir. Bunun tersi de doğrudur, farklı bir eleman seçimi ise, devre düzenlemesinin değiştirilmesini gerektirebilir. Bir mikroişlemci-tabanlı sistem geliştirirken birbirine bağımlı üç etken göz önünde bulundurulmalıdır. Bunlar aşağıdadır:

1 Bileşen seçimi
2 Devre ve sistem düzenlemesi
3 Yazılım

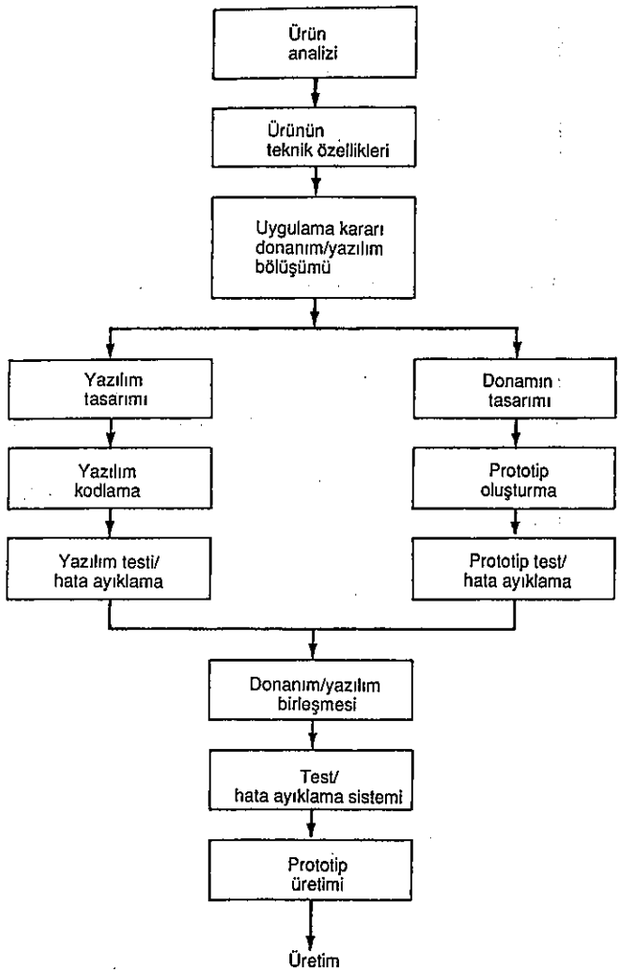
Bunlardan herhangi birinde yapılan bir değişikliğin diğer ikisinde de değişiklikler gerektirme olasılığı vardır. Mikroişlemci-tabanlı bir sistemde ek olarak ortaya çıkan yazılım sorunu, tasarım/geliştirme programının önemini özel bir konuma getirmektedir. Planlama aşamasında doğru kararların verilmesi, geliştirme ve üretim maliyetlerini çok yüksek düzeyde etkileyebilir. Sistemin tamamlanmış halinin gerekli özellikleri ne ölçüde karşılayacağı da planlama kararlarına bağlı
S6.1 olacaktır.

6.2 Geliştirme aşamaları

Bir mikroişlemci-tabanlı ürünün geliştirilmesinde, başlangıç tanımlamasından imalata kadar olan çeşitli aşamalar, Şekil 6.1'de gösterilmiştir. Şekilden, bazı donanım ve yazılım geliştirme çalışmalarının paralel şekilde yürüyebileceği de anlaşılmaktadır. Ancak son listeleme ve hata ayıklama işlemleri için donanım ve yazılımın bir araya getirilmesi gerekir. Okuyucu Şekil 6.1'de gösterilen akış diyagramında, geliştirme programlarının çoğunluğu için biraz idealize edilmiş bir yakıştırma yapıldığını değerlendirmelidir. Bir gerçek uygulama programında çeşitli aşamaların tekrarlanması genelde gerekli olur. Örneğin, donanım ve yazılım bir araya getirildiğinde, bunlardan herhangi birinde, daha önce karşılaşılmamış olan tasarım hataları ortaya çıkabilir. Böyle bir durum olsa olsa, donanımda, yazılımda ya da her ikisinde birden, temel özelliklerine yol açabilir. Geliştirme progamının planlanmasında gösterilecek yeterli dikkat, bu gibi durumların önlenmesi için
S6.3 oldukça yararlı olacaktır.

6.3 Sistem özellikleri

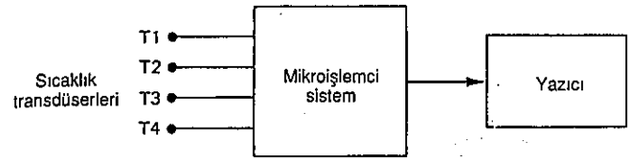
Bir geliştirme programını planlamak için, geliştirilen ürün ya da sistemin kesin, ve farklı biçimlerde yorumlanması mümkün olmayan bir özelliklerinin bulunması önemlidir. Bu özellik ilk olarak, ürünün dış özelliklerinin tanımından oluşmalıdır.



S6.2 *Şekil 6.1* Mikroişlemci sistem geliştirme programı aşamaları

Dış özellik, olası bütün girişler için sistem çıkışlarını ve sistemin cevaplama biçimini tanımlar. Adından da anlaşılacağı gibi, sistemin dıştan bakılarak yapılan tanımlamasıdır. Bu özelliklerinde, ayrıca, girişlerin normal sınırları aşması durumunda beklenen sistem cevabı da belirtilmelidir.

Denetleyici birimler olarak çalışan mikroişlemciler, genelde ardışık işlemler uygularlar. Bölüm 4 te anlatılan çamaşır makinesi programlarının mikroişlemci-tabanlı denetimi bunun iyi bir örneğidir. Sistemin özelliklerinde ayrıca ardışık çalışma



Şekil 6.2 Sıcaklık kayıt sistemi şematik diyagramı

biçimi de tanımlanmalıdır. Akış diyagramları, genelde en etkili tanımlama biçimlerini sağlar.

Örnek olarak, küçük bir veri kayıt ve saklama sistemini ele alalım. Veri kayıt sisteminin, dört ölçme noktasındaki sıcaklığı her saatte bir örneklemesi ve ölçüm sonuçlarını yazması gerekmektedir. Şekil 6.2'de, sistemin, mikroişlemci sisteminin kendisinin iç ayrıntıları olmaksızın, dış görüntüsü verilmektedir. Bu sistem için özellik aşağıda gösterilmektedir.

Veri Kayıt Sistemi Özellikleri

Ölçüm noktaları sayısı	4
Ölçülen değişken	Sıcaklık
Normal sınırlar:	
T1	85-110°C
T2	0-220°C
T3	50-150°C
T4	50-150°C
Mutlak maksimum hata	± 2°C
En büyük tam-ölçek artım hatası	± 1°C
Çözünürlük	1°C
Bir kanal için örnekleme süresi	1 örnek/saat
Maksimum kısa-dönemli örnekleme süresi hatası	0.1 sn/saat
Maksimum uzun-dönemli örnekleme süresi hatası	1 sn/gün
Güç kaynağı (şebeke) frekansı	50 Hz

Özelliklerinden anlaşılacağı gibi, sıcaklıklar, değer normal sınırların dışına çıkarsa, bir hata mesajı ile birlikte yazılacaktır. Şekil 6.3'de, basılı çıktı formatı gösterilmiştir.

Yukarıda gösterilen dış özelliklerinden, kayıt sisteminin bazı iç özellikleri hemen tesbit edilebilir. Örneğin, sıcaklık transdüserlerinden gelen analog çıkışı sayısal

```
DAY 1                    09.00 HR

TEMPERATURES
T1 =  105 DEGREES
T2 =  16 DEGREES
T3 =  102 DEGREES
T4 =  56 DEGREES

DAY 1                    10.00 HR

TEMPERATURES
T1 =  111 DEGREES
T2 =  18 DEGREES
T3 =  155 DEGREES
T4 =  60 DEGREES

! WARNING! T1 OUTSIDE LIMITS
! WARNING! T3 OUTSIDE LIMITS
```

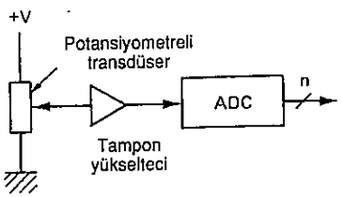
Şekil 6.3 Sıcaklık kayıt sistemi basılı çıktı formu

biçime değiştirmek için, bir ya da daha çok sayıda analogdan sayısala dönüştürücü (ADC) gerekecektir. Sistem özellikleri ayrıca, kullanılacak ADC'lere ilişkin belli ölçütleri belirlememize de olanak tanımaktadır. İlk dikkatimizi çeken şey 1 °C 'lik bir sıcaklık çözünürlüğü gerektiğidir. Bu, ADC çıkışının aşağıda gösterildiği gibi en az 8 bit içermesinin gerekli olduğu anlamına geldiğinden, önemlidir.

Bir ölçme sisteminin çözünürlüğü, çıkışta algılanabilen en küçük değişikliktir. Veri kayıt sistemi için bu, basılı çıktının 1 °C kadar küçük değişiklikleri içermesinin gerektiğini gösterir. Herhangi bir ölçüm noktasında beklenen en yüksek sıcaklık değişimi 220 °C olduğundan, sistem, 220'de 1 çözünürlük olmalıdır. 8-bitlik bir ikili sözcük 0-225 arasındaki sayıları temsil edebilir, dolayısıyla bu bit sayısı yeterli çözünürlüğü sağlar.

Çözümlü örnek

Bir mikroişlemci sisteminde milin açısal konumunu ölçmek için bir hassas potansiyometrik bir transdüser kullanılmıştır. Şekil 6.4'te gösterildiği gibi, transdüser çıkışı bir ADC'ye giriş olarak uygulanmaktadır. Milin maksimum toplam açısal dönüşü 200 derecedir ve transdüserin kullanıldığı sistem 0.5 derecelik bir çözünürlük gerektirir. ADC çıkışının ikili biçimde olduğunu varsayarsak gerekli olan en küçük bit sayısını tahmin ediniz.



Şekil 6.4 Bir milin açısal konumunun sayısal ölçümü

Çözüm Çözünürlük, toplam dönüş olan 200 derecede 0.5 derecedir, yani 400'de 1'dir. Dolayısıyla ADC çıkışı, en büyük işaretsiz 8bitlik sayı, yeterli çözünürlüğü sağlamayan 255 olduğundan, en az 9 bit içermelidir. En büyük işaretsiz 9-bitlik sayı 511 dir, bu ise amaç için uygundur.

Sıcaklık transdüseri/ADC kombinasyonunun hasssalığı gibi diğer ölçütler de bu noktada belirtilebilir. Ancak genelde sistem elemanlarının çoğunluğunun belirtilmesinden önce sistem düzenlemesiyle ilgili başlangıç kararları alınmalıdır.

S6.4
6.5

6.4 Maliyet-verimlilik ölçümleri

Bir tasarım/geliştirme programının nasıl gerçekleştirildiği konusu ele alınırken, tasarımcının, bir uygulama problemine yalnızca bir çözüm bulabildiği durumların çok ender olduğunu kabul etmek gerekir. Tasarımcı çoğu kez, uygulamaya konulduklarında hepsi de sistem özelliklerini karşılayan birkaç strateji arasından bir tanesini seçmek durumundadır. Böyle bir durumla karşı karşıya kalan tasarımcı, çeşitli seçeneklerin hepsinin de gereken performans ölçütlerini karşıladığı varsayıldığında, yapacağı seçimi, gerçekçi şekilde maliyet sorusuna dayandırmalıdır.

Belirli bir tasarım stratejisinin uygulama maliyetini tahmin ederken, mamul ürün veya sistemin satış fiyatını etkileyen etkenlerin hepsinin hesaba katılması gerekir. Tasarım, geliştirme, üretim ve eleman maliyetleri bu etkenler arasındadır. Son ürün satış fiyatına tasarım ve geliştirmenin yapacağı katkı tabii ki toplam üretim sayısına bağlıdır. Kazancı hariç tutarsak birim maliyet aşağıdaki gibi hesaplayabiliriz:

$$\text{Maliyet} = C + P + (D/N)$$

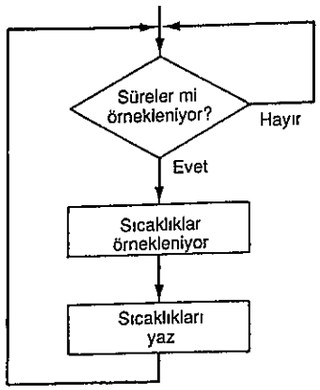
Burada C, her birime düşen toplam eleman maliyetidir. P, bir birim için üretim maliyetini, D, tasarım/geliştirme programı toplam maliyetini ve N, üretilen birim sayısını göstermektedir.

Geliştirme yazılımı, tasarım/geliştirme programının toplam maliyetine dahil edilmelidir. Yazılım geliştirme maliyetlerinin toplam tasarım/geliştirme bütçesinin içinde % 50 oranını aşması seyrek karşılaşılan bir durum değildir, % 80 - 90 gibi oranlara ise hiç bir şekilde ulaşmaz.

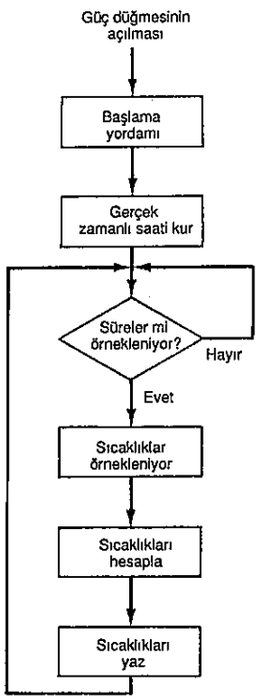
İki tasarım seçeneği için maliyet tahminlerinin nasıl yapıldığına ilişkin bir örnek, Kısım 6.8'da verilmiştir. Maliyet tahminlerinin kesinliği, proje planlayıcılarının, donanım ve yazılım geliştirme için gerekli olan zaman ve olanakların değerlendirilmesindeki beceri ve deneyim düzeyine bağlıdır. Deneyim eksikliği hemen her zaman, donanım ve yazılım geliştirme maliyetlerinin genelde, büyük ölçüde daha düşük olarak tahmin edilmesiyle sonuçlanır.

6.5 Sistem akış diyagramı

Kısım 6.3'te akış diyagramının, birçok mikroişlemci-denetimli sistemde ortak özellik olan ardışık tipteki işlemleri tanımlamanın en uygun yöntemi olduğunu söylemiştik. Aslında geliştirme programının planlanması sırasında birkaç akış



Şekil 6.5 Sıcaklık kayıt sisteminin başlangıç akış diyagramı



Şekil 6.6 Sıcaklık kayıt sisteminin genişletilmiş akış diyagramı

diyagramı hazırlanır. Bu diyagramlardaki ayrıntıya gitme-zeyi öngörülen sistem tasarımı ortaya çıktıkça artar.

Kısım 6.3'te tanıtılan veri kayıt sistemine dıştan bakıldığı gözlenen çalışma biçimini anlatan bir akış diyagramı Şe 6.5'de görülmektedir. Bu diyagram Şekil 6.6'da, daha fa ayrıntının gösterilebilmesi için genişletilmiştir. Şekil 6.6'd sistemin daha önce açıklanmayan belli bazı özellikleri tanı lanmaktadır. Örneğin, ADC'den alınan sayısal sıcaklık ölçü lerinin, gerekli veri mesajını oluşturmak üzere yazıcı iletilmek için uygun duruma gelmeden önce işlenmesi gere tiği, dikkatimizi çekmektedir. Verinin işlenmesinin nasıl ve kapsamda gerçekleşeceği kullanılan mikroişlemcinin tipi gib donanım elemanlarının seçiminde kolaylıkla etken olabilir.

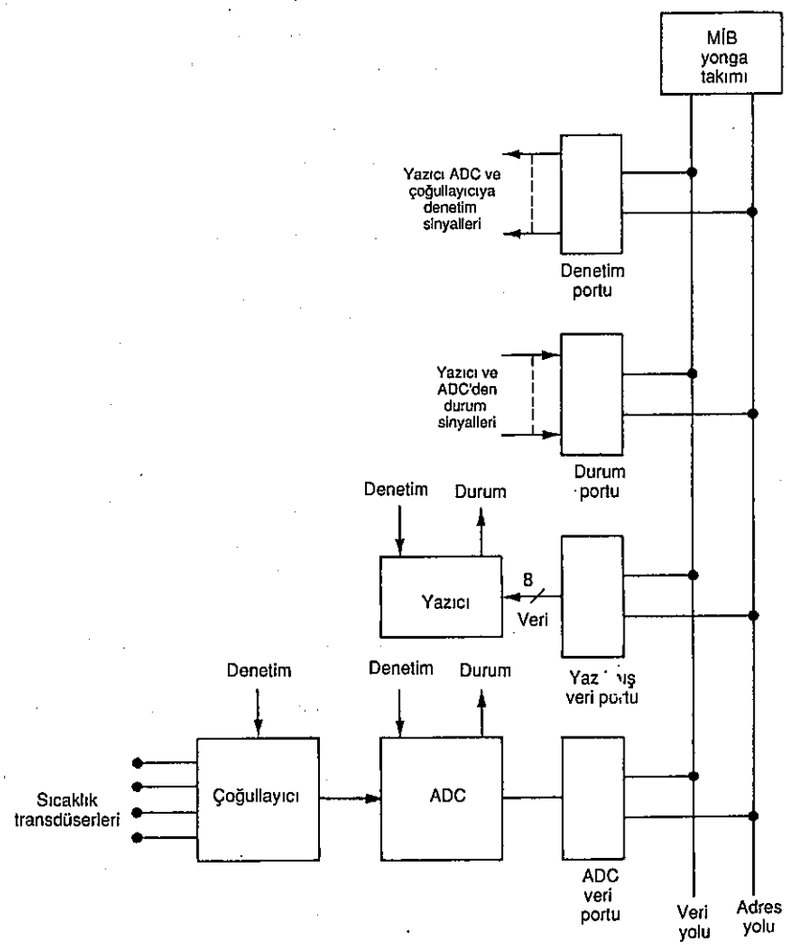
Şekil 6.5'te gösterilmeyip 6.6'da gösterilen bir başka özelli başlama işlemleriyle ilişkilidir. Şekil 6.6'da herhangi bir aşa mada enerji anahtarı açıldığı ve sistemin istenen biçimde çalışmaya başladığı gösterilmektedir. Veri kayıt sistemi "gerçek-zamanlı" bir sistem olarak tanımlanabilir, çünkü sı caklıkların örneklendiği zamanlar kaydedilmektedir. Mikroişlemcinin iç zamanlama sistemini, "gerçek-zaman" ile eşzamanlı duruma getirmek için, bu iç saatin bir şekilde ayarlanması gerekmektedir. Bu gereklilik, Şekil 6.6'da dikkate alınmıştır. Başlatma ya da kullanıma hazırlama işlemleri bu noktada ayrıntılı biçimde anlatılmamış, ancak kitabın ileriki bölümlerinde bu amaçla kullanılan yöntemler daha ayrıntılı olarak ve geniş şekilde ele alınmıştır.

6.6 Başlangıç aşamasındaki donanım kararları

Geliştirme programının bu aşamasında, sistemde kullanılacak olan elektronik birimlerla ilgili ilk kararlar alınır. Mikroişlemcinin kendisi, çevresel birimler ve arabirim elemanlarının tamamı, ilk kez bu aşamada belirlenirler. Kısım 6.5'de anlatılan sistem akış diyagramıyla uyumlu biçimde, bu belirleme işlemi kullanılması düşünülen ve daha sonra kullanılmasından vazgeçilebilecek olan elemanların belirlendiği bir geliştirme işlemidir. Seçilen sistem elemanlarının çalışma biçimi açıklığa

kavuştukça sistem akış diyagramı da, daha fazla ayrıntıyı içermek için genişleyecektir. Geliştirme programının bu aşaması, doğal olarak, bir sonraki bölümde tartışılan donanım/yazılım bölüşümü konusu ile paralel bir seyir izler. Kısım 6.1'de söylediğimiz gibi donanım seçimleri yazılımı, yazılım da donanım seçimlerini etkiler.

Şekil 6.7'de, bu aşamada sistemin iç düzenlemenin ilk belirlemesinin nasıl yapılabileceğini gösterilmektedir. Şekil 6.7'de yalnızca G/Ç işlemleriyle ilişkili elemanlar gösterilmiştir. Gösterilen elemanlara ek olarak RAM ve ROM biçiminde bellek birimlerine olacağı açıktır. Şekil 6.7'de gösterilen sistem, dört sıcaklık transdüse-



Şekil 6.7 Sıcaklık kayıt sistemi için başlangıç donanım taslağı

rinin sırayla örneklenmesi için çoğullayıcıyla birlikte bir ADC kullanmakta şemanın bir başka seçeneği, her bir sıcaklık transdüseri için bir ADC kullanm Şekil 6.7'deki gibi bir sistem diyagramı oluşturmak için tasarımcı, sistem ele larına ilişkin belli bazı kararlar almıştır. Dört ADC yerine çoğullayıcı ile bi tek bir ADC kullanmak konusunda bir karara varırken, deneyimli bir tasarımcı iki seçenekteki maliyet yapısını inceleyecektir. Bu gibi durumlarda kararlara varıldığına ilişkin bir örnek Kısım 6.7'de verilmektedir.

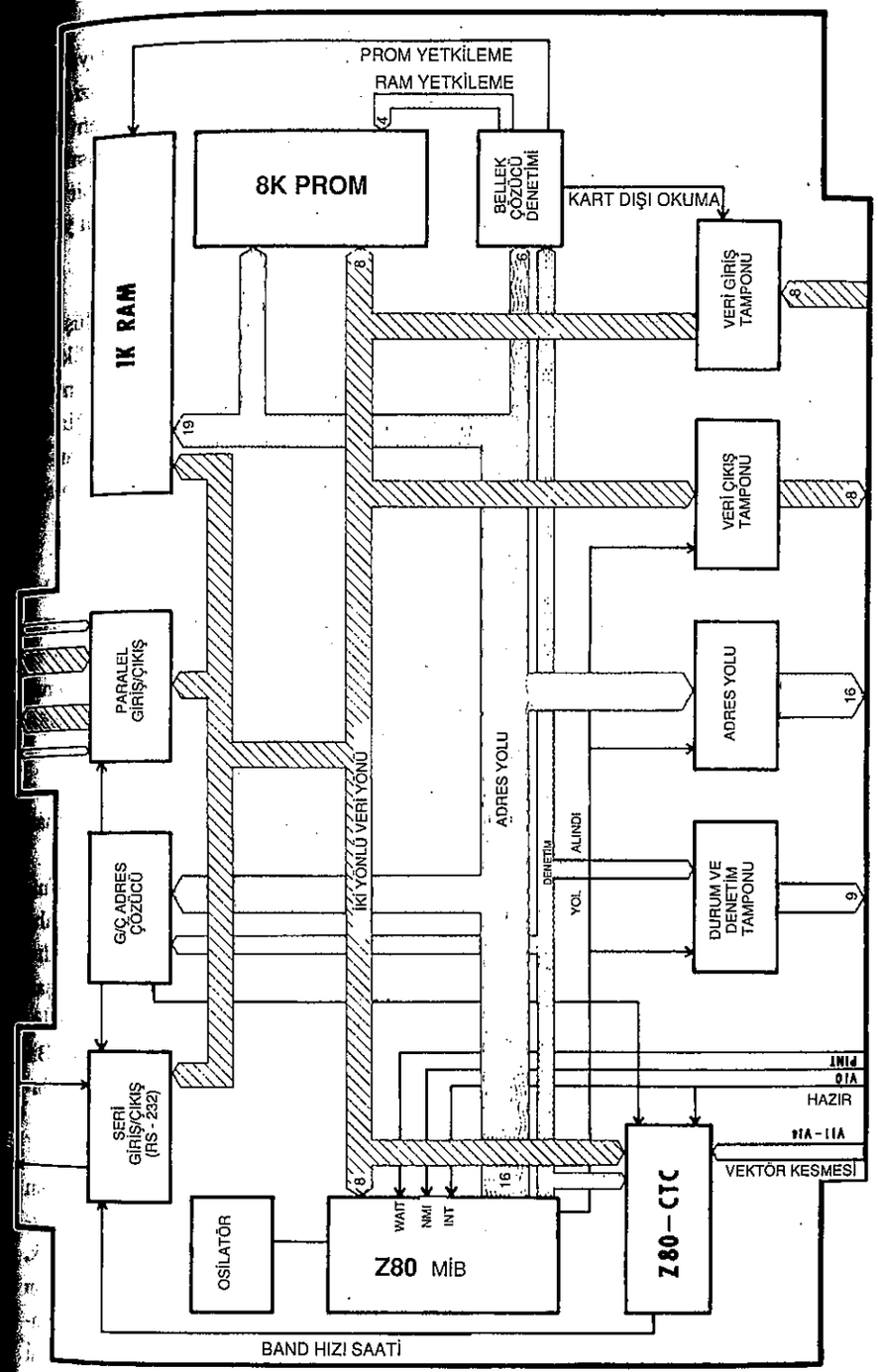
Hem donanım hem de yazılımı içine alan önemli bir karar ise mikroişlem seçimidir. Bu seçimde birçok etkenin etkisi vardır, kaydedilen veriye uygular gereken işlemlerin miktarı bu etkenler arasındadır. Bölüm 2 ve 3'de bu ayrıntılı biçimde tartışılmıştır. Mikroişlemci seçiminden belki de daha önemli karar ise MİB ile ilişkili devre düzenlemesinin tasarımında rol alıp almama kar dır. Bu donanım tasarımı için başka bir seçenek, markasız bir tek-sistemk bilgisayar kullanmaktır. Çeşitli düzenlemelerde mevcut olan bu kartlar, no olarak, bir baskılı devre kartı üzerinde, mikroişlemci, ROM veya EPROM, R ve bunlarla birlikte G/Ç portları içerirler.

Bileşen maliyeti açısından tek-sistemkartlı bilgisayar kendi elemanlarından d pahalıdır. Fakat donanım geliştirme ve üretim maliyetlerinde sağlanan tasa oldukça iyi bir düzeydedir. Özellikle üretim sayıları çok yüksek değilse bir tek temkartlı bilgisayar, özel-amaçlı bir MİB devresinin tasarımı ve geliştirilmesi ekonomik ve zaman kazandırıcı bir seçenek sağlayabilir. Şekil 6.8'de, tipik tek-sistemkartlı bilgisayarın düzenlemesini blok diyagram biçiminde gösterilm tedir. Bu bilgisayar Z80-tabanlı SBC100'dür.

Tek-sistemkartlı bilgisayar MİB'den ve MİB'e sayısal verinin aktarılması için G portları bulundurur. Sistem tasarımının yaz-boz kaydedicisinden yapıldığı duru larda G/Ç portları tasarımcı tarafından seçilmelidir. Basit 8-bitlik sistemle paralel bir sayısal çıkış arabirimi, yalnızca, 74LS373 gibi 8bitlik bir mandal ve adres kod çözücüsünden biraz daha fazlasını içerebilir. Giriş portları benzer şekil bir adres kod çözücüsü ve bir üç-durumlu ya da açık-kollektörlü tampon takımı uygulanabilir. Daha kapsamlı G/Ç olanakları gereken durumlarda özel-ama programlanır bir arabirim kullanımı daha ekonomik olabilir.

6.7 Donanım ve yazılım bölüşümü

Sistem akış diyagramı, belirli bir sıralama içerisinde yer almaları gereken çeşi işlemleri tanımlamaktadır. Birçok durumda tasarımcı, belli bir işlemin, özel olar tasarlanmış donanım tarafından mı, yoksa, mikroişlemci tarafından programın



Şekil 6.8 SBC 100 tek-sistem kartlı bilgisayarın genel düzenlemesi

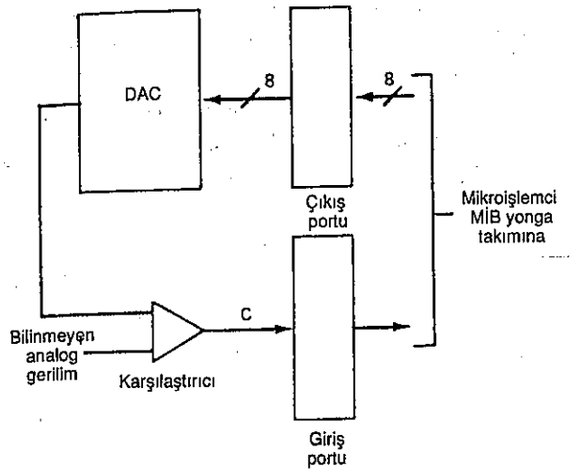
bölümünün yürütülmesi yoluyla mı uygulanacağını kendi isteğiyle saptayı
Bazı işlemlerde seçme olanağı yoktur. Örneğin sonuçların yazılmasının yazıc
oluşan donanım tarafından gerçekleştirildiği açıktır. Başlangıçtaki sistem
diyagramları yalnızca gerçekleştirilecek işlemleri tanımlamakta bunların nasıl
çekleştirileceği ile ilgili herhangi bir işaret taşımamaktadır. Dolayısıyla bir
diyagramı, bir ardışık donanım işlemleri dizisini, bir bilgisayar programını y
bunların her ikisinin bir kombinasyonunu anlatır.

Diğer tasarım/geliştirme kararlarında olduğu gibi, donanım ve yazılım bölüş
seçiminde yer alan iki ana etken performans ve maliyettir. Performans dik
alındığı zaman gerekli görülen işlem hızı önemlidir. Belli bir işlem, özel-am
donanım ile, mikroişlemci tarafından çalıştırılan bir programla olabileceğin
hemen her zaman daha hızlı gerçekleştirilebilir. Bu nedenle tasarımcılar zaman
nırlılığı sözkonusu olan tasarım sorunlarında donanım çözümlerini uygulam
koyma zorunluluğu ile karşı karşıya kalabilirler.

Bir donanım/yazılım seçim tipi örneği olarak, uygulamada karşılaşılabilen
durumu, Bölüm 6.3'de tanıtılan sistemdeki veri kaydetme işleminin zamanlama
ele alalım. Anımsanacağı gibi sıcaklıklar bir saatlik aralarla örneklenmekte
yazılmaktadır. Bu nedenle, bu kayıt işleminin gerçekleşeceği zamanı belirleyen
sabitleyecek bir yolun belirlenmesi gerekmektedir. İzlenebilecek üç yöntem aşa
dadır:

1 Kaydetme işlemleri arasındaki zaman aralığında yürütülen ve gerekli zam
gecikmesini tam olarak üreten bir program yazmak,
2 Yüksek düzeyde kararlı olan, kristal-denetimli, çıkışı bir mikroişlemci tarafı
dan örneklenen, ya da kesme üreten bir osilatör kullanmak. Döngüler, dah
sonra gerekli zaman gecikmesi oluşuncaya kadar bir program tarafından sayıl
3 Yüksek düzeyde kararlı olan, her saatte bir çıkış darbesi verecek şekilde çalış
bir frekans bölücüsü kullanmak. Her kaydetme işleminden sonra frekans bölü
cüsünün çıkışı, bir sonraki zamanlama darbesi algılanıncaya kadar taranır. B
başka seçenek ise frekans bölücüsünün çıkışının bir kesme üretmek için kulla
nılmasıdır.

Yaklaşımlar bütünüyle yazılım çözümü (1) ile donanım tekniği temelinde çözüm
(3) arasında değişmektedir. (2) ve (3) nolu yaklaşımlarda kullanılan yüksek düzey
de kararlı osilatör, "gerçek zaman saati" olarak anılmaktadır. Dördüncü bir seçenek
kristal denetimli bir osilatör çıkışı yerine şebeke frekansı ile çalışan gerçek zaman
saati tekniğini kullanmaktır. Kısa-dönemli zamanlama duyarlılığı, kristal-denetim
li bir osilatör yöntemine göre azalmıştır fakat hala başta belirlenen özelliklerini

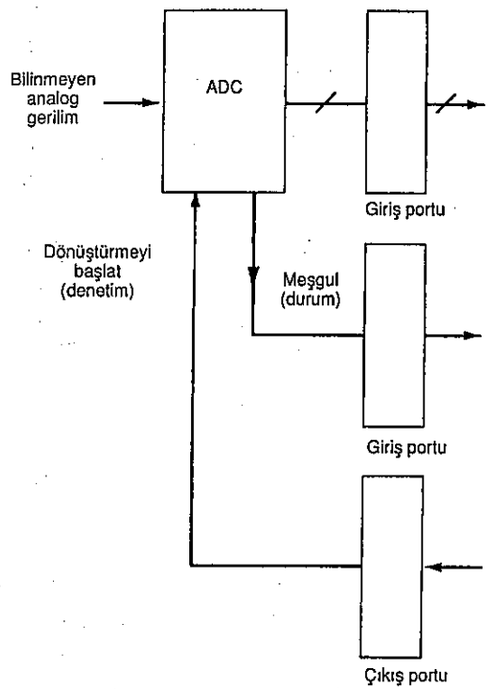


Şekil 6.9 DAC kullanarak Analogdan-sayısala dönüştürme

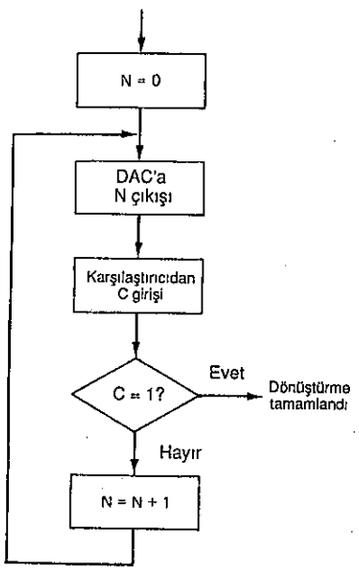
karşılayacak düzeydedir. Şebeke frekansının uzun-dönemli duyarlılığı, çok iyidir ve bu amaç için uygun olan duyarlık değerinden daha yüksektir. Önerilen çözümlerden herhangi birisi zamanlama kesinliği gereksinimini karşılamaktadır. Bu nedenle, tasarımcının görevi üretilen her sistemin maliyetine en az katkıyı yapan yöntemi seçmektir.

Maliyet açısından, kaydetme işlemleri arasındaki zaman aralıklarının donanım yardımıyla mı yoksa yazılımla mı belirleneceği kararı belki de çok önemli değildir. Çünkü gereken işlem, nisbeten basit bir işlemdir ve donanım ya da yazılım tarafından uygulanması çok düşük maliyetli bir uygulamadır. Ancak bu örnek, donanım/yazılım değerlendirmesi ile ilgili etkenlerin gösterildiği yararlı bir örnektir.

Gerekli donanım/yazılım kararlarına bir başka örnek olarak da AD dönüşümünü ele alacağız. Veri kaydetme sisteminde her saatte yalnızca dört AD dönüştürmesi gerekmektedir. Dönüştürme hızı çok önemli değildir. Dolayısıyla tasarımcı, markasız bir ADC modülü kullanmak ve bir eleman grubunu birbirine bağlamak arasında seçim yapabilir. Bileşen parçalar takımının seçilmesi durumunda mikroişlemci, AD dönüştürme işleminin bazı kısımlarını gerçekleştirmek için programlanabilir. Şekil 6.9'da bu tipte bir donanım gösterilmektedir. Şekil 6.9'da gösterilen şema, uzun AD dönüştürme zamanları ile sonuçlanır bu nedenle yüksek-hızlı işlem gerektiği zaman uygun değildir. Böyle sınırlamalar bulunmaz. Şekil 6.10'da bir markasız ADC modülünün kullanımı gösterilmektedir. Bazı durumlarda Şekil



Şekil 6.10 ADC bağlantıları



Şekil 6.11 Analog-sayısal dönüştürmeye ilişkin basit bir sayım algoritması

6.9'daki sistemin Şekil 6.10'dakine yeğlenmesinin neden Şekil 6.9'daki sistemin Şekil 6.10'daki sistemden kolaylıkla daha düşük eleman maliyetine sahip olabilmesidir.

Şekil 6.9'da gösterilen ADC dönüştürme şeması düşük-maliyetli bir DAC kullanmaktadır. DAC çıkışı, yine düşük-maliyetli bir parça olan bir karşılaştırıcı kullanılarak, bilinmeyen gerilim ile karşılaştırılmaktadır.

AD dönüşümünü gerçekleştirmek için mikroişlemci, girişi çıkışı bilinmeyen gerilimle eşitlenene kadar, DAC'a ayarlar. DAC'a sayısal giriş daha sonra bilinmeyen bir analog gerilimin temsili durumuna gelir. ADC girişinin, iki karşılaştırıcı girişi eşit oluncaya kadar, ayarlanma yöntemi, dönüştürmeyi uygulamak için kullanılan programa bağlıdır. Olası yöntemlerden birini anlatan akış diyagramı Şekil 6.11'de gösterilmek-

tedir. Bu akış diyagramı, DAC'dan gelen giriş, bilinmeyen gerilimden daha büyük olduğu zaman, karşılaştırıcı çıkışının 0 dan 1'e kadar değiştiğini varsaymaktadır. İyi bilinen ardışık-yaklaştırma algoritması, daha çıktır ve programı daha karmaşık duruma getirmesine karşın, ortalama olarak daha hızlı bir dönüştürme verir. Şekil 6.11'de gösterilen basit sayma algoritmasının hızı öngörülen uygulama için yeterinden fazla olduğundan, bu konu burada belki de dikkate alınmamıştır.

İki teknik seçenek arasında seçim yaparken, üretilen toplam sistem sayısı çok önemli bir etkendir. Bu konu Kısım 6.4'te nicelik açısından tartışılmıştır. Aşağıdaki çözümlü örnek bu niceliğe ilişkin yaklaşımı, iki AD dönüştürme yöntemindeki maliyetlerin değerlendirilmesine uygulamaktadır.

6.8 Çözümlü örnek

Bir tasarım ekibinin başkanı, AD dönüştürmesi yönteminin aşağıdaki seçeneklerden hangisi ile uygulanacağına karar vermeye çalışmaktadır:

1 Şekil 6.10'da gösterildiği gibi bir markasız ADC kullanılarak;
2 Şekil 6.9'da gösterildiği gibi bir DAC ve karşılaştırıcı yardımıyla.

Her iki şema için başlangıç evresi eleman ve yazılım tahminleri aşağıdaki gibidir:

	Şema 1	*Şema 2*
Elemanlar	ADC + 1 tane IC	4 tane IC
	Toplam maliyet 30 sterlin	Toplam maliyet 8 sterlin
Yazılım	Bellekte 8 bayt işgal eden	Bellekte 60 bayt işgal eden
	5 komut	40 komut

Sözkonusu firma, donanım tasarımı ve üretim maliyetleri için, elemanlar hariç, kullanılacak IC veya başka modüllerin sayısını esas almaktadır. Yazılım tasarım maliyeti ise komut sayısı tabanındadır. Yazılım üretim maliyeti program tarafından kaplanan bellek konumları ile sabitlenmiştir. Maliyetler aşağıda gösterilmektedir:

Donanım tasarımı	IC, veya ADC gibi diğer modüllerin her biri için 20 sterlin
Donanım üretimi	Her bir IC için 2 sterlin
Yazılım tasarımı	Her komut için 5 sterlin
Yazılım üretimi	Her bayt için 1/100 sterlin

(a) 5 birim, *(b)* 25 birimlik üretim turları için, Şema 1 ve Şema 2'ye göre gerçekleştirildiğinde oluşacak bir ADC maliyetini hesaplayınız.

AD dönüştürme maliyetlerinin Şema 1 ve Şema 2'deki uygulamalar ile yaklaşık olarak eşit bulunacağı üretim hacmi değerini hesaplayınız.

Çözüm

Toplam tasarım maliyetine D, bir birimin üretim maliyetine P, bir birim için eleman maliyetine C ve üretilen birim sayısına N diyelim.

O halde bir birim için maliyet = $P + C + (D/N)$ olur.

Şema 1

Donanım tasarımı maliyeti= 2 x 20£ = 40£
Yazılım tasarımı maliyeti= 5 x 5£ = 25£
Dolayısıyla D = 40£ + 25£ = 65£
C = 35£

Bir birim için donanım üretim maliyeti= 2 x 2£ = 4£
Bir birim için yazılım üretim maliyeti= 8 x 1p = 0.08£
Bu nedenle P = 4.08£
Dolayısıyla 5 birim için birim başına maliyet
4.08£ + 30£ + (65/5£) = 47.08£
ve 25 birim için birim başına maliyet
4.08£ + 30£ + (65/25£) = 36.68£ olur.

Şema 2

Donanım tasarımı maliyeti= 4 x 20£ = 80£
Yazılım tasarımı maliyeti= 40 x 5£ = 200£
Dolayısıyla D = 80£ + 200£ = 280£
C = 8£
Bir birim için donanım üretim maliyeti= 4 x 2£ = 8£
Bir birim için yazılım üretim maliyeti= 40 x 1p = 0.40£
Bu nedenle P = 8.40£
Dolayısıyla 5 birim için birim başına maliyet
8.40£ + 8£ + (280/5£) = 72.40£
ve 25 birim için birim başına maliyet
8.40£ + 8£ + (280/25£) = 27.60£ olur.
"Denk düşen" üretim sayısı seçeneğini saptamak için:
4.08£ + 30£ + (65/N£) = 8.40£ + 8£ + (280/N£)

Yeniden düzenlersek
34.08£ - 16.40£ = (280£ - 65£)/N
Dolayısıyla
N = (280 - 65)/(34.08 - 16.40)
= 12.16

Bu analizden de anlaşılacağı gibi, üretilen sistem sayısı 12'den fazla olmadıkça, Şema 2'deki düşük eleman maliyeti çok daha yüksek olan tasarım/geliştirme maliyetini kaldırmaz. Okuyucu bu örnekte verilen maliyet rakamlarının her dönemdeki gerçek eleman maliyeti olduğunu düşünmemelidir. ADC ve DAC gibi birimlerin fiyatları teknoloji ilerledikçe değişmektedir. Uzun vadede donanım eleman maliyetleri azalma, tasarım/geliştirme maliyetleri ise, özellikle yazılım olmak üzere artma eğilimindedir. Bu, yazılım tasarımının emek-yoğun niteliğini yansıtmaktadır. Okuyucu uzun vadedeki bu eğilimin, yukarıdaki örnekte ele alınan durumdaki gibi, tasarım kararlarına etkisi konusunda varsayımlar geliştirebilir.

Sorular

6.1 Mikroişlemci içermeyen bir sayısal sistem geliştirilirken dikkate alınacak iki önemli unsur, eleman seçimi ve ilgili devrenin tasarımıdır. Mikroişlemci-tabanlı sayısal bir sistem geliştirilirken ek olarak hangi etkeni dikkate almak gerekir?

6.2 Tipik bir mikroişlemci-tabanlı sistemin tasarım/geliştirme çalışmasındaki çeşitli aşamaları bir akış diyagramı ile anlatınız.

6.3 Bir mikroişlemci sistemin donanımındaki bazı parçaların ilgili yazılımdan bağımsız olarak neden test edilemeyeceğini açıklayınız.

6.4 Bir basınç transdüserinin çıkışı, ADC aracılığıyla sayısallaştırılması gereken bir gerilimdir. Basınç 0 ile 100 N/m^2 arasında değişmekte ve kullanıcı 0.25 N/m^2 düzeyinde bir çözünürlük istemektedir. Bu çözünürlüğü elde etmek için ADC çıkışın olması gereken minimum bit sayısını hesaplayınız.

6.5 Bir ADC'ye giriş, 0 ile 10 V arasında değişmektedir. Transdüser çıkışının 12-bitlik bir ikili sayı olması halinde çözünürlüğü volt cinsinden hesaplayınız.

6.6 Bir mikroişlemci sistem tasarımı belirli tipte bir adres kod çözücüsünün kullanımını öngörmektedir, bu adres kod çözücüsü için uygun bir entegre

devre kullanıma hazır bulunmaktadır. Tasarım ayrıca, bağımsız kapıları kullanılarak da gerçekleştirilebilmektedir, bu durumda üç devre gerekli olacaktır. Sözkonusu firma sayısal donanım tasarım mal her bir entegre devre için 30 sterlin, ve üretim maliyetini (elemanla olmak üzere) her bir entegre devre için 1 sterlin olarak tahmin etme Bileşen maliyetleri aşağıdadır:

Tek bir entegre devre	5 sterlin
Bağımsız kapılar kullanan devre	2 sterlin

Bağımsız kapıların kullanıldığı tasarımın, tek bir entegre devrenin kul dığı durumdan daha etkin-maliyetli olduğu üretim düzeyini hesaplay

6.7 Bir mikrobilgisayar sistemi, aşağıda gösterilen farklı donanım ve y maliyetleri ile, A ve B şeklinde iki yolla uygulanabilmektedir:

	A. £	B, £
Donanım tasarım/geliştirme maliyeti	5.000	1.000
Her bir sistem için (elemanlar dahil olmak üzere) donanım üretim maliyeti	200	100
Yazılım tasarım/geliştirme maliyeti	10.000	20.000
Her bir sistem için yazılım üretim maliyeti	20	40

B'nin A'dan daha verimli-maliyetli duruma gelmesinden önce, üretil gereken sistem sayısını hesaplayınız.

Bölüm 7 Mikroişlemci Geliştirme Sistemleri

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde;*

1 *Mikroişlemci geliştirme sisteminin, bir mikroişlemci sistem tasarım/geliştirme çalışmalarındaki;*
   - *(a) Sistem programının geliştirilmesi ve yazılması*
   - *(b) Sistem programının test edilmesi*
   - *(c) Gerektiğinde program üzerinde değişiklikler yapılması*
   - *(d) Sistemin donanım ve yazılım simülasyonu/emülasyonu aşamalarında kullanıldığını değerlendirebilmeli,*

2 *Bir geliştirme sistemi içinde bulunan tipik olanakların;*
   - *(a) İşletim sistemi ve izleme programı, düzenleyici, çevirici, hata ayıklayıcı, PROM progamcısı, yükleyici, bağlayıcı konumlayıcı ve yazılım izlemeden oluşan bir işletim programı.*
   - *(b) Program geliştirme uygulamaları için; işletim sistemi için, kalıcı yığın bellek için; Bellek,*
   - *(c) Yazıcı ile birlikte bir VDT ve klavyeden oluşan konsol.*
   - *(d) PROM programlayıcısı*
   - *(e) G/Ç olanakları*
   - *(f) Devre-içi emülatör*

   olduğunu değerlendirebilmelisiniz.

7.1 Donanım ve yazılım geliştirme

6. Bölüm'de, bir mikroişlemci sisteminin geliştirilmesi sırasında başlangıç aşamasındaki tasarım kararlarının nasıl alındıklarını ele aldık. Donanım ve yazılım tasarımının daha ileri aşamalarında kullanılan yöntemleri incelemeye geçmeden önce elimizde bulunan gereçleri gözden geçirmek yararlı olacaktır. Bu gereçlerden en önemlisi, mikroişlemci geliştirme sistemi ya da MDS'dir, MDS kısaca Bölüm 5'te tanıtılmıştır. Şimdi tipik bir MDS tarafından sağlanan olanakları daha ayrıntılı olarak ele alacağız.

"Mikroişlemci geliştirme sistemi" (MDS) adı, ve hep birlikte, kullanıcıya mikroişlemci sistemlerin tasarım ve geliştirilmesinde yardımcı olanaklar olan birimler topluluğunu, yani donanımı ve programlar yani yazılımı tanımlamak için kullanıl-

maktadır. Bu tipteki geliştirme sistemleri karmaşıklık bakımından çok değişk gösterirler ve maliyetleri, sundukları olanaklara bağlıdır. Ancak bunların tem özelliği, herhangi bir bilgisayar içeriyor olmalarıdır, bu bilgisayar bir mikroişl olabildiği gibi bir minibilgisayar ya da başka kullanıcılarla ortaklaşa kull büyük bir anabilgisayar dahi olabilir.

Bölüm 6'da gördüğümüz gibi bir mikroişlemci sistemi geliştirmek için iki bileş tasarlanması gerekir: donanım ve yazılım. Şekil 6.1'de, bu donanım ve ya tasarımı çalışmasının en azından bir kısmının birbirine paralel biçimde sürdür bileceğini göstermektedir. Komple bir MDS hem donanım hem de yazıl tasarımına yardımcı olur ve geliştirmenin bu iki bileşenini bir araya ge aşamasında özellikle yararlıdır. "Geliştirme sistemi" terimi bazen yalnızca ya tasarımı ve yazılım geliştirmeyi destekleyen bir sistemi tanımlamak için de k nılmaktadır. Fakat bunun için, "yazılım geliştirme istasyonu" adı daha uygu Ancak ister bir yazılım geliştirme istasyonu kullanıyor olalım, isterse tam bir M kullanıyor olalım, kullanmakta olduğumuz olanak bize, geliştirilmekte olan si için gereken programların yazılması, test edilmesi ve değiştirilmesi konula yardım sağlamalıdır.

7.2 Geliştirme sistemi düzenlemesi

Tipik bir MDS'nin bileşenleri, Bölüm 5.4 ve Şekil 5.4'te anlatılmıştır. Bütü bakımından sistem bileşenleri burada da yeniden listelenmiştir. Tipik bir tam çekli geliştirme sistemi aşağıdakilerden oluşur:

1 Diğer sistem bileşenleriyle G/Ç arabirimleri bulunan bir MİB.
2 Kullanıcı tarafından geliştirilen uygulama programlarının ve kullanıcının tırdığı çevirici, derleyici vb. destek programlarının saklandığı RAM. saklamanın bu çeşidi ana bellek olarak anılmaktadır.
3 Yedek yığın bellek. Kalıcı özelliktedir ve büyük çoğunlukla bir manyetik sistemi biçimindedir. Bu saklama ortamı, kullanıcının programlarını ve de programları, yürütümleri gerekli olmadığı zamanlarda saklamak için bir a sistemi olarak kullanılmaktadır. Bir programın yürütümünü gerçekleşti için, o programın yedek bellekten ana RAM'e belleğe aktarılması gerekir.
4 Normalde bir ekran ve daktilo-tipinde klavye ile birlikte video görüntü ortamından (VDT) oluşan bir konsol. Bunlar kullanıcı ve geliştirme sistem birbiriyle iletişim kurmasına yarayan araçlardır.
5 Programların ve verinin kalıcı bir kaydının edinilmesini sağlayan yazıcı.

Buraya kadar listelenen olanaklar esas olarak bir yazılım geliştirme istasyonunun olanaklarıdır. Tam-ölçekli bir MDS, ek olarak aşağıdakileri de içerecektir:

6 Programlanır sadece-okunur bellek (PROM), yongalarında veri veya program kodunun kalıcı olarak saklanmasına olanak veren bir birim. Pratikte programcılar, geliştirme amaçları için daha uygun olduklarından silinebilir PROM'ları (EPROM'lar) programlamaktadır.
7 Bir devre-içi emülatör. Bu birim, donanım geliştirme ve donanım/yazılım entegrasyonu amaçlarına yönelik yardımlar sağlayan çok önemli bir birimdir. İşlevleri Bölüm 7.5'te anlatılmıştır.
8 Genel-amaçlı veya "yedek" G/Ç portları. Bunlar, kullanıcı tarafından, gereken her türlü özel-amaçlı çevresel birimin birleştirilmesi için kullanılabilirler. Bu G/Ç portları, sözgelimi, geliştirilmekte yani "hedef" sistemdeki çevresel birimlere yönelik G/Ç programlarını test etmek amacıyla kullanılabilirler.

Yukarıda listelenen bileşenler MDS donanımını oluşturur. Ancak her türlü bilgi işlem sisteminde olduğu gibi, buraya kadar belirtilenler olayın yalnızca bir bölümüdür. Sistemin gözümüze daha az çarpan bölümü ise, destek yazılımıdır.

7.3 Destek yazılımı

Bu bölümde kullanıcıya, diğer programları geliştirmesinde yardımcı olan programları tartışacağız. Bir MDS seçerken, destek yazılımın kalite ve kapsamı genellikle donanımdan daha önemlidir. Bu, iyi tasarımlanmış ve güvenilir bir donanımın önemli olmadığı anlamına gelmez. Ancak, kullanıcının diğer programları geliştirmesindeki hız ve elde ettiği kolaylık destek yazılımı tarafından belirlenir. MDS'nin donanım sorunlarının çözümü için bulunan bileşenleri - örneğin devre-içi emülatör, bile denetleyen destek yazılıma bağımlıdır.

Destek yazılım normalde birkaç ayrı program ya da "yazılım paketleri" biçiminde gelir, bunlardan her biri tanımlanmış bir işleve sahiptir. İzleyen bölümlerde MDS ile birlikte sağlanan standart yazılım paketlerinin bazılarını anlatacağız.

İşletim sistemi

Daha önce açıklandığı gibi, bir yazılım geliştirme istasyonu ya da MDS tarafından sağlanan program geliştirme desteği, çeşitli yazılım paketlerinden oluşur. Bunlar bir çeviri, düzenleyici, izleme programı/hata algılayıcı, derleyici vb.'dir. Yazılım paketleri normalde disket gibi bir yedek bellekte saklanırlar. Belli bir paket, örneğin bir düzenleyici gerekli olduğunda, bu paket yedek saklama ortamından ana RAM'e

aktarılmalıdır. Kullanıcı tarafında yaratılan programlar ve verinin de ana ve yedek bellek arasında karşılıklı olarak aktarılmaları gerekir.

Yazılım geliştirme istasyonunda, ana ve yedek bellek arasındaki bu program veri aktarımını düzenlemek için, bir işletim sistemi bulunacaktır. Bazen disk işletim sistemi adı verilen bu sistemin kendisi de bir programdır. Kullanıcı ve geliştirme sisteminin donanım/yazılım olanakları arasında bir arabirim gibi işlev görür. Kullanıcı, klavye aracılığıyla işletim sistemine komutlar girebilir. İşletim sistemi kullanıcıya gönderilen mesajlar sistem konsolunun ekranında görüntülenir.

Metin düzenleyici

Hem assembly hem de yüksek-düzeyli dilde program geliştirme desteği herhangi bir biçimde bir metin düzenleyiciye gereksinim vardır. Bir metin düzenleyici (ya da kısaca düzenleyici), kullanıcının, kodlanmış metni sistem klavyesi aracılığıyla sistem belleğine girebileceği bir programdır. Metin, bir kez girildikten sonra, satır ve karakter silme ve ekleme gibi çeşitli işlemlerle işlenebilir. Sakladığı metnin içeriği, sistem ekranı ya da yazıcı kullanılarak görüntülenir. Yedek saklama ortamı içeren bir sistemde düzenleyici, kodlanmış metnin RAM'dan yedek saklama ortamındaki bir dosyaya yazılması ve yedek saklama ortamındaki bir dosyanın RAM'a okunması için komut girilebilmesine izin verir. Bu ise, metin parçalarının isteğe göre saklanması ve çağrılmasına olanak tanır. Her ne kadar burada yüksek-düzeyli ya da assembly dilindeki programların hazırlanmasını sağlayan düzenleyiciyi tanımlıyorsak da, her türlü metin benzer biçimde işlenebilir. Düzenleyici ilkel bir kelime işlemci olarak kabul edebiliriz.

Metin düzenleyici, program kodunun girilmesi veya değiştirilmesi sırasında programcı için temel bir gereç olduğundan, birçok kişi, metin düzenleyiciyi bütün yazılımın en önemli bileşeni olarak kabul eder. Program değişikliklerinin yapılmasındaki hız, bir programcının verimliliğini ve dolayısıyla da program geliştirme maliyetini doğrudan etkileyecektir. Normalde programcıların çalışma zamanının, diğer destek yazılım paketlerine göre daha fazla bir bölümünü metin düzenleyiciler almaktadır, hata ayıklayıcı burada bir istisna oluşturabilir.

Metin düzenleyiciler iki biçimdedir. Daha çok bilineni, satır modunda çalışan; bu durumda program her zaman, kodlanmış metin içinde o anda üzerinde bulunduğu yeri tanımlayan bir işaretçiye sahiptir. Bu işaretçinin metin üzerindeki yeri, konsol klavyesi aracılığıyla komutlar verilerek değiştirilebilir. Böyle bir komutun tipik örneği "Move" komutudur, klavyede "nM" yazılarak uygulanır, burada n, işaretçinin hareket edeceği karakter pozisyonu sayısıdır. Daha açık anlatabilmek

için, bir imlecin aşağıdaki metin satırında okun gösterdiği yerde bulunduğunu varsayalım:

```
THIS  IS A  XX   FAULTY   LINE
  ↑
```

İşaretçinin her zaman iki karakter pozisyonu arasındaki konumlarda bulunduğuna ve boş yerlerin de birer karakter olarak kabul edilmesi gerektiğine dikkat ediniz. İşaretçiyi xx karakter çiftinin başlangıcına götürmek için "8M" komutu girilir, bu komut işaretçiyi 8 konum ileriye getirir. Ardından, işaretçi aşağıda gösterildiği gibi konumlanır.

```
THIS  IS A  XX   FAULTY   LINE
          ↑
```

Bundan sonra, işaretçinin bulunduğu yerden başlamak üzere "n" sayıda karakteri silmek için başka bir komut, "nD" komutu girilebilir. Böyle bir durumda, örneğin "10D" komutu 10 (ondalık) karakteri siler, satırın son durumu ve işaretçinin konumu aşağıdaki gösterildiği gibi olur:

```
THIS  IS A  XX   LINE
               ↑
```

Burada verilen örnekle okuyucuya, bir satır düzenleyicisinin nasıl kullanıldığına ilişkin bir fikir verilmesi amaçlanmıştır. Komutların biçiminin farklı düzenleyicilerde değişeceği açıktır. Anlatılan komutlar tipik komutlardır. Bulunan diğer komutlardan bazıları ise kullanıcının karakter dizileri eklemesini ve değiştirmesini sağlarlar. Bazı düzenleyiciler ayrıca, metin bloklarının taşınmasına da olanak tanırlar.

Her ne kadar programcıların bir bölümü yukarıda açıklanan gibi bir satır düzenleyicinin kullanımında fazla deneyim kazanabilecekse de, bu düzenleyicinin nispeten yavaş ve yetersiz bir metin işlem düzenleyicisi olduğuna kuşku yoktur. Yeni mikroişlemci geliştirme sistemlerinin hemen hemen tümünde denetimli imleç kullanımına dayalı daha üst düzeyde bir düzenleyici seçeneği sunulmaktadır. İmleç genellikle, aşağıda gösterildiği gibi, konsol ekranındaki bir karakter konumunun altında yanıp sönen bir çizgidir:

```
THIS  IS A  XX   FAULTY   LINE
```

Yanıp sönen imleç

Metin silme ve ekleme komutları sıklıkla yukarıda anlatılan satır düzenle
komutlara benzer. Denetimli imleç kullanımının temel avantajları yer
zaman hemen görülebilir olması ve ekranda kolaylıkla hareket ettirilebil
Normalde, konsol klavyesinde, imleci ekranda çeşitli yerlere götürebilmek
tuşlar bulunur. Yaygın olan düzenleme, klavyede yukarı, aşağı, sağa
hareketi sağlayan dört tuşun yer almasıdır.

Denetimli doğru imleç tipi metin düzenleyici kullanımı, özel tipte bir VDT
Display Terminal)'nin mevcut bulunmasını gerektirir. Böyle bir görüntüle
minin en önemli özelliği, bağlı bulunduğu bilgisayar sisteminden gelen im
netim sinyallerini kabul etmesidir. Bu imleç-denetim sinyalleri düz
program tarafından, kullanıcının klavye aracılığıyla girdiği komutlara yanıt
S7.5 üretilir. Daha basit tipte bir VDT, imleç-denetim özelliğine sahip değilde
7.6 nedenle denetimli-imleç tipinde düzenleyici kullanımına olanak tanımaz.

Çevirici

Assembly dili ve tipik çeviricilerin özellikleri Bölüm 1'de ayrıntılı olarak
mıştır. Bu bölümde okuyucuya, bir çeviricinin, mikroişlemci sistem yazılım
rımcısı için önemli olan bazı belirli özellikleri anımsatılacaktır. Bütün çev
temelde, komut kısaltmaları biçiminde yazılmış bulunan metni makine
dönüştürme olanağı sunarlar. Fakat yalnızca bu olanağı sunan bir çevirici
müzde çok sınırlı kullanımı olan bir çevirici olarak kabul edilir.

Bölüm 8'de, modern program tasarım yöntemlerinin, büyük programları
nedenlerle birkaç bölüm ya da modül olarak tasarlamayı gerektirdiğini gö
Bu modüller daha sonra, bütün bir programı oluşturmak için birbirine bağla
Bağlanmanın assembly dili programlama ile uygun biçimde yapılabilme
bağlayıcı adı verilen başka bir yazılım paketi ile bir arada çalışabilen
çeviriciye gerek vardır. Bağlayıcılar bir sonraki bölümde anlatılmıştır.
bağlama olanağı tanıyan çeviriciler hemen her zaman yeri değiştirilebilir
üretimine izin verirler. Yeri değiştirilebilir programlar, Bölüm 1'de anlatılmı
bunlar, taşıyıcı veya yer değiştirici adı verilen başka bir özel yazılım pa
kullanılmasını gerektirirler. Taşıyıcılar aşağıda anlatılmıştır. Birçok sistemde
layıcı ve taşıyıcının işlevleri tek bir yazılım paketinde birleştirilmiştir.

Yazılımın büyük bir bölümü, yüksek-düzeyli bir dille yazılacak olsa bile, mi
lemci sistemlerin tasarımında verimli bir çeviricinin hazır bulunması yine öne
Çünkü, doğrudan sistem donanımı ile etkileşime giren programlar için as
dilinden yararlanmak hemen her zaman gereklidir. Örneğin, ana işlevi stok

timi olan bir sistem tasarımı yaptığımızı varsayalım. Programlama zamanının kullanımı açısından yüksek düzeyli bir dille karşılaştırıldığında ekonomik olması hiç bir zaman söz konusu olmadığından, deneyimli hiç bir programcı assembly dilini kullanarak bir stok denetim programı yazmayı düşünmez. Ancak bazı aşamalarda, sistemin dış dünyadan veri alması ve dış dünyaya veri iletmesi gerekir. Giriş birimi bir klavye ya da VDT, ve çıkış birimi de kolaylıkla bir yazıcı olabilir.

Klavye ve yazıcı gibi çevresel birimleri denetleyen ve bunlarla iletişim kuran programlar için assembly dili, olası tek seçenek durumunda bulunabilir. Dolayısıyla yukarıda öngörülen stok denetim programları, en etkin biçimde, çevresel denetim gereken her yerde assembly-diliyle-üretilmiş makine-kodu bölümlerinin ekli olduğu bir yüksek-düzeyli dil kullanılarak gerçekleştirilebilir. Bu nedenle MDS yazılımının bu tipte "karma" programların kullanımına olanak tanıması gereklidir.

Bağlayıcılar

Bölüm 1'de ve bir önceki kısımda modüler programlamanın önemi tanıtılmıştır. Bağlayıcı, program modülü bağlama işlemini daha uygun biçime getirmek için çeviriciyle birlikte çalışan bir yazılım paketidir. Diğer olanaklarla birlikte verimli bir bağlayıcı/çevirici birleşimi, "global" şeklinde etiketler bildirilmesine olanak tanır. Bu, sözü edilen etiketlerin kullanılabileceğini ve etiketlerin bağlanmış tüm modüllerde aynı anlama sahip olacaklarını göstermektedir. Dolayısıyla bir modülde yaratılan veri bir başka modülde kullanılabilir. Programları bir bağlayıcı olmaksızın modüler biçimde yazmak mümkündür fakat bu işlem çok daha elverişsiz bir işlem olacaktır.

Bazen bağlayıcılar, kaynak program düzeyinde çalışırlar, bu, tek bir bağlanmış program oluşturmak için çeşitli modüllerin orijinal kaynak kodunda değişiklik yapıldığı anlamına gelir. Bağlanma işleminin amaç-kodu düzeyinde gerçekleşmesi daha sık karşılaşılan bir durumdur. Böyle bir durum söz konusu ise çevirici çeşitli kaynak program modüllerini ayrı ayrı çevirir ve sonuçta elde edilen amaç-kodu modüllerini ayrı disk dosyalarında saklar. Bunun ardından bağlayıcı bu amaç programları, bağlanmış tek bir amaç programın oluşması için gerekli değişiklikleri yaparak birleştirir.

Yerleştiriciler

Bölüm 1'de gördüğümüz gibi birçok çevirici, bir programın, ya mutlak ya da yeri değiştirilebilir şeklinde bildirilmesine olanak tanımaktadır. Yeri değiştirilebilir bir programda amaç kodun bellekte nerede saklanması gerektiği, kaynak programda

belirtilmez. Bunun yerine, bu karar, yerleştirme ya da yer değiştirme işlemi g
leşinceye kadar ertelenir. Taşıyıcı program, kullanıcının, bir yeri değiştiril
programın başlama adresini girmesine olanak tanır. Daha sonra, belirtilen kom
başarılı şekilde çalıştırılmasını sağlayabilmek için, program amaç kodunda ge
S7.7 değişiklikler, taşıyıcı tarafından yapılır. Bu işlem Bölüm 1'de daha ayrıntılı
7.9 anlatılmıştır.

Yüksek-düzeyli diller

Assembly dilinin alternatifi, PASCAL, COBOL, BASIC vb. gibi bir yüksek-d
dildir. Bu yüksek-düzeyli dillerde tek bir deyim birçok makine-kodu komu
dönüşür. Dolayısıyla programcının verimliliği etkin biçimde artar. Bu ned
yüksek-düzeyli diller mümkün olan her yerde assemly diline tercih edilmeli

X=Y+Z

gibi bir yüksek-düzeyli dil deyim dizisini, daha sonra çalıştırılabilen bir mak
kodu komut dizisine dönüştüren programa derleyici adı verilir. Yüksek-düz
dilden makinekoduna çevirme işlemi yapan bir başka program tipi de yorumla
cıdır. Özel amaçlı sistemler için program yazarken normalde, yorumlanmış y
sek-düzeyli dilden çok, derlenmiş yüksek-düzeyli dil gerekmektedir. Bunun ned
çoğunlukla yorumlayıcının iç çalışması, derleyiciden, ROM'da saklanabilenl
kine-kodu program üretimine uygun olmayan biçimde farklı olmasıdır. Bu
üstesinden gelmek için yöntemler bulunmaktadır fakat normalde yorumlam
yüksek-düzeyli dil yerine derlenmiş yüksek-düzeyli dili tercih etmek daha elve
lidir.

İzleme programı/hata ayıklama programı

Program hata ayıklama olanakları, kullanıcının kendi programlarını oluşturmas
izin veren her sistemin temel gereksinimidir. Bilgisayar, giriş/çıkışın onaltılı klav
ve ekran ile sınırlı olduğu bir mikroişlemci değerlendirme sistemi kadar b
yapıda olsa bile, bu gereksinim yine sözkonusudur. Bu sistemleri kullananlar, ço
kez sağlanan tek sistem yazılımı olan izleme programını bilirler. İzleme progra
yardımıyla kullanıcı, makine-kodu komutları için onaltılı kod girebilir, bunlar da
sonra bellekte saklanır. Onaltılı klavye aracılığıyla girilen komutlar program
durma noktaları girilmesine olanak tanırlar ve programlar sürekli biçimde, veya b
seferde bir komut biçiminde yürütülürler. Kullanıcı bu olanaklardan yararlanar
basit programların hata ayıklama işlemini gerçekleştirebilir.

Yazılım geliştirme istasyonları ve MDS'ler genellikle, yukarıda anlatılanlardan daha kapsamlı izleme programı/hata ayıklama olanakları sağlarlar. Sistem klavyesi aracılığıyla girilen komutlar, bellek konumları ya da çalışan kaydedici içeriklerini görüntüleme ve değiştirme, program yürütümü, tek-adımlı yürütüm vb. gibi bilinen işlevleri yapabilmeye olanak verirler. Ek olarak, uygun komutu girip yedek saklama ortamından RAM'e veri ya da program yüklemesi yapmak ta olanaklıdır. Veri ve programlar benzer şekilde, RAM'den yedek saklama ortamında tutulan bir dosyaya yazılabilir. Disk-tabanlı bir sistemde bulunan tipik bir izleme programı/-hata ayıklayıcı tarafından sağlanan olanaklar aşağıda gösterilmektedir. Bu izleme programına DDT (Dinamik Hata Ayıklama Gereci) adı verilmekte ve 8080 programlarında çalışmaktadır. DDT, bilinen CP/M disk işletim sistemi ile birlikte kullanıma sunulan yazılım paket takımının bir parçasıdır. Komut listeleri, bir komut el kitabının işini görmesi için değil, okuyucuya tipik bir disk-tabanlı izleme programının sağladığı olanaklar hakkında fikir vermek amacıyla eklenmiştir.

DDT komutları

A (Assemble) "As" komutu DDT'yi, komut kısaltmalarını çevirmeye (assemble etmeye) başlamak ve sonuçta oluşan makine komut kodlarını "s" onaltılı adresinden başlayarak saklamak üzere yönlendirir. Komutun klavye aracılığıyla girilmesinden sonra DDT, kullanıcıyı, ilk komut kısaltması için uyarır, örneğin A1000 girilmesi ve daha sonra MOV E,A girilmesi, 5F kodunun 1000 konumuna saklanmasına neden olur.

D (Görüntüle: Display) "D" komutu bellek içeriğinin görüntülenmesine izin verir. Bu komutun üç formu bulunmaktadır:

D	Bellek içeriği yürürlükteki görüntüleme adresinden görüntülenir ve 16 görüntüleme satırı boyunca devam eder;
Ds	"D" komutuna benzer fakat belleğin görüntülenen bölgesi "s" konumundan başlar.
Ds,f	"Ds" komutuna benzer fakat belleğin görüntülenen bölgesi "f" konumunda sona erer.

F(fill:doldur) "F" komutu bir bellek alanını her konumda verilen bir sayı ile doldurur. Genel biçimi aşağıdaki gibidir:

"Fs,f,c" burada "s" başlangıç adresini, "f" tamamlanış adresini ve "c" de saklanacak sayıyı gösterir. Örneğin F1000,1500,FF girilmesi 1000 ve 1500 arasındaki tüm konumların FF sayısı ile doldurulmasına neden olur.

G(Go: git) "G" komutu bir program yürütümünün başlamasına neden olur. G biçiminin çeşitli seçenekleri bulunmaktadır:

G Yürütümü, program sayıcısının yürürlükteki değerinden başlatır;
Gs Yürütümü, program sayıcısı, "s" ye kurulu durumdayken başlatır;
Gs,b "Gs" ye benzer, fakat program durma noktası "b" konumundadır;
Gs,b,c "Gs,b" ye benzer, fakat "c" konumunda ikinci bir program durma no vardır;
G,b Yürütümü, program sayıcısının yürürlükteki değerinden başlatır; "b" d durma noktası vardır;
G,b,c "G,b" ye benzer, "c" konumunda ikinci bir program durma noktası v

Örneğin G1000,1020, yürütümün 1000 den başlamasına ve 1020 de bir du noktasının bulunmasına yol açar; G,1000,1020, yürütümü, program sayıcı yürürlükteki değerinden başlatır ve durma noktaları 1000 ve 1020 ye ayarlanm

L(List: listele) "L" komutu bellekteki makine-kodu komutlarının tekrar uy komut kısaltmalarına dönüşmelerine ve daha sonra da görüntülenmelerine ne olur. Bu işlem disassembly olarak bilinir. Komut üç biçimde olabilmektedir:

L Yürürlükteki listeleme adresinden başlayarak disassemble edilmiş an satıcı komutlardan on iki tanesini görüntüler;
Ls "L" ye benzer, liste adresi "s" ye ayarlanmıştır;
Ls,f "Ls" ye benzer, disassembly "f" konumunda sona erer;

Örneğin 1000-1004 konumları içeriklerinin aşağıdaki gibi olduğunu varsayalı

Konum	*İçerik*
1000	FE
1001	FF
1002	D2
1003	00
1004	15

L1000,1000 komutu aşağıdaki görüntüleme ile sonuçlanır:

```
1000 CPI  FF
1002 JNC  15   00
```

M(Move: Taşı) "M" komutu veri bloğunun bellekte bir alandan diğerine hare

etmesine neden olur. Genel biçimi şöyledir:

Ms,f,d "s" den "f" ye kadar olan bellek konumlarında saklanan veri bloğunu "d" adresi ile başlayan bir alana taşır.

Örneğin M1000,1FFF,3000; komutu 1000-1FFF adresleri içeriğinin 3000-3FFF konumlarına kopyalanmasına yol açar.

S(Set: Ayarla) "S" komutu belirli bellek konumları içeriklerinin incelenmesine ve gerekirse değiştirilmesine izin verir.

Ss "s" konumu içeriği görüntülenir. Bir veri baytının girilmesi "s" de bulunan verinin üzerine yazılmasına neden olur. Bir satır atlatma girilmesi verinin değişmeden kalmasını sağlar. Her iki durumda da bir sonraki daha yüksek adres içeriği benzer biçimde görüntülenir. Nokta (.) girilmesi ise komutu sona erdirir.

T(Trace: izleme) "T" komutu bir programın yürütümüne neden olur ve her komutun çalıştırılmasından sonra bütün kaydedicilerin içeriklerini görüntüler. Bu komut iki biçimdedir:

T Bütün kaydedicilerin yürürlükteki içerikleri görüntülenir ve bir komut yürütülür. Bu komut bir programın adım adım çalıştırılmasına olanak tanır.
Tn "T" ye benzer fakat izlem işlemi "n" (onaltılı) adım süresince devam eder.

U(Untrace: izlemeyi geri çek) "U" komutu "T" ile aynıdır fakat buradaki fark programın ara adımlarının görüntülenmemesidir.

E(Examine: incele) "E" komutu mikroişlemci kaydedicilerinin içeriklerini görüntüler. Kullanıcı ayrıca kaydedici içeriklerini isteğe bağlı olarak değiştirme seçeneğine de sahiptir. Komutun iki biçimi vardır:

X A,BC,DE,HL,S,P ve F kaydedicilerinin içerikleri görüntülenir.
Xr "r" kaydedicisi içeriği görüntülenir. Kullanıcı daha sonra klavye aracılığıyla yeni veri girme seçeneğine sahiptir.

Örneğin KA ve bunun arkasından 3F, A kaydedicisinin 3F ile yüklenmesine neden olur.

Diğer komutlar işletim sistemi (CP/M) ile etkileşen I (giriş) ve R(okuma) komutlarıdır.

7.4 Çapraz-yazılım

Bir kuruluş mikroişlemci-tabanlı sistemlerin tasarım ve geliştirmesini destekl
bir olanak düzenlerken izleyebileceği iki seçenek vardır. Birisi, özel olarak ka
nılacak belirli mikroişlemcileri desteklemek için bir ya da daha fazla MDS
minde donanım satın almaktır. Diğer seçenek ise, halen kullanılabilecek duru
hazır bulunan güçlü bir minibilgisayar ya da anabilgisayarı kullanmaktır, bu
çapraz-yazılım genel adı altında yer alan özel programların yardımıyla kullanılı

Çapraz-çeviriciler

Çeviricilerin işlevini temel olarak, mikroişlemci imalatçısının komut kısaltma
ile yazılmış bir programı ikili makine koduna dönüştürmek şeklinde tanımlam
Bu işlem, uygun bir programın sağlandığı herhangi bir bilgisayar tarafından
çekleştirilebileçek bir veri dönüştürme işlemidir. Komut kısaltmalarından-mak
koduna dönüştürmeyi yapan bilgisayar, makine kodunun çalıştırılacağı bilgisa
ile aynı tipte değilse bu işleme çapraz-çevirme adı verilir. Yeterli sayıda çapra
S7.10 virici bulunduğu zaman bir programcı yalnızca bir bilgisayar kullanarak çe
7.11 mikroişlemciler için programlar yazabilir.

Simülatörler ve emülatörler

Çapraz-çevirici kavramı, uygulaması ve kullanımı, çeviriciden biraz daha farklı
Ancak programın test edilmesi ve hata ayıklama işleminin yapılması gerektiğind
makine-kodu programın çevrilmesi, programlama işleminin sonu değildir. E
bilgisayarın makine-kodu programı, farklı bir bilgisayarda çalıştırılan bir çap
çevirici tarafından üretilirse ortaya bir sorun çıkar. Örneğin bir M6800 mikroişl
cisine makine-kodu program üretmek için bir PDP11 minibilgisayar
çapraz-çevirici kullanabiliriz. Ancak M6800 makine-kodu PPD11'in makine-ko
ile aynı olmadığından M6800 makine-kodu programı PDP11 tarafından hem
çalıştırılamaz.

M6800 makine-kodu programının doğru olduğundan kesinlikle emin olmamız
bu programın bir M6800'de çalışması ve doğru sonuçları vermesi gerekir. Ö
M6800 donanımı satın almadan bu sorunun kısmen üstesinden gelmenin bir yo
simülatör veya emülatör adı verilen bir program kullanmaktır. Simülatör ya
emülatör normalde, çapraz-çevirici ile aynı bilgisayar üzerinde çalışır. Bu, bilgi
yarın M6800 gibi diğer işlemcilerin programlarını çalıştırabilmesine olanak ve
Örneğin, benzetimci yardımıyla PDP11, verimli şekilde bir M6800 "rolü" oynar
M6800 makine-kodu komutlarını çalıştırır.

Bütün mikroişlemci çeşitleri için yazılım geliştirme desteği verebilen bir bilgisayarın kullanımı çok ilgi çekici bir düşüncedir. Ancak bu yaklaşımda önemli temel sorunlar bulunmaktadır. Bu sorunlar program hata ayıklaması ve simülatör kullanımı ile ilgili sorunlardır. Bir mikroişlemcinin gerçek pratik yaşamda bir başka bilgisayar tarafından simülasyonu genellikle mümkün değildir. Örneğin 8080'i simüle eden bir minibilgisayarda bir 8080 makine-kodu komutunun yürütümü, aynı komutun gerçek bir 8080 tarafından yapılan yürütümünden genellikle daha uzun zaman alır. Dolayısıyla bir programın süre açısından çok önemli olan bölümleri bu yolla sınanamaz.

Simülatörler, donanım-bağımlı programların test edilmesi ve hata ayıklaması sırasında da sorun çıkarırlar. G/Ç işlemleri bulunduran yordamlar buna iyi birer örnektir. Bu tipte programlar için, az da olsa, programın yazıldığı pratik devreyi içeren özel donanım seçeneği bulunmaktadır.

Zorluklarına karşın çapraz-yazılım, mikroişlemci-tabanlı sistemler için programlar geliştirmede uygulanabilir bir yöntemdir ve birçok tasarım ekibi tarafından başarıyla kullanılmaktadır. Bu yöntemin uygulanmasının en iyi yolu bir çapraz-çevirici kullanarak makine-kodu programlar üretmektir. Bu programlar daha sonra sınama ve hata ayıklama işlemleri için, EPROM'lar ya da alt-yükleme (down-loading) aracılığıyla özel donanım sistemine aktarılırlar. Böyle bir durumda özel donanım sistemi nispeten basit ve ucuz olabilir.

7.5 Donanım destek olanakları

Bu bölümde buraya kadar, MDS tarafından yazılım tasarımı ve geliştirmesine yardımcı olmak için sağlanan olanaklar üzerine eğildik. Sistemlerin çoğunluğunda geliştirme bütçesinin en büyük bölümünü programların üretimi oluşturduğundan, yazılım olanaklarını en ön sıraya koymak doğru ve yerinde bir tutumdur. Ancak donanım geliştirme desteği de önemlidir ve gerçek anlamda bir MDS bunu sağlamalıdır.

Mikroişlemci-tabanlı donanım için kullanılabilir durumda bulunan destek olanakları ele alındığında, donanımın büyük bir kısmının tam anlamıyla test edilebilmesinin yazılım yardımıyla mümkün olabildiğini bilmek önem taşımaktadır. Örneğin belleği yalnızca üzerine veri yazarak ya da kendisinden veri okuyarak test edebiliriz. Bunun en doğal yolu, belleği, bu işlemleri gerçekleştiren ve sonuçları denetleyen bir bilgisayar ile birlikte kullanmaktır. MDS'nin sağladığı kolaylıklar böyle bir test edilmesinin elverişli ve hızlı bir biçimde yapılabilmesine olanak tanımalıdır.

Devre-içi emülatörler (ICE)

Bir devre-içi emülatör, mikroişlemci-tabanlı devrelerin ve, donanımın ve doğrudan ilişkili yazılımın sınanması için belki de en yararlı tek birimdir. D ICE olanağı, MDS'nin ana gövdesinden ayrılan çok yönlü bir kılavuz gibi Kılavuz, emülasyon uygulanan mikroişlemci ile aynı düzenlemeye başlık fişiyle son bulur. Emülatörü kullanmak için kullanıcı, geliştirilmek sistemin mikroişlemci yongasını soketinden çıkarır ve başlık fişini bunun takar. Bu aşamada MDS "host" ve geliştirilmekte olan sistem de "hedef" adlandırılır.

ICE etkin olduğunda host sistem, çalışan bir mikroişlemci yongasının bacaklarındaki sinyallerin tam olarak aynısını prob ucu bacaklarında üretir. Programlar sistemde çalıştırıldığında hedef sisteme, program sanki sistemin kendi içinde mikroişlemcide çalıştırılıyormuş gibi görünür. Farklılık tabii ki, host sistemin hata ayıklama/izleme programının sunduğu bütün olanaklarla birlikte program tam denetimini elinde bulundurmasıdır. Durma noktaları eklenebilir, kaydedilip incelenebilir, tek-adımlık yürütüm uygulanabilir vb. Program host sistemde çalıştırılıyor olsa da hedef sistemin donanımında örneğin giriş veya çıkış işlemleri uygulayan sinyaller üretilebilir. Bu, G/Ç devrelerinin kolaylıkla test edilebilirliğini sağlar. Birçok ICE, işlem zamanlamasının ya host ya da hedef sistemin saat tarafından denetlenmesine olanak tanır. Saat dalgabiçimi gerekli özelliklerinde olmasa da olursa işlem normal seyrini sürdüremeyeceğinden, bu yolla geçersiz bir dalgabiçimi saptanabilir.

Bellek yerini alma, ICE'nin sunduğu bir başka temel kolaylıktır. Bu olanak yardımıyla bellek okuma ve yazma işlemlerine ilişkin adresler ve denetim/durum sinyalleri ya hedef sisteme ya da host sisteme yönlendirilebilir. Dolayısıyla kullanıcı, belleğin belirli bir alanını host veya hedef sistemde yer alması için seçebilir. Bu olanağı kullanarak kullanıcı örneğin, hedef sistemde 4000-5000 alanında bulunan bir veriyi işleyen, host sistemde 2000-3000 alanında saklı bir programı çalıştırabilir. Ya da veri host sistemde, program hedef sistemde bulunabilir. Hedef sistemdeki belleğin kullanılabilirliğini kontrol etmek için bu olanağın yararlı olduğu açıktır.

Bir sonraki kısımda anlatıldığı gibi EPROM programcısıyla birleştirildiğinde bellek değiştirme olanağı sistem yazılımını donanım ile bağlantılı biçimde yerel olarak sınama konusunda büyük oranda yardımcı olabilir. Bu kolaylık en çok, geliştirmenin, ayrı ayrı yerlerde geliştirilen donanım ve yazılımın bir araya getirilmesi işleminin yer aldığı, biraz daha güç olan aşamasında yarar sağlamaktadır. Geliştirme çalışmalarının bu kısmını tamamlarken hedef sistemi, kısmen host sistemde kısmen de hedef sistemde çalıştırılan bir program ile işletmek yaygın bir uygulamadır.

PROM programcısı

MDS genelde, yazılımın bir çeşit ROM içinde bulunmasının gerektiği hedef sistemlerin geliştirilmesinde yardımcı olarak kullanılmaktadır. Bu gibi sistemlerin geliştirilmesinde yaygın olarak, kullanıcı tarafından programlanabilen ROM çeşitleri kullanılır. Geliştirme tamamlandığı zaman PROM, maske-programlı bir birimle değiştirilebilir.

ROM-tabanlı sistemlerin desteklenmesindeki bu gereksinimin ışığında, MDS'lerin çoğunluğunun bir PROM tipini programlama yöntemini de içermeleri şaşırtıcı değildir. Kitabın yazımı sırasında, 2708, 2716, 2732 serilerinden mor ötesi silinebilir PROM'lar veya EPROM'lar gibi birimlerin programlayıcılarının sağlanması en yaygın uygulamaydı. Ek N'de 2716 için bir bilgi sayfası bulunmaktadır.

EPROM programcısının işlevini açıklığa kavuşturmak için tipik bir donanım/yazılım geliştirme çalışmalarını kısaca anlatacağız. Planlama aşamasından sonra donanım/yazılım alanları belirlenir, donanım ve yazılım tasarımı birbiriyle paralel biçimde sürer. Yazılım, program için modüler bir yapı kullanılarak tasarlanır. Donanım tasarlanır ve bir prototip oluşturulur. Bu aşamada donanım ve yazılımın bir araya getirildiği noktaya gelinir. Önce, programın bütünüyle host sistemde saklanmasına olanak veren bellek değişimli ICE kullanılarak donanım/yazılım birleşimi test edilir. Daha sonra tek tek program modülleri test edilir. Bir program modülünün doğruluğu saptandığında, kod, EPROM programlayıcı aracılığıyla EPROM'a aktarılır.

EPROM'da modül kodunu içeren bellek alanı daha sonra hedef sisteme ayrılır. Dolayısıyla bu aşamada program kısmen host sistemde kısmen de hedef sistemde saklanmaktadır. Her bir program modülünün testi sonunda doğru olduğu anlaşıldıkça o modül EPROM'a programlanır. Dolayısıyla program host sistemden hedef sisteme tek tek bitler halinde aktarılmaktadır. Sonunda programın tamamı hedef sistemdeki EPROM'a aktarılır. Bu noktada ICE prob ucu host sistemde mikroişlemci yuvasından çıkarılır ve mikroişlemci yongası yeniden yerine yerleştirilir. Hedef sistem bundan sonra, MDS'den bağımsız olarak kendi başına çalışabilmelidir.

Kaynakça ve bibliyografya

7.1 B. E. Gladstone, "Comparing microcomputer development systems", *puter Design*, pp: 83-90 (February 1979).

7.2 "Microcomputers, part 2: the development system", *Digital Design*, Vol

Sorular

7.1 Tipik bir MDS'nin donanım bileşenlerini sıralayınız.

7.2 Bir yazılım geliştirme istasyonu ile tam-ölçekli MDS arasındaki farkı

7.3 MDS'de yedek saklama ortamının işlevi nedir?

7.4 Tipik bir MDS'nin sağladığı destek yazılımdaki birbirinden bağımsız p leri sıralayınız.

7.5 Metin düzenleyicinin işlevini açıklayınız.

7.6 Satır düzenleyici ve imleç-denetimli düzenleyici arasındaki fark n Bunlardan hangisi daha elverişli olarak kabul edilir?

7.7 Mutlak ve yeri değiştirilebilir assembly dili programları arasındaki f açıklayınız.

7.8 Assembly dili sözde-işlemi ile bir çalıştırılabilir komut arasındaki f açıklayınız.

7.9 Ek D'de bulunan Z80 komut takımı yardımıyla aşağıdakilerden hangile sözde-işlem olduğunu ve hangilerinin çalıştırılabilir komutlar oldu belirtiniz.

PSECT
OUTD
RETI
EJECT

7.10 Çevirici ve çapraz-çevirici arasındaki farkı açıklayınız.

7.11 Özel mikroişlemci sistemleri için program geliştirmede çevirici yerine ç raz çevirici kullanmanın dezavantajları nelerdir?

lüm 8 Yazılım Tasarımı Yöntemleri

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde;*

1 *Mikroişlemci sistemleri için yazılım geliştirmede, aşağıda verilen aşamaları analiz edebilmeli,*
   (a) *Program akış diyagramının geliştirilmesi*
   (b) *Program yazımı*
   (c) *Programın test edilmesi*
   (d) *Gerekli yerlerde değişiklik*
2 *Bilgisayar programlarının tasarımında yukarıdan aşağıya yapılandırma yaklaşımını tanımlayabilmesiniz.*

8.1 İyi yazılım tasarımı yöntemlerinin önemi

Bölüm 6'da yazılımın tipik olarak, bir mikroişlemci sistemi tasarım/geliştirmeye ait toplam bütçenin %50'den fazlasını oluşturduğunu belirtmiştik. Yazılımın, toplam geliştirme maliyetinin % 80-90 kadarını oluşturması olağan dışı değildir. Bundan dolayı, yazılım geliştirme maliyetlerini aşağı düzeylerde tutmak, düşük-maliyetli sistem üretmek için çok etkili bir yoldur. Bilgisayar programlarını yazmak, test etmek, hatalarını ayıklamak ve belgelemek emek-yoğun işlemlerdir ve bu nedenle yazılım maliyetlerinin azaltılmasında programcının verimliliğini arttırmak en başta gelen unsurdur. Bu, programcının belirli bir zaman diliminde olabilecek maksimum sayıda yüksek-nitelikli program satırı üretebilmesini sağlayıcı yöntemlerin uyarlanması anlamına gelir. Bu bölümde önerilen yöntemler, bu amaç göz önünde tutularak önerilen yöntemlerdir.

Programcı verimliliğinin sistem geliştirme maliyetlerini düşük düzeyde tutmadaki rolünü vurguladıktan sonra şimdi de yazılım kalitesi sorusuna gelmiş bulunuyoruz. Düşük-kaliteli yazılım, sistemin gerekli özellikleri hangi düzeye kadar karşılayacağı konusu üzerinde temel bir etkiye sahiptir. Sistemin güvenilirliği de çoğu kez düşük-kaliteli yazılımdan etkilenebilir. Yeterli bilgi ve deneyime sahip olmayan bir programcı için yazılım kalitesi kavramını anlamak belki biraz zor olabilir. Bu nedenle Kısım 8.2'de iyi-kalitedeki yazılımın özelliklerini tartışacağız.

Programın kalitesi elbette programcının temel yeteneklerine bağlıdır. Ancak belki

de bundan da önemli olan unsur, benimsenen yaklaşım yöntemidir. Yeni başlay-
lar ve standart eğitimi olmayanlar sezgisel bir yaklaşımı benimseme eğiliminde-
ler, bu da her zaman zayıf programın üretilmesine yol açar. Formal program tasa-
yöntemleri bazen sıkıcı olsa da, yüksek-kalitede yazılım üretmenin tek yoludur.

8.2 Program özellikleri

Bölüm 8.1'de yazılım kalitesi kavramı tanıtıldı. Buradan, yüksek-kaliteli progr-
düşük kaliteli olandan ayıran belli özellikleri tanımlamanın mümkün olduğu so-
cu çıkarılabilir. Dolayısıyla, yüksek-kaliteli programların tasarım tekniklerini la-
latmaya geçmeden önce, bu istenen özellikleri tanımlamak daha doğru olacak.
Tahmin edilebileceği gibi yazılım tasarımcıları arasında belli bazı program öz-
liklerinin mi yoksa arzu edilen özelliklerin mi önde gelmesi gerektiği konusun-
görüş ayrılıkları bulunmaktadır. Bunun yanında, iyi bir programda en az, aşağı-
sıralanan ve temel özellikler olarak kabul edilen dört özelliğin bulunması gerek-
konusunda genel bir görüş birliği de vardır:

(a) Program doğru olmalıdır, bir başka deyişle gereken özellikleri karşılamalıd-
Belirlenen bilgi işlemlerini gerçekleştiremeyen bir program yararsız olacağında-
bu özelliğin, diğer özelliklerin hepsinden daha öncelikli olduğu açıktır. Fakat b-
programın doğru olduğundan kesin olarak emin olmak için, program performan-
nın kendisine göre kontrol edileceği kesin, ve farklı biçimde yorumlanamayaca-
bir özellikler bütününün hazır bulunması gereklidir.

(b) Program uyarlanabilir olmalı ve bakımı kolaylıkla yapılabilmelidir. Bunla-
ikisi de, temel yapısal değişikliklere ihtiyaç olmaksızın gerekli değişiklikler-
programa uygulanabilmesi anlamına gelmektedir. Program bakımının önemi Kısı-
8.3'te tartışılmıştır.

(c) Program yeterli düzeyde belgelenmiş olmalıdır. Kolaylıkla değiştirilebilen-
aynı zamanda da, iyi belgelenmemiş durumda olan bir program düşünmek çok gü-
olduğundan, bu özellik, b şıkkında belirtilen özellikle iç içedir. Belgelendirm-
yalnızca programcının kendisi için yapılmaz, hatta belgelendirme, öncelikle ve-
çok programcının yararlandığı bir olgu da değildir, programcının bunların bilinci-
de olması önemlidir. Program değişikliği görevinin, programın ilk yazarı olmayan-
bir programcıya verilmesi hiç karşılaşılmayan bir durum değildir.

(d) Program sağlam yapıda olmalıdır. Bu, programın, hata durumları veya opera-
törün basit yanlışlarından ortaya çıkan sonuçları tolere edebilmesi gerektiği anla-
mına gelmektedir. Kontrol paneli anahtarlarını tarayan ve denetim işlemi-

uygulayan bir programı ele alalım. Sağlam yapılı bir program, insan hatalarından kaynaklanan ve denetim anahtarlarının geçersiz ve yanlış çalıştırılmalarının söz konusu olduğu durumların üstesinden gelebilir. Program, bunun bir insan hatası olduğunu tesbit edilebilmeli ve denetim sisteminin güvenlik ve bütünlüğünü sürdürebilmelidir. Olabilen her durumda program, operatöre anlamlı hata mesajları vermelidir. Sistem konsolunda VDT varsa kontrol panelinin geçersiz şekilde çalıştırılması sonucunda anlaşılabilir bir hata mesajı görüntülenmelidir.

8.3 Programın testi ve bakımı

Bilgisayar programlarının bakıma gereksinim duyabilecekleri düşüncesi birçok okuyucuya şaşırtıcı gelecektir. Bu, mekanik bir sistemin, örneğin bir otomobilin yıprandığı gibi bir programın da yıpranacağı anlamına gelmez. Program bakımı genelde gerekli olmaktadır, çünkü ilk kez kullanıcıya verildiğinde hiç hata içermeyen yazılım paketi bulunsa bile bunların sayısı oldukça azdır. Büyük bir programın üstüste birçok kez çalıştırılması, gizli hataları ortaya çıkarır, bu hatalar ortaya çıktığında düzeltilmelidir. İşte bakım olarak adlandırılan işlem programın bu ilk hatalarının düzeltilmesi işlemidir. Büyük programların hemen hemen her zaman önceden bulunmamış olan hatalar içermeleri, bu programların yetersiz düzeyde programcılar tarafından yazıldıkları anlamına gelmez. Kuşkusuz kötü bir programcının yazdığı programlar, iyi bir programcının yazdıklarından daha fazla sayıda bulunmamış hata içerecektir. Ancak iyi bir programcı tarafından yazılmış olsa da, büyük ve karmaşık bir programın hata içermesi kaçınılmazdır. Bunun temel nedeni, pratikte büyük programların çoğunluğunu, tam olarak test edebilmenin imkansız olmasıdır. Amacı, bir miktar giriş verisini çıkış verisi üretmek üzere işlemek olan bir programı ele alırsak, birkaç basit hesaplama, test sorununun ne olduğunu gözler önüne serecektir.

İki 16-bitlik ikili sayıyı çarparak 32-bitlik bir sonuç verecek şekilde tasarlanmış bir programı ele alalım. Programın hiç hata içermediğinden kesinlikle emin olmak için, kullanıma verildiği zaman karşılaşabileceği, akla gelebilecek her giriş verisi altında test edilmesi gerekir. Böyle bir durumda geniş kapsamlı test, 16-bitlik ikili sayıların olabilecek tüm sayı çiftlerini içermelidir. Böyle bir test işlemi ise, toplam olarak dört bin iki yüz doksan beş milyon kere test edilmeyi gerektirecektir. Bunların her birinin bir saniyede yapılabildiği varsayılırsa, test işlemini tamamlamak için toplam 136 yıllık bir süre gerekecektir. Anlatılan program nispeten basit bir programdır. Orta düzeyde karmaşık olan bir programın bile geniş kapsamlı olarak test edilmesi, rahatça bir kaç milyon yıl sürebilir. Bu verdiğimiz örnek, yalnızca büyük programların yazılması sırasında gerçek anlamda farkına varılır duruma gelen programlama güçlüğüdür. Az bir pratik çalışmayla yeni bir programcı bile, on ya da yirmi

komutluk programları hatasız şekilde yazabilir. Küçük programlara geniş k
test işlemi uygulamak mümkün olabilir. Büyük programları geniş kapsamlı
test etmenin imkansızlığı, programın yazarı açısından iki sonuç verir: Ö
programlar, hata oluşumunu en aza indirdikleri kabul edilen yerleşmiş yö
kullanılarak tasarlanmalı ve uygulamaya konulmalıdır. İkinci olarak da, pr
lar, değişiklik ve hata düzeltme işlemleri, kolayca yapılabilecek şekilde dü
melidirler.

Bakım olarak adlandırılan bir başka uygulama ise bir programda, onu daha
duruma getirecek ya da ek olanaklar sağlayacak şekilde değişiklik yapılm
Böyle bir işlem özel bir sistem için bile gerekli olabilir. Bölüm 5'te anl
mikroişlemci-denetimli çamaşır makinesi gibi bir tüketim ürününü ele alalı
ürünlerin imalatçılarının, periyodik şekilde mallarını ek olanaklarla don
yenilemeleri yaygın bir uygulamadır. Çamaşır makinesi örneğinde bu yenile
yıkama programlarının makine özelliklerine eklenmesi şeklinde olabilir. Bi
böyle bir yenileme, ROM'da saklanmış denetim programının değiştirilme
gerçekleştirilecektir. Biraz özel bir durum olsa da bu örnek, yazılım tasarım
kolayca bakım yapılabilecek biçimde yapılmasının ne denli önemli oldu
S8.4 vurgulamaktadır.

8.4 Büyük ve küçük programlar

Daha pratik uygulamalar içeren programlar genellikle daha büyük programla
bu bölümdeki temel amacımız, program yazarken, küçük programlardan bu bü
programlara geçiş sırasında bulunması gereken yaklaşım farkını vurgulama
Özellikle bir komutlar dizisini kağıt üzerine yazma işleminin program kodla
adını alan bir işlem olduğunu belirtmek istiyoruz. Program yazmak, tek ba
program kodlamasından oluşmaz, hatta program kodlaması program yazmanı
önemli bölümü de değildir.

Mühendislik uygulamaları için mikroişlemci yazılımı öğrencisi, işe, çoğu k
makine kodunda basit programlar yazarak başlar. Programlama alıştırmaları iç
kullanılan mikroişlemcinin mimarisi ve komut takımı özelliklerinin iyi anlaşılm
S8.3 sına yol açtığından, bu iyi bir yaklaşım olabilir. Ancak bir dezavantajı vardır. B
durumda öğrenci, program tasarımı ve program kodlamasının aynı şey olduğu
düşünebilir. Çünkü kısa programlarda program, çoğunlukla, bir makine-kodu k
mutlar dizisinin kısa sürede yazılmasıyla oluşturulabilir. Biraz daha uzun progra
lar, sözgelimi döngü içerenler, bir akış diyagramının kullanılmasın
gerektirebilirler.

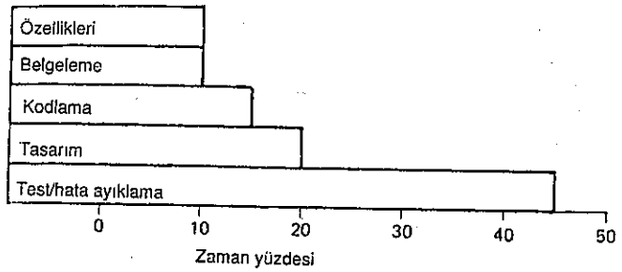
Kurallara uymayan bu yöntemle, kısa programlar üretme konusunda yüksek düzeyde beceri kazanmak mümkündür. Bazı programcılar, mikroişlemci komut takımı ve tipik programlama problemlerine o düzeyde aşina olurlar ki program tanımından yola çıkarak doğrudan makine-kodu kısaltmaları dizisini yazabilirler. Diğer bazı programcılar ise programın kapladığı bellek alanını azaltmak veya programın daha hızlı yürütümüne imkan tanımak için komutların standart-dışı ya da alışılmamış biçimde kullanıldığı programlama hilelerinden hoşlanır. Bunlar genellikle gereksiz ve istenmeyen uygulamalardır.

Öğrenme sırasında alıştırma olarak kullanılan programlarla gerçek mühendislik uygulamaları için tasarlananlar arasındaki temel fark, genellikle programın büyüklüğündedir. Bir mühendislik sistemini denetleyen mikrobilgisayarda çalıştırılan programın birkaç yüz komuttan daha az komut içermesi normal değildir. Böyle bir progam tipik olarak birkaç bin komut içerir. Bu tip büyük programları alışılagelmiş yöntemler kullanarak üretmeye çalışmak çok büyük sorunları davet etmek olur. Program yapısının tasarımı kesinlikle temel bir unsurdur. Örneğin yirmi komutluk programlar normal bir program planlaması ile başarılı şekilde yazılabilirler. Aynı durum bin komut içeren bir program için geçerli değildir. Yerleşmiş planlama uygulamasından küçük programlar bile yarar sağlar.

Herhangi bir program tasarımı çalışmasının sonuç amacı, söz konusu olan mikroişlemci tarafından yürütümü gerçekleştirilebilecek bir makine-kodu komutlar dizisinin üretilmesidir. Mikroişlemciler temel olarak, makine-kodu düzeyinde iş gördüklerinden kullanılan program dili ne olursa olsun bu geçerlidir. Ancak kullanılan dil ne olursa olsun program kodlaması, toplam program tasarımı işlemi içerisinde küçük ve nispeten ikincil önemde bir bölümdür. Program yapısının tasarımı ve programın belgelenmesi unsurlarının her ikisi de, çok daha önemli ve programcıya görevler yükleyen unsurlardır. Sonuç olarak, başarılı ve iyi tasarlanmış bir programın üretiminde planlama ve belgelemeye, programın son kodlama işleminden çok daha fazla zaman ayrılmaktadır. Genel olarak planlama ve belgelemeye ne kadar fazla zaman ayrılırsa, hata ayıklama ve bakım için o kadar daha az zaman gerekmektedir. Şekil 8.1'de, bir programın geliştirilmesi için harcanan toplam sürenin çeşitli işlemler arasında nasıl paylaştırıldığını, sık karşılaşılan bir duruma örnek alarak göstermektedir (8.1 nolu kaynağa bakınız).

8.5 Modüler programlar

Yazılacak mikroişlemci programlarının ne kadar büyük ve ne ölçüde karmaşık olması gerektiği konusunu ele alırken mühendisler, analog donanım tasarım durumunu dikkate almayı çoğunlukla yararlı bulmaktadır. Örneğin çok özel nitelikte bir



S8.3 *Şekil 8.1* Tipik bir program geliştirmesi sırasında çeşitli aktivitelerin aldığı zamanın yüzde olarak gösterim. Program bakımı dahil edilmemiştir.

yükselteç için bir tasarıma ihtiyaç duyulduğunu varsayalım. Devrenin sonuçta, transistör içereceği tahmin edilmektedir. Yaklaşımlardan birisi taslaktan başlam bütün devreyi tasarlamak, imal etmek ve anahtarı açarak özellikleri karşılama beklemektir. Yapılan taslağın, başta öngörülen özellikleri karşılamadığı görül güngde, tasarım mühendisi devrede hata ayıklama işlemine başlar, yani bir geliş kaydetmesi ümidiyle devrede değişiklik yapar. Alternatif bir yaklaşım özelliklerin incelenmesi ve yükseltme işlevinin kaç kattan oluşması gerekti karar vermektir. Bunun ardından da her katın kazanç ve diğer özellikleri belirle bilir. Daha sonra ilk kat tasarlanır ve yapılır. Bu kat test edilir ve gerekirse özellik karşılayana kadar üzerinde değişiklik yapılır. Aynı şekilde ikinci kat da tasarımla ve sınanır. Her iki evre de kendi özelliklerini karşıladığı zaman bunlar birleştiril ve iki-katlı yapı test edilir. Bu şekilde ilerleyerek, yükseltici, bütün sist özelliklere uygun biçimde işlev görünceye kadar kat-kat tasarlanır ve test edil Çok az da olsa deneyim sahibi mühendisler, bu yaklaşımla, daha önce anlatıl tasarım yöntemine göre daha kısa zamanda ve daha iyi bir tasarımın üretilebilm sinin mümkün olduğunu bilecektir.

Yukarıdaki yükselteç için anlatılan ikinci tasarım işleminde genel tasarım gör birkaç alt-göreve bölünmüştür. Bu altgörevler yükselteicin alt- bölümlerinin tasa rımını ve test edilmesini gerektirirler. Alt-bölümlerin önemli bir özelliği, bağıms olarak tasarlanabilmeleri ve test edilebilmeleridir. Tek tek yükselteç katlarını gerektirdiği nipeten basit olan tasarım sorunları, çok büyük ve karmaşık tek b devreye göre çok daha kolay ve güvenilir biçimde çözümlenebilir. Aynı modülerli ilkesi büyük bilgisayar programlarının yazımına da simülasyonu çok fazla zorla maksızın uygulanmalıdır. Modülerlik, büyük programlar için belki de en önemli tek tasarım tekniğidir.

Son program, birkaç alt-program ya da modüle bölünmüş olarak görünür. İdeal

her modül diğer program modüllerinden bütünüyle bağımsız bir alt-programdan oluşmalıdır. Bu, bir modülün, başka bir program modülünün doğru olması koşulunu gerektirmeden, bağımsız olarak yazılabilmesi, test edilebilmesi ve üzerinde hata ayıklama işlemi yapılabilmesi anlamına gelmektedir. Bu yolla birbiriyle ilişkili program hataları önlenebilir.

Program modüllerinin bütünüyle bağımsız olmaları arzu edilmesine karşın her zaman tam olarak sağlanamaz. Bu konudaki güçlüklerden birisi, program modüllerinin, diğer modüllerde üretilen veriyi işleyebilme özelliğindedir. Test programıyla, test edilen program arasında bir karşılıklı bağımlılık sorunu ortaya çıkarmasına karşın, modülün, özel olarak yazılmış bir test programı tarafından üretilen "kukla" veriler ile denenmesi bazen bu sorunun üstesinden gelinmesini sağlayabilir.

Modüler yaklaşımın programlamada sağladığı bazı avantajlar aşağıdadır:

1 Modüler yöntemler kullanılarak büyük bir programın tamamlanması için gereken süre, az ya da çok, programın son halinin içerdiği komut sayısı ile doğru orantılıdır. Bu, 4000 komut içeren bir programın yazımının kabaca, 2000 komutluk bir programın yazımı için gereken zamanın iki katı kadar zaman alacağı anlamına gelir. Eğer programı alt-programlara bölmek için herhangi bir girişimde bulunulmazsa, 4000 komutluk bir programın tamamlanması için harcanan süre, rahatlıkla, 2000 komutluk program için gereken zamanın iki katından daha fazla olabilir.
2 Program hatalarının çoğunluğu tek bir modül içinde ortaya çıkarılabilir. Yani hataların bulunması daha kolaydır. Hataların düzeltilmesi çoğu kez yalnızca hatanın ortaya çıktığı modülü etkiler. Bu ise program değişiklikleri ve bakımına büyük ölçüde yardımcı olur.
3 Modüller aynı program içinde birçok kez kullanılabilirler. Başka programlarda da kullanılabilirler.
4 Modüler programlar programlama ekiplerinin daha kolay şekilde görevlendirilmelerine olanak tanır. Her bir ekip üyesi farklı bir modülün geliştirilmesiyle görevlendirilebilir.

Modüler programlama yöntemlerinin, bir çok üstünlüğe sahip olmalarına ve büyük programların yazımı için tek anlamlı yolu temsil etmelerine karşın, beraberlerinde bazı güçlükler ve özel kurallar getirdiklerinin farkında olmak gerekir. Bunalardan bir kısmı aşağıdadır:

a Her bir modülün işlevi çok kesin biçimde belirlenmelidir. Bir modül, bir ya da

daha fazla sayıda modül tarafından üretilen veriyi kullanabilir. Kendi çı verileri de daha başka modüller tarafından giriş verisi olarak kullanılabilir. B nedenle giriş ve çıkış verilerinin biçimleri, özellikle bir program geliştirm ekibinin farklı üyeleri tarafından farklı modüllerin geliştirildiği durumlarda çe dikkatli şekilde tanımlanmalıdır. Birçok kimse bunu bir dezavantaj olarak de bir üstünlük olarak görmektedir.

b Modülleri test etmek için özel programların yazılması gerekebilir.

c Program modüllerinin dikkatli ve ayrıntılı biçimde belgelenmeleri gerekir. B kural yalnız modüler programlamaya özgü bir kural değildir. Hangi tasar yöntemi kullanılırsa kullanılsın bütün programlar gereken biçimde belgelenm lidir.

d Modüler programlar daha fazla bellek alanına gereksinim duyabilirler ve bu ların yürütümü modüler olmayan formdaki programlara göre daha yavaş olab lir. Bu çoğu kez büyük bir dezavantaj değildir ve genellikle modüler yapı

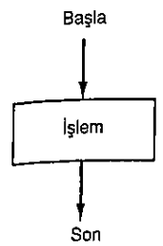
S8.5 üstünlükleri tarafından fazlasıyla telafi edilir.

8.6 Bölümlendirme

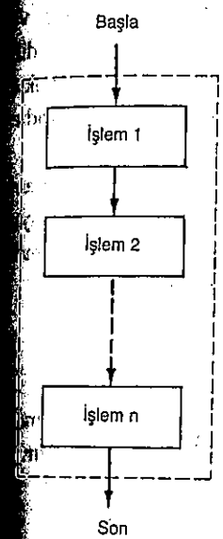
Buraya kadar anlatılanların, okuyucuyu, program tasarımında tek anlamlı yolu modüler yaklaşım olduğu konusunda yeterince aydınlattığını umarak şimdi pro ram modüllerinin nasıl tanımlandıkları konusunu ele alacağız. Programlama iş minin bu bölümünün amacı, gerekli özellikleri karşılayacak komple bir progra oluşturmak için birbirine bağlanabilen bir modül grubunun belirlenmesidir. Bu sembolik olarak tanımlarsak, Şekil 8.2'de gösterilen tek bir işlem uygulamasın yerine, Şekil 8.3'teki bağlantılı uygulamaları getirmekteyiz. Bağlanmış uygulam lardan her biri bir program modülü tarafından gerçekleştirilir.

Komple bir programı oluşturmak üzere bağlanmış program modüllerinin öne bir özelliği, her bir modülde yalnızca tek bir giriş ve yalnızca tek bir çıkış noktasın bulunmasının gereğidir. Bunun başlıca nedeni şudur: İki işlem arasındaki belirli yol ayrımına, tüm program boyunca yalnızca bir yolla ulaşılabildiğinin kesin olar bilinmesi, hata ayıklama yardımının sınırsız olmasını sağlamaktadır. Bunun alternatifi, modül giriş ve çıkış noktalarının sayısında bir sınırlama bulunmayan "fare boynu" şeklinde adlandırılan programlama stilidir. Sonuçta ortaya progra döngülerinin çoklu olarak içiçe kilitlenmesi durumu çıkar, bu yöntemi kullanm olan okuyucular içinden çıkılması zor olan bu durumu anımsayacaktır.

Tam bir programı oluşturan modüllerin tanımlanması işlemi "bölümleme" olar bilinir ve normalde aşama aşama gerçekleştirilir. Tekniği anlatmak için basit b



Şekil 8.2 Hesaplama işleminin sembolik tanımı



Şekil 8.3 Bağlanmış hesaplama

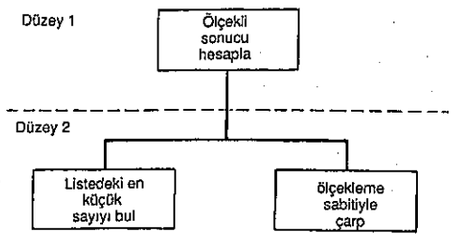
örnekle başlayacağız. Bellekte saklanmakta olan ve 20 sayı içeren bir liste içindeki en küçük sayıyı bulup bu sayıyı bir ölçekleme değişmezi ile çarpan bir program gerekli olsun. Programın simgesel tanımlaması Şekil 8.4'te gösterilmektedir. Programın kullandığı veri sayı listesi ve sıralama sabitidir. Ürettiği veri ise tabii ki listedeki en küçük sayı ile sıralama sabitinin çarpımıdır. Programın tek bir blok halindeki tanımlamasına 1. Düzey tanımlama adı verilir.

Şimdi programı, 8080, Z80 ya da 6800 gibi 8-bitlik mikroişlemcilerden biri için assembly dilinde yazmak istediğimizi düşünelim. Programlama konusunda herhangi bir deneyimi olan herkes bu programın kendiliğinden iyi tanımlanmış iki parçaya bölündüğünü farkedecektir. Birinci parça, listedeki en küçük sayıyı bulmak için listede bulunan her sayıyı inceler. İkinci parça ise bu sayıyı sıralama sabiti ile çarpar. Şekil 8.5 bunu sembolik olarak göstermektedir.

Şekil 8.5'te 3 program modülü tanımlamaktayız. 1. Düzey modülü, iki 2. düzey modülünden yararlanmaktadır. Bir başka deyişle, 1. Düzey modülünün iki 2. düzey modülüne bölündüğünü söyleyebiliriz. Bu bölümleme, 1. Düzey modülüne olan gereksinimi ortadan kaldırmaz fakat 1. Düzey'in yapısını önemli ölçüde değiştirir. Artık bu noktada, 1. Düzey modülü, iki 2. Düzeyli modülü birleştiren basit bir programdan oluşabilir.

Şekil 8.4 1. Düzey tanımı

Programlama görevini Şekil 8.4 ve 8.5'te gösterildiği gibi program modülleri tanımlayarak parçalara ayırdıktan sonra gerekirse modüllerin kendileri de daha başka modüllere bölümlenebilir. Bu işlem, modül tanımları, istenilen şekilde yönetilebilecek büyüklükteki alt-programlarla gerçekleştirilebilir basitliğe gelinceye kadar sürdürülür.



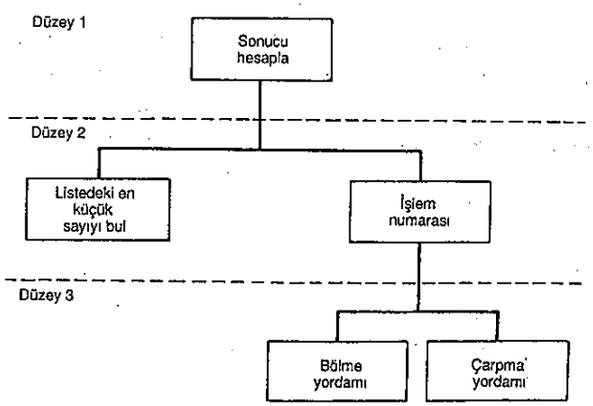
Şekil 8.5 2. Düzey'in bölümlenmesi

Okuyucu, bu noktada henüz yazılmamış olan bir programın yönetilebilecek büyü lükte olup olmadığının nasıl bilineceği sorusunu sorabilir. Programcının yönetil bilir olarak kabul edeceği program satır sayısı, kişiden kişiye değişir programcının deneyimine bağlıdır. Programcıların çoğunluğu 50 assembly komutundan daha az sayıda komut içeren modüllerin kabul edilebilir büyüklük olduğu görüşündedir. Şekil 8.5'te tanımlanan 2. Düzey modülleri buna uymakta

2. Düzey'in de ötesinde bir bölümleme örneği olarak yukarıdaki örnekte, progra tanımının değiştirildiğini düşünelim. Programın yine listedeki en küçük sayı "N"'yi bulmak için arama yapması gerekmektedir. Sonucun bulunması için şim aşağıdaki hesaplamanın yapılması söz konusudur:

$$\text{Sonuç} = (N + K)/(KN - 2)$$

Buradaki K ölçekleme sabitidir. Bu hesaplama, önceki örnekteki basit hesaplama dan çok daha karmaşıktır. Bu defa hesaplamada bölme işlemi de yer almaktadır.



Şekil 8.6 3. Düzey'in bölümlenmesi

Şekil 8.6, yeni özelliklerin daha ileri düzeyde bölümlemeye nasıl yol açtığını göstermektedir. Sonucun hesaplanması şimdi, daha-alt düzeydeki çarpma ve bölme modüllerini kullanan bir "işlem numarası" modülü tarafından yapılmaktadır. Bağlantılı modüller şeklinde bir program tanımlanırken program tasarımcısı, bölümlemenin yeterli düzeye inip inmediğini belirlemek amacıyla, çeşitli modüllerin uygulanabilmesi için gereken komut sayısını tahmin etmelidir. Burada deneyimin büyük ölçüde etkisi bulunmaktadır. Yanlış hesaplama yapılır ve belli bir modülün elverişli biçimde yönetilebilecek büyüklükten daha büyük bir boyuta ulaştığı ortaya çıkarsa, modül tanımlamasına geri dönerek bunu daha başka parçalara ayırmak kolay bir işlemdir.

8.7 Yukarıdan-aşağıya yöntemler

Bölümleme işlemi sonunda program tasarımcısı, her biri bir program modülünü belirten bir kutu topluluğu elde eder. Eğer tasarımcı doğru bir tasarım çalışması yapmışsa, her bir program modülünün gerçekleştirdiği hesaplama işlemi kesin biçimde tanımlanmış durumdadır. Dolayısıyla bu aşama, her bir program modülünün tasarım sorununun ele alınacağı aşamadır. Birinci adım olarak hangi program modülünün ilk önce tasarlanacağı konusunu ele alabiliriz. Göreve sistematik bir yolla yaklaşıldıysa iki seçenek söz konusu olacaktır. Bunlardan birincisi enalt düzeydeki modüllerin ilk önce tasarlanmasıdır ve buna, aşağıdan yukarıya yaklaşımı adı verilmektedir. Diğer yöntem ise en üstdüzeyli modülden başlar ve yukarıdan-aşağıya yaklaşım adını alır. Program tasarımcılarının çoğunluğu bu iki yöntem arasından, yukarıdan-aşağıya yaklaşım yöntemini tercih etmekte ve burada da, bu yöntem önerilmektedir.

Assembly dili programlarında program modülleri için en uygunu, altyordam biçimidir. Ancak program modülleri için hangi biçim kullanılırsa kullanılsın, verinin bir modülden diğerine geçmesinin gerekli olduğu birçok durum sözkonusu olacaktır. Assembly dilinde bu normalde veri-üreten program modülü tarafından yapılmaktadır, bu modül, veri-alan modül için bir altyordam çağrısı, dönüş ya da atlama komutunu uygulamaya koymadan önce, veriyi, mikroişlemcinin kaydedicilerinden birine veya özel bir bellek konumuna yerleştirmektedir.

Yukarıda anlatılan veri aktarımlarının nasıl gerçekleşeceğini kesin biçimde tanımlamak, program tasarımının önemli bir kısmını oluşturur. Hem verinin biçimi hem de ne aracılığıyla aktarılacağı, açık bir şekilde tanımlanmalı ve programın belgelendirilmesi sırasında belirtilmelidir. Böyle bir planlamaya gereksinim duyulması, bazen modüler programlama yaklaşımının dezavantajı olarak kabul edilmektedir. Daha gerçekçi olan diğer bir görüş ise bu kuralın istenebilir bir kural olduğu ve

hangi program tasarım yöntemi kullanılırsa kullanılsın uygulanması gerek şeklindedir.

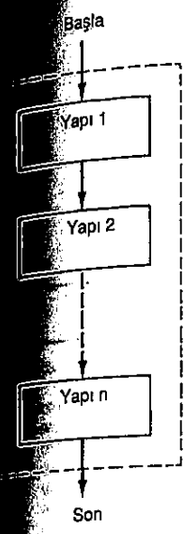
Yukarıdan-aşağıya yaklaşımının başlıca dezavantajlarından biri, daha alt- düz deki bir modülde hesaplanan sonuçlara bağımlı olan bir modülün tam ola sınanmasındaki güçlüktür. Şekil 8.6'da gösterilen bölümlemeyi ele alalım. "İşl numarası" yordamının yazımında, çarpma ve bölme altyordamlarına çağrı kom ları eklenmelidir. Bu durumda bu çarpma ve bölme altyordamlarının kendil yazılmadan önce programı nasıl test edebileceğiz? Bunun normal yolu, çarp veya bölme işlemin sonucu olarak bilinen bir sabiti üreten kukla altyordam yazmaktır. Doğru biçimde yapılırsa bu test yöntemi yüksek düzeyde güvenilirl
S8.8 yöntemdir.

8.8 Yapısal Programlar

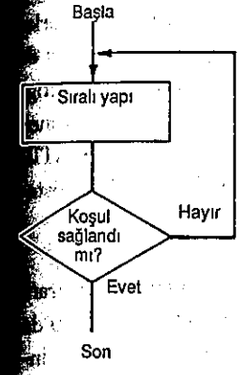
Yapısal yaklaşım

Bölümleme ile genel programlama görevini birkaç alt-göreve böldükten som program tasarımcısı, her bir program modülünün tasarımına eğilmelidir. Bu aş mada program kodlamasına, yani komut dizilerini yazmaya başlama konusun aceleci davranılmamalıdır. Program kodlaması başlamadan önce ilk olarak, pro ram modülünün yapısı tasarlanmalıdır. En yaygın biçimde kabul edilen progra yazma yaklaşımı, bütün programların standart blok ya da yapılar kullanıla tasarlanmasıdır. Aslında her tür programın, üç temel yapı birleşimi kullanıla yazılabileceği öne sürülmektedir. Bu bölümde biz, assembly dilinde ya da makin kodunda çalışılırken daha elverişli olduğundan dört yapı kullanacağız. Progra tasarımına yapısal yaklaşımın bir üstünlüğü, bu yaklaşımda, programın iç ya özelliklerinin daha kolay tanımlanabilmesidir. Bu özellik okuyucunun, progra akışını izleyebilmesine yardımcı olur, bu da hata ayıklama ve bakım sorunları büyük ölçüde kolaylaştırır. Ayrıca bu özellik, program üzerinde mantıklı ve sıral maya uygun biçimde ilerlemeye olanak verir.

Yapısal yöntemler ilk olarak, 8.3 nolu kaynakta gösterildiği şekilde ALGOL FORTRAN ve COBOL gibi yüksek-düzeyli dillerin kullanıldığı program tasarla rına yardımcı olmak amacıyla geliştirilmişlerdir. Ancak teknik genel bir teknikti ve bu bölümde assembly dili program geliştirme konusunu zihnimizde canlandıra bilmek için kullanılmıştır. Doğrudan makine kodunda programlama yaparken bil yapısal yöntemler avantajlı şekilde kullanılabilir. Bazı programlama dillerinin yapısal programlamaya diğerlerinden daha yatkın olduğu doğrudur. Örneğin PAS CAL dili, günümüzde yapısal yöntemler kullanmaya özellikle elverişli olan bir dil



Şekil 8.7 Sıralı yapı



Şekil 8.8 DO UNTIL akış diyagramı

olarak kabul edilmektedir. Ancak bu, PASCAL kullanmakla otomatik olarak verimli yapısal programlar yazılacağı anlamına gelmez. İyi nitelikte yapısal programlar "yapısal olmayan" diller kullanılarak yazılabilirler ve ne yazık ki bunun tersi de doğrudur.

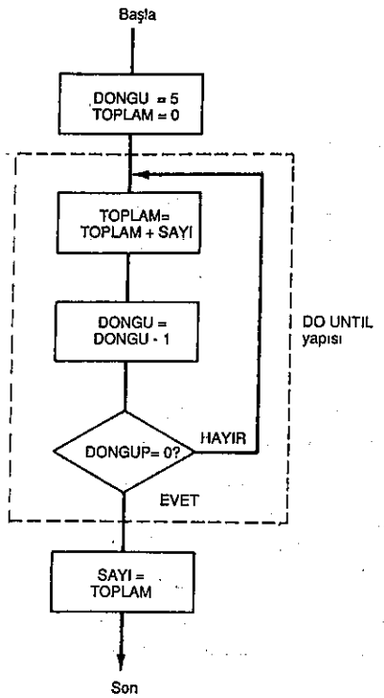
Sıralı yapı

Tanımlayacağımız en basit yapı budur. Her bir program modülünün bir işlem gerçekleştirdiği kabul edilir. Sıralı bir yapıda bir işlemin yerini daha basit bir işlemler dizisi alır. Bu, sembolik olarak Şekil 8.7'de gösterilmektedir. İşlemlerin her biri başka yapılar içerebilir.

DO UNTIL yapısı

Bu yapı okuyucunun daha önce karşılaştığı bir program özelliğini gösterir. İşlevi, bir komut dizisinin belli bir koşul sağlanıncaya kadar tekrar tekrar yürütülmesine neden olmaktır. Bu koşul çoğu kez bir döngü sayıcının sıfır değerini içermesidir. Şekil 8.8'de, bu yapı için bir akış diyagramını göstermektedir. Bu yapıda geriye atlama koşulunun dizi yapısının her yürütümünden sonra kontrol edildiğine dikkat edilmelidir. Bu, koşul sağlanmış olsa da, yapıya girişte dizinin her zaman, en az bir kez yürütüldüğü anlamına gelmektedir.

Assembly dilinde programlar yazarken birçok tasarımcı, programın yapısal özelliğini tanımlamak için formal bir yöntem saptamayı elverişli bulmaktadır. Günümüzde en yaygın biçimde kullanılan yaklaşım, yüksek-düzeyli dillerde yazılan bilgisayar programlarındakine oldukça benzer bir tanımlamadır. Buna sözde-dil ya da sözde-kod adı verilmektedir. Biz burada sözde-kod olarak adlandıracağız. Sözde-koddan assembly diline çeviri kolaydır ve elle kolayca yapılır. Sözde-kod programcı tarafından kendi amaçlarına uygun şekilde yaratılır. Nasıl tanımlanırsa tanımlansın sözde-kodun sürekli aynı biçimde kullanılması önemlidir. Bu metinde kullanılan sözde-kodun PASCAL ile birçok benzerliği bulunmaktadır. Sözde-kodun bir üstünlüğü, program tanımlamasının assembly listesinde açıklamalar biçiminde yer almasının olanak-



Şekil 8.10 DO UNTIL yapısını kullanan çarpma programının akış diyagramı

```
DO UNTIL KOŞUL SAĞLANDI
                ISLEM 1
                ISLEM 2
                  .
                  .
                  .
                ISLEM N
END DO
```

Şekil 8.9 DO UNTIL yapısının sözde-kod tanımlaması

```
DONGU = 5
TOPLAM = 0
DO UNTIL  DONGU = 0
          TOPLAM = TOPLAM + SAYI
          DONGU = DONGU - 1
END DO
SAYI = TOPLAM
```

Şekil 8.11 DO UNTIL yapısını kullanan çarpma programının sözde-kod tanımlaması

lı olmasıdır, akış diyagramları gibi diğer tanımlayıcı unsurlar ise ayrı şekilde hazırlanır. Hem akış diyagramlarının hem de sözde-dilin üstünlük ve dezavantajları bulunmaktadır. Programcı, üzerinde çalışılan program yapısı için en uygun görünen tanımlama yöntemi hangisi ise onu seçmelidir. DO UNTIL yapısı için genelleştirilmiş bir tanımlama Şekil 8.9'da gösterilmektedir.

Şekil 8.10'da, akış diyagramı kullanılan belirli bir program tanımlaması yer almaktadır. Aynı program, Şekil 8.10'da sözde-kod ile tanımlanmaktadır. Program tanımlamasının incelenmesiyle, programın, 0151 bellek konumunda saklı işaretsiz bir ikili sayının 5 ile çarpılması amacıyla tasarlandığı görülebilir. Çıkan sonuç, 0151 bellek konumunda saklanmaktadır. Tekrarlanan toplama algoritması kullanılmış ve sonucun 255'i (ondalık) aşmayacağı öngörülmüştür. Program tanımlamasının bu aşamada yalnızca mantıksal yapı ile ilgili olduğuna ve belirli herhangi bir mikroişlemcinin mimarisine bağımlı özellikler içermediğine dikkat edilmelidir. Her zaman olanaklı değilse de normalde bu aşamada, program tanımlamasını


```
8080/85 ASSEMBLER V01                                          PAGE  1
                                                               FUTUREDATA
        * ************************************************ *
        * BU PROGRAM X'0105'TE SAKLANAN BİR SAYIYI 5 İLE ÇARPAR, SONUCU X'0105' TE *
        * SAKLAR                                                                   *
        *                                  YAPI                                    *
        *                                  ****                                    *
        * DONGU = 5                                                                *
        * TOPLAM = 0                                                               *
        * DO UNTIL  DONGU = 0                                                      *
        *           TOPLAM = TOPLAM +SAYI                                          *
        *           DONGU = DONGU - 1                                              *
        * END DO                                                                   *
        * SAYI =TOPLAM                                                             *
        * ************************************************ *

0000                        RSEG        TEC612
                  *
0000  3A5101                LDA         X '0115'        A'DAKİ SAYIYI AL
0003  4F                    MOV         C, A            C= SAYI
0004  0605                  MVI         B, 5            B (DONGU) = 5
0006  3E00                  MVI         A, 0            A (TOPLAM) = 0
0008  81          D01       ADD         C               TOPLAM =TOPLAM + SAYI
0009  05                    DCR         B               DONGU = DONGU - 1
000A  C20800                JNZ         D01             DONGU = SIFIR İSE
                  *                                     GERİYE ATLA
000D  325101                STA         X '0151'        SAYI = TOPLAM
                  *
                  *
0010                        END
```

Şekil 8.12 8080 DO UNTIL program yapısı

```
6800/02 ASSEMBLER V01                                          PAGE  1
                                                               FUTUREDATA

0000                        RSEG        TEC613
                  *
0000  4F                    CLRA                        A (TOPLAM) = 0
0001  C605                  LDAB        #5              B (DONGU) = 5
0003  BB0151      D01       ADDA        X '0151'        TOPLAM = TOPLAM + SAYI
0006  5A                    DECB                        DONGU = DONGU - 1
0007  26FA                  BNE         D01
                  *
0009  B70151                STAA        X '0151'        SAYI = TOPLAM
                  *
                  *
000C                        END
```

Şekil 8.13 M6800 DO UNTIL program yapısı

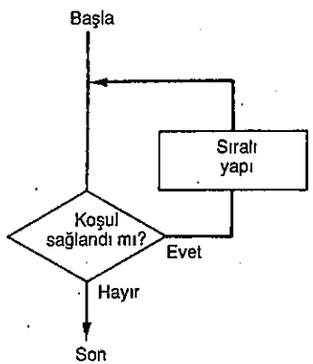
assembly dili veya makine-kodu programlar için bile işlemciden bağımsız olarak yapabilmek istenir.

Şekil 8.11'deki yapısal bir programın assembly listesi Şekil 8.12'de gösterilmektedir. Liste, yapıyı tanımlayan sözde-yüksek düzeyli dil formunda açıklamalar içer-

```
----- Z80 ASSEMBLER V01 ------------------------------------ SAYFA 1
                                                              FUTUREDATA ---------

0000                          RSEG      TEC614
                  *
0000  3A5101                  LD        A,(X'0151')         A'DAKİ SAYIYI AL
0003  4F                      LD        C,A                 C= SAYI
0004  0605                    LD        B,5                 B (DONGU) = 5
0006  3E00                    LD        A,0                 A (TOPLAM) = 0
0008  81            DO1       ADD       A,C                 TOPLAM = TOPLAM + SAYI
0009  10FD                    DJNZ      DO1                 DONGU = DONGU - 1
                  *                                         SIFIR DEĞİLSE GERİYE ATLA
000B  325101                  LD        (X'0151'),A         SAYI = TOPLAM
                  *
                  *
000E                          END
```

Şekil 8.14 Z80 DO UNTIL program yapısı



Şekil 8.15 DO WHILE akış diyagramı

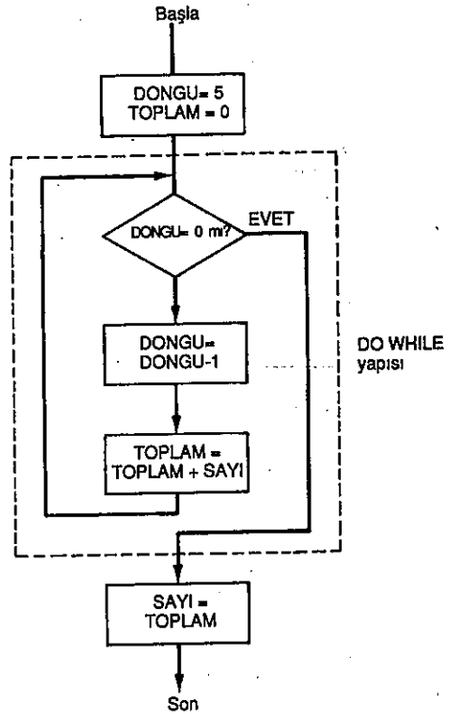
```
DO WHILE KOŞUL SAĞLANMADI
                    ISLEM 1
                    ISLEM 2
                    .
                    .
                    .
                    ISLEM N
END DO
```

Şekil 8.16 DO WHILE yapısının sözde-kod tanımı

```
DONGU = 5
TOPLAM = 0
DO WHILE DONGU ≠ 0
        DONGU = DONGU + 1
        TOPLAM = TOPLAM + SAYI
END DO
SAYI = TOPLAM
```

Şekil 8.17 DO WHILE yapısı kullanan çarpma programının sözde-kod tanımı

mektedir. Bu tip açıklamalar assembly listelemesinin önemli bir özelliğidir ve daima listede bulunmalıdır. Şekil 8.12'de gösterilen program 8080 mikroişlemcisi içindir. Şekil 8.13 ve 8.14, yine Şekil 8.11'deki DO UNTIL yapısı programlarını göstermektedir, fakat bu kez programlar, sırasıyla Z80 ve 6800 mikroişlemcileri içindir. Şekil 8.12'de yer alan tanımlayıcı



Şekil 8. 18 DO WHILE yapısını kullanan çarpma programının akış diyagramı

```
----- Z80 ASSEMBLER V01 ------------------------------------ PAGE 1
                                                              FUTUREDATA ---------

0000                          RSEG      TEC620
                  *
0000  3A5101                  LD        A,(X'0151')         A'DAKİ SAYIYI AL
0003  4F                      LD        C,A                 C= SAYI
0004  0605                    LD        B,5                 B (DONGU) = 5
0006  1600                    LD        D,0                 D (TOPLAM) = 0
0008  78            DO1       LD        A,B                 A'DAKİ DONGU'YÜ AL
0009  FE00                    CP        0                   DONGU = 0 MI?
000B  CA1600                  JP        Z,DO2
000E  05                      DEC       B
000F  7A                      LD        A,D                 A'DAKİ TOPLAMI AL
0010  81                      ADD       A,C                 TOPLAM = TOPLAM + SAYI
0011  57                      LD        D,A                 D = TOPLAM
0012  C30800                  JP        DO1
                  *
0015  7A            DO2       LD        A,D                 A'DAKİ TOPLAMI AL
0016  325101                  LD        (X'0151'),A         SAYI = TOPLAM
                  *
                  *
0019                          END
```

Şekil 8.20 Z80 DO WHILE program yapısı

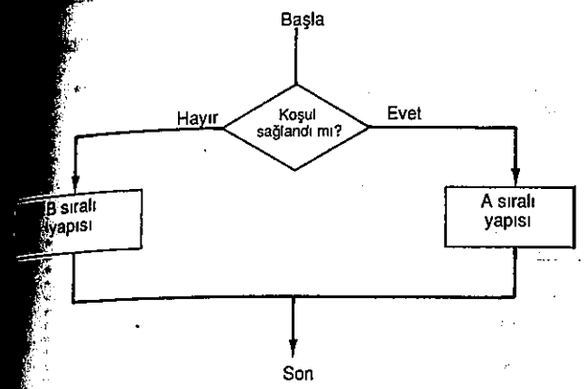
```
----- 8080/85 ASSEMBLER V01 ------------------------------------ PAGE 1
                                                                  FUTUREDATA -----
                 * ****************************************
                 *  BU PROGRAM X'0105'TE SAKLANAN BİR SAYIYI 5 İLE ÇARPAR, SONUCU YENİDEN
                 *  X'0105'E SAKLAR
                 *
                 *                                YAPI
                 *                                ****
                 *  DONGU = 5
                 *  TOPLAM = 0
                 *  DO WHILE DONGU NOT EQUAL 0
                 *          DONGU = DONGU - 1
                 *          TOPLAM = TOPLAM + SAYI
                 *  END DO
                 *  SAYI = TOPLAM
                 * ****************************************

0000                                RSEG        TEC619
                 *
0000   3A5101                      LDA         X '0151'
0003   4F                          MOV         C, A                    C = SAYI
0004   0605                        MVI         B, 5                    B (DONGU) = 5
0006   1600                        MVI         D, 0                    D (TOPLAM) = 0
0008   78          DO1             MOV         A, B                    A'DAKİ DONGU'YÜ AL
0009   FE00                        CPI         0                       DONGU = 0 MI?
000B   CA1500                      JZ          DO2                     EVETSE ATLA
000E   05                          DCR         B                       DONGU = DONGU - 1
000F   7A                          MOV         A, D                    A'DAKİ TOPLAMI AL
0010   81                          ADD         C                       TOPLAM = TOPLAM + SAYI
0011   57                          MOV         D, A                    D = TOPLAM
0012   F20800                      JP          DO1
                 *
0015   7A          DO2             MOV         A, D                    A'DAKİ TOPLAMI AL
0016   325101      *               STA         X '0151'                SAYI = TOPLAM
                   *
0019                               END
```

Şekil 8.19 8080 DO WHILE yapısı

```
----- 6800/02 ASSEMBLER V01 ------------------------------------ PAGE 1
                                                                  FUTUREDATA --------
0000                                RSEG        TEC621
                 *
0000   C605                        LDAB        #5
0002   4F                          CLRA                                B (DONGU) = 5
0003   5D          DO1             TST         B                       A (TOPLAM) = 0
0004   2706                        BEQ         DO2                     DONGU = 0 MI?
0006   5A                          DECB                                EVETSE DALLAN
0007   BB0151                      ADDA        X '0151'                DONGU = DONGU - 1
000A   20F7                        BRA         DO1                     TOPLAM = TOPLAM + SAYI
                 *
000C   B70151      DO2             STAA        X '0151'                SAYI = TOPLAM
                   *
                   *
000F                               END
```

Şekil 8.21 M6800 DO WHILE program yapısı



Şekil 8.22 IF THAN ELSE yapısı akış diyagramı

```
IF CONDITION TRUE THEN
                    A SIRASI
        ELSE
                    B SIRASI
END IF
```

Şekil 8.23 IF THEN ELSE Sözde-kodu

açıklamalar, Şekil 8.13 ve 8.14'ün listelerinde de yer almalıdır, fakat burada yer darlığı nedeniyle tekrarlanmamıştır.

DO WHILE yapısı

Bir sıra bloğunun üstüste yürütümü gerektiğinde bu yapı da kullanılmaktadır. DO UNTIL yapısından, koşul kontrolünün Şekil 8.15'teki akış diyagramında gösterildiği gibi, yapının baş kısmında gerçekleştirilmesiyle farklılık gösterir. Yapının sözde-kod biçimi, Şekil 8.16'da verilmiştir. Şekil 8.15'teki koşul kontrol yerinin bir sonucu, girişte koşulun sağlanmış olması durumunda sıra yapısının hiç bir zaman yürütülmemesidir, bu durum, sıra yapısının her zaman en az bir kez yürütüldüğü DO UNTIL'de farklıdır.

Şekil 8.17'de, bir DO WHILE yapısı ile, 5 ile çarpmak amacıyla üstüste toplama programının nasıl uygulandığını gösterilmiştir. Programın akış diyagramının tanımlaması, Şekil 8.18'de verilmektedir. Programın assembly listesinin 8080, Z80 ve M6800 versiyonları Şekil 8.19, 8.20 ve 8.21'de yer almaktadır. Okuyucu, bu üç

```
                                                                    PAGE  1
..... 8080/85 ASSEMBLER V01 ..............................  FUTUREDATA .........

                 * ****************************************** *
                 * BU PROGRAM X'1000'DA SAKLI İŞARETSİZ İKİLİ SAYIYI TEST EDER. EĞER 127'DEN *
                 * KÜÇÜKSE 'X100B' KONUMUNDAKİ SAYI 1 ARTIRILIR. 127'DEN KÜÇÜK DEĞİLSE X '100B' *
                 * KONUMUNDAKİ SAYI 1 ARTTIRILIR.                                              *
                 *                                YAPI                                         *
                 *                                ****                                         *
                 * IF N1 LESS THAN 127 THEN                                                    *
                 *                    N2 = N 2 + 1                                             *
                 *            ELSE                                                             *
                 *                    N3 = N3 + 1                                              *
                 * END IF                                                                      *
                 * NOTE                                                                        *
                 * *****                                                                       *
                 *      N1 = (X '1000')                                                        *
                 *      N2 = (X '100A')                                                        *
                 *      N3 = (X '100B')                                                        *
                 * ****************************************** *

0000             *                  RSEG          TEC624

0000  3A0A10                        LDA           X '100A'          A'DAKİ N2'Yİ AL
0003  4F                            MOV           C, A              C = N2
0004  3A0B10                        LDA           X '100B'          A'DAKİ N3'Ü AL
0007  57                            MOV           D, A              D = N3
                 * IF-THEN-ELSE YAPISI BURADAN İTİBAREN BAŞLAR
0008  3A0010                        LDA           X '1000'          A'DAKİ N1'İ AL
000B  FE7F                          CPI           127               127 İLE KARŞILAŞTIR
000D  D21400                        JNC           IF1               KÜÇÜK DEĞİLSE ATLA
                 * THEN
0010  0C                            INR           C                 N2 = N2 + 1
0010  F21500                        JP            IF2
                 *ELSE
0014  14         IF1                INR           D                 N3 = N3 + 1
                 *
0015  79         IF2                MOV           A, C              A'DAKİ N2'Yİ AL
0016  320A10                        STA           X '100A'          N2'Yİ GERİYE AL
0019  7A                            MOV           A, D              A'DAKİ N3'Ü AL
001A  320B10                        STA           X '100B'          N3'Ü GERİYE AL
                 *
001D                                END
```

Şekil 8.24 8080 IF THEN ELSE program yapısı

assembly dili programı çalışırken, DO WHILE yapısının 8080 ve Z80'deki uygulamalarının nispeten daha elverişsiz olduğunu fark edebilir. Bu, birincil olarak, bu mikroişlemcilerin kaydedici-yönelimli mimarilerinin bir sonucudur.


```
                                                                    PAGE  1
..... Z80 ASSEMBLER V01 ..................................  FUTUREDATA .........

                 * Z80 IF-THEN-ELSE
                                    RSEG          TEC625
                 *
                                    LD            A, (X'100A')      A'DAKİ N2'Yİ AL
3A0A10                              LD            C, A              C = N2
4F                                  LD            A, (X'100B')      A'DAKİ N3'Ü AL
3A0B10                              LD            D, A              D = N3
57
                 * IF-THEN-ELSE YAPISININ BAŞLANGICI
                                    LD            A, (X'1000')      A'DAKİ N1'İ AL
3A0010                              CP            127               127 İLE KARŞILAŞTIR
FE7F                                JP            NC, ELSE          KÜÇÜK DEĞİLSE ATLA
D21400
                 * THEN
                                    INC           C                 N2 = N2 + 1
0C                                  JP            END IF            END IF'E ATLA
C31500
                 * ELSE
14                  ELSE            INC           D                 N3 = N3 + 1
                 * END IF
                    END IF          LD            A, C              A'DAKİ N2'Yİ AL
79                                  LD            (X '100A'), A     N2'Yİ GERİYE AL
320A10                              LD            A, D              A'DAKİ N3'Ü AL
7A                                  LD            (X'100B'), A      N3'Ü GERİYE AL
320B10
                 *
                                    END
```

Şekil 8.25 Z80 IF THEN ELSE program yapısı

```
                                                                    PAGE  1
..... 8080/85 ASSEMBLER V01 ..............................  FUTUREDATA .........

                 * ****************************************** *
                 * BU PROGRAM X'1000' ADRESİNDE SAKLI OLAN İŞARETSİZ İKİLİ N1 SAYISINI KONTROL *
                 * EDER, 127'DEN KÜÇÜKSE X'100A' ADRESİNDE SAKLI OLAN N2 SAYISI 1 ARTIRILIR. DİĞER *
                 * DURUMLARDA HERHANGİ BİR İŞLEM YAPILMAZ.                                     *
                 *                                YAPI                                         *
                 *                                ****                                         *
                 * IF N1 LESS THAN 127 THEN                                                    *
                 *                    N2 = N2 + 1                                              *
                 *            ELSE                                                             *
                 *                    CONTINUE                                                 *
                 * END IF                                                                      *
                 * ****************************************** *

0000                                RSEG          TEC627
                 *
0000  3A0010                        LDA           X '0000'          A'DAKİ N1'İ AL
0003  FE7F                          CPI           127               127 İLE KARŞILAŞTIR
0005  D20F00                        JNC           IF1               KÜÇÜK DEĞİLSE ATLA
0008  3A0A10                        LDA           X '100A'          A'DAKİ N2'Yİ AL
000B  3C                            INR           A                 N2 = N2 + 1
000C  320A10                        STA           X '100A'          N2'Yİ GERİYE AL
000F  00         IF1                NOP                             BİR SONRAKİ KOMUT
                 *
0010                                END
```

Şekil 8.27 IF THEN program yapısı

```
----- 6800/02 ASSEMBLER V01 ----------------------------------  PAGE 1
                                                                FUTUREDATA --------
                  * M6800-THEN-ELSE
0000                        RSEG        TEC626
                  *
                  * IF-THEN-ELSE YAPISININ BAŞLANGICI
0000  867F                  LDAA        #127          N1'i 127 İLE KARŞILAŞTIR
0002  B11000                CMPA        X '1000'      KÜÇÜK DEĞİLSE DALLAN
0005  2506                  BCS         ELSE
                  * THEN
0007  7C100A                INC         X '100A'      N2 = N2 + 1
000A  2003                  BRA         END IF
                  * ELSE
000C  7C100B      ELSE      INC         X '100B'      N3 = N3 + 1
                  * END IF
000F  01          END IF    NOP                       BİR SONRAKİ KOMUT
                  *
0010                        END
```

Şekil 8.26 M6800 IF-THEN-ELSE program yapısı

IF THEN ELSE yapısı

Bu yapı da yine, program yazma yaklaşımları ne olursa olsun, okuyucuların çoğunluğunun tanışık olduğu bir programlama tekniğine örnektir. Daha önceki iki yapıda olduğu gibi burada da koşul testi yapılmaktadır. IF THEN ELSE yapısında iki alternatif davranış bulunmaktadır. Şekil 8.22 ve 8.23'de, yapının akış diyagramı ve sözde-kod ile nasıl tanımlandıklarını gösterilmektedir. 8080, Z80 ve M6800 için IF THEN ELSE yapısını içeren program örnekleri sırasıyla Şekil 8.24, 8.25 ve 8.26'da verilmiştir. Yer darlığı nedeniyle, tam tanımlayıcı biçimde yapısal bilgi yalnızca, 8080 listesinde sunulmuştur. Şekil 8.25 ve 8.26'da verilen yapılar birbirinin aynısıdır.

Belli bir işlemin yalnızca bir koşulun sağlanması durumunda gerçekleştirilmesi istendiğinde, IF THEN ELSE yapısının özel bir durumu ortaya çıkar. Koşul sağlanmamışsa o zaman herhangi bir işlem yapılmaz. Bu türde bir yapıyı içeren bir program, Şekil 8.27'de gösterilmiştir. Sözde- kod tanımında CONTINUE deyimi, programın herhangi bir işlem yapılmaksızın sürdürülmesi anlamına gelir.

8.9 İçiçe yapılar

Buraya kadar program yapılarını tek başlarına anlattık. Komple bir programda normalde, her biri kendine özgü yapısal özelliklere sahip olan birçok modül bulunur. Bu durumda IF THEN ELSE gibi bir yapı, bir program bloğu olarak değerlendirilebilir, bu yapı daha sonra başka bir yapının bir parçasını oluşturabilir.


```
8080/85 ASSEMBLER V01 ----------------------------------  PAGE 1
                                                          FUTUREDATA --------
*  ****************************************************  *
*  BU PROGRAM X'1000' ADRESİNDEN BAŞLAYAN 10 SAYILIK BİR İŞARETSİZ İKİLİ SAYI  *
*  LİSTESİNİ TARAR. 127'DEN KÜÇÜK GİRİŞLER SAYILIR VE SONUÇ X'100A'DA SAKLANIR.  *
*                                                                                *
*                                YAPI                                            *
*                                ****                                            *
*  ISARETCI = X'1000'                                                            *
*  DONGU = 10                                                                    *
*  SONUC = 0                                                                     *
*  DO UNTIL DONGU = 0                                                            *
*      GET POINTED NUMBER                                                        *
*      IF SAYI LESS THAN 127 THEN                                                *
*                  SONUC = SONUC + 1                                             *
*                  ELSE                                                          *
*                  CONTINUE                                                      *
*      END IF                                                                    *
*      ISARETCI = ISARETCI + 1                                                   *
*      DONGU = DONGU - 1                                                         *
*  END DO                                                                        *
*  ****************************************************  *

0000                        RSEG        TEC628
0000  210010                LXI         H, X'1000'    ISARETCI = HL = X'1000'
0003  060A                  MVI         B, 10         B (DONGU) = 10
0005  0E00                  MVI         C, 0          C = SONUC
                  * START OF DO LOOP
0007  7E          DO        MOV         A, M          A'DAKİ İŞARETLENEN SAYIYI AL
                  * IF-THEN-ELSE YAPISININ BASLANGICI
0008  FE7F                  CPI         127           127 İLE KARŞILAŞTIR
000A  D20E00                JNC         ENDIF         DAHA KÜÇÜK DEĞİLSE ATLA
                  * THEN
000D  0C                    INR         C             SONUC = SONUC + 1
                  * ELSE CONTINUE
000E  23          ENDIF     INX         H             ISARETCI = ISARETCI + 1
000F  05                    DCR         B             DONGU = DONGU - 1
0010  C20700                JNZ         DO
                  * END DO
0013  79                    MOV         A, C          A'DAKİ SONUCU AL
0014  320A10                STA         X'1000'       SONUCU GERİYE AL
                  *
0017                        END
```

Şekil 8.28 DO UNTIL döngüsü içinde IF THEN ELSE yapısı

Şekil 8.28'de, bir IF THEN ELSE yapısının bir DO UNTIL döngüsü içine gömülü olarak bulunduğu bu tip bir programın assembly listesini gösterilmiştir. Şekil 8.27'de gösterilen programın kendisi bir blok gibi kabul edilebilir, bu da daha sonra başka bir yapının bir parçasını oluşturabilir.

Kaynaklar ve bibliyografi

8.1 M. V. Zelkowitz, "Perspectives on software engineering", *ACM Computing Surveys*, p198 (June 1978)

8.2 A. D. Hearn, "Top-down modular programming", *Byte*, Vol.3 (7), 1978)

8.3 G. L. Richardson, C. W. Butler ve J. D. Tomlinson, *A Primer on Structured Program Design*, Petrocelli Books (1980).

Sorular

8.1 Bir mikroişlemci sistemin toplam tasarım/geliştirme bütçesi donanım yazılım geliştirme maliyetleri biçiminde bölümlendirilebilir. Normalde hangi bileşen daha büyük olacaktır?

8.2 İyi tasarlanmış bir programda bulmayı beklediğiniz dört özelliği anlatınız.

8.3 Programlama ve program kodlama arasındaki farkı açıklayınız.

8.4 Program bakımı ile ne anlatılmak istendiğini açıklayınız.

8.5 Program tasarımına modüler yaklaşımın dört avantajını ve dört dezavantajını söyleyiniz.

8.6 Programlama verimliliği genel olarak, verilen bir hesaplama görevini gerçekleştirmek amacıyla hata ayıklaması yapılmış bir program üretmekte harcanan zaman olarak tanımlanmaktadır. Bu tanımdan hareketle assembly dilinin mi yoksa BASIC veya PASCAL gibi yüksek-düzeyli bir dilin mi daha verimli olduğunu söyleyiniz.

8.7 Hangi tip programda yüksek-düzeyli bir dil yerine assembly dilinin tercih edilmesi gerekir?

8.8 Program yazımında yukarıdan-aşağıya yaklaşım ile ne anlatılmak istenmektedir?

8.9 Bir mikroişlemci, klavye aracılığıyla verilen koordinatlarla belirlenen yerlere delikler açan bir matkap makinesini denetlemektedir. Programcı, delik denetim programını aşağıda *(a)*, *(b)*, *(c)* ve *(d)* biçiminde gösterilen dört alt-bölüme ayırmaya karar vermiştir:

(a) Klavyeden veri girişi
(b) Matkap ucunun yerinin denetimi
(c) Delme işleminin uygulanması
(d) Sistemin başlatılması ve kullanıma hazırlanması

Ayrıca matkap ucunun yerini denetleyen program modülünün aşağıda gösterilen iki alt-bölüme ayrılması da kararlaştırılmıştır:

(e) Matkap ucunun X ekseni boyunca denetimi
(f) Matkap ucunun Y ekseni boyunca denetimi

Programı hiyerarşik ayrışma diyagramını kullanarak tanımlayınız.

8.10 Bir akış diyagramı ve sözde-kod yardımıyla, tekrarlanan toplama yöntemini kullanarak, 500 bellek konumu içeriğini 7 ile çarpan bir program tanımlayınız. Hesaplamanın sonucu 600 adresinde saklanmalıdır.

8.11 500 adresinden başlayan elli sayılık bir listedeki her sayıya 5 ekleyen bir program tasarımı yapınız. Program bir DO UNTIL döngüsü kullanmalıdır. Programın akış diyagramı ve sözde-kodu tanımı verilmelidir.

8.12 1000-10FF bellek alanında saklanan sayıların kaç tanesinin pozitif ve kaç tanesinin negatif sayı olduğunu belirleyen bir program gereklidir. DO UNTIL döngüsü içinde gömülü bir IF THEN ELSE yapısı kullanan uygun bir program tasarımı yapınız. Programı a) bir akış diyagramı, b) sözde-kod kullanarak tanımlayınız.

8.13 Soru 8.10'daki tekrarlanan toplama algoritmasını kullanarak 1000 adresinden başlayan elli sayılık bir listedeki her bir girişi 5 ile çarpan bir program tasarlayınız. Program yapısı bir DO UNTIL döngüsü içinde yer alan başka bir DO UNTIL döngüsü biçiminde olmalıdır. Sözde-kod kullanarak programı tanımlayınız.

8.14 1000 adresinde işaretli bir ikili sayı saklıdır. Sayının işaretini test eden bir program tasarlayınız. Sayı pozitif ise bir sembolik adres olan POS'un içeriği belli bir sayı kadar arttırılmalı, sayı pozitif değilse NEG'in içeriği arttırılmalıdır. Program IF THEN ELSE yapısı formunda *(a)* bir akış diyagramı *(b)* sözde-kod kullanılarak tanımlanmalıdır. [*Not:* Negatif sayılar en büyük değerlikli (işaret) bite sahiptir =1.]

Bölüm 9 Donanım Geliştirme

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde;*

1 *Mikroişlemci sistemleri geliştirmesinde,*
   (a) *Prototip donanım ve firmware yapımı ve testi*
   (b) *Sistem bileşenlerinin, (a) aşamasına yardım amacıyla simülasyon emülasyonu*
   (c) *Donanım ve/veya firmware'in (a) ve (b) aşamalarının sonucuna değiştirilmesi olarak belirtilen ileri aşamaları analize edebilmeli,*

2 *Test ve hata ayıklama periyodunun ilk geliştirme çalışmalarına oranının, görevinin karmaşıklığına paralel olarak önemli ölçüde arttığını kavrayabil*

3 *Mikroişlemci geliştirme sistemlerinin, donanım ve yazılım tasarım ve tes yardımcı unsurlar olarak kullanımı konusunda daha geniş bir anlayış kaza olmalısınız.*

9.1 Donanım geliştirme

Bu bölümde, mikroişlemci sistem donanım geliştirmenin ileri aşamalarından b larını ele almaktayız. Sistem planlama işleminin doğru şekilde gerçekleştirildi ve ana kararların Bölüm 6'da anlatılan şekilde alındığını varsaymaktayız. noktada görev, ilk devre diyagramında çalışan bir donanım prototipi üretmek Prototipin üretilmesi ve test edilmesinde devre-içi emülasyon olanakları bulun bir mikroişlemci geliştirme sisteminin nasıl yardım sağladığını anlatacağız.

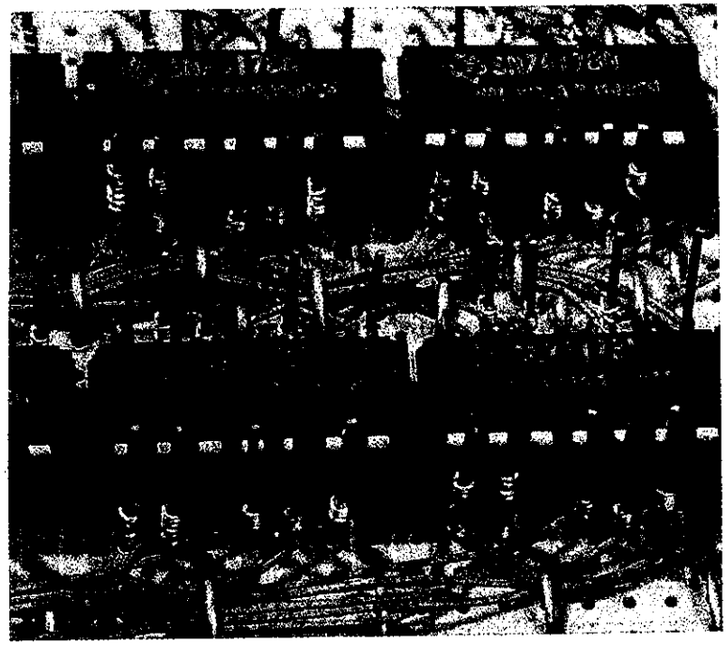
9.2 Prototip donanımın yapım ve testi

Çok sayıda üretilen sistemler normalde entegre devre, kondansatör ve direnç gib çeşitli bileşenlerin özel olarak tasarlanmış baskılı devre kartlarına monte edilme siyle oluşturulurlar. Ancak baskılı devre tasarımının aşama kaydedebilmesi iç önce, devre tasarımının tamamlanması ve tam olarak test edilmesi gerekir. Don nım prototipinin amacı budur.

Prototip devre kartı yapım yöntemleri içinde en çok bilineni, tel sargı yöntemidir Bu yöntem ayrıca, çok küçük sayılarda üretilen sistemlerin yapımlarında da yaygın

olarak kullanılmaktadır. Bu yöntemde entegre devreler uygun prototip kartlardaki soketlere takılır. Bu kartlar standart çift sıralı entegre devre paketinin bacaklarının yerleşim düzeni ile aynı şekilde 0.1- inçlik matrislerde delikler açılarak delinir. Entegre devre soketlerinde tel-sargılı bağlantılara uygun bacaklar bulunmaktadır. Bu nedenle ara bağlantılar, entegre devre soketlerinin bacaklarına bağlı teller kullanılarak yapılmaktadır. Tel sargı tekniği, gereken yerlerde devre değişikliği yapılmasına olanak veren, yüksek düzeyde güvenilir bağlantılar yapmak için elverişli bir yöntemdir. Şekil 9.1'de, tipik bir tel-sargılı prototip kartını gösterilmektedir.

Entegre devrelerin prototip kart üzerindeki yerleşim düzenlerini, ara bağlantıları olabilecek en kısa ölçüde tutmak amacıyla, dikkatli bir şekilde düşünerek gerçekleştirmek gerekmektedir. Arzu edilen fiziksel düzene sahip devrelerin elde edilmesinde, deneyimin büyük rolü bulunmaktadır. Prototip işlevi gören kartların çoğunluğunda baskılı devre üzerinde d.c. güç kaynağı çıkış kabloları bulunur, entegre devreler o şekilde yerleştirilmelidir ki, bu kablolara yapılacak bağlantılar olabilecek en kısa uzunlukta olmalıdır. Bu uygulama tüm sayısal sistemlerde izlenmesi gerken bir tasarım uygulamasıdır.



Şekil 9.1 Tel-sargılı kart

8.2 A. D. Hearn, "Top-down modular programming", *Byte*, Vol.3 (7, 1978)

8.3 G. L. Richardson, C. W. Butler ve J. D. Tomlinson, *A Primer on Structured Program Design*, Petrocelli Books (1980).

Sorular

8.1 Bir mikroişlemci sistemin toplam tasarım/geliştirme bütçesi donanım yazılım geliştirme maliyetleri biçiminde bölümlendirilebilir. Normalde hangi bileşen daha büyük olacaktır?

8.2 İyi tasarlanmış bir programda bulmayı beklediğiniz dört özelliği anlatınız.

8.3 Programlama ve program kodlama arasındaki farkı açıklayınız.

8.4 Program bakımı ile ne anlatılmak istendiğini açıklayınız.

8.5 Program tasarımına modüler yaklaşımın dört avantajını ve dört dezavantajını söyleyiniz.

8.6 Programlama verimliliği genel olarak, verilen bir hesaplama görevini çekleştirmek amacıyla hata ayıklaması yapılmış bir program üretmek için harcanan zaman olarak tanımlanmaktadır. Bu tanımdan hareketle assembly dilinin mi yoksa BASIC veya PASCAL gibi yüksek-düzeyli bir dilin mi daha verimli olduğunu söyleyiniz.

8.7 Hangi tip programda yüksek-düzeyli bir dil yerine assembly dilinin tercih edilmesi gerekir?

8.8 Program yazımında yukarıdan-aşağıya yaklaşım ile ne anlatılmak istenmektedir?

8.9 Bir mikroişlemci, klavye aracılığıyla verilen koordinatlarla belirlenen yerlere delikler açan bir matkap makinesini denetlemektedir. Programcı, delme denetim programını aşağıda *(a)*, *(b)*, *(c)* ve *(d)* biçiminde gösterilen dört alt-bölüme ayırmaya karar vermiştir:

(a) Klavyeden veri girişi
(b) Matkap ucunun yerinin denetimi
(c) Delme işleminin uygulanması
(d) Sistemin başlatılması ve kullanıma hazırlanması

Ayrıca matkap ucunun yerini denetleyen program modülünün aşağıda gösterilen iki alt-bölüme ayrılması da kararlaştırılmıştır:

(e) Matkap ucunun X ekseni boyunca denetimi
(f) Matkap ucunun Y ekseni boyunca denetimi

Programı hiyerarşik ayrışma diyagramını kullanarak tanımlayınız.

8.10 Bir akış diyagramı ve sözde-kod yardımıyla, tekrarlanan toplama yöntemini kullanarak, 500 bellek konumu içeriğini 7 ile çarpan bir program tanımlayınız. Hesaplamanın sonucu 600 adresinde saklanmalıdır.

8.11 500 adresinden başlayan elli sayılık bir listedeki her sayıya 5 ekleyen bir program tasarımı yapınız. Program bir DO UNTIL döngüsü kullanmalıdır. Programın akış diyagramı ve sözde-kodu tanımı verilmelidir.

8.12 1000-10FF bellek alanında saklanan sayıların kaç tanesinin pozitif ve kaç tanesinin negatif sayı olduğunu belirleyen bir program gereklidir. DO UNTIL döngüsü içinde gömülü bir IF THEN ELSE yapısı kullanan uygun bir program tasarımı yapınız. Programı a) bir akış diyagramı, b) sözde-kod kullanarak tanımlayınız.

8.13 Soru 8.10'daki tekrarlanan toplama algoritmasını kullanarak 1000 adresinden başlayan elli sayılık bir listedeki her bir girişi 5 ile çarpan bir program tasarlayınız. Program yapısı bir DO UNTIL döngüsü içinde yer alan başka bir DO UNTIL döngüsü biçiminde olmalıdır. Sözde-kod kullanarak programı tanımlayınız.

8.14 1000 adresinde işaretli bir ikili sayı saklıdır. Sayının işaretini test eden bir program tasarlayınız. Sayı pozitif ise bir sembolik adres olan POS'un içeriği belli bir sayı kadar arttırılmalı, sayı pozitif değilse NEG'in içeriği arttırılmalıdır. Program IF THEN ELSE yapısı formunda *(a)* bir akış diyagramı *(b)* sözde-kod kullanılarak tanımlanmalıdır. [*Not:* Negatif sayılar en büyük değerlikli (işaret) bite sahiptir =1.]

Bölüm 9 Donanım Geliştirme

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde;*

1 *Mikroişlemci sistemleri geliştirmesinde,*
   (a) *Prototip donanım ve firmware yapımı ve testi*
   (b) *Sistem bileşenlerinin, (a) aşamasına yardım amacıyla simülasyo emülasyonu*
   (c) *Donanım ve/veya firmware'in (a) ve (b) aşamalarının sonucuna değiştirilmesi olarak belirtilen ileri aşamaları analize edebilmeli,*

2 *Test ve hata ayıklama periyodunun ilk geliştirme çalışmalarına oranının görevinin karmaşıklığına paralel olarak önemli ölçüde arttığını kavrayabil*

3 *Mikroişlemci geliştirme sistemlerinin, donanım ve yazılım tasarım ve te yardımcı unsurlar olarak kullanımı konusunda daha geniş bir anlayış kaza olmalısınız.*

9.1 Donanım geliştirme

Bu bölümde, mikroişlemci sistem donanım geliştirmenin ileri aşamalarından larını ele almaktayız. Sistem planlama işleminin doğru şekilde gerçekleştirild ve ana kararların Bölüm 6'da anlatılan şekilde alındığını varsaymaktayız noktada görev, ilk devre diyagramında çalışan bir donanım prototipi üretm Prototipin üretilmesi ve test edilmesinde devre-içi emülasyon olanakları bul bir mikroişlemci geliştirme sisteminin nasıl yardım sağladığını anlatacağız.

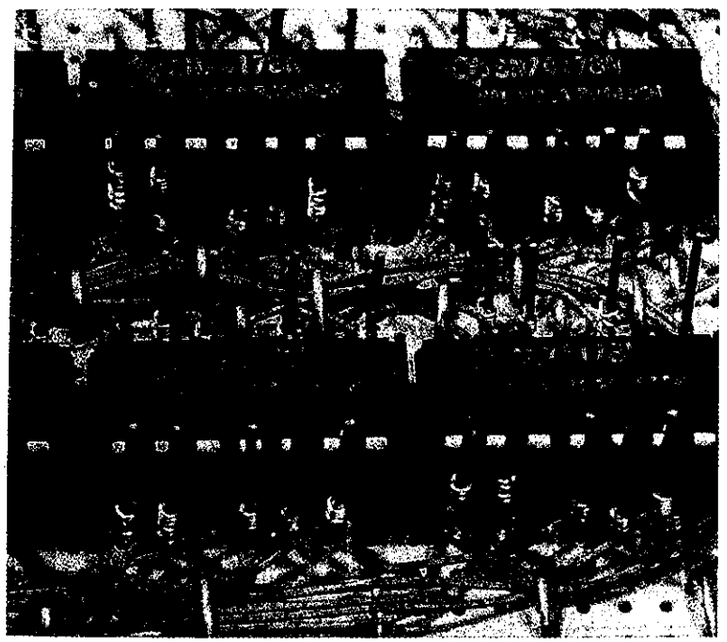
9.2 Prototip donanımın yapım ve testi

Çok sayıda üretilen sistemler normalde entegre devre, kondansatör ve direnç çeşitli bileşenlerin özel olarak tasarlanmış baskılı devre kartlarına monte edil siyle oluşturulurlar. Ancak baskılı devre tasarımının aşama kaydedebilmesi önce, devre tasarımının tamamlanması ve tam olarak test edilmesi gerekir. Do nım prototipinin amacı budur.

Prototip devre kartı yapım yöntemleri içinde en çok bilineni, tel sargı yöntem Bu yöntem ayrıca, çok küçük sayılarda üretilen sistemlerin yapımlarında da y

olarak kullanılmaktadır. Bu yöntemde entegre devreler uygun prototip kartlardaki soketlere takılır. Bu kartlar standart çift sıralı entegre devre paketinin bacaklarının yerleşim düzeni ile aynı şekilde 0.1- inçlik matrislerde delikler açılarak delinir. Entegre devre soketlerinde tel-sargılı bağlantılara uygun bacaklar bulunmaktadır. Bu nedenle ara bağlantılar, entegre devre soketlerinin bacaklarına bağlı teller kullanılarak yapılmaktadır. Tel sargı tekniği, gereken yerlerde devre değişikliği yapılmasına olanak veren, yüksek düzeyde güvenilir bağlantılar yapmak için elverişli bir yöntemdir. Şekil 9.1'de, tipik bir tel-sargılı prototip kartını gösterilmektedir.

Entegre devrelerin prototip kart üzerindeki yerleşim düzenlerini, ara bağlantıları olabilecek en kısa ölçüde tutmak amacıyla, dikkatli bir şekilde düşünerek gerçekleştirmek gerekmektedir. Arzu edilen fiziksel düzene sahip devrelerin elde edilmesinde, deneyimin büyük rolü bulunmaktadır. Prototip işlevi gören kartların çoğunluğunda baskılı devre üzerinde d.c. güç kaynağı çıkış kabloları bulunur, entegre devreler o şekilde yerleştirilmelidir ki, bu kablolara yapılacak bağlantılar olabilecek en kısa uzunlukta olmalıdır. Bu uygulama tüm sayısal sistemlerde izlenmesi gerken bir tasarım uygulamasıdır.



Şekil 9.1 Tel-sargılı kart

Güç kaynağı kablolarının uygun aralıklarda ayırma (dekuplaj) işleminin ya da önem taşır. Kablolar boyunca her iki ya da üç entegre devrede bir, poz toprak arasına 10 ile 100 nF arasında seramik disk kondansatörler bağlan Tel-sargılı prototiplerde her bir entegre devrenin ardından ayırma yapma say uygulama değildir. Ayırma kondansatörünün uçlarının uzunluklarını kısa da önemlidir. Kablolardaki bir milimetre uzunluğundaki fazlalık bile, kond
S9.1 rün empedansını arttıracak ve ayırmadaki verimliliği azaltacaktır.

Bazı mühendisler tüm mikroişlemci sistemine tel-sarma uygulamasını ter mekte ve bunun ardından sistematik bir test ve hata ayıklama işlemi gerçe mektedir. Burada ise farklı bir seçenek sunulmaktadır. Bu seçenekte dev parça parça oluşturulması söz konusudur. Her bölüm bir sonraki bölüme geç önce test edilmekte ve hata ayıklama işlemi uygulanmaktadır. Ancak bunun da bu yaklaşım, pek yüksek bir olasılık olmasa da, birkaç tasarım ve/veya kal ma hatası bulunan bir sistemin hata ayıklama işlemiyle karşı karşıya bul olasılığını ortadan kaldırmaktadır.

9.3 Prototip yapımında adım-adım yaklaşımı

Burada anlatılan yapım uygulamasının sırası, başka bazı çeşitlemeler mümkü de, tipik bir uygulama sırasıdır. Test, her aşamada, MDS'de çalışan basit, özel programları kullanılarak gerçekleştirilir. Amaç, bir araya getirilmeden önce, sı donanımı ve sistem yazılımının işlevsel olarak doğru olduğundan emin olun dır.

Ele aldığımız prototip yapım/test sırasını anlatırken sistemin düzenlemesini, § 4.30'dakine benzer olduğunu varsayacağız. Yöntem genel çerçevede tanıtılm dır. Daha özgün bir donanım prototipi yapım ve testi, ayrıntılı olarak Bölüm 1 anlatılmıştır. Devre-içi emülatörün (ICE) kullanımını tanıtırken test edilen prot hedef sistem olarak adlandıracağız. Bu terminolojiyi kullanan bazı yazarlar ay MDS'nin kendisini host sistem olarak adlandırırlar. Yapım/test işlemleri aşağı gibidir:

1 Bütün entegre devre soketleri, prototip kartı üzerine monte edilir ve üy bacaklara d.c. güç kaynağı bağlantıları yapılır. Ayrıca bu aşamda, ayırma kond satörleri de yerleştirilir.

2 Bu aşamada MİB yonga takımı kablolama işlemi tamamlanır. d.c. güç kayn hat bağlantılarını yaparken bir osiloskop kullanılarak sistem saatinin dalgabiç gözlenebilmelidir. Saat dalgabiçiminin, gereken özellik sınırları içinde oldu

görmek üzere denetlenebilir. Bu aşamada MİB yonga takımı, Kısım 12.10'da anlatıldığı gibi boşta-çalışma durumunda çalıştırılabilir. Böyle yapılırsa, doğru çalışıp çalışmadığı bir kez daha kontrol edilmiş olur.

3 Bundan sonra artık RAM kablolanması gerçekleştirilebilir. Eğer RAM büyük boyutta ise bu kablolama işlemi en önemli kablolama işlemlerinden birini olacaktır. RAM'in test edilmesi için geliştirme sisteminin ICE olanağı kullanılır. İlk önce MİB yongası soketinden çıkarılır ve yerine emülatör probu yerleştirilir. ICE'nin bellek yerini alma olanağının, iç MİB'den kaynaklanan bellek okuma/yazma isteklerinin, geliştirme sisteminin iç belleğine veya dış belleğe yöneltilmesine imkan tanıdığı, Kısım 7.5'ten hatırlanacaktır. Bu durumda hedef sistemin, RAM adres alanındaki adreslere yönelik okuma/yazma işlemlerinin burada belirtilen dış RAM'e yöneltildiğini tanımlarız. Bu bellek yerini alma yöntemi, MDS'nin, hedef sistemin RAM alanından veri okumasına ve bu alana veri yazmasına olanak tanımaktadır.

```
                                                                    PAGE  1
..... 8080/85 ASSEMBLER V01  ..............................  FUTUREDATA  .........

                 * ****************************************** *
                 *                                            *
                 * BU PROGRAM 1000-1FFF BELLEK ALANINI TEST EDER. TEST BAŞARILI OLURSA PROGRAM *
                 * ÇIKIŞTA HL'DE 2000'İ BIRAKIR. TEST BAŞARISIZ OLURSA PROGRAM ÇIKIŞTA HL'DE İLK *
                 * HATALI KONUMU BIRAKIR.                     *
                 *                                            *
                                          YAPI
                                          ****
                 *                                            *
                 * ISARETCI= 1000                             *
                 * DONGU = 1000                               *
                 * DO UNTIL DONGU = 0                         *
                 *     00'İ İŞARET EDİLEN ADRESE SAKLA        *
                 *     BAYTI İŞARET EDİLEN ADRESTEN OKU       *
                 *     IF BAYT = 00                           *
                 *         THEN                               *
                 *             CONTINUE                       *
                 *         ELSE                               *
                 *             DONGU = 1 (ÇIKIŞ DEĞERİ)       *
                 *             ISARETCI = ISARETCI - 1 (ÇIKIŞ DEĞERİ) *
                 *     END IF                                 *
                 *     FF'Yİ İŞARET EDİLEN ADRESE SAKLA       *
                 *     BAYT'I İŞARET EDİLEN ADRESTEN OKU      *
                 *     IF BAYT = FF                           *
                 *         THEN                               *
                 *             CONTINUE                       *
                 *         ELSE                               *
                 *             DONGU = 1 (ÇIKIŞ DEĞERİ)       *
                 *             ISARETCI = ISARETCI - 1 (ÇIKIŞ DEĞERİ) *
                 *     END IF                                 *
                 *     ISARETCI = ISARETCI + 1                *
                 *     DONGU = DONGU - 1                      *
                 * END DO                                     *
                 *                                            *
                 * ****************************************** *

0000                          RSEG        TEC92
                 *
0000  210010                  LXI         H,X'1000'          ISARETCI = 1000
0003  010010                  LXI         B,X'1000'          DONGU = 1000
                 *
0006  3E00       TEST         MVI         A,0
0008  77                      MOV         M,A                0'I SAKLA
0009  7E                      MOV         A,M                BAYT AL
```

Şekil 9.2

```
                                                                    PAGE  1
..... 8080/85 ASSEMBLER V01 ..............................          FUTUREDATA .........

                    * IF THEN ELSE BURADA BAŞLAR
000A  FE00          IF1         CPI     0           BAYT = 0 MI?
000C  CA1300                    JZ      ENDIF1      EVETSE ATLA
000F  010100        ELSE1       LXI     B, 1        DONGU = 1
0012  2B                        DCX     H           ISARETCI'YI AZALT
0013  3BFF          ENDIF1      MVI     A, X'FF'
0015  77                        MOV     M, A        FF'YI SAKLA
0016  7E                        MOV     A, M        BAYTI AL
                    * IF THEN ELSE BURADA BAŞLAR
0017  FEFF                      CPI     X'FF'       BAYT = FF MI?
0019  CA2000                    JZ      ENDIF2      EVETSE ATLA
001C  010100        ELSE2       LXI     B, 1        DONGU = 1
001F  2B                        DCX     H           ISARETCI'YI AZALT
0020  23            ENDIF2      INX     H           ISARETCI'YI ARTTIR
0021  0B                        DCX     B           DONGU'YÜ AZALT
0022  3E00                      MVI     A, 0
0024  B9                        CMP     C           C = 0 MI?
0025  C22D00                    JNZ     TEKRAR      HAYIRSA ATLA
0028  B8                        CMP     B           D = 0 MI?
0029  C22D00                    JNZ     TEKRAR      HAYIRSA ATLA
002C  C9                        RET                 ÇIK
002D  C30600        TEKRAR      JMP     TEST        YENIDEN YAP
0030                            END
```

Şekil 9.3

Dolayısıyla ICE yardımıyla, RAM adreslerini veriyle yükleme ve RAM adresl nin içeriğini okuma işlemlerini elle yapabilmekteyiz. Ancak hedef sistemi test menin çok daha etkin bir yolu, bir bellek test programını kullanmaktır. Bu progr geliştirme sisteminin belleğinde çalışacaktır. Bir örnek bu anlatılanların daha biçimde anlaşılmasını sağlayacaktır. Hedef sistemin 4000-5000 adres alanında alan bir RAM'i bulunduğunu varsayalım. Geliştirme sistemine, belleğin bu alan yönelik bütün okuma/yazma işlemlerinin hedef sisteme yöneltileceğine ilk komutlar veririz. Diğer bütün okuma/yazma işlemleri ise geliştirme sis- temin belleğine yöneltilecektir. Bunları belirledikten sonra, bir bellek test programıy ve sözgelimi 1000 konumuna yerleştirilir. Bu şekilde program geliştirme sisten iç belleğinde çalışabilir, fakat hedef sistemin belleğini test etmiş olur.
Şekil 9.2 ve 9.3'de, sabitlenmiş baytları, bu örnekte 00 ve FF'yi, sırayla h RAM adresine yazan basit bir bellek test programı gösterilmiştir. Adresin iç daha sonra doğru baytın bulunup bulunmadığını kontrol etmek amacıyla okum tadır. Bu kapsamlı bir bellek testi değildir ve genel bir şekilde genel il göstermektedir. Daha etkin bellek test programları her RAM konumuna mü olan bütün 8-bitlik desenleri yazar ve buna ek olarak, doğru baytın bul bulunmadığını kontrol etmek için bir okuma işlemini de içerir.

Eğer bir bellek test işlemi başarısızlığa uğrarsa, yapım çalışmasının daha aşamalara geçebilmesi için önce hatanın giderilmesi gerekir. 1 ve 0 konumla

sürekli biçimde sıkışan veri satırlarının saptanması, eğer ICE kullanılırsa zor olmayacaktır. Eğer bir ya da daha çok tel 0 düzeyinde kalmışsa, bir adrese 00 yazılması işlemi başarıya ulaşır. Aynı adrese FF yazma girişimi ise, başarılı sonuç vermez, bu girişimin sonunda, hatalı veri satırlarına ilişkin bit konumlarında sıfırlar kalır.

Bu aşamada bulunabilen bir başka hata ise, RAM adres alanlarına yönelik bellek okuma/yazma işlemleri sırasında RAM yongalarının bir bölümünde ya da tamamında etkin yonga yetkilendirme sinyallerinin bulunmamasıdır. Burada da yine geliştirme sisteminde çalışan bir test programı, hatanın yanlış kablolamadan mı yoksa yanlış adres kod çözücü tasarımından mı kaynaklandığını belirlemede yardımcı olabilir. Hatalı alandaki RAM'e sürekli olarak yazma ve okuma işlemleri uygulayan, basit ve sonsuz bir döngü programı yazılır. Eğer yonga adres kod çözümleme devresi gerektiği gibi işlev görüyorsa, bir osiloskop kullanılarak düzenli bir yonga yetkilendirme darbeleri dizisi gözlenebilmelidir. Bu yöntem, bu bölümün daha ileriki kısımlarında ve Kısım 12.18'de anlatılan G/Ç portlarındaki hataların yerlerinin belirlenmesi yöntemine benzer bir yöntemdir.

4 RAM, kullanılabilir durumda olduğuna göre artık ROM'un kablolanması yapılabilir. Burada "yazma ve ardından okuma" tipindeki test programları, bellek birimlerinin sadece-okunur olma özelliğinden dolayı kullanılamaz. Ancak bilinen bir veri içeren özel olarak programlanmış ROM, uygun bir test programı ile birlikte kullanılabilir. Program, her ROM adresinde bulunan veriyi okur ve bunu adresin bilinen içeriği ile karşılaştırır. Bu aşamada daha sık karşılaşılan hatalardan bazıları, adres ve/veya veri tellerinin kısa devre olmaları ve yonga yetkilendirme kod çözücü devrelerin yetersizliğidir. Bu hataların saptanması MDS ICE olanağının RAM hata ayıklamasında anlatıldığı gibi kullanılmasını gerektirir.

ROM'da saklanan bir programın başarılı şekilde çalıştığı görülmeden ROM'un tam olarak test edildiği söylenemez. Bunun için de yine uygun bir test programı içeren özel bir ROM'un bulunması gerekmektedir. ROM'un programlanır ve tercihen silinebilir özellikte olması çok yardımcı olur. Bu özellikler yardımıyla test programları, kolaylıkla saklanabilir ve daha sonra gerekirse silinebilir. Test programının gerçek işlevi, programın başarılı bir şekilde çalışmakta olduğunun gösterilmesi koşuluyla, önemli değildir.

5 Artık, giriş ve çıkış port devreleri, her bir port tamamlandıkça test işlemi de yapılarak sisteme kablolanır. Portların test edilmesi amacı için genellikle G/Ç birimlerinin simülasyonu gereklidir. Örneğin eğer sistem 100 tonluk bir basıncı denetliyorsa, bu birimi, denetleme sisteminin ilk testlerinde kullanmak mantıklı değildir. Sayısal çıkışlar, LED kullanılarak uygun bir biçimde izlenebilir. Herbir

sayısal giriş, anahtarlar kullanılarak kolaylıkla simüle edilebilir. Analog çıkı
testmetreler kullanılarak izlenebilir ve analog girişler potansiyometreler ve geril
kaynakları kullanılarak benzetimlenebilir. Tabii ki sonuçta, gerçek sistem çevre
rimleri bağlanmış olarak testler yapılacaktır, ancak bu aşama bunun için her
S9.4 erkendir.

G/Ç portlarının test edilmesi için kullanılan programların çalıştırılmasında da yi
MDS kullanılır. Örneğin bir çıkış portu hatlarına LED'lerin bağlanmış olma
durumunda, programın yalnızca tamamı sıfır'larla dolu bir bayt ve bunun arkasın
dan tamamı bir'lerle dolu bir bayt çıkışı yapması gereklidir. 0 veya 1'de satır
(hat) kısa devresi varsa bunun saptanması, portun veri çıkışı yapmadaki başarısız
lığının saptanması gibi kolaylıkla gerçekleştirilebilir. Eğer böyle bir durum varsa
porttan, bir sonsuz döngü içinde sürekli olarak veri çıkışı yapmaya çalışan bir basit
program kullanılır. Bu program çalışırken bir osiloskopla port adresi kod çözücü
lerinin çıkışı gözlenebilmelidir. Düzenli bir port yetkileme darbe dizisi elde edil
melidir. Bu yöntem Bölüm 12.8'de de tartışılmıştır.

Lambalar ve anahtarlar, sistem bileşenlerinin testi için her zaman uygun olmazlar.
Seri G/Ç portlarını test etmek için uyumlu seri veriyi alan ve veren çevresel birimler
kullanmak gerekir. Örneğin bir VDT, asenkron veriyi alabildiği ve iletebildiğinden
asenkron G/Ç portları için yararlı bir test birimidir. MDS'de çalışan test programı
ları, VDT klavyesinin tuşlarına basıldığında karakterleri alabilir. Asenkron çıkışı
test etmek için, basit bir test programı, karakterleri terminal ekranında görüntüle
yebilir.

G/Ç arabirim ve port devresinin tamamlanmasından sonra donanımın tek başına test
edilmesi yeterince gerçekleştirilmiş olur. ROM'da bulunan ve bütün donanım
sistemlerinin denenmesini sağlayan özel bir test programını kullanmak mümkündür,
fakat bu aşamada bir sonraki ana geliştirme aşamasına geçmek belki de daha yararlı
olur. Bu aşama, sistem donanım ve yazılımının bir araya getirilmesi aşamasıdır.

9.4 Donanım ve yazılım/firmware'in bir araya getirilmesi

Bu bölümde ve daha önceki bölümlerde donanım ve yazılım geliştirmenin ilk
planlama aşamasından sonra nasıl bir yol izleyebileceğini anlatmış bulunuyoruz.
Daha önce de belirttiğimiz gibi, donanım ve yazılım geliştirmenin büyük bir
bölümü paralel olarak yürüyebilir. Bundan dolayı, tasarım/geliştirme ekibi belli bir
aşamada öyle bir noktaya ulaşır ki, donanım ve yazılımın ayrı ayrı geliştirilme ve
testi açısından yapılabilecek her şey yapılmıştır. İşte bu noktada donanım ve

yazılımın bir araya getirilmesi, sistemin bir bütün olarak test edilmesi ve hata ayıklama işleminin gerçekleştirilmesi gerekir.

Eğer sistem belirli bir denetim veya aletlendirme (enstrümantasyon) işlevi için geliştirilen özel bir sistem ise, program sonuçta, tabii ki bir ROM'a yerleştirilecektir. Ancak RAM'a yerleştirilirse, sistemin test edilmesi ve hata ayıklama sırasında program değişikliklerinin kolayca gerçekleştirilmesine yardımcı olur. Bunu MDS ICE olanağının kullanılması sağlamaktadır. Testin son aşamalarında bir ROM-/PROM emülatörü de yardımcı olabilir.

MDS'nin sağladığı sistem belleği yerini alma olanağı, başlangıçtaki donanım/yazılım sistem testinde kullanılmaktadır. Bellek yerini almayı Kısım 7.5'te anlatmıştık. Bu aşamada, hedef sistem yazılımının tümü, simüle edilmiş çevresel birimler veya eğer fiziksel olarak mümkünse bağlantıları yapılmış gerçek çevresel birimlerle çalıştırılabilir. Çevresel birimlerin G/Ç denetim programlarının her biri bağımsız modüllerden oluşacağından, eğer yazılım anlatıldığı gibi modüler biçimde tasarlandıysa her bir program modülünü hedef sistem donanımı ile test etmek mümkün olmalıdır.

Program modüllerinin tek tek test edilmesinin ardından, bütün program, gerçek çevresel birimler kullanılarak çalıştırılabilir. Bu ilk önce MDS ICE ve bellek yerini alma olanağı kullanılarak yapılır. Eğer herşey yolundaysa program ROM'a, ya da çoğu kez olduğu gibi EPROM'a aktarılır. MDS, test edilen hedef sistem programı ile EPROM'u doğrudan programlayabilme özelliğinde olmalıdır. Artık son test işlemi, hedef sisteme takılı EPROM'da çalışan progam ile gerçekleştirilebilir, fakat denetim yine MDS ICE olanağı tarafından yapılmaktadır. Test başarılı ise ICE probu yerinden çıkarılır ve mikroişlemci buraya yerleştirilir. Sistemin performansı, daha sonra orijinal özellikler ile karşılaştırılabilir.

9.5 Test ve hata ayıklamanın önemi

Buraya kadar anlatılanlardan okuyucu, test ve hata ayıklama için harcanan zamanın, toplam sistem geliştirme süresinin çok anlamlı bir bölümünü oluşturabileceğinin farkına varmıştır. Bölüm 8'de yazılım hata ayıklama işlemi tarafından harcanan sürenin, programların büyüklük ve karmaşıklığı arttıkça oransal olarak denetimsiz biçimde yükselebildiğini vurgulamıştık. Geliştirmenin bu aşamasını gereken biçimde kontrol altına almak için düzenli biçimde yukarıdan-aşağıya yapılanmış programlama yöntemleri kullanılmaktadır. Bunun amacı tabii ki küçük program modüllerindeki tasarım hatalarının yalıtılması edilmesi ve böylelikle de hata belirlemelerinin ve giderilmelerinin kolaylaştırılmasıdır.

Büyük ve karmaşık donanım sistemlerinin test edilme ve hata ayıklama işlem yazılımda karşılaşılan sorunların çoğunu gündeme getirir. Bir mikroişlemci sist de bileşen ya da devre elemanlarının işlevlerinde başarısızlık göstermele sonuçları, genellikle hatanın olduğu bölgede sınırlı kalmazlar, bunlar tüm sist boyunca etkilerini gösterirler. Dolayısıyla sistemin diğer bölümleri de doğru olar işlev görmeyebilir. Tamamlanmış bir sistemde çok sayıda hatanın bulunması sor olarak aynı yazılım hatalarındaki gibi hata ayıklama süresinin kontrol edilem biçimde artmasına yol açar. Bu bölümde anlatılan yapım/test çalışmaları, h ayıklama işleminin sıralı bir biçimde gerçekleştirilmesine olanak vermek iç benimsenmiş uygulamalardır. Başlıca amaçlardan birisi bileşen ve devre hatala nın prototip yapımı sırasında ortaya çıktıkları anda tanımlanmaları ve izole edilm leridir. Bu yöntemlerin kullanılması mühendis veya teknisyenin, olabilen en k
S9.5 zamanda çalışan bir prototip üretebilmesine olanak tanıyacaktır.

Sorular

9.1 Tipik bir mikroişlemci sisteme ait tel-sargılı prototip kartında, güç kaynağı kablosu ayırması için ne gibi olanaklar içermektedir?

9.2 Prototip yapımı sırasında bir devre-içi emülatörün a) bir G/Ç portu b) RAM testinde nasıl kullanılabileceğini anlatınız.

9.3 ROM veya PROM test işleminin RAM test işleminden ne şekilde farklılık göstermesi gerektiğini açıklayınız.

9.4 Sistem testi amacıyla kullanılan iki basit simülasyon çevresel birimini tanımlayınız.

9.5 Donanım test ve hata ayıklama periyodunun başlangıç geliştirme periodun oranı, sistem görevinin karmaşıklığına paralel olarak neden önemli ölçüde artar, açıklayınız.

Bölüm 10 Mikroişlemci Sistem Tasarım Çalışması

Bu bölümün amaçları - *Bu bölümü bitirdikten ve alıştırmaların tümünü yaptıktan sonra tipik, küçük, mikroişlemci-tabanlı bir sıralı denetim sisteminin tasarım/geliştirme aşamalarını analiz edebilmelisiniz.*

10.1 Tasarım/geliştirme programı

Bu bölümde mikroişlemci sistem tasarımının genel konularını bir tarafa bırakacak ve belirli bir durumu ele alacağız. Oldukça basit yapıdaki bir mikroişlemci-tabanlı işlem denetim sisteminin donanım ve yazılım tasarımı ve geliştirmesinin çeşitli aşamalarını inceleyeceğiz. Devam etmeden önce mikroişlemci sistemlerinin geliştirilmesinde kullanılan tipik aşamaları hatırlamakta yarar vardır:

1 Tüm sistemin ilk özellikleri
2 Özellikle donanım/yazılım bölüşümü konusu başta olmak üzere maliyet-verimlilik ölçümleri için formüller oluşturulması
3 Sistem akış diyagramının çizimi.
4 Gerekli donanım birimlerinin seçimi: mikroişlemci, arabirim bileşenleri vb.
5 Mikrobilgisayar programının sınırlarının tanımlanması; zamanlama ve adres alanı
6 Program görevinin modüler biçimde bölümlenmesi. Mümkünse bir program akış diyagramının geliştirilmesi
7 Modül yapısı ve program kodlamasının tasarımı
8 Programın test edilmesi
9 Donanım ve yazılımın birleştirilmesi. Gerekli değişikliklerin yapılması
10 Sistemin simülasyon/emülasyonu
11 Gereken daha ileri düzeydeki değişikliklerin yapılması
12 Donanım ve sabit bellek temel örnekleme
13 Gerekiyorsa, tasarım/geliştirme/test çevriminin tekrarlanması

Bu liste, çeşitli uygulamaların daima kesinlikle bu sırada gerçekleştiği şeklinde anlaşılmamalıdır. Tipik bir tasarım/geliştirme programında birçok tekrarlama gerekebilecektir. Örneğin, sistem donanım ve yazılımının ilk kez bir araya getirildiği 9. aşamaya ulaşıldığında, programın bazı ayrıntılarının, önceden saptanamayan bir
S10.1 takım güçlükleri ortadan kaldırmak için düzeltilmesi gerekebilecektir.

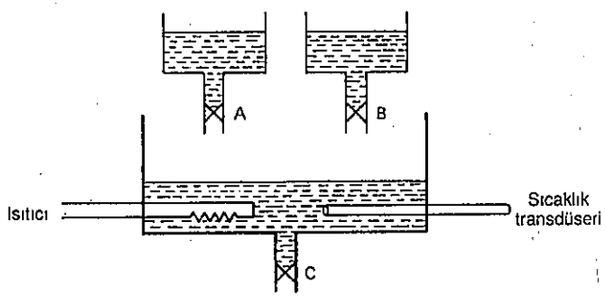
Gerçek geliştirme sırası belli ölçüde, destekleyen donanım ve yazılıma da bağlı olacaktır. Ele aldığımız örnekte yazılım destek sisteminin, ayrı ayrı program modüllerinin tek bir program içinde birleştirilmesine imkan veren bir bağlayıcı ile birlikte, tam anlamıyla yeri değiştirilebilir bir çevirici sağlamakta olduğunu varsaymaktayız.

10.2 Sistem özellikleri

Denetim sistemine ihtiyaç duyan bir sistemin şematik diyagramı Şekil 10.1'de gösterilmiştir. A ve B valfleri açık olduklarında sıvı üst kaplardan daha aşağıdaki C kabına akmaktadır. C valfinin açılması C kabını boşaltmaktadır. Aşağıdaki kapta ayrıca bir sıcaklık transdüseri ve bir de ısıtıcı vardır. Şekil 10.1'de gösterilen sistem iki kimyasal işlemi gerçekleştirmede kullanılmaktadır. Bu kimyasal işlemlerin her ikisinde de sıvılar karıştırılmakta ve çeşitli sürelerde ısıtılmaktadır.

Bölüm 6'dan hatırlanacağı gibi, tasarımın ilk döneminde kesin bir sistem özelliklerini belirlemek önem taşımaktadır. Bu durumda özelliklerin iki sistem değişkeninin duyarlık ve çözünürlüklerini belirlemesi gerekmektedir, bu değişkenler sıcaklık ile sürelerdir.

Sistem özelliklerinde yer alması zorunlu olmasa da, tasarımın son durumunun maliyet-verim açısından yüksek düzeyde olmasını sağlayabilmek için bu noktada daha fazla bilgiye ihtiyaç vardır. Bu bilgilerden en önemlisi üretilecek sistem sayısıdır. Burada yalnızca iki sistem gerekmektedir. Donanım bileşeni seçimi açısından aynı ölçüde önemli olan bir başka bilgi ise, tasarım ekibinin kullanabileceği destek araç ve gereçleridir. Ekibin 8080-tabanlı sistemlerin tasarımında daha önceden deneyimli olduğunu ve kullanılacak mikroişlemci için bir geliştirme sisteminin hazır bulunduğunu varsayacağız.



Şekil 10.1 Kimyasal işleme tesisi

Sistemin özelliklerinin şimdiye kadar ele alınmayan bir başka önemli yönü ise, operatör ve sistem arasındaki iletişim ile ilgilidir. Belirtilen sistem tam otomatik bir sistem değildir ve bir operatör tarafından denetlenecektir. Kullanılacak denetim düğmeleri, Şekil 10.2'de gösterilmiştir. Sistemde normal bir güç kaynağı açma/kapama anahtarı bulunmaktadır. 1 ve 2 nolu işlemlerin seçilebilmeleri için iki basma düğmesi vardır. Bu düğmeler, ikisinin aynı anda seçilmesini önlemek için, mekanik olarak birbirine kilitlenmişlerdir. Yukarıdaki iki kap yeterli miktarda sıvı ile dolduğu zaman operatör, daha önce seçilmiş olan işlem sırasını, başlatma düğmesine basarak başlatır. Yanlışlık durumunda durdurma düğmesine basılarak işlem durdurulur. Sistemlerin çoğunluğunda ön panel denetimleri önemlidir. Bunların belirlenmesini, bazen ayrıntılı olarak son-kullanıcı yapar, fakat işlevleri çoğu kez sistem tasarımına bırakılır.

Açma/Kapama	İşlem 1	İşlem 2	Çalıştır	Durdur

Şekil 10.2 Denetim anahtarları

Ön panelde bir açma/kapama anahtarının bulunması, başlatma durumunu dikkate almak gerektiğini gösterir. Başlatma durumu, sistemin, anahtarın açılmasından hemen sonra ve başlama düğmesine basılmasından önce bulunmasını istediğimiz durumudur. Sistemin güvenliğini sağlayabilmek için başlatma durumunda A, B ve C valflerin ve ısıtıcının kapalı olmasını belirtiriz. Bu, ayrıca, bir işlemler sırasının tamamlanmasından sonraki durumdur.

10.3 Sistem akış diyagramı

Bir başlangıç sistem akış diyagramı genellikle sistemin özelliklerinden sonra oluşturulabilir. Örneğimizdeki sistem nispeten basit sıralı tarzda çalışan bir sistemdir. Bu nedenle Şekil 10.4'te gösterilen akış diyagramı, sistem çalışmasının Şekil 10.3'te yer alan tablo gösteriminden, denetim paneli işlevlerinin ayrıntıları eklenerek elde edilebilir.

10.4 Maliyet-verimlilik kriterlerinin formüllendirilmesi

Bu noktada geliştirme programının her bir aşaması tam açıklığa kavuşmuş değildir. Sistem düzenlemesi, donanım/yazılıma katılacak bileşenlerin maliyetlerine göre belirlenmesi vb. ile ilgili başlangıç kararları alınır. Bu kararlar daha sonra, sistemin ayrıntılı biçimde çalışmasını etkileyebilir, bu durumda sistem çalışması, daha kapsamlı bir akış diyagramı ya da başka bir biçimde anlatılmalıdır.

Sistem, 1. ve 2. kimyasal işlemlere ilişkin iki denetim sırasının birinin uygulanmasını gerektirir. İki işlem sırası aşağıda gösterilmiştir.

1. İşlem Sırası

a A valfini aç.
b Isıtıcıyı aç, sıcaklığın 60 °C'ye kadar yükselmesine izin ver.
c Sıcaklığı, 30 ± 2 dakika süreyle 60 ± 3 °C'da tut.
d B valfini aç.
e (Eğer kapalı ise) ısıtıcıyı aç, sıcaklığın 80 °C'ye kadar yükselmesine izin ver.
f Sıcaklığı, 40 ± 2 dakika süreyle 80 ± 3 °C'da tut.
g C valfini en azından 3 dakika süreyle açık tut.
h Tüm valfleri kapa.

2. İşlem Sırası

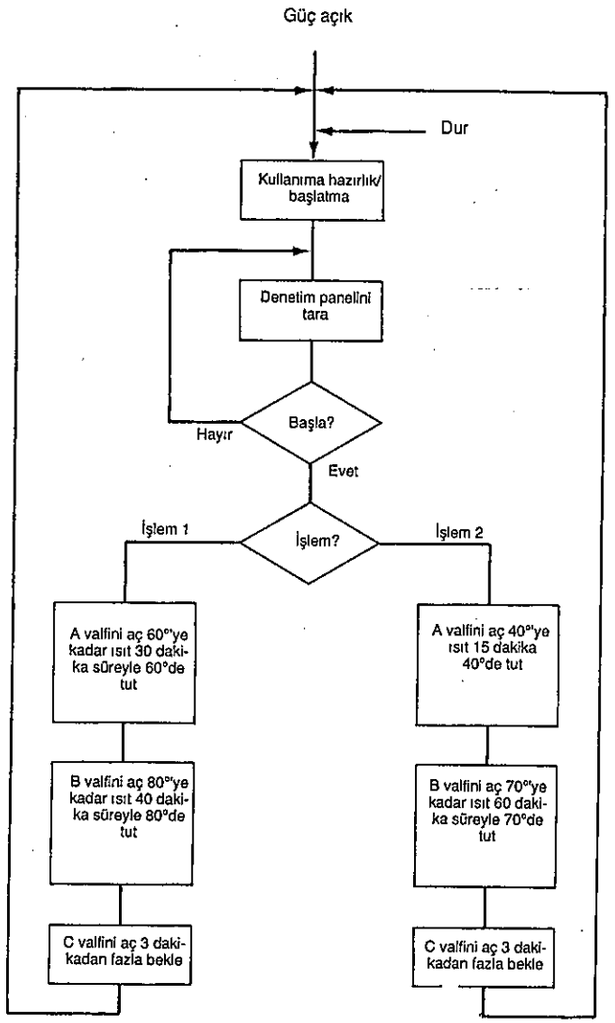
a A valfini aç.
b Isıtıcıyı aç, sıcaklığın 40 °C'ye kadar yükselmesine izin ver.
c Sıcaklığı, 50 ± dakika süreyle 40 ± 2 °C'da tut.
d B valfini aç.
e (Eğer kapalı ise) ısıtıcıyı aç, sıcaklığın 70 °C'ye kadar yükselmesine izin ver.
f Sıcaklığı, 60 ± 2 dakika süreyle 70 ± 3 °C'da tut.
g C valfini en azından 3 dakika süreyle açık tut.
h Tüm valfleri kapa.

• *Şekil 10.3* İşlem denetim sırası

Şekil 10.3'de dikkatimizi çeken ilk noktalardan birisi, iki işlemin genelde birbirinden farklı değildir. Fakat işlemler, sıcaklık ve süreler gibi ayrıntılar bakımından farklılık gösterirler. Bu nedenle, genel işlem sırasını; ilgili süre ve sıcaklık parametrelerinin, bir kullanıma hazırlama bölümünde programa girilmesi şeklinde düzenlemek daha makul görünmektedir. Şekil 10.5'de, bu ana işlem çevriminin sözde-kod kullanılarak genel terimlerle ifadesi görülmektedir. Bu aynı zamanda, program modülünün temel işlem sırasını uygulaması için bir taban oluşturacaktır.

```
SIRA PARAMETRELERİNİ KUR
A VALFİNİ AÇ
T1'E KADAR ISIT
P1 SÜRESİNCE T1'DE TUT
B VALFİNİ AÇ
T2'YE KADAR ISIT
P2 DAKİKA SÜREYLE T2'DE TUT
3 DAKİKA SÜREYLE BEKLE
```

Şekil 10.5 Temel işlem sırasının denetimi



Şekil 10.4 Kimyasal işlem denetimi akış diyagramı

Tasarım çalışmalarının daha ileri aşamalarında, maliyet-verimlilik konusunun diğer yönleri ele alınacaktır. Sistem görevlerinin donanım ve yazılım arasında bölünme yöntemi, bu yönlerin önemli bir kısmını oluşturacaktır. Bu da yine önemli bir karardır ve kullanılacak mikroişlemci tipi belirleninceye kadar ertelenecektir.

10.5 Donanım birimlerinin seçimi

Tasarım/geliştirme çalışmalarının bu aşamasında, donanım düzenlemesi ve donanım elemanlarının seçimi konularıyla ilgili bazı başlangıç kararları vermemiz gerekir. Bu kararlardan diğerlerine göre daha önemli olanı, mikroişlemcinin seçimidir. Sistem özelliklerinden, buradaki basit sıralı işleyişi uygulamak için gereken işlem gücünün en düşük düzeyde olduğunu tahmin edebiliriz. Dolayısıyla ilk tepki, üretim maliyetini en düşük düzeyde tutmak için yonga sayısı az olan bir sistem elde etmek amacıyla, pahalı olmayan 4-bitlik bir mikroişlemciyi seçmek olabilir. Fakat buradaki görevde daha önemli başka etkenler bulunmaktadır. Bu etkenler şunlardır: yalnızca üç sisteme gereksinim bulunması, tasarım ekibinin önceki deneyiminin ve mevcut desteğin 8080 için söz konusu olması. Bu etkenler her türlü teknik varsayımın önünde yer alır.

Sözgelimi 1000 birimlik bir üretim serisi ile, yeni bir mikroişlemci desteğinin ve belki de tasarım ekibinin, bu mikroişlemciyi yeterince tanımaması nedeniyle geliştirme süresinin uzaması ve bunun sonucu olarak maliyete etkisi, üretilen toplam miktar arasında paylaştırılabilir. Böyle bir durum, muhtemelen 8080'den farklı bir birimin kullanılmasının daha mantıklı olacağını ortaya koymaktadır. Üç birimlik
S10.2 bir üretim serisi içinse, 8080 tartışmasız biçimde tek seçenek olmaktadır.

G/Ç gereksinimlerine ilişkin tartışmada, mikroişlemcinin kendisi üzerine yoğunlaşacağız. Bu da, A, B ve C valflerinin, mantıksal sinyallerle etkinleştirilen solenoid sürücü birimleri ile birlikte verildiğini varsaydığımız anlamına gelir. Isıtıcı benzer biçimde denetleyici mantık düzeyinin değiştirilmesiyle açılıp kapanabilir.

Sıcaklık transdüseri bir analog çıkış üretecektir. Bu nedenle bir tür analog-sayısal dönüştürme işlemi gerekli olacaktır. Ayrıca sıcaklık transdüserini sürmek ve seçilen ADC'ye uyumlu bir çıkış gerilimi sağlamak için bir yükseltecin bulunması gerektiğini de öngörmeliyiz.

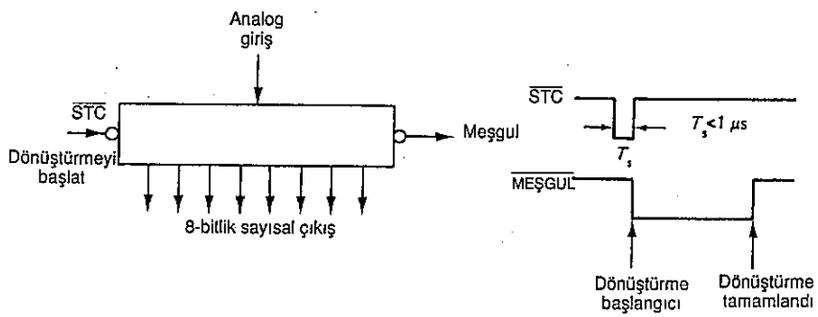
Analog-sayısal dönüştürmesi durumundaki donanım/yazılım bölüşümü, Bölüm 6'da geniş biçimde tartışılmıştır ve orada ulaşılan sonuçlar doğrudan, buradaki duruma da uygulanabilir. Sonuç olarak, sıcaklığın ölçülmesi için sistemin içinde bir ADC kullanılması kararına varılır. Bir DAC/karşılaştırıcı birleşimi kullanarak elde edilecek kısmi yazılım çözüm seçeneği yalnızca üç sisteme gerek duyulması nedeniyle ortadan kalkmaktadır. Bu durumda 8-bitlik bir ADC, yerinde bir seçim olacaktır. Sınırlı sıcaklık çözünürlüğünün gerekli olması daha az sayıda bit bulunduran bir dönüştürücünün kullanılmasına imkan verir, ancak bunun sağlayacağı maliyet avantajı çok düşük düzeydedir. Entegre devre biçiminde çeşitli 8-bitlik

SDC'lerin bulunması uygun bir birimin seçimini kolaylaştırmaktadır. Bu çeşit tipik bir dönüştürücünün düzenlemesi ve zamanlama karakteristikleri, Şekil 10.8'de gösterilmiştir. Görüldüğü gibi dönüştürme, en az 1 mikrosaniyelik bir nega- tife giden darbe ile başlatmaktadır. Dönüştürme gerçekleşirken meşgul hattı alçak mantık düzeyindedir, dönüştürme tamamlandığında yüksek mantık düzeyine döner ve sayısal veri çıkışta kullanıma hazır duruma gelir. Bu çeşit birimlerin bazılarında bulunan üç-durumlu çıkış olanağı, Şekil 10.6'daki ADC'de mevcut değildir.

Sistem özelliklerinin hem donanımı hem de yazılımı etkileyen bir özelliği, işlem sırasındaki zamanlama aralıklarının düzenlenmesinin gerekmesidir. Bunun yapılabilmesinin çok çeşitli yöntemleri vardır, bunlar Kısım 6.7'de tartışıldığı gibi tümüyle donanım veya tümüyle-yazılım yöntemlerinden oluşabilir, ya da bu iki uç nokta arasında değişen nitelikte yöntemler olabilirler. Tümüyle-yazılım yöntemi iyi bilinen ve bir gecikme döngü programının, gereken sayıda tekrarlanasını içeren basit bir yöntemdir. Bu yöntemi kullanmak, zaman aralığı süresince başka herhangi bir bilgi işlem uygulamasının gerekli olmaması koşuluyla, kolaydır. Gecikme döngüsü içine başka yordamları dahil eden zamanlama programlarını tasarımlamak olanaklıdır fakat dikkate alınacak düzeyde sorunlar çıkabilir.

Tümüyle-donanım yöntemi, gecikme üretimine dayalıdır ve bu yöntemde işlemciye, dışardan programlanabilir bir zamanlayıcı devresi vardır. Gecikme gerektiğinde işlemci, zamanlayıcıyı uygun gecikmeyi üretecek şekilde hazırlar ve zamanlama işlemini başlatmak için bir sinyal gönderir. Süre tamamlandığında, progamlanabilir zamanlayıcı, işlemci tarafından taranabilecek bir sinyal üretir ya da alternatif olarak bir gecikme üretir.

Zamanlama aralıkları üretiminde, hem tümüyle-donanım hem de tümüyle-yazılım yöntemlerinin burada ele alınan sistem için dezavantajları bulunmaktadır. Benim-



Şekil 10.6 ADC karakteristikleri

senen uygulama kısmen-donanım ve kısmen-yazılım yöntemidir ve bir gerçek-zamanlı saat kullanılır. Bu yöntem, zamanlama aralıkları üretilmesi için kullanılan çok yaygın bir tekniktir. Gerçek-zamanlı saat, sabit aralıklarda darbeler üretir. Bu darbeler işlemci tarafından, istenen süre tamamlanıncaya kadar sayılır. Burada elde alınan örnekte olduğu gibi eğer zaman aralığı sırasında başka işlem yapılması gerekiyorsa, gerçek-zamanlı saat darbelerinin, saat sinyal hattının bir giriş portu aracılığıyla taranmasından çok, kesmeler gibi kullanılması genellikle daha elverişlidir.

S10.3

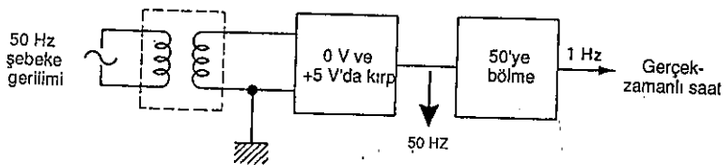
Gerçek-zamanlı saat kullanma yaklaşımının, bir saat darbesi kaynağı gibi donanımla ilgili bazı gereksinimlere yol açması söz konusudur. Ayrıca bu darbelerin hangi frekansta olması gerektiği konusu da hesaba katılmalıdır. Teorik olarak bu frekansın üst sınırı, mikroişlemcinin yanıt verme hızı ile belirlenir. Eğer çok yüksek bir frekans seçilirse darbe sayma programı uzun zaman alacak ve işlemci, zamanının büyük bir bölümünü, kesmelere yanıt üretmek için harcayacaktır. Frekansın alt sınırı ise gereken zaman çözünürlüğü tarafından belirlenir, örneğimizde bu çok düşüktür. Gerçek-zamanlı saat darbeleri, üç yoldan üretilebilir:

1 Bağımsız bir osilatör
2 Uygun biçimde bölümlenmiş mikroişlemci saati
3 Bölünebildiği takdirde, 50 Hz'lik şebeke frekansı

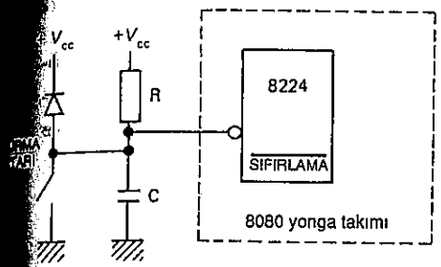
Bu üç olanağın incelenmesi sonunda hepsinin de yeterli bir gerçek-zamanlı saat oluşturabileceği görülmektedir, ayrıca bunların tasarım ve üretim açısından maliyetlerinin birbirinden çok farklı olmaması da şaşırtıcıdır. Ele aldığımız örnekte, sayma programına ilişkin özellikleri basitleştirmek için, 3. yöntemi kullanmaktayız. Şebeke gerilimi bir transformatör tarafından uygun bir düzeye indirilmekte ve sonuçta elde edilen 50 Hz'lik dalgabiçimi 0 ve +5 V'ta kırpılmaktadır. Ardından, bu sinyal bir frekans bölücü ile 50'ye bölünerek 1 saniye aralıklı darbeler üretilmektedir. Devre şematik olarak Şekil 10.7'de gösterilmiştir.

Bu kısımda son olarak Kısım 10.2'de tartışıldığı gibi, güvenli başlatma koşulları sağlamak için söz konusu olan donanım gereklerini ele almalıyız. Bununla bağlantılı olan işlev de denetim panelinde DURDURMA (STOP) anahtarı çalıştırıldığında yapılmakta olan işlemin güvenli biçimde sona erdirilmesidir. Eğer anahtar açık iken ya da DURDURMA anahtarına basıldığında bir SIFIRLAMA (RESET) sinyali üretilecek şekilde düzenleme yaparsak, her iki sorun benzer biçimde çözümlenebilir. Sıfırlama ile 8080, komut yürütümüne, 0 adresinden itibaren başlayacaktır, 0. adres konumu, sistemi güvenli bir biçimde başlatma programını saklamamız gereken yerdir.

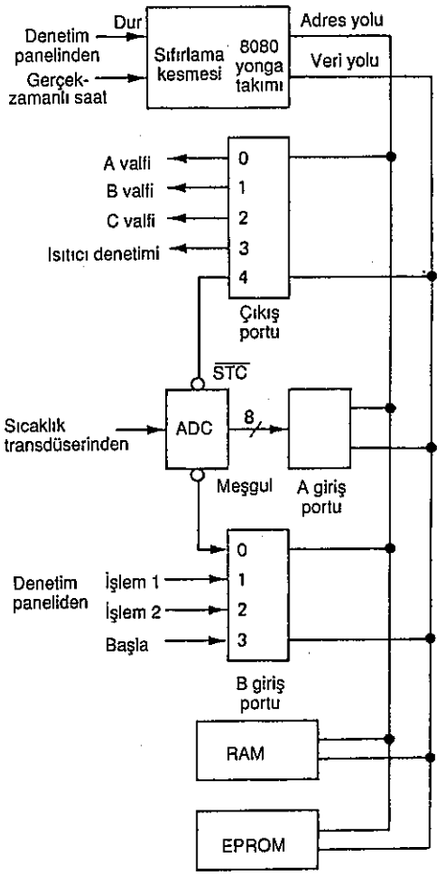
S10.8



Şekil 10.7 Gerçek-zamanlı saat devresi



Şekil 10.8 Açılışta SIFIRLAMA ve DURDURMA anahtarı devresi



Şekil 10.9 Genel sistem düzenlemesi

"Açılışta sıfırlama" adını alan bu gereksinim çok yaygındır. Sıfırlama özelliği, 8080 yonga takımı, daha doğrusu 8224 saat üreteci tarafından sağlanmaktadır. Bu sıfırlama özelliği Ek F'de ayrıntılı olarak anlatılmaktadır ve Şekil 10.8'de uygun bir devre gösterilmiştir. Güç anahtarı ilk kez açıldığında C kondansatörünün dolması belli bir süre alacaktır. Bu, mikroişlemcinin devreye girmesinden sonra sıfırlama hattını bir süre için alçak mantık düzeyinde tutar, böylece bir sıfırlama durumu ortaya çıkar. Durdurma anahtarına basılması da, benzer biçimde sıfırlama
S10.6 oluşturur.

10.6 Sistem donanım düzenlemesi

Maliyet-verimlilik kriterlerinin diğerlerine göre daha önemli olanlarından bir bölümünü ve donanım/yazılım bölüşümü konusunu inceledikten sonra sistem donanım düzenlemesi konusuna gelmiş bulunuyoruz. Bu noktada G/Ç hatlarının sayı ve çeşitleri aşağıdaki gibi tanımlanır:

Giriş hatları

Denetim paneli, başlat	1 hat
Denetim paneli, 1. işlem seçimi	1 hat
Denetim paneli, 2. işlem seçimi	1 hat
Denetim paneli, durdurma	1 hat (sıfırlama)
ADC meşgul	1 hat
ADC veri	8 hat
Gerçek-zamanlı saat	1 hat (kesme)

Çıkış hatları

A valfi denetimi	1 hat
B valfi denetimi	1 hat
C valfi denetimi	1 hat
Isıtıcı denetimi	1 hat
ADC dönüştürmeyi başlatma	1 hat

Buradan, sistemin bir sıfırlama ve bir de kesme girişine sahip olduğu görülmektedir. Bu hatlara ek olarak sistemde, 12 giriş ve 5 çıkış hattı bulunması gerekmektedir. Dolayısıyla 2 giriş ve 1 çıkış portuna ihtiyacımız vardır. Şekil 10.9'da, uygun bir düzenleme gösterilmiştir, bu düzenlemede adres kod çözücüsü ve denetim/durum sinyallerinin ayrıntılarına yer verilmemiştir.

Tartışmamızın bu noktasında, Şekil 10.9'da gösterilen bellek bloklarının büyüklükleri hakkında fazla bir şey söyleyemeyiz. Ancak programın saklandığı bellek bölümünün sadece-okunur türden olacağını söyleyebiliriz. Mikroişlemci yığınının yer alması ve program değişkenlerinin saklanması için bir de RAM'a gerek bulunmaktadır. EPROM birimleri tasarım aşamasında, silinmeyen tiplere göre daha fazla esneklik sunduklarından, Şekil 10.9'da sadece-okunur bellek bölümü için EPROM önerilmiştir. Tasarım, gerçek anlamda son durumuna geldiğinde, istenirse EPROM'ların, diğer tipte bacak-uyumlu ROM'larla değiştirilmeleri olanaklıdır. Bu tasarımda eriyebilir-bağlantılı programlanır ROM kullanılabilirse de, az sayıda elemanın yer almakta oluşu maskeprogramlı ROM kullanımını imkansızlaştırmak-
S10.7 tadır.

Başlangıç donanım kararlarından bir bölümü, bellek adres alanının belli işler için ayrılma yöntemini belirleyecektir. Sistemin, anahtar açık olduğunda veya DURDURMA anahtarına her basışta sıfırlandığını gördük. 8080 mikroişlemcisinde bu, 0 adresine bir atlama gerçekleştirilmesini zorunlu kılar. Bu nedenle uygun program modülünün başlama kodunun bu adreste bulunması esastır. Program kodu EPROM'da tutulacaktır, dolayısıyla sistem EPROM'u, bellek alanının 0'dan başlayan altsıralı adreslerine yerleştirilmiş olmalıdır. Normalde 8080 kesme hizmet yordamı kodu da, belleğin alt-sıralı adreslerine doğru yer alacaktır, bunun tam başlama adresi kesme donanımı tarafından belirlenmektedir. Bu konu Kısım 10.9'da ele alınmıştır.

RAM için adres alanı ayrılması, RAM yalnızca, 8080'in yığını ve geçici verilerin saklanması için kullanılacağından, daha fazla esneklik olanağı sağlar. Ancak RAM adres alanının dikkatli bir biçimde seçilmesi, bellek yongaları için yonga yetkilendirme sinyalleri üreten devre düzenlemesinin çok kolaylıkla yapılmasını sağlayabilir. RAM'e 8000'den başlayarak bellek alanı ayıracağız. Özellikle böyle bir seçim yapılmasının nedeni, Kısım 10.8'de, belleğin tasarımı konusunu tartıştığımızda daha açık olarak anlaşılabilecektir.

Bir tasarım uygulaması durumunda donanım ve yazılım tasarımı normalde, Şekil 6.1'de gösterildiği gibi ancak belli bir yere kadar paralel olarak ilerleyebilir. Bu nedenle biraz deneyimi bulunan bir yazılım tasarımcısı, bu aşamada, gereken bellek büyüklüğü hakkında oldukça yaklaşık bir fikre sahip olabilir. Ancak prototip amaçları için bu mutlaka gerekli değildir, çünkü bellek özellikleri daha sonraya bırakılabilir ve böylelikle G/Ç donanımı tasarımının devam etmesine imkan tanınmış olur.

10.7 G/Ç port tasarımı

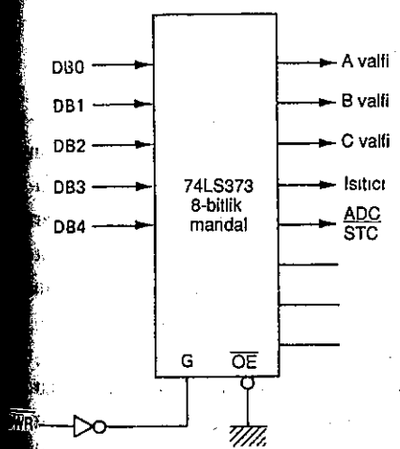
Tasarım işlemi bu ana kadar G/Ç portları için herhangi bir olağan dışı gereksinim ortaya çıkarmamıştır. Dolayısıyla, üç-durumlu tampon ve mandal kullanımına dayalı çok basit devreler kullanılabilir. Fakat port adreslerinin kodlarının çözülmesi sorusunu, port adreslerinin dikkatlice seçilmesiyle donanımda önemli tasarruflar sağlanabileceğinden, ayrıntılarıyla ele almakta yarar vardır. Örneğin ilk dikkatimizi çeken şey, sistemin yalnızca bir çıkış portuna ihtiyacının bulunmasıdır. Bu, adres kodlarının çözümüne hiç bir biçimde gerek olmadığı anlamına gelmektedir. Port, 8080 yonga takımından gelen $\overline{IOWR}$ sinyali ile kolayca yetkilenirler. Adres ne olursa olsun herhangi bir OUT işlemi, veriyi bu porta aktaracaktır. Programlama elverişliliği bakımından port için 0 adresini ayıracağız, fakat bunun donanım açısından hiçbir anlamı olmayacaktır. Çıkış portu devresi, Şekil 10.10'da gösterilmiştir.

Birden fazla giriş portu gerektiğinden, IN işlemleri için bazı port adres kod çözümlemeleri gerekecektir. Burada da yine donanım, adres seçimi yoluyla en düşük düzeyde tutulabilir. Şekil 10.11'de gösterilen giriş portları devresinde A portu, 0.bitin 0 olduğu bir adrese sahip herhangi bir IN işlemi tarafından etkinleştirilir. Benzer biçimde B portu, 1. bitin 0 olduğu port adreslerine sahip IN işlemleri tarafından etkinleştirilir. Diğer başka adreslerin de kullanılabilmesine karşın yine elverişlilik nedeniyle, iki giriş portu için FE ve FD adreslerini kullanacağız.

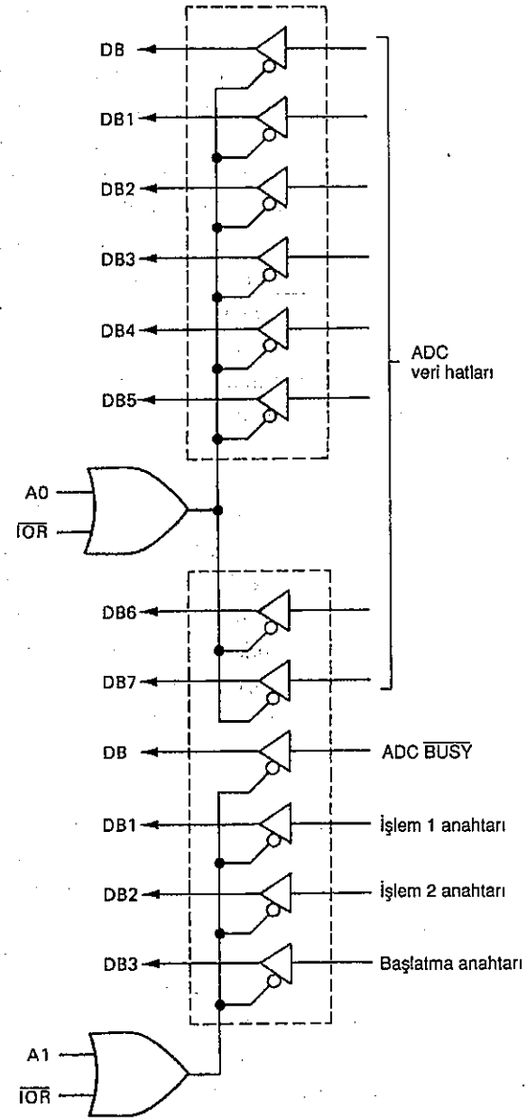
10.8 Bellek tasarımı

Normal koşullarda bellek devre düzenlemelerinin tasarımı, donanım tasarım çalışmalarının son aşamalarından biridir. Bunun nedeni, gerekli RAM ve ROM/PROM büyüklükleri konusunda gerçekçi bir tahmin yapılabilmesi için önce, yazılım tasarımının tamamlanma yolunda en azından belli bir aşamaya ulaşılmış olmasının gerektiğidir. Daha elverişli olması için bu noktada, yazılımın, ROM/PROM'un 2 Kbayttan daha az ve RAM'in en çok 256 bayt olarak belirlendiği bir aşamaya kadar geliştirildiğini varsayarak bellek tasarımını inceleyeceğiz.

Kısım 10.6'da anlatıldığı gibi EPROM, örneğimizdeki program kodunun saklanması için en elverişli bellek biçimidir. Dolayısıyla Ek N'de özellikleri belirtilen tek bir 2716 ile 2.048 x 8 bitlik EPROM'un seçilmesi doğaldır. RAM seçimi tanımlaması ise bu kadar açık değildir. 8080'in ortalama düzeydeki RAM'i ve otomatik tazeleme olanaklarına sahip olmaması, bizi, statik birimlerin seçimine yöneltmektedir. Kullanılabilecek statik RAM yongaları çok geniş bir çeşitliliğe sahiptir.

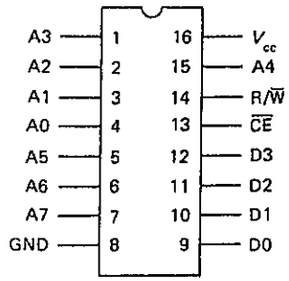


Şekil 10.10 Çıkış portu



Şekil 10.11 0 ve 1. giriş portları

Burada iki 2112 yongası kullanacağız. Bacak bağlantıları, Şekil 10.12'de verilmiştir. Basit yonga seçim denetimi, 16-bacaklı paketin düşük maliyeti ve yalnızca bir gerilim kaynağına gerek duyması, bu elemanın avantajları arasında yer almaktadır. 8080 yonga takımının kendisi de üç farklı gerilim beslemesi gerektirdiğinden, bu



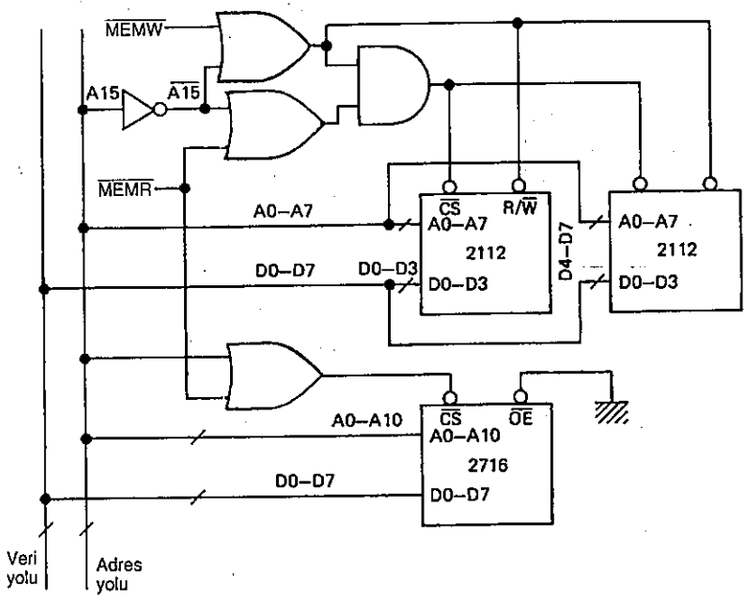
Şekil 10.12 Tip 2112, 256 x 4 statik RAM yongası bacak bağantıları

birimin gerilim kaynağı konusundaki üstünlüğünün buradaki sistemin tasarımı için çok fazla önemi bulunmamaktadır. Şimdi ROM ve RAM 'e adres ayırılması işlemine gelmiş bulunuyoruz. Yalnızca küçük boyutlu bellek içeren sistemlerde, yonga seçimi için adreslerin en büyük değerlikli bitlerinin tümünün kodlarının çözülemesi zorunluluğundan kaçınılabilir. Kısım 10.6'da belirttiğimiz gibi başlatma ve kesme yanıtı, EPROM'un, belleğin 0'dan başlayan adreslerde bulunmasını gerektirir. EPROM ile çakışmaması gereği dışında RAM adreslerinde bu tip kısıtlamalar söz konusu değildir. Dolayısıyla EPROM ve ROM bellek alanlarını biribirinden, en basit biçimde, toplam adresin tek bir bitiyle ayırabiliriz. En büyük değerlikli biti 0 olanlar, EPROM'a, diğer bütün adresler de RAM'e ayrılmıştır. Adresin, Şekil 10.13'te gösterildiği gibi, yalnızca 15 nolu biti yonga seçimlerinin yapılabilmesinde rol almaktadır. Tabii ki bu adreslerin birden fazla şekilde algılanmalarına yol açar, örneğin 8000, 8100, 8200 ,..., FF00 adreslerinin tamamı RAM'de aynı konuma erişecektir. Fakat bunun, programcının bellek düzenlemesini bilmesi koşuluyla, herhangi bir çakışmaya yol açmaması gerekir.

10.9 Kesme kullanımı

Kısım 10.5'te, iki işlemin gerektirdiği çeşitli zamanlama aralıklarını oluşturmak için sistem yazılımı tarafından kullanılan gerçek zaman saatinin sağlanması konusunu tartışmıştık. Yazılımın basitleştirilmesi için saatin kesmeler üretmesi kararı alınmıştı. Ancak böyle bir karar sonuç olarak sistem donanımına kesme kullanımı devre düzenlemesinin dahil edilmesini gerektirir.

Kısım 3.3'te 8080'in kesmeye nasıl yanıt verdiği anlatılmıştır. Şekil 10.14'te bu yanıta uyumlu bir devre gösterilmektedir. RTC (gerçek zamanlı saat) hattında bir alçaktan-yükseğe geçiş ortaya çıktığında, iki kararlı devreye "1" mandallanır. Bu mandallama işlemi özellikle, kesmenin yalnızca kısa bir zaman için mevcut olması durumunda önemlidir. Kısım 3.3'ten hatırlanacağı gibi, kesme hattı 8080 tarafından yalnızca her bir komut çevriminin son saat saykılı sırasında örneklenmektedir. Dolayısıyla INT hattındaki kısa bir darbe, bir örnek noktası ile

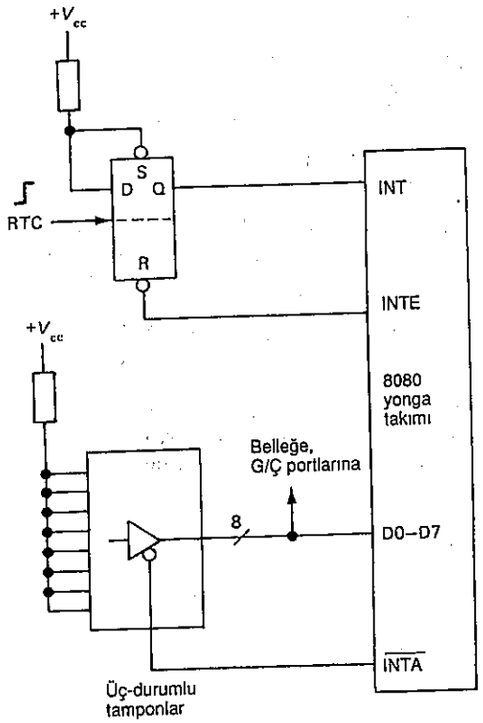


Şekil 10.13 Sistem RAM ve EPROM'u

S10.9 bir sonraki arasında ortaya çıkarsa, tesbit edilemeyebilir. Mandallama, ne kadar kısa olursa olsun, kesmenin her zaman tanınabilmesini sağlar.

8080 kesmeye, önce kesmeyi yetkisizleştirerek ve kesme yetkileme hattı INTE'yi, alçak düzeye çekerek yanıt verir. Bu, Ek A'daki Şekil 2.8'de gösterildiği gibi, M1 makine çevriminin T1 saat saykılı sırasında ortaya çıkar. T2 saat çevrimi sırasında durum bilgisi veri yolunda görünür ve Şekil 4.13'te gösterildiği gibi kesme alındı sinyali $\overline{INTA}$; 8080'in MİB yonga takımı tarafından üretilir. Bu noktada kesme yapan birim, veri yoluna yürütülecek bir sonraki komutun kodunu "sıkıştırmalı" ya da yerleştirmeye zorlamalıdır.

Şekil 10.14'te gösterilen devre INTE sinyalini, kesme iki-kararlı devresini sıfırlamak için kullanır. INTE alçak mantık düzeyinde olduğu sürece iki kararlı devre, bir başka kesmeye yanıt veremez. Bu süre içinde 8080 MİB de, kesmeye yanıt veremez. Kesmenin uygun noktada yeniden-yetkilenmesini, programa bir kesme yetkileme komutu olan EI komutunu dahil ederek sağlamak, programcının sorumluluğundadır. Bu komut 8080'in kesmelere yeniden yanıt vermesine olanak tanır ve INTE hattını yüksek mantık düzeyine çeker, böylece bir sonraki RTC alçaktan-yükseğe geçişinin "1"i, iki kararlıya mandallamasına izin verir.



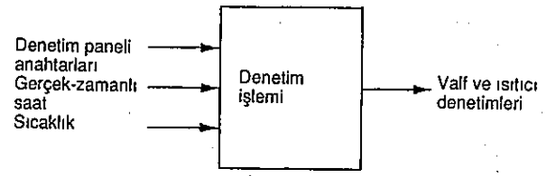
Şekil 10.14 Kesme devresi

Şekil 10.14'de, kesme alındı sinyalinin bir sonraki komut kodunu sıkıştıran üç-durumlu tamponları yetkilendirmek için kullanıldığı da gösterilmektedir. Burada kod, bir RST 38 komutuna uyan FF kodudur. Gerçek zaman saati kesme hizmet yordamının ilk komutu, bu nedenle 0038 adresinde yerleştirilmelidir. Bu yerleştirme, sistemde 2716 EPROM için ayrılan adres alanıyla uyumludur.

10.10 Yazılım tasarımı

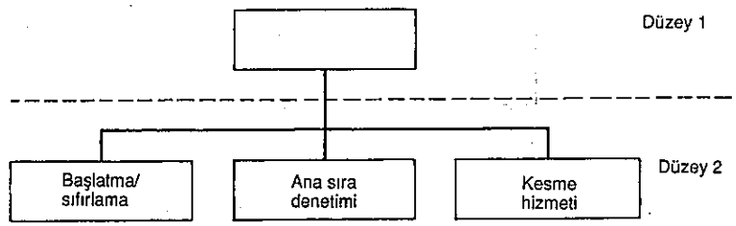
Geliştirme maliyeti açısından şimdi tasarım/geliştirme çalışmasının en ağırlıklı bölümüne gelmiş bulunmaktayız. Üretilecek serinin az sayıda olması nedeniyle yazılım geliştirmenin maliyete katkısı örneğimiz için özellikle önemlidir. Tasarım uygulamasının olabilecek en yüksek verimlilik düzeyinde kalmasını sağlamak için Bölüm 6'da anlatılan yukarıdan aşağıya yapılanma yaklaşımı kullanılacaktır.

Şekil 10.15'de, programın 1. düzey tanımlamasını gösterilmiştir. Başlatma/sıfırlama durumu ve kesmeleri düzenleyecek yordamların temel sıra denetim programın-



Şekil 10.15 1. Düzey özellikleri

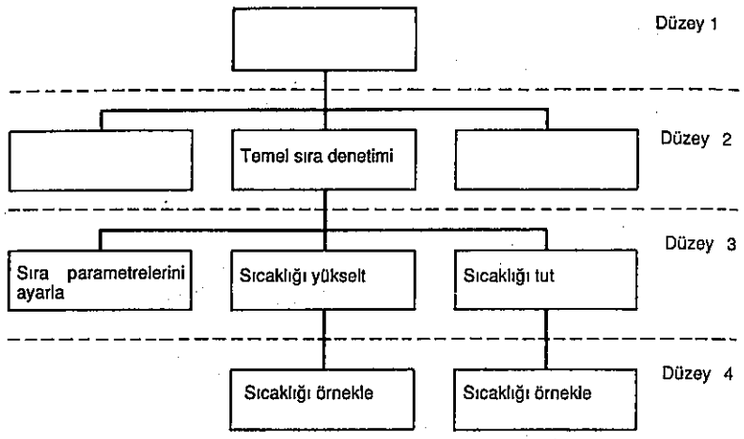
dan oldukça farklı olması gerektiğinden, ele alınan bu örnekte 2. düzey'ye bölümlenme, çok iyi şekilde tanımlanmıştır. Denetim, temel sıradan kesme hizmetine ya da sıfırlama hizmetine, yalnızca dıştan gelen bir olayın sonucu olarak, asenkron biçimde geçebilir. Dıştan gelen olay, DURDURMA anahtarına basılması ya da bir gerçek-zamanlı saat darbesinin ulaşmasıdır. Dolayısıyla hem kesme hem de sıfırlama hizmeti, iyi tanımlı 2. düzeyli program modüllerini oluştururlar. 2. Düzeye bölümlenme, Şekil 10.16'da gösterilmektedir.



Şekil 10.16 2. düzeye bölümlenme

Temel sıra denetimi, büyük farkla en karmaşık 2. düzey modülüdür ve hemen daima daha ileri düzeyde bölümlenmeyi gerektirir. İlk olarak, bir sıra denetim programı kullanarak iki farklı uygulamanın gerçekleştirildiği görülmektedir. Dolayısıyla programa başlangıçta uygun zaman ve sıcaklık parametrelerinin verilmesi gereklidir. Bu ise denetim anahtarlarının taranması ve 1. ve 2. işlem anahtarlarından hangisine basılacağının saptanması için bir 3. düzey modülünü gerektirmektedir. Daha sonra uygun süre ve sıcaklık parametreleri yeniden ana düzey 2 modülüne aktarılır.

İşlem sırasının bir bölümü olarak zaman zaman sıcaklığın belirli düzeylere yükseltilmesi gerekir. Bu işlevi gerçekleştiren programa 3. düzey modülü adını veririz. Benzer şekilde, sıcaklığı, verilen bir düzeyde tutan program, bir başka 3. düzey modülünü oluşturmaktadır. Verilen hedef sıcaklığa kadar ısıtma veya sıcaklığı



Şekil 10.17 4. düzeye bölümlenme

korumada, C kabındaki gerçek sıcaklığın örneklenmesi gerekmektedir. Bunun için bir AD dönüştürmesi ve çıkan veri baytının alınması söz konusu olacaktır. Bu işlemi gerçekleştirmek için bir 4. düzey modülü tanımlanmaktadır.

Şekil 10.17'de gösterilen bölümlenme diyagramının asla değiştirilemeyeceği düşünülmemelidir. Program tasarımı ilerledikçe daha ileri düzeyde bölümlenmeye gerek olması oldukça muhtemeldir. Böyle bir gereksinim doğduğunda, program tekrar bölümlenmelidir, çünkü bölümlenme, program yapısının değerli bir kaydını oluşturmaktadır.

10.11 Modüller arasında veri aktarımı

Bölümlenme işlemini gerekli ölçüde gerçekleştirdikten sonra program modülleri arasında verinin nasıl taşınacağını belirlemek yararlı bir alıştırma olacaktır. Assembly dili programlarda en yaygın veri aktarım yöntemi, belirli kaydediciler veya bellek konumları kullanmaktır. Örneğimizde bu yöntemlerin ikisini de kullanacağız.

Şekil 10.17'de gösterilmekte olan bölümlenme diyagramında tanımlanan modüller arasında gerçekleşecek veri aktarımlarına göz atalım. Oldukça belirgin olan bir aktarım, sıcaklık ölçümünün, 4. düzey modülünden sıcaklık-denetleyici 3. düzey modüllerinin birine aktarılmasıdır. Burada, 8-bitlik sıcaklık ölçümünün, A kaydedicisi aracılığıyla aktarılacağını belirtmekteyiz. Bu, "sıcaklık örnekleme" modülün-

den çıkışta, sıcaklık ölçümünün A kaydedicisinde bulunması gerektiği anlamına gelmektedir.

Sıcaklık ve zamanlama aralıkları ile ilgili sayılar da, sıvıyı belli bir sıcaklığa kadar ısıtan ve belli bir periyot süresince bir sıcaklıkta tutan 3. düzey modüllerine aktarılmalıdır. Buradaki sıcaklığın B kaydedicisi, ve periyodun da C kaydedicisi aracılığıyla aktarılacağını belirtmekteyiz.

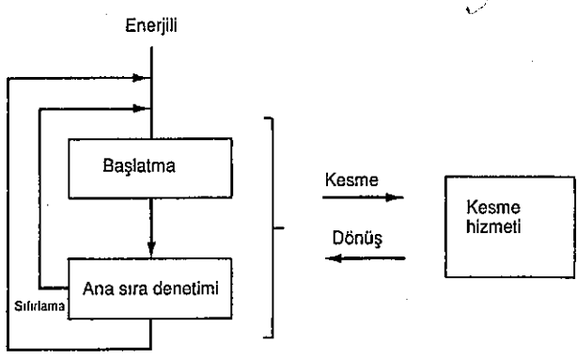
Bir başka ana veri aktarımı ise işlem süresi ve sıcaklık parametrelerinin üretildikleri modülden tekrar temel sıra denetim modülüne aktarılmalarıdır. Aktarımı söz konusu olan dört parametre bulunmaktadır: ikişer adet süre ve iki sıcaklık değerleri. Süre değerleri 30-60 dakika arasında, sıcaklık değerleri ise 40-80°C arasında yer almaktadır. Dolayısıyla her bir parametre için tek bir bayt yeterli olacaktır, aktarılacak bayt sayısı da böylelikle 4 olmaktadır. Bayt sayısının bu düzeyde olması, özel bellek konumlarının kullanılmasını gerektirmektedir ve bundan dolayı biz de sıcaklıkları T1 ve T2, , zaman aralıklarını ise P1 ve P2 sembolik adreslerinde saklamaktayız.

Program içinde verinin gösterimi ve aktarımı ile ilgili son bir nokta ise, denetim işleminin bir parçası olarak, çıkış portunun bazı hatlarını, diğerlerini etkilemeksizin değiştirmek gerektiği olduğunu görmekteyiz. Valf, röle ve benzerleri için denetimin gerekli olduğu durumlarda, bu durumla sık sık karşılaşılır. Bunu gerçekleştirmenin basit bir yolu, her bir çıkış portu hattının yürürlükteki durumunun kaydını içeren bir bellek konumunun korunmasıdır. Bu konuma sembolik olarak OPSTAT adını vermekteyiz. Eğer OPSTAT, 3 değerini içeriyorsa, bu, çıkış portunun 0 ve 1 hatlarının, diğer tüm hatlar gibi 1'de olduğunu göstermektedir.

Herhangi bir çıkış hattının durumunu değiştirmek için mantıksal VE veya VEYA işlemlerinin kullanılması gerekir. Port 0'ın 2 nolu hattını diğer hatları değiştirmeden, mantıksal 1 durumuna getirmek istediğimizi varsayalım. Bu durumda, OPSTAT ve 02'nin içerikleri arasında bir mantıksal VEYA işlemi gerçekleştirilir. 2 nolu hattı mantıksal 0 durumuna getirmek için ise OPSTAT ve FB içerikleri arasında bir mantıksal VE işlemi gerçekleştirilmelidir. Her bir durumda sonuç baytı daha sonra porta çıkış yapılır ve aynı zamanda OPSTAT'ın yürürlükteki durumu olarak saklanır.

10.12 Modüler yapı

Yeniden herbir modül yapısının tasarımına dönersek, yukarıdan aşağıya yaklaşımın normal seyrinde olduğu gibi 1. düzey modül ile başlarız. Burada yapı hemen hemen



Şekil 10.18 Ana program akışı

önemsiz denecek kadar çok basittir. Ancak, düzey 1 yapısında kesme hizmeti bulunmamakta ise de, kesme ve sıfırlama hizmet yordamlarının sistemin nasıl çalışacağını belirlemede önemli oldukları anımsanmalıdır. Karşı karşıya bulunduğumuz bu durum, yüksek-düzeyli dildeki program yapılarına ilişkin standart tanımlama yöntemlerinin, assembly dili programlama için tam anlamıyla yeterli olmadığı bir durumdur. Şekil 10.18'de, 1. düzey modülü ile kesme/sıfırlama eylemi arasındaki ilişki gösterilmeye çalışılmıştır. Kurallara uygun modül yapısı Şekil 10.19'da gösterilmektedir. 1. düzey modülü için assembly dili kaynak kodu, Şekil 10.20'de yer almaktadır. Başlatma yordamı çağrılmadan önce yığın işaretçisinin kurulması ve kesmelerin yetkisizlenmesinin gerekli olduğuna dikkat ediniz. Bunun nedeni hem CALL komutunun ve hem de bir kesme yanıtının yığından yararlanmalarıdır. Kesmelerin yetkisizleme işlemi, 8080, işlemleri, kesmeleri yetkisizleme ile başlatacağından, bir ihtiyaçtan çok bir önlem durumundadır.

```
KESMEYİ YETKİSİZLE
YIĞIN İŞARETÇİ'SİNİ KUR
DO SONSUZ
        BAŞLATMA SIRASI
        ANA SIRA
END DO
```

Şekil 10.19 1. düzey modül yapısı

2. düzey modüllerin yapıları, Şekil 10.5, 10.21 ve 10.22'de gösterilmektedir. Bunların hepsi bu aşamada kodlanabilir ve en azından kısmen test edilebilirler. Temel sıra denetim ve başlatma modüllerinin tam anlamıyla test edilmesi, yalnız sistem G/Ç donanımı ile birlikte mümkündür. 1 saniyeli gerçek-zamanlı saat darbelerinin sayımı için kullanılan yöntem, Şekil 10.22'de, verilmektedir. Yer darlığı nedeniyle bü-


```
..... 8080/85 ASSEMBLER V01 ..............................  PAGE  1
                                                            FUTUREDATA  .........

        * ****************************************** *
        * KİMYASAL İŞLEM DENETLEYİCİSİ                 *
        * DÜZEY 1 MODÜL                                *
        *                                              *
        *                          YAPI                *
        *                          ****                *
        * KESMEYİ YETKİSİZLE                           *
        * YIĞIN İŞARETÇİSİNİ RAM'IN ÜSTÜNÜ GÖSTERECEK ŞEKİLDE KUR *
        * DO  SONSUZ                                   *
        *       BAŞLATMA İŞLEMİNİ YÜRÜT                *
        *       ANA SIRA DENETİM İŞLEVİNİ YÜRÜT        *
        * END DO                                       *
        * ****************************************** *

0000                        ASEG    TECT3
                            GLBL    STRTUP, MAIN        GLOBAL ADLAR 0 DA OLMALI
0000                        ORG     0                   KESMEYİ YETKİSİZLE
                    *
0000  F3            INIT    DI                          YIĞIN İŞ.'Nİ RAM'İN BAŞINA KUR
0001  31FF83                LXI     SP, X'83FF'         BAŞLATMA YORDAMI
0004  CD0000                CALL    STRTUP              ANA SIRA DENETİMİ
0007  CD0000                CALL    MAIN                SONSUZ DÖNGÜ
000A  F20000                JP      INIT
                    *
000D                        END
```

Şekil 10.20 Düzey 1 modül program kodu

```
KESMEYİ YETKİSİZLE
ISITICIYI KAPALI
A VALFİNİ KAPA
C VALFİNİ KAPA
DO UNTIL START = 1
   START'I TARA
END DO
```

Şekil 10.21 Başlatma/sıfırlama yordamı

```
ÇALIŞMA KAYDEDİCİLERİNİ SAKLA
SANY = SANY + 1
IF SANY = 60
   THEN
        SANY = 0
        DAK = DAK + 1
   ELSE
        DEVAM ET
END IF
ÇALIŞMA KAYDEDİCİLERİNİ GERİYE
AL
KESMEYİ YETKİLE
KESMEDEN GERİYE DÖN
```

Şekil 10.22 Kesme hizmet yordamı

tün modüllerin assembly listesi verilmeyecektir. Ancak her düzeyden liste örnekleri bulunacaktır. 10.23-10.26 nolu Şekiller sırasıyla başlatma/sıfırlama, kesme hizmeti ve temel sıra denetim yordamlarının listelerini göstermektedir.

Aşağı doğru kademe kademe inelim ve sıvıyı TX sıcaklığına kadar ısıtan modülü tanımlayan sözde-kodların gösterildiği Şekil 10.25'i ele alalım. Yapı yüzeysel olarak bakıldığında tatmin edici görünmektedir fakat Kısım 8.2'de tartışılan sağlamlık özelliğinden yoksundur. Bu özellik, programın beklenmeyen sorunların üstesinden gelme yeteneğinin bir ölçüsüdür. Şekil 10.27'deki yapıyı uygulamaya koyduğumuzu ve sıcaklığın bir şekilde TX'i aştığını düşünelim. Program, T=TX koşulunun sağlandığını hiç bir zaman tesbit edemeyeceğinden, ısıtıcı açık kalacak ve sıvı giderek daha çok ısınacaktır. Şekil 10.26'da, işletimin güvenliği için ek sağlamlık sağlayan alternatif bir yapı gösterilmektedir. Geriye kalan 3. düzey modüllerinin sözdekodu Şekil 10.29 ve 10.30'da verilmektedir.

```
                                                        PAGE  1
..... 8080/85 ASSEMBLER V01 ..............................  FUTUREDATA  .........

                  * ****************************************  *
                  *                                          *
                  * KİMYASAL İŞLEM DENETLEYİCİSİ               *
                  * BAŞLATMA/SIFIRLAMA YORDAMI                 *
                  *                                          *
                  *                          YAPI            *
                  *                          ****            *
                  *                                          *
                  * KESMEYİ YETKİSİZLE                         *
                  * ISITICI KAPALI                             *
                  * A VALFİNİ KAPAT                            *
                  * C VALFİNİ KAPAT                            *
                  * DO UNTIL START = 1                         *
                  *      START! TARA                           *
                  * END DO                                     *
                  * ****************************************  *

0000                          RSEG        TECT1
                              GLBL        OPSTAT              GLOBAL SEMBOL
                  *
0000  F3          STRTUP      DI                              KESMEYİ YETKİSİZLE
0001  3E00                    MVI         A, 0
0003  320000                  STA         OPSTAT              ÇIKIŞ PORTU DURUMUNU HAZIRLA
0006  D300                    OUT         0                   ISITICI KAPALI
                  *                                           A, B, C VALFLERİNİ KAPAT
0008  DB01        STRTU1      IN          FD                  DENETİM PANELİNİ TARA
000A  E608                    ANI         X'08'               START = 1 Mİ?
000C  CA0800                  JZ          STRTU1              HAYIRSA ATLA
000F  C9                      RET
                  *
0010                          END
```

Şekil 10.23 Başlatma program kodu

10.13 Prototip donanım yapımı, test ve hata ayıklama

10.6-10.9 nolu kısımlarda anlatıldığı şekilde, donanım tasarımı sonuçlandırıldığında, tel-sargılı bir prototip yapılabilir. Bölüm 9'da anlatılan adım-adım yöntemlerinin kullanımı avantaj sağlayacaktır. İlk önce entegre devre soketlerine, uygun noktalara ayırma kondansatörleri konulmak kaydıyla, güç kaynağı çıkış kablolarına bağlanır. MİB yonga takımı; 8080 MİB, 8228 yol denetleyicisi ve uygun bir kristalle birlikte 8224 saat üretecinden oluşur. Bu yonga takımı Şekil 4.13'te gösterildiği gibi bağlanır. Yonga takımının test edilmesi, Kısım 9.3'te anlatılmıştır.

Yapımın bir sonraki aşamasında, sisteme RAM sağlamak için, Şekil 10.13'te gösterildiği gibi iki adet 2112 bellek yongasının kablolanması gerçekleştirilir. Bu, daha sonra, Kısım 9.3'te anlatıldığı gibi test edilir. 2716 EPROM benzer şekilde kablolanır ve bilinen veri ile programlanmış olan bir test EPROM'undan veri okumak için ICE kullanılarak test edilir. Bu noktadan sonra basit test programları için kodlar 2716'da programlanabilir ve ICE denetiminde yürütülür. Giriş ve çıkış portları, sayısal veri kaynakları ve gösterge olarak lamba ve anahtarlar kullanılarak tek tek kablolanıp test edilebilir. ICE probu bağlı olarak MDS'de çalıştırılan test programları, bu amaç için kullanılmaktadır.


```
                                                        PAGE  1
..... 8080/85 ASSEMBLER V01 ..............................  FUTUREDATA  .........

                  * ****************************************  *
                  *                                          *
                  * KİMYASAL İŞLEM DENETLEYİCİSİ               *
                  * KESME HİZMET YORDAMI                       *
                  *                                          *
                  *                          YAPI            *
                  *                          ****            *
                  *                                          *
                  * ÇALIŞMA KAYDEDİCİLERİNİ SAKLA              *
                  * SANY = SANY + 1                            *
                  * IF SANY= 60                                *
                  *      THEN                                  *
                  *           SANY = 0                         *
                  *           DAKK = DAKK + 1                  *
                  *      ELSE                                  *
                  *           CONTINUE                         *
                  * END IF                                     *
                  * KESMEYİ YETKİLE                            *
                  * DÖN                                        *
                  * ****************************************  *

0000                          ASEG        TECT2
                              GLB         SANY, DAKK          GLOBAL ADLAR
0000                          ORG         X'0038'
                  *
                  * KAYDEDİCİLERİ SAKLA*
0036  F5                      PUSH        PSW
0039  E5                      PUSH        H
003A  D5                      PUSH        D
003B  C5                      PUSH        B
                  * KAYDEDİCİLER SAKLANDI *
                  *
003C  3A0000                  LDA         SANY                SANİYELERİ AL
003F  3C                      INR         A                   SANY= SANY + 1
0040  320000                  STA         SANY                SANİYELERİ SAKLA
                  * IF-THEN-ELSE YAPISININ BAŞLANGICI*
0043  FE3C                    CPI         60                  SANY = 60 Mİ?
0045  C25400                  JNZ         INT1                HAYIRSA ATLA
                  * THEN
0048  3E00                    MVI         A, 0                SANİYELER = 0
004A  320000                  STA         SANY                DAKİKALARI AL
004D  3A0000                  LDA         DAKK                DAKK = DAKK + 1
0050  3C                      INR         A                   DAKİKALARI SAKLA
0051  320000                  STA         DAKK
                  * END IF
                  *
                  * KAYDEDİCİLERİN GERİYE ALINMASI *
0054  C1          INT1        POP         B
0055  D1                      POP         D
0056  E1                      POP         H
0057  F1                      POP         PSW
                  * KAYDEDİCİLER GERİYE ALINDI *
                  *
0058  FB                      EI          KESMEYİ YETKİLE
0059  C9                      RET
                  *
005A                          END
```

Şekil 10.24

```
                                                                    PAGE   1
----- 8080/85 ASSEMBLER V01 ------------------------------- FUTUREDATA ---------

                 * ***************************************** *
                 *  KİMYASAL İŞLEM DENETİMİ                  *
                 *  ANA SIRA DENETİMİ                        *
                 *                                           *
                 *                    YAPI                   *
                 *                    ****                   *
                 *                                           *
                 *  SIRA PARAMETRELERİNİ KUR                 *
                 *  A VALFİNİ AÇ                             *
                 *  T1'E KADAR ISIT                          *
                 *  P1 DAKİKA SÜREYLE T1'DE TUT              *
                 *  B VALFİNİ AÇ                             *
                 *  T2'YE KADAR ISIT                         *
                 *  P2 DAKİKA SÜREYLE T2'DE TUT              *
                 *  C VALFİNİ AÇ                             *
                 *                                           *
                 *  DAKK = 0                                 *
                 *  SANY = 0                                 *
                 *  KESMEYİ YETKİLE                          *
                 *  DO UNTIL DAKK = 3                        *
                 *      DAKK'I  TARA                         *
                 *  END DO                                   *
                 *  KESMEYİ YETKİSİZLE                       *
                 *  ***************************************** *

0000                        RSEG        TECT5
                            GLBL        ANA, ISIT, TUT, DAKK, SANY, SETSEQ, OPSTAT
                            GLBL        T1, T2, P1, P2
                 *
0000   CD00000   ANA        CALL        SETSEQ              SIRA PARAMETRELERİNİ KUR
                 *
                 A VALFİNİ AÇ*
0003   3A0000               LDA         OPSTAT              ÇIKIŞ PORTUNUN DURUMUNU AL
0006   F601                 ORI         X'01'               BIT 0 = 1
0008   D300                 OUT         0                   A VALFİ AÇIK
000A   320000               STA         OPSTAT              PORTUN DURUMUNU SAKLA
                 *
                 * T1'E KADAR ISIT*
                 * NOT: T1, B KAYDEDİCİSİ ARACILIĞIYLA ISITMA YORDAMINA GEÇTİ
000D   3A0000               LDA         T1
0010   47                   MOV         B, A                B'DEKİ T1'İ AL
0011   CD0000               CALL        ISIT
                 *
                 * P1 SÜRESİNECE T1'DE TUT*
                 * NOT:   T1, B KAYDEDİCİSİ ARACILIĞIYLA TUTMA YORDAMINA GEÇTİ
                         P1, C KAYDEDİCİSİ ARACILIĞIYLA TUTMA YORDAMINA GEÇTİ
0014   3A0000               LDA         T1                  B'DEKİ T1'İ AL
0017   47                   MOV         B, A
0018   3A0000               LDA         P1                  C'DEKİ P1'İ AL
001B   4F                   MOV         C, A
001C   CD0000               CALL        ISIT
                 *
                 * B VALFİNİ AÇ*
001F   3A0000               LDA         OPSTAT              ÇIKIŞ PORTUNUN DURUMUNU AL
0022   F602                 ORI         X'02'               BIT 1 = 1
0024   D300                 OUT         0                   B VALFİ AÇIK
0026   320000               STA         OPSTAT              PORTUN DURUMUNU SAKLA
                 *
                 * T2'YE KADAR ISIT*
```

Şekil 10.25


```
                                                                    PAGE   1
----- 8080/85 ASSEMBLER V01 ------------------------------- FUTUREDATA ---------

                 * NOT:   T2 B KAYDEDİCİSİ
                 ARACILIĞIYLA ISITMA YORDAMINA
                 GEÇTİ
0029   3A0000               LDA         T2
002C   47                   MOV         B, A                B'DEKİ T2'Yİ AL
002D   CD0000               CALL        ISIT
                 *
                 * P2 DAKİKA SÜREYLE T2'DE TUT*
                 * NOT:   T2 B KAYDEDİCİSİ ARACILIĞIYLA TUTMA YORDAMINA GEÇTİ
                         P2 C KAYDEDİCİSİ ARACILIĞIYLA TUTMA YORDAMINA GEÇTİ
0030   3A0000               LDA         T2
0033   47                   MOV         B, A                B'DEKİ T2'Yİ AL
0034   3A0000               LDA         P2
0037   4F                   MOV         C, A                C'DEKİ P2'Yİ AL
0038   CD0000               CALL        TUT
                 *
                 * C VALFİNİ AÇ*
003B   3A0000               LDA         OPSTAT              ÇIKIŞ PORTUNUN DURUMUNU AL
003E   F604                 ORI         X'04'               BIT 2 = 0
0040   D300                 OUT         0                   C VALFİ AÇIK
0042   320000               STA         OPSTAT              PORT UN DURUMUNU SAKLA
                 *
                 * 3 DAKİKA BEKLE*
0045   3E00                 MVI         A, 0
0047   320000               STA         DAKK                DAKK = 0
004A   320000               STA         SANY                SANY = 0
004D   FB                   EI                              KESMEYİ YETKİLE
                 * DO UNTIL DAKK = 3
004E   3A0000   TARA        LDA         DAKK                DAKK'I TARA
0051   FE03                 CPI         3                   DAKK = 3?
0053   C24E00               JNZ         TARA                HAYIRSA ATLA
                 * END DO
                 *
0056   F3                   DI                              KESMEYİ YETKİSİZLE
0057   C9                   RET
                 *
0058                        END
```

Şekil 10.26

```
ISITICI AÇIK
DO UNTIL T = TX
   T ZAMANINI ÖRNEKLE
END DO
```

Şekil 10.27 DO UNTIL yapısı kullanarak TX değerine kadar ısıtma

```
T ZAMANINI ÖRNEKLE
DO WHILE L ≤ X
   ISITICI AÇIK
   T ZAMANINI ÖRNEKLE
END DO
```

Şekil 10.28 DO WHILE yapısı kullanarak TX değerine kadar ısıtma

```
SANY = 0
DAKK = 0
KESMEYİ YETKİLE
DO UNTIL DAKK = PX
   T ZAMANINI ÖRNEKLE
   IF   T ≤ TX
        THEN
             ISITICI  AÇIK
        ELSE
             IF T ≥ TX
                THEN
                     ISITICI KAPALI
                ELSE
                     DEVAM ET
        END IF
   END IF
END DO
KESMEYİ YETKİSİZLE
: NOT:  PX GİRİŞTE C KAYDEDİCİSİNDE
        TX GİRİŞTE B KAYDEDİCİSİNDE
```

Şekil 10.29 PX dakika süreyle TX'te tut

```
ISLEM 1 DENETİM ANAHTARINI TARA
IF ISLEM 1 = 1
    THEN
        P1 = 30
        P2 = 40
        T1 = 60
        T2 = 80
    ELSE
        P1 = 50
        P2 = 60
        T1 = 40
        T2 = 70
END IF
```

Şekil 10.30 İşlem sırası parametrelerinin belirlenmesi

ADC ile ilişkili G/Ç portlarının test edilmesinin ardından, devreye ADC'nin kendisinin de bağlanması gerekir. Test amacıyla giriş olarak, değişken bir gerilim kaynağı kullanılabilir. Burada da yine, MDS tarafından yürütülen bir test programı çalıştırılır. Bu test programı, Şekil 10.6'da gösterildiği gibi, bir "dönüştürmeyi başlat" darbesi (STC) üreterek bir AD dönüştürmesi başlatmalıdır. Bu darbe, Şekil 10.9'da gösterildiği gibi, çıkış portunun 4 nolu hattında ortaya çıkmalıdır. Daha sonra B giriş portunun 0 hattında, dönüştürmenin tamamlandığına gösteren bir yüksek mantık düzeyi saptanıncaya kadar, meşgul sinyali taranır. Bunun ardından B portu aracılığıyla bir veri baytı alınır. Test programını sözde-kod biçimi, Şekil 10.31'de anlatılmaktadır.

```
STC'ye 1 ÇIKIŞI
STC'ye 0 ÇIKIŞI
STC'ye 1 ÇIKIŞI
DO UNTIL MEŞGUL = 1
   GİRİŞ MEŞGUL
END DO
```

Şekil 10.31 ADC ve giriş portlarına ilişkin test programı

Donanım geliştirmenin daha sonraki aşamasında, Şekil 10.4'te gösterilen kesme devresinin kablolanması ve test edilmesi yer almaktadır. Test amacıyla, bu devrede kesme başlatmak üzere gereken 0'dan 1'e geçişi için, bir anahtar kullanılmaktadır. MDS'de herhangi bir uygun konumda çalışacak olan test programı aşağıda verilen iki komuttan oluşmaktadır:

```
INTEST   NOP
         JP     INTEST
```

Bu program tabii ki bir sonsuz döngüdür. 8080'in kesme alındısı vermesi sırasında Şekil 10.14'te yer alan devrenin RF

kodunu veri yoluna sıkıştırdığını hatırlayınız. Bu bir yeniden başlama komutudur ve 0038 adresine bir atlamanın gerçekleşmesine neden olur. Bu nedenle kesme devresini test etmek için 0038'e bir durdurma noktası yerleştiririz ve yukarıda gösterilen programı yürütürüz. Bir kesme oluşturan test anahtarına basıldığında, 0038'e atlama gerçekleşir ve program durdurma noktası etkinleşir.

Şekil 10.7'de gösterilen gerçek-zamanlı saat devresinin kablo bağlantısı yapıldıktan sonra bu devre, a.c. şebeke girişi uygulanarak ve kırpma devresinin çıkışı bir osiloskopla denenerek test edilebilir. Bölücünün çıkışının denetlenmesi için bir darbe sayıcı daha elverişlidir fakat bu osiloskop kullanarak da yapılabilir.

10.14 Son test

Prototip yapımı ve testinin tamamlanmasından sonra geliştirmenin son aşaması gelir. Bu aşaması geliştirilen programlar donanım prototipi ile bir arada test edilir. Burada ele alınan sistem için başlangıç testi denetim paneli ve valfler bağlı iken gerçekleştirilebilir fakat ısıtıcı için yük simülasyonu ve sıcaklık dönüştürücüsü simülasyonu için gerilim kaynağı gerekir. Kısım 9.4'te anlatıldığı gibi MDS ICE olanağı, sistem programlarının tamamen monitör denetimi altında çalıştırılmasına olanak tanıdığından, bu test işleminde önemli bir rol oynamaktadır.

Başlangıç sistem testinin yeterli başarıyı sağladığını varsayarsak, gerçek ısıtıcı ve sıcaklık dönüştürücüsü bağlı olarak bir başka test daha yapılabilir. Bu durumda A ve B kaplarında sıvı bulunmalıdır. Bütün donanım/yazılım sistemi ilk kez tek bir birim olarak işletildiğinden bu aşama, test uygulamalarında kritik bir aşamadır. Bu test sırasında süre ve sıcaklık değerlerinin hassaslığı kontrol edilebilir. Son olarak program MDS RAM'inden EPROM'a aktarılır. Bu aktarma tek bir işlem olarak gerçekleştirilebileceği gibi, Kısım 7.5'te anlatıldığı şekilde adım adım olarak da yapılabilir.

Sorular

10.1 Tipik bir mikroişlemci sistemi için tasarım/geliştirme uygulamasının çeşitli aşamalarını sıralayınız.

10.2 Bu konuda anlatılan örnek çalışmasında 8080 mikroişlemcisinin seçilmesinin iki ana nedeni nedir?

10.3 Saatin basitçe bir giriş portu aracılığıyla taranması yerine kesme kaynağı olarak gerçek-zamanlı saatinin kullanılması konusunda hangi gerekçeler verilmiştir?

10.4 Sistem tarafından gereksinim duyulan zaman aralıklarının üretilmesi için dış gerçek-zamanlı saat kullanımının, program zamanlama döngüsüne tercih edilmesinin nedeni nedir?

10.5 Kullanılabilir gerçek-zamanlı saat darbe frekanslarının alt sınırını oluşturan nedir?

10.6 "Açılışta-sıfırlama"nın nasıl uygulandığını açıklayınız.

10.7 Özel bir görev için ayrılmış mikroişlemci sisteminde RAM'in işlevleri nelerdir?

10.8 8080 mikroişlemcisinde bir sıfırlama işleminden sonra yürütülecek ilk komutun adresini söyleyiniz.

10.9 8080 kesme arabiriminde kesme kaynağı, çok geniş ya da çok dar darbeler üretmekteyse, bir mandal devresi kullanmanın neden önemli olduğunu açıklayınız.

10.10 Şekil 10.29 ve 10.30'da anlatılan program modüllerini tanışık olduğunuz herhangi bir mikroişlemcinin komut takımını ve bulabildiğiniz bir çeviriciyi kullanarak kodlayınız

Bölüm 11 Programlanır Belleğin Pratik Yönleri

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde ve önerilen alıştırmaları yaptığınızda;*

1. *Programlanır bellek birimleri arasında seçim yapabilmek için imalatçı bilgi sayfalarını kullanabilmeli,*
2. *Programlanır bellek birimlerinin çalışma koşullarını tanımlamak için imalatçı bilgi sayfalarını kullanabilmeli,*
3. *Bir geliştirme programını saklamak için EPROM'u kullanabilmeli,*
4. *Programlanır bir bellek birimini programlayabilmeli,*
5. *EPROM'un silinmesi için yapılması gereken işlemleri söyleyebilmelisiniz.*

11.1 Mikroişlemci sistem geliştirmede EPROM'un rolü

Buraya kadar kitabın çeşitli yerlerinde EPROM'un donanım/yazılım geliştirme çalışmasındaki belirgin rolü vurgulanmıştır. Kuşkusuz Bölüm 7'de EPROM'un bulunmasının, bir programın host sistem RAM'ından hedef sistem EPROM'una, hedef sistem tam anlamıyla bağımsız şekle gelinceye kadar, modül-modül aktarılmasına nasıl olanak tanıdığı ayrıntılı olarak anlatılmıştır.

Mikroişlemci sistemlerin geliştirilmesinde EPROM'un çok yararlı olması nedeniyle, kapsamında bu kitabın da bulunduğu TEC standart kitap U80/674, öğrencinin EPROM'un belli kullanım özellikleri hakkında pratik deneyim sahibi olmasını gerektirmektedir. Bu bölümde günümüzde kullanımda bulunan çeşitli EPROM tipleri anlatılmakta ve bunların okuyucu tarafından yararlanılabilecek özellikler ile ilgili pratik alıştırmalar önerilmektedir. Okuyucunun bu alıştırmaları tam olarak belirtilen şekilde uygulayabileceği, elbette kullanabileceği olanaklarla belirlenecektir. Diğer bölümlerde önerilen alıştırmalarda olduğu gibi, okuyucunun, assembly dilini destekleyen en azından bir yazılım geliştirme istasyonuna erişim fırsatının bulunduğu varsayılmıştır. Buna ek olarak, bu bölümdeki alıştırmaları tamamlamak için, geliştirme istasyonunun RAM'ındaki amaç programların EPROM'a aktarılmasına olanak veren bir EPROM programlayıcısına gerek bulunmaktadır. Bu çeşit birimler aynı zamanda, verinin programlanmış bir EPROM'dan okunmasına olanak tanırlar. Şekil 11.1'de uygun bir sistem gösterilmektedir.

Şekil 11.1 EPROM programlayıcı içeren bir yazılım geliştirme istasyonu

11.2 Programlanır bellek birimlerinin seçimi

Kalıcı programlanır bellek içeren bir mikroişlemci sistemi geliştirirken, planlama kararlarından birisi de, kullanılacak bellek birimlerinin sayı ve tipleridir. Bu kısımda bu seçimi etkileyen etkenleri ele almaktayız. Herşeyden önce, kullanmak istediğimiz birim tipi konusunda yeterince aydınlanmış olmalıyız. Birçok mikroişlemci sisteminde, birden fazla tipte kalıcı bellek kullanılır. Manyetik yüzeyli saklama ortamları, yani sabit disk ya da disket, bant veya tambur biçiminde saklama ortamları, genel-amaçlı mikroişlemci sistemlerindeki yedek saklama ortamları olarak yaygın şekilde kullanılır. Bizim ele aldığımız bellek tipi bu değildir. Bu bölümde RAM'de olduğu biçimde, bellek adres alanının bir kısmını kaplayan yüksek-hızlı, kalıcı, monolitik bellek konusuyla ilgilenmekteyiz. Bu bellek program kodunu ve sürekli veriyi saklar, ve yaygın olarak EPROM şeklinde adlandırılır.

S11.7

EPROM teriminin anlamı "silinebilir programlanır sadece-okunur bellek"tir. Dolayısıyla EPROM'lar veri ya da program kodunun saklanabildiği birimlerdir.

Veri ya da program kodunun saklanma işlemi EPROM'un programlanması olarak adlandırılır. Adından da anlaşılabildiği gibi bu birimler, içerikleri silinebilir olan birimlerdir yani saklanan veriler, bazen EPROM'un "yıkanması" olarak tanımlanan bir işlemle, silinebilirler. Yıkama sonrasında EPROM'a yeni veriler yüklenebilir. Programlanır bir bellek birimi seçerken saklanacak verinin miktarı, dikkate alınması gereken önemli bir unsurdur. Eğer EPROM, program kodu saklayacaksa toplam bayt sayısı, programın geliştirilmesi tamamlanıncaya kadar tam olarak bilinemeyebilir. Bazı durumlarda bu, donanım tasarım kararlarını geciktirebilecektir. Ancak deneyimli bir programcı programın yaklaşık olarak büyüklüğünü, tasarım uygulamasının oldukça erken bir döneminde genellikle tahmin edebilir. Dolayısıyla donanım tasarımcısının, ortaya çıkabilecek hata sayısının yüksek olmasını çok önemli bulmaması koşuluyla, bu sorun donanım tasarımını geciktirmez. EPROM'ların daha popüler olan tipleri düşük-maliyetli birimlerdir, bundan dolayı gerekenden fazla sayıda satın alınmaları, normalde önemli bir kısıtlama oluşturmaz.

EPROM seçimini etkileyen diğer etkenler, başlıca, çalışma özellikleri etkenlerdir. Bunlardan en önemlileri aşağıda sıralanmıştır. Yukarıda incelenmiş olmasına karşın, bütünlük sağlaması için bu listede, bellek büyüklüğü konusuna da yer verilmiştir.

EPROM seçimini etkileyen etkenler

1 *Gereken bellek büyükliği:* Bu, saklanacak sözcüklerin sayı ve miktarı anlamına gelir.
2 *Programlama yöntemi:* Programlama hemen her zaman, özel-amaçlı bir programlama birimini gerektirir, bu birim farklı EPROM tiplerinde birbirinin aynı değildir. Uygun bir programlayıcının bulunup bulunmadığı ve eğer varsa, maliyetinin EPROM seçimini etkileyeceği açıktır. Eğer belli bir EPROM tipi için kullanılabilecek bir programlayıcı mevcutsa, mümkün olan her durumda bu EPROM tipinin kullanılmasının arzu edileceği açıktır.
3 *Silme yöntemi:* Bir EPROM'un silinmesi ya da bir başka deyişle yıkanması da özel-amaçlı donanım gerektirir. Bu durum, programlama konusunda belirtilen nedenlerden dolayı EPROM seçimini etkileyecektir.
4 *Erişim süresi:* EPROM'ların birçoğunun, sözgelimi monolitik RAM birimlerine kıyasla oldukça uzun erişim süreleri vardır. Bu nedenle yavaş EPROM'lar için bellek yazma ve okuma işlemlerinde bekleme durumlarının olması gerekebilir. Bu durum sistem donanımında sorunlara yol açar.
5 *Programlama hızı:* Veriyi bir EPROM'a programlamak için gereken süre, RAM yazma işleminde gereken süreden daha uzun ise de bu, EPROM seçiminde çoğu kez önemli bir etken değildir. EPROM'un programlanması normal sistem çalışmasının bir bölümü değildir, böyle bir programlama yalnızca EPROM'da saklanan veri ya da program değiştirileceği zaman gerekli olmaktadır.
6 *Silme hızı:* Bazı EPROM'lar için yıkama uzun bir işlemdir, bazen 30 ile 60 dakika arasında sürebilir. Bu süre programlama için gerekenden süreden çok daha uzundur. Her ne kadar yıkama işlemi EPROM içeren tamamlanmış bir sistemde ortaya çıkmaz ise de, bu işlem için gereken süre, sistemin geliştirilmesi sırasında geliştirme çalışmasının sık sık tekrarlanan bir bölümü varsa, önemli olabilir. Bu nedenle geliştirmenin ana parçası, normalde program RAM'de iken gerçekleştirilmeli, kod EPROM'a, yalnızca en son aşamada aktarılmalıdır.

Bazı EPROM tiplerinde bulunan bir başka elverişsizlik ise verinin, ancak tümüyle silinebilir olmasıdır. Örneğin herhangi bir adresin ya da adres grubunun içeriğini silmek mümkün değildir. En yaygın EPROM tipleri olan UV-silinebilir EPROM'lar (Kısım 11.3'te anlatıldığı şekilde) bunlar arasındadır. Dolayısıyla yıkama sonrasında EPROM'un tamamı yeniden programlanmalıdır. "Elektriksel olarak

Şekil 11.1 EPROM programlayıcı içeren bir yazılım geliştirme istasyonu

11.2 Programlanır bellek birimlerinin seçimi

Kalıcı programlanır bellek içeren bir mikroişlemci sistemi geliştirirken, planlama kararlarından birisi de, kullanılacak bellek birimlerinin sayı ve tipleridir. Bu kısımda bu seçimi etkileyen etkenleri ele almaktayız. Herşeyden önce, kullanmak istediğimiz birim tipi konusunda yeterince aydınlanmış olmalıyız. Birçok mikroişlemci sisteminde, birden fazla tipte kalıcı bellek kullanılır. Manyetik yüzeyli saklama ortamları, yani sabit disk ya da disket, bant veya tambur biçiminde saklama ortamları, genel-amaçlı mikroişlemci sistemlerindeki yedek saklama ortamları olarak yaygın şekilde kullanılır. Bizim ele aldığımız bellek tipi bu değildir. Bu bölümde RAM'de olduğu biçimde, bellek adres alanının bir kısmını kaplayan yüksek-hızlı, kalıcı, monolitik bellek konusuyla ilgilenmekteyiz. Bu bellek program kodunu ve sürekli veriyi saklar, ve yaygın olarak EPROM şeklinde adlandırılır.

S11.7

EPROM teriminin anlamı "silinebilir programlanır sadece-okunur bellek"tir. Dolayısıyla EPROM'lar veri ya da program kodunun saklanabildiği birimlerdir.

Veri ya da program kodunun saklanma işlemi EPROM'un programlanması olarak adlandırılır. Adından da anlaşılabildiği gibi bu birimler, içerikleri silinebilir olan birimlerdir yani saklanan veriler, bazen EPROM'un "yıkanması" olarak tanımlanan bir işlemle, silinebilirler. Yıkama sonrasında EPROM'a yeni veriler yüklenebilir. Programlanır bir bellek birimi seçerken saklanacak verinin miktarı, dikkate alınması gereken önemli bir unsurdur. Eğer EPROM, program kodu saklayacak olan toplam bayt sayısı, programın geliştirilmesi tamamlanıncaya kadar tam olarak bilinemeyebilir. Bazı durumlarda bu, donanım tasarım kararlarını geciktirebilecektir. Ancak deneyimli bir programcı programın yaklaşık olarak büyüklüğünü, tasarım uygulamasının oldukça erken bir döneminde genellikle tahmin edebilir. Dolayısıyla donanım tasarımcısının, ortaya çıkabilecek hata sayısının yüksek olmasını çok önemli bulmaması koşuluyla, bu sorun donanım tasarımını geciktirmez. EPROM'ların daha popüler olan tipleri düşük-maliyetli birimlerdir, bundan dolayı gerekenden fazla sayıda satın alınmaları, normalde önemli bir kısıtlama oluşturmaz.

EPROM seçimini etkileyen diğer etkenler, başlıca, çalışma özellikleriyle ilgili etkenlerdir. Bunlardan en önemlileri aşağıda sıralanmıştır. Yukarıda incelenmiş olmasına karşın, bütünlük sağlaması için bu listede, bellek büyüklüğü konusuna da yer verilmiştir.

EPROM seçimini etkileyen etkenler

1 *Gereken bellek büyükliği:* Bu, saklanacak sözcüklerin sayı ve miktarı anlamına gelir.
2 *Programlama yöntemi:* Programlama hemen her zaman, özel-amaçlı bir programlama birimini gerektirir, bu birim farklı EPROM tiplerinde birbirinin aynı değildir. Uygun bir programlayıcının bulunup bulunmadığı ve eğer varsa, maliyetinin EPROM seçimini etkileyeceği açıktır. Eğer belli bir EPROM tipi için kullanılabilecek bir programlayıcı mevcutsa, mümkün olan her durumda bu EPROM tipinin kullanılmasının arzu edileceği açıktır.
3 *Silme yöntemi:* Bir EPROM'un silinmesi ya da bir başka deyişle yıkanması da özel-amaçlı donanım gerektirir. Bu durum, programlama konusunda belirtilen nedenlerden dolayı EPROM seçimini etkileyecektir.
4 *Erişim süresi:* EPROM'ların birçoğunun, sözgelimi monolitik RAM birimlerine kıyasla oldukça uzun erişim süreleri vardır. Bu nedenle yavaş EPROM'lar için bellek yazma ve okuma işlemlerinde bekleme durumlarının olması gerekebilir. Bu durum sistem donanımında sorunlara yol açar.
5 *Programlama hızı:* Veriyi bir EPROM'a programlamak için gereken süre, RAM yazma işleminde gereken süreden daha uzun ise de bu, EPROM seçiminde çoğu kez önemli bir etken değildir. EPROM'un programlanması normal sistem çalışmasının bir bölümü değildir, böyle bir programlama yalnızca EPROM'da saklanan veri ya da program değiştirileceği zaman gerekli olmaktadır.
6 *Silme hızı:* Bazı EPROM'lar için yıkama uzun bir işlemdir, bazen 30 ile 60 dakika arasında sürebilir. Bu süre programlama için gerekenden süreden çok daha uzundur. Her ne kadar yıkama işlemi EPROM içeren tamamlanmış bir sistemde ortaya çıkmaz ise de, bu işlem için gereken süre, sistemin geliştirilmesi sırasında geliştirme çalışmasının sık sık tekrarlanan bir bölümü varsa, önemli olabilir. Bu nedenle geliştirmenin ana parçası, normalde program RAM'de iken gerçekleştirilmeli, kod EPROM'a, yalnızca en son aşamada aktarılmalıdır.

Bazı EPROM tiplerinde bulunan bir başka elverişsizlik ise verinin, ancak tümüyle silinebilir olmasıdır. Örneğin herhangi bir adresin ya da adres grubunun içeriğini silmek mümkün değildir. En yaygın EPROM tipleri olan UV-silinebilir EPROM'lar (Kısım 11.3'te anlatıldığı şekilde) bunlar arasındadır. Dolayısıyla yıkama sonrasında EPROM'un tamamı yeniden programlanmalıdır. "Elektriksel olarak

silinebilir" şeklinde adlandırılan EPROM'lardan bazıları, konumların seçimli olarak tek tek silinebilmelerine olanak tanır, fakat kitabın yazıldığı dönemdeki duruma göre bunlar çok yönlü birimler değildir ve UVsilinebilir olan çeşitlere göre fiyatları
S11.1 daha yüksektir.

11.3 Kalıcı monolitik bellek birimleri

Mühendislerin çoğunluğu için, ideal bir monolitik bellek birimi, hem günümüzde mevcut olan statik RAM yongalarının hız ve elverişliliğine, hem de saklanan veriyi, enerji kaynağı kesildiğinde sınırsız bir süreyle koruyabilme yeteneğine sahip olmalıdır. Yarıiletken imalatçıları bunu, bu kitabın yayınlanma süreci içerisinde belki de başaracaktır. Ancak kitabın yazımı sırasında, yukarıda anlatılan ideal bellek özelliklerinin hepsini olmasa da bir kısmını içeren, çeşitli programlanır kalıcı bellek birimleri bulunmaktaydı. Bu bölümde, sözü edilen bu programlanır bellek yongalarından seçilen bazı örnekleri anlatacağız.

UVEPROM'lar

Bunlar programlanır birimlerdir, veri bitlerini entegre devre içindeki kondansatörlerde yük "paketleri" şeklinde saklarlar. Saklanan yük zamanla kaybolsa da bu kaybolma çok yavaştır. Bu bellek çeşidinin tipik örnekleri veriyi 10 yıl ya da daha fazla süre korumaktadır. UVEPROM paketinde belleği oluşturan entegre devrenin hemen üstünde, saydam bir pencere bulunmaktadır. Devrenin yüksek-enerjili mor-ötesi ışığa tutulması, yük saklayan kondansatörlerden bir ışıkakımının akmasına
S11.2 neden olur. Bu kondansatörlerin yüklerinin boşalması, saklanan veriyi siler.

Tipik bir UVEPROM örneği olan 2716, Ek N'deki bilgi föyünde tanımlanmaktadır, burada da görüleceği gibi veri silmek için 2716, 120 mW/m^2 yoğunluktaki bir mor-ötesi lambadan 2,5 cm uzağa yerleştirilmelidir. Normal kullanım sırasında saklanan veri, saydam pencere ışık geçirmeyen bir malzeme ile kapatılarak korunabilir.

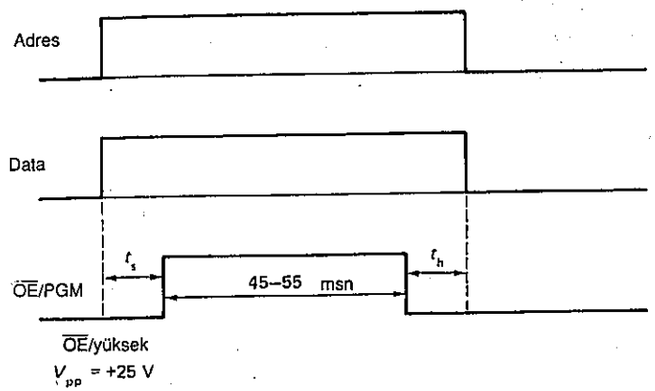
Daha yakın geçmişte kullanıma sunulan EPROM'ların hepsi de, benzer programlama gereklerine sahiptir. Tipik olarak, yonga giriş hatlarından birine, uygun hatlarda adres ve veri mevcutken bir programlama darbesi uygulanır. Ek olarak, kaynak geriliminin veri okuma işlemi sırasında kullanılan değerden daha yüksek bir değere çıkarılması gerekebilir. Örnek olarak, 2716 için programlama işlemi alınmış olup, bu bölümün daha ileriki kısımlarında ve Ek P'de ayrıntılarıyla anlatılmıştır. En yaygın UVEPROM tipleri için kısa bir tanımlama aşağıda verilmiştir:

1702 İlk popüler UVEPROM tipidir. 256 x 8-bitlik olarak düzenlenmiştir, daha önceki programlanır kalıcı bellek birimlerine göre anlamlı bir gelişmeyi simgelemiştir. 1702'nin ana dezavantajları nispeten düşük hız, karmaşık programlama kuralları ve çok sayıda gerilim kaynağına gereksinim olmasıdır. Programlama güçlüklerinden birisi, yaklaşık 30 V genliği olan hızlı yükselme-süreli programlama darbeleri gerektirmesidir. Bu, programcının gerektirdiği devrenin oldukça karmaşık ve bu nedenle de pahalı olması sonucunu doğurur.

1702'nin yerini büyük oranda aşağıda sıralanan daha elverişli UVEPROM'lar almıştır. Yeni tasarımlarda 1702'ye yer verilmesi normalde ekonomik bir tutum olarak görülmez. 2708 Bu birimin düzeni 1.024 x 8 bit'tir. Belki de günümüzdeki en popüler UVEPROM'dur, çok çeşitli sistemlere ekonomik, kalıcı bellek olanağı sunar. Tek kaynak gerektiren tipleri de bulunabilir, fakat 2708'in tiplerinin çoğunluğu çok sayıda gerilim kaynağı gerektirir. 2708'in bir dezavantajı, gerekli olan darbe genliğinin tipik olarak 26 V olmasıdır. Bu yine programcının tasarımını, diğer bazı UVEPROM tiplerindekine göre daha karmaşık duruma getirir.

2716, 2732 ve 2764 Bu UVEPROM'lar sırasıyla 2.048 x 8, 4.096 x 8 ve 8.192 x 8 bit şeklinde düzenlenmişlerdir, işletim özellikleri birbirine benzer ve bunlarla aynı anda çalışılabilir. 2708 ve 1702 gibi daha eski EPROM'larla karşılaştırıldığında 2716, 2732 ve 2764'in ana üstünlüklerinden birisi programlanma kolaylıklarıdır. Bu üstünlük, basitleştirilmiş bir programlama zamanlama diyagramı versiyonu olan Şekil 11.2 ile görselleştirilmiştir. 2716, programlama modunda, $\overline{OE}$'yi yüksek mantık düzeyine getirerek ve Vpp'yi +25 V'a bağlayarak etkinleştirilir. Programlanacak adres, adres hatlarına ve istenen veri de veri hatlarına uygulanır. Daha sonra, adres ve verinin sabitlenmesi için en az 2 mikrosaniyelik bir t_s periyodunun geçmesi gerekir. Bunun ardından $\overline{CE}$/PGM (hattı-çev.) 45 ile 55 ms arasında bir süre için yüksek mantık düzeyine alınır. Bu periyodun sonunda veri, veri hatlarında, $\overline{CE}$/PGM (hattı-çev.) yeniden alçak mantık düzeyine döndükten sonra 2 mikrosaniye= t_h'lik bir süre için daha, kararlı durumda kalmalıdır. $\overline{CE}$/PGM'nin bu şekilde darbelenmesinin sağlanması ve sürekli olarak yüksek mantık düzeyinde kalmasının önlenmesi önem taşımaktadır. Programlama darbesinin yalnızca standart TTL
S11.5 mantık düzeylerini gerektirdiğine dikkat ediniz.

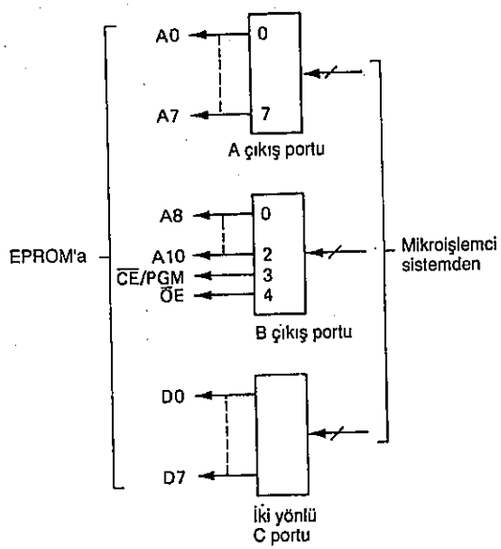
Piyasada çok çeşitli EPROM programlayıcıları bulunmaktadır. Bunların sağladıkları olanaklar, fiyatlarına bağlı olarak çok büyük değişiklikler gösterir. En basit tipte olanlar adres ve verinin düzenlenebilmesi için onaltılı bir klavye içerirler. Programlama temelde "elle ve her seferde bir adres" olacak biçimde yapılır. Daha gelişmiş ve maliyeti daha yüksek olan birimlerde, verinin EPROM'a programlanmasından önce saklandığı bir bellek bulunmaktadır. Bu bellek kullanıcının, verinin doğrulu-



Şekil 11.2 2716 EPROM'u için programlama dalgabiçimleri

ğunu kontrol etmesine olanak verir. Bütün programcılar ayrıca verinin programlanan EPROM'dan okunmasına olanak tanır.

En yararlı EPROM programlayıcıları MDS'nin bir bölümünü oluşturur. Bu durumda programlar, güçlü yazılım destek olanakları kullanılarak geliştirilir ve hata ayıklama işlemine tabi tutulurlar. Donanım ve yazılım, daha sonra bellek yerini alma olanağı ve ICE yardımıyla bir araya getirilirler. Program tamamen kontrol



Şekil 11.3 2716 EPROM'unu programlama ve okuma için bağlantılar

edildikten sonra makine kodu, bir EPROM'a doğrudan doğruya ve otomatik olarak programlanabilir.

Eğer bir ticari EPROM programlayıcısı yoksa, Şekil 11.2'de gösterilen programlama için basit bir zamanlama uygulanması kolayca yapılır. En kolay yöntem doğrudan doğruya, çıkış portları aracılığıyla veri ve $\overline{CE}$/PGM hatları, sürücü adresi ve mevcut bir mikroişlemci sistemden yararlanmaktır. Benzer şekilde EPROM'dan veri okumak için bir giriş portu kullanılabilir. Şekil 11.3'te, 8-bitlik iki çıkış portu ve iki yönlü bir G/Ç portunun 2716 EPROM'larından okuma ve programlama için gerekli olanakları nasıl sağladığı gösterilmektedir. EPROM programlamaya Şekil 11.3'te gösterildiği gibi çok basit bir yaklaşım kullanıldığı zaman, programlama modundayken Vpp 'yi +25 V 'ta tutmak amacıyla elle kullanılan bir anahtarın bulunmasının gerekli olduğu unutulmamalıdır. Ayrıca sistemin kullanıma hazırlanması sırasında, $\overline{CE}$/PGM 'nin sürekli olarak yüksek mantık düzeyinde kalmamasını sağlayabilmek için özel bir dikkat gösterilmelidir.

Aşağıdaki işlem sırasını uygulayan bir program, Şekil 11.3'te gösterildiği gibi V pp =+25 V olarak bağlandığında EPROM'un bir konumunu programlayacaktır:

1 A0-A7 adres bitlerini A portunun 0-7 hatlarına çık
2 A8-A10 adres bitlerini B portunun 0-2 hatlarına çık; 3 hattı 0'da ve 4 hattı 1'de
3 Veri baytını C portu aracılığıyla çık
4 En az 2 mikrosaniye bekle. Normal komut yürütümü süresinden kaynaklanan gecikme bu beklemeyi sağlamak için yeterli olabilir. Bir başka seçenek ise bir ya da daha çok NOP komutunun yürütümüdür, bu da gerekli gecikmeyi sağlayacaktır.
5 B portunun 3 nolu hattına 1 çık, diğer bütün hatları aynen koru.
6 45 msn bekle. Bu amaçla bir gecikme döngüsü kullanılmalıdır.
7 B portunun 3 nolu hattına 0 çık, diğer bütün hatları aynen koru.
8 En az 2 mikrosaniye bekle.

Normalde bir EPROM konumunu programlamak için kullanılan komut, 2716'nın tamamını ya da bir kısmını programlayan daha büyük bir yordamın bir bölümünü oluşturur. Ardından programlanmış EPROM'u, programlamanın başarılı olup olmadığını denetlemek için okuyan bir yordam gelmelidir.

Yukarıda anlatılan 2716 programlama yönteminde ortaya çıkan bir sorun, donanım veya yazılım işlevlerindeki bir arızanın, $\overline{CE}$/PGM'nin sürekli olarak yüksek mantık düzeyinde kalmasına neden olmasıdır. Başka bir seçenekte bu nedenle, $\overline{CE}$/PGM hattını çekmek için bir tek-kararlı devre kullanılmaktadır. Tek-kararlı devre, mik-

roişlemci çıkış hattındaki durum değişikliği ile tetiklenir. Bu düzenleme, $\overline{CE}$/PGM hattının yüksek mantık düzeyinde tek kararlı periyodundan daha uzun bir süre kalmasını önler. 11.4 nolu kaynakta 2716 programlayıcısının tasarımı için biraz daha farklı bir yaklaşım anlatılmaktadır; bu yaklaşımda da programlama darbesini üretmek için bir tek kararlı devre kullanılmaktadır. 11.1 nolu kaynak 2708 programlayıcısını bir ölçüde ayrıntılı olarak anlatmaktadır.

Eriyebilir-bağlantılı PROM'lar

Bu birimler her bir bit saklama devresinin bir eriyebilir bağlantı ile imal edildiği birimlerdir. Programlama, bağlantıları eritmek veya "koparmak" için uygun bir yüksek gerilimin seçmeli olarak uygulanması yoluyla gerçekleştirilir. Eriyebilir bağlantılı PROM'ların, erişim süresi 50 nsn'den az olan yüksek-hızlı iki kutuplu birimlerden, düşük-güçlü CMOS birimlerine kadar değişen çeşitli tipleri bulunmaktadır. Burada geçerli olan genel kural, hız arttıkça tüketilen enerji miktarının da artmasıdır.

Eriyebilir-bağlantılı PROM'ların çalışma özellikleri bir çok yönden daha önce anlatılan ideal kalıcı bellek birimlerinin özelliklerine uymaktadır. Fakat ne yazık ki bunların ana dezavantajları, silinebilir ve programlanabilir olmamalarıdır. Eriyebilir bağlantıların eritilmesi geri dönüşü olmayan bir işlemdir; bu nedenle hata ayıklama işlemi tümüyle tamamlanmış program kodunun saklanmasında yararlı olsalar da, eriyebilir bağlantılı PROM'lar normalde, tasarım/geliştirme çalışmalarında büyük bir role sahip değildirler.

Eriyebilir-bağlantılı PROM'lara bir örnek olarak Şekil 11.4'te, erişim süresi 60 nsn'den daha iyi düzeyde bulunan ve 4.096 bitlik iki kutuplu bir PROM olan Am27S15'in bacak bağlantıları ve blok diyagramı gösterilmektedir. Veri, PROM'da 512 X 8 bit biçiminde düzenlenmiştir. Bellek çıkışı olan veri, üç-durumlu çıkışları bulunan bir kaydediciye mandallanabilir. Strob hattı ST yüksek mantık düzeyinde olduğu zaman 8-bitlik çıkış kaydedicisi saydamdır. $\overline{E1}$ alçak ve E2 yüksek mantık düzeyinde olduğunda üçdurumlu çıkış kapıları yetkilendirilir. ST'nin alçak mantık düzeyine getirilmesi veriyi çıkış kaydedicisine mandallar ve adres değişse bile, ST yüksek mantık düzeyine geçinceye kadar çıkışta başka herhangi bir değişiklik ortaya çıkmaz. Programlanmış durumdayken PROM içeriğinin tamamı sıfırlarla kaplıdır. Bir bağlantının koparılması ilgili bit konumuna 1'i programlar. Kaynak 11.5'te tanınmış bir eriyebilir-bağlantılı PROM serisi için bir programlayıcı anlatılmaktadır.

EAROM'lar

Şekil 11.5'de, elektriksel olarak silinebilir olan bir PROM'un ayrıntılarını gösterilmiştir. Bu birim, bacak düzenlemesi, tanınmış 2716UVEPROM'dakine benzediğinden ve ek olarak, yeterli bir silme gerilimi uygulandığında çabuk ve kolayca silinebilme avantajına sahip olduğundan, ilgi çekici bir birimdir. En yüksek erişim süresi 450 ns'dir.

Şekil 11.6'da biraz daha farklı bir birim gösterilmektedir. Bu birim seri olarak yazılabilen ve okunabilen 256 bitlik veriyi saklar. Şekil 11.5'te gösterilen birim gibi MCM2801 de elektriksel olarak silinebilir özelliktedir. Ancak bunda, silme gerilimi uygulandığında tüm adresleri silen MCM2816'dan farklı olarak, her bir sözcük tek başına silinebilir ve yeniden yazılabilir.

11.4 EPROM programlama

Buraya kadar öğrenci, bir mikroişlemci sistemin geliştirilmesindeki ileri aşamalarda, program kodu ve verinin saklanması için EPROM'ların kullanımı hakkında genel bir anlayış edinmiş olmalıdır. Bu anlayışı perçinlemek için, uygun bir EPROM'un programlanması amacıyla programlayıcı kullanımında pratik deneyim kazanmak önemlidir. Bölümün başında söylediğimiz gibi böyle bir deneyim, öğrencinin bir EPROM programlayıcısına erişebilir durumda olmasını gerektirmektedir. Sözkonusu programlayıcı idealde bir MDS ya da yazılım geliştirme istasyonuyla birlikte bulunmalıdır. Böyle olursa sistem RAM'indeki program kodunun doğrudan EPROM'a aktarılması mümkün olur. Ayrıca, veriyi yeniden okuyarak EPROM'un doğru biçimde programlanıp programlanmadığını denetlemek için gerekli olanaklar da bulunmalıdır.

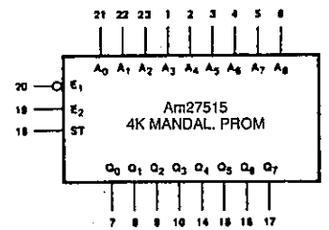
Öğrencinin pratik çalışmada kullanması için, diğerlerine göre çok daha fazla düzeyde elverişli olan birimler UVEPROM'lardır. Donanım çok özelleşmiş bir donanım olmadıkça, ticari EPROM programlayıcıları bir ya da daha çok UVEPROM için özellikle tasarlanmış birimlerdir. Bu nedenle bir alıştırma olarak öğrenci, EPROM'da küçük bir alanı kolaylıkla fark olunabilir bir veri ile programlayacaktır. 00'dan 0F'ye kadar değerlerin ilk 16 konuma saklanması, iyi bir başlangıç olacaktır. EPROM programlayıcısının bir MDS ya da yazılım geliştirme istasyonuna bağlanmış olduğunu varsayarak izlenecek adımları aşağıdaki gibi gösterebiliriz:

1 EPROM'un programlama yapmak istediğiniz alanının boş olduğunu belirleyiniz. UVEPROM'larda boş konumlar normalde FF içerirler. Kullandığınız sistemde EPROM'un boş olup olmadığını otomatik olarak belirleyebilen bir yazı-

AYIRICI ÖZELLİKLER

- Yonga-yerleşik veri mandalları
- Sözcük genişletme kolaylığı için mandallanmış gerçek ve tümlenmiş çıkışlar
- Enerji uygulanma durumunda önceden belirlenmiş KAPALI durum çıkışları
- 82S115 için takılabilme değişikliği
- Hızlı erişim süresi - ticari olarak en yüksek 60 nsn, askeri olarak en yüksek 90 nsn
- N^2 modelleriyle önceden test edilmiş performans
- Yüksek düzeyde güvenilir, ultra-hızlı programlama Platin-Silisid sigortalar - Yüksek programlama sayısı
- Alçak akım PNP girişleri
- Yüksek akım üç-durumlu çıkışlar
- PROM serisi ortak özellikleri ve programlama işlemleri

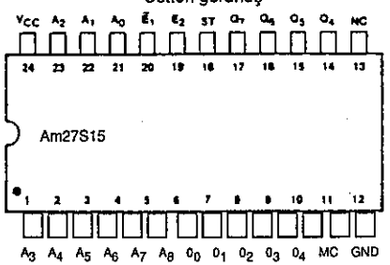
MANTIKSAL SEMBOL



VCC = Pin 24
GND = Pin 12
(11 ve 13 no.lu pinler açık)

BPM-010

BAĞLANTI DİYAGRAMI
Üstten görünüş



Not: 1 no.lu pin doğrultu için işaretlenmiştir.
NC= Bağlantı yok.

BPM-011

SİPARİŞ

Paket tipi	Sıcaklık Sınırları	Sıra Numarası
Hermetic DIP	0°C +75°	Am27S15DC
Hermetic DIP	-55°C - + 125°C	Am27S15DM

İŞLEVSEL TANIMLAMA

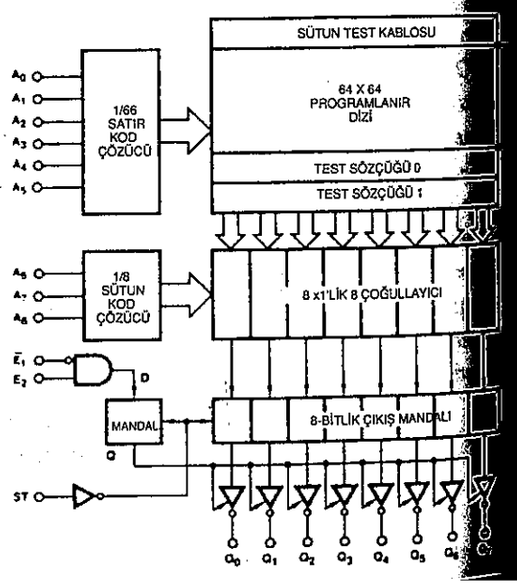
Am27S15, yonga-yerleşik veri ve yetkilendirme mandallarını ren, elektriksel olarak programlanır bir Schottky sadece-okunur bellektir. Birim, 8-bitlik 512 sözcük biçiminde düzenlenmiştir. 16mA tam sürebilme olanağı bulunan üç-durumlu çıkış özellik gösterir.

Saydam moddayken ve strob (ST) girişi YÜKSEK MANTIK DÜZEYİ durumundayken saklı verinin okunması, yonganın yetkilendirilmesi (E 1 ALÇAK MANTIK DÜZEYİ'nde ve E 2 YÜKSEK MANTIK DÜZEYİ'nde) ve A_0-A_8 adres girişlerine ikili sözcük adresinin uygulanmasıyla gerçekleştirilir. Bu modda adres girişlerindeki değişiklikler O_0-O_7 çıkışlarının farklı bir saklı sözcük okumalarına neden olur; herhangi bir yetkilendirme giriş düzeyindeki değişiklikler, çıkışları yetkisizlenir, bunların yüksek empedans durumuna geçmesine neden olur.

Strob girişinin ALÇAK MANTIK DÜZEYİ'ne düşürülmesi birimi mandallı işletim moduna getirir. Strob ALÇAK MANTIK DÜZEYİ durumuna geçtiğinde mevcut çıkış durumu (saklı veriden bir sözcük okuma ya da yetkisizleme), bir başka adres veya yetkilendirme mesajından etkilenmeksizin, pozitif bir (ALÇAK MANTIK DÜZEYİ'nden YÜKSEK MANTIK DÜZEYİ'ne) strob mesajı gelinceye ya çıkıncaya kadar çıkışlarda bulunmaya devam eder. Strob YÜKSEK MANTIK DÜZEYİ durumundayken O_0-O_7, yine adres ve yetkilendirme giriş durumlarına yanıt verir.

V cc gücü ilk kez uygulandığında strob ALÇAK MANTIK DÜZEYİ'nde (mandallı mod) ise çıkışlar yetkisiz durumda olacak ve enerjilenme-durumu önlemlerine olan gereksinimi ortadan kaldıracaktır.

BLOK DİYAGRAM



MCM2816

2048x8-BİTLİK ELEKTRİKSEL OLARAK SİLİNEBİLİR PROM

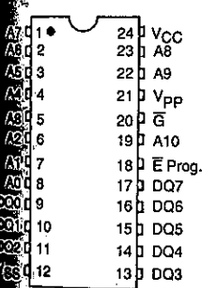
MCM2816, 16,384 bitlik Elektriksel Olarak Silinebilir Programlanır Sadece Okunur bir Bellek'tir, hem kalıcı bellek hem de sistem-içi yeniden programlama gerektiren uygulamalarda, veri kullanımı için tasarımlanmıştır. 24-pinlik çift-sıralı pakette yer alan endüstri standardındaki pin çıkışları MCM2816 EPROM'u popüler olan MCM2716 EPROM'a uyumlu duruma getirmektedir.

MCM2816, sistem-içi silme ve yenilden programlama olanakları nedeniyle zaman ve para tasarrufu sağlar. V pp 25 V'ta ve G hattı- V IL'dayken E/Progr pinine 50 msn süreli yüksekte-etkin bir TTL darbesinin uygulanması, bütün belleğin 1 durumuna geçecek şekilde silinmesine olanak verir. Yeni bir PROM türü olan bu birim, sistem-içi programlanabilirlik özelliğine ek olarak standart EPROM programlayıcısında da programlanır.

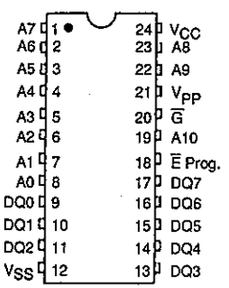
Tasarımının kolay olması için birim, okuma modunda, tek bir güç kaynağı ile çalışır ve statik bir güç-düşürme moduna sahiptir. MCM2816 yüksek güvenilirlik ve kullanılabilirlik amaçlarıyla, yüzer kapı teknolojisi kullanılarak imal edilmiştir.

- Tek +5 V luk güç kaynağı
- Otomatik güç-düşürme modu (Bekleme'de)
- Silme ve programlama için tek bir +25 V luk güç kaynağı
- 8-bitlik 2048 baytlar halinde düzenlenme
- En yüksek erişim süresi=450 nsn MCM2816
- MCM68316E ve MCM2716 'ya pin uyumlu
- Sistem-içi Programlama/Silme yetisi
- 10 msn'lik yonga silme süresi

2816 VE 2716'NIN PİN ÇIKIŞLARI KARŞILAŞTIRMASI



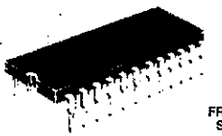
MCM2816



MCM2716

MOS
(N-KANAL, SİLİKON KAPI)

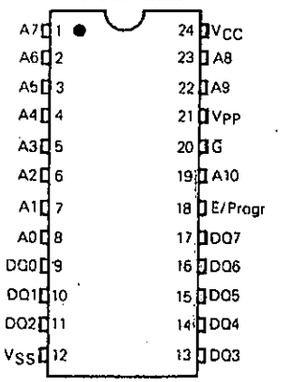
2048x8 BIT ELEKTRİKSEL OLARAK SİLİNEBİLİR PROGRAMLANABİLİR SADECE OKUNUR BELLİK



C SUFFIX
FRIT-YAPIŞTIRMALI
SERAMİK PAKET
KILIF 623-05

L SUFFIX
SERAMİK PAKETTE
MEVCUTTUR
OLGU 716-06

PİN TANIMI



*** Pin adları**

A .....Adres
DQ .....Veri giriş/çıkış
E/Progr. .....Yonga yetkileme/Programlama-silme
G.....Çıkış yetkileme

* Yeni standart sanayi terminolojisi.

MCM2801

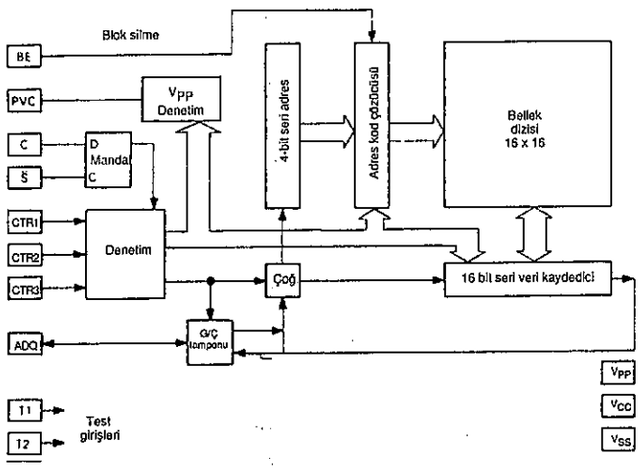
16 X 16-BİTLİK ELEKTRİKSEL OLARAK SİLİNEBİLİR SERİ PROM

MCM2801, 256 bitlik Elektriksel Olarak Silinebilir Programlanır bir Bellek'tir, hem kalıcı bellek hem de sistem-içi bilgi güncelleştirme gerektiren uygulamalarda, az miktarda verinin kullanımı için tasarlanmıştır.

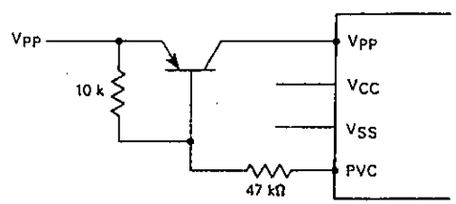
MCM2801, sistem-içi silme ve yeniden programlama olanakları nedeniyle zaman ve para tasarrufu sağlar. Zamanlama işlevleri için dış denetime, ve, veri ve adres için seri formata sahiptir. MCM2801 yüksek güvenilirlik ve üretilebilirlik amaçlarıyla, yüzer kapı teknolojisi kullanılarak imal edilmiştir.

- Tek +5 V luk güç kaynağı
- 16 bitlik 16 sözcük halinde düzenlenme
- Tamamen TTL uyumlu
- Silme ve programlama için tek bir +25 V luk güç kaynağı
- Sistem-içi Programlama/Silme yetisi

BLOK DİYAGRAM



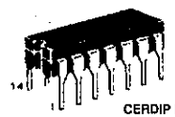
ŞEKİL 1- V_{PP} DENETİMİ



Bu bilgi ve özellik tanımları; bildirimde bulunulmadan değiştirilebilir.

MOS
(N-KANAL, SİLİKON KAPI)

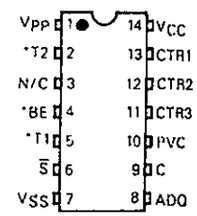
16 x 16 BIT ELEKTRİKSEL OLARAK SİLİNEBİLİR ROM



CERDIP PAKETİ
KILIF 632.06

PLASTİK PAKETLER DE VARDIR
KILIF 646-05

PİN TANIMI



* Normal işleyiş için bu girişler Vss'ye kablolanmalıdır.

PİN ADLARI

ADO .....Çoğullanmış adres/veri giriş/veri çıkışı
C .....
PVC .....Program gerilim denetimi
CTR1, 2, 3, ...Kontrol
BE .....Blok silme
S .....Yonga seçme
T1, T2 .....Test pinleri

Bu birim, girişleri, yüksek statik gerilim ya da elektriksel alanlar nedeniyle oluşan hasarlara karşı koruyan devre düzenlemesine sahiptir, fakat yüksek empedanslı bu devreye, sağlanan maksimum gerilimden daha yüksek bir gerilim uygulanmasının önlenmesi için normal önlemlerin alınması tavsiye edilmektedir.

Şekil 11.6

lım paketi de bulunuyor olabilir.

2 Veri baytlarını sistem RAM'indeki uygun bir alanın ardışık konumlarına saklayınız. Bu işlem, sistemde bulunan ve kullanıcının bellek konumları içeriğini değiştirmesine olanak veren ekranda izleme/hata ayıklayıcı kullanılarak yapılabilir.

3 Veriyi RAM'den EPROM'a aktarınız. EPROM programlayıcısının yazılımında bu amaçla bir komut bulunur.

Bir EPROM programlayıcısıyla birlikte bulunan yazılıma örnek olarak Futuredata genel-amaçlı MDS'ni göstereceğiz. 2704, 2708, 2758 ve 2716 UVEPROM programlayıcıları bu sistemde mevcuttur. Yazılım paketi aşağıdaki komutları sağlamaktadır:

C	EPROM'un boş olup olmadığını kontrol et
P,nn	0'dan başlayarak EPROM'un "nn" baytını programla. Sistem EPROM programlama modunda iken RAM'de istenen herhangi bir bölgenin içeriğini görüntüler. Veri RAM'den EPROM'a, imlecin belirttiği adresten başlamak üzere, aktarılır.
I,nn	Verinin "nn" baytını EPROM'dan sistem RAM'ine, görüntü imlecinin belirttiği adresten başlamak üzere, aktarır.
V,nn	EPROM'daki veriyi geliştirme sisteminin RAM'indeki veriye göre doğrular.

Eğer, öğrencinin kullanabileceği programlayıcı, bir geliştirme sisteminin içinde yer almıyorsa, EPROM programlama alıştırması elle kullanılan bir programlayıcı kullanılarak yine gerçekleştirilmelidir. EPROM başarılı bir biçimde programlandıktan sonra silinebilir ve yeniden programlanabilir. Öğrenci, UVEPROM yongasını bir morötesi ışık kaynağına yeterli uzaklıkta bir süre tutarak yıkama işlemini gerçekleştirmelidir. Bu donanımın elemanları, insan gözüne istenmeyen bir etkilerde bulunmaması için uygun düzeyde morötesi ışık kaynağı emme olanağı ile birlikte sağlanması gerektiğinden, ticari UVEPROM'ların kullanılması önerilmektedir.

Yukarıda anlatılan uygulamalar, bir EPROM'un programlanması ve silinmesi konusunda yararlı bir deneyim kazanılmasını sağlar. Her fırsatta bunun ardından, EPROM'u program kodu saklamak için kullanan bir pratik alıştırma yapılmalıdır. Bunun için uygun EPROM tipini kabul edecek soketleri bulunan bir mikroişlemci sistemi gereklidir. Bu amaçla kullanılabilen ve nispeten düşük maliyetli olan birçok tek-kartlı bilgisayar mevcuttur. Tipik bir örnek olan SBC100 Kısım 6.6'da anlatılmıştır.

Sözkonusu alıştırma temel olarak, bir programın kodunun EPROM'a saklanması, EPROM'un deney sistemine eklenmesi ve programın doğru olarak çalışıp çalışmadığının kontrol edilmesinden oluşur. Bu aşamada test programının gerçekleştirdiği işlemler, doğru çalıştığının kontrol edilebilmesi koşuluyla, önemli değildir. Eğer öğrencinin ICE olanakları bulunan bir MDS'e erişimi mümkünse, test programının kontrolü herhangi bir sorun yaratmaz. Deney sistemi, ICE probu Kısım 7.5'te anlatıldığı biçimde mikroişlemcinin yerindeyken, geliştirme sisteminin denetimi altında çalıştırılır. Bellek yerini alma, EPROM'da yerleşik bulunan programın tam bir denetim altında çalıştırılabilmesine olanak tanır.

EPROM'da yerleşik bir programın çalıştırılması için ICE olanağının kullanımına bir başka seçenek de, kendi içinde bir ekran izleme/hata ayıklama programları bulunduran bir deney sisteminin kullanılmasıdır. Basit bir mikroişlemci değerlendirme sistemi, eğer gereken tipte EPROM'u kabul edecek olanakları varsa, uygundur. Böyle bir sistem kullanıldığında deney sistemininin, ekranda izleme/hata ayıklayıcısı (genellikle ROM veya EPROM'da bulunur), test programının yürütülmünü bilinen şekilde denetler. EPROM'da yerleşik programları bu yolla test ederken, ekranda izleme/hata ayıklayıcıların çoğunluğunun, program içinde durdurma noktaları kullanımına olanak vermedikleri göz önünde bulundurulmalıdır. Bunun nedeni izleme programının (monitörün) durma noktasını, durmanın ortaya çıkacağı komutu değiştirerek işleme koymasıdır. EPROM'da kalıcı olarak saklanmış olan bir programla, bunun mümkün olmadığı açıktır. Bunun çözümü EPROM'da yerleşik programı, RAM'de durma noktasının ayarlanabileceği bir adrese, bir atlama komutu ile değiştirmektir.

11.5 ROM simülasyonu

EPROM'ların geliştirilmekte olan programları ya da geçici test programlarını saklamak için kullanılması sırasında doğan sorunlardan birisi, program değişiklikleri yapmak için harcanan zamandır. 2708, 2716 ve 2732 serileri gibi UVEPROM tiplerinde silme işlemi, 30 dakika kadar zaman alabilmekte ve yeniden programlama için bundan birkaç dakika daha fazlası gerekmektedir. Eğer bu ailenin tipik bir üyesini, 2716'yı ele alırsak, özellik tanımından, bunun 2 Kbaytlık program kodu saklayabildiğini görürüz.

Tipik bir 2-Kbaytlık programın geliştirilmesi sırasında yapılan değişikliklerin sayısı, EPROM'un silinmesi artı yeniden programlama süresi ile, sözgelimi 35 dakika ile çarpılırsa, ürün geliştirme aşamasına, fazladan ne kadar süre ekleneceği konusunda bir fikir edinilecektir. Tabii durum her zaman bunun kadar kötü değildir, fakat yine de, EPROM'ları, geliştirme sırasında geçici program saklama ortamları olarak kullanmak belli bir zaman kaybını da birlikte getirecektir.

ICE olanaklarının bulunmadığı durumlarda bir ROM simülatörü (benzetimcisi), program geliştirme sürecini hızlandırmaya yarayan düşük-maliyetli bir araç işlevi görebilir. ROM simülatörü terimini, ana ilke hepsinde de aynı olduğundan, PROM ve EPROM da dahil herhangi bir ROM tipini simüle eden birim için kullanacağız. Temel olarak bir ROM simülatörü, simüle edilen ROM ile aynı düzenlemeye sahip olan bir RAM'dır. Dolayısıyla 2716 tipi bir EPROM'u simüle etmek için, 2716'nın saklama kapasitesi 2,048 X 8 bit olduğundan, 2,048 X 8 bit saklayan bir RAM'a gerek bulunmaktadır. ROM simülatöründen esnek bir kılavuz kablo çıkar, bu kılavuz kablo, simüle edilen ROM'da bulunan bacakların sayısı kadar pine sahip olan bir DIL probunda sonlanır.

ROM simülatörünü kullanmak için, gerçek ROM, test edilen sistemdeki soketinden çıkarılır ve yerine simülatörün probu takılır. Program kodunu ve veriyi simülatör RAM'ında saklamak için, elbette bir çözüm bulunması gerekir. Bunun yapılması için mevcut olanaklar simülatörün ne ölçüde gelişmiş bir birim olduğuna bağlıdır.

Simülatörün denetim devresi de yine gelişmişlik düzeyine bağlıdır. Birçok ROM simülatörü, kullanıcının okuma/yazma olanaklarını denetlemek için özel bir mikroişlemci içerir. Ancak proba ulaşan çıkış, simülatör, olabildiği ölçüde simüle edilen ROM gibi işlev görmek durumunda olduğundan, muhtemelen tamponlar aracılığıyla doğrudan RAM'den alınır. Probun pin bağlantıları da simüle edilen ROM'dakiler gibi olmalıdır. Bu, simüle edilen her ROM için ayrı bir simülatörün gerekli olduğu anlamına gelir. Bu konuya farklı bir yaklaşım ise ortak bir RAM'in bulunması ve bu ortak RAM'de simüle edilecek farklı ROM'lar için, birkaç "özel" fişli modülün yer almasıdır. Bir EPROM simülatörünün devre tanımlaması, 11.6 nolu kaynakta verilmektedir.

Pratik alıştırmalar

Not: Aşağıdaki alıştırmalar, mevcut programlama olanaklarına ve deney sistemine bağlı olarak 2708, 2716, 2732 veya 2764 UVEPROM'ları kullanımıyla gerçekleştirilebilir:

1 Yeterli bir süre morötesi ışığa tutarak bir UVEPROM'u siliniz. Silme işleminden sonra 0 ile F arasındaki adreslerin içeriğini okuyunuz ve kaydediniz. Daha sonra 0 adresini 00 ile, 1 adresini 01 ile, ve bu şekilde F adresine kadar devam ederek, F adresini 0F ile programlayınız. Daha sonra verinin bulunduğunu

görmek için bu adreslerin içeriğini okuyunuz. EPROM'u siliniz ve programlama işlemini farklı bir veri ile tekrarlayınız.

2 Aşağıda sıralanan işlemler için, kendi deneysel sisteminizde kullanılmakta olan mikroişlemciye uygun program kodunu kağıda yazınız:

A kaydedicisini 0F ile yükle
B kaydedicisini F0 ile yükle
Uygun RAM adresine atla

Bu işlemlerden sonuncusu yalnızca, devre-içi emülasyon kullanmıyorsanız gereklidir. Kodu kendi deney sisteminize kurulabilecek bir EPROM'a programlayınız. Programı sistem izleme/hata ayıklayıcısının denetimi altında çalıştırınız. Programın yürütümünden sonra A kaydedicisinin 0F ve B kaydedicisinin, F0 içerip içermediğini kontrol ediniz.

Kaynaklar ve bibliyografya

11.1 J. W. Coffron, *Practical hardware details for 8080, 8085, Z80 ve 6800 microprocessor systems*, Chapter 7 and 8, Prentice Hall (1981).

11.2 *Electrically alterable memory reliability*, Application Report, NCR corporation, Dayton, Ohio, USA.

11.3 *How to survive in the bipolar PROM, FPLA, PAL, MOSEPROM, diode matrix, PMUX and gate array programming jungle*, Application report, Data I/O Corporation, PO Box 308, 1297 NW Mall, Issaquah, Washington USA.

11.4 "EPROMs survey", *Systems International (GB)*, Vol. 8(10), pp: 35-39 (October 1980).

11.5 "Bench programmer for Fairchild TTL PROMs', *Electron Product Design (GB)*, Vol. 1(8), pp: 26 (November 1980).

11.6 S.K. Roberts, "Simulating microcomputer software development by simulating an EPROM', *Electronic Design News (USA)*, pp: 171-8 (20 April 1979).

Sorular

11.1 Belirli bir uygulama için bir EPROM seçerken göz önünde bulundurulması gereken başlıca etkenleri söyleyiniz.

11.2 Tipik bir UVEPROM'da saklı verinin silinmeden kalacağı süreyi söyleyiniz.

11.3 2716 EPROM'un silinmesi için işlemler dizisini anlatınız.

11.4 2716 EPROM tarafından, veri okuma modunda tüketilen enerji miktarını söyleyiniz.

11.5 2716 EPROM'un 18 nolu pinine programlama sırasında uygulanan darbe süresinin kabul edilebilir alt ve üst sınırlarını belirtiniz.

11.6 27162'nın 21 nolu pinine *(a)* programlama *(b)* veri okuma işlemleri sırasında uygulanması gereken gerilimi söyleyiniz.

11.7 2716 gibi tipik bir EPROM'un erişim süresi ile 2114 gibi yaygın olarak kullanılan bir statik RAM biriminin erişim süresi arasındaki fark nedir?

11.8 Bir mikroişlemci sisteminin zaman zaman fasikül basması gerekmektedir. Bu sistemin fasikül metinlerini, 2716 EPROM'larında saklanan ASCII kodu formunda tutması öngörülmüştür. Bir fasikül, her biri 60(ondalık) karakter içeren 50(ondalık) satırdan oluşmaktadır. Üç farklı fasikülün metnini saklamak için kaç tane 2716 EPROM gereklidir?

11.9 Am27S15 gibi bir PROM'un, mikroişlemci-tabanlı sistemlerin geliştirilmesinde yardımcı olması için kullanıldığında, bir EPROM'a göre ana dezavantajı nedir?

11.10 2811 gibi EAROM'larla kıyaslandığında, 2716 gibi UVEPROM'ların iki dezavantajını söyleyiniz.

Bölüm 12 Mikroişlemci hata-bulma

Bu bölümün amaçları *Bu bölümü bitirdiğinde;*

1 *Mikroişlemci sistemlerde ortaya çıkabilen tipik hataları;*
   (a) *eleman arızası*
   (b) *açık-devre ara bağlantısı*
   (c) *köprüleme ya da kısa-devre ara bağlantısı*
   (d) *harici endüklemenin yarattığı girişim*
   (e) *orijinal yazılım tasarımı hataları*
   şeklinde sıralayabilmeli,
2 *Hata-bulma için sıralanan şu tekniklerin kullanımını açıklayabilmeli,*
   (a) *d.c. testi*
   (b) *mantık probu kullanımı*
   (c) *Katot ışınlı osiloskop (CRO) kullanımı*
   (d) *görsel kontrol*
3 *Alışılmış test birimleriyla hata-bulmaya yardımcı yazılım teşhis programlarının kullanımını açıklamak*

12.1 Hata-bulma cihazları

Elektronik sistemlerde bir ya da daha çok mikroişlemcinin yer almasının, tasarım/geliştirme aşamasında gereken becerileri çok büyük oranda etkilediğini görmüştük. Donanıma ilişkin hata-bulma, mikroişlemcilerin, yeni yaklaşımlar ve özel beceriler gerektiren bir başka alandır. Bir sistemde mikroişlemcilerin bulunması da hata-bulmada kullanılan test donanımı seçimini etkileyecektir.

Elektronik servis mühendisi, genelde, multimetre, katot ışınlı osiloskop ve mantık probu gibi araçlarla çalışır. Bu cihazlar aynı zamanda, mikroişlemci sistemde arızasının belirlenmesinde gereken test işlemlerinin bir bölümünü gerçekleştirmek için de kullanılabilirler. Ancak mikroişlemcilerin kullanıma girmesi, bunlara ek olarak mantık analizörü ve işaret analizörü gibi bazı yeni test cihazlarının kullanıma girmelerine yol açmıştır. Bazı mikroişlemci geliştirme sistemlerinin özellikleri arasında bulunan devre-içi emülasyon da, arıza teşhisinde güçlü bir araçtır. Bu bölümde, mikroişlemci-tabanlı sistemlere uygulanan alışılmış hata-bulma cihaz ve yöntemlerinin kullanımı incelenmektedir. Bölüm 13'te mantık analizörü ve işaret analizörü kullanan daha özel test teknikleri anlatılacaktır.

12.2 Hatalar ne zaman ortaya çıkar?

Bir mikroişlemci sistemde hataların teşhis edilmesi gereği, tasarım/geliştirme/üretim çalışmasının çeşitli aşamalarında ortaya çıkabilir. Bu gereksinimin ortaya çıktığı bazı durumlar aşağıdadır:

1 Yeni yapılmış bir prototip deney kartında hata bulma. Sistem arızası; hatalı donanım tasarımı, hatalı yazılım tasarımı, hatalı donanım kablolaması, eleman arızası ya da bu dördünün herhangi bir birleşimi ile ilişkili olabilir. Bu aşamada tasarım/geliştirme mühendisi aynı zamanda, sıklıkla prototip hatalarının teşhisini koyar ve düzeltir.
2 Tasarımının doğruluğu kanıtlanmış, yapımı yeni tamamlanmış bir devre kartında hata-bulma. Bu, daha çok ürün testi sırasında karşılaşılabilecek bir durumdur. Test mühendisi normalde arızalı kartları iyi örneklerle karşılaştırabilecek kapasitededir. Sorunların çoğunluğu eleman arızası ve üretim sırasında yanlış lehimlemenin neden olduğu kısa veya açık devrelerle ilgilidir.
3 Bir süre doğru şekilde çalıştırılmış donanımın hatalarının teşhis edilmesi ve düzeltilmesi. Bu daha çok saha mühendisinin (field engineer) karşılaştığı bir durumdur. Bu hataların çoğunluğu eleman arızası sonucu olarak ortaya çıkar. Sahada en yaygın hata giderme yöntemi belki de, hatalı kartların değiştirilmesidir. Ancak, yeterli araç gerecin bulunması koşuluyla eleman düzeyinde hata bulmak mümkündür ve ayrıca maliyet-verimlilik açısından da daha yararlı olabilir.

12.3 Tipik hatalar

Mikroişlemci sistemlerdeki arıza nedenlerinin birçoğu diğer elektronik sistemlerdeki arıza nedenlerinin aynısıdır. Aşağıdakiler bunlar arasında sayılabilir:

(a) *Eleman arızası :* Mikroişlemci sistemlerde genel olarak dirençler ve kondansatörler gibi pasif elemanla, özel bir kaç yarıiletken eleman (transistör ve diyot gibi) ve bilgisayardaki işlemleri gerçekleştiren entegre devreler bulunur. Bu elemanlardan herhangi birindeki arıza, sistem arızasına yol açabilir.
(b) *Devre kartında* bulunan "açık-devre" ler, bunların bağlanması gerekir.
(c) *Devre kartında* bulunan "kısa-devre" ler, bunların bağlanmaması gerekir. Bu sorun genellikle, bir baskılı devrede birbiriyle yanyana bulunan yollar arasında ortaya çıkar.
(d) *Harici endüklemenin yarattığı girişim:* Şebekeden-doğan girişimler belki de bu sorunun en yaygın olarak gözlenen tipini oluşturmaktadır. Aynı şebekeyi paylaşan devrelerle ve anahtarlamalı endüktif devreler, genelde bu tipte bir

girişimin oluşmasına yol açarlar. Her ne kadar çözümü güç olan bir sorun olsa da, mikroişlemci sistemindeki d.c. güç kaynağı ve dağıtım şebekesi dikkatle incelenerek, bu girişimin etkileri en alt düzeye indirilebilir.

(e) *Yazılım hataları* : Mikroişlemci sistemlerin müşteriye, saptanamamış yazılım hataları ile birlikte verilmesi, seyrek karşılaşılan bir durum değildir. Bu tip yazılım hataları yalnızca çok ender durumlarda, sisteme normal dışı girişler
S12.1 gibi arızalara neden olabilir.

12.4 Görsel inceleme

Dikkatle yapıldığı zaman görsel inceleme bütün hata-bulma tekniklerinin en maliyet-verimli olanıdır. Bir baskılı devre kartının büyüteç yardımıyla, gözle dikkatli ve ayrıntılara eğilerek incelenmesine yirmi otuz dakikanın ayrılması çoğu kez, hata bulmak için birçok saat harcanılmasını önler. Bu, özellikle, iyi tasarlandığı bilinen
S12.3 baskılı devre kartlarındaki hatalar için geçerlidir.

Baskılı devre kartlarında görülen yaygın sorunlardan biri, ince tel artıklarının bulunmasıdır. Bunlar, elemanların montajından sonra geride kalan küçük kablo parçaları ya da muhtemelen lehim artıklarıdır. Mikroişlemci sistemlerde kullanılan baskılı kartların çoğunluğunda yollar birbirine çok yakın biçimde gider ve bu artıklar tarafından kolaylıkla köprülenirler, bu da kısadevreye neden olur. Kartın (tabii ki elemanların bulunmadığı tarafının) alkollü bir diş fırçasıyla fırçalanması bu hatayı gidermek için oldukça pratik bir çözümdür.

Baskılı bir devrede bulunan yollar arasındaki lehim "köprüleri" de, kablo artıklarının yol açtığı sorunlara yol açarlar, ancak daha kolay göze çarparlar. Bunlar, elemanları yerleştirirken yapılan lehimlemeden kaynaklanırlar. Lehim köprüleri, çiftsıralı entegre devrelerin pinleri arasında sıkça görülürler. Fakat bu lehim çapak-
S12.2 ları, tel artıklarının tersine, fırçalama ile temizlenemezler.

"Soğuk" lehim, elektronik servis mühendislerinin yakından bildiği sorunlardan birisidir. Bunlar hatalı lehimleme tekniğinin sonucudur, görünüşte hatasızdırlar, fakat elektriksel olarak açık-devredirler. Bunların hiç fark edilememesi şeklinde bir kural olmasa da, genelde bilinenin aksine daha seyrek olarak karşımıza çıkarlar. Sorunlarının teşhis edilebilmesinde uzun süre çok yaygın olmalarının bir nedeni de, daha az açıklanabilir nitelikteki hatalar için uygun bir bahane gibi kullanılabilmeleridir. Çok daha yaygın olan bağlantı çeşitleri, yetersiz lehimleme veya çok düşük sıcaklıkta lehimlemenin sonucu olan, hem mekanik hem de elektriksel olarak eksik durumdaki bağlantılardır. Bu çeşit bağlantı kusurları dikkatli bir görsel inceleme ile genellikle belirlenebilir.

Elemanlar, görsel inceleme sırasında dikkatle kontrol edilmelidir. Eleman uçlarının bozulmasından başka, transistörlerin yanlış bağlanması, elektrolitik ya da tantal kondansatörlerin polaritelerinin tersine dönmesi, diyotların ters bağlanmaları vb. arızaların hepsi bu aşamada teshis edilebilir. Görsel olarak her zaman teşhis edilemeyen yaygın bir hata da, çift sıralı entegre devrelerin soketlere ters takılmasıdır.

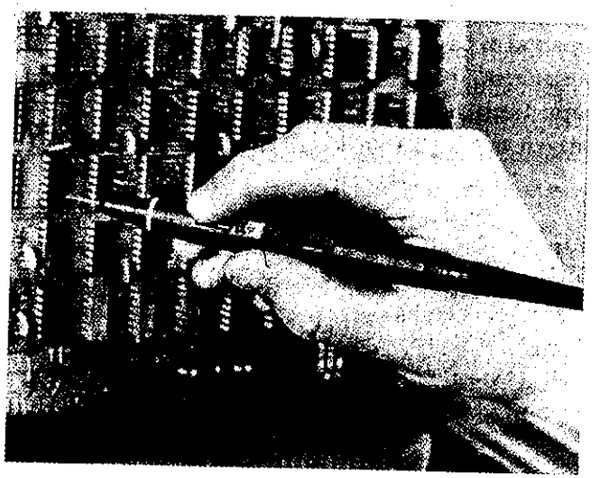
Bu işlem yapılırken, bir veya birkaç bacağın bükülmesi ve sokete girmemesi kolayca mümkündür. Hata, dıştan bakıldığında görülemeyebilir. Soketlerdeki entegre devrelerin hepsinin tek tek çıkarılarak yeniden takılması, bu hatanın saptanması için elverişli bir yoldur.

12.5 Multimetre

İster analog isterse sayısal olsun, alışılagelmiş tipte multimetre, analog veya klasik sayısal devreler için geniş ölçüde kullanılan bir test cihazıdır. Yararları mikroişlemci sistemler için biraz sınırlıdır. Sabit mantıksal düzeylerin, entegre devre kaynak gerilimlerinin vb. kontrolü için multimetrenin tek bir seçenek olduğu düşünülebilir. Gerçekte ise osiloskop, çoğu kez çok daha tatmin edici bir cihazdır. Güç kaynağı çıkış hatlarındaki dalgalılık miktarı ve istenmeyen darbelerin ya da çapakların varlığı, hata-bulmada çok büyük öneme sahip olabilir. Standart d.c. ölçümleri yaparak bu tip bilgiler elde edilmez. Sabit mantık düzeylerinin ölçümü
S12.4 yapılırken de aynı şey geçerlidir.

Sabit mantık düzeyleri, mikroişlemci sistemlerde çok yaygın olarak ortaya çıkmaz. Deneyimi, sözgelimi TTL mantığını kullanan sıralı mantıksal sistemlerle ilgili olan mühendisler çoğunlukla bunu unuturlar. "Klasik" sıralı mantıksal sistemleri testin standart yöntemlerinden birisi bütün sürekli saatlerin bağlantısını ayırmak ve saat hattına bağımsız darbeler uygulayarak sistemi, tek-vuruşlu modda çalıştırmaktır. Önemli bağlantı noktalarındaki mantık düzeyleri her adımdan sonra kontrol edilir ve devrenin istenen şekilde çalıştığı doğrulanır ya da duruma göre, istenen şekilde çalışmadığı belirlenmiş olur. Multimetreler bu amaç için oldukça yeterli araçlardır.

Yukarıda anlatılan tek-vuruş tekniği ne yazık ki mikroişlemcilerin çoğunluğuna uygulanamaz. Bunun nedeni bu birimlerin en düşük saat frekansı açısından belli bir sınırı bulunan dinamik birimler olmalarıdır. Eğer saat, bu en alt frekansın altına düşerse, mikroişlemci verimli şekilde işlev görmez. Tek-vuruşlu olarak adlandırılan çalışma yapılabilir, fakat bu tekli saat darbelerini uygulamaya sokmaz ve işlemci çalışmasını sürekli olarak sürdürür.



Şekil 12.1 Mantık probu

Mikroişlemci sistemlerin test edilmesinde multimetre ile yapılabilen yararlı işlevlerden birisi sürekliliğin kontolüdür. Birim, direnç ölçme modunda çalışırken, kısa ve açık-devreler de bulunabilir. Normalden çok daha yüksek bir değerin okunması bir devre arızasını göstereceğinden, ortalama d.c. kaynak akımının ölçümü de bazen gerekli olmaktadır.

12.6 Mantık pobu

Mantık probları, başlangıçta sabit mantıksal düzeyleri göstermek için kullanıma sunulmuştur. Normalde ince ve kurşun kaleme benzer görünümde olan bu birimlerin, devredeki bağlantı noktalarıyla temas edebilen bir ucu vardır. Ayrıca üzerinde, bir ya da birkaç LED bulunur. Genel çalışma ilkeleri, uç, mantıksal "1" ile bağlantı kurduğunda ışığın yanması, bunun tersi sözkonusu olduğunda sönmesi şeklindedir. Daha iyi bir seçenek ise biri mantıksal 0 durumunu, diğeri de mantıksal 1 durumunu gösteren iki LED'in bulunmasıdır. Devredeki yüzer (floating) noktalar ise, hiç bir LED'in yanmaması ile belirlenmektedir. Tipik bir mantık probu, Şekil 12.1'de gösterilmiştir.

Günümüzde mevcut olan mantık problarının çoğunluğu, bazı darbe algılama özelliğine de sahiptir. Olanaklardan bir tanesi sürekli darbe dizilerinin saptanmasıdır. Kalemin ucu, gerilim dalgabiçimi sürekli darbe dizisi olan bir devre moduyla temasa geçtiğinde, bir LED yanmaktadır.

Darbe uzatma özelliği da mantık problarında, yaygın olarak bulunan bir özelliktir. Algılanan darbe, bu çalışma modunda, bir LED'in, farkedilebilecek bir süreyle yanmasını sağlamak üzere, uzatılmaktadır; söz konusu süre genellikle 1 saniyedir. Sabit bir mantıksal düzeyde olması gereken hatlarda yalancı darbe ya da çapak ararken bu olanak yararlı olabilir. Darbe uzatma olanağının yararlı olabileceği bir başka durum ise, bir giriş ya da çıkış komutunun yürütümü sırasında, sözgelimi bir G/Ç port adres kod çözücüsünün çıkışında bir darbenin bulunup bulunmadığının saptanmasıdır.

Mantık probları, mikroişlemci sistemlerdeki ve klasik mantık devrelerdeki hataların teşhis edilmesi için yararlı bilgiler sağlarlar. Test donanımının diğer cihazlarıyla kıyaslandıklarında nispeten daha düşük maliyetli birimlerdir ve bu nedenle test
S12.5 veya servis mühendisinin kullanabileceği olanaklara maliyet-verimli bir ek olarak
12.6 düşünülebilirler.

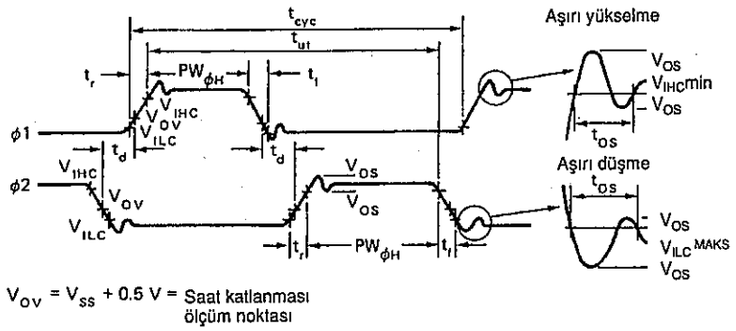
12.7 Osiloskop

Geleneksel analog ve sayısal elektronik sistemlerin çoğunluğunda hata-bulma işlemi için katot ışınlı osiloskoplar, diğerlerine göre çok büyük oranda daha yararlı cihazlardır. Birçok mühendis, mikroişlemci-tabanlı sistemlerde durumun her zaman böyle olmamasını şaşırtıcı bulmaktadır. Bunun iki nedeni vardır. Birinci neden aynı anda gözlenmesi gereken dalgabiçimi sayısı ile ilgilidir. İkinci neden ise dalgabiçimlerinin özellikleriyle ilişkilidir.

Analog bir sistemin bir birimini test ederken, giriş ve çıkış dalgabiçimlerinin gözlenmesi, genellikle o birimin istenen şekilde iş görüp görmediğini belirlemek için yeterlidir. Bu işlem bir çift kanallı osiloskop kullanılarak yapılabilir. Oldukça karmaşık testler için bile, dört kanallı osiloskop analog sistemlerin çoğunluğu için uygundur. Mikroişlemci sistemlerin testi söz konusu olduğunda ise durum oldukça farklıdır.

8-bitlik tipik bir "orta karar" mikroişlemcide 8-hatlık veri yolu ve 16- hatlık adres yolu bulunur. Sistemdeki veri işlemlerini kesin biçimde izlemek için, 24 hattın tamamındaki mantıksal düzeyleri aynı anda gözlememiz gerekir. İdealde, aynı anda diğer denetim ve durum hatlarında neler olduğunu da görmek isteyebiliriz. Klasik çift kanallı osiloskop bu durumda bize çok fazla yardımcı olmaz.

Analog elektronik sistemler yaygın olarak, genelde sinüzoidal olan periyodik sinyaller kullanılarak test edilirler. Bu sinyallerin düzenliliği, osiloskop görüntüsünün, bir noktada tetiklenmesine imkan verir, böylece sabit bir görüntü elde edilmiş



Şekil 12.2 M6800 saat dalgabiçimi özellik tanımı

olur. Bunu bir mikroişlemci sistemindeki normal durumla karşılaştırınız. Adres veya veri yolunun bir hattını incelersek, iki mantık düzeyi arasında değişen bir gerilim görürüz. Değişmeler eşzamanlıdır, sistem saat saykılının belirli bir noktasında ortaya çıkarlar. Ancak değişmelere ait desenin düzenli bir desen olması gerekmez. Bu nedenle dalgabiçimi tekrarlamasız olabilir. Sinyal tekrarlamalı (periyodik) olsa bile bir periyodu, kolayca birkaç yüz ile birkaç bin saat saykılı uzunlukta olabilir. Bir osiloskop, kararlı bir görüntü elde etmek için normal yoldan eşzamanlı yapılamaz. Osiloskop, bir veri yolunda bir aktivitenin ortaya çıkmakta olduğunu, yani gerilimin iki mantık düzeyi arasında değiştiğini gösterir. Ancak bu değişikliklerin zamanlamaları ile ilgili herhangi bir belirli bir ayrıntıyı görmek
S12.7 mümkün değildir.

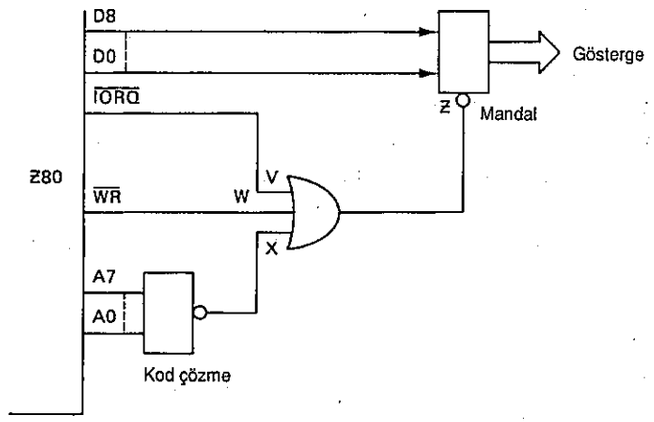
Yukarıda açıklanan kısıtlamalara karşın bir osiloskopun, mikroişlemcilerin test edilmesinde yararlı bir amaca hizmet ettiği uygulamalar vardır. Bu uygulamalardan en göze çarpanı, sistem saat dalgabiçiminin incelenmesidir. Örnek olarak M6800 mikroişlemcisi için saat tanımları, Şekil 12.2'de, gösterilmiştir, bu tanımlar Ek D'den alınmıştır. Yükselme ve düşme zamanları gibi bazı saat parametreleri oldukça kritik olabilmektedir. Kaliteli bir osiloskop, bu parametrelerin değerlerini ölçmek için genellikle en iyi araç durumundadır. Aynı şey herhangi bir "gerçek zaman" ölçümü için de doğrudur. Mantık analizörü (Bölüm 13.2'de tartışılmıştır) gibi cihazlar gerilimleri, belli bir eşik düzeyin altında ya da üstünde olmalarına bağlı olarak, yüksek veya düşük biçiminde gösterirler. Bu özellikleri nedeniyle bu cihazlar, dalgabiçimi yükselme ve düşme zamanı gibi büyüklüklerin ölçümü için uygun değillerdir. Osiloskop ayrıca, darbelerin ön ve arka kenarlarındaki geçici osilasyonları gözlemek için de en iyi cihazdır, bu osilasyonlar genelde çınlama şeklinde anılırlar.

Kısım 12.5'te söylediğimiz gibi, anma değeri sabit olan bir gerilimin ölçümü, osiloskobun yararlı bir rol üstlenebileceği bir başka uygulamadır. Elbette, örneğin bir sayısal voltmetre ile sağlanan hassasiyet elde edilemez. Ancak mikroişlemci sistemlerinde, bir "sabit" gerilimin önemli bir özelliği, genelde, herhangi bir değişim gösterip göstermeyeceğidir. Birkaç nanosaniye gibi düşük genişlikteki yalancı darbeler kolaylıkla sistem arızasına neden olabilir. Bu nedenle mikroişlemci sistemlerdeki d.c. gerilim düzeylerini gözlemek için kullanılan osiloskopların, aynı zamanda çok dar darbeleri de algılayabilme yeteneğine sahip olmaları önem-
S12.8 lidir.

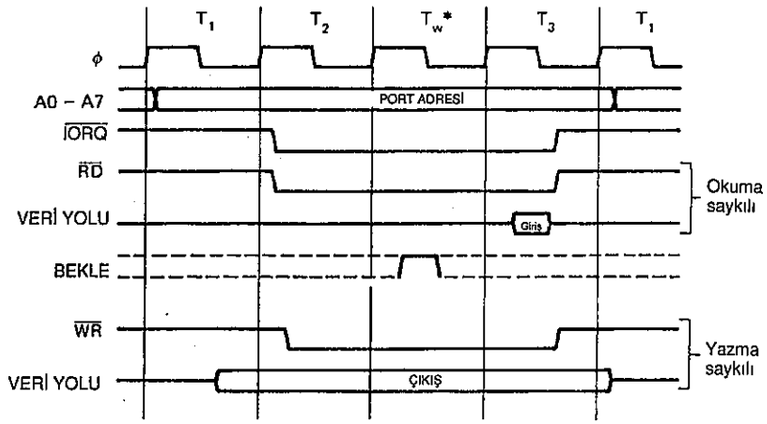
12.8 Tekrarlamalı test dalga biçimlerinin üretilmesi

Eğer tekrarlamalı dalgabiçimleri içeren bir test uygulaması yapılırsa, mikroişlemcili sistemlerde hataların teşhis edilmesinde osiloskobun yararlılığı büyük ölçüde arttırılabilir. Özel test programlarıyla, bir mikroişlemcinin osiloskop tarafından gözlenebilir dalgabiçimleri üretmeye zorlanması çoğu kez mümkündür. Bu programlarda oldukça kısa süreli döngüler bulunur.

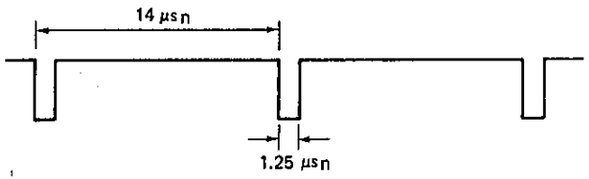
Bir sistemin Z80 mikroişlemci içeren bölümü, Şekil 12.3'de gösterilmektedir. Z80'in görevinin, bir dış ekranı, bir kaydırmalı kaydediciye iki tane 4-bitlik BCD sayı mandallayarak, bir-dakikalık aralıklarla güncellemek olduğunu varsayalım. Ayrıca, yanlış bir görüntüye neden olan bir donanım hatasının da mevcut olduğunu düşünelim. Bu hatalı görüntü dışında sistem normal biçimde çalışmaktadır.



Şekil 12.3



Şekil 12.4 Z80 G/Ç işlemlerinin zamanlaması



Şekil 12.5 Z noktasındaki dalgabiçimi

Sistem normal çalışma programı ile çalışmaktayken her dakikada bir ekran portuna bir çıkış gerçekleşmektedir. Bu, çıkış işlemi ortaya çıktığında tetiklenen bellek içeren bir osiloskop yardımıyla dalgabiçimleri gözlenerek araştırılabilir. Daha iyi bir yöntem, çıkış işlemlerini daha sık biçimde yürüten özel bir test programı kullanmaktır. Böylece normal bir osiloskop kullanılarak gözlem yapmaya uygun kısalıktaki periyodlarla tekrarlamalı dalga biçimlerinin üretilmesi gerçekleşecektir.

Şekil 12.5'te, Z80 mikroişlemcisi tarafından OUT 0F komutu yürütümü gerçekleştirildiğinde, $\overline{IORQ}$ ve $\overline{WR}$ hatlarının komut yürütüm saykılı boyunca alçak mantık düzeyine geçtiği görülmektedir, bu şekil Ek F'den alınmıştır. 0F port adresi de A0-A7'de, adres yolunun en büyük değerlikli sekiz hattında görünür, çıkış verisi veri yolunda ortaya çıkar. Şekil 12.4'te, kodu çözülmüş port adresi $\overline{IORQ}$ ve $\overline{WR}$ hatları ile birlikte, adres yolundaki veriyi 8-bitlik mandal içine saatleyen bir darbe üretir.

Durumun ilk değerlendirmesini yaparak, sistem (ekran dışında) normal biçimde çalışıyor göründüğünden, adres veya veri yolu hatlarının hiç birinin toprak ya da pozitif kaynağa kısa devre yapma olasılığının bulunmadığını anlarız. Bu sinyal de bellek yazma işlemlerine katıldığından, aynı şey $\overline{WR}$ hattı için de geçerlidir.

Daha ileriye geçmeden, bir hatanın gerçekten mevcut olduğunu saptamak gerekir. Bunu yapmak için aşağıdaki iki komuttan oluşan bir test programı yürütülür:

```
LD    A,xx
OUT   (0F),A
```

Bu program A kaydedicisini onaltılı "xx" sayısı ile yükler ve bu sayıyı 0F portuna çıkarır. Program "xx"=00, 11, 22, 44 ve 88 değerlerindeyken beş kez yürütülür. "xx" in bu değerleri kullanılarak, mandal giriş ve çıkışının her hattı hem 0 ile hem de 1 ile denenir.

Program her çalıştığında, görüntülenen sayı ve mandal çıkışlarındaki mantık düzeyleri denetlenir. Bu uygulama sonucu, bir hata olup olmadığı ortaya çıkacaktır. Eğer bir hata varsa bu işlem, hatanın, ekranın kendisinde bulunup bulunmadığını da saptayacaktır.

Hatanın, görüntülemede bulunmadığı varsayıldığında, diğer yüksek olasılıklı hata yerleri şunlardır:

1 Mandal ve ilişkili olduğu hatlar
2 Üç girişli VEYA kapısı ve ilişkili olduğu hatlar
3 Port adres kod çözücü ve ilişkili olduğu hatlar

Bunların her birinde test yöntemi, ilgili elemanin giriş ve çıkışını denetlemek olacaktır. Bunu normal bir osiloskop kullanarak yapabilmek için aşağıdaki özel test programı kullanılır:

```
TEKRAR   LD    A,xx
         OUT   (0F),A
         JP    TEKRAR
```

Bu durumda OUT (0F),A komutunun her yürütümünde, veri yolundaki veriyi 8-bitlik kaydediciye saatlemek için üç girişli VEYA kapısının çıkışında bir darbe görünmelidir. Gösterilen programda Z80 saatinin 2 MHz lik bir frekansa sahip olduğunu varsayarsak, komut yürütüm süreleri aşağıdaki gibi olur:

```
TEKRAR  LD   A,xx      3.5 mikrosaniye
        OUT  (0F),A    5.5 mikrosaniye
        JP   TEKRAR    5.0 mikrosaniye     Toplam 14.0 mikrosaniye
```

Şekil 12.4'te Z80 G/Ç komut yürütümü için verilen zamanlama diyagramına bakıldığında, OUT komutunun yürütümünde $\overline{WR}$ ve $\overline{IDRQ}$ hatlarının yaklaşık 2.5 saat saykılı, ya da 1.25 mikrosaniye süresince alçak mantık düzeyine geçtiği görülebilir. Dolayısıyla program, Z noktasında, Şekil 12.5'te gösterildiği gibi her 14 mikrosaniyede bir yineleyen, 1.25 mikrosaniyelik genişlikte bir mandal zamanlama darbesi üretmelidir. Böyle bir tekrarlamalı darbe dizisi normal bir osiloskop kullanılarak kolaylıkla gözlenebilir.

Z noktasında normal bir darbe dizisinin bulunması kaydediciye ilişkin bir hatayı gösterir. Bu, hatalı bir yonga ya da muhtemelen veri hatlarının birinde "soğuk" lehimden kaynaklanan bir açık-devre olabilir. Bu çıkış portu dışında veri yolunda kısa devrenin bulunmasının, sistem işletiminin tatmin edici olduğu göz önünde bulundurulduğunda, pek olası olmadığını, daha önce belirtmiştik.

Eğer Z noktasında bir darbe dizisi bulunmuyorsa, VEYA kapısının üç girişi osiloskop kullanılarak incelenir. Bu incelemeyle hatanın VEYA kapısında mı, yoksa giriş hatlarından birinde mi olduğu saptanır. Port adres kod çözücüsü benzer
S12.9 yolla denetlenebilir.

12.9 Test ve self-test (kendi kendine test) programı

Bölüm 12.8'de özel test programlarının, osiloskobun mikroişlemci sistemdeki hata-bulma işleminde kullanılmasına olanak vermek için, tekrarlamalı dalgabiçimlerini nasıl ürettiğini gördük. Test programlarının kullanımı bu sınırlı örnekte gösterilebileceğinden çok daha geneldir. Standart bir test ve hata-bulma uygulaması, amacı sistemin çeşitli parçalarını denemek olan özel bir test programını çalıştırmalıdır. Bu çeşit test programları PROM veya EPROM'da saklanabilir. Test mühendisi daha sonra, test programı ROM veya EPROM'unu, normal sistem yazılımını içeren ROM'un yerine, bu ROM'un bulunduğu sokete yerleştirebilir. Bir başka seçenek ise sistemde bu amaç için bir yedek soketin bulunmasıdır.

Daha gelişmiş ve karmaşık bir test ve hata-bulma yaklaşımıyla tasarımlanan sistemlerde, özel test programları, sistem yazılımı ile zorunlu olarak sistem yazılımının içindedir. Böyle bir durumda servis mühendisi, hatta sistemin kullanıcısı, her hata kuşkusunda kolaylıkla test yordamlarını çalıştırabilir. Yazıcılar bu düşünüş biçiminin güzel örneklerini oluşturur. Günümüzde kullanımda bulunan hemen


```
:CDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgh
DEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghi
EFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghij
FGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
GHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijkl
HIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklm
IJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmn
JKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno
KLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnop
LMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopq
MNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr
NOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrs
OPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrst
PQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstu
QRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuv
RSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvw
STUVWXYZ[\]^_`abcdefghijklmnopqrstuvwx
```

Şekil 12.6 Self-test çıktısı

hemen bütün yazıcılarda, denetim devresinin bir parçası olarak bir mikroişlemci yer almaktadır. Çoğu kez ROM'da saklanan denetim programı, aynı zamanda test yordamları da içerir.

Tipik bir küçük matris yazıcı, örneğin EPSON MX80, kullanıcının, satır atlatma kontrol anahtarı basılı durumdayken enerji anahtarını açık duruma getirerek bir test yordamını çalıştırmasına olanak tanır. Test programı, yazılabilen tüm karakterleri içeren bir basılı çıktı üretir. Şekil 12.6'da, bu test çıktısının bir bölümünü göstermektedir.

Bölüm 13'te göreceğimiz gibi, özel test programlarının kullanılması, işaret analizörü gibi belirli bazı özel hata-bulma cihazlarının kullanılması için temel niteliklerden biridir. Mantık analizörleriyle hata-bulma işlemi de test programlarının yardımından çok büyük oranda yararlanır.

Daha yüksek gelişmişlik düzeyinde bile, günümüzün sistemleri, kendi test yordamlarını gerçekleştirecek biçimde tasarlanmaktadır. Bu, tasarım çalışmasının en ba-

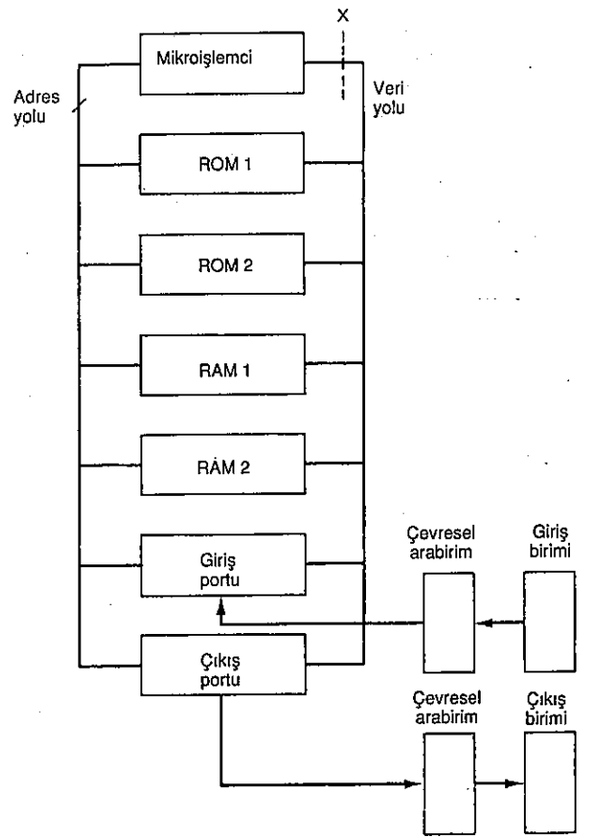
şında işe koyulmak koşuluyla, uygulanması çok güç olan bir işlem değildir. Sistem yazılımı düzenli aralıklarla test yordamının yürütülmesine neden olur. Yordamların sonuçları program tarafından denetlenmekte ve bu sonuçlar olması gereken şekilde değilse bir hata mesajı yazılmaktadır. Böyle bir self-test olanağının gelecekte tüm mikroişlemcilerin ortak özellikleri arasına girmesi beklenebilir.

Hata-bulma olanakları olmaksızın tasarlanan sistemlerde test programlarının çalıştırılması her zaman elverişli olmayabilir. Test programları içeren ROM'ların, sistem ROM'larının yerine nasıl yerleştirildiklerini anlatmıştık. Ancak bu, programın çalışması üzerinde bir denetim sağlamaz. Böyle bir durumda yapılabilecek en iyi şey, enerji anahtarının açılması ya da sıfırlama sırasında girilen bir test programını çalıştırmaktır. Özel bir mikroişlemci sisteminin, test mühendisinin sistemle iletişim kurabileceği, klavye ve ekran gibi olanakları bulundurması da olası değildir. Bu sorunların üstesinden gelmek için özel test cihazları gereklidir. Bu amaca yönelik olarak kullanılabilecek cihazların bazıları bir sonraki bölümde anlatılmaktadır.

12.10 Serbest-çalışan test

Hata, sözgelimi bir çevresel arabirim devresinde ya da bir G/Ç portuyla sınırlı olduğu zaman, tekrarlamalı dalgabiçimleri üreten test programlarının kullanım yöntemi (Bölüm 12.8 ve 12.9'da anlatılmıştır) yararlıdır. Bu programın başarılı olması için, mikroişlemcinin kendisinin doğru biçimde çalışması bir ön koşuldur. Tekrarlamalı dalgabiçiminin üretimi, test programının komutlarının düzgün biçimde yürütümüne bağlı olduğundan, böyle bir ön koşul bulunmasının zorunlu olduğu açıktır. Bu bölümde, daha temel nitelikte bir hata mikroişlemcinin komut yürütümünü görünür biçimde önlediğinde kullanılabilecek bir test uygulaması anlatacağız. Herhangi bir özel test programından yararlanabilmemiz, önce, böyle bir hatayı ortadan kaldırmamızı gerektirir.

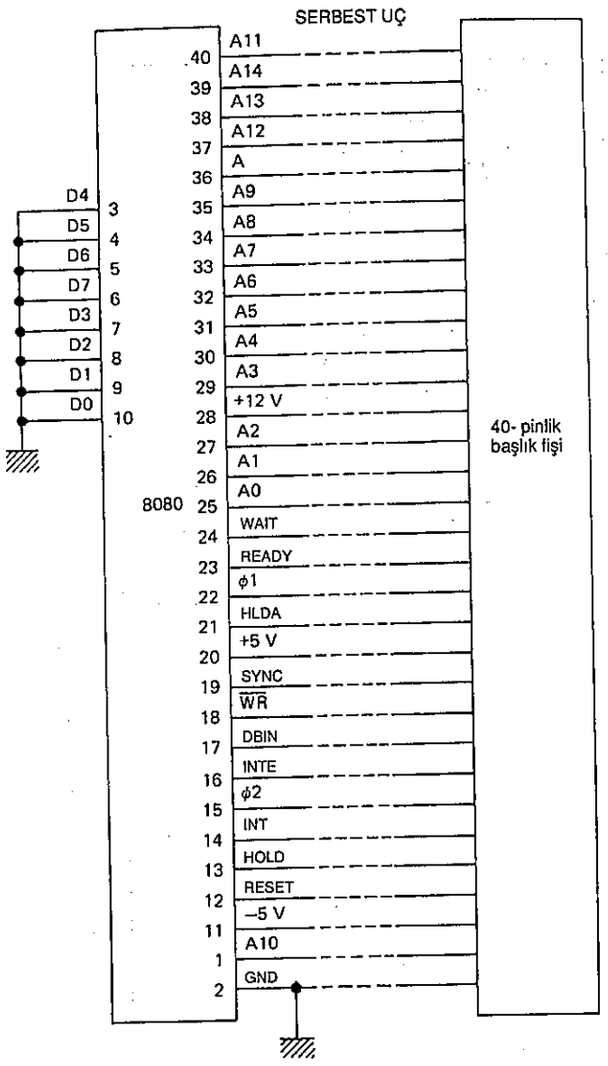
Hatalı sistemin Bölüm 12.4'te anlatıldığı biçimde görsel olarak incelendiğini varsaydıktan sonra, bazı başlangıç testleri yapılabilir. Bütün entegre devrelerin uygun pinlerinde doğru statik gerilimin bulunduğunu denetlemek için bir multimetre, ya da daha iyisi, bir osiloskop kullanılabilir. Yeterli bant genişliğine sahip bir osiloskkop kullanılarak sistem saati de incelenmelidir. Saat dalgabiçiminin yükselme zamanı, düşme zamanı ve darbe genişliği gibi önemli parametreleri, mikroişlemcinin imalatçının verdiği tanımlarla karşılaştırılarak denetlenmelidir. Bir mikroişlemci sistemde saati ya da başka herhangi bir dalgabiçimini incelemek için osiloskop kullanıldığı zaman, gözlenen sinyalin şekil değiştirmesine neden olabilen kapasitif yüklenmeyi önlemek için, yüksek empedanslı bir prob kullanılmalıdır.



Şekil 12.7

Başlangıç denetimlerini tamamladıktan sonra, eğer her şey tatmin edici bulunduysa, bir sonraki aşama, mikroişlemciyi bir "serbest-çalışma" durumuna getirmektir. Bu tekniği gözümüzde canlandırabilmek için Şekil 12.7'de gösterilen sistemi bir örnek olarak alacağız. Bu şekle bakıldığında, sistemde, program ve sabit veriyi saklayan iki ROM, iki RAM yongası, bir giriş portu ve bir çıkış portu bulunduğu görülecektir. Sistemin hatalı olduğunu ve programın yürütülemediğini varsayacağız.

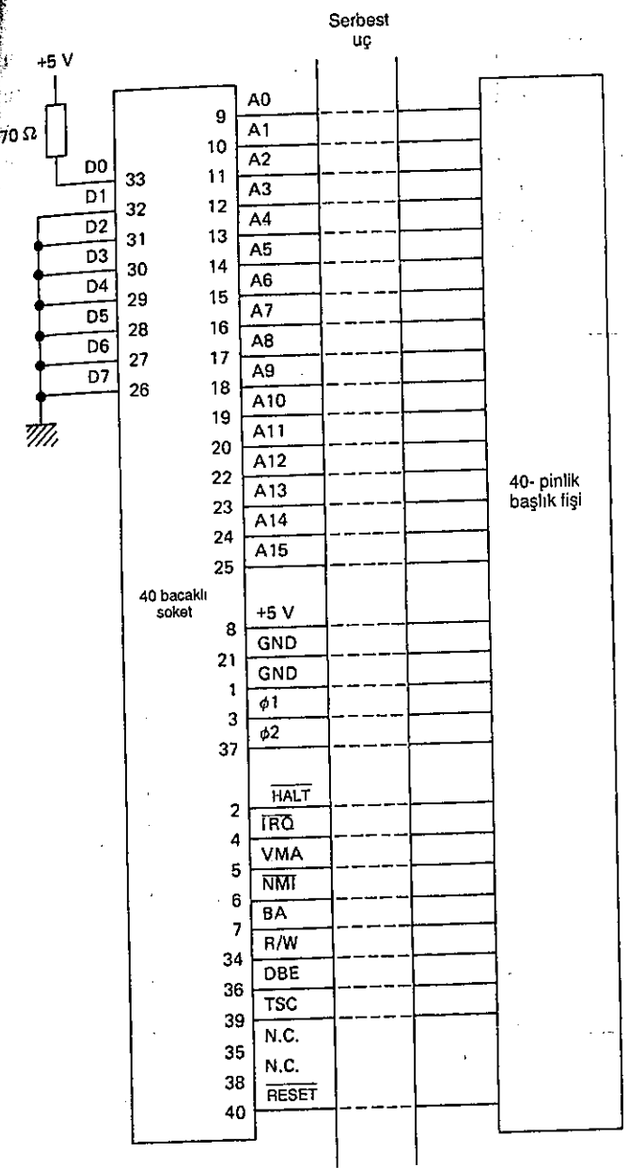
Mikroişlemciyi serbest-çalışma durumuna getirmek için, veri yolunu mikroişlemciye yakın bir noktada kırmak gereklidir, bu nokta Şekil 12.7'de "X" ile gösterilmektedir. Mikroişlemci veri hatları daha sonra, adres yolu işlemi içermeyen bir komut kodunu simgeleyen mantıksal düzeylerde tutulmalıdır. Böyle bir komut bellek okuma/yazma işlemi veya G/Ç işlemi gerektirmeyen bir komuttur. Mikro-



Şekil 12.8

işlemcilerin çoğunluğunun komut takımında, bu amaç için çok uygun olan bir "işlem-yok" (NOP) komutu bulunmaktadır. Ancak diğer komutlar da aynı amaca uygundur.

Veri yolundan araya girilmesi ve mikroişlemci hatlarına bir komut kodunun yerleştirilmesi, özel bir test kartının kullanılmasını gerektirir. Bu kartta mikroişlemciyi



Şekil 12.9 M6800 için test kartı

tutmak için bir soket ile serbest bir kablonun ucunda yer alan ve mikroişlemciyle aynı bacak düzenlemesine sahip olan bir başlık fişi bulunmaktadır. Test kartındaki mikroişlemci soketinin veri hatları, NOP'u ya da başka bir uygun komutu simge-

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Onaltılı
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0003
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0004
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0005
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0006
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0007
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0008
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	FFFE
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
										v.s						

Şekil 12.10 Serbest-çalışan mikroişlemciye ait adres yolundaki mantık düzeyleri

leyen mantık düzeylerine getirilir. Şekil 12.8'de 8080-tabanlı sistemlerin test edilmesinde kullanılan bir test kartı için bir devre gösterilmektedir. 8080 NOP komutu, 00 şeklinde bir işkoduna sahiptir. Bu nedenle Şekil 12.8'deki veri hatları topraklanmıştır. 6800 mikroişlemcisi için NOP kodu 01 olan bir test kartı, Şekil 12.9'da gösterilmiştir.

Test kartını kullanmak için Şekil 12.8'deki sistemde mikroişlemci yongası, soketinden çıkarılır ve test kartındaki sokete takılır. Daha sonra test kartının serbest ucunda yer alan 40-bacaklı başlık fişi sistem mikroişlemcisinin soketine yerleştirilir. Ayrıca, sistemde yer alan ve adres yoluyla herhangi bir şekilde bağlantısı bulunan bütün entegre devreler de çıkarılır.

Test edilen sistemin anahtarı açıldığı zaman, 8080 başlatma yordamı, program sayıcısının içeriği 0000'a getirilerek çalışmaya başlar. Daha sonra komut, normalde bu konumdan getirilerek yürütülür. Test kartı kullanılırken bir NOP komutu yürütümü gerçekleşecek ve program sayıcısının içeriği 0001'e arttırılacaktır. Bunun arkasından başka bir NOP komutu yürütümü gerçekleşecek ve program sayıcısının içeriği yeniden arttırılacaktır, bu işlem bu şekilde devam edecektir. Her bir komut saykılının getirme aşaması, program sayıcısı içeriğinin adres yoluna yerleştirilmesine neden olur. Bu işlem sürerken program sayıcısının içeriği, Şekil 12.10'da gösterilen değerler üzerinde bir döngüye girecek ve bu adresler adres yolunun hatları üzerinde sırayla uygun biçimde ortaya çıkacaklardır.

Dikkatimizi bu noktada herhangi bir adres hattı üzerinde yoğunlaştırırsak, osiloskopla incelemede bir kare dalga göreceğimiz açıktır. Ayrıca, An adres hattı üzerindeki kare dalga, A(n+1) adres hattındaki frekansın tam iki katı olan bir frekansa sahiptir. Bu, bütün adres hatlarını denetlemek için kolay bir yol sağlamaktadır. İlk önce, mikroişlemci yongasının pinleri üzerindeki sinyaller denetlenmelidir. Eğer adres hatlarından hiç biri etkin değilse bu, mikroişlemci yongasının ya da denetim hatlarından birinin hatalı olduğuna işaret eder. Örneğin sıfırlama hattı ve toprak arasında bir kısa devre bu etkiyi ortaya çıkarabilir. Eğer adres hatlarından yalnızca biri değişmeyen bir mantık düzeyindeyse bu, hat üzerinde bir yerde kaynak ya da toprağa bir kısa devre olduğunu gösterir.

Mikroişlemcinin adres yolu pinlerinin hepsinde doğru sinyallerin bulunduğu varsayılırsa, adres hatları üzerindeki sinyaller yol üzerindeki çeşitli noktalarda denetlenmelidir. RAM ve ROM gibi entegre devrelerin soketlerindeki adres hatları bu noktaların başında yer alır. Bir adres hattı üzerinde ilerlemekteyken eğer bir sinyal kaybolursa bu, bir açık devreyi işaret eder. Test bütünüyle iyi sonuç verirse, program ROM'larından başlayarak entegre devrelerin tümü, teker teker olmak koşuluyla yerlerine takılır. Adres hatları her entegre devrenin yerleştirilmesinden sonra, daha önceki gibi denetlenir. Entegre devrelerin hepsi takıldıkatan sonra adres hat sinyalleri yine mevcutsa o zaman, hatanın adres yoluyla ilişkili olmadığı kabul edilebilir. Bu yöntemle elbette, eğer varsa, adres hatlarındaki herhangi bir entegre devre kısa devresi bulunmuş olacaktır.

İlk program ROM'u takıldığı zaman, en azından sürenin bir bölümünde verinin ROM'lardan okunmasına neden olan komut getirme saykılları yürütüldüğünden, veri, sistem veri yolunda görünecektir. Sistem veri yolundaki veri mikroişlemciye ulaşmamaktadır, bunun nedeni test kartını kullanarak yola yerleştirdiğimiz yapay durdurma noktasıdır. Her bir ROM ayrı ayrı yerleştirildikçe osiloskop kullanarak her bir veri hattına bakılması yararlı olur. Her ne kadar osiloskop, Bölüm 12.7'de tartışılan nedenlerden dolayı veri dalgabiçiminin ayrıntılarını analiz edemez ise de, kalıcı olarak mantıksal 0 ya da 1 durumunda bulunan hatlar saptanabilir. Böyle bir durum kaynak ya da toprağa bir kısa devreyi, veya belki de bir hatalı ROM'un

bulunduğunu gösterir. Bir ROM'un takılmasından sonra, veri yolunda verinin bulunmaması da hatalı bir ROM yongasına işaret eder. RAM'ler normalde sistem anahtarı açıldıktan sonra rastgele veri içerirler ve bundan dolayı veri yoluna bun da, bir RAM yongasına ayrılan bellek alanı üzerinden, adres yolu çevrimleri gi veri üretmek durumundadır. Bu da yine hatalı RAM yongalarının belirlenebilm sini sağlar. Tek bir ROM ya da RAM'ın takılmasından sonra veri yolunda hiç v bulunmaması, ya hatalı bir bellek yongasını, ya da yonga yetkilendirme hattın enerjilendirilmesindeki bir arızayı gösterir. Bu da yine bir osiloskop kullana yapılacak incelemeyle doğrulanabilir.

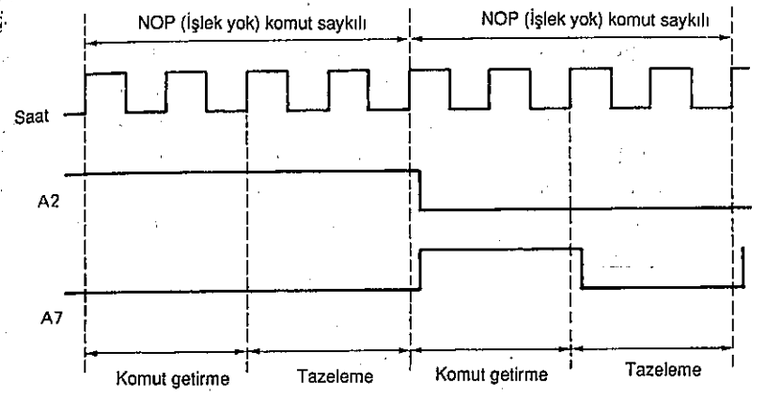
Mikroişlemci serbest çalışma durumunda bulunduğu zaman adresler olası bütü değerler üzerinde çevrim yaparken, bellek yonga seçimlerinin tamamı, sürenin b bölümünde belli bir süreyle enerjilendirilecektir. Yongaların hepsi ya da herhang birisi için durum böyle değilse, hata, mikroişlemciden gelen bir "kısa devre denetim hattında veya muhtemelen bir adres kod çözücü devresindedir. Bu noktad denetlenecek hatların başında $\overline{WR}$ ve SYNC hatları, ve bunlarla birlikte adres ko çözücülerindeki ara noktalarda bulunan ve yonga seçim sinyalleri gelir.

Burada anlatılan test uygulaması sırasında osiloskopta görüntülenen dalgabiçimle rinin yorumu söz konusu olan mikroişlemciye bağlıdır. Adres hatları, mikroişlemci serbest çalışma durumunda kare dalga taşıdığından, 8080 oldukça yerinde bir seçim olacaktır. Z80 ile dinamik belleğin tazelenmesi olanağı, durumu karmaşıklaştırır. Bu olanak yalnızca adres yolunun A0-A6 hatlarını kullanır. Ek B'de görülece gibi her bir komut saykılı sırasında bu hatlar üzerinde bir süre için bir tazelem adresi görünür. Z80 serbest çalışma durumundayken bu tazeleme adresi, her komu saykılı sırasında program sayıcısının değerinin arttırılmasının sonucu olarak ola bütün değerler üzerinde, aynı, yolda görünen adres gibi, çevrim yapar. Böylec tazeleme adresi ve komut getirme adresi A0-A6 hatları için tamamen aynıdır.

S12.10

Şekil 12.11'de, iki yürütüm saykılı sırasında A2 ve A7 hatlarında görünen dalga biçimlerini göstermektedir, bu yürütüm çevrimleri sırasında program sayıcısın içerikleri 007F ve 0080'dir. Bundan dolayı program sayıcısının 2 nolu biti 1'de 0'a, 7 nolu biti ise 0'dan 1'e değişir. A2 dalgabiçimi her zaman, 2 nolu program sayıcısı bitinin mantık düzeyini izlerken, A7'nin tazeleme periyodu boyunca h zaman düşük mantık düzeyinde olduğu, Şekil 12.11'de görülmektedir. Burad anlatılan etki normal bir osiloskop ile gözlenecektir. Ancak ayrıntıları, Şekil 12.11'deki ölçüde görmek için, çift zaman-tabanlı bir birim gereklidir. Ya d tek-vuruşlu modda çalışan bir bellekli osiloskobu da uygun olabilir.

Diğer mikroişlemcilerle başka sorunlar ortaya çıkmaktadır. 8085 gibi bazı birimler



Şekil 12.11 Z80 serbest-çalışma durumunda adres hattı dalgabiçimleri

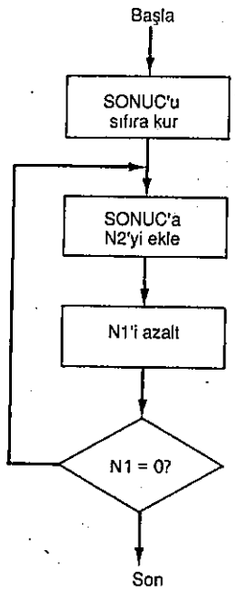
adres yolunun bir bölümüne ikili bir rol vererek kullanır. 8085 komut saykılının bir bölümü için, AD0-AD7 hatları bir adres taşır. Çevrimin geri kalan bölümü için ise bu hatlar, bir veri yolu gibi iş görürler. Serbest-çalışma durumunda adres yolu gözlemlerini yorumlarken, bunun dikkate alınması gerektiği açıktır. 8085 ile kullanılan test kartında, AD0-AD7 hatlarının ikili rolüne olanak tanımak için ek tamponlar bulunmaktadır.

Burada anlatılan hata bulma yaklaşımı, klasik, genel-amaçlı test donanımının kullanımı esasına dayanmaktadır. Mantık analizörü ve özellikle de işaret analizörü gibi özel-amaçlı cihazlar da, serbest-çalışma durumundaki mikroişlemci test edilmesine uygulanabilir. Bu cihazlar sahada bakımı için önemi açık olan hata teşhisi uygulamasını genellikle hızlandırırlar. Mantık ve işaret çözümlemesinin kullanımı Bölüm 13'te tam olarak anlatılmaktadır.

12.11 Yazılım hataları

Yazılımdaki hatalar temel olarak program tasarım yanlışlarıdır. Dolayısıyla bunları, örneğin eleman arızasından kaynaklanan donanım hatalarıyla doğrudan doğruya karşılaştırmak mümkün değildir. Buna karşın hatanın yol açtığı etki aynı olabilir, örneğin sistem, kullanıcı tarafından yararlanılamaz duruma gelir.

Sağduyulu hiç bir imalatçı, açıkça görülen yazılım hataları bulunan mikroişlemci-tabanlı bir sistemi, alıcısına vermez. Yazılımı görünüşte tatmin edici olan, fakat yeterli ölçüde test edilmemiş bir sistemin alıcıya verilmesi daha muhtemel bir durumdur. Bazı yazılım hataları yalnızca belirli bir veri birleşimi ile çalıştı-



Şekil 12.12 Veriye-bağlı hata içeren programın akış diyagramı

rılırken ortaya çıkar. Dolayısıyla bir programın, bir birleşimi hatayı tetiklemeden önce, oldukça uzun bir tatmin edici biçimde çalışması mümkündür.

Şekil 12.12'de, böyle bir hata içeren bir programın akış diyagramını göstermektedir. Aynı program Şekil 12.13'te, bir de-yüksek-düzey dil kullanılarak tanımlanmış ve program 8080 mikroişlemcisi için kodlanan assembly listesi Şekil 12.14'te gösterilmiştir. Program N1 ve N2 simgesel adreslerinde saklanan 8-bitlik iki sayıyı çarpmakta ve sonucu bir kaydedici çiftinde bırakmaktadır. Tekrarlanan toplama algoritması kullanılmaktadır. Okuyucu bu örneği, iyi bir programlama uygulaması modeli olarak kullanabilir.

```
SONUC = 0
DO UNTIL N1 = 0
        SONUC = SONUC + N2
        N1 = N1 - 1
END DO
END
```

Şekil 12.13 Veriye-bağlı hata içeren programın sözde-kodu

```
----- 8080/85 ASSEMBLER V01 ------------------------------ PAGE 1
                                                            FUTUREDATA ---------
                 * ****************************************** *
                 *  TEKRARLANAN TOPLAMA YOLUYLA ÇARPMA        *
                 *  N1, N2 İKİ İŞLENENİN SEMBOLİK ADRESLERİDİR *
                 *  SONUÇ, KAYDEDİCİ ÇİFTİ HL'DE BIRAKILIR.   *
                 *                                            *
                 *  BU PROGRAM HATALI VERİLER İÇERİR          *
                 *  N2 = 0 OLDUĞUNDA SONUÇ, YANLIŞTIR         *
                 *  ****************************************** *

0000                          RSEG    TEC9        SONUÇ = 0
0000  210000                  LXI     H, 0
0003  3A1100                  LDA     N1
0006  4F                      MOV     C, A        N1, C'DE TUTULUR
0007  0600                    MVI     B, 0        0, D'DE TUTULUR
0009  3A1200                  LDA     N2          N2, A'DA TUTULUR
000C  09          TEKRAR      DAD     B           SONUÇ'A N2'YI EKLE
000D  3D                      DCR     A           N2'YI AZALT
000E  C20C00                  JNZ     TEKRAR
                  *
                  *
0011              N1          DS      1           İŞLENEN N1
0012              N2          DS      1           İŞLENEN N2
0013                          END
```

Şekil 12.14

Şekil 12.14'te gösterilen program, çoğunlukla mükemmel düzeyde iyi çalışmaktadır. Ancak kısa bir denetim sonunda, N2'nin sıfır olduğu her durumda yanlış bir sonucun elde edileceği görülecektir. Bu program özel bir mikroişlemci sistem yazılımının bir bölümü olarak kurulursa, hatanın hiç bir zaman ortaya çıkmaması bile mümkündür. Kullanıcıların çoğunluğu bu ölçüde şanslı olmayacak ve zaman zaman yanlış sonuçlar elde edeceklerdir. Yazılım hataları saptanır saptanmaz program değiştirilmelidir. Yukarıda anlatılan programdaki hata, 8080 assembly dilini destekleyen bir yazılım geliştirme istasyonu yardımıyla düzeltilebilir. Donanıma-bağımlı programlardaki hatalar için, devre-içi emülasyon olanağı bulunan bir MDS, daha uygun bir araçtır.

Kaynaklar ve bibliyografya

12.1 J.D. Lenk, *How to Troubleshoot and Repair Microcomputers*, Reston Publishing Co., Inc., USA (1980).

12.2 *Using Delayed Sweep in Measuring Digital Word Trains, Tektronix* T900 Application note 41G1.0 (1976).

Sorular

12.1 Bir mikroişlemci sistemin arızası için beş olası neden sıralayınız.

12.2 Lehim köprüleri ve kablo artıklarının bir mikroişlemci sistemde arızalara nasıl neden olduklarını açıklayınız.

12.3 Lehim köprülerinin saptanmasında en maliyet-verimli yol genellikle hangisidir?

12.4 Bir mikroişlemci sistemde statik gerilimleri test etmek için osiloskobun, bir testmetresinden neden genellikle daha tatmin edici olduğunu açıklayınız.

12.5 Mantık problarının işlevini anlatınız.

12.6 Mantık problarının darbe-uzatma olanağını kullanan bir mikroişlemci sistemde geçekleştirilebilen test için bir örnek anlatınız.

12.7 Klasik bir osiloskobun, mikroişlemci sistemlerinin test edilmesinde, diğer analog ve sayısal elektronik sistemlerin test edilmesine göre genellikle daha az yararlı olmasının nedenini anlatınız.

12.8 Osiloskobun, bir mikroişlemci sistemde saat dalgabiçiminin denetlenm
için, örneğin mantık analizöründen daha uygun bir cihaz olmasının neden
açıklayınız.

12.9 Bölüm 10'da anlatılan sistemde, Şekil 10.13'te gösterilen RAM'de bir h
bulunduğundan kuşkulanılmaktadır. $\overline{CS}$ ve $R/\overline{W}$ sinyallerinin iki 2112 RA
yongasına ulaşıp ulaşmadığını denetlemek için bir test programının n
kullanılabileceğini, test cihazı olarak klasik bir osiloskop kullanımını e
alıp açıklayınız.

12.10 Bir mikroişlemci sistemin serbest-çalışma testi ile ne anlatılmak istendiğ
açıklayınız.

Bölüm 13 Özelleşmiş Test Donanımı

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde;*

1 *Hata-bulma için mantık ve işaret analizörü ile teşhis programlarının kullanımını açıklayabilmeli,*
2 *Mikroişlemci sistemlerdeki arızaların yerlerinin belirlenmesi için mevcut olan geleneksel ve özelleşmiş yöntemleri karşılaştırabilmelisiniz.*

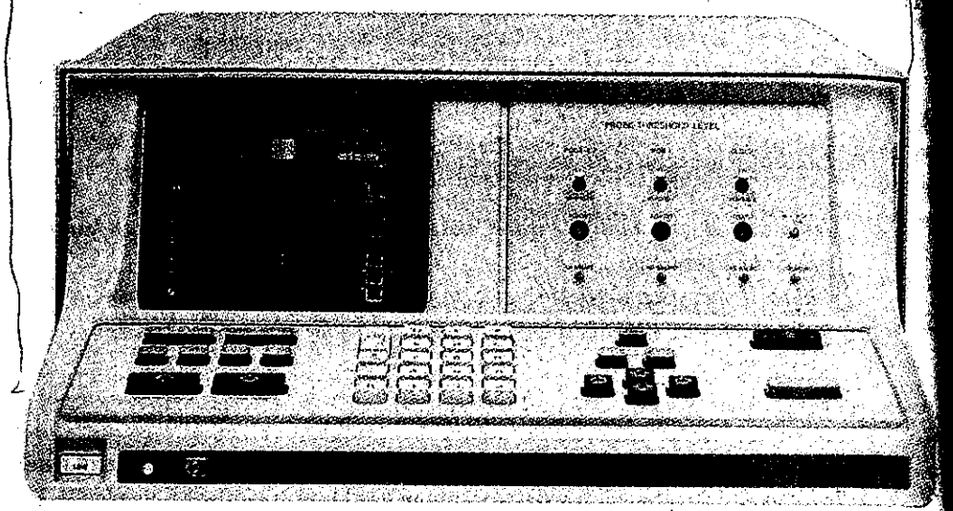
13.1 Giriş

Bölüm 12'de mikroişlemci sistemlerde hata teşhisi için multimetre ve katot ışınlı osiloskop gibi genel-amaçlı test donanımının nasıl kullanılabileceği incelendi. Bu bölümde ise bazı özel-amaçlı test cihazlarının çalıtırılmasındaki ilkeler ve yöntemler tartışılacaktır. Bu cihazlar, ki bunların başında mantık analizörü ve işaret analizörü gelmektedir, mikroişlemcilerle çalışan geliştirme ve servis mühendislerinden gelen ve giderek artan isteklerin karşılanması için kullanıma sunulmuşlardır ve diğer cihazlara göre daha yeni cihazlardır.

13.2 Mantık analizörü

Mantık analizörü, öncelikle, yol düzeni bulunan sistemlerdeki veri akışını gözlemeye olanak tanımak için kullanıma sunulmuş bir cihazdır. Çok büyük çoğunlukla tek başına çalışabilir özelliktedirler. Tipik bir örnek Şekil 13.1'de gösterilmektedir. Şekil 13.2'de, başka bir alternatif olarak, fişle takılabilen özel mantık analizörüne bağlı bulunan normal bir osiloskop gösterilmektedir. Bu tür gözlemler yapılırken osiloskopların kullanılmasında karşılaşılan güçlükleri Bölüm 12'de tartışmıştık. Osiloskoptan farklı olarak, mantık analizörü temelde bir kayıt cihazıdır, gözlenen sinyalleri düzenli aralıklarla örnekler ve bu örnekleri sak..r. Saklanan örnekler daha sonra çeşitli biçimlerde görüntülenebilir.

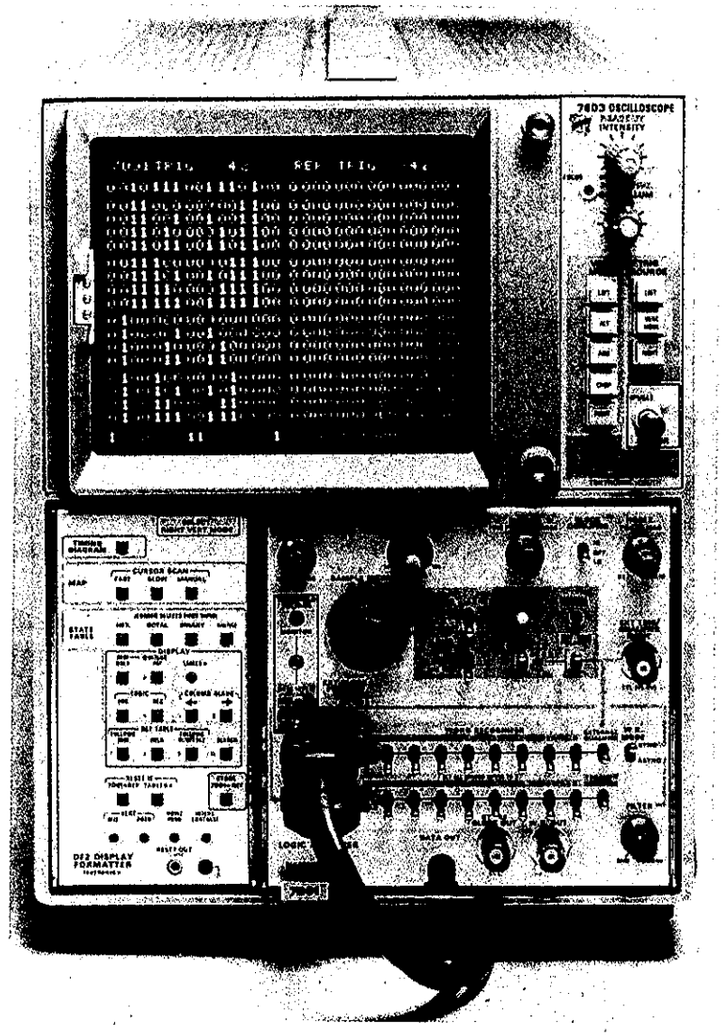
Her ne kadar mantık analizörleri mikroişlemcilerdeki olayları gözleme konusunda osiloskoplara oranla birçok avantaj sunuyorlarsa da bunların bazı dezavantajları da bulunmaktadır. Gerilim düzeyleri kabaca "yüksek" veya "alçak" biçiminde kaydedildiği için sinyal düzeylerindeki değişikliklerin ince ayrıntıları gözden kaçmaktadır. Bu nedenle yükselme ve düşme süresi gibi değerleri ölçmek için osiloskop en uygun cihaz olma özelliğini korumaktadır.



Şekil 13.1 Hewlett Packard model 1615A bağımsız çalışabilir mantık analizörü

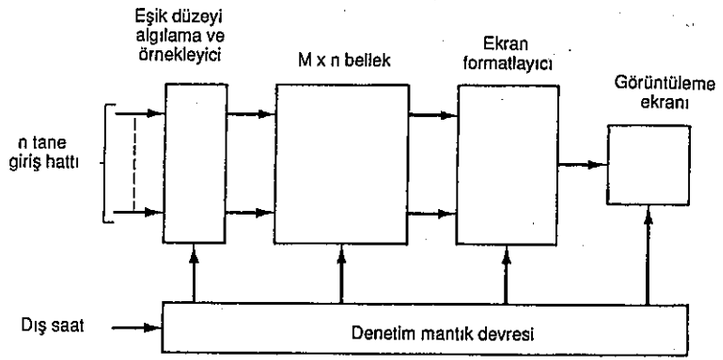
Şekil 13.3'de, bir mantık analizörünün blok diyagram olarak şematik tanımı gösterilmiştir. Düzenli aralıklarda, bir saat, "n" giriş hatlarının her birinde bir örnekleme işlemi başlatır. Daha sonra her giriş hattındaki gerilim, bir eşik düzeyi ile karşılaştırılır ve bu eşik düzeyin altında ya da üstünde olmasına bağlı olarak 0 veya 1 şeklinde sınıflandırılır. Bu işlem, Şekil 13.4'de gösterilmektedir.

Bilgisayar tarafından kullanılan eşik düzey, sinyalleri üreten mantık birimlerinin tipine bağlıdır. Tanınmış 74LS serisinden TTL entegre devrelerini örnek olarak alırsak, bunların özellikleri, bir çıkışın alçak mantık düzeyi olan 0'da bulunması durumunda bunun en yüksek geriliminin 0.5 olduğunu bildirmektedir. Çıkış yüksek mantık düzeyi olan 1 düzeyinde bulunduğunda ise en düşük gerilim 2.7 V'dur. Bu durumda bu iki düzey arasında ortalarda bulunan 1.1 V gibi bir eşik değeri uygun olacaktır. Mantıksal unsurların diğer tipleri farklı eşikler gerektirecektir. Örneğin 15 V'luk bir kaynakla çalışan CMOS, 7.5 V'luk bir eşik değerine gerek duyacaktır. Şekil 13.3'te gösterilen mantık analizöründeki "n" karşılaştırıcılarının çıkışları bir n-bitlik ikili sözcükten oluşur, bu sözcük M analizörünün 0 konumunda, n-bitlik bellek tarafından saklanır. Ayrıca beleğin daha önceki içeriklerinin hepsi bir konum kadar kaydırılır. Böylece 1 adresinin daha önceki içerikleri artık 0 adresinde, 2 adresinin daha önceki içerikleri artık 1 adresinde saklanır ve kaydırma işlemi bu şekilde diğer adreslere de uygulanır. Belleğin en üstündeki en yüksek adrese daha önce saklanmış olan sözcük ise atılır.

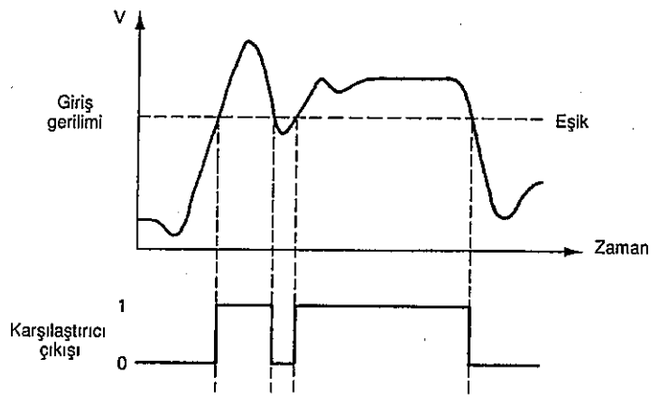


Şekil 13.2 Textronics model 7603 osiloskopla bağlantılı mantık analizörü

Herhangi bir anda bellek, daha önceki M örnekleme periyodlarında giriş hatlarındaki mantıksal düzeylerin bir kaydını içerir. Her yeni örnek alınışında bu kayıt güncelleştirilir. Örnekleme işleminin herhangi bir anda durdurulması kaydı etkin biçimde "dondurur". Bu durumda mantık analizörü belleğinin içeriği, giriş hatlarının mantıksal düzeylerinde, örneklemenin durdurulduğu ana kadar gerçekleşen değişikliklerin bir "anlık" durumundan oluşur. Bu kaydı elde ettikten sonra analizör, veriyi çeşitli biçimlerde görüntüleyebilir.



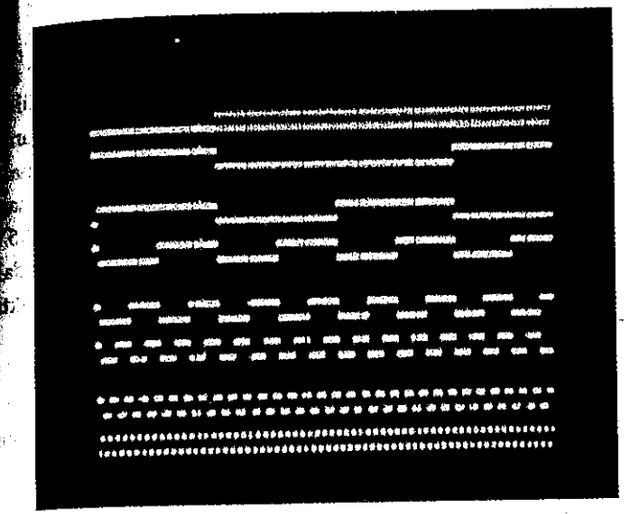
Şekil 13.3 Mantık analizörü



Şekil 13.4 Eşik algılama

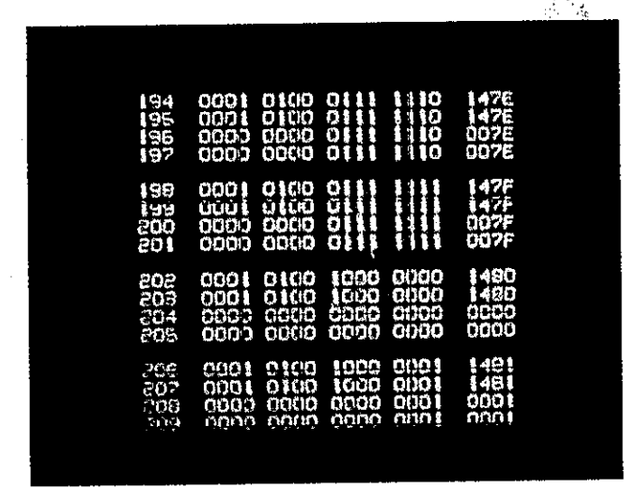
Örnekleme işleminin durdurulması çeşitli yollarla gerçekleştirilebilir. En kestirme yöntem, dışsal olarak üretilen bir tetikleyici ya da eşzamanlama sinyali kullanmaktır. Analizörlerin çoğunluğunda yüksekten-alçağa veya alçaktan-yükseğe geçişte seçmeli olarak çalışabilen bir tetikleyici girişi bulunmaktadır. Ancak en yararlı tetikleme olanağı analizörün bir tetikleme olayını tanımasını sağlayan olanaktır. Kullanıcı, genellikle analizörünün ön panelinde bulunan anahtarlarla n bitlik bir tetikleme deseni düzenler. Her örnekleme zamanında giriş verisi tetikleyici modeli ile karşılaştırılır. Eşleşme belirlendiğinde örnekleme işlemi durdurulur.

Tetikleme olanaklarının bir başka yönü de mantık analizörünün istenirse, örnekleme işleminin, tetikleme olayı algılandıktan sonra bir periyod süresince sürdürül-



Şekil 13.5 Mantık analizörü zaman ekseni görüntüsü

mesine olanak tanıyabilmeleridir. Bu durumda saklanan veri, tetikleme olayının hemen arkasından gelen ve işlemlerin devamını sağlayan bir olaylar dizisidir. Bu elbette, veri saklamaya yalnızca tetikleyici olayın ortaya çıkmasından sonra başlayan saklama osiloskobu tarafından yakalanamayan verinin elde edilmesini sağlar. Kullanıcı, mantık analizörü ile tetikleme olayından sonra alınan örneklerin sayısını da seçebilir.



Şekil 13.6 Mantık analizörü veri ekseni görüntüsü

Şekil 13.5'de, saklanan verinin bir görüntülenme biçimini göstermektedir. Bu bir zaman ekseninde görüntüdür. Görüntü, çok kanallı osiloskopla elde edilen görüntüye benzemektedir. Ancak okuyucu mantık analizörünün, osiloskopla görülebilen gerilim değişikliklerinin ince ayrıntısını kapsamayıp, yalnızca 1 ve 0 mantıksal düzeylerini görüntülediğini unutmamalıdır. Şekil 13.5'teki görüntü, 16-kanallı bir mantık analizörü tarafından üretilmiştir. Zaman ekseninde görüntünün karmaşıklığını önlemek için kullanıcı, bazı kanallara ait izleri engelleyebilir. Şekil 13.5'teki 1 den 16 ya kadar olan kullanılabilir kanalların yalnızca 8 ile 16 arasındakiler görüntülenmiştir, 16 nolu kanal en altta ve 9 nolu kanal en üstte yer almaktadır. Görüntünün sol tarafındaki düşey çizgi, işlevi aşağıda anlatılan bir imleçtir.

Şekil 13.6'da, Şekil 13.5'te gösterilen zaman ekseninde görüntüyü oluşturan saklanmış bilginin bir bölümüne ait veri ekseni görüntüsü gösterilmektedir. Burada kullanılan cihaz ile, veri ekseni her zaman 16 kanalın tamamını gösterir, sağ taraftaki sütunlar örnek numarasını, sol taraftakiler ise 16-bitlik ikili sözcüğün onaltılı biçimini verir. Cihaz ayrıca, Şekil 13.6'da gösterilen onaltılı gösterim yerine veri sözcüğünün sekizli gösterimi seçeneğini de sunmaktadır.

Şekil 13.6'daki veri ekseninde görüntü analizörünün 16 giriş hattında 194- 209 şeklinde numaralanmış 16 örnekleme noktasındaki mantıksal düzeyleri göstermektedir. Söz konusu cihaz, veriyi, toplam 512 örnekleme noktasından alarak saklamaktadır. Ön panelde yer alan bir elle kontrol olanağı ile kullanıcı, saklanan örnekler üzerinde hareket edebilir. Zaman ve veri ekseninde görüntüler arasındaki ilişki zaman ekseninde görüntünün imleci tarafından belirlenir. İmlecin konumu veri ekseninde görüntüde en üstteki örneğe uyar.

Giriş hatlarının örneklenme zamanlarını belirlemek için iki mekanizma seçeneğinden birisi kullanılabilir. Eşzamanlı örnekleme, test edilen sistemden gelen bir eşzamanlama sinyalini kullanır. Bir mikroişlemci için bu çoğunlukla sistem saatidir.

Mantık analizörü tarafından saklanan veri daha sonra her saat çevrimi sırasında, analizörün giriş hatlarındaki mantık düzeyleri olacaktır. Şekil 13.6 ve Şekil 13.5'te gösterilen görüntüler bu şekilde elde edilmektedir. Bazı mikroişlemci imalatçıları, çeşitli komutlar yürütülürken her saat aralığı süresince veri ve adres hatlarındaki mantıksal düzeylere ilişkin tablolar oluşturmaktadır. Dolayısıyla eşzamanlı örnekleme ile, adres ve veri hatları üzerindeki mantıksal düzeylerin beklenen şekilde gerçekleşip gerçekleşmediğini görmek kolay olmaktadır. Daha gelişmiş bazı mantık analizörleri bu olanağı, saklanan örnek sözcükleri belli bir mikroişlemcinin makine-kodu komut kısaltmalarına dönüştürerek daha da genişletir. Eğer mantık


```
                ADDRESS   DATA   EXTERNAL

TRIGGER          000F

                          PRE-TRIGR=7
                                  LINE    0
ADRS     OPCODE/DATA          EXTERNAL
0009     LD C,B               0000 0000
000A     LD L,00              0000 0000
000C     LD H,D               0000 0000
000D     LD A,(HL)            0000 0000
0400      00   READ           0000 0000
000E     LD H,B               0000 0000
000F     LD (HL),A            0000 0000
3400      00   WRITE          0000 0000
0010     INC L                0000 0000
0011     JP NZ,000C           0000 0000
000C     LD H,D               0000 0000
000D     LD A,(HL)            0000 0000
0401      31   READ           0000 0000
000E     LD H,B               0000 0000
000F     LD (HL),A            0000 0000
3401      31   WRITE          0000 0000
```

Şekil 13.7 Veri yollarındaki verinin tersine çevrilme işlemi (inverse assembly), yedi özel modülle mümkündür. Bu komut kısaltması kodlarının görüntüsü, kolayca yorumlanabilmesi için tanıdık çevirici (assembler) formatında verilmiştir.

analizörü, her komut çevriminin getirme aşaması sırasında örnekleme yapmak üzere tetiklenirse, veri yolunda arka arkaya ortaya çıkan komut kodları yakalanabilir ve görüntülenebilir. Buna bir örnek olarak, Ek B'deki Şekil 4.0-1'de Z80 mikroişlemcisinin, her komut getirme aşamasında etkin durumda olan bir M1 hattının bulunduğunu görmekteyiz. Bu hattaki sinyali bir mantık analizörünü tetiklemek için kullanmak, ardışık komut kodlarının örneklenmesine olanak verir. Şekil 13.7'de bu komut kısaltması kodlarını çözme olanağını sunan bir mantık analizöründen elde edilen görüntü gösterilmektedir. Kısaltma kodunun çözülmesi olanağının bulunması, mantık analizörünün yalnızca bir mikroişlemci çeşidi ile çalışması için tasarımlandığını ortaya koyar. Pratikte cihaz, farklı mikroişlemcilere uyum sağlamak için, bir dizi özel modülleri bulunan bir genel-amaçlı anabilgisayardan oluşur.

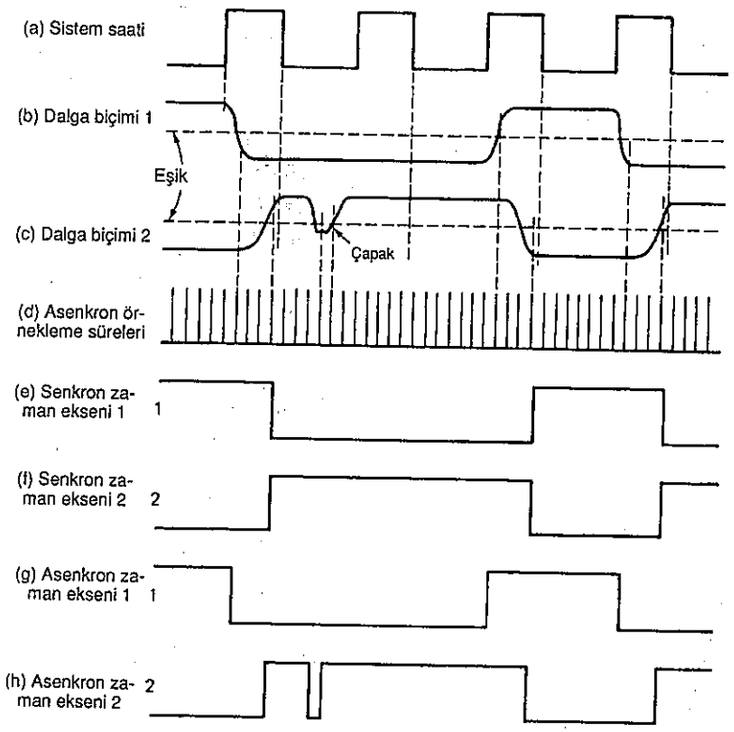
Mantık analizörünün eşzamanlı çalışmasıyla elde edilen en yüksek örnekleme hızı test edilen sistemin her saat çevriminde bir örnektir. Test edilen sistemde bir hata durumu varsa, mantıksal düzeylerinde saat periyodu sırasında değişikliklerin ortaya çıktığından kuşkulanılabilir. Bu çeşit kısa geçişler "çapak" olarak bilinmektedir. Bunların ayırdedilmesine yardımcı olmak için asenkron örnekleme kullanılabilir.

Asenkron örneklemede mantık analizörü içindeki bir osilatör, örnekleme zamanlarını sabitleştirir. Bu ise, kullanıcının örnekleme frekansını seçmesine olanak verir, örnekleme frekansı, gelişmiş cihazlarda 100 MHz değerine kadar çıkabilir. Test edilen sistemin saat frekansından daha yüksek bir örnekleme hızı ile veri ve adres

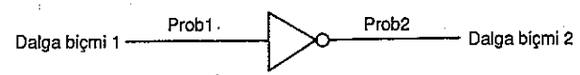
hatları her saat aralığı süresince birkaç kez örneklenebilirler. Asenkron örnekleme
S13.1 ayrıca sistem saati dalgabiçiminin kendisinin de incelenmesine olanak verir.

Bir mantık analizörünün en etkin şekilde kullanılma yolu, eşzamanlı örnekleme ile hataların yaklaşık olarak nerede olduklarının bulunmasıdır. Bunun ardından bir çapağın kesin konumu, daha yüksek bir örnekleme hızında asenkron olarak çalıştırılarak belirlenebilir.

Şekil 13.8'de, eşzamanlı örnekleme yapılırken görünmeyen dalgabiçimleri ayrıntılarının, asenkron örnekleme ile ne kadar hızlı bir biçimde elde edildiği gösterilmektedir. Tersleyici bir kapının giriş ve çıkışındaki dalgabiçimlerini incelemek için Şekil 13.9'da gösterildiği gibi bir mantık analizörünün iki kanalı kullanılmaktadır. Söz konusu dalgabiçimlerinin gerçek halleri Şekil 13.8'de *(b)* ve *(c)*'de gösterilmektedir. Eşzamanlı örnekleme, sistem saatinin yüksekten-alçağa geçişinde ortaya çıkar, bu da *(e)* ve *(f)*'deki 1 ve 2 nolu dalgabiçimlerinin zaman eksenindeki görüntüleriyle sonuçlanır.



Şekil 13.8 Mantık analizörünün senkron ve asenkron örneklemesi



Şekil 13.9 Mantık analizörünün prob bağlantıları

Hızlı asenkron örnekleme ile üretilen zaman ekseninde görüntü *(g)* ve *(h)*'de gösterilmektedir. Kapı giriş ve çıkış dalgabiçimleri arasındaki gecikmenin burada ortaya çıktığı görülmektedir. Eşzamanlı örnekleme ile hiç görünmeyen 2 nolu dalgabiçimindeki çapak da bu görüntüde belli olmaktadır. Mantık analizörlerinin çoğunluğunda aslında, "çapak yakalama" için bir sinyalin geçişleri tarafından tetiklenen bir mandalı kullanan başka olanaklar bulunmaktadır. Eşzamanlı işlemlerde bile mandallama modu olarak adlandırılan bu olanağın kullanılması, bulunduğu konumun saptanmasındaki hassaslık daha az olsa da, 2 nolu dalgabiçimindeki
S13.3 çapağı algılayabilmektedir.

Şekil 13.3'te gösterilen görüntü, 16-kanallı bir mantık analizörü tarafından üretilmiştir. Daha anlaşılır olması için yalnızca sekiz kanal etkinleştirilmiştir. Görüntüdeki sekiz iz, serbest çalışma modundaki bir Z80 mikroişlemcisinin A0-A7 adres hatlarında bulunan aktiviteyi görüntüler.

A0 adres hattı diyagramın tabanında, A7 ise en üstündedir. A7 dalgabiçiminin A0-A6 hatlarından farklı özellikte olduğu görülecektir. Bu, A0-A6 hatlarının, bölüm 12.10'da tartışıldığı gibi, her bir komut yürütüm çevriminin bir bölümü için bir tazeleme adresi taşımalarından kaynaklanmaktadır.

Mantık analizörlerinin çoğunluğunda, zaman ekseninde görüntülemenin genişletilmesi de dahil, daha gelişmiş başka olanaklar bulunmaktadır. Şekil 13.10'da imleç dikey bir hat olarak görülmektedir. Şekil 13.5, aynı verinin bir görüntüsünü göstermektedir fakat burada veri, imlecin sağ tarafına doğru beş kez genişletilmiştir. A0-A7'ye kıyasla A7 dalgabiçiminin özelliğindeki farklılık burada daha açık bir biçimde görülebilmektedir.

Bir mantık analizörü yardımıyla yol aktivitesi kayıtlarını elde eden kullanıcı daha sonra, yorum sorunu ile karşı karşıya kalır. Mikroişlemcilerin çoğunluğunun imalatçı bilgi sayfalarında, her bir komutun yürütümü sırasında veri ve adres yollarında ne görmeyi bekleyebileceğimize ilişkin bilgiler bulunmaktadır. Bunun en iyi örneği Ek C'deki Tablo 9'da yer almaktadır. Bu tablo, komut takımındaki her bir komutun yürütümü gerçekleştikçe her bir saat çevrimi sırasındaki veri ve adres yolu aktivitesini göstermektedir. Veri yolundaki 8-bitlik sözcüğün mikroişlemci tarafından

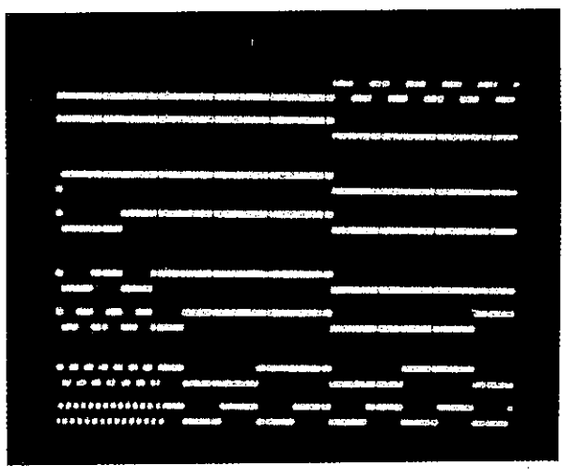
her zaman saptanamayabileceğine ve bundan dolayı bazı komutlar için kesin biçimde her zaman tahmin edilemeyeceğine dikkat edilmelidir.

Şekil 13.11'de, aşağıdaki iki komutun yürütümünden sonra görülebilecek bir mantık analizörü zaman ekseninde görüntü gösterilmektedir. Eşzamanlı örneklemeyi yapıldığı kabul edilmiştir.

```
LDX   #X'1000'
STX   X'1027'
```

Bu iki komut bellekte 2000 konumundan başlayarak saklanır. Birinci komut X kaydedicinini onaltılı 1000 sayısı ile yükler. İkinci komut ise X kaydedicinin 16-bitlik içeriğini 1027 ve 1028 adreslerine saklar. Anlaşılır olması açısından bu komutlar için zamanlama bilgisi Tablo 13.1'de verilmiştir (Ek C'deki Tablo 9'dan alınmıştır.).

Şekil 13.11'deki zaman ekseninde görüntü 16 giriş hattı bulunan bir mantık analizörü tarafından üretilmiştir. Bunlardan en küçük değerlikli olan sekiz tanesi A7-A0 adres yolu hatlarına bağlanmışlardır. En ağırlıklı olan sekiz tanesi D7-D0 veri yolu hatlarına bağlanmışlardır. Verinin okunabilir onaltılı gösterimini sağlayabilmek için giriş hatlarının yol hatlarına bağlanma sırası önemlidir.



Şekil 13.10 Mantık analizörünün, imlecin sağına doğru iz genişletilmiş zaman ekseni görüntüsü

Şekil 13.11

Zaman ekseninde görüntü																Onaltılı		
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	28	00	
0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	27	10	STX X'1027'
0	0	1	0	0	1	1	1	0	0	1	0	0	1	1	1	27	27	
0	0	0	0	0	1	0	0	0	0	1	0	0	1	1	1	05	27	
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	04	10	
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	03	FF	LDX #X'1000'
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	02	00	
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	01	10	
0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	0	00	CE	
A7–A0								D7–D0										

Tablo 13.1 *LDX ve STX komutları için M6800 komut saykılı özeti*

Komut	Saykıl	Adres yolu	Veri yolu
LDX	1	İşkodu adresi	Opkod adresi
(dolaysız	2	İşkodu adresi + 1	İşlenen veri EBDB'i
adresleme)	3	İşkodu adresi + 2	İşlenen veri EKDB'i
STX	1	İşkodu adresi	Opkod adresi
(genişletilmiş	2	İşkodu adresi + 1	İşlenen adresinin EBDB'i
adresleme)	3	İşkodu adresi + 2	İşlenen adresinin EKDB'i
	4	İşlenenin adresi	İlgisiz veri
	5	İşlenenin adresi	İşlenen veri EBDB'i
	6	İşlenenin adresi + 1	İşlenen veri EKDB'i

Şekil 13.11'deki görüntü, mantık analizörünün giriş sözcüğü 2800 olarak algılandığında tetiklenmesiyle üretilmiştir. Bu nedenle Şekil 13.11'de en üstteki satır en son saat çevrimini gösterir, aşağıya inildikçe daha önceki saat çevrimleri birbiri ardından sırayla gösterilir.

Görüntünün en alt satırından başlarsak, A7-A0=00 dır, bu elbette işkodu adresi 2000 'in en küçük değerlikli sekiz bitidir. D7-D0 veri hatları LDX işkodu olan CE'yi taşır. Bir sonraki saat darbesi sırasında, görüntünün bir sonraki en yüksek satırı, A7A0'ın, 2001'in en küçük değerlikli sekiz biti olan 01'i taşıdığını göstermektedir. Bunun doğru olduğunu Tablo 13.1'den bakarak denetleyebiliriz.

Tablo 13.1 bize ayrıca, veri yolunun bu kez işlenenin en büyük değerlikli baytının adresini taşıması gerektiğini göstermektedir. İşlenen 1027 olduğundan veri yolu bu

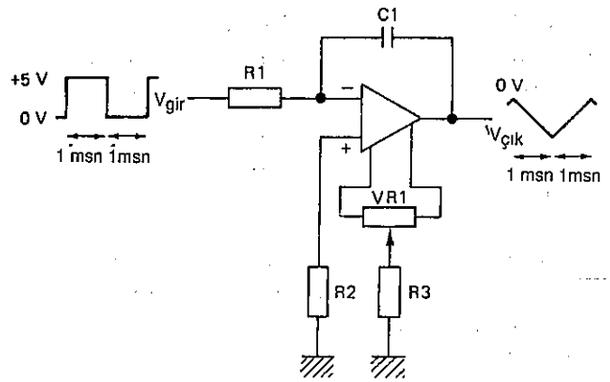
nedenle, görüntüde gösterildiği gibi, 10'u taşır. Dolayısıyla görüntü satır satır incelenerek sistemin doğru çalıştığı denetlenebilir. Beklenen çalışma biçiminden herhangi bir sapma olursa bu, programın yeniden çalıştırılması ve bozukluğun saptandığı bölgede daha yüksek bir hızda asenkron örnekleme yapılması ile, daha yakından araştırılabilir.

13.3 İşaret analizörleri

Her ne kadar mantık analizörü laboratuar ortamında hata teşhisi için kullanılabilecek en güçlü araçlardan biri ise de, bu cihazın sahadaki test mühendisi açısından belli bazı dezavantajları bulunmaktadır. Güçlüklerden birisi cihazın elde ederek kullanıcıya sunduğu bilgi hacminin küçüklüğüdür. Örnek olarak en çok 512 örnek saklayabilen 16 giriş hatlı ortalama bir mantık analizörünü alırsak, bu cihaz 8.192 bitlik bir verinin incelenmesine olanak tanır. Veri elverişli bir biçimde sunulmaktadır, ancak yine de bir hata durumunun belirlenmesi için çözümlenmesi gerekir. Mantık analizörünün sistem adres ve veri yolları boyunca birkaç noktadaki veri akışını gözlemek için kullanılması gerekebilir, böylelikle ek veri hacmi elde edilir.

Mikroişlemci-tabanlı sistemlerle ilgilenen saha servis mühendisinin karşılaştığı özel sorunları göstermek için bu durumu, analog elektronik sistemlerin servisiyle ilgilenen bir mühendisin sorunlarıyla karşılaştırabiliriz. Analog sistemlerde standart hata belirleme araçları multimetre ve osiloskoptur. Mühendis, bu cihazları kullanarak devre üzerindeki sinyalleri izleyebilir. Servis elkitapları çoğunlukla, doğru biçimde çalışırken devredeki çeşitli düğümlerde ortaya çıkan dalgabiçimlerinin biçim ve genliklerini gösteren devre diyagramları içerir. Şekil 13.12'de bunun bir örneği yer almaktadır. Şekil 13.12'de gösterilen devrenin tanımladığı sistem bölümünün düzgün biçimde çalışıp çalışmadığını denetlemek çok kolaydır. Servis mühendisi bir osiloskop kullanarak giriş ve çıkış dalgabiçimlerinin biçim ve büyüklüğünü gözler. Giriş dalgabiçimi olması gerektiği gibi ise ve çıkış dalgabiçimi değilse o zaman söz konusu devrenin bir yerinde bir hata bulunmaktadır. Giriş dalgabiçimi olması gerektiği biçimde bulunmasa bile test mühendisi giriş hattını, bir sinyal üreteci yardımıyla doğru biçim ve büyüklükte bir test dalgabiçimini kullanarak kırabilir ve devreyi uyarabilir. Bu olanaklarla saha servis mühendisinin hataların yerlerini eleman düzeyine kadar belirlemesi, seyrek bir durum değildir.

Gördüğümüz gibi mikroişlemci-tabanlı sistemlerdeki sorun, devre düğümlerindeki dalgabiçimlerinin, yalnızca mantık analizörü gibi özel bir cihaz kullanılarak ayrıntılı biçimde incelenebilen karmaşık darbe dizileri oluşturmaya eğilimli bulunmalarıdır. Bu darbe dizileri, analog devre servis bilgilerinde olduğu gibi, basit genlik/biçim diyagramlarıyla genellikle tanımlanamazlar. Böylece eleman düze-



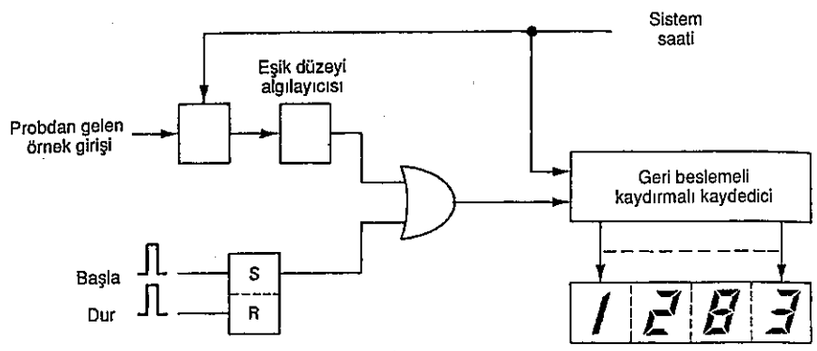
Şekil 13.12 Analog devrelerde hata-bulmaya yardımcı dalgabiçimleri

yinde hata bulma konusu servis mühendisinden beklenenleri arttırmakta ve devre çalışmasının analog sistemlerde gerekenden çok daha geniş biçimde anlaşılmasını gerektirmektedir. Bunun bir sonucu, mikroişlemci donanımı servis hizmetlerinde, kart değişiminin yaygın biçimde kullanılmasıdır.

İşaret analizörleri, mikroişlemci sistemlerin saha servis hizmetlerinde görülen güçlüklerin giderilmesi için özel olarak tasarlanmış cihazlardır. İşaret analiz tekniği, karmaşık bir darbe dizisini basit bir parametre ya da "işaret" ile karakterize etmemize olanak verir, bu işlem birçok yönden bir analog dalgabiçiminin biçim ve büyüklüğü ile karakterize edilişine benzer.

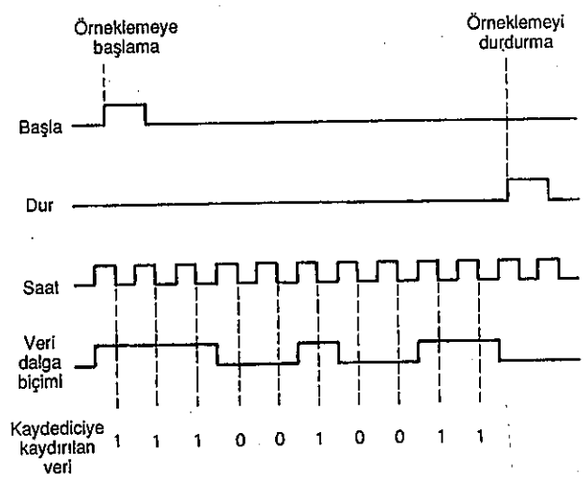
İşaret analizörleri, test edilen sistemin saati ile eşzamanlı olarak çalışırlar. İşaret analizörünün tek veri probu, denetlenen devrenin test yapılacak düğümü ya da noktasına yerleştirilir. Probdaki gerilim, her sistem saat çevrimi sırasında bir kez örneklenir ve bu, mantık analizöründe kullanılan ve Şekil 13.4'te gösterilen tekniğe benzer bir eşik algılama tekniği kullanılarak, 1 ya da 0 mantıksal düzeyleri şeklinde sınıflandırılır. İşaret analizörünün sistem saat dalgabiçiminden oluşan ikinci bir girişe gereksinimi olduğu açıktır.

Dolayısıyla prob çıkışında, iki mantıksal düzey arasında değişen bir dalgabiçimi elde ederiz. Değişiklikler eşzamanlıdır, saat çevriminin aynı noktasında her zaman bir değişiklik gerçekleşir. Fakat bunlar tekrarlamalı değildir. İşaret analizörü tarafından yapılan ölçüm, test edilen sistem tarafından üretilmesi gereken başlatma ve durdurma darbelerinin belirlediği zaman aralığı içinde gerçekleşir. Şekil 13.13'te bir işaret analizörünün basitleştirilmiş şematik diyagramı gösterilmektedir. RS

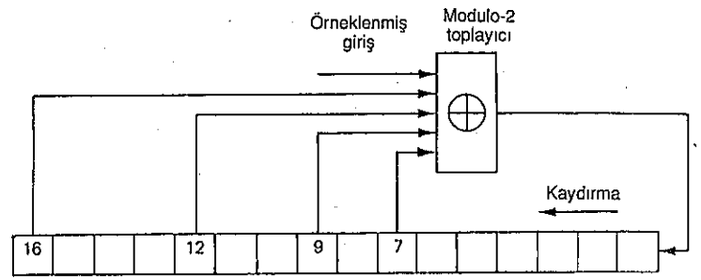


Şekil 13.13 İşaret analizörünün basitleştirilmiş şematik bağlantı düzeni

çiftkararlı devresi, başlatma darbesi tarafından ayarlandığında, veri örnekleri geribeslemeli kaydırmalı kaydedici olarak bilinen özel bir devreye saatlenirler. Ardışık örnekler bu şekilde, RS çiftkararlı devresi yeniden durdurma darbesine ayarlanıncaya kadar, geribeslemeli kaydırmalı kaydediciye saatlenir. Şekil 13.14, bir işaret analizöründeki tipik bir veri dalgabiçimi için zamanlama ilişkilerini göstermektedir, burada RS çift-kararlı devresi, başlatma/durdurma darbelerinin yüksekten-alçak duruma geçişleri tarafından tetiklenmektedir. Veri, sistem saatinin yüksekten-alçak duruma geçişlerinde, geribeslemeli kaydırmalı kaydediciye saatlenmektedir.



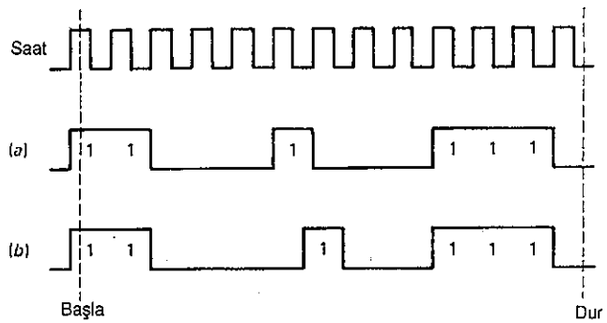
Şekil 13.14 İşaret analizörü zamanlama diyagramı



Şekil 13.15 Onaltı katlı geribeslemeli kaydırmalı kaydedici

Şekil 13.15'te 16-bitlik bir geribeslemeli kaydedici örneği yer almaktadır. Burada 16, 12, 9 ve 7 nolu katların çıkışlarının bir modulo-2 toplayıcısında, kaydedicinin 1 nolu katı için bir sonraki girişi sağlamak amacıyla, giriş bit akışıyla birleştirildiği görülmektedir. Modulo-2 toplayıcısı mantıksal bir elemandır, 1 mantık düzeyindeki çıkışların sayısı bir çift sayı ise bu toplayıcının çıkışı mantıksal 1 değeridir. Bu tanımlamadaki amaca uygun olması için sıfır, bir çift sayı olarak kabul edilir. Örneğin Şekil 13.6'da gösterilen devrede, Q16, Q12, Q9, Q7 = 1101 ve giriş biti 1 ise, modulo-2 toplayıcısının çıkışı, çıkışların dördü 1 mantıksal düzeyinde olduğundan, sıfırdır. Benzer şekilde Q16, Q12, Q9, Q7 = 0010 ve giriş biti 0 ise, modulo-2 toplayıcısının çıkışı, çıkışların yalnızca biri (bir tek sayı) 1 mantıksal düzeyinde olduğundan, 1'dir.

Giriş biti akışı, geribeslemeli kaydırmalı kaydediciye başlama darbesinden sonra verilir. Kaydırma, durdurma darbesi tarafından sonlandırılır ve daha sonra kaydırma kaydedicininin içeriği başlama ve durma arasındaki darbe dizisinin işaretinden

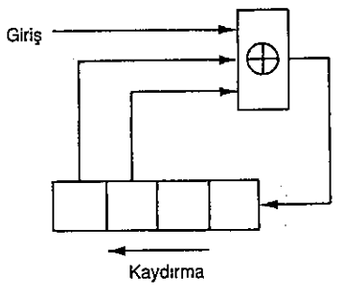


Şekil 13.16 İşaret analizörü test dalgabiçimi

oluşur. İşaret analizörlerinin çoğunluğunda kaydırmalı kaydedicinin içeriğine kod çözümlemesi uygulanır ve onaltılı ya da benzer formatta görüntülenir. Bu noktada okuyucu haklı olarak, giriş dalgabiçimi işaretini üretmek için neden bir geribeslemeli kaydırmalı kaydedici seçildiğini sorabilir. Başka yöntemler de olanaklıdır. Kullanılmış olan yöntemlerden birisi, başlama ve durdurma darbeleri arasında dalgabiçimlerindeki yüksek-alçak ve alçak-yüksek geçişlerin sayılmasıdır. Bu sayı daha sonra darbe dizisi için bir işaret olarak kullanılabilir. O zaman geribeslemeli kaydırmalı kaydedici tarafından üretilen işaretin özelliği nedir?

Bu soruyu yanıtlamak için önce, bir işaret analizörünün yararlılığının, doğru ve yanlış darbe dizileri arasındaki farkları algılama yeteneği tarafından belirlendiğini iyice anlamamız gerekir. Eğer bir yanlış darbe dizisi, doğru sinyalin işaretiyle aynı işareti üretebilirse o zaman cihaz öngörülen amacı başaramamış demektir. Burada anlatılan geribeslemeli kaydırmalı kaydedici mantık analizörlerinin bir özelliği, birbirinden çok az farklı olan darbe dizileri için farklı işaretler üretebilme yetenekleridir.

Şekil 13.16'da iki darbe dizisi gösterilmektedir, bunlardan *(a)* hatasız, *(b)* ise hatalı bir sistemden elde edilmiştir. *(a)* ve *(b)* arasındaki tek farkın, dizinin ortasındaki dar darbenin yerinin değişmesinde olduğu görülecektir. Düzey geçişlerini bir işaret olarak kullanmak, hem *(a)* hem de *(b)*'de bunun gibi beş komut bulunduğundan, sonuçta her hangi bir hata ortaya çıkarmaz. Şimdi de Şekil 13.17'de gösterilen ve çok basit dört katlı geribeslemeli kaydırmalı kaydedici tabanındaki bir işaret analizörünü ele alalım. Örneklenmiş girişin ardışık basamakları içeriye saatlendikçe kaydedici içeriğinin nasıl değiştiği, Şekil 13.18 ve 13.19'da gösterilmektedir. Doğru dalgabiçimi 0010, yanlış darbe dizisi ise 1001 işaretini bırakır.



Şekil 13.17 Dört katlı geribeslemeli kaydırmalı kaydedici

Yukarıdaki örnek, geribeslemeli kaydırmalı kaydedicinin, belirli bir durumdaki hata algılama aracı olarak, geçiş sayımına olan üstünlüğünü kolayca göstermektedir. Geribeslemeli kaydırmalı kaydedici yöntemlerinin genel üstünlüklerini matematiksel olarak kanıtlamak mümkündür.

S13.6

Durum	Bir sonraki giriş	Kaydırmalı kaydediciye gelen bir sonraki girişi
0 0 0 0	1	1
0 0 0 1	1	1
0 0 1 1	0	0
0 1 1 0	0	1
1 1 0 1	0	0
1 0 1 0	1	0
0 1 0 0	0	1
1 0 0 1	0	1
0 0 1 1	0	0
0 1 1 0	1	0
1 1 0 0	1	1
1 0 0 1	1	0
0 0 1 0	0	

Şekil 13.18 Doğru dalgabiçimi (a) için geribeslemeli kaydırmalı kaydedici sıra durumu

Durum	Bir sonraki giriş	Kaydırmalı kaydediciye gelen bir sonraki girişi
0 0 0 0	1	1
0 0 0 1	1	1
0 0 1 1	0	0
0 1 1 0	0	1
1 1 0 1	0	0
1 0 1 0	0	1
0 1 0 1	1	0
1 0 1 0	0	1
0 1 0 1	0	1
1 0 1 1	1	0
0 1 1 0	1	0
1 1 0 0	1	1
1 0 0 1	0	

Şekil 13.19 Hatalı dalgabiçimi (b) için geribeslemeli kaydırmalı kaydedici sıra durumu

Kaydırmalı kaydedicideki geribesleme bağlantılarının seçimi bu uygulamada önemlidir. Uygun geribesleme bağlantılarıyla ilgili temel kural ve bilgiler 13.1 ve 13.3 nolu kaynaklarda bulunmaktadır.

Hatanın bulunduğu yerin belirlemesinde işaret analizörü kullanırken servis mühendisi, hatalı devredeki çeşitli test noktalarında ya da "düğümlerdeki" işaretleri kontrol edecektir. Bu işaretler doğru biçimde çalışan bir devredeki benzer düğümlerde bulunan işaretlerle karşılaştırılır. Bu uygulamanın başarılı olması için, çeşitli düğümlerde uygun darbe dizilerini üretecek bir yol bulunmalıdır. Burada gerekli olan şey, bir analog devrenin düğümlerini uyaran sinyal üretecinin eşdeğerinde bir devredir.

Mikroişlemci sistemlerin çoğunluğunda düğümlerin birçoğunu uyarabilmen elverişli bir yolu, sistemi serbest-çalışma durumuna getirmektir. Bu işlem (B 12.10'da anlatılmıştır) adres yolundaki adresin mümkün bütün değerler üzer sürekli olarak bir döngüye girmesine neden olur. Bu adresler 16-hatlık bir yolu bulunan sistemdeki 0000-FFFF boyunca döngü yaparken, en büyük değe hat olan A15, 0'dan 1'e bir geçiş yapar. Bu geçiş 7FFF-8000 adres geçişi sıra ortaya çıkar. Ticari işaret analizörlerinin çoğunluğunda A15 hattındaki sinyali başlatma hem de durdurma denetim hatlarına verilebilir. Bunun arkasından ilk geçişi işaret çözümlemesini başlatır, işaret çözümleme işlemi daha sonra ik geçiş tarafından sona erdirilir. 13.12 nolu kaynak tekniği daha ayrıntılı biçim anlatmaktadır.

A15'i, işaret analiz "penceresini" açıp kapayan bir araç gibi kullanmak, adres olabilecek tüm değerler üzerinde döngüdeyken A0-A14 adres hatlarındaki işare kontrol edilmesine olanak verir. 8080-tabanlı bir sistem için, 65.536 adreslik bi döngü, her NOP komutu yürütüm için dört saat saykılına gerek duyduğunda 262,144 saat darbesi gerektirecektir. İşaretler adres yolunun değişik noktalarında kontrol edilmelidir.

Mikroişlemci serbest çalışma modunda ve adresler olabilecek tüm değerler üzerin de döngüdeyken, adres kod çözücülerinin çıkışları uygun adres göründüğünde etkin duruma geçecektir, daha sonra da girişler için aynı şey gerçekleşecektir. Dolayısıyla adres kod çözücülerinin çıkışlarındaki işaretler devrenin doğru biçimde işlev görüp görmediğini gösterebilirler.

Eğer adres kod çözücüleri, 8080 $\overline{\text{MEMRD}}$ ya da $\overline{\text{IOWR}}$ gibi sinyaller tarafından yetkilendirilirlerse, bu hatları, işaretin üretilebilmesi için 0 mantıksal düzeyinde tutmak gerekebilir. Bütün durumlarda işaretler, doğru şekilde işlev gören bir
S13.5 sistemde benzer test durumları sırasında bulunan işaretlerle karşılaştırılır.

Kaynaklar ve bibliyografya

13.1 C.H. House, "Engineering in the time domain calls for a new kind of digital instrument", *Electronics*, pp: 75-81 (1 May 1975).

13.2 W.A. Farnbach, "Systematic turn-on of microprocessor systems using logic state analysers" *Electronic Design*, pp: 15 (19 July 1976)

13.3 W.A. Farnbach, "Logic state analysers-a new instrument for analysing sequential digital processes", IEEE *Transactions on instrumentation and mesurement*, Vol. IM 24(4), pp 353-6 (December 1975).

13.4 C.T. Small and J.S. Morrill, Jr, "The logic state analyser, a viewing port for the data domain", *Hewlett-Packard Journal*, pp 210 (August 1975).

13.5 W.A. Farnbach, "Troubleshooting in the data domain is simplified by logic state analysers", *Electronics*, pp 103-105 (15 May 1975)

13.6 R.L. Down, "Understanding logic analysers", *Computer design*, Vol. 16(6), pp 188-91 (June 1977)

13.7 T. Clark, "Troubleshooting microprocessor-based systems" *Digital Design*, pp 56-66 (February 1978)

13.8 G.Gordon and H.Nadig, "Hexadecimal signitures identify troubleshoots in microprocessor systems", *Electronics* (3 March 1977)

13.9 R.A.Frohwerk, "Signiture analysis: a new digital field service method" *Hewlett-Packard Journal* (May 1977)

13.10 D.A.Sharrit "Designing serviceability into the Model 8568 A spectrum analyser", *ibid.*(June 1978)

13.11 D.A.Sharrit "Team up a microP with signiture analysis and ease troubleshooting in the field", *Electronic Design*, (4 January 1979)

13.12 A.Stefanski "Free-running signiture analysis simplifies troubleshooting", *Electronic Design News* (5 February 1979)

Sorular

13.1 Bir mantık analizörünün eşzamanlı ve asenkron örnekleme yapması arasındaki farkı açıklayınız.

13.2 Basit bir blok diyagram yardımıyla bir mantık analizörünün iç işlemlerini anlatınız.

13.3 Mantık analizörü çapak saptama amacıyla kullanıldığında eşzamanlı örneklemenin asenkron örneklemeden niçin daha iyi olduğunu açıklayınız.

13.4 Çizimler yardımıyla, mantık analizöründe zaman ekseninde ve veri ekseninde görüntülemeyi anlatınız.

13.5 İşaret analizörünün temel ilkesini açıklayınız.

13.6 Şekil 13.17'de gösterilen geribeslemeli kaydırmalı kaydedicinin içeriğini 11001111000 sırası olarak eşzamanlı biçimde kaydırıldığında belirleyiniz. Kaydedicinin başlangıç içeriğinin 0000 olduğunu varsayabilirsiniz.

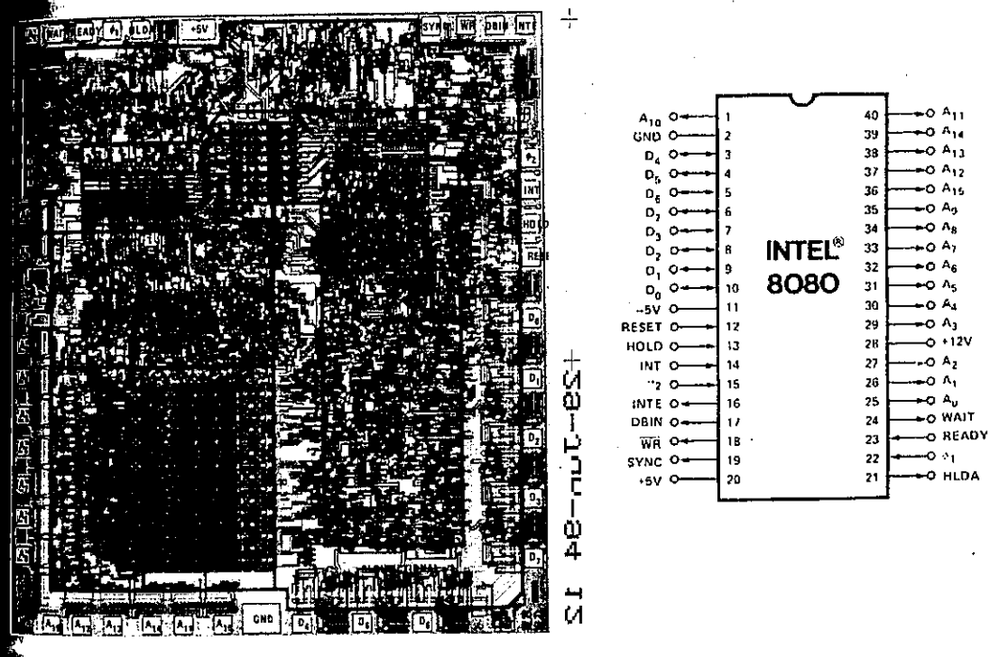
Not:

Aşağıda verilen Ekler, çeşitli firmaların yüksek müsadeleri ile kopyalanmış olan üretici firma ürün bilgilerindeki örneklerden oluşmaktadır. Bu bilgiler sadece, mevcut kaynaklara ve ürün karşılaştırmalarına örnek olarak verilmiştir. Bunların, tasarım bilgisi için ilk kaynak gibi kullanılmamaları gerekir. Mikroelektroniğin hızlı gelişimi nedeniyle, siz bu kitabı okurken büyük olasılıkla daha güncel bilgiler mevcut olacaktır.

Intel 8080 merkezi işlem birimi

8080 MERKEZİ İŞLEM BİRİMİ

8080, genel amaçlı sayısal bilgisayar sistemlerinde kullanmak üzere tasarlanmış olan, tam 8-bitlik paralel bir merkezi işlem birimidir. Intel n-kanal silisyum MOS işlemi kullanılarak, tek bir LSI(Large Scale Integration) yongası üzerinde üretilmiştir (Şekil 3-1'e bakınız). 8080, verileri ve iç durum bilgilerini, 8-bitlik, iki-yönlü, 3-durumlu bir Veri Yolu (D_0-D_7) aracılığıyla aktarır. Bellek ve çevre birimlerinin adresleri, 16-bitlik, 3-durumlu ayrı bir Adres Yolu (A_0-A_{15}) üzerinden gönderilir. Dört denetim girişi (READY, HOLD, INT ve RESET), dört güç girişi (+12 V, +5 V, -5 V ve GND) ve iki saat girişi (ø1 ve ø2) 8080'e girer, ve altı zamanlama ve denetim çıkışı (SYNC, DBIN, WAIT, $\overline{WR}$, HLDA ve INTE) 8080'den çıkar.



Şekil 2-1. 8080'nin, Bacak Tanımları İle Birlikte Fotomikrografı

8080 MİB'NİN MİMARİSİ

8080 MİB aşağıdaki işlevsel birimlerden oluşur:

- Kaydedici dizisi ve adres mantık devresi
- Aritmetik ve mantık birimi (ALU)
- Komut kaydedicisi ve denetim bölümü
- İki-yönlü, 3-durumlu veri yolu tamponu

Şekil 2-2'de, 8080 MİB'ndeki işlevsel bloklar çizimle gösterilmiştir.

Kaydediciler:

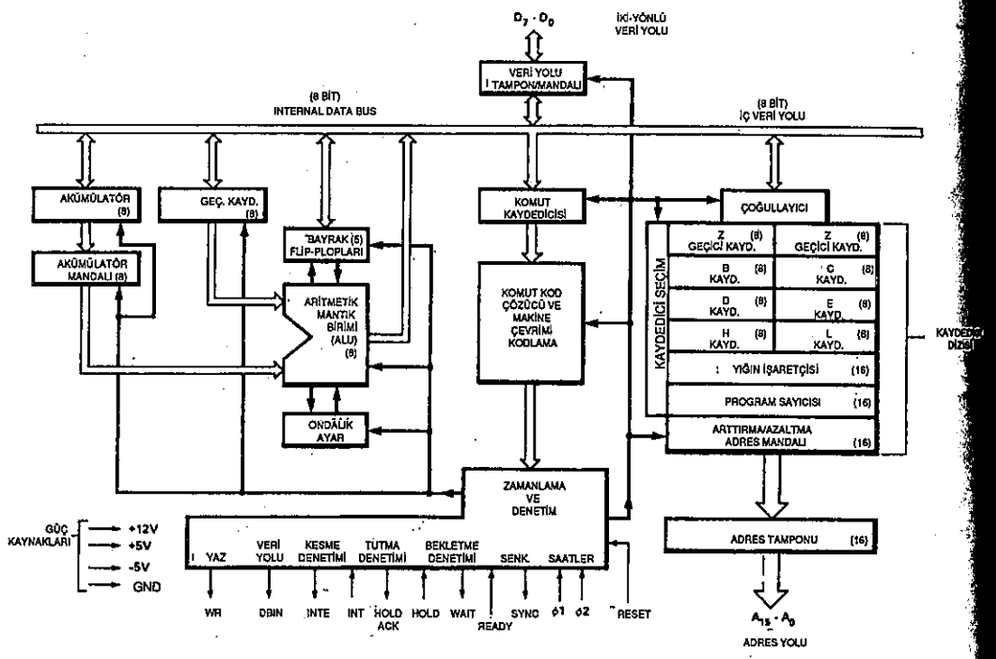
Kaydedici bölümü, altı adet 16-bitlik kaydedici halinde düzenlenmiş olan statik bir RAM dizisinden oluşur:

- Program sayıcısı (PC)
- Yığın işaretçisi (SP)
- B,C;D,E; ve H,L harfleri ile gösterilen, çiftler halinde düzenlenmiş 6 adet 8-bitlik genel amaçlı kaydedici
- W,Z olarak adlandırılan geçici bir kaydedici çifti

Program sayıcısı, yürürlükteki program komutunun bellek adresini içerir ve her komut getirmede otomatik olarak arttırılır. Yığın işaretçisi, bellekte, kullanılabilecek bir sonraki yığın konumu adresini içerir. Yığın işaretçisine, oku-yazılabilen herhangi bir parçası yığın olarak kullanılacak şekilde bir başlangıç değeri yüklenebilir. Yığın işaretçisi, yığına veri "itildiğinde" azaltılır, veri yığından geriye "çekildiğinde" arttırılır (yani yığın "aşağıya doğru" büyür).

6 adet genel amaçlı kaydedici, (8-bitlik) tek kaydedici olarak ya da (16-bitlik) kaydedici çifti olarak kullanılabilir. W,Z geçici kaydedici çifti, programla adreslenemez ve sadece komutların iç yürütümü için kullanılır.

Sekiz-bitlik veri baytları, kaydedici-seçme çoğullayıcısı üzerinden, iç yol ile kaydedici dizisi arasında aktarılabilir. Onaltı-bitlik aktarmalar, kaydedici dizisi ile adres mandalı ya da arttırma/azaltma devresi arasında oluşabilir. Adres mandalı, üç kaydedici çiftinin herhangi birinden veri alır ve arttırma/azaltma devresi ile birlikte 16 adres çıkış tamponunu (A_0-A_{15}) sürer. Arttırma/azaltma devresi, adres mandalından veri alarak kaydedici dizisine gönderir. 16-bitlik veri, arttırılabilir, azaltılabilir ya da sadece kaydediciler arasında aktarılabilir.



Şekil 2-2. 8080 MİB'nin İşlevsel Blok Diyagramı

Aritmetik ve Mantık Birimi (ALU):

ALU aşağıdaki kaydedicileri içerir:

- 8-bitlik akümülatör
- 8-bitlik geçici akümülatör (ACT)
- 5-bitlik bayrak kaydedicisi: sıfır, elde, işaret, eşlik ve yardımcı elde
- 8-bitlik geçici kaydedici (TMP)

Aritmetik, mantıksal ve döndürme işlemleri, ALU içerisinde yerine getirilir. ALU, geçici kaydedici (TMP), geçici akümülatör (ACT) ve elde flip-flopu ile beslenir. İşlemin sonucu, iç yola ya da akümülatöre aktarılabilir; ALU, aynı zamanda bayrak kaydedicisini de besler.

Geçici kaydedici (TMP), iç yoldan bilgi alır ve aldığı bilginin tamamını ya da parçalarını, ALU'ya, bayrak kaydedicisine ve iç yola gönderebilir.

Akümülatör (ACC), ALU ve iç yoldan yüklenebilir ve geçici akümülatör (ACT) ve iç yola veri aktarabilir. Akümülatörün (ACC) içeriği ve yardımcı elde flip-flopu, DAA komutunun yürütülmesi sırasında (Bölüm 4'e bakınız) ondalık düzeltme için test edilebilir.

Komut Kaydedicisi ve Denetim:

Herhangi bir komut getirmede, bir komutun (OP kodunu içeren) ilk baytı, iç yoldan 8-bitlik komut kaydedicisine aktarılır.

Komut kaydedicisinin içeriği, ardışık olarak komut kod çözücüsüne uygulanır. Çeşitli zamanlama sinyalleri ile birleştirilen kod çözücü çıkışı, kaydedici dizisi, ALU ve veri tampon blokları için gerekli denetim sinyallerini sağlar. Ayrıca, komut kod çözücü çıkışları ile dış denetim sinyalleri, durum ve çevrim zamanlama sinyallerini üreten zamanlama ve durum denetim bölümüne de beslenirler.

Veri Yolu Tamponu:

Bu 8-bitlik, iki yönlü, 3-durumlu tampon, MİB'nin iç yolunu dış veri yolundan (D_0-D_7) yalıtmak için kullanılır. Çıkış modunda, iç yol içeriği, ardışık olarak veri yolu çıkış tamponlarını süren 8-bitlik bir mandala yüklenir. Çıkış tamponları, giriş ya da aktarma ile ilgili olmayan işlemler sırasında devre dışına anahtarlanır.

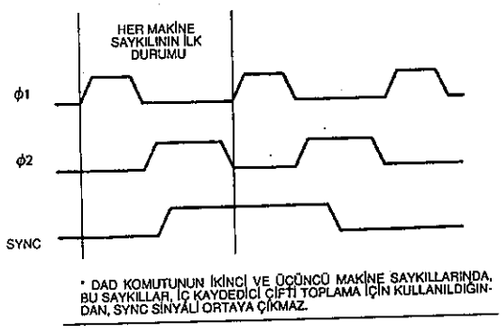
Giriş modu sırasında veri, dış veri yolundan iç yola aktarılır. İç yol, aktarma durumu (T_3-bu bölümde daha sonra açıklanacaktır) dışında, her iç durumun başlangıcında önceden yüklenir.

İŞLEMCİ SAYKILI

Komut saykılı, bir komutu getirmek ve yürütmek için gerekli süre olarak tanımlanır. Getirme sırasında, seçilen bir komut (bir, iki ya da üç baytlık) bellekten çekilir ve MİB'nin komut kaydedicisine yerleştirilir. Yürütme fazı sırasında, komutun kodu çözülerek belirli işleme etkinliklerine dönüştürülür.

Her komut saykılı, bir, iki, üç, dört ya da beş makine saykılından oluşur. MİB'nin belleğe ya da bir giriş çıkış portuna her erişiminde bir makine saykılı gerekir. Bir komut saykılının getirme parçası, getirilecek olan her bayt için bir makine saykılı gerektirir. Komut saykılının yürütme parçasının süresi, getirilen komutun türüne bağlıdır. Bazı komutlar, komutu getirmek için gerekli olanlardan başka hiç bir makine saykılını gerektirmez; ancak diğer komutlar, belleğe ya da giriş çıkış birimlerine veri yazmak veya bunlardan veri okumak için ilave makine saykıllarına gerek duyarlar. DAD komutu, bir iç kaydedici çifti toplama işlemini tamamlamak için iki ilave makine saykılına gerek gösterdiğinden dolayı, bir istisnadır (Bölüm 4'e bakınız).

Her makine saykılı, üç, dört ya da beş durumdan oluşur. Durum, işleme etkinliğinin en küçük birimidir ve 1'den sürülen saat darbelerinin pozitife giden ardışık iki geçişi arasındaki aralık olarak tanımlanır. 8080, iki-fazlı bir saat osilatörü ile sürülür. Bütün işleme etkinlikleri için bu saatin peryodu referans alınır. 1 ve 2 ile etiketlenmiş olan, birbiri üzerine binmeyen iki saat darbesi, dış devre tarafından sağlanır. Her makine saykılını durumlara bölen, 1 saat darbesidir. 8080'deki zamanlama mantık devresi, saat girişlerini, her makine saykılının başlangıcını belirten bir SYNC (senkronizasyon) darbesi üretmek için kullanır. SYNC darbesi, Şekil 2-3'de gösterildiği gibi, 2'nin alçaktan-yükseğe geçişi ile tetiklenir.



Şekil 2-3 1, 2 Ve SYNC Zamanlaması

Durum için tanımlanan sürenin üç istisnası vardır. Bunlar, bekle (WAIT) durumu, tutma (HLDA) durumu ve durdurma (HLTA) durumudur. Bu durumlar, bu bölümde daha sonra tanımlanacaktır. WAIT, HLDA ve HLTA durumları dış olaylara bağlı olduğundan, bu durumların süreleri dış olayların belirsiz uzunluklu doğaları tarafından belirlenir. Ancak bu istisnai durumların bile, sürülen saatin darbeleri ile senkronize edilmeleri gerekir. Bundan dolayı, bütün durumların süreleri saat peryodunun üst üste toplamı halindedir.

Buna göre özetlersek, her saat peryodu bir durumu işaret eder; üçden beşe kadar olabilen sayıda durum bir makine saykılını oluşturur; ve birden beşe kadar olabilen sayıda makine saykılı de bir komut saykılını oluşturur. Bütün bir komut saykılının tamamlanması için, herhangi bir yerde, kapsanan komutun türüne bağlı olarak dörtten onsekize kadar durum gerekir.

Makine Saykıllarının Tanımlanması:

DAD komutunun istisnai durumu ile birlikte, verilen herhangi bir komut saykılında kaç makine saykılının gerekli olduğunun belirlenmesinde göz önüne alınması gereken, sadece tek bir nokta vardır: işlemcinin, komutu getirmek ve yürütmek için bir bellek adresini ya da adreslenebilen bir çevre birimini işaret ettiği zamanların sayısı. Çoğu işlemciler gibi 8080 işlemcisi de, makine saykılı başına sadece bir adres gönderecek şekilde yapılmıştır. Bu yüzden, eğer bir komutun getirme ve yürütülmesi iki bellek işaretlemeye gerek gösterirse, o zaman o komuta ilişkin komut saykılı iki makine saykılından oluşur. Eğer böyle beş işaretleme gerekirse, o zaman komut saykılı beş makine saykılını içerir.

Her komut saykılında, en azından komutun getirilmesindeki bellek işaretlemesi mevcuttur. Bir komut saykılında, komutun yürütülmesi için başkaca bellek işaretlemeye gerek olmasa bile, her zaman getirme makine saykılını bulunmak zorundadır. Bu yüzden, her komut saykılındaki ilk makine saykılı, bir GETİRME saykılıdır. Bunun ötesinde çok katı kurallar yoktur. Sonraki saykıllar getirilen komutun türüne bağlıdır.

Şimdi bazı örnekleri inceleyelim. Kaydediciyle-topla (ADD r) komutu, tamamı için sadece tek bir makine saykılı (GETİRME) gerektiren bir komuttur. Bu tek-baytlık komutta, MİB'nin altı genel amaçlı kaydedicisinden birinin içeriği, akümülatörün mevcut içeriğine eklenir. Komutu yürütmek için gerekli olan tüm bilgi komut kodundaki sekiz bitin içinde kapsanıldığından dolayı, sadece bir tek bellek işaretleme gerekir. Komutu bellekten çekmek için üç durum ve istenen toplamayı gerçekleştirmek için de bir ilave durum kullanılır. Böylece bütün komut saykılı, dört durumdan ya da dış saatin dört peryodundan oluşan, sadece tek bir makine saykılını gerektirir.

Fakat şimdi belirli bir bellek konumunun içeriğini, akümülatörün o anki içeriğine eklemek istediğimizi (ADD M) varsayalım. Bu örnek temel olarak az önce sözü [illegible] örneğe çok benzemekle birlikte, bu durumda başka bi[illegible] adım gerekir. Ayrıca, istenen bellek gözünü adreslem[illegible] de fazladan bir makine saykılı kullanılır.

Uygulamadaki sıra aşağıdaki gibidir. Önce, işlemci[illegible] lekten program sayıcısı tarafından adreslenen tek-b[illegible] komut sözcüğünü çeker. Bu işlem üç durum alır. GETİR[illegible] makine saykılında elde edilen sekiz-bitlik komut sözcü[illegible] MİB'nin komut kaydedicisine koyulur ve komut saykılının[illegible] kalanındaki etkinlikleri yönlendirmek için kullanılır. Son[illegible] işlemci H ve L kaydedicilerinin içeriklerini adres olarak d[illegible] gönderir. Bu BELLEK OKUMA makine saykılında yan[illegible] rak gelen sekiz-bitlik veri sözcüğü, 8080 MİB'nin içeris[illegible] geçici kaydediciye yerleştirilir. Buraya kadar üç saat pe[illegible] daha geçmiş oldu. Yedinci ve son durumda, geçici kay[illegible] cinin içeriği akümülatörünkine eklenir. Toplam yedi dur[illegible] dan oluşan iki makine saykılı, "ADD M" komut say[illegible] tamamlar.

Başka bir örnekte, beş makine saykılını gerektiren [illegible] L kaydedicilerini sakla (SHLD) komutu yer alır. "SHLD" [illegible] mut saykılı sırasında, işlemcinin H ve L kaydedicile[illegible] içeriği, ardışık olarak bitişik iki bellek gözüne konulur; var[illegible] adresi, işlem kodu baytından hemen sonra gelen iki bel[illegible] konumunda saklanan iki adres baytı ile belirtilir. Bu komut saykılı sırasında, aşağıdaki olaylar sırası gerçekleşir;

(1) Dört durumdan oluşan bir GETİRME makine say[illegible] kılı. Bu makine saykılının ilk üç durumunda, işlemci program sayıcısı tarafından işaret edilen komutu getirir. Sonra program sayıcısı arttırılır. Dördüncü durum iç komut kod çözümü için kullanılır.

(2) Üç durumdan oluşan bir BELLEK OKUMA say[illegible] Bu makine saykılı sırasında, program sayıcısı tara[illegible] fından işaret edilen bayt bellekten okunur ve işlem[illegible] cinin Z kaydedicisine yerleştirilir. Program say[illegible] yine arttırılır.

(3) Üç durumdan oluşan diğer bir BELLEK OKUM[illegible] saykılı. Bu saykılda, işlemcinin program sayac[illegible] tarafından işaret edilen bayt bellekten okunarak W[illegible] kaydedicisine yerleştirilir. Program sayıcısı, bir sonraki komutun getirilmesi için şimdiden arttırılır.

(4) Üç durumluk bir BELLEK YAZMA makine say[illegible] Bu makine saykılında, L kaydedicisinin içeriği W[illegible] Z kaydedicilerinin şimdiki içerikleri tarafından işa[illegible] edilen bellek konumuna aktarılır. Aktarmanın ardı[illegible] dan, W, Z kaydedici çiftini arttırmak için kullanılı[illegible] böylece bu kaydedici çifti veri alınacak olan bi[illegible] sonraki bellek konumunu işaret eder.

(5) Üç durumluk bir BELLEK YAZMA makine say[illegible] Bu makine saykılında H kaydedicisinin içeriği, W,Z kaydedici çifti tarafından işaret edilen yeni bellek konumuna aktarılır.

Özet olarak, "SHLD" komut saykılı beş makine saykılını [illegible]ir ve yürütülmesi için 16 durum gerekir.

Çoğu komutlar, "ADD r" ve "SHLD" komutları ile simge[illegible] [illegible]nan sınırlar arasında bir yerlere düşer. Örneğin, giriş (INP) [illegible] çıkış (OUT) komutları, üçer makine saykılı gerektirirler: [illegible]komutu elde etmek için GETİRME makine saykılı; hedef [illegible]birimin adresini elde etmek için BELLEK OKUMA makine [illegible]saykılı; ve aktarmayı tamamlamak için de GİRİŞ ya da ÇIKIŞ [illegible]makine saykılı.

Beş makine saykılından daha fazla sayıda makine say[illegible]kılından oluşan bir komut saykılı yoktur, ancak bir komut say[illegible]kılında aşağıdaki on farklı tür makine saykılı ortaya çıkabilir:

(1) GETİRME (M1)

(2) BELLEKTEN OKUMA

(3) BELLEĞE YAZMA

(4) YIĞINDAN OKUMA

(5) YIĞINA YAZMA

(6) GİRİŞ

(7) ÇIKIŞ

(8) KESME

(9) DURDURMA

(10) DURDURMA • KESME

Herhangi bir komut saykılındaki ilk makine saykılının her zaman bir GETİRME olduğu koşulu ile birlikte, belirli bir komut saykılında fiilen ortaya çıkacak olan makine saykılları, komutun tipine bağlıdır.

İşlemci, sürmekte olan makine saykılını, her makine saykılının ilk durumunda sekiz-bitlik bir durum sözcüğünü göndermek suretiyle tanımlar. Güncellenmiş durum bilgisi, 8080'nin veri hatlarına (D_0-D_7) SYNC aralığında çıkılır. Bu veri, mandallarda saklanarak dış devre için gereken denetim sinyallerini geliştirmek için kullanılmalıdır. Tablo 2-1, pozitif-doğru durum bilgisinin, işlemcinin veri yolu üzerinde nasıl dağıldığını göstermektedir.

Durum sinyalleri, temel olarak dış devrenin denetimi için üretilirler. Makine saykılının tanımlanmasından çok, arabağlantı basitliği, ayrık durum bitlerinin mantıksal tanımını belirler. Bu yüzden, belirli işlemci makine saykıllarının tek bir durum biti ile tek olarak tanımlandıklarını, fakat diğerleri için bunun geçerli olmadığını göreceksiniz. Örneğin M1 durum biti (D6), GETİRME makine saykılını hiç bir karışıklık olmadan tanımlar. Diğer taraftan, YIĞIN OKUMA makine saykılı, STACK (YIĞIN) ve MEMR sinyallerinin çakışımıyla belirtilir. Makine saykılı tanımlama verisi, ayrıca sistem geliştirmenin test ve hata ayıklama aşamalarında da değerlidir. Tablo 2-1'de, her makine saykılı türü için durum biti çıkışları listelenmiştir.

Durum Geçiş Sırası:

Bir komut saykılının içerisindeki her makine saykılı, üçten beşe kadar sayıda olabilen (T_1,T_2,T_3,T_4,T_5 ya da TW ile gösterilen) etkin durumlardan oluşur. Fiili durum sayısı yürütülmekte olan komuta ve daha büyük komut saykılları içerisindeki özel makine saykıllarına bağlıdır. Şekil 2-4'deki durum geçiş diyagramı, 8080'nin bir makine saykılı boyunca durumdan duruma nasıl ilerlediğini göstermektedir. Bu diyagram, aynı zamanda READY (HAZIR), HOLD (TUTMA) ve INTERRUPT (KESME) hatlarının makine saykılı sırasında nasıl örneklendiğini ve bu hatlar üzerindeki durumların temel geçiş sırasını nasıl değiştirebileceğini de göstermektedir. Şimdi, sadece temel sırayı ve READY işlevini inceleyeceğiz. HOLD ve INTERRUPT işlevlerini daha sonra tartışacağız.

8080 MİB, her durum sırasında bir "durum denetim" çıkışı göndererek doğrudan iç durumunu belirtmez; bunun yerine, dış devre tarafından kullanılması için doğrudan denetim çıkışları (INTE, HLDA, DBIN, $\overline{WR}$ ve WAIT) verir.

Her durumun, $\phi 1$ saatinin ardışık alçaktan-yükseğe geçişleri ile tanımlandığını ve 8080'nin, her makine saykılında en azından üç durumdan geçtiğini hatırlayınız. Şekil 2-5, tipik bir GETİRME makine saykılındaki zamanlama ilişkilerini göstermektedir. Her durumda gerçekleşen olaylar, $\phi 1$ ve $\phi 2$ saat darbelerinin geçişlerinden referans alırlar.

SYNC sinyali, her makine saykılındaki ilk durumu (T_1) tanımlar. Şekil 2-5'de gösterildiği gibi, SYNC sinyali $\phi 2$ saatinin ön kenarı ile ilişkilidir. $\phi 2$'nin alçaktan-yükseğe geçişi ile SYNC darbesinin pozitife-giden kenarı arasında bir gecikme (t_{DC}) vardır. Ayrıca, sonraki $\phi 2$ darbesi ile SYNC sinyalinin düşen kenarı arasında da, ilk gecikmeye karşılık gelen bir gecikme (yine t_{DC}) vardır. Durum bilgisi, D_0-D_7 üzerinde aynı 2-2 aralığında gösterilir. Durum sinyallerinin anahtarlanması da, aynı şekilde $\phi 2$ tarafından denetlenir.

T_1 sırasındaki $\phi 2$'nin çıkan kenarı, işlemcinin adres hatlarını da (A_0-A_{15}) yükler. Bu hatlar, $\phi 2$ saatleme darbesinin kısa bir gecikme süresi (t_{DA}) içerisinde kararlı hale gelirler ve T_3 durumundan sonraki ilk $\phi 2$ darbesine kadar kararlı kalırlar. Bu durum, işlemcinin bellekten yanıt olarak gelen veriyi okuması için geniş bir zaman aralığı sağlar.

İşlemci, belleğe bir adres gönderdiğinde, belleğin bir WAIT (BEKLE) durumu istemesi için yeterli bir zaman vardır. Bellek bu isteği, T_2ya da TW durumu içerisindeki $\phi 2$ darbesi sırasında gerçekleşen "READY sinyalini kurma" aralığından (t_{RS}) önce, işlemcinin READY (HAZIR) giriş hattını alçaktan durumuna çekmek suretiyle yapar. READY hattı alçak durumunda kaldığı sürece, işlemci boşta çalışarak, belleğe, adreslenmiş veri isteğine yanıt vermesi için zaman tanır. Şekil 2-5'i inceleyiniz.

İşlemci, bekleme isteğini, T_2'nin sonunda doğrudan doğruya T_3 durumuna geçmek yerine alternatif bir duruma (T_W) girmek suretiyle yanıtlar. T_W durumuna giriş, işlemciden

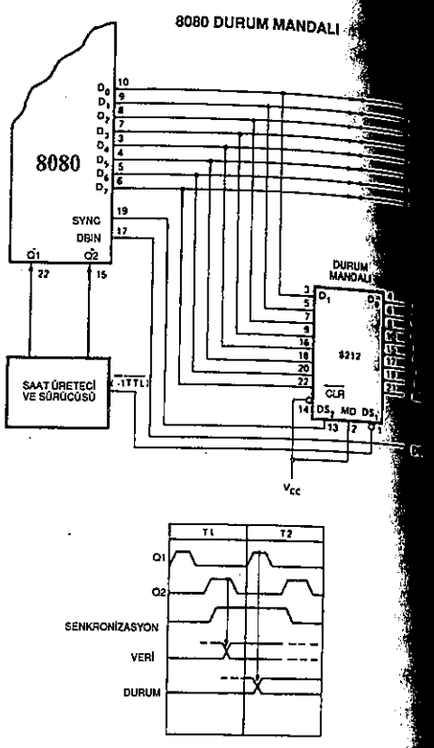
çıkan bir WAIT sinyali ile belirtilir, WAIT hattı üzerindeki, belleğin isteğinin kabul edildiğini gösteren bir alçaktan-yükseğe geçişi, ø1 saatinin çıkan kenarı ile tetiklenir ve T_W durumuna fiili girişden itibaren kısa bir gecikme süresi (t_{DC}) içerisinde gerçekleşir.

Bekleme peryodu sonsuz süreli olabilir. İşlemci, READY hattı tekrar yüksek durumuna gidinceye kadar bekleme durumunda kalır. T_W durumundan çıkışı garanti etmek için, READY belirtimi ø2 saatinin düşen kenarından belirli bir aralık (t_{RS}) kadar önce gelmelidir. O taktirde saykıl, ø1 saatinin sonraki çıkan kenarıyla başlayarak devam edebilir. Bu yüzden, herhangi bir WAIT aralığı, T_W durumlarının üst üste toplanan bir sayısından oluşur ve her zaman saat peryodunun katlarına eşit olur.

8080'nin komutları, tam yürütme için birden beşe kadar olabilen sayıda makine saykıllarını gerektirirler. 8080, her makine saykılının başlangıcında (SYNC anında) 8 bitlik bir durum bilgisini dışarıya veri yoluna gönderir. Aşağıdaki tabloda durum bilgisi tanımlanmıştır.

DURUM BİLGİSİ TANIMI

Semboller	Veri Yolu Biti	Tanım
INTA*	D_0	KESME isteği için alındı sinyali. Bu sinyal, DBIN etkin olduğu zaman veri yoluna bir yeniden başlatma komutunu kapılamak için kullanılmalıdır.



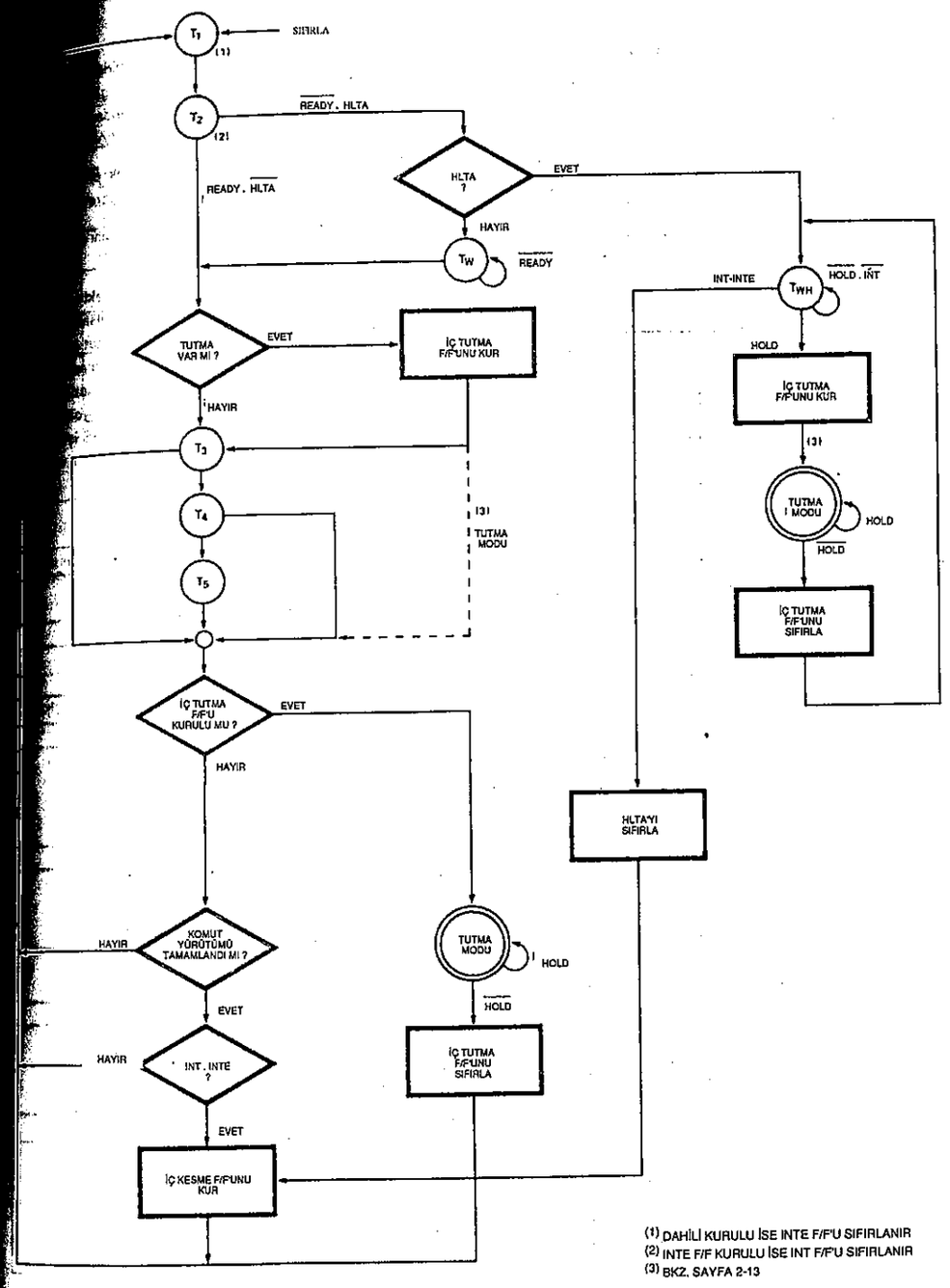
DURUM SÖZCÜK TABLOSU

MAKİNE SAYKILI TİPİ

	VERİ YOLU BİTİ	DURUM BİLGİSİ	KOMUT GETİRME	BELLEKTEN OKUMA	BELLEĞE YAZMA	YIĞINDAN OKUMA	YIĞINA YAZMA	GİRİŞ OKUMA	ÇIKIŞ YAZMA	KESME ALINDISI	DURDURMA ALINDISI	DURDURMA DURUMUNDA KESME ALINDISI
			①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
D_0	INTA		0	0	0	0	0	0	0	1	0	1
D_1	$\overline{WO}$		1	1	0	1	0	1	0	1	1	1
D_2	STACK.		0	0	0	1	1	0	0	0	0	0
D_3	HLTA		0	0	0	0	0	0	0	0	1	1
D_4	OUT		0	0	0	0	0	0	1	0	0	0
D_5	M_1		1	0	0	0	0	0	0	1	0	1
D_6	INP		0	0	0	0	0	1	0	0	0	0
D_7	MEMR		1	1	0	1	0	0	0	0	1	0

Ⓝ DURUM SÖZCÜĞÜ NUMARASI

Tablo 2-1. 8080 Durum Biti Tanımları

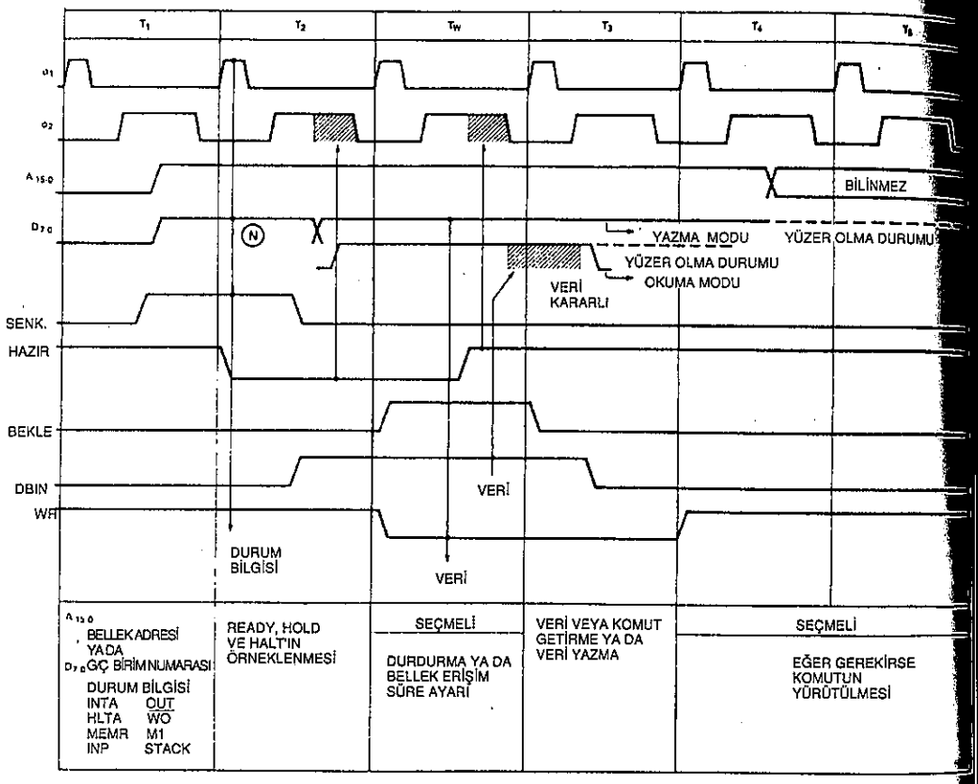


Şekil 2-4. MİB Durum Geçiş Diyagramı

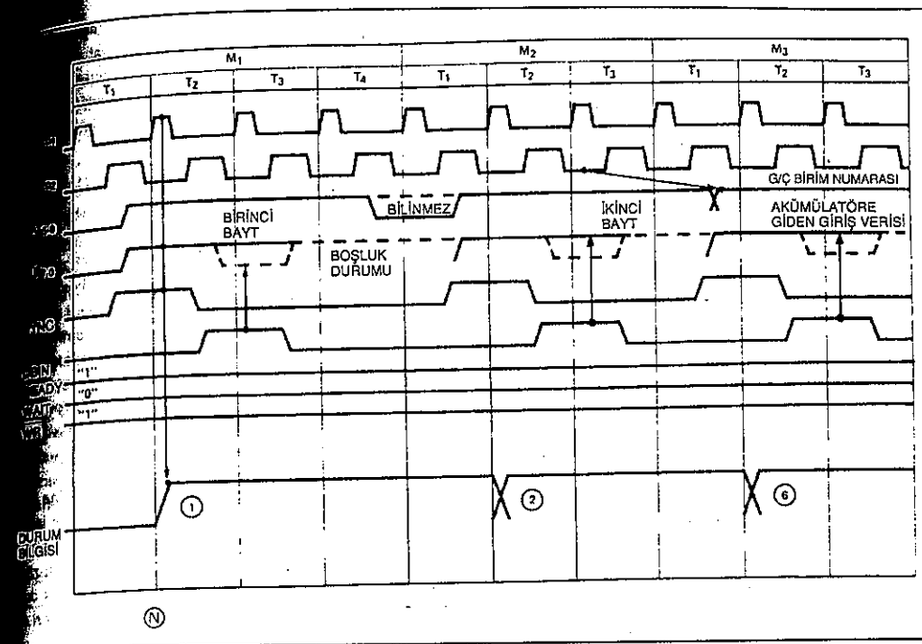
$\overline{WO}$	D_1	Yürürlükteki makine saykılındaki işlemin, bellek YAZMA ya da ÇIKIŞ işlevi ($\overline{WO}$ = 0) olacağını belirtir. Aksi halde, bellek OKUMA ya da GİRİŞ işlemi yürütülür.
STACK	D_2	Adres yolunun, Yığın İşaretçisinden gelen, aşağı itilen yığın adresini tutmakta olduğunu belirtir.
HLTA	D_3	HALT komutu için alındı sinyalidir.
OUT	D_4	Adres yolunun, bir çıkış biriminin adresini içerdiğini ve veri yolunun, $\overline{WR}$ etkin olduğunda çıkış verisini içereceğini belirtir.
M_1	D_5	MİB'nin, bir komutun ilk baytı için getirme saykılında olduğunu belirten bir sinyal sağlar.
INP*	D_6	Adres yolunun, bir giriş birimi adresini içerdiğini ve giriş verisi DBIN etkin olduğunda verinin yerleştirilmesi gerektiğini belirtir.
MEMR*	D_7	Veri yolunun, bellek okuma işlemi için kullanılacağını işaret eder.

* Bu üç durum biti, 8080 veri yolu üzerindeki veri akışını denetlemek için kullanılabilir.

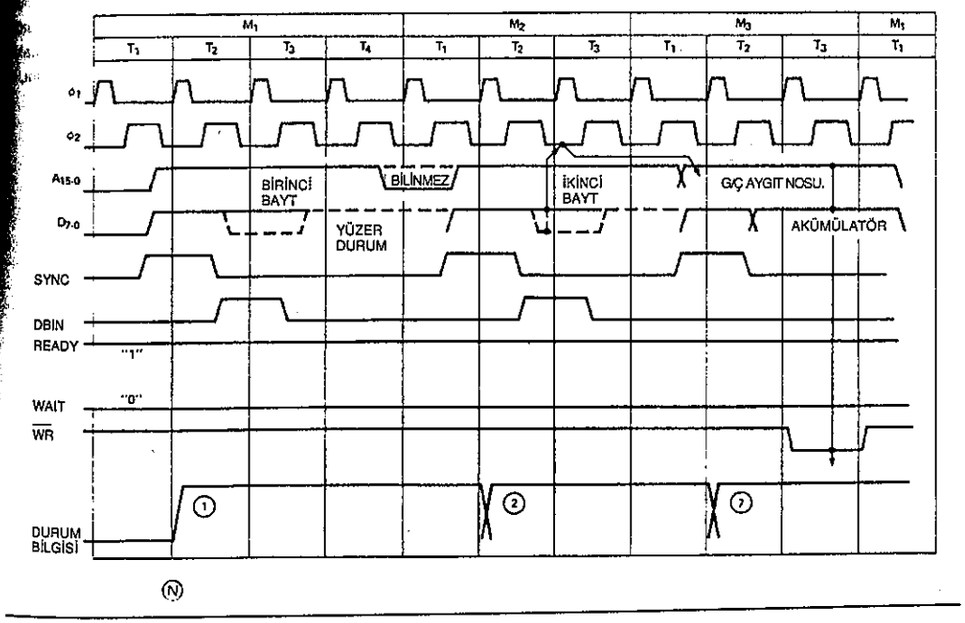
T3 durumunda ortaya çıkan olaylar, sürmekte olan makine saykılının türü tarafından belirlenir. GETİRME makine saykılında, işlemci veri yolundaki veriyi bir komut olarak yorumlar. BELLEK OKUMA ya da YIĞIN OKUMA sırasında, veri yolu üzerindeki bir veri, veri sözcüğü olarak yorumlanır. Bellek, BELLEK YAZMA makine saykılı sırasında bu yolu sürer. G/Ç işlemleri sırasında, ÇIKIŞ ya da GİRİŞ işlemi yürütülmesine bağlı olarak, işlemci ya veri gönderir ya da veri alır.



Şekil 2-5. 8080 Temel Komut Saykılı



Şekil 2-6. Giriş Komut Saykılı



Şekil 2-7. Çıkış Komut Saykılı

Şekil 2.6'da, veri giriş işlemi için karakteristik zamanlama diyagramı gösterilmiştir. Görüldüğü gibi, ø2'nin T_2 sırasındaki alçaktan-yükseğe geçişi, durum bilgisini işlemcinin veri hatlarından siler, böylece bu hatları, gelen verinin alınmasına hazırlar. İşlemciye uygulanan verinin, hem ø1 darbesinin T_3 durumunu tanımlayan düşen kenarından önceki "ø1veri kurma" aralığından (t_{DS1}) önce, hem de ø2'nin T_3 durumundaki çıkan kenarından önceki "ø2-veri kurma" aralığından (tDS2) önce kararlı hale gelmiş olması gerekir. Aynı verinin, ø2 darbesinin çıkan kenarını izleyen "veri tutma" aralığı (t_{DH}) boyunca da kararlı kalması gerekir. Bu hatlara, bellek ya da diğer dış aygıtlar tarafından yerleştirilen veri, T_3 sırasında örneklenir.

İşlemciye veri girişi sırasında, 8080, aktarmayı sağlayabilmek için dışarıdan kullanılması gereken bir DBIN sinyali üretir. DBIN'nin mevcut olduğu makine saykıllarını: GETİRME, BELLEK OKUMA, YIĞIN OKUMA ve KESME makine saykıllarını içerir. DBIN, ø2'nin T_2 durumu sırasındaki çıkan kenarı ile başlatılır ve ø2'nin T_3 sırasındaki karşılık gelen kenarı ile sonlandırılır. Bu yüzden, T_2 ile T_3 arasına giren herhangi bir sayıdaki T_W fazları, DBIN'i bir ya da daha fazla sayıda saat peryodu kadar genişletir.

Şekil 2-7, işlemcinin veri çıkardığı bir makine saykılının zamanlamasını göstermektedir. Çıkış verisi, ya bellek ya da çevre birimlerine yöneltilebilir. ø2'nin T_2 durumu içindeki çıkan kenarı, durum bilgisini MİB'nin veri hatlarından siler ve dış birimlere çıkılacak olan veriyi yükler. Bu yerleşme işlemi, ø2 saatinin ön kenarını izleyen "veri çıkış gecikmesi" aralığı (t_{DD}) içerisinde gerçekleşir. Yol üzerindeki veri, bir sonra gelen T_1 durumundaki güncellenmiş durum bilgisi ile yerdeğiştirilinceye kadar, makine saykılının geri kalanı boyunca kararlı kalır. Bir ÇIKIŞ makine saykılının tam[illegible] bir READY sinyalinin gerekli olduğuna d[illegible] bir belirtme yapılmadığı sürece, işlemci T_2 durum[illegible] dan T_W durumuna girer. Çıkış hatları üzerind[illegible] zamanda kararlı kalır ve işleme saykılı READY [illegible] yüksek durumuna gidinceye kadar ilerlemez.

8080 MİB, işlemcinin veri çıktığı makine [illegible] sında, dışarı aktarmaların senkronizasyonu için [illegible] üretir. Bu makine saykılları, BELLEĞE YAZ[illegible] YAZMA ve ÇIKIŞ makine saykılları ni içerir. W[illegible] giden ön kenarı, T_2'yi izleyen ilk ø1 saat darb[illegible] kenarına referanslanır ve kısa bir gecikme sürec[illegible] sinde gerçekleşir. $\overline{WR}$, T_3'ü izleyen durum sıras[illegible] ön kenarı tarafından tekrar tetikleninceye kada[illegible] munda kalır. Çıkış makine saykılının T_2 ile T_3'ü [illegible] herhangi bir sayıdaki T_W durumlarının, veri giri[illegible] sırasında DBIN'nin etkilendiği tarza çok benzer bi[illegible] zorunlu olarak $\overline{WR}$'ı genişleteceğine dikkat ediniz.

Az önce de tanımlandığı gibi, her işlemci makin[illegible] en azından üç durumdan oluşur: T_1,T_2 ve T_3. Eğer [illegible] haberleşmekte olduğu çevre birimi ya da bellek[illegible] için beklemesi gerekiyorsa, o zaman makine saykılı [illegible] bir ya da çok T_W durumunu da içerebilir. Üç temel [illegible] sırasında, ya işlemciye ya da işlemciden dışarıya [illegible] rılır.

Ancak, T_3 durumundan sonrası için genelleştirm[illegible] mak zordur. Eğer özel bir komutun yürütülmesi gerek[illegible] T_4 ve T_5 durumları da kullanılabilir. Fakat bu durumlar[illegible] makine saykılları kullanmaz. Bunların kullanılması, yürü[illegible] komutun türüne ve komut saykılı içerisindeki özel[illegible] saykılına bağlıdır. İşlemci, herhangi bir makine saykılı[illegible]

DURUM	DURUMA İLİŞKİN ETKİNLİKLER
T_1	Adres Yoluna (A_{15-0}) bir bellek adresi ya da G/Ç numarası yerleştirilir; Veri Yoluna (D_{7-0}) durum bilgisi yerleştirilir.
T_2	MİB, READY ve HOLD girişlerini örnekleyerek durdurma komutu için kontrol eder.
TW (seçmeli)	Eğer READY alçak durumunda ise ya da HALT komutu yürütülürse, işlemci bekleme durumuna girer.
T_3	Bir komut baytı (GETİRME makine saykılı), veri baytı (BELLEK OKUMA, YIĞIN OKUMA) ya da kesme komutu (KESME makine saykılı) Veri Yolundan MİB'ne girilir; ya da bir veri baytı veri yoluna çıkılır (BELLEĞE YAZMA, YIĞINA YAZMA ya da ÇIKIŞ makine saykılı).
T_4 T_5 (seçmeli)	Özel bir komutun yürütülmesi için gerekiyorsa T_4 ve T_5 durumları da kullanılabilir; eğer bu durumların kullanılması gerekmiyorsa o zaman MİB bunlardan birini ya da ikisini birden atlayabilir. T_4 ve T_5 sadece işlemci içindeki işlemler için kullanılır.

Tablo 2-2. Durum Tanımları

[illegible] T_5 durumları boyunca devam etmek yerine, [illegible] tikleri tamamlanır tamamlanmaz sonlandırır. Bu[illegible] 8080, bir makine saykılından T_3, T_4 ya da T_5 durumu [illegible] ğılarak, doğrudan doğruya bir sonraki makine say[illegible] durumuna geçebilir.

[illegible]ME İŞLEMLERİNİN SIRALANMASI

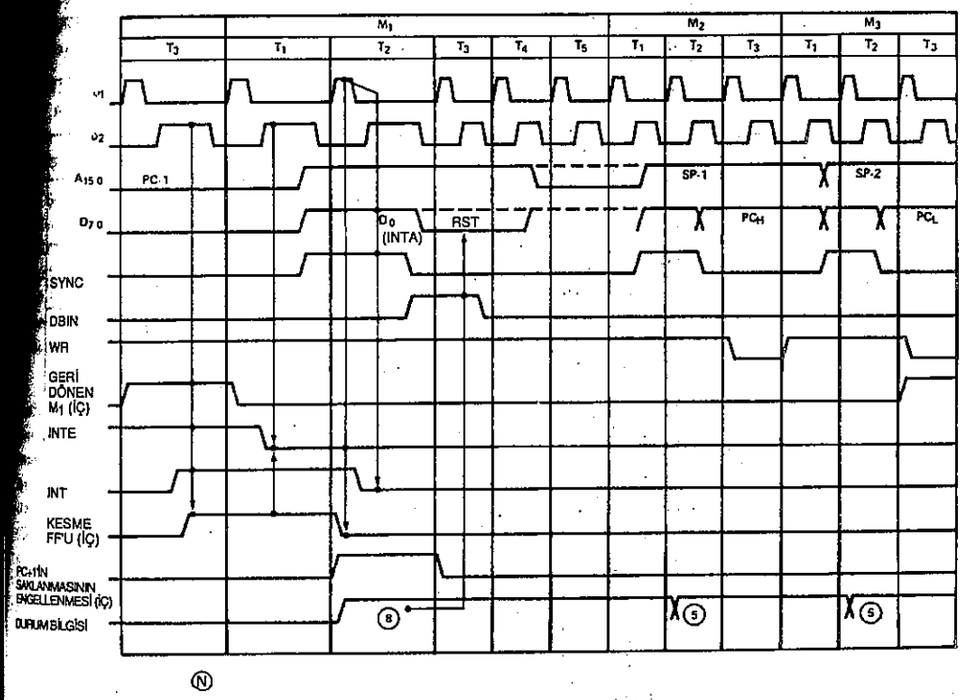
[illegible] 8080, dış kesme isteklerini işletmek için gerekli yapısal [illegible] sahiptir. Herhangi bir çevre birimi, sadece işlem[illegible] hattını (INT) yüksek durumuna sürmek suretiyle [illegible] başlatabilir.

[illegible] girişi (INT) asenkrondur, bu yüzden bir kesme [illegible] herhangi bir komut saykılı süresinde, herhangi bir [illegible] gelebilir. İç mantık, dış kesme isteğini yeniden-saatler, [illegible] sürücü saat ile uygun bir çakışma sağlanır. Şekil [illegible] gösterildiği gibi, kesme yetkilendirme hattı (INTE) [illegible] ken gelen bir kesme isteği (INT), iç kesme mandalını [illegible] için ø2 saatiyle uyumlu olarak iş görür. Bu olay, [illegible] isteği geldiği anda yürütülmekte olan komut saykılının [illegible] durumu sırasında gerçekleşir, böylece sürmekte olan [illegible] hangi bir komutun, kesmenin işlenebilir hale gelmesinden [illegible] tamamlanması sağlanır.

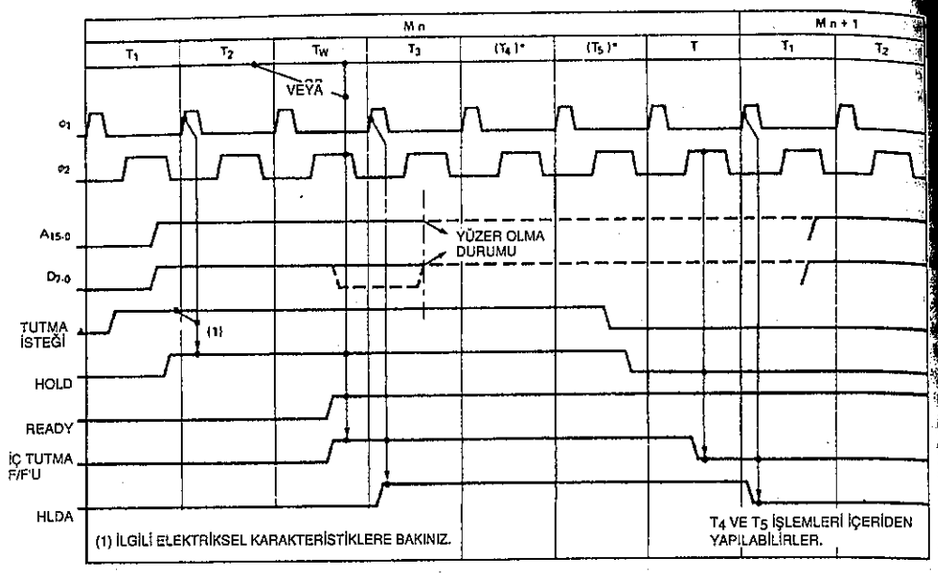
Yetkilendirilmiş bir kesme isteğinin gelişini izleyen KESME makine saykılı, bir çok bakımdan herhangi bir GETİRME makine saykılına benzer. M_1 durum biti, her zaman olduğu gibi, SYNC aralığında gönderilir. Ancak, bu durum biti, dış isteğin alındığını belirten INTA durum biti (D_0) ile birlikte gönderilir. T_1 sırasında, program sayıcısının içeriği MİB'nin adres hatlarına mandallanır, fakat sayıcısının kendisi, KESME makine saykılı sırasında diğer zamanlarda olduğu gibi arttırılmaz. Bu yolla, program sayıcının kesme-öncesi durumu saklanmış olur, böylece sayaçtaki veri, kesme isteği işlendikten sonra kesilen program tarafından geri alınabilir.

Kesme saykılı, diğer bakımlardan herhangi bir GETİRME makine saykılından ayırt edilemez. İşlemcinin kendisi, başka hiç bir özel işlem yapmaz. T_3 durumu sırasında işlemcinin veri yoluna sekiz-bitlik bir kesme komutunun "sıkıştırıldığını" algılamak, çevresel mantık devresinin sorumluluğudur. Tipik bir sistemde, bunun anlamı bellekten gelen veri-giriş yolunun işlemcinin ana veri yolundan geçici olarak izole edilmesi gerektiğidir, bunun amacı kesme yapan birimin anayola, karışıklık olmaksızın hakim olmasını sağlamaktır.

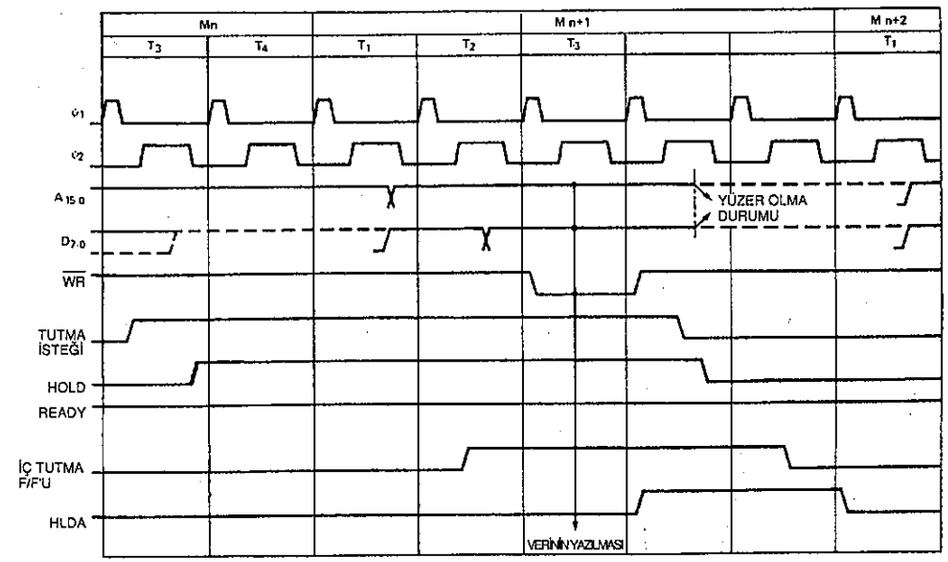
8080'nin komut takımında, kesmelerin işlenmesini kolaylaştıran tek-baytlık özel bir çağrı komutu mevcuttur (nor-



Şekil 2-8. Kesme Zamanlaması



Şekil 2-9. TUTMA işlemi (Okuma Modu)



Şekil 2-10. TUTMA işlemi (Yazma Modu)

program çağrı komutu üç baytlıktır). Bu komut RESET (RST) (YENİDEN BAŞLAT) komutudur. RST'nin sekiz kodlanında ayrılan ve üç bitlik yer işgal eden bir değişken, kesmeye neden olan birimin, sekiz sabit bellek konumundan birisine çağrı yönelteceğini belirler. Bu ayrılmış konumların adresleri: 0, 8, 16, 24, 32, 40, 48, ve 56'dır. Bu adreslerden herhangi biri, kesme yapan bir birimin gereksinmelerine hizmet vermek için tasarlanmış bir yordamın ilk komut (ya da komutlarını) saklamak için kullanılabilir. (RST) bir çağrı komutu olduğundan, komutun tamamlanması aynı zamanda program sayıcısının eski içeriğini de YIĞIN'a saklar.

TUTMA İŞLEMLERİNİN SIRALANMASI

8080A MİB, Doğrudan Bellek Erişimi (Direct Memory Access - DMA) işlemleri için gerekli hazırlıkları içerir. Herhangi bir dış birim, işlemcideki uygun denetim pinine bir HOLD (TUTMA) sinyali uygulamak suretiyle, MİB'nin normal işlemlerini geçici olarak askıya alarak adres ve veri yolları üzerindeki kontrolünü bırakmasına yol açabilir. İşlemci böyle bir isteğe, yollarını bölüşmek için diğer birimler tarafından kullanılmak üzere adreslerini yüzer (3-durumlu konum) durumuna almak suretiyle yanıt verir. İşlemci aynı anda HLDA çıkış pinine yüksek düzey sinyali çıkarmak suretiyle HOLD'un alındığını belirtir. Alınmış bir HOLD sinyali sırasında, adres ve veri yolları, isteği yapan çevre biriminin denetimi altındadır, böylece o çevre biriminin, işlemcinin müdahalesine gerek olmadan bellek aktarımlarını yürütmesi mümkün olur.

HOLD girişi de kesme girişi gibi içeriden senkronize edilir. Bir HOLD sinyalinin, ø2'nin yükselen kenarından önce gelen "Tutma kurma" aralığından (t_{HS}) önce kararlı hale gelmiş olması gerekir.

Şekil 2-9 ve 2-10'da, HOLD işlemlerinde gereken zamanlama grafik olarak açıklanmıştır. Asenkron TUTMA İSTEĞİ ile yenidensaatlenmiş HOLD arasındaki gecikmeye dikkat ediniz. Diyagramda gösterildiği gibi, READY, HOLD ve ø2 saatlerinin çakışması, iç tutma mandalını kurar. Mandalın kurulması, HLDA çıkışını tetiklemesi için ø1 saat darbesinin bir sonraki yükselen kenarını yetkilendirir.

TUTMA İSTEĞİ'nin alındısı, işlemcinin adres ve veri hatlarının fiilen yüzer durumuna alınışlarından kısa bir süre önce gelir. HOLD sinyali, eğer bir okuma ya da bir giriş makine saykılı sürmekte iken gelirse (Şekil 2-9'a bakınız), işlemci tarafından T_3'ün başlangıcında kabul edilir. Aksi halde alındı, T_3'ü izleyen durumun başlangıcına kadar ertelenir (Şekil 2-10'a bakınız). Bununla birlikte, her iki halde de HLDA çıkışı, seçilen ø1 saat darbesinin yükselen kenarının belirli bir gecikme süresi (t_{DC}) içerisinde yüksek durumuna gider. Adres ve veri hatları, bir sonraki 2 saat darbesinin yükselen kenarından sonraki kısa bir gecikme içerisinde yüzer durumuna getirilir. Bu ilişki diyagramda da gösterilmiştir.

Adres ve veri yolları bir kez yüzer duruma alındığında, işlemci dışarıyla ilgili olan tüm işlemlerini askıya alır. Bununla birlikte, içeride bazı işlemler devam edebilir. Eğer bir TUTMA İSTEĞİ T3 durumunda kabul edilirse ve eğer işlemci, tamamlanması için dört ya da daha fazla durum gerektiren bir makine saykılının ortasında ise, MİB serbest duruma geçmeden önce T4 ve T5 durumlarından geçer. Makine saykılının sonuna ulaşılmadıkça işlemler sona ermez. Buna göre, iç işlemlere ile dış DMA aktarımının üst üste getirilmesi mümkündür, böylece tüm sistemin hem etkinliği hem de hızı arttırılmış olur.

İşlemci, tutma durumundan, girerken geçtiği sıraya benzer bir sıra içerisinden geçerek çıkar. TUTMA İSTEĞİ, dış birim veri aktarımını tamamladığında asenkron olarak sonlandırılır. HLDA çıkışı, bir sonraki 1 saat darbesinin ön kenarı ardından alçak düzeyine geri döner. Tutma isteğinden önce yürütülen son saykılı izleyen makine saykılıyla normal işlemlere geri dönülür.

DURDURMA İŞLEMLERİNİN SIRALANMASI

Bir durdurma komutu (HLT) yürütüldüğü zaman, Şekil 2-11'de gösterildiği gibi MİB, bir sonraki makine saykılının T2 durumundan sonra durdurma durumuna (T_{WH}) girer. 8080, durdurma durumundan sadece üç yoldan biriyle çıkabilir:

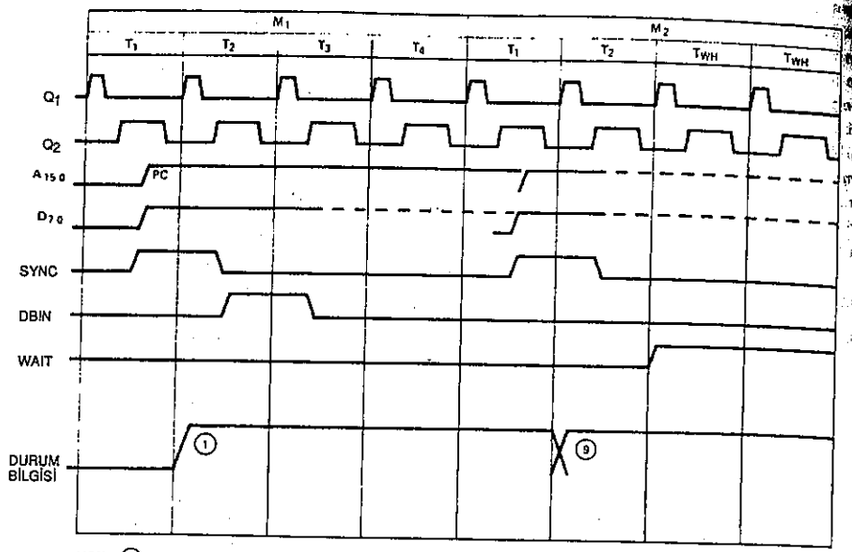
- RESET hattına uygulanan bir yüksek düzeyi, 8080'i daima T1 durumuna sıfırlar; RESET, aynı zamanda program sayıcısına da sıfırlar.
- Daha önce de anlatıldığı gibi, bir HOLD girişi 8080'nin tutma durumuna girmesine neden olur. HOLD hattı alçak düzeye gittiği zaman, 8080 bir sonraki ø1 saati darbesinin yükselen kenarında yeniden durdurma durumuna girer.
- Herhangi bir kesme (yani, INTE yetkilendirilmişken INT girişinin yüksek düzeye gitmesi), 8080'nin Durdurma durumundan çıkmasına ve bir sonraki ø1 saat darbesinin yükselen kenarında T1 durumuna girmesine neden olur. NOT: Durdurma durumuna girildiğinde, kesme yetkilendirme (INTE) bayrağı kurulu olmalıdır; aksi halde 8080 durdurma durumundan sadece bir RESET sinyali ile çıkabilir.

Şekil 2-12'de durdurma sıralaması, akış diyagramı formunda gösterilmiştir.

8080 MİB'NİN BAŞLATILMASI

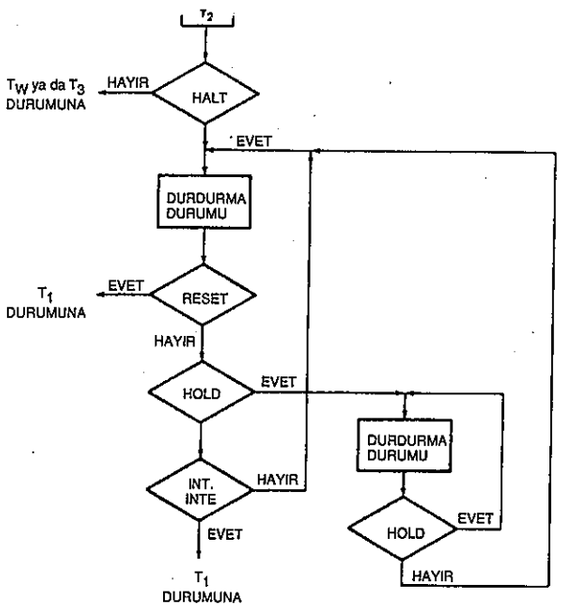
8080'e ilk olarak güç uygulandığında, işlemci hemen çalışmaya başlar. Program sayıcısı, yığın işaretçisi ve diğer çalışma kaydedicilerinin ilk andaki içerikleri için, doğal olarak rastgele etkenler söz konusudur ve belirlenemezler. Bu nedenle, güç uygulama anındaki sırayı RESET'le başlatmak gerekir.

En az üç saat peryodu süreli olan bir dış RESET sinyali, işlemcinin içinde bulunan program sayacını sıfırlar. Bundan dolayı, herhangi bir RESET'in ardından program çalıştırma, sıfırıncı bellek konumundan başlar. İşlemcinin, açık bir baş-



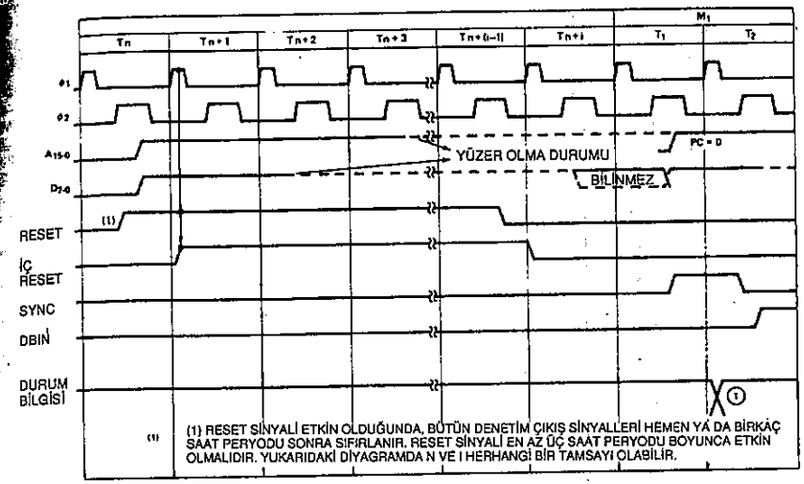
NOT: (N) → (Sayfa 2-6 daki Durum Sözcük Tablosuna Bakınız)

Şekil 2-11. DURDURMA Zamanlaması



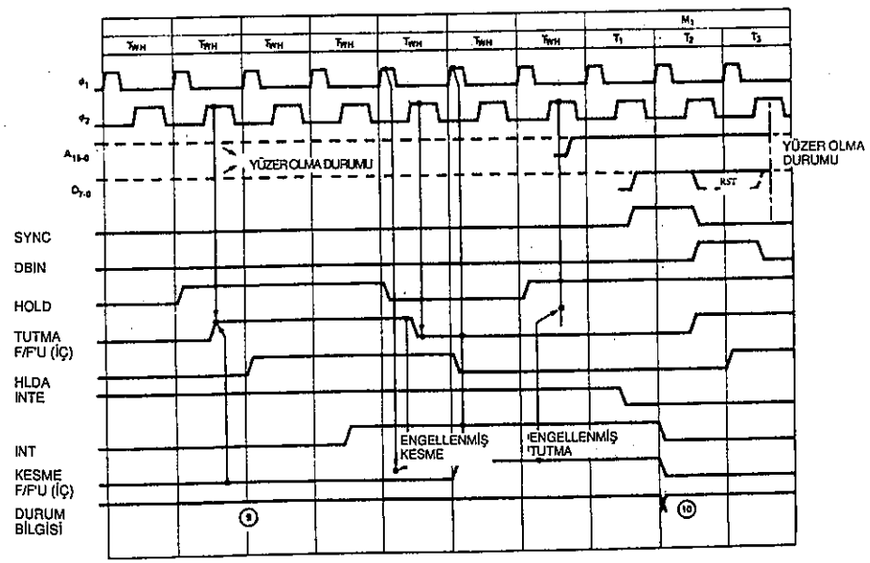
Şekil 2-12. DURDURMA Sırası Akış Diyagramı

sinyalini beklemesini gerektiren sistemlerde, ilk iki umda bir durdurma komutu (EI, HLT) saklanır. Başlatma manuel veya otomatik bir KESME kullanılır. Diğer sis- lerde, işlemci saklı programı hemen yürütmeye başlaya- Bununla birlikte, RESET'in durum bayrakları veya işlemcinin çalışma kaydedicilerinden (akümülatör, kaydedici- ler, veya yığın işaretçisi) hiç biri üzerinde herhangi bir etkisi olmadığına dikkat ediniz. Bu kaydedicilerin içerikleri, ilk de- ğerleri program tarafından açıkça atanmadıkça belirsiz ola- rak kalır.



NOT: (N) → (Şekil 2-8'de verilenle aynı)

Şekil 2-13. Sıfırlama (Reset)



NOT: (N) → (Şekil 2-8'de verilenle aynı)

Şekil 2-14. DURDURMA durumunda HOLD ile INT arasındaki ilişki

KOM KISLT.	İŞ KODU		M1[1]					M2			M3			M4			M5				
	$D_7D_6D_5D_4$	$D_3D_2D_1D_0$	T1	T2[2]	T3	T4	T5	T1	T2[2]		T1	T2[2]	T3	T1	T2[2]	T3	T1	T2[2]	T3	T4	T5
MOV r1, r2	0 1 D D	D S S S	PC ÇIKIŞ DURUMU	PC=PC+1	INST-TMP/R	(SSS) →TMP	(TMP) →DDD														
MOV r, M	0 1 D D	D 1 1 0				x[3]		HL ÇIKIŞ DURUMU[6]	VERİ → DDD												
MOV M, r	0 1 1 1	0 S S S				(SSS) →TMP		HL ÇIKIŞ DURUMU[7]	(TMP) → VERİ												
SPHL	1 1 1 1	1 0 0 1				(HL) → SP															
MVI r, veri	0 0 D D	D 1 1 0				x		PC ÇIKIŞ DURUMU[6]	B2 → DDD												
MVI M, veri	0 0 1 1	0 1 1 0				x			B2 → TMP		HL ÇIKIŞ DURUMU[7]	(TMP) → VERİ YOLU									
LXI rp, veri	0 0 R P	0 0 0 1				x			PC = PC + 1 B2 → r1		PC ÇIKIŞ DURUMU[6]	PC = PC + 1 83 → rh									
LDA adres	0 0 1 1	1 0 1 0				x			PC = PC + 1 B2 → Z			PC = PC + 1 83 → W		WZ ÇIKIŞ DURUMU[6]	VERİ → A						
STA adres	0 0 1 1	0 0 1 0				x			PC = PC + 1 B2 → Z			PC = PC + 1 83 → W		WZ ÇIKIŞ DURUMU[7]	(A) → VERİ YOLU						
LHLD adres	0 0 1 0	1 0 1 0				x			PC = PC + 1 B2 → Z			PC = PC + 1 83 → W		WZ ÇIKIŞ DURUMU[6]	VERİ WZ = WZ+1 → L		WZ ÇIKIŞ DURUMU[6]	VERİ → H			
SHLD adres	0 0 1 0	0 0 1 0				x		PC ÇIKIŞ DURUMU[6]	PC = PC + 1 B2 → Z		PC ÇIKIŞ DURUMU[6]	PC = PC + 1 83 → W		WZ ÇIKIŞ DURUMU[7]	(L) WZ = WZ+1 → VERİ YOLU		WZ ÇIKIŞ DURUMU[7]	(H) → VERİ YOLU			
LDAX rp[4]	0 0 R P	1 0 1 0				x		rp ÇIKIŞ DURUMU[6]	VERİ → A												
STAX rp[4]	0 0 R P	0 0 1 0				x		rp ÇIKIŞ DURUMU[7]	(A) → VERİ												
XCHG	1 1 1 0	1 0 1 1				(HL) ↔ (DE)															
ADD r	1 0 0 0	0 S S S				(SSS) → TMP (A) → ACT		[9]	(ACT)+(TMP) → A												
ADD M	1 0 0 0	0 1 1 0				(A) → ACT		HL ÇIKIŞ DURUMU[6]	VERİ → TMP		[9]	(ACT) + (TMP) → A									
ADI data	1 1 0 0	0 1 1 0				(A) → ACT		PC ÇIKIŞ DURUMU[6]	PC = PC + 1 B2 → TMP		[9]	(ACT) + (TMP) → A									
ADC r	1 0 0 0	1 S S S				(SSS) → TMP (A) → ACT		[9]	(ACT)+(TMP)+CY → A												
ADC M	1 0 0 0	1 1 1 0				(A) → ACT		HL ÇIKIŞ DURUMU[6]	VERİ → TMP		[9]	(ACT) -(TMP) -CY → A									
ACI data	1 1 0 0	1 1 1 0				(A) → ACT		PC ÇIKIŞ DURUMU[6]	PC = PC + 1 B2 → TMP		[9]	(ACT) -(TMP) -CY → A									
SUB r	1 0 0 1	0 S S S				(SSS) → TMP (A) → ACT		[9]	(ACT)+(TMP) → A												
SUB M	1 0 0 1	0 1 1 0				(A) → ACT		HL ÇIKIŞ DURUMU[6]	VERİ → TMP		[9]	(ACT) + (TMP) → A									
SUI data	1 1 0 1	0 1 1 0				(A) → ACT		PC ÇIKIŞ DURUMU[6]	PC = PC + 1 B2 → TMP		[9]	(ACT) + (TMP) → A									
SBB r	1 0 0 1	1 S S S				(SSS) → TMP (A) → ACT		[9]	(ACT)+(TMP)+CY → A												
SBB M	1 0 0 1	1 1 1 0				(A) → ACT		HL ÇIKIŞ DURUMU[6]	VERİ → TMP		[9]	(ACT) -(TMP) -CY → A									
SBI data	1 1 0 1	1 1 1 0				(A) → ACT		PC ÇIKIŞ DURUMU[6]	PC = PC + 1 B2 → TMP		[9]	(ACT) -(TMP) -CY → A									
INR r	0 0 D D	D 1 0 0				(DDD) → TMP (TMP) +1 → ALU	ALU → DDD														
INR M	0 0 1 1	0 1 0 0				x		HL ÇIKIŞ DURUMU[6]	VERİ → TMP (TMP)+1 → ALU		HL ÇIKIŞ DURUMU[7]	ALU → VERİ YOLU									
DCR r	0 0 D D	D 1 0 1				(DDD) → TMP (TMP) +1 → ALU	ALU → DDD														
DCR M	0 0 1 1	0 1 0 1				x		HL ÇIKIŞ DURUMU[6]	VERİ → TMP (TMP) -1 → ALU		HL ÇIKIŞ DURUMU[7]	ALU → VERİ YOLU									
INX rp	0 0 R P	0 0 1 1				(RP)+1 → RP															
DCX rp	0 0 R P	1 0 1 1				(RP)-1 → RP															
DAD rp[8]	0 0 R P	1 0 0 1				x		(ri) → ACT	(L) → TMP, (ACT)+(TMP) → ALU	ALU → L	(rh) → A CT	(HL) → TMP (ACT)+(TMP)+CY→ ALU	ALU → H, CY									
DAA	0 0 1 0	0 1 1 1				DAA → A,BAYRAKLAR[10]															
ANA r	1 0 1 0	0 S S S				(SSS) → TMP (A) → ACT		[9]	(ACT)+(TMP) → A												
ANA M	1 0 1 0	0 1 1 0	PC ÇIKIŞ DURUMU	PC=PC+1	INST-TMP/R	(A) → ACT		HL ÇIKIŞ DURUMU[6]	VERİ → TMP		[9]	ACT) + (TMP) → A									

KOM KISLT.	İŞ KODU		M1[1]					M2			M3		M4			M5				
	$D_7D_6D_5D_4$	$D_3D_2D_1D_0$	T1	T2[2]	T3	T4	T5	T1	T2[2]		T2[2]	T3	T1	T2[2]	T3	T1	T2[2]	T3	T4	T5
ANI veri	1 1 1 0	0 1 1 0	PC ÇIKIŞ DURUMU	PC=PC+1	INST-TMP/IR	(A) → ACT		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → TMP		(ACT)+(TPM) → A									
XRA r	1 0 1 0	1 S S S				(A) → ACT (SSS) → TMP		[9]	(ACT)+TPM) → A											
XRA M	1 0 1 0	1 1 1 0				(A) → ACT		HL ÇIKIŞ DURUMU(6)	DATA → TMP		(ACT)+(TPM) → A									
XRI veri	1 1 1 0	1 1 1 0				(A) → ACT		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → TMP		(ACT)+(TPM) → A									
ORA r	1 0 1 1	0 S S S				(A) → ACT (SSS) → TMP		[9]	(ACT)+TPM) → A											
ORA M	1 0 1 1	0 1 1 0				(A) → ACT		HL ÇIKIŞ DURUMU(6)	DATA → TMP		(ACT)+(TPM) → A									
ORI veri	1 1 1 1	0 1 1 0				(A) → ACT		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → TMP		(ACT)+(TPM) → A									
CMP r	1 0 1 1	1 S S S				(A) → ACT (SSS) → TMP		[9]	(ACT)-TPM), BAYRAKLAR											
CMP M	1 0 1 1	1 1 1 0				(A) → ACT		HL ÇIKIŞ DURUMU(6)	DATA → TMP		(ACT)+(TPM); BAYRAKLAR									
CPI veri	1 1 1 1	1 1 1 0				(A) → ACT		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → TMP		(ACT)+(TPM); BAYRAKLAR									
RLC	0 0 0 0	0 1 1 1				(A) → ALU DÖNDÜRME		[9]	ALU → A, CY											
RRC	0 0 0 0	1 1 1 1				(A) → ALU DÖNDÜRME		[9]	ALU → A, CY											
RAL	0 0 0 1	0 1 1 1				(A), CY → ALU DÖNDÜRME		[9]	ALU → A, CY											
RAR	0 0 0 1	1 1 1 1				(A), CY → ALU DÖNDÜRME		[9]	ALU → A, CY											
CMA	0 0 1 0	1 1 1 1				$\overline{(A)} \rightarrow A$														
CMC	0 0 1 1	1 1 1 1				$\overline{CY} \rightarrow CY$														
STC	0 0 1 1	0 1 1 1				1 → CY														
JMP adres	1 1 0 0	0 0 1 1				x		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → Z		PC = PC + 1 B3 → W								WZ ÇIKIŞ DURUMU(11)	(WZ)+1 → PC
J koşl adr. [17]	1 1 C C	C 0 1 0				KARAR KOŞULU		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → Z		PC = PC + 1 B3 → W								WZ ÇIKIŞ DURUMU(11,12)	(WZ)+1 → PC
CALL adres	1 1 0 0	1 1 0 1				SP = SP - 1		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → Z		PC = PC + 1 B3 → W		SP ÇIKIŞ DURUMU(16)	(PCH) → SP = SP - 1	VERİ YOLU	SP ÇIKIŞ DURUMU(16)	(PCL) →	VERİ YOLU	WZ ÇIKIŞ DURUMU(11)	(WZ)+1 → PC
C koşl adr.[17]	1 1 C C	C 1 0 0				KARAR KOŞULU EĞER DOĞRUYSA, SP = SP - 1		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → Z		PC = PC + 1 B3 → W(13)		SP ÇIKIŞ DURUMU(16)	(PCH) → SP = SP - 1	VERİ YOLU	SP ÇIKIŞ DURUMU(16)	(PCL) →	VERİ YOLU	WZ ÇIKIŞ DURUMU(11,12)	(WZ)+1 → PC
RET	1 1 0 0	1 0 0 1				x		SP ÇIKIŞ DURUMU(15)	SP = SP + 1 VERİ → Z		SP = SP + 1 VERİ → W								WZ ÇIKIŞ DURUMU(11)	(WZ)+1 → PC
R koşl adr.[17]	1 1 C C	C 0 0 0			INST-TMP/IR	KARAR KOŞULU(14)		SP ÇIKIŞ DURUMU(15)	SP = SP + 1 VERİ → Z		SP = SP + 1 VERİ → W								WZ ÇIKIŞ DURUMU(11,12)	(WZ)+1 → PC
RST n	1 1 N N	N 1 1 1			↑→W INST→TMP/IR	SP = SP - 1		SP ÇIKIŞ DURUMU(16)	SP = SP - 1 (PCH) → VERİ YOLU		(TMP = 00NN000) → Z (PCL) → Z								WZ ÇIKIŞ DURUMU(11)	(WZ)+1 → PC
PCHL	1 1 1 0	1 0 0 1			INST-TMP/IR	(HL) → PC														
PUSH rp	1 1 R P	0 1 0 1				SP = SP - 1		SP ÇIKIŞ DURUMU(16)	SP = SP - 1 (rh) → VERİ YOLU		(rl) → VERİ YOLU									
PUSH PSW	1 1 1 1	0 1 0 1				SP = SP - 1		SP ÇIKIŞ DURUMU(16)	SP = SP - 1 (A) → VERİ YOLU		BAYRAKLAR → VERİ YOLU									
POP rp	1 1 R P	0 0 0 1				x		SP ÇIKIŞ DURUMU(15)	SP = SP + 1 VERİ → r1		SP = SP + 1 VERİ → rh									
POP PSW	1 1 1 1	0 0 0 1				x		SP ÇIKIŞ DURUMU(15)	SP = SP + 1 VERİ → BAYRAKLAR		SP = SP + 1 VERİ → A									
XTHL	1 1 1 0	0 0 1 1				x		SP ÇIKIŞ DURUMU(15)	SP = SP + 1 VERİ → Z		VERİ → W		SP ÇIKIŞ DURUMU(16)	(H) →	VERİ YOLU	SP ÇIKIŞ DURUMU(16)	(PCL) →	VERİ YOLU	(WZ) → H-L	
IN port	1 1 0 1	1 0 1 1				x		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → Z, W		VERİ → A									
OUT port	1 1 0 1	0 0 1 1				x		PC ÇIKIŞ DURUMU(6)	PC = PC + 1 B2 → Z, W		A → VERİ YOLU									
EI	1 1 1 1	1 0 1 1				INTE F/F'UNU KUR														
DI	1 1 1 1	0 0 1 1				INTE F/F'UNU SIFIRLA														
HLT	0 1 1 1	0 1 1 0				x		PC ÇIKIŞ DURUMU(6)	DURDURMA MODU(20)											
NOP	0 0 0 0	0 0 0 0	PC ÇIKIŞ DURUMU	PC=PC+1	INST-TMP/IR	x														

NOTLAR:

1. İlk bellek saykılı (M1) her zaman bir komut getirme saykılıdır; opkodunu içeren ilk (ya da tek) bayt bu saykılda getirilir.

2. İşlemci, herhangi bir bellek saykılının T2 durumu sırasında bellekten gelen READY girişi yüksek düzeyde değilse, READY girişi YÜKSEK olarak örneklenineeye kadar, bekleme (TW) durumuna girer.

3. T4 ve T5 durumları, MİB'deki tümüyle içsel olan işlemler için, gerekli olduklarında ortaya çıkarlar. T4 ve T5 sırasında iç yol içeriği veri yolunda da görünür; bunun tek amacı test işlemleri için olanak tanımaktır. "X" işareti, veri yoluna bir durum çıkıldığını ifade eder, ancak sadece komut kod çözümü gibi iç işlemler için kullanılır.

4. Sadece rp = B (B ve C kaydedicileri) veya rp = D (D ve E kaydedicileri) kaydedici çiftleri belirtilebilir.

5. Bu durumlar atlanır.

6. Bellek okuma alt-saykılları ; bir komut ya da veri sözcüğü okunur.

7. Bellek yazma alt-saykılı.

8. İkinci ve üçüncü alt-saykıllar (M2 ve M3) sırasında READY sinyaline gerek duyulmaz. HOLD sinyali M2 ve M3 sırasında kabul edilir. SYNC sinyali M2 ve M3 sırasında üretilmez. DAD komutunun yürütülmesi sırasında, iç kaydedici çifti toplama işlemi için M2 ve M3 gerekli olur; bellek işaretlenmez.

9. Bu aritmetik, mantıksal ya da döndürme komutlarının sonuçları, bir sonraki komut saykılının T2 durumuna varılmadıkça akümülatöre (A) taşınmaz. Yani, A bir sonraki komut getirilirken yüklenir; işlemlerin bu şekilde üst üste bindirilmesi, daha hızlı işleme yapılabilmesini sağlar.

10. Eğer akümülatörün en düşük değerlikli 4-bitinin değeri 9'dan daha büyükse ya da yardımcı elde biti kurulmuşsa, akümülatöre 6 eklenir. Eğer akümülatörün en büyük değerlikli 4-bitinin değeri 9'dan daha büyükse ya da elde biti kurulmuşsa, akümülatörün en büyük değerlikli 4-bitine 6 eklenir.

11. Bu durum, bir sonraki komut saykılının ilk alt-saykılını (komut getirme saykılı) gösterir.

12. Eğer koşul sağlanırsa, adres hatlarına (A0-A15), program sayıcısının (PC) içeriği yerine WZ kaydedici çiftinin içeriği çıkarılır.

13. Eğer koşul sağlanmazsa, M4 ve M5 saykılları atlanır; işlemci, bu saykılları n yerine doğrudan doğruya bir sonraki komut saykılının komut getirme makine saykılına (M1) geçer.

14. Eğer koşul sağlanmazsa, M2 ve M3 alt-saykılları atlanır; işlemci, bu saykılları n yerine doğrudan doğruya bir sonraki komut saykılının komut getirme makine saykılına (M1) geçer.

15. Yığın okuma alt-saykılı.

16. Yığın yazma alt-saykılı.

17. KOŞUL — CCC

NZ	- sıfır değil (Z=0)	000
Z	- sıfır (Z=1)	001
NC	- elde yok (CY=0)	010
C	- elde var (CY=1)	011
PO	- eşlik tek (P=0)	100
PE	- eşlik çift (P=1)	101
P	- artı (S=0)	110
M	- eksi (S=1)	111

18. G/Ç alt-saykılı: G/Ç portunun 8-bitlik seçme kodu A0-A7 ve A8-A15 adres hatlarına çift olarak çıkarılır.

19. Çıkış alt-saykılı.

20. İşlemci, herhangi bir kesme, sıfırlama ya da tutma alınana kadar, durdurma durumunda boşta çalışma durumunda kalır. Bir tutma isteği alındığında, MİB tutma moduna girer; işlemci, tutma modu sonlandırıldıktan sonra durdurma durumuna geri döner. Bir sıfırlama sinyali alındığında, işlemci yürütmeye sıfırıncı bellek konumundan başlar. Bir kesme alındığında da, işlemci, veri yoluna uygulanan komutu (genellikle bir yeniden başlatma komutudur) yürütür.

21. Tablolardaki kısaltmaların anlamları:

A	AKÜMÜLATÖR
ACT	GEÇİCİ AKÜMÜLATÖR
B2	2'NCİ BAYT
B3	3'NCÜ BAYT
C	koşul bitleri
CY	elde biti
D	varış değişkeni bitleri
DAA	ONDALIK AYAR AKÜMÜLATÖRÜ
INS	KOMUT
IR	KOMUT KAYDEDİCİSİ
n	yeniden başlatma komutunun yöneltildiği adres
N	yeniden başlatma komutunun yöneltildiği adresin bitleri
M	HL kaydedici çiftinin gösterdiği bellek gözü
r	kaydedici
rp	kaydedici çifti
RP	kaydedici çifti bitleri
rl	kaydedici çiftinin üst sıralı kaydedicisi
rh	kaydedici çiftinin alt sıralı kaydedicisi
S	kaynak değişkeni bitleri
TMP	GEÇİCİ KAYDEDİCİ

SSS veya DDD	Değer	rp	Değer
A	111	B	00
B	000	D	01
C	001	H	10
D	010	SP	11
E	011		
H	100		
L	101		

8080 KOMUT TAKIMI

İşlemci Komutlarının Özeti

Kom. Kis.	Tanım	Komut Kodu [1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Saat [2] Periyotları
TAŞIMA, YÜKLEME VE SAKLAMA										
MOV r1,r2	Kaydediciden kaydediciye taşı	0	1	D	D	D	S	S	S	5
MOV M,r	Kaydediciden belleğe taşı	0	1	1	1	0	S	S	S	7
MOV r,M	Bellekten kaydediciye taşı	0	1	D	D	D	1	1	0	7
MVI r	Kaydediciye dolaysız taşı	0	0	D	D	D	1	1	0	7
MVI M	Belleğe dolaysız taşı	0	0	1	1	0	1	1	0	10
LXI B	Kaydediciye dolaysız yükle BC çifti	0	0	0	0	0	0	0	1	10
LXI D	Kaydediciye dolaysız yükle DE çifti	0	0	0	1	0	0	0	1	10
LXI H	Kaydediciye dolaysız yükle HL çifti	0	0	1	0	0	0	0	1	10
STAX B	A'yı dolaylı sakla	0	0	0	0	0	0	1	0	7
STAX D	A'yı dolaylı sakla	0	0	0	1	0	0	1	0	7
LDAX B	A'yı dolaylı yükle	0	0	0	0	1	0	1	0	7
LDAX D	A'yı dolaylı yükle	0	0	0	1	1	0	1	0	7
STA	A'yı dolaysız sakla	0	0	1	1	0	0	1	0	13
LDA	A'yı dolaysız yükle	0	0	1	1	1	0	1	0	13
SHLD	HL'i dolaysız sakla	0	0	1	0	0	0	1	0	16
LHLD	HL'i dolaysız yükle	0	0	1	0	1	0	1	0	16
XCHG	DE ve HL kaydedici içeriklerini karşılıklı değiştir	1	1	1	0	1	0	1	1	4
YIĞIN İŞLEMLERİ										
PUSH B	BC kaydedici çiftini yığına it	1	1	0	0	0	1	0	1	11
PUSH D	DE kaydedici çiftini yığına it	1	1	0	1	0	1	0	1	11
PUSH H	HL kaydedici çiftini yığına it	1	1	1	0	0	1	0	1	11
PUSH PSW	A'yı ve Bayrakları yığına it	1	1	1	1	0	1	0	1	11
POP B	BC kaydedici çiftini yığından çıkar	1	1	0	0	0	0	0	1	10
POP D	DE kaydedici çiftini yığından çıkar	1	1	0	1	0	0	0	1	10
POP H	HL kaydedici çiftini yığından çıkar	1	1	1	0	0	0	0	1	10
POP PSW	A'yı ve Bayrakları yığından çıkar	1	1	1	1	0	0	0	1	10
XTHL	Yığının tepesiyle HL'yi karşılıklı değiştir	1	1	1	0	0	0	1	1	18
SPHL	HL'in içeriğini yığın işaretçisine yükle	1	1	1	1	1	0	0	1	5
LXI SP	Hemen yığın işaretçisine yükle	0	0	1	1	0	0	0	1	10
INX SP	Yığın işaretçisini arttır	0	0	1	1	0	0	1	1	5
DCX SP	Yığın işaretçisini azalt	0	0	1	1	1	0	1	1	5
ATLAMA										
JMP	Koşulsuz atla	1	1	0	0	0	0	1	1	10
JC	Elde varsa atla	1	1	0	1	1	0	1	0	10
JNC	Elde yoksa atla	1	1	0	1	0	0	1	0	10
JZ	Sıfırsa atla	1	1	0	0	1	0	1	0	10
JNZ	Sıfır değilse atla	1	1	0	0	0	0	1	0	10
JP	Pozitifse atla	1	1	1	1	0	0	1	0	10
JM	Negatifse atla	1	1	1	1	1	0	1	0	10
JPE	Eşlik çiftse atla	1	1	1	0	1	0	1	0	10
JPO	Eşlik tekse atla	1	1	1	0	0	0	1	0	10
PCHL	HL'in içeriğini program sayıcısına yükle	1	1	1	0	1	0	0	1	5

Kom. Kis.	Tanım	Komut Kodu [1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Saat [2] Periyotları
ÇAĞRI										
CALL	Koşulsuz çağır	1	1	0	0	1	1	0	1	17
CC	Elde varsa çağır	1	1	0	1	1	1	0	0	11/17
CNC	Elde yoksa çağır	1	1	0	1	0	1	0	0	11/17
CZ	Sıfırsa çağır	1	1	0	0	1	1	0	0	11/17
CNZ	Sıfır değilse çağı	1	1	0	0	0	1	0	0	11/17
CP	Pozitifse çağır	1	1	1	1	0	1	0	0	11/17
CM	Negatifse çağır	1	1	1	1	1	1	0	0	11/17
CPE	Eşlik çiftse çağır	1	1	1	0	1	1	0	0	11/17
CPO	Eşlik tekse çağır	1	1	1	0	0	1	0	0	11/17
GERİYE DÖNME										
RET	Geriye dön	1	1	0	0	1	0	0	1	10
RC	Elde varsa geri dön	1	1	0	1	1	0	0	0	5/11
RNC	Elde yoksa geri dön	1	1	0	1	0	0	0	0	5/11
RZ	Sıfırsa geri dön	1	1	0	0	1	0	0	0	5/11
RNZ	Sıfır değilse geri dön	1	1	0	0	0	0	0	0	5/11
RP	Pozitifse geri dön	1	1	1	1	0	0	0	0	5/11
RM	Negatifse geri dön	1	1	1	1	1	0	0	0	5/11
RPE	Eşlik çiftse geri dön	1	1	1	0	1	0	0	0	5/11
RPO	Eşlik tekse geri dön	1	1	1	0	0	0	0	0	5/11
YENİDEN BAŞLATMA										
RST	Yeniden başlat	1	1	A	A	A	1	1	1	11
ARTTIRMA VE AZALTMA										
INR r	Kaydedici içeriğini artır	0	0	D	D	D	1	0	0	5
DCR r	Kaydedici içeriğini azalt	0	0	D	D	D	1	0	1	5
INR M	Bellek içeriğini artır	0	0	1	1	0	1	0	0	10
DCR M	Bellek içeriğini azalt	0	0	1	1	0	1	0	0	10
INX B	BC kaydedici içeriklerini artır	0	0	0	0	0	0	1	1	5
INX D	DE kaydedici içeriklerini artır	0	0	0	1	0	0	1	1	5
INX H	HL kaydedici içeriklerini artır	0	0	1	0	0	0	1	1	5
DCX B	BC'yi azalt	0	0	0	0	1	0	1	1	5
DCX D	DE'yi azalt	0	0	0	1	1	0	1	1	5
DCX H	HL'yi azalt	0	0	1	0	1	0	1	1	5
TOPLAMA										
ADD r	Kaydediciyi A'ya ekle	1	0	0	0	0	S	S	S	4
ADC r	Kaydediciyi, elde ile birlikte A'ya ekle	1	0	0	0	1	S	S	S	4
ADD M	Belleği A'ya ekle	1	0	0	0	0	1	1	0	7
ADC M	Belleği, elde ile birlikte A'ya ekle	1	0	0	0	1	1	1	0	7
ADI	A'ya dolaysız ekle	1	1	0	0	0	1	1	0	7
ACI	A'ya, elde ile dolaysız ekle	1	1	0	0	1	1	1	0	7
DAD B	BC'yi HL'e ekle	0	0	0	0	1	0	0	1	10
DAD D	DE'yi HL'e ekle	0	0	0	1	1	0	0	1	10
DAD H	HL'i HL'e ekle	0	0	1	0	1	0	0	1	10
DAD SP	Yığın işaretçisini HL'e ekle	0	0	1	1	1	0	0	1	10

8080 KOMUT TAKIMI

İşlemci Komutlarının Özeti

Kom. Kıs.	Tanım	Komut Kodu [1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Saat [2] Periyotları
ÇIKARTMA										
SUB r	Kaydediciyi A'dan çıkar	1	0	0	1	0	S	S	S	4
SBB r	Kaydediciyi, ödünç ile birlikte A'dan çıkar	1	0	0	1	1	S	S	S	4
SUB M	Belleği A'dan çıkar	1	0	0	1	0	1	1	0	7
SBB M	Belleği, ödünç ile birlikte A'dan çıkar	1	0	0	1	1	1	1	0	7
SUI	A'dan dolayısız çıkar	1	1	0	1	0	1	1	0	7
SBI	A'dan, ödünç ile dolayısız çıkar	1	1	0	1	1	1	1	0	7
MANTIKSAL										
ANA r	Kaydediciyi A ile VE'le	1	0	1	0	0	S	S	S	4
XRA r	Kaydediciyi A ile Özel VEYA işlemden geçir	1	0	1	0	1	S	S	S	4
ORA r	Kaydediciyi A ile VEYA'la	1	0	1	1	0	S	S	S	4
CMP r	Kaydediciyi A ile karşılaştır	1	0	1	1	1	S	S	S	4
ANA M	Belleği A ile VE'le	1	0	1	0	0	1	1	0	7
XRA M	Belleği A ile Özel VEYA işlemden geçir	1	0	1	0	1	1	1	0	7
ORA M	Belleği A ile VEYA'la	1	0	1	1	0	1	1	0	7
CMP M	Belleği A ile karşılaştır	1	0	1	1	1	1	1	0	7
ANI	Doğrudan A ile VE'le	1	1	1	0	0	1	1	0	7
XRI	Doğrudan A ile Özel VEYA işlemden geçir	1	1	1	0	1	1	1	0	7
ORI	Doğrudan A ile VEYA'la	1	1	1	1	0	1	1	0	7
CPI	Doğrudan A ile karşılaştır	1	1	1	1	1	1	1	0	7
DÖNDÜRME										
RLC	A'yı sola döndür	0	0	0	0	0	1	1	1	4
RRC	A'yı sağa döndür	0	0	0	0	1	1	1	1	4
RAL	A'yı, elde üzerinden sola döndür	0	0	0	1	0	1	1	1	4
RAR	A'yı, elde üzerinden sağa döndür	0	0	0	1	1	1	1	1	4
ÖZEL KOMUTLAR										
CMA	A'nın tümleyenini al	0	0	1	0	1	1	1	1	4
STC	Elde bayrağını kur	0	0	1	1	0	1	1	1	4
CMC	Elde bayrağının tümleyenini al	0	0	1	1	1	1	1	1	4
DAA	A'yı ondalık tabana ayarla	0	0	1	0	0	1	1	1	4
GİRİŞ / ÇIKIŞ										
IN	Giriş	1	1	0	1	1	0	1	1	10
OUT	Çıkış	1	1	0	1	0	0	1	1	10
DENETİM										
EI	Kesmeleri yetkilendir	1	1	1	1	1	0	1	1	4
DI	Kesmeyi yetkisizle	1	1	1	1	0	0	1	1	4
NOP	Hiç işlem yapma	0	0	0	0	0	0	0	0	4
HLT	Dur	0	1	1	1	1	0	1	0	7

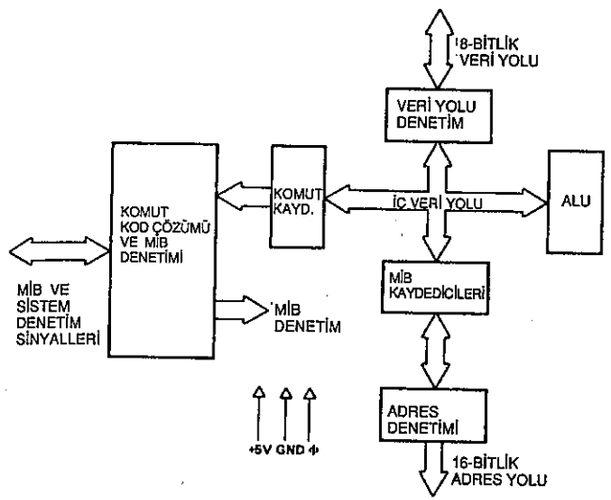
NOTLAR: 1. DDD veya SSS, B=000, C=001, D=010, E=011, H=100, L=101, Bellek = 110, A=111.
2. Olası iki saykıl süresi (6/12), komut saykıllarının koşul bayraklarına bağlı olduğunu gösterir.

EK B Zilog Z80

2.0 Z80-MİB'NİN MİMARİSİ

Z80-MİB'nin iç mimarisinin blok diyagramı Şekil 2.0-1'de gösterilmiştir. Bu diyagram MİB'nin ana elemanlarının tamamını göstermektedir ve aşağıda verilen anlatımın başından sonuna kadar referans olarak kullanılacaktır.

Z80-MİB BLOK DİYAGRAMI



Şekil 2.0-1

2.1 MİB KAYDEDİCİLERİ

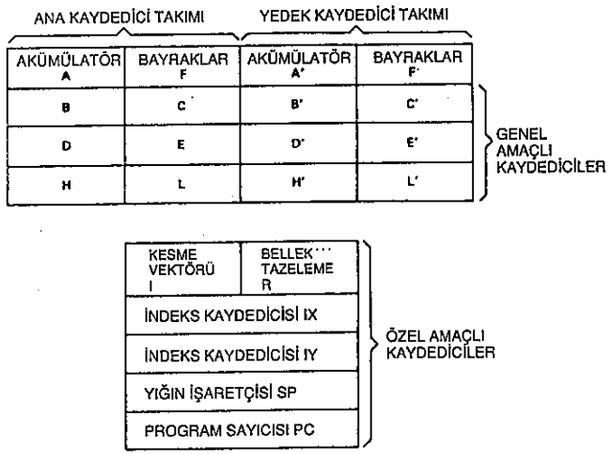
Z80-MİB, programcı tarafından erişilmesi mümkün olan 208-bitlik R/W bellek içerir. Bu belleğin, onsekiz tane 8-bitlik kaydedici ve dört tane 16-bitlik kaydedici halinde nasıl düzenlendiği Şekil 2.0-2'de gösterilmiştir. Tüm Z80 kaydedicileri, statik RAM kullanılarak gerçekleştirilmiştir. Bu kaydediciler, 8-bitlik kaydedici olarak tek başlarına ya da 16-bitlik kaydedici olarak çiftler halinde kullanılabilecek olan altı adet genel amaçlı kaydediciden oluşan iki takımı içerir. Ayrıca, akümülatör ve bayrak kaydedicilerinden oluşan iki takım daha mevcuttur.

Özel Amaçlı Kaydediciler

1. **Program Sayıcısı (PC):** Program sayıcısı bellekten getirilecek olan komutun 16-bitlik adresini tutar. PC, içeriğinin adres hatlarına aktarılmasından sonra otomatik olarak arttırılır. Herhangi bir program atlama komutu yürütüldüğünde, arttırıcı aşılarak, yeni değer PC'ye otomatik olarak yerleştirilir.

2. **Yığın İşaretçisi (SP):** Yığın işaretçisi, dış sistem RAM belleğindeki herhangi bir yere yerleştirilmiş olan bir yığının yürürlükteki tepesinin 16-bitlik adresini tutar. Dış yığın belleği, son giren ilk çıkar (LIFO) düzeninde bir dosya şeklinde düzenlenir. PUSH ve POP komutlarının yürütülmesi yoluyla, yığına belirli MİB kaydedicilerinden veri itilebilir ya da yığından belirli MİB kaydedicilarına veri çekilebilir. Yığından çekilen veri, her zaman yığına daha önce itilmiş olan son veridir. Yığın, çok düzeyli kesmelerin kolayca gerçeklenmesini, sınırsız sayıda alt yordamın iç içe geçirilebilmesini, ve bir çok veri işleme türlerinin basitleştirilmesini sağlar.

Z80-MİB KAYDEDİCİ DÜZENİ



ŞEKİL 2.0-2

3. **İndeks Kaydedicileri (IX ve IY):** Birbirinden bağımsız iki adet indeks kaydedicisi, indekslenmiş adresleme modlarında kullanılan 16-bitlik bir taban adresini tutar. Bu adresleme modunda indeks kaydedicilarından biri, verinin bellekte saklanacağı ya da geri alınacağı bölgeyi işaret etmek için bir taban olarak kullanılır. Bu tabandan itibaren olan kaymayı belirtmek için, indekslenmiş komutlarda ilave bir bayt içerilir. Bu kayma, ikinin tümleyeni halindeki işaretli bir tamsayı olarak belirtilir. Bu adresleme modu, özellikle veri tablolarının kullanıldığı bir çok program türlerini büyük ölçüde basitleştirir.

4. **Kesme Sayfası Adres Kaydedicisi (I):** Z80-MİB, herhangi bir kesmeye cevap olarak herhangi bir bellek konumuna dolaylı bir çağrının gerçekleştirilebileceği bir modda çalıştırılabilir. Bu amaçla I kaydedicisi, dolaylı adresin üst sıralı 8-bitini saklamak için kullanılır, adresin alt 8-bitini ise kesmeyi yapan birim sağlar. Bu özellik, kesme yordamlarının bellekteki herhangi bir yere dinamik olarak yerleştirilebilmelerini sağlar, ayrıca yordama erişim süresinin de mutlak minimum değerine indirilmesini sağlar.

5. **Bellek Tazeleme Kaydedicisi (R):** Z80-MİB, statik belleklerle aynı kolaylıkla kullanılan dinamik bellekleri yetkilendirmek için bir bellek tazeleme sayıcısını içermektedir. Bu 7-bitlik kaydedici, her komut getirme saykılından sonra otomatik olarak arttırılır. MİB, getirilen komutun kodunu çözerken ve yürütürken tazeleme sayıcısındaki veri, bir tazeleme denetim sinyali ile birlikte adres yolunun alt yarısı üzerinden dışarıya gönderilir. Bu tazeleme modu, programcıya bütünüyle açıktır ve MİB'nin çalışmasını yavaşlatmaz. Programcı R (Tazeleme) kaydedicisini, test etme amaçları için yükleyebilir, fakat bu kaydedici normalde programcı tarafından kullanılmaz.

Akümülatör ve Bayrak Kaydedicileri

MİB, birbirinden bağımsız iki adet 8-bitlik akümülatörü ve bunlarla birleştirilmiş 8-bitlik bayrak kaydedicilerini içermektedir. Bayrak kaydedicisi 8 ya da 16-bitlik işlemler için, örneğin bir işlemin sonucunun sıfıra eşit olup olmadığının belirtilmesi gibi özel koşulları gösterirken, akümülatör 8-bitlik aritmetik ya da mantıksal işlemlerin sonuçlarını tutar. Programcı, çalışmak istediği akümülatör ve bayrak çiftini tek bir değiş-tokuş komutu ile seçer. Bu suretle, programcının çiftlerden herhangi biri ile kolaylıkla çalışması mümkündür.

Genel Amaçlı Kaydediciler

Z80-MİB'nde genel amaçlı kaydedicilerden oluşan, eşleştirilmiş iki kaydedici takımı mevcuttur. Bu takımların her ikisi de, programcı tarafından 8-bitlik kaydedici olarak tek başlarına ya da 16-bitlik kaydedici olarak çiftler halinde kullanılabilecek olan altı adet 8-bitlik kaydedici içermektedir. Takımlardan birincisindeki kaydedici çiftleri, BC, DE, ve HL ile adlandırılırlar, eşlenik takımdakiler ise BC', DE', ve HL' şeklinde adlandırılır. Programcı, çalışacağı kaydedici takımını, bütün takım için tek bir değiş-tokuş komutu yardımıyla seçebilir. Hızlı kesme cevabının gerekli olduğu sistemlerde, genel amaçlı kaydediciler ile akümülatör/bayrak kaydedicisinden oluşan takımlardan birisi, bu çok hızlı yordamın işletilmesine ayırılabilir. Yordamdan yordama geçiş için sadece tek bir değiştokuş komutunun yürütülmesi gerekir. Bu özellik, kesme ya da alt yordam işlemleri sırasında kaydedici içeriklerini dış yığına saklama ve geri alma gereksinimlerini ortadan kaldırmak suretiyle, kesmeye cevap verme süresini büyük ölçüde azaltır. Bu genel amaçlı kaydediciler, programcı tarafından çok geniş aralıktaki uygulamalar için kullanılırlar. Ayrıca, özellikle çok az dış oku/yaz belleğinin mevcut olduğu ROM-tabanlı sistemlerde programlamayı da basitleştirirler.

2.2 ARİTMETİK VE MANTIK BİRİMİ (ALU)

MİB'nin 8-bitlik aritmetik ve mantıksal komutları, ALU'da yürütülürler. ALU, MİB içerisinde iç veri yolu üzerinden, kaydediciler ve dış veri yoluyla iletişimde bulunur. ALU tarafından yerine getirilen işlevlerin türleri şöyledir:

Toplama	Sola ya da sağa kaydırma ya da döndürme (aritmetik ve mantıksal)
Çıkartma	Arttırma
Mantıksal VE	Azaltma
Mantıksal VEYA	Bit kurma
Mantıksal Özel VEYA	Bit sıfırlama
Karşılaştırma	Bit test etme

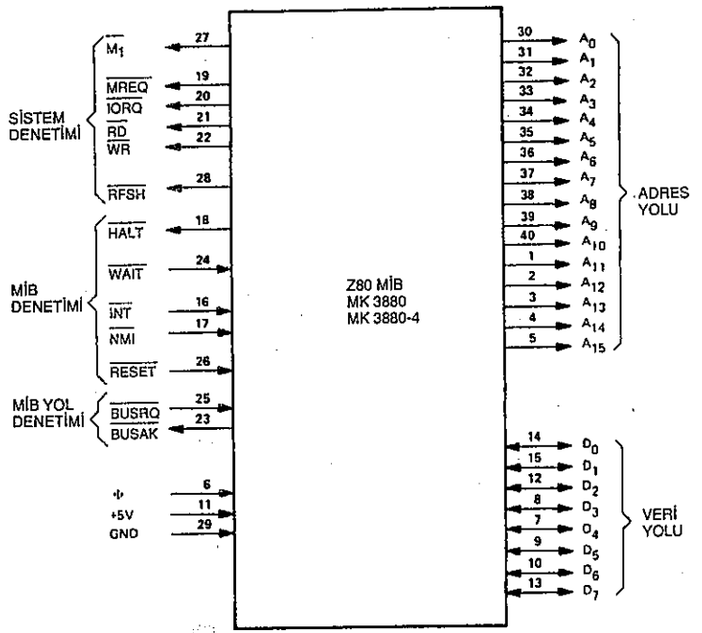
2.3 KOMUT KAYDEDİCİSİ VE MİB DENETİMİ

Bellekten her komut getirme işleminde, getirilen komut komut kaydedicisine yerleştirilerek kodu çözülür. Bu işlevi denetim bölümü yerine getirir ve sonra kaydedicilerden veri okumak ya da kaydedicilere veri yazmak için gerekli olan bütün denetim sinyallerini üretir ve sürer, ALU'yu denetler ve gerekli olan bütün dış denetim sinyallerini sağlar.

3.0 Z80-MİB BACAKLI TANIMLARI

Z80-MİB, 40 bacaklı bir endüstri standardı olan Çift-Sıralı (DIL) Paket içinde paketlenmiştir. G/Ç bacakları Şekil 3.0-1'de gösterilmiş ve bu bacakların her birinin işlevleri şeklin altında tanımlanmıştır.

Z80 BACAK DÜZENİ



ŞEKİL 3.0-1

A_0-A_{15} Üç-durumlu yüksekte-etkin çıkış. A_0-A_{15}, (Adres Yolu) 16-bitlik bir adres yolu oluşturur. Adres yolu, (64 Kbayta kadar) bellek veri değiş-tokuşları ve G/Ç birimleri veri değiş-tokuşları için gerekli adresi sağlar. G/Ç adreslemesi, kullanıcının 256 adete kadar giriş ya da çıkış portunu doğrudan doğruya seçebilmesini sağlamak üzere alt sıralı 8 adres bitini kullanır. A_0 en küçük değerlikli adres bitidir. Tazeleme süresi sırasında, alt sıralı 7 bit geçerli bir tazeleme adresini içerir.

D_0-D_7 Üç-durumlu yüksekte-etkin giriş/çıkış. D_0-D_7, (Veri Yolu) 8-bitlik bir iki yönlü veri yolu oluşturur. Veri yolu, bellek ve G/Ç birimleri ile veri değiş-tokuşu için kullanılır.

$\overline{M_1}$ Alçakta-etkin çıkış. $\overline{M_1}$, yürürlükteki makine (Birinci Makine Saykılı) saykılının, bir komutun yürütümünün OP kodu getirme saykılı olduğunu belirtir. 2-baytlık op-kodlarının yürütümü sırasında, $\overline{M_1}$'in her iki op kodu baytının getirilmesi sırasında da üretildiğine dikkat ediniz. Bu iki baytlık op-kodları her zaman CBH, DDH, EDH, ya da FDH ile başlar. $\overline{M_1}$ ayrıca, kesme alındı saykılını belirtmek için $\overline{IORQ}$ ile de ortaya çıkar.

$\overline{MREQ}$ Üç-durumlu alçakta-etkin çıkış. Bellek (Bellek İsteği) isteği sinyali, adres yolunun bellek okuma ya da bellek yazma işlemi için geçerli bir adresi tutmakta olduğunu belirtir.

$\overline{IORQ}$ Üç-durumlu alçakta-etkin çıkış. $\overline{IORQ}$ sinyali, (Giriş/Çıkış İsteği) adres yolunun alt sıralı yarısının, G/Ç okuma ya da yazma işlemi için geçerli bir G/Ç adresini tutmakta olduğunu belirtir. $\overline{IORQ}$ sinyali ayrıca, herhangi bir kesme alındığında da, veri yoluna kesme yanıt vektörünün yerleştirilebileceğini belirtmek için M_1 sinyali ile birlikte üretilir. Kesme Alındı işlemleri M1 süresi sırasında gerçekleşir, G/Ç işlemleri ise hiç bir zaman M1 süresi sırasında gerçekleşmez.

$\overline{RD}$ Üç-durumlu alçakta-etkin çıkış. $\overline{RD}$, MİB'nin (Bellekten Oku) bellek ya da bir G/Ç biriminden veri okumak istediğini belirtir. Adreslenen G/Ç aygıtı ya da bellek, bu sinyali, veriyi MİB'nin veri yoluna kapılamak için kullanmalıdır.

$\overline{WR}$ Üç-durumlu alçakta-etkin çıkış. $\overline{WR}$, MİB veri (Belleğe Yaz) yolunun, adreslenen bellek ya da G/Ç biriminde saklanacak olan geçerli bir veri tutmakta olduğunu belirtir.

$\overline{RFSH}$ Alçakta-etkin çıkış. $\overline{RFSH}$, adres yolunun alt (Tazele) sıralı 7 bitinin, dinamik bellekler için bir tazeleme adresi içerdiğini ve yürürlükteki $\overline{MREQ}$ sinyalinin, tüm dinamik bellekler için tazeleme amaçlı bir okuma işlemini yapmak amacıyla kullanılması gerektiğini belirtir. Tazeleme sırasında, A_7 adres hattı mantıksal sıfırdır ve Adres Yolunun üst sıralı 8-biti, İ.Kaydedicisini içerir.

$\overline{HALT}$ Alçakta-etkin çıkış. $\overline{HALT}$, MİB'nin bir HALT (Durdurma Durumu) yazılım komutunu yürütmüş olduğunu ve işlemin geri alınabileceği hale gelinmeden önce, maskelenemez ya da (maske yetkilenmişken) maskelenebilir bir kesme için hazır beklediğini belirtir. Durdurma durumuna alınmışken MİB, bellek tazeleme etkinliğini sürdürmek için sürekli olarak NOP komutları yürütür.

$\overline{WAIT}$ Alçakta-etkin giriş. $\overline{WAIT}$ (Bekle) girişi Z80-MİB'e, adreslenen bellek ya da G/Ç biriminin veri aktarma için hazır olmadığını belirtir. MİB, bu sinyal aktif olduğu sürece, bekleme durumları girmeye devam eder. Bu sinyal, herhangi bir hızdaki bellek ya da G/Ç birimlerinin MİB ile senkronize edilmesini sağlar.

$\overline{INT}$ Alçakta-etkin giriş. Kesme İstek sinyali, G/Ç (Kesme İsteği) birimleri tarafından üretilir. Kesme isteği, eğer yazılım denetimli iç kesme yetkilendirme flip-flobu (IFF) yetkilendirilişse ve $\overline{BUSRQ}$ sinyali aktif değilse, yürürlükteki komutun sonunda dikkate alınır. MİB kesmeyi aldığında, bir sonraki komut saykılının başlangıcında, dışarıya bir alındı sinyali (M_1 süresi sırasındaki $\overline{IORQ}$) gönderilir. MİB, bir kesmeyi, bölüm 8'de ayrıntılı olarak anlatılan üç ayrı modda yanıtlayabilir.

$\overline{NMI}$ Negatif kenar tetiklemeli giriş. Maskelenemez kesme isteği girişi, $\overline{INT}$'den daha yüksek bir önceliğe sahiptir ve kesme yetkilendirme flip-flobunun durumundan bağımsız olarak her zaman için yürürlükteki komutun sonunda tanınır. $\overline{NMI}$, Z80-MİB'ni otomatik olarak 0066H konumundan yeniden başlatır. Program sayıcı, kullanıcının kesilen programa geri dönebilmesi için otomatik olarak dış yığında saklanır. Ardarda girilen WAIT (BEKLE) saykıllarının yürürlükteki komutun sonlanmasını engelleyebileceğine ve herhangi bir $\overline{BUSRQ}$'un $\overline{NMI}$'yı aşacağına dikkat ediniz.

$\overline{RESET}$ Alçakta-etkin giriş. $\overline{RESET}$, program sayıcısına sıfır yükler ve MİB'i başlangıç durumuna getirir. MİB'i başlangıç durumuna getirme işlemi aşağıdaki aşamaları kapsar:

1) Kesme yetkilendirme flip-flobunun yetkisiz kılınması

2) I kaydedicisinin 00H'e kurulması
3) R kaydedicisinin 00H'e kurulması
4) 0'ncı Kesme Modunun kurulması

Sıfırlama süresi sırasında, adres ve veri yolları yüksek empedans durumuna girerler ve tüm denetim çıkış sinyalleri de etkisiz duruma gider. Bu sırada, tazeleme işlemi de yapılmaz.

$\overline{\text{BUSRQ}}$ Alçakta-etkin giriş. Yol isteği sinyali, (Yol İsteği) MİB'nin adres yolu, veri yolu ve üç-durumlu çıkış denetim sinyallerinin yüksek empedans durumuna geçmesini istemek için kullanılır, böylece diğer birimlerin bu yolları denetlemesi mümkün olur. $\overline{\text{BUSRQ}}$ aktif duruma getirildiğinde, MİB, yürürlükteki MİB makine saykılı sonlandırılır sonlandırılmaz bu yolları yüksek empedans durumuna getirir.

$\overline{\text{BUSAK}}$ Alçakta-etkin çıkış. Yol alındısı, istek (Yol Alındısı) yapan birime, MİB'in adres yolu, veri yolu ve üç-durumlu denetim yolu sinyallerinin yüksek empedans durumlarına geçirildiklerini ve artık dış birimin bu sinyalleri denetleyebileceğini belirtmek için kullanılır.

Φ Tek fazlı sistem saati.

* Z80-MİB, $\overline{\text{WAIT}}$ (BEKLE) ya da Yol Alındısı durumunda iken, Dinamik Bellek Tazeleme işlemi yapılmaz.

4.0 MİB ZAMANLAMASI

Z80-MİB, komutları, birkaç temel işlemden oluşan çok hassas bir takım işlemlerden adım adım ilerlemek suretiyle yürütür. Bu işlemler şunlardır:

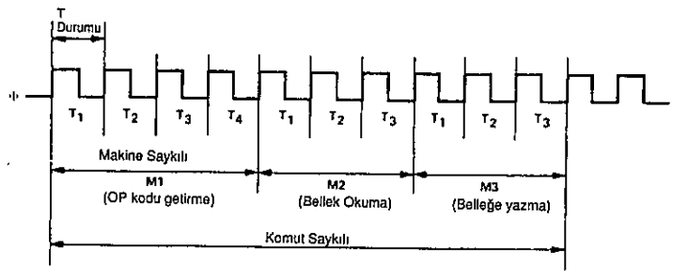
Bellek okuma ya da yazma

G/Ç birimin okunması ya da yazılması

Kesme alındısı

Bütün komutlar yalnızca, bu temel işlemlerden oluşan birer diziden ibarettirler. Bu temel işlemlerin her birinin tamamlanması için üçden altıya kadar olabilen sayıda saat peryodu gerekir, ancak MİB'i dış birimlerin hızına senkronize etmek için saat peryodu sayısı arttırılabilir. Temel saat peryotları, T durumları ile, temel işlemler ise M (makine) saykılları ile gösterilirler. Şekil 4.0-0'da, tipik bir komutun nasıl yalnızca belli M ve T saykıllarından oluşan bir dizi olduğu gösterilmiştir. Bu komutun üç makine saykılından (M1, M2 ve M3) oluştuğuna dikkat ediniz. Herhangi bir komutun ilk makine saykılı, (bir sonraki bölümde tümüyle anlatılacağı gibi, bekletme sinyali ile uzatılmadığı taktirde) dört, beş ya da altı T durumu uzunluğunda olan bir getirme saykılıdır. Getirme saykılı (M1), yürütülecek olan bir sonraki komutun OP kodunu getirmek için kullanılır. Sonra gelen makine saykılları , MİB ile bellek ya da G/Ç birimleri arasında veri taşırlar ve üçten beşe kadar olabilen herhangi bir sayıdaki T peryotları uzunluğunda olabilirler (bu makine saykılları da, dış birimleri MİB'ne senkronize etmek için bekleme durumları ile uzatılabilirler). Aşağıdaki paragraflar, herhangi bir temel makine saykılında gerçekleşen zamanlamayı anlatmaktadır.

TEMEL MİB ZAMANLAMA ÖRNEĞİ



ŞEKİL 4.0-0

Bütün MİB zamanlamaları, Şekil 4.0-1'den 4.0-7'ye kadar olan şekillerde gösterildiği gibi, çok az sayıdaki çok basit zamanlama diyagramlarına bölünebilirler. Bu diyagramlar, bekleme durumları ile birlikte (bekleme durumları, MİB'ni yavaş bellek ya da G/Ç birimlerine senkronize etmek için makine saykılına eklenirler) ya da bekleme durumları olmaksızın, aşağıdaki temel işlemler olarak ortaya çıkabilirler.

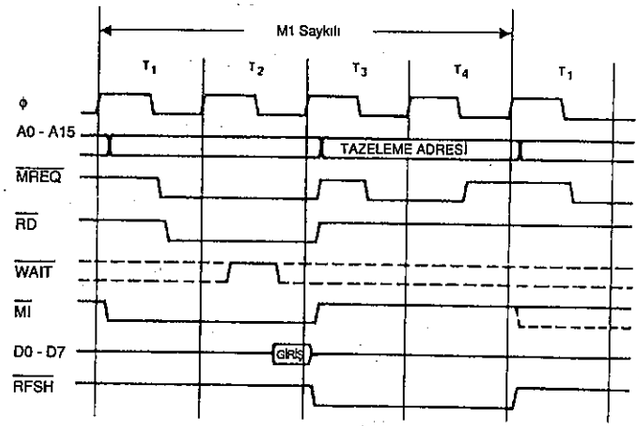
4.0-1. Komut OP kodu getirme (M1 saykılı)
4.0-2. Bellek veri okuma ya da yazma saykılları
4.0-3. G/Ç okuma ya da yazma saykılları
4.0-4. Yol İsteği/Alındısı Saykılı
4.0-5. Kesme İsteği/Alındısı Saykılı

4.0-6. Maskelenemez Kesme İsteği/Alındısı Saykılı
4.0-7. Herhangi bir HALT komutundan çıkış

KOMUT GETİRME

Şekil 4.0-1'de, M1 (OP kodu getirme) saykılına ilişkin zamanlama görülmektedir. PC'nin, adres yoluna M1 saykılının başında yerleştirildiğine dikkat ediniz. Saat peryodunun yarısına eşit uzunluktaki bir süre sonunda $\overline{MREQ}$ sinyali aktif duruma gider. Bu süre, belleğe uygulanan adresin kararlı hale gelmesi için yeterlidir, bu yüzden $\overline{MREQ}$'nun düşen kenarı, dinamik bellekler için doğrudan yonga yetkilendirme saati olarak kullanılabilir. Aynı zamanda $\overline{RD}$ hattı da, bellek okuma verisinin, MİB'in veri yoluna açılması gerektiğini belirtmek için aktif duruma gider. MİB, veri yolundaki bellekten gelen veriyi, T3 durumundaki saatin yükselen kenarıyla örnekler ve aynı kenar, MİB tarafından $\overline{RD}$ ve $\overline{MREQ}$ sinyallerinin etkisiz duruma getirilmesi için de kullanılır. Böylece veri, $\overline{RD}$ sinyali etkisiz hale getirilmeden önce, MİB tarafından örneklenmiş olur. Getirme saykılının T3 ve T4 saat durumları, dinamik bellekleri tazelemek için kullanılır. (MİB bu süreyi, getirilen komutun kodunu çözmek ve yürütmek için kullanır, bu yüzden bu süre boyunca başka hiç bir işlem yapılamaz). T3 ve T4 sırasında, adres yolunun alt sıralı 7 biti, bellek tazeleme adresini içerir ve $\overline{RFSH}$ sinyali, tüm dinamik bellekler için tazeleme amaçlı bir okuma işleminin gerçeklenmesi gerektiğini belirtmek amacıyla aktif duruma gider. Tazeleme süresi boyunca, değişik bellek dilimlerinden gelen verinin, veri yoluna kapılanmasını önlemek için $\overline{RD}$ sinyalinin üretilmediğine dikkat ediniz. Tazeleme süresi boyunca $\overline{MREQ}$ sinyali, tüm bellek elemanları için tazeleme amaçlı bir okuma işlemini gerçekleştirmek için kullanılmalıdır. Tazeleme adresinin kararlılığı sadece $\overline{MREQ}$ süresi boyunca garanti edildiğinden, tazeleme sinyali kendi başına kullanılamaz.

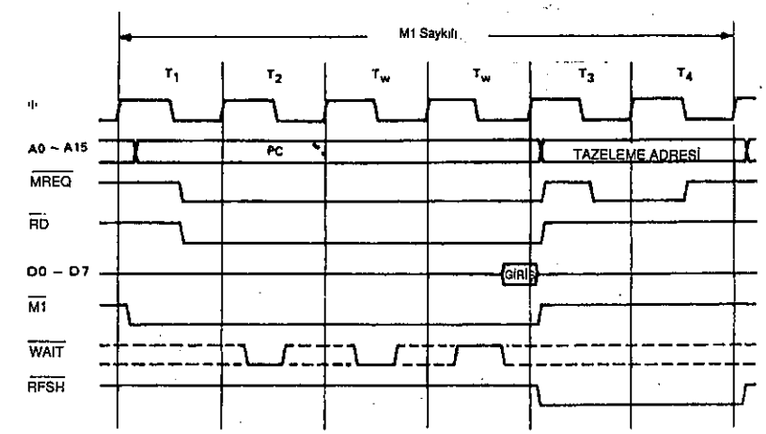
KOMUT OPKODU GETİRME SAYKILI



ŞEKİL 4.0-1

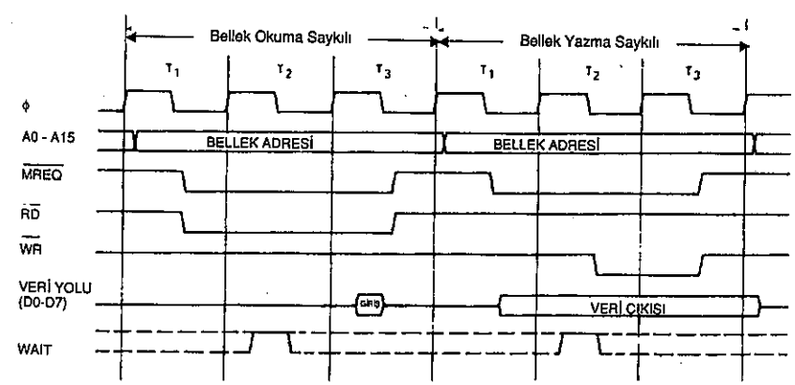
Şekil 4.0-1A, belleğin $\overline{WAIT}$ hattını aktif duruma getirmesi halinde, getirme saykılının nasıl geciktirildiğini göstermektedir. MİB, T2 durumu ve bu durumu izleyen her TW durumu sırasında, φ'nin düşen kenarı ile $\overline{WAIT}$ hattını örnekler. Eğer örnekleme anında $\overline{WAIT}$ hattı aktif durumda ise, bir sonra gelen saat saykılında, makine saykılına yeni bir bekleme durumu girilir. Bu teknik kullanılarak okuma saykılı, herhangi bir türden olan bellek biriminin erişim süresine uygun olarak uzatılabilir.

BEKLEME DURUMLARI İLE BİRLİKTE KOMUT İŞKODUNU GETİRME SAYKILI



ŞEKİL 4.0-1A

BELLEK OKUMA YA DA YAZMA SAYKILLARI

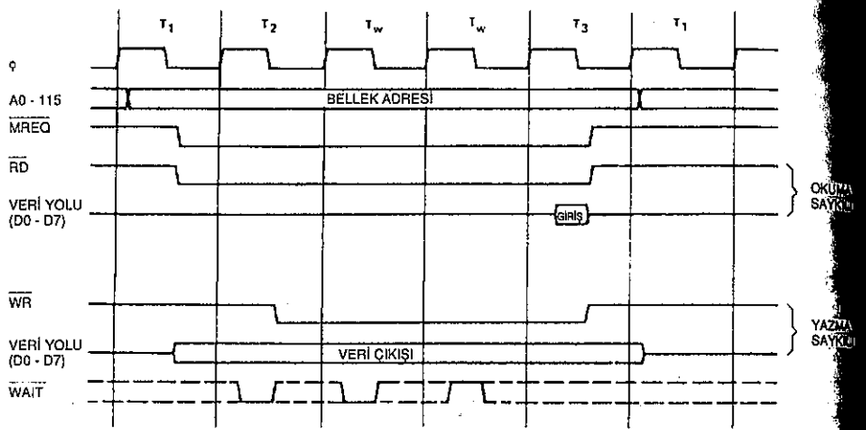


ŞEKİL 4.0-2

BELLEKTEN OKUMA YA DA BELLEĞE YAZMA

Şekil 4.0-2'de, Opkodu getirme saykılı (M1 saykılı) dışındaki bellek okuma ya da yazma sa rının zamanlaması görülmektedir. Bu saykılar, bellek tarafından $\overline{WAIT}$ sinyali yoluyla be durumları istenmediği taktirde, genellikle üç saat peryodu uzunluğundadırlar. $\overline{MREQ}$ ve sinyalleri, tıpkı getirme saykıllarında kullanıldıkları gibi kullanılırlar. Belleğe yazma saykılı dur da, $\overline{MREQ}$ sinyali, adres yolu kararlı hale geldiğinde de aktif olur, böylece bu sinyalin dı bellekler için doğrudan yonga yetkilendirme sinyali olarak kullanılması mümkün olur. $\overline{WR}$ ha yolundaki veri kararlı hale geldiğinde aktif olur, bu yüzden fiilen herhangi bir yarıiletken bell için R/W (Oku/Yaz) darbesi olarak doğrudan kullanılması mümkündür. Ayrıca $\overline{WR}$ sinyali ve veri yolu içerikleri değiştirilmeden yarım T durumu süresi önce etkisiz duruma gider, bö fiilen herhangi bir yarıiletken bellek türü için gerekli olan üst üste binme gereksinimleri karşı olur.

BEKLEME DURUMLARI İLE BİRLİKTE BELLEK OKUMA YA DA YAZMA SAYKILLARI



ŞEKİL 4.0-2A

Şekil 4.0-2A, herhangi bir $\overline{WAIT}$ (BEKLE) isteği sinyalinin, bellek okuma ya da yazma işlemini nas uzatacağını göstermektedir. Bu işlem, getirme saykılı için daha önce anlatılmış olanla aynıdır. Şek 4.0-2A'da, okuma ve yazma saykılları hiç bir zaman eşzamanlı olarak ortaya çıkmadıkları halde, ayrık birer okuma ve yazma saykılının aynı şekil içinde gösterilmiş olduğuna dikkat ediniz.

GİRİŞ YA DA ÇIKIŞ SAYKILLARI

Şekil 4.0-3'de, herhangi bir G/Ç okuma ya da yazma işlemi görülmektedir. G/Ç işlemlerinde, tek bir bekleme durumunun makine saykılına otomatik olarak girildiğine dikkat ediniz. Bunun nedeni, G/Ç işlemlerinde, $\overline{IORQ}$ sinyalinin etkin duruma gittiği andan MİB'nin $\overline{WAIT}$ hattını örneklemesi gereken ana kadar olan sürenin çok kısa olması ve bu ilave durum olmadan herhangi bir G/Ç portunun, adresin kodunu çözmesi ve eğer bir bekleme durumu gerekiyorsa, $\overline{WAIT}$ hattını aktif duruma getirmesi için yeterli sürenin olmayışıdır. Ayrıca bu bekleme durumu olmadan, tam MİB

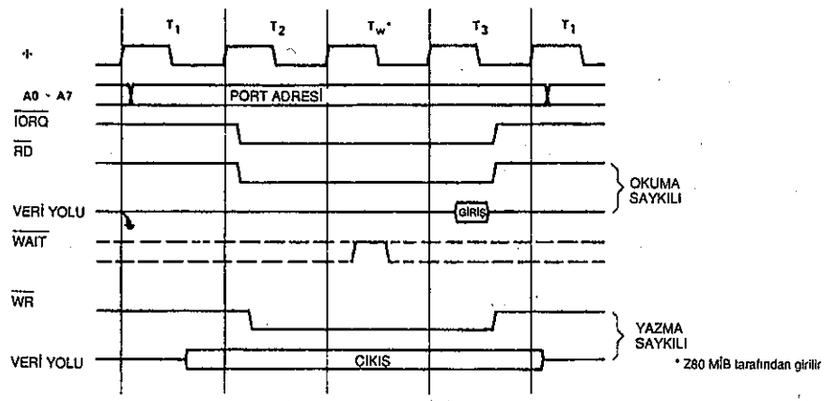
hızında çalışabilen MOS G/Ç birimlerini tasarlamak da zordur. Bu bekleme durumu süresi sırasında, $\overline{WAIT}$ (BEKLEME) isteği sinyali örneklenir. G/Ç okuma işleminde $\overline{RD}$ hattı, tıpkı bellek okumada olduğu gibi, adreslenen portu veri yoluna erklemek için kullanılır. G/Ç yazma işleminde ise, $\overline{WR}$ hattı G/Ç portu için saat olarak kullanılır, yükselen kenarın veri saati olarak kullanılabilmesi için gerekli örtüşme süresi yine otomatik olarak sağlanır.

Şekil 4.0-3A'da, ilave bekleme durumlarının $\overline{WAIT}$ hattı ile nasıl eklenebilecekleri gösterilmiştir. Bunun için gerekli işlem, daha önce anlatılanla aynıdır.

YOL İSTEĞİ/ALINDISI SAYKILI

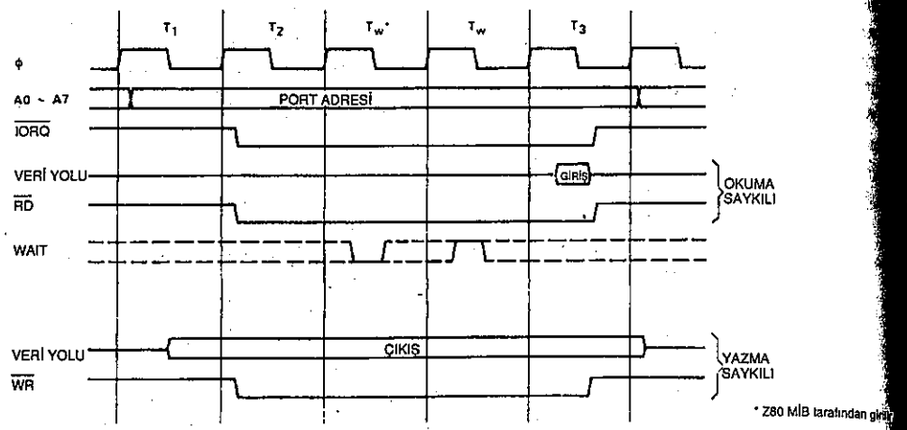
Şekil 4.0-4'de, Yol İsteği/Alındısı saykılının zamanlaması gösterilmiştir. $\overline{BUSRQ}$ sinyali, MİB tarafından, herhangi bir makine saykılının son saat peryodunun yükselen kenarı ile örneklenir. Eğer $\overline{BUSRQ}$ sinyali aktif durumda ise, MİB, adres, veri ve üç-durumlu denetim sinyallerini, bir sonraki saat darbesinin yükselen kenarı ile yüksek empedans durumuna getirir. Bu durumda, herhangi bir dış birimin, MİB'nin yollarını, bellek ile G/Ç birimleri arasında veri aktarmak için denetlemesi mümkündür. (Bu durum genellikle, saykıl çalma yolunu kullanan Doğrudan Bellek Erişimi [DMA] olarak bilinir). MİB'nin herhangi bir yol isteğine yanıt vermesi için gerekli maksimum süre, bir makine saykılının uzunluğuna eşittir ve dış denetleyici, yol denetimini istediği sayıdaki saat peryotları boyunca sürdürebilir. Bununla birlikte, eğer çok uzun süreli DMA saykılları kullanılacaksa ve aynı zamanda dinamik bellekler kullanılıyorsa, bu durumda dış denetleyicinin tazeleme işlemini de yerine getirmesi gerekir. Bu durum, sadece, eğer DMA denetimi altında çok büyük veri blokları aktarılması halinde ortaya çıkar. Ayrıca, herhangi bir yol isteği saykılı sırasında, MİB'nin, ne bir $\overline{NMI}$ ne de bir $\overline{INT}$ sinyali tarafından kesmeye uğratılamayacağına da dikkat ediniz.

GİRİŞ YA DA ÇIKIŞ SAYKILLARI



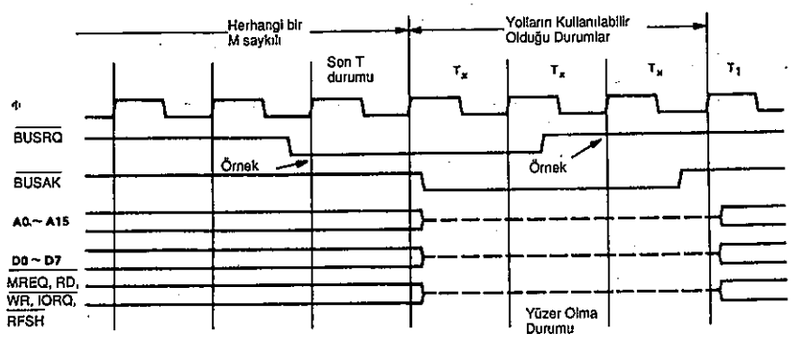
ŞEKİL 4.0-3

BEKLEME DURUMLARI İLE BİRLİKTE GİRİŞ YA DA ÇIKIŞ SAYKILLARI



ŞEKİL 4.0-3A

YOL İSTEĞİ/ALINDISI SAYKILI



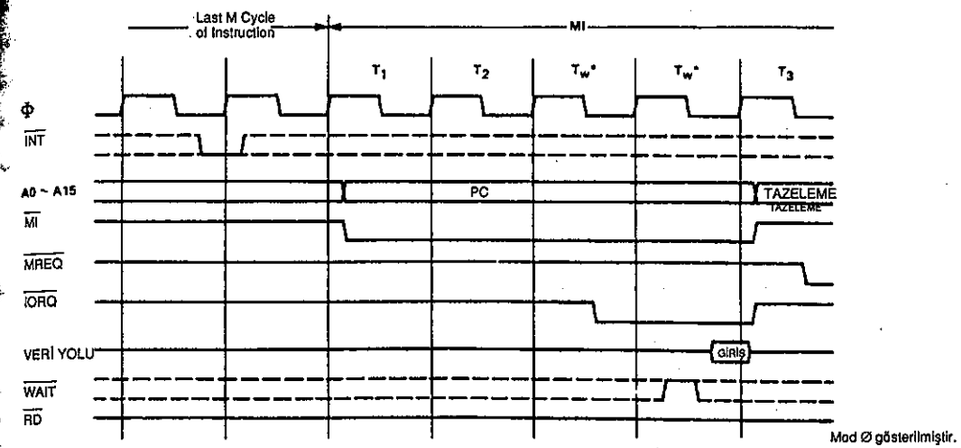
ŞEKİL 4.0-4

KESME İSTEĞİ/ALINDISI SAYKILI

Şekil 4.0-5'de, herhangi bir kesme saykılına ilişkin zamanlama görülmektedir. Kesme sinyali ($\overline{INT}$), MİB tarafından herhangi bir komutun sonundaki son saatin yükselen kenarı ile örneklenir. MİB'ndeki

yazılım denetimli iç kesme yetkilendirme flip-flobu kurulmamış ya da $\overline{BUSRQ}$ sinyali etkin durumda ise, kesme sinyali alınmaz. Bu sinyal alındığında, özel bir M1 saykılı üretilir. Bu özel M1 saykılı sırasında (normal olarak $\overline{MREQ}$ yerine) $\overline{IORQ}$ sinyali, kesmeye neden olan birimin, veri yoluna 8-bitlik bir vektör yerleştirebileceğini belirtmek için etkin duruma geçer. Bu saykıla, otomatik olarak iki adet bekleme durumunun eklenildiğine dikkat ediniz. Bu durumlar, asenkron öncelikli kesme yapısının kolaylıkla gerçeklenebilmesi için yetkilenirler. Bu bekleme durumları, asenkron sinyallerin kararlı hale gelerek hangi G/Ç biriminin cevap vektörünü girmesi gerektiğini belirlemeleri için yeterli süreyi sağlarlar. Kesme cevap vektörünün, MİB tarafından nasıl kullanıldığına ilişkin ayrıntılar için Bölüm 8.0'a bakınız.

KESME İSTEĞİ/ALINDISI SAYKILI



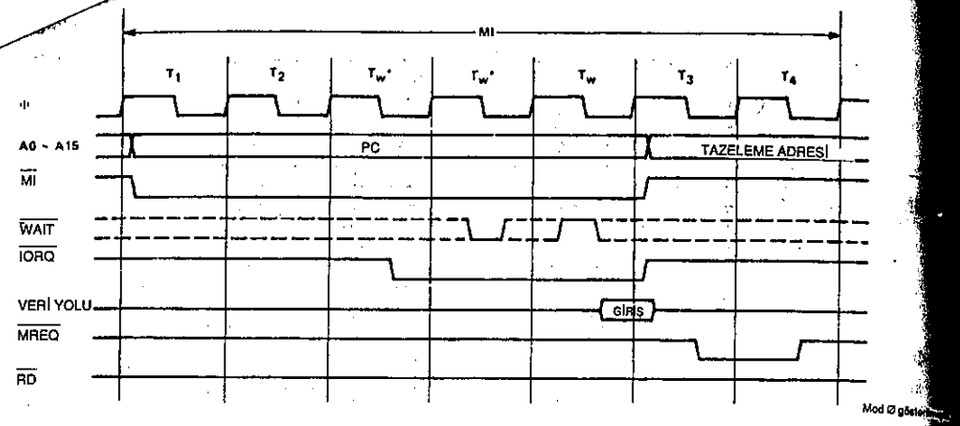
ŞEKİL 4.0-5

Şekil 4.0-5A'da, kesme cevabı saykılına ilave bekleme durumlarının nasıl eklenebileceği görülmektedir. Bu işlem de, daha önce anlatılanla aynıdır.

MASKELENEMEZ KESME CEVABI

Şekil 4.0-6'da, maskelenemez kesme için istek/alındı saykılı görülmektedir. $\overline{NMI}$ girişine uygulanan darbe, MİB tarafından her komutun sonunda test edilen bir iç NMI mandalını 1'e kurar. Bu NMI mandalı, maskelenemez kesme girişi ile aynı anda örneklenir, fakat bu giriş, normal kesme girişine göre daha yüksek bir önceliğe sahiptir ve yazılım denetimi altında yetkisiz kılınamaz. Bu kesme girişinin bilinen görevi, örneğin, olması muhtemel bir güç arızası gibi kritik sinyallere hemen yanıt verilmesidir. MİB'nin maskelenemez bir kesmeye yanıtı, normal bir bellek okuma işlemine benzer. Aradaki tek fark, işlemcinin, veri yolu içeriğini göz önüne almayıp, PC'yi otomatik olarak dış yığında sakladıktan sonra 0066H konumuna atlamasıdır. Eğer maskelenemez kesme kullanılacaksa, kesme hizmet yordamı bu konumdan başlamalıdır.

RI İLE BİRLİKTE KESME İSTEĞİ/ALINDISI SAYKILI

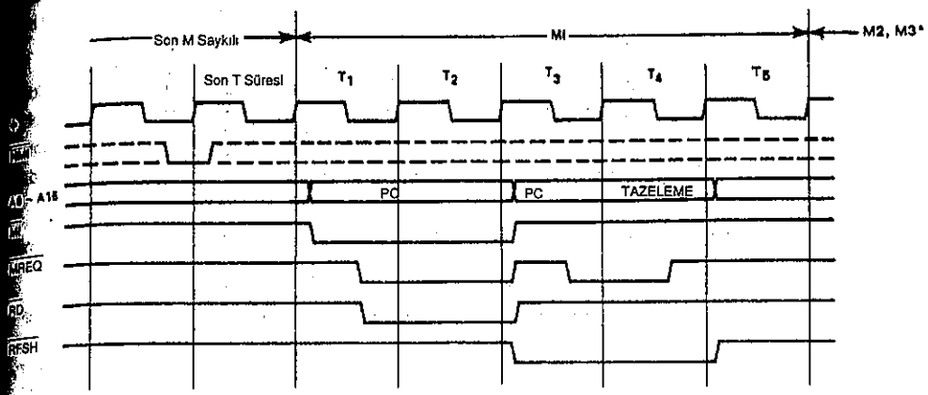


ŞEKİL 4.0-5A

DURDURMA DURUMUNDAN ÇIKIŞ

Yazılımsal bir durdurma komutu yürütüldüğünde, MİB, (kesme flip-flobu yetkilenmiş durumdayden, maskelenemez ya da maskelenebilir olabilen) herhangi bir kesme alınıncaya kadar, sürekli olarak NOP komutları yürütmeye başlar. Şekil 4.0-7'de gösterildiği gibi, kesme girişlerinin her ikisi de, her bir T4 durumundaki yükselen saat kenarı ile örneklenir. Maskelenebilir ya da maskelenemez, herhangi bir kesme alındıysa ve kesme yetkilendirme flip-flobu kurulmuş durumda ise, o zaman bir sonraki yükselen saat kenarında, durdurma durumundan çıkılır. Bu durumda, hemen sonra girilecek olan saykıl, alınan kesmenin türüne karşılık gelen bir kesme alındı saykılı olacaktır. Kesme girişlerinin her ikisinden de aynı anda kesme gelirse, o zaman, en yüksek önceliğe sahip olması nedeniyle, maskelenemez olan kesme alınır. Durdurma durumundayken NOP komutlarının yürütülmesinin amacı, bellek tazeleme sinyallerini etkin durumda tutmaktır. Durdurma durumundaki her saykıl, bellekten alınan veri dikkate alınmayarak MİB'e içeriden bir NOP komutu uygulanması dışında, normal bir M1 (getirme) saykılıdır. Bu süre boyunca, durdurma alındı sinyali, işlemcinin durdurma durumunda olduğunu belirtmek için etkin durumdadır.

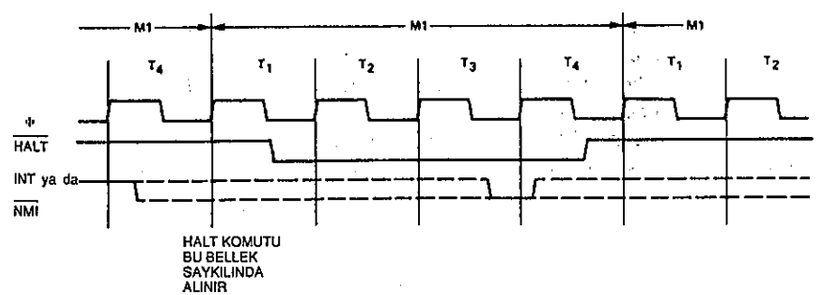
ASKELENEMEZ KESME İSTEK İŞLEMİ



* M2 ve M3, yığına yazma işlemleridir.

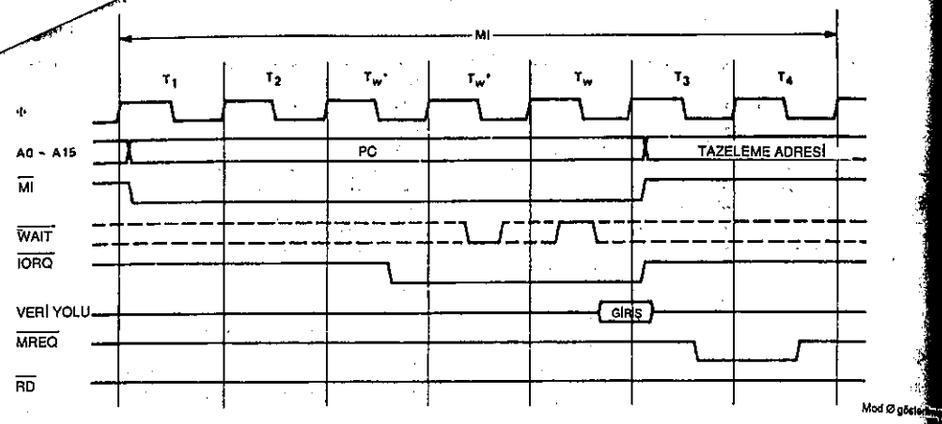
ŞEKİL 4.0-6

DURDURMA DURUMUNDAN ÇIKIŞ



ŞEKİL 4.0-7

RI İLE BİRLİKTE KESME İSTEĞİ/ALINDISI SAYKILI

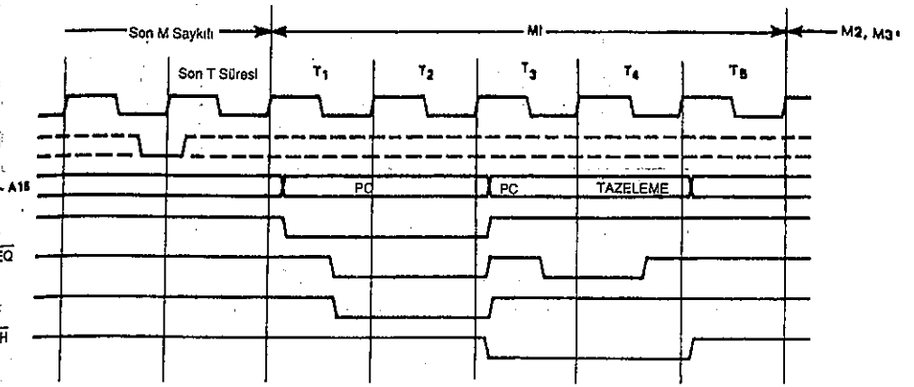


ŞEKİL 4.0-5A

DURDURMA DURUMUNDAN ÇIKIŞ

Yazılımsal bir durdurma komutu yürütüldüğünde, MİB, (kesme flip-flobu yetkilenmiş durumdayden maskelenemez ya da maskelenebilir olabilen) herhangi bir kesme alınıncaya kadar, sürekli olar NOP komutları yürütmeye başlar. Şekil 4.0-7'de gösterildiği gibi, kesme girişlerinin her iklsi de her bir T4 durumundaki yükselen saat kenarı ile örneklenir. Maskelenebilir ya da maskelenem herhangi bir kesme alındıysa ve kesme yetkilendirme flip-flobu kurulmuş durumda ise, o zam bir sonraki yükselen saat kenarında, durdurma durumundan çıkılır. Bu durumda, hemen son girilecek olan saykıl, alınan kesmenin türüne karşılık gelen bir kesme alındı saykılı olacaktır. Kesm girişlerinin her ikisinden de aynı anda kesme gelirse, o zaman, en yüksek önceliğe sahip olm nedeniyle, maskelenemez olan kesme alınır. Durdurma durumundayken NOP komutlarının yü tülmesinin amacı, bellek tazeleme sinyallerini etkin durumda tutmaktır. Durdurma durumunda saykıl, bellekten alınan veri dikkate alınmayarak MİB'e içeriden bir NOP komutu uygulanm dışında, normal bir M1 (getirme) saykılıdır. Bu süre boyunca, durdurma alındı sinyali, işlemci durdurma durumunda olduğunu belirtmek için etkin durumdadır.

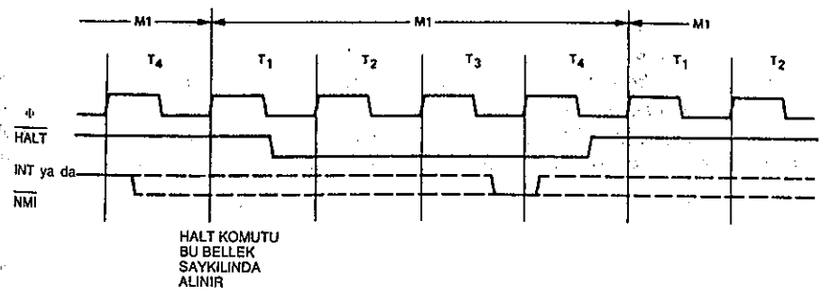
MASKELENEMEZ KESME İSTEK İŞLEMİ



* M2 ve M3, yığına yazma işlemleridir.

ŞEKİL 4.0-6

DURDURMA DURUMUNDAN ÇIKIŞ



ŞEKİL 4.0-7

6.0 BAYRAKLAR

Z80-MİB Bayrak kaydedicilerinin her ikisi de, altı bitlik bilgi içerir. Bu bilgilerin bitleri, çeşitli işlemleri sonucunda kurulabilir ya da sıfırlanabilir. Bu bitlerden dördünü test etmek mümkün; yani, bu test edilebilen bitler, atlama, çağırma, ya da geriye dönme komutları için birer koşul olarak kullanılırlar. Örneğin herhangi bir atlama komutunun, sadece bayrak kaydedicisindeki belirli bit kurulu ise yürütülmesi istenebilir. Test edilebilen dört adet bayrak biti şunlardır:

1) Elde Bayrağı (C) - Bu bayrak, akümülatörün en yüksek sıralı bitinden çıkan elde. Örneğin, akümülatörün en yüksek sıralı bitinden çıkan bir eldenin üretildiği bir toplama komutu sırasında, elde biti kurulur. Bu bayrak ayrıca, herhangi bir çıkarma komutu sırasında borç üretilmesi halinde de kurulur. Kaydırma ve döndürme komutları da bu biti etkiler.

2) Sıfır Bayrağı (Z) - Bu bayrak, yapılan işlemin sonucunda akümülatöre sıfır yüklenmesi halinde kurulur. Aksi halde sıfırlanır.

3) İşaret Bayrağı (S) - Bu bayrak, işaretli sayılarla kullanılmak üzere tasarlanmıştır. Bu yapılan işlemin sonucu negatif olduğunda kurulur. Bu bayrak, sayının işaretini 7'nci bit (EDB) gösterdiğinden dolayı (Negatif bir sayının 7'nci biti 1'dir), akümülatördeki 7'nci bitin durumunu saklar.

4) Eşlik/Taşma Bayrağı (P/V) - Bu çift amaçlı bayrak, mantıksal işlemler (örneğin AND, OR, B gibi) yapıldığında akümülatördeki sonucun eşliğini belirtir, ikiye tümleyeni ile ilgili aritmetik işlemler yerine getirildiğinde ise taşmayı gösterir. Z80 taşma bayrağı, akümülatördeki ikiye tümleyen halindeki sayının, ikiye tümleyeni notasyonunda gösterebilen maksimum sayıyı (+127) aşması ya da minimum sayıdan (-128) daha az olması halinde, hatalı olduğunu belirtir. Örneğin aşağıdaki toplamayı göz önüne alalım:

```
+120  =      0111 1000
+105  =      0110 1001
  C   = 0    1110 0001   = - 95 (yanlış) Taşma ortaya çıktı
```

Bu sonuç doğru değildir. Taşma ortaya çıkmıştır, ancak hata olduğunu belirten bir elde mevcut değildir. Bu durumda taşma bayrağı kurulur. Şimdi de, iki negatif sayının toplandığı, aşağıdaki toplamayı göz önüne alalım:

```
-5    =      1111 1011
-16   =      1111 0000
  C   = 1    1110 0001   = -21 doğru
```

Sonuç doğru olduğu halde, elde bayrağının kurulmuş olduğuna dikkat edilmelidir, bu yüzden elde bayrağı bir taşma tanıtıcısı olarak kullanılamaz. Böyle bir durumda taşma bayrağı kurulmaz.

Bu bayrak mantıksal işlemler (VE, VEYA, ÖZEL VEYA) için, sonucun eşliği çift ise kurulur, tek ise sıfırlanır.

Bayrak kaydedicisinde ayrıca iki adet de test edilemeyen bit mevcuttur. Bu bitlerin her ikisi de aritmetik için kullanılır. Bu bitler şunlardır:

1) Yarım elde (H) - Bu bit, işlemin en küçük değerlikli dört bitinden elde edilen BCD ya da ödünçtür. Bu bayrak, DAA (Ondalık Ayar Komutu) kullanılırken, önceden lenmiş ondalık toplama ya da çıkarmadan elde edilen sonucu düzeltmek için

2) Toplama/Çıkarma Bayrağı (N) - Bu bayrak, BCD toplama ve çıkar ma işlemlerini düzeltmek için kullanılan algoritmalar farklı olduklarından dolayı, son olarak yürütülen komutun türünü belirtmek için kullanılır. Böylece DAA komutunun toplama ya da çıkarma için düzeltme yapması sağlanmış olur.

Bayrak kaydedicisine programcı tarafından erişilmesi mümkündür ve formatı aşağıdaki gibidir:

D7							Dø
S	Z	X	H	X	P/V	N	C

X harfi, bayrağın belirsiz olduğunu gösterir.

Tablo 6.0-1'de, her bayrağın çeşitli MİB komutları tarafından nasıl etkilendikleri listelenmiştir. Bu tablodaki, '.' işareti, komutun bayrağı değiştirmediğini belirtir, 'X' harfi bayrağın belirsiz bir duruma geçtiğini ifade eder, '0' sayısı bayrağın sıfırlandığına gösterir, '1' sayısı bayrağın kurulduğunu gösterir ve ‡ sembolü bayrağın, yapılan işlemin sonucuna göre kurulduğunu ya da sıfırlandığını belirtir. Bu tabloda görünmeyen komutların, hiç bir bayrağı etkilemeyeceğine dikkat ediniz.

Tablo 6.0-1'de, açıklık için tanımlanması gereken bir kaç adet özel durum mevcuttur. Blok arama komutunun, yapılan son karşılaştırma işlemi, kaynak ile akümülatör verisi arasında uygunluk olduğunu belirttiği taktirde Z bayrağını 1'e kuracağına dikkat ediniz. Bu durumda ayrıca, bayt sayıcısı (BC kaydedici çifti) sıfıra eşit olmadığı taktirde eşlik bayrağı da kurulur. Eşlik bayrağının bu kullanılış şeklinin aynısı, blok taşıma komutları için de geçerlidir. Diğer bir özel durum, giriş ve çıkış komutları sırasında ortaya çıkar, bu durumda Z bayrağı, bayt sayıcısı olarak kullanılan B kaydedicisinin durumunu belirtmek için kullanılır. G/Ç blok aktarımı tamamlandığı zaman sıfır bayrağının sıfırlanacağına (yani B=0), herhangi bir blok taşıma komutu durumunda ise eşlik bayrağının işlem tamamlandığında sıfırlanacağına dikkat ediniz. Son bir özel durum, tazeleme ya da İndeks kaydedicisi akümülatöre yüklendiği anda ortaya çıkar, bu anda kesme yetkilendirme flip-flobu eşlik bayrağına yüklenir, böylece, herhangi bir anda MİB'nin tüm durumunu saklamak mümkün olur.

7.0 İŞLEM KODLARININ VE YÜRÜTME SÜRELERİNİN ÖZETİ

Aşağıdaki bölümde, Z80 komut takımı için bir özet verilmiştir. Komutlar, Tablo 7.0-1'den 7.0-11'e kadar gösterildiği gibi, mantıksal gruplar halinde düzenlenmiştir. Her tablo, assembly dili anımsatıcı Opkodunu, fiili Opkodunu, sembolik işlemi, her komutun yürütümünün ardından bayrak kaydedicisinin içeriğini, her komutun getirilmesi ve yürütülmesi için gerekli olan bellek saykıllarının sayısı ve T durumlarının (dış saat peryotları) toplam sayısı ile birlikte, her komut için gerekli olan bayt sayısını gösterir. Her tablonun, metin ya da diğer tablolara başvurulmasına gerek kalmayacak şekilde, kendi başına anlaşılabilir olmasına dikkat edilmiştir.

BAYRAK İŞLEMLERİNİN ÖZETİ

Komut	D7 S	Z		H		P/V	N	C D0	Açıklamalar
ADD A,s; ADC A,s	↕	↕	X	↕	X	V	0	↕	8-bitlik toplama veya eldeli toplama
SUB,s; SBCA,s; CP,s; NEG	↕	↕	X	↕	X	V	1	↕	8-bitlik çıkarma, eldeli çıkarrma, akümülatörle karşılaştırma ve ek
AND s	↕	↕	X	1	X	P	0	0	Mantıksal işlemler
OR s; XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	•	8-bitlik artırma
DEC s	↕	↕	X	↕	X	V	1	•	8-bitlik azalıma
ADD DD, SS	•	•	X	X	X	•	0	↕	16-bitlik toplama
ADC HL, SS	↕	↕	X	X	X	V	0	↕	16-bitlik eldeli toplama
SBC HL, SS	↕	↕	X	X	X	V	1	↕	16-bitlik eldeli çıkarma
RLA; RLCA; RRA; RRCA	•	•	X	0	X	•	0	↕	Akümülatörü döndürme
RL s; RLC s; RR s; RRC s; SLA s; SRA s; SRL s	↕	↕	X	0	X	P	0	↕	Konumları döndürme ve kaydırma
RLD; RRD	↕	↕	X	0	X	P	0	•	Basamağı sağa ve sola kaydırma
DAA	↕	↕	X	↕	X	P	•	↕	Akümülatörü ondalık ayarlama
CPL	•	•	X	1	X	•	1	•	Akümülatörü tümleme
SCF	•	•	X	0	X	•	0	1	Eldeyi kurma
CCF	•	•	X	X	X	•	0	↕	Eldeyi tümleme
IN r, (C)	↕	↕	X	0	X	P	0	•	Dolaylı kaydedici girişi
INI; IND; OUTI; OUTD	X	↕	X	X	X	X	1	•	Blok giriş ve çıkışı
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1	•	*Eğer B ≠ 0 ise Z = 0, değilse Z = 1
LDI; LDD	X	X	X	0	X	↕	0	•	Blok aktarım komutları
LDIR; LDDR	X	X	X	0	X	0	0	•	Eğer B ≠ 0 ise P/V = 1, değilse P/V = 0
CPI; CPIR; CPD; CPDR	X	↕	X	X	X	↕	1	•	Blok arama komutları Eğer A = (HL) ise Z = 1, değilse Z = 0 Eğer B ≠ 0 ise P/V = 1, değilse P/V = 0
LD A, I; LD A, R	↕	↕	X	0	X	IFF	0	•	Kesme yetkilendirme flip-flobu (IFF) içeriği P/V bayrağına kopyalanır
BIT b, s	X	↕	X	1	X	X	0	•	s konumunun b bitinin durumu Z bayrağına kopyalanır.

Bu tabloda aşağıdaki notasyon kullanılmıştır:

SEMBOL	İŞLEM
C	Elde/bağlantı bayrağı. Yapılan işlem, işlenen ya da sonucun EDB'sinden çıkan bir elde üretmişse C=1 olur.
Z	Sıfır bayrağı. İşlemin sonucu sıfırsa Z=1 olur.
S	İşaret bayrağı. Sonucun EBDB'si birse S=1 olur.
P/V	Eşlik ya da taşma bayrağı. Eşlik (P) ve taşma (V), aynı bayrağı ortaklaşa kullanırlar. Mantıksal işlemler bu bayrağı sonucun eşliği ile etkilerler, aritmetik işlemler ise bu bayrağı sonucun taşması ile etkilerler. Eşliği P/V bayrağı tutuyorsa, işlemin sonucu çift olduğunda P/V=1 olur, tek olduğunda ise P/V=0 olur. P/V bayrağı taşmayı tutuyorsa, işlemin sonucunda taşma ortaya çıktığı taktirde P/V=1 olur.
H	Yarım-elde bayrağı. Herhangi bir toplama işlemi sonucunda, akümülatörün 4'ncü bitine eklenen bir elde üretilmişse ya da herhangi bir çıkartma işlemi sonucunda, akümülatörün 4'ncü bitinden çıkarılan bir ödünç üretilmişse, H=1 olur.
N	Toplama/çıkarma bayrağı. Yapılan son işlem bir çıkarma işlemiyse, N=1 olur. H ve N bayrakları, BCD formatında paketlenmiş işlenenler kullanılan toplama ya da çıkartma işlemleri sonrasında BCD formatında paketlenmiş sonucu uygun bir şekilde düzeltmek için ondalık ayar komutu (DAA) ile bağlantılı olarak kullanılırlar.
↕	Bayrak, işlemin sonucuna göre etkilenir.
•	Bayrak, işlemden etkilenmez.
0	Bayrak, işlem tarafından sıfırlanır.
1	Bayrak, işlem tarafından kurulur.
X	Bayrak, "dikkate alınmaz".
V	P/V bayrağı, işlemin taşma sonucuna göre etkilenir.
P	P/V bayrağı, işlemin eşlik sonucuna göre etkilenir.
r	A, B, C, D, E, H, L MİB kaydedicilerinden herhangi biri.
s	Bu komutla kullanılabilecek olan bütün adresleme modları için, 8bitlik herhangi bir bellek konumu.
ss	Bu komutla kullanılabilecek olan bütün adresleme modları için, 16-bitlik herhangi bir bellek konumu.
II	IX ve IY indeks kaydedicilerinden herhangi biri.
R	Tazeleme sayıcısı.
n	<0,255> aralığındaki 8-bitlik bir sayı.
nn	<0,65535> aralığındaki 16-bitlik bir sayı.

TABLO 6.0-1

8-BİTLİK YÜKLEME GRUBU

Komut Kısaltması	Sembolik İşlem	Bayraklar S	Z		H		P/V	N	C	İş-kodu 76 543 210	Önalıtılı	Bayt Sayısı	M Saykıllarını Sayısı	T Durumlarını Sayısı	*Açıklamalar	
LD r, s	r ← s	•	•	X	•	X	•	•	•	01 r s		1	1	4	r, s	Kayd.
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110 ← n →		2	2	7	000	B
															001	C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010	D
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101 01 r 110 ← d →	DD	3	5	19	011	E
															100	H
															101	L
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101 01 r 110 ← d →	FD	3	5	19	111	A
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7		
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101 01 110 r ← d →	DD	3	5	19		
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101 01 110 r ← d →	FD	3	5	19		
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110 ← n →	36	2	3	10		
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101 00 110 110 ← d → ← n →	DD 36	4	5	19		
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101 00 110 110 ← d → ← n →	FD 36	4	5	19		
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7		
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7		
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010 ← n → ← n →	3A	3	4	13		
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7		
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7		
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010 ← n → ← n →	32	3	4	13		
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101 01 010 111	ED 57	2	2	9		
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101 01 011 111	ED 5F	2	2	9		
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101 01 000 111	ED 47	2	2	9		
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101 01 001 111	ED 4F	2	2	9		

Notlar: r,s; A, B, C, D, E, H, L kaydedicilerinden herhangi birini gösterir.
Kesme yetkileme flip-flobununun IFF, P/V bayrağına kopyalanır.

Bayrak Sembolleri: • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
↕ = bayrak, işlemin sonucuna bağlı olarak etkilenir

16- BİTLİK YÜKLEME GRUBU

Kom.Kısaltması	Sembolik İşlem	S	Z		H		P/V	N	C	İş-Kodu 76 543 210	Onaltılı	Bayt sayısı	M Saykıllarının Sayısı	T Durumlarının Sayısı	Açıklamalar
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001 ← n → ← n →		3	3	10	dd CPU 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11 011 101 00 100 001 ← n → ← n →	DD 21	4	4	14	
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11 111 101 00 100 001 ← n → ← n →	FD 21	4	4	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	•	•	X	•	X	•	•	•	00 101 010 ← n → ← n →	2A	3	5	16	
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	X	•	X	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	ED	4	6	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	X	•	X	•	•	•	11 011 101 00 101 010 ← n → ← n →	DD 2A	4	6	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	X	•	X	•	•	•	11 111 101 00 101 010 ← n → ← n →	FD 2A	4	6	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	•	•	X	•	X	•	•	•	00 100 010 ← n → ← n →	22	3	5	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	X	•	X	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	ED	4	6	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	X	•	X	•	•	•	11 011 101 00 100 010 ← n → ← n →	DD 22	4	6	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	X	•	X	•	•	•	11 111 101 00 100 010 ← n → ← n →	FD 22	4	6	20	
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP-2) ← qq_L (SP-1) ← qq_H	•	•	X	•	X	•	•	•	11 qq0 101		1	3	11	qq CPU 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	•	•	X	•	X	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	•	•	X	•	X	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP)	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	

Notlar: dd;BC,DE,HL,SP kaydedici çiftlerinden birini gösterir.
qq;AF,BC,DE,HL kaydedici çiftlerinden birini gösterir.
$(ÇİFT)_H$, $(ÇİFT)_L$; kaydedici çiftlerinin, sırasıyla üst-sıralı ve alt -sıralı 8 bitini gösterir.Örn.BC_L = C , AF_H = A

Bayrak Sembolleri: • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
↕= bayrak, işlemin sonucuna bağlı olarak etkilenir

DEĞİŞ - TOKUŞ GRUBU İLE BLOK AKTARMA VE ARAMA GRUBU

Kom.Kısaltması	Sembolik İşlem	S	Z		H		P/V	N	C	İş-Kodu 76 543 210	Onaltılı	Bayt sayısı	M Saykıllarının Sayısı	T Durumlarının Sayısı	Açıklamalar
EX DE, HL	DE↔HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF↔AF'	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	(BC↔BC' DE↔DE' HL↔HL')	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Kaydedici grubu ile yedek kaydedici grubu arasında yer değiştirme
EX (SP), HL	H ↔(SP+1) L ↔(SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	IX_H ↔(SP+1) IX_L ↔(SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	IY_H ↔(SP+1) IY_L ↔(SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE)←(HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	•	•	X	0	X	① ↕	0	•	11 101 101 10 100 000	ED A0	2	4	16	(HL)'i (DE)' yükle, işaretçileri artır, ve bayt sayıcı (BC) azalt
LDIR	(DE)←(HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 BC = 0 olana dek tekrarla	•	•	X	0	X	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	Eğer BC ≠ 0 ise Eğer BC = 0 ise
LDD	(DE)←(HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	•	•	X	0	X	① ↕	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE)←(HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 BC = 0 olana dek tekrarla	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	Eğer BC ≠ 0 ise Eğer BC = 0 ise
CPI	A – (HL) HL ← HL+1 BC ← BC-1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A – (HL) HL ← HL+1 BC ← BC-1 A = (HL) ya da BC= 0 olana dek tekrarla	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	Eğer BC ≠ 0 ve A ≠ (HL) ise Eğer BC = 0 ve ya A = (HL) ise
CPD	A – (HL) HL ← HL-1 BC ← BC-1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A – (HL) HL ← HL-1 BC ← BC-1 A = (HL) ya da BC= 0 olana dek tekrarla	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	Eğer BC ≠ 0 ve A ≠ (HL) ise Eğer BC = 0 ve ya A = (HL) ise

Notlar: ① Eğer BC-1 = 0 ise, P/V bayrağı sıfırlanır, aksi halde P/V = 1'dır.
② Eğer A-=(HL) ise, Z bayrağı kurulur, aksi halde Z = 0'dır.

Bayrak Sembolleri: • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
↕ = bayrak, işlemin sonucuna bağlı olarak etkilenir

8- BİTLİK ARİTMETİK VE MANTIKSAL GRUP

Kom.Kısaltması	Sembolik İşlem	Bayraklar S	Z		H		P/V	N	C	İş-Kodu 76 543 210	Hex	Bayt sayısı	M Saykıllarının Sayısı	T Durumlarının Sayısı	Açıklamalar
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 [000] r		1	1	4	r Kayd.
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 [000] 110 ← n →		2	2	7	000 B 001 C 010 D
ADD A, (HL)	A ← A+(HL)	↕	↕	X	↕	X	V	0	↕	10 [000] 110		1	2	7	011 E
ADD A, (IX+d)	A ← A+(IX+d)	↕	↕	X	↕	X	V	0	↕	11 011 101 10 [000] 110 ← d →	DD	3	5	19	100 H 101 L 111 A
ADD A, (IY+d)	A ← A+(IY+d)	↕	↕	X	↕	X	V	0	↕	11 111 101 10 [000] 110 ← d →	FD	3	5	19	
ADC A, s	A ← A+s+CY	↕	↕	X	↕	X	V	0	↕	[001]					s; ADD komutunda görülen gibi, r, n, (HL), (IX + d), (IY+d)'den herhangibirini gösterir. Gösterilen bitler yukarıdaki ADD gurubundaki [000] ile yer değişir.
SUB s	A ← A - s	↕	↕	X	↕	X	V	1	↕	[010]					
SBC A, s	A ← A - s - CY	↕	↕	X	↕	X	V	1	↕	[011]					
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	[100]					
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0	[110]					
XOR s	A ← A ⊕ s	↕	↕	X	0	X	P	0	0	[101]					
CP s	A - s	↕	↕	X	↕	X	V	1	↕	[111]					
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r [100]		1	1	4	
INC (HL)	(HL)←(HL)+1	↕	↕	X	↕	X	V	0	•	00 110 [100]		1	3	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	↕	↕	X	↕	X	V	0	•	11 011 101 00 110 [100] ← d →	DD	3	6	23	
INC (IY+d)	(IY+d) ← (IY+d)+1	↕	↕	X	↕	X	V	0	•	11 111 101 00 110 [100] ← d →	FD	3	6	23	
DEC s	s ← s - 1	↕	↕	X	↕	X	V	1	•	[101]					s; INC komutunda görülen gibi, r, n, (HL), (IX + d), (IY+d)'den herhangi birini gösterir. DEC, aynı formattadır ve INC gibi kodlanır. İş kodundaki [101], [100] ile yer değişir.

Notlar: P/V bayrak sütunundakiV sembolü, P/V bayrağının, işlem sonucunda üretlen taşmayı içerdiğini gösterir. Benzer şekilde, P de eşliği gösterir. V = 1 ise taşma vardır, V = 0 ise taşma yoktur. P = 1 ise elde edilen sonucun eşliği çift, P = 0 ise tektir.

Bayrak Sembolleri: • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
↕= bayrak, İşlemin sonucuna bağlı olarak etkilenir

GENEL AMAÇLI ARİTMETİK VE MİB DENETİM GRUPLARI

Kom.Kısaltması	Sembolik İşlem	Bayraklar S	Z		H		P/V	N	C	İş-Kodu 76 543 210		Bayt sayısı	M Saykıllarır Sayısı	T Durumların Sayısı	Açıklamalar
DAA	İşlenenleri paketlenmiş BCD formalındaki toplama veya çıkarma işlemi sonrasında, ACC'nin içeriğini paketlenmiş BCD formalına dönüştürür.	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Ondalık Ayar Akümülatörü
CPL	A ← A	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Akümülatörü tümle (Bir'e tümleme)
NEG	A ← $\overline{A}$ + 1	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Akümülatörü eksile (İki'e tümleme)
CCF	CY ← $\overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Elde bayrağını tümle
NOP	Hiç işlem yapma	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	Elde bayrağını kur
HALT	MİB durdurulur	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI*	IFF → 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI*	IFF → 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Kesme Modu 0'a kurulur	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Kesme Modu 1'e kurulur	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Kesme Modu 2'yekurulur	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

Notlar: IFF sembolü, kesme yetkileme flip-flobunu gösterir.
CY sembolü, elde flip-flobunu gösterir.

Bayrak Sembolleri: • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
↕= bayrak, İşlemin sonucuna bağlı olarak etkilenir

* Kesme girişleri, EI ve DI komutlarının sonunda örneklenmez.

16- BİTLİK ARİTMETİK GRUP

Kom.Kısaltması	Sembolik İşlem	Bayraklar S	Z		H		P/V	N	C	İş-Kodu 76 543 210		Bayt sayısı	M Saykl. Sayısı	T Durumlar Sayısı	Açıklamalar
ADD HL, ss	HL ← HL+ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss Kayd. 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL+ss+CY	↕	↕	X	X	X	V	0	↕	11 101 101 01 ss1 010	ED	2	4	15	
SBC HL, ss	HL ← HL-ss-CY	↕	↕	X	X	X	V	1	↕	11 101 101 01 ss0 010	ED	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 00 pp1 001	DD	2	4	15	pp Kayd. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 00 rr1 001	FD	2	4	15	rr Kayd. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 00 100 011	DD 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 00 100 011	FD 23	2	2	10	
DEC ss	ss ← ss - 1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX ← IX - 1	•	•	X	•	X	•	•	•	11 011 101 00 101 011	DD 2B	2	2	10	
DEC IY	IY ← IY - 1	•	•	X	•	X	•	•	•	11 111 101 00 101 011	FD 2B	2	2	10	

Notlar: ss;BC,DE,HL,SP kaydedici çiftlerinden birini gösterir.
pp;BC,DE,IX, SP kaydedici çiftlerinden birini gösterir.
rr,BC,DE,IY, SP kaydedici çiftlerinden birini gösterir.

Bayrak Sembolleri: • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
↕ = bayrak, İşlemin sonucuna bağlı olarak etkilenir

DÖNDÜRME VE KAYDIRMA GRUBU

Kom. Kısaltması	Sembolik İşlem	Bayraklar S	Z		H		P/V	N	C	İş-kodu 76 543 210	Onaltılı	Bayt sayısı	M Saykl. Sayısı	T Durum. Sayısı	Açıklamalar
RLCA	CY ← 7 ← 0, A	•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Akümülatörü, dairesel olarak sola döndür
RLA	CY ← 7 ← 0, A	•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Akümülatörü, sola döndür
RRCA	7 → 0 → CY, A	•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Akümülatörü, dairesel olarak sağa döndür
RRA	7 → 0 → CY, A	•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Akümülatörü sağa döndür
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 r	CB	2	2	8	r kaydedicisini, dairesel olarak sola döndür
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 110	CB	2	4	15	r Kayd. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX+d)	CY ← 7 ← 0 r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 ← d → 00 000 110	DD CB	4	6	23	
RLC (IY+d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 ← d → 00 000 110	FD CB	4	6	23	
RL s	CY ← 7 ← 0 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	010					Komut formatları ve durum sayın, RLC komutları için verilenlerle aynıdır.
RRC s	7 → 0 → CY s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	001					Yeni İŞ-Kodunu oluşturmak için, RLC komutlarında 000 olarak gösterilen bitler, burada verilen kodlarla değiştirilir.
RR s	7 → 0 → CY s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	011					
SLA s	CY ← 7 ← 0 ← 0 s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	100					
SRA s	7 → 0 → CY s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	101					
SRL s	0 → 7 → 0 → CY s ≡ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	111					
RLD	A 7-4 3-0 7-4 3-0 (HL)	↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	
RRD	A 7-4 3-0 7-4 3-0 (HL)	↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	Akümülatör ile (HL) konumu arasında, sola ve sağa basamak döndürme. Akümülatörün üst yarısının içeriği etkilenmez.

Bayrak Sembolleri • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
↕ = bayrak, İşlemin sonucuna bağlı olarak etkilenir.

BİT KURMA, SIFIRLAMA VE TEST GRUBU

Kom. Kısaltması	Sembolik İşlem	Bayraklar S	Z		H		P/V	N	C	İş-kodu 76 543 210	Onaltılı	Bayt sayısı	M Saykıl. Sayısı	T Duruml. Sayısı	Açıklamalar
BIT b, r	$Z = \overline{r_b}$	X	‡	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8	
BIT b, (HL)	$Z = \overline{(HL)_b}$	X	‡	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX+d)	$Z = \overline{(IX+d)_b}$	X	‡	X	1	X	X	0	•	11 011 101 11 001 011 ← d → 01 b 110	DD CB	4	5	20	
BIT b, (IY+d)	$Z = \overline{(IY+d)_b}$	X	‡	X	1	X	X	0	•	11 111 101 11 001 011 ← d → 01 b 110	FD CB	4	5	20	
SET b, r	$r_b = 1$	•	•	X	•	X	•	•	•	11 001 011 11 b r	CB	2	2	8	
SET b, (HL)	$(HL)_b = 1$	•	•	X	•	X	•	•	•	11 001 011 11 b 110	CB	2	4	15	
SET b, (IX+d)	$(IX+d)_b = 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 ← d → 11 b 110	DD CB	4	6	23	
SET b, (IY+d)	$(IY+d)_b = 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 ← d → 11 b 110	FD CB	4	6	23	
RES b, s	$s_b = 0$ s ≡ r, (HL), (IX+d), (IY+d)	•	•	X	•	X	•	•	•	10					

r	Kayd
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	Test edilen bit
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Yeni İş-Kodunu oluşturmak için SET b,s komutlarında 11 olan gösterdiğimiz olan bitler 10 ile değiştirilir. Bayrakların durum sayıları, SET komutları için verilenlerle aynıdır.

Notlar: sb sembolü, bit b'yi (bit 0'dan bit 7'ye) ya da s gözünü gösterir.

Bayrak Sembolleri • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
‡ = bayrak, işlemin sonucuna bağlı olarak etkilenir.

ATLAMA GRUBU

Kom. Kısaltması	Sembolik İşlem	Bayraklar S	Z		H		P/V	N	C	İş-kodu 76 543 210	Onaltılı	Bayt sayısı	M Saykıl. Sayısı	T Duruml. Sayısı	Açıklamalar
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 ← n → ← n →	C3	3	3	10	
JP cc, nn	cc koşulu doğruysa PC ← nn, aksi halde devam.	•	•	X	•	X	•	•	•	11 cc 010 ← n → ← n →		3	3	10	
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 ← e-2 →	18	2	3	12	
JR C, e	C = 0 ise, devam C = 1 ise, PC ← PC+e	•	•	X	•	X	•	•	•	00 111 000 ← e-2 →	38	2 2	2 3	7 12	Koşul sağlanmıyorsa Koşul sağlanıyorsa
JR NC, e	C = 1 ise, devam C = 0 ise, PC ← PC+e	•	•	X	•	X	•	•	•	00 110 000 ← e-2 →	30	2 2	2 3	7 12	Koşul sağlanmıyorsa Koşul sağlanıyorsa
JR Z, e	Z = 0 ise, devam.... Z = 1 ise, PC ← PC+e	•	•	X	•	X	•	•	•	00 101 000 ← e-2 →	28	2 2	2 3	7 12	Koşul sağlanmıyorsa Koşul sağlanıyorsa
JR NZ, e	Z = 1 ise, devam Z = 0 ise, PC ← PC+e	•	•	X	•	X	•	•	•	00 100 000 ← e-2 →	20	2 2	2 3	7 12	Koşul sağlanmıyorsa Koşul sağlanıyorsa
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B - 1 B = 0 ise, devam	•	•	X	•	X	•	•	•	00 010 000 ← e-2 →	10	2	2	8	B = 0 İçin
	B <>0 ise, PC ← PC+e											2	3	13	B ≠ 0 İçin

cc	Açıklamalar
000	NZ sıfır değil
001	Z sıfır
010	NC elde yok
011	C elde var
100	PO eşlik tek
101	PE eşlik çift
110	P İşaret pozitif
111	M İşaretnegatif

Notlar: e sembolü, bağıl adresleme modundaki genişlemeyi gösterir.
e, <-128, 129 > aralığında bulunan, ikinin tümleyeni halindeki işaretli bir sayıdır.
OP- kodundaki e-2, PC adres işaretçisi, e sayısı eklenmeden önce 2 artırıldığı için pc+e'ye eşit bir efektif adres sağlar.

Bayrak Sembolleri • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
‡ = bayrak, işlemin sonucuna bağlı olarak etkilenir.

ÇAĞIRMA VE GERİYE DÖNME GRUBU

Kom. Kısaltması	Sembolik İşlem	Bayraklar S	Z		H		P/V	N	C	İş-kodu 76 543 210	Onaltılı	Bayt sayısı	M Saykıl. Sayısı	T Duruml. Sayısı	Açıklamalar
CALL nn	(SP-1) ← PC_H (SP-2) ← PC_L PC ← nn	•	•	X	•	X	•	•	•	11 001 101 ← n → ← n →	CD	3	5	17	
CALL cc, nn	cc koşulu yanlışsa devam, aksi halde CAL nın'deki gibi	•	•	X	•	X	•	•	•	11 cc 100 ← n → ← n →		3 3	3 5	10 17	cc yanlış ise cc doğru ise
RET	PC_L ← (SP) PC_H ← (SP+1)	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
RET cc	cc koşulu yanlışsa devam, aksi halde RET'deki gibi RET	•	•	X	•	X	•	•	•	11 cc 000		1 1	1 3	5 11	cc yanlış ise cc doğru ise
RETI	Kesmeden geri dön	•	•	X	•	X	•	•	•	11 101 101 01 001 101	ED 4D	2	4	14	
RETN[1]	Maskelenmez kesmeden geri dön	•	•	X	•	X	•	•	•	11 101 101 01 000 101	ED 45	2	4	14	
RST p	(SP-1) ← PC_H (SP-2) ← PC_L PC_H ← 0 PC_L ← p	•	•	X	•	X	•	•	•	11 t 111		1	3	11	

cc	
000	NZ sıfır değil
001	Z sıfır
010	NC elde yok
011	C elde var
100	PO eşlik tek
101	PE eşlik çift
110	P işaret pozitif
111	M işaretnegatif

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

1 RETN komutu IFF2'yi IFF1'e yükler

Bayrak Sembolleri • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
↕ = bayrak, işlemin sonucuna bağlı olarak etkilenir.

GİRİŞ VE ÇIKIŞ GRUBU

Kom. Kısaltması	Sembolik İşlem	Bayraklar S	Z		H		P/V	N	C	İş-kodu 76 543 210	Onaltılı	Bayt sayısı	M Saykıl. Sayısı	T Duruml. Sayısı	Açıklamalar
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	2	3	11	n, AO-A7'ye Acc, A8-A15'e
IN r, (C)	r ← (C) eğer r=110 ise sadece bayraklar etkilenir	↕	↕	X	↕	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C, AO-A7'ye B, A8-A15'e
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 100 010	ED A2	2	4	16	C, AO-A7'ye B, A8-A15'e
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 B = 0 olana dek tekrarla	X	1	X	X	X	X	1	•	11 101 101 10 110 010	ED B2	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C, AO-A7'ye B, A8-A15'e
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 101 010	ED AA	2	4	16	C, AO-A7'ye B, A8-A15'e
INDR	(HL) ← (C) B = 0 olana dek tekrarla	X	1	X	X	X	X	1	•	11 101 101 10 111 010	ED BA	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C, AO-A7'ye B, A8-A15'e
OUT (n), A	B = 0 (n) ← A	•	•	X	•	X	•	•	•	11 010 011	D3	2	3	11	n, AO-A7'ye Acc, A8-A15'e
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C, AO-A7'ye B, A8-A15'e
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 100 011	ED A3	2	4	16	C, AO-A7'ye B, A8-A15'e
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 B = 0 olana dek tekrarla	X	1	X	X	X	X	1	•	11 101 101 10 110 011	ED B3	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 101 011	ED AB	2	4	16	C, AO-A7'ye B, A8-A15'e
OTDR	(C) ← (HL) B ← B - 1 HL ← HL - 1 B = 0 olana dek tekrarla	X	1	X	X	X	X	1	•	11 101 101 10 111 011	ED BB	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C, AO-A7'ye B, A8-A15'e

Notlar: ① Eğer B-1 işleminin sonucu sıfırsa Z bayrağı kurulur aksi halde sıfırlanır.

Bayrak Sembolleri • = bayrak etkilenmez, 0 = bayrak sıfırlanır, 1 = bayrak kurulur, X = bayrağın durumu bilinmez
↕ = bayrak, işlemin sonucuna bağlı olarak etkilenir.

Ek C Motorola 6800

MİKROİŞLEMCİ BİRİMİ (MİB)

MC6800, Motorola M6800 ailesinin merkezi denetim işlevini oluşturan 8-bitlik bir monolitik mikroişlemcidir. Bütün M6800 sistem parçalarında olduğu gibi TTL uyumlu olan MC6800, yalnızca tek bir +5.0 volt'luk güç kaynağı gerektirir ve yol arabirimi için dış TTL birimlerine gerek yoktur.

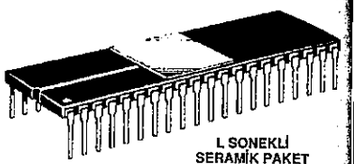
MC6800, 16-bitlik adres yoluyla 65 Kbaytlık belleği adresleyebilir. Doğrudan bellek adresleme ve çoklu-işlem uygulamalarının yapılabilmesini sağlayan 8-bitlik veri yolu, üçdurumlu ve iki yönlüdür.

- Sekiz-Bitlik Paralel İşleme
- İki Yönlü Veri Yolu
- 16-bitlik Adres Yolu - 65 K Bayt'lık Adresleme
- 72 Komut - Değişik Uzunlukta
- Yedi adresleme Modu - Doğrudan, Bağıl, Dolaysız, İndeksli, Genişletilmiş, Mutlak ve Akümülatör
- Değişebilir Uzunlukta Yığın
- Vektörlenmiş Yeniden Başlatma
- Maskelenebilir Kesme Vektörü
- Ayrı Maskelenemez Kesme - İç kaydediciler yığında saklanır.
- Altı İç Kaydedici - İki Akümülatör, İndeks Kaydedicisi, Program Sayıcısı, Yığın İşaretçisi ve Koşul Kodu Kaydedicisi
- Doğrudan Bellek Adresleme (DMA) ve Çoklu İşlem Yeteneği
- 1 MHz'e Kadar Saat Hızları
- TTL'siz Basit Yol Arabirimleri
- Durdurma ve Tek Komut Yürütme Yeteneği

MOS

(N-KANAL, SİLİKON-KAPI)

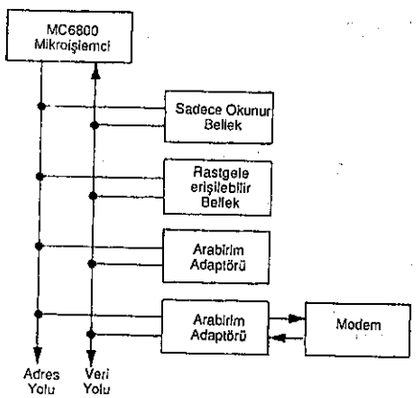
MİKROİŞLEMCİ



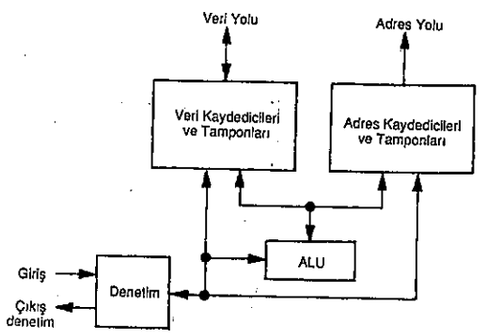
L SONEKLİ SERAMİK PAKET KILIF 715

GÖSTERİLMEYEN: P SONEKLİ PLASTİK PAKET KILIF 711

M6800 MİKROBİLGİSAYAR AİLESİ BLOK ŞEMASI



M6800 MİKROİŞLEMCİ BLOK ŞEMASI

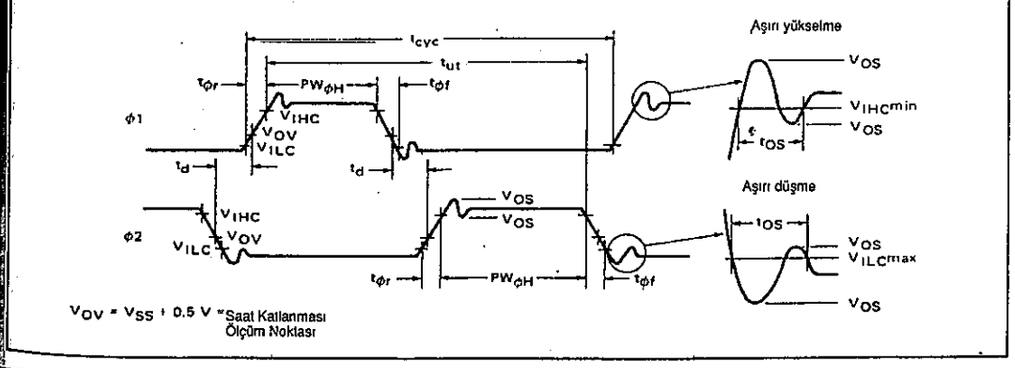


ELEKTRİKSEL KARAKTERİSTİKLER (Aksi belirtilmedikçe, V_{CC} = 5.0 V ± %5, V_{SS} = 0, T_A = 0 - 70 °C)

Karakteristik		Sembol	Min.	Tipik	Maks.	Birimi
Giriş Yüksek Gerilimi	Mantıksal φ1,φ2	V_{IH} V_{IHC}	V_{SS} + 2.0 V_{CC} – 0.3	– –	V_{CC} V_{CC} + 0.1	Vdc
Giriş Alçak Gerilimi	Mantıksal φ1,φ2	V_{IL} V_{ILC}	V_{SS} – 0.3 V_{SS} – 0.1	– –	V_{SS} + 0.8 V_{SS} + 0.3	Vdc
Saat Aşırı Yükselmesi / Aşırı Düşmesi - Giriş Yüksek Düzeyi - Giriş Alçak Düzeyi		V_{OS}	V_{CC} – 0.5 V_{SS} – 0.5	– –	V_{CC} + 0.5 V_{SS} + 0.5	Vdc
Giriş Kaçak Akımı (V_{gir} = 0 - 5.25 V, V_{CC} = maks) (V_{gir} = 0 - 5.25 V, V_{CC} = 0.0 V	Mantıksal* Mantıksal* φ1,φ2		– –	1.0 –	2.5 100	μAdc
Üç-Durumlu (KAPALI) Giriş Akımı (V_{gir} = 0.4-2.4 V, V_{CC} = maks)	D0-D7 A0-A15,R/W	I_{TSI}	– –	2.0 –	10 100	μAdc
Çıkış Yüksek Gerilimi ($I_{yük}$ = -205 μAdc, V_{CC} = min.) ($I_{yük}$ = -145 μAdc, V_{CC} = min.) ($I_{yük}$ = -100 μAdc, V_{CC} = min.)	D0-D7 A0-A15,R/W,VMA BA	V_{OH}	V_{SS} + 2.4 V_{SS} + 2.4 V_{SS} + 2.4	– – –	– – –	Vdc
Çıkış Yüksek Gerilimi ($I_{yük}$ = 1.6 m Adc, V_{CC} = min.)		V_{OL}	–	–	V_{SS} + 0.4	Vdc
Güç Harcaması		P_D	–	0.600	1.2	W
Kapasitans # (V_{gir} = 0, T_A = 25 °C, f = 1.0 MHz)	φ1,φ2 TSC DBE D0-D7 Mantıksal Girişler A0-A15,R/W,VMA	C_{gir} $C_{çık}$	80 – – – – –	120 – 7.0 10 6.5 –	160 15 10 12.5 8.5 12	pF pF
Çalışma Frekansı		f	0.1	–	1.0	MHz
Saat Zamanlaması (Şekil 1) Saykıl Süresi		t_{cyc}	1.0	–	10	μs n
Saat Darbe Genişliği (V_{CC} - 0.3 V'da Ölçülmüştür)	φ1 φ2	$PW_{\phi H}$	430 450	– –	4500 4500	ns n
		t_{ut}	940	–	–	ns n
Toplam Ø1 ve Ø2 Yükselme Süresi Yükselme ve Düşme Süreleri (V_{SS} + 0.3 V ve V_{CC} - 0.3 V'da Ölçülmüştür)	φ1,φ2	$t_{\phi r}$, $t_{\phi f}$	5.0	–	50	ns n
Gecikme Süresi veya Saat Ayrılması (V_{OV} = V_{SS} + 5 V'da Ölçülmüştür)		t_d	0	–	9100	ns n
Aşırı Yükselme Süresi		t_{OS}	0	–	40	ns n

* $\overline{IRQ}$ ve $\overline{NMI}$ hariç, optimum çalışmada, telli - VEYA için 3 k Ω'luk yukarı çekme yük dirençleri gerekir.
# Kapasiteler % 100 test çekme yük dirençleri gerekir.

ŞEKİL 1 - SAAT ZAMANLAMA DALGABİÇİMLERİ



MAKSİMUM ANMA DEĞERLERİ

Anma değerleri	Sembol	Değeri	Birimi
Güç kaynağı	V_{CC}	-0.3 - +7.0	Vdc
Giriş Gerilimi	V_{in}	-0.3 - +7.0	Vdc
Çalışma Sıcaklık Aralığı	T_A	0 - +70	°C
Saklama Sıcaklık Aralığı	T_{stg}	-55 - +150	°C
Sıcaklık Direnci	θJA	70	°C/W

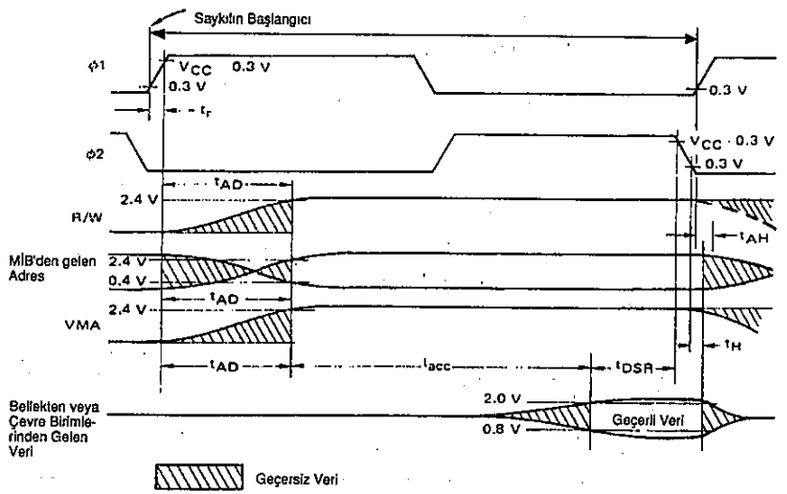
Bu birim, elektrik alanlarından veya statik gerilimlere karşı girişleri koruyucu bir devre içerir. Bununla beraber, yüksek empedanslı devreye maksimum ma gerilimlerinden daha yüksek ger uygulanmasını önlemek için normal ön rin alınması tavsiye edilir.

OKUMA/YAZMA ZAMANLAMASI Şekil 2 ve 3, f = 1,0 MHz, Şekil 6'daki Yük Devresi

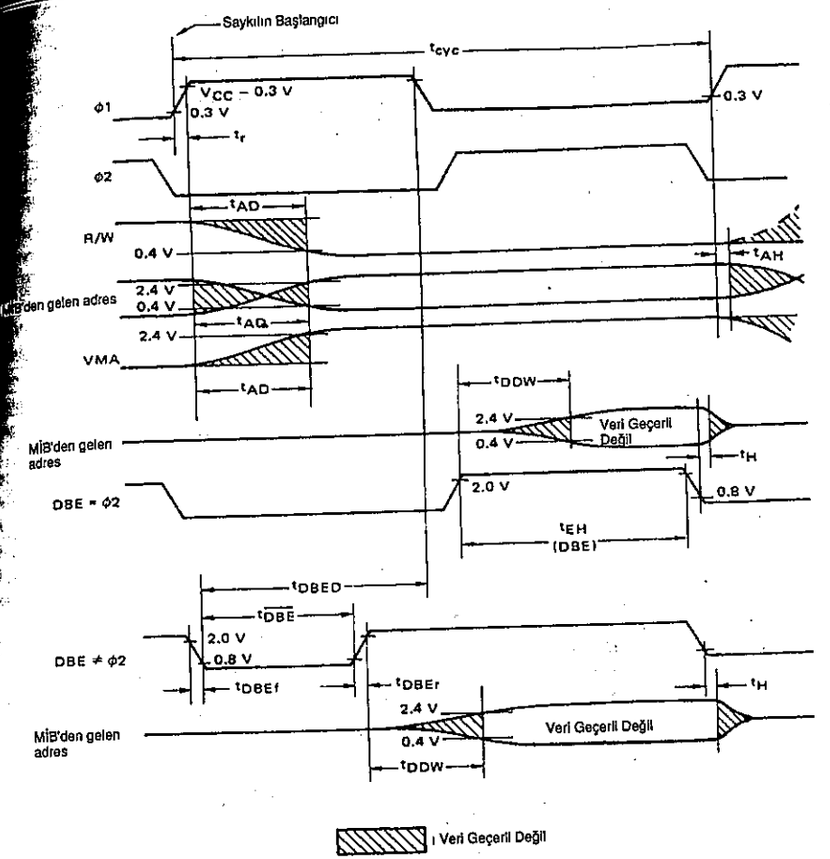
Karakteristikler	Sembol	Minimum	Tipik	Maksimum	Birim
Adres Gecikmesi	t_{AD}	-	220	300	nsn
Çevrebirimi Okuma Erişim Süresi $t_{acc} = t_{cyc} - (t_{AD} + t_{DSR})$	t_{acc}	-	-	540	nsn
Veri Kurma Süresi (okuma)	t_{DSR}	100	-	-	nsn
Giriş Verisi Tutma Süresi	t_H	10	-	-	nsn
Çıkış Verisi Tutma Süresi	t_H	10	25	-	nsn
Adres Tutma Süresi (Adres, R/W, VMA)	t_{AH}	50	75	-	nsn
DME Girişi için YetkilendirmeYüksek Düzey Süresi	t_{EH}	450	-	-	nsn
Veri Gecikme Süresi (Yazma)	t_{DDW}	-	165	225	nsn
İşlemci Denetimleri					
İşlemci Denetimi Kurma Süresi	t_{PCS}	200	-	-	nsn
İşlemci Denetimi Yükselme ve Düşme Süresi	t_{PCr}, t_{PCf}	-	-	100	nsn
Yok Kullanabilirliği Gecikmesi	t_{BA}	-	-	300	nsn
Üç Durumluluk Yetkilendirmesi	t_{TSE}	-	-	40	nsn
Üç Durumluluk Gecikmesi	t_{TSD}	-	-	700	nsn
φ1'in Yüksek Düzey Süresindeki Veri Yolu Yetkilendirmesinin Düşük Kalma Süresi (Şekil 3)	$t_{\overline{DBE}}$	150	-	-	nsn
Veri Yolu Yetkilendirme Gecikmesi (Şekil 3)	t_{DBED}	300	-	-	nsn
Veri Yolu Yetkilendirme Yükselme ve Düşme Süreleri (Şekil 3)	t_{DBEr}, t_{DBEf}	-	-	25	nsn

Ek Bilgiler, Şekil 12'den Şekil 16'ya kadar olan Aile Karakteristiklerinde verilmiştir - bkz. 17-20. sayfaları.

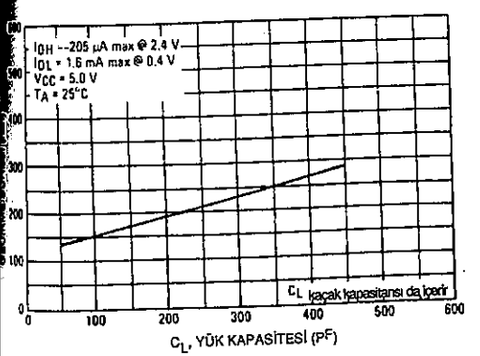
ŞEKİL 2 - BELLEKTEN VEYA ÇEVREBİRİMLERİNDEN VERİ OKUMA



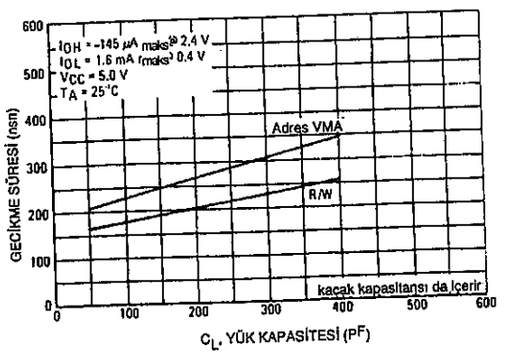
ŞEKİL 3 - BELLEĞE VEYA ÇEVREBİRİMLERİNE YAZMA



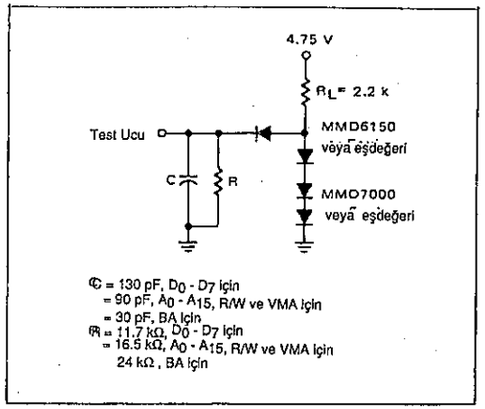
ŞEKİL 4 - KAPASİTİF YÜKTE TİPİK VERİ YOLU ÇIKIŞ GECİKMELERİ



ŞEKİL 5 - KAPASİTİF YÜKTE TİPİK OKUMA/YAZMA, VMA, VE ADRES ÇIKIŞ GECİKMELERİ

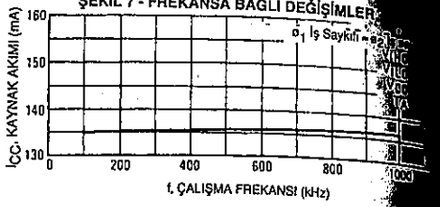


ŞEKİL 6 - YOL ZAMANLAMA TEST YÜKÜ

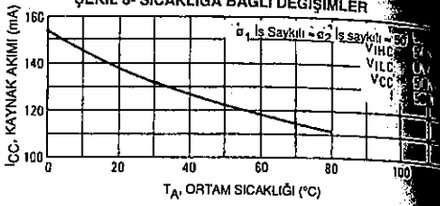


TİPİK GÜÇ KAYNAĞI AKIMI

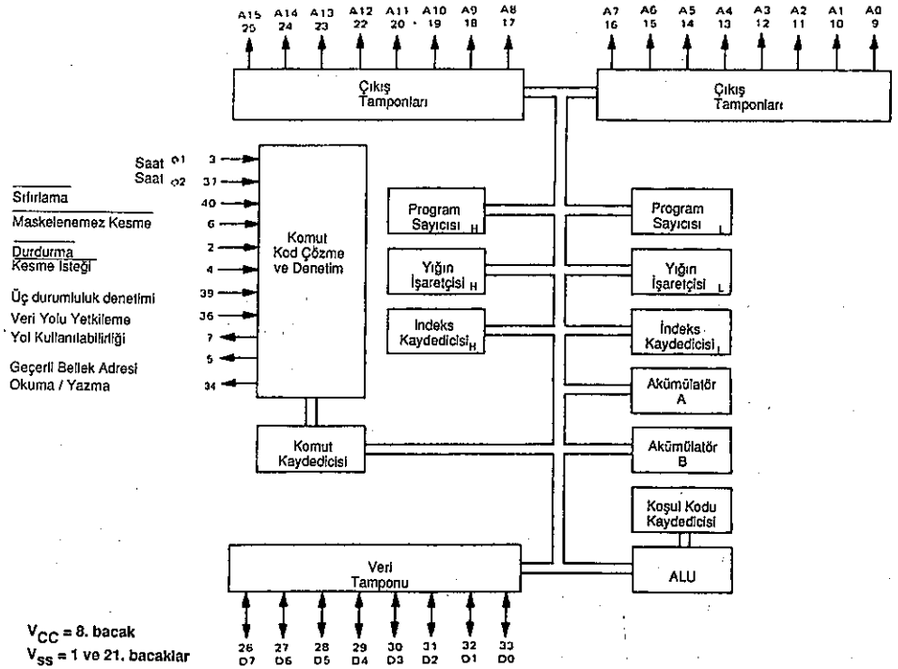
ŞEKİL 7 - FREKANSA BAĞLI DEĞİŞİMLER



ŞEKİL 8- SICAKLIĞA BAĞLI DEĞİŞİMLER



GENİŞLETİLMİŞ BLOK DİYOGRAM



MİB SİNYAL TANIMLARI

MİB'nin düzgün çalışması, belirli işlevlerin başarılması bağlanmış belli denetim ve zamanlama sinyallerine ve onun durumunu belirlenebilmei için diğer sinyal hatlarının gözlenmesine gerek vardır.

Faz ø1 ve Faz ø2 Saatleri - İki bacak, V_{CC} gerilim altında çalışan üstüste çakışmayan iki fazlı saat için kullanılır.

Adres Yolu (A0-A15) - 16 bacak adres yolu için kullanılır. Çıkışlar, standart bir TTL yükünü ve 130pF'ı sürebilme yeteneğinde üç durumlu yol sürücüleridir. Çıkış KAPALI durumunda (kesimde) olduğunda, elektriksel olarak açık devre olur. Bu, MİB'nin DMA uygulamalarında kullanılabilmesine izin verir.

Veri Yolu (D0-D7) - 8 bacak veri yolu için kullanılır. Veriyin bellek ve çevresel birimler arasında aktarımını sağlayan veri yolu, iki yönlüdür. Aynı zamanda, standart bir TTL yükünü ve 130pF'ı sürebilme yeteneğinde üç durumlu çıkış tamponlarına sahiptir.

Durdurma(Halt) - Bu giriş alçak durumda olduğunda, makinenin bütün faaliyetleri duracaktır. Bu giriş, düzeye-duyarlıdır. Durdurma (Halt) modunda makine, komutun sonunda duracak ve Yol Kullanılabilir (Bus Available) "1" düzeyinde, Geçerli Bellek Adresi (Valid Memory Address) "0" düzeyinde ve diğer bütün üç durumlu hatlar, üç-durumlu modda olacaktır.

1. faz sinyalinin son 250 nsn'si süresince Durdurma hattı düzeyinde bir geçiş ortaya çıkmamalıdır. Tek komutlu çalışmayı sağlayabilmek için, Durdurma hattı, bir Saat Saykılı süreyle yüksek düzeye gitmelidir.

Üç-Durumluluk Denetimi (Three State Control-TSC) - Bu giriş bütün adres hatlarının ve okuma/yazma hattının KAPALI (kesim) veya yüksek empedans durumuna gitmesine neden olur. Bu durum, TSC=2.0 V olduktan 700 nsn sonra oluşacaktır. Geçerli Bellek Adresi ve Yol Kullanılabilir sinyalleri, alçak düzeye çekilmeye zorlanır. Veri yolu, kendi Yetkilendirmesine sahip olduğundan (Veri Yolu Yetkilendirmesi (Data Bus Enable)), TSC tarafından etkilenmez. DMA uygulamalarında, üç durumluluk denetim hattı, 1. faz saatinin düşen kenarında yüksek düzeye getirilmelidir. Bu işlevin uygun şekilde çalışması için ø1 saati yüksek düzeyde ve ø2 saati alçak düzeyde tutulmalıdır. Diğer cihazların doğrudan adres bellek erişimi yapabilmeleri için, Adres yolu kullanılabilir duruma gelecektir. Dinamik bir birim olduğu için MİB, bu durumda yalnızca 4,5 µsn süreyle tutulabilir veya MİB içindeki veri bozulacaktır.

Okuma/Yazma (Read/Yazma-R/W) - Bu TTL uyumlu çıkış, MİB'nin okuma modunda (yüksek düzey) mi yoksa yazma modunda (alçak düzey) mi olduğunu çevrebirimlerine ve bellek birimlerine bildirir. Bu sinyalin normal bekleme durumu Okuma'dır. (Yüksek düzey) Üç durumluluk denetiminin yüksek düzeye gitmesi, okuma/yazma hattını KAPALI durumuna (yüksek empedans) götürür. Ayrıca, işlemci durdurulmuşken, yine KAPALI durumunda olacaktır. Bu çıkış, 90 pF ve bir standart TTL yükü sürme yeteneğine sahiptir.

Geçerli Bellek Adresi (VMA) - Bu çıkış, çevrebirimleri adres yolu üzerinde geçerli adres olduğunu bildirir. Normal çalışmada, bu sinyal PIA ve ACIA gibi çevresel arabirimlerin yetkilendirilmesi için kullanılmalıdır. Bu sinyal üç durumlu değildir. Bir standart TTL yükü ve 90 pF, bu yüksek-etkin sinyal tarafından doğrudan sürülebilir.

Veri Yolu Yetkilendirmesi (Data Bus Enable-DBE) - Bu giriş MİB veri yolu için üç-durumlu denetim sinyalidir ve yüksek düzeyde olduğunda, zaman yol sürücülerini yetkilendirecektir. Bu giriş TTL uyumludur, ancak normal çalışmada, 2. faz saati tarafından sürülmelidir. MİB okuma saykılı sırasında, veri yolu sürücüleri devre içinden yetkisizlenecektir. Doğrudan Bellek Erişimi (DMA) uygulamalarında olduğu gibi veri yolu diğer birimlerce denetlenmek istendiğinde, DBE alçak düzeyde tutulmalıdır.

Yol Kullanılabilir (Bus Available-BA) - Yol Kullanılabilir sinyali, normalde alçak düzeyde olacaktır; etkinleştirildiğinde, mikroişlemcinin durdurulduğunu ve adres yolunun kullanılabilir durumda olduğunu gösteren yüksek düzeye gidecektir. Eğer Durdurma (Halt) hattı alçak düzeydeyse veya mikroişlemci, BEKLEME (WAIT) komutunun yürütülmesinin sonucu olarak BEKLEME durumundaysa, bu sinyal ortaya çıkacaktır. Böyle bir durumda, bütün üç-durumlu çıkış sürücüleri KAPALI durumlarına ve diğer çıkışlar da normal pasif düzeylerine dönecektir. İşlemci, BEKLEME durumundan, maskelenebilir (maske biti I=0) veya maskelenemez kesmenin ortaya çıkması halinde çıkacaktır. Bu çıkış, 30 pF ve bir standart TTL yük sürme yeteneklidir.

Kesme İsteği (Interrupt Request-IRQ) - Bu düzeye-duyarlı giriş, makine içerisinde bir kesme sırasının üretilmesini gerektirir. İşlemci, yapılan istek dikkate alınmadan önce o anda yürütülmekte olan komutun tamamlanmasına kadar bekleyecektir. Tam sırada, Koşul Kodu Kaydedicisindeki kesme maskeleme biti kurulmamışsa, makine kesme sırasına başlayacaktır. İndeks Kaydedicisi, Program Sayıcısı, Akümülatörler ve Koşul Kodu Kaydedicisi yığına saklanır. Ardından MİB, başka kesmelerin ortaya çıkmaması için kesme maskeleme bitini yüksek düzeye kurarak kesme isteğine cevap verecektir. Saykılın sonunda, FFF8 ve FFF9 bellek konumlarında bulunan vektör adresinin gösterdiği 16-bitlik adres, Program Sayıcısına yüklenecektir. Bu konumlardaki adresin yüklenmesi MİB'nin, bellekteki kesme yordamına dallanmasına neden olur.

Durdurma hattı, kesmelerin yürütülmesi esnasında yüksek düzeyde olmalıdır. Durdurma hattı alçak düzeyde iken, kesmeler içten mandallanacaktır.

IRQ, yongaya içten, yüksek empedanslı yukarı çekme direnciyle bağlıdır, bununla birlikte, kesmelerin en uygun biçimde denetimi ve telli-VEYA için 3 kΩ'luk bir direnç dıştan V_{CC}'ye bağlanmalıdır.

Sıfırlama (Reset) - Bu giriş, işlemcinin ilk başlatılmasında veya güç arızasının sonucunda oluşan güç düşmesi durumundan dolayı MİB'ni sıfırlamak ve başlatmak için kullanılır. Girişte eğer yüksek düzey algılanırsa, bu MİB'nin

yeniden başlatma sırasını yürütmeye başlayacağını gösterecektir. Bu işlem, işlemciyi, sıfırlama durumundan başlatma durumuna geçirmek için bir yordamı yürütmeye başlayacaktır. Tüm üst-sıralı adres hatları yüksek düzeye çekilecektir. Yeniden başlatmada, Program Sayıcısı tarafından adreslenmiş programı yüklemek için, bellekteki en son iki konum (FFFE, FFFF) kullanılacaktır. Yeniden başlatma yordamı sırasında, kesme maskeleme biti kurulur ve MİB, $\overline{IRQ}$ tarafından kesilmeden önce sıfırlanmalıdır.

Şekil 9'da, yeniden başlatma sonrasında sonrasında işlemcinin başlatılması gösterilmiştir. Sıfırlama ucu, Vcc 4,75 volta ulaştıktan itibaren en az sekiz saat peryodu boyunca alçak düzeyde tutulmalıdır. Eğer sıfırlama ucu, ø2'nin düşen kenarından önce yüksek düzeye giderse, bir sonraki 1'de, ilk yeniden başlatma bellek vektör adresi (FFFE), adres hatları üzerinde görünecektir. Bu bellek konumu, program sayıcısına saklanacak üst-sıralı sekiz biti içermelidir. Bunun ardından, bir sonraki adres olan FFFF, program sayıcısına saklanacak alt-sıralı sekiz biti içermelidir.

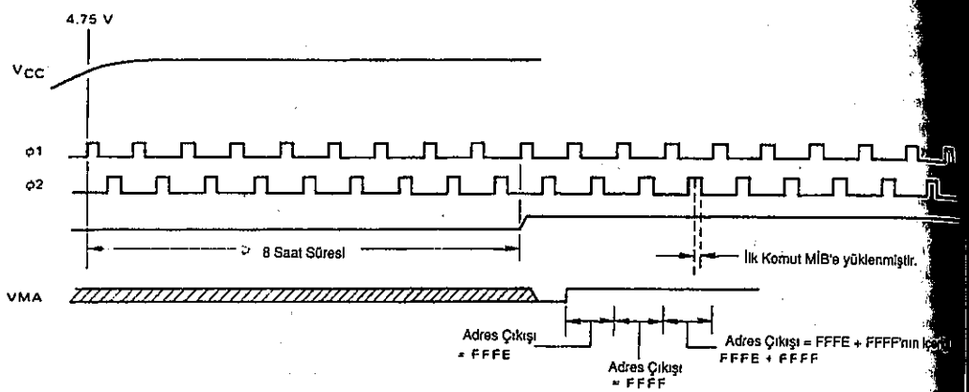
$\overline{\text{Maskelenemez Kesme}}$ ($\overline{NMI}$) - Bu giriştek[illegible] den kenar, işlemcide, bir maskelenemez kesme [illegible] üretilmiş olmasını gerektirir. Kesme istek sinyali[illegible] gibi MİB, $\overline{NMI}$ sinyalini dikkate almadan önce, [illegible] tülmekte olan komutun tamamlanmasını bekley[illegible] şul Kodu Kaydedicisindeki kesme maskeleme [illegible] etkilemez.

İndeks Kaydedicisi, Program Sayıcısı, Akümü[illegible] Koşul Kodu Kaydedicisi yığına saklanır. Saykılın [illegible] FFFC ve FFFD bellek konumlarındaki vektör adre[illegible] terdiği 16-bitlik adres, program sayıcısına yüklene[illegible] konumlardaki adresin yüklenmesi, MİB'nin bellekte[illegible] lenemez kesme yordamına dallanmasına neden olur.

$\overline{MMI}$, yongaya içten yüksek empedanslı yuka[illegible] direnciyle bağlıdır, bununla beraber, kesmelerin [illegible] biçimde denetimi ve telli-VEYA için 3 kΩ'luk bir direnç [illegible] Vcc'ye bağlanmalıdır.

$\overline{IRQ}$ ve $\overline{NMI}$ girişleri ø2 esnasında örneklenen [illegible] kesme hatlarıdır ve kesme yordamı, komutun tamam[illegible] sını izleyen ø1'de başlayacaktır.

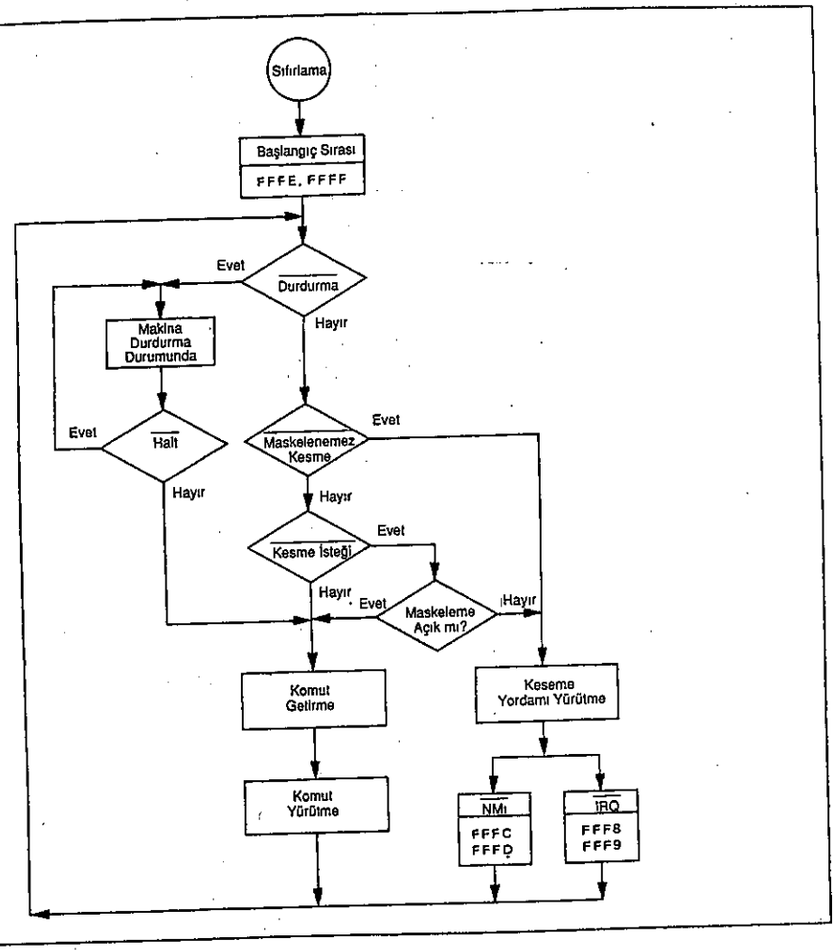
ŞEKİL 9 - YENİDEN BAŞLATMA SONRASINDA MİB'NİN BAŞLATILMASI



TABLO 1 - KESME VEKTÖRLERİ BELLEK HARİTASI

Vektör		Tanımı
EBD	EKD	
FFFE	FFFF	Tekrar başlatma
FFFC	FFFD	Maskelenemez kesme
FFFA	FFFB	Yazılım kesmesi
FFF8	FFF9	Kesme isteği

ŞEKİL 10 - MİB AKIŞ DİYAGRAMI



MİB KAYDEDİCİLERİ

MİB, programcı tarafından kullanılabilen üç adet 16-bitlik ve üç adet 8-bitlik kaydediciye sahiptir (Şekil 1).

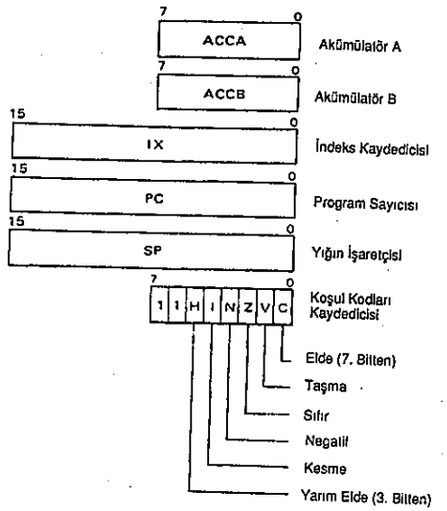
Program Sayıcısı - Program sayıcısı, yürürlükteki program adresini gösteren 2 bayt'lık (16 bit) kaydedicidir.

Yığın İşaretçisi - Yığın işaretçisi, harici bir it/çek yığınındaki kullanılabilir bir sonraki gözün adresini içeren iki bayt'lık bir kaydedicidir. Bu yığın normal olarak herhangi uygun bir konumda (adreste) bulunan rastgele erişilebilir okuma/yazma belleğidir. Yığının içine bilgi saklanmasını gerektiren birçok uygulamalarda, güç kesildiği zaman, yığındaki bilgi silinmemelidir.

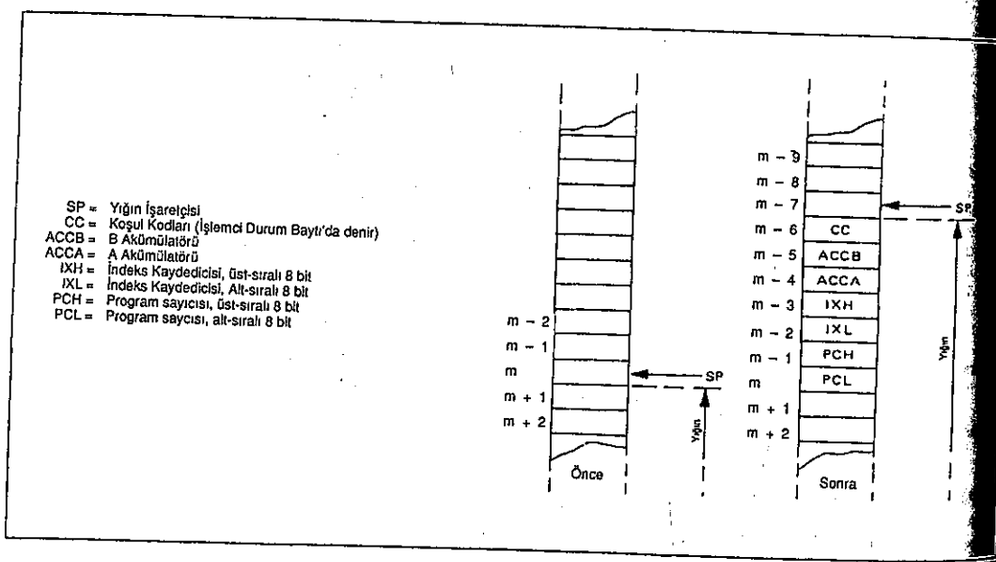
İndeks Kaydedicisi - İndeks kaydedicisi, veriyi saklamak veya bellek adreslemesinin İndeksli modda 16-bitlik bellek adresi olarak kullanılan iki bayt'lık kaydedicidir.

Akümülatörler - MİB, işlenenleri ve aritmetik mantık biriminden (ALU) gelen sonuçları tutmak için kullanan iki adet 8-bitlik akümülatör içerir.

ŞEKİL 11 - MİKROİŞLEMCİ BİRİMİNİN PROGRAMLAMA MODELİ



ŞEKİL 12 - MİKROİŞLEMCİ DURUMLARININ YIĞINDA SAKLANMASI



Koşul Kodu Kaydedicisi - Koşul Kodu Kaydedicisi, aritmetik Mantık Birimi işlemleri olan: Negatif (N), Sıfır (Z), Taşma (V), 7. bitten gelen Elde (C) ve 3. bitten gelen yarım elde (H) işlemlerinin sonuçları gösterir. Koşul Kodu Kaydedicisinin bu bitleri, koşullu dallanma komutları için test koşulu olarak kullanılır. 4. bit kesme maskeleme biti (I) 'dir. Koşul Kodu Kaydedicisinin kullanılmayan (b6 ve b7) bitleri, 1'e kurulu durumdadır.

Şekil 12'de, mikroişlemci durumlarının yığında saklanması sırası gösterilmiştir.

MİB KOMUT TAKIMI

MC6800, 72 farklı komutluk bir takıma sahiptir. İkili ve ondalık aritmetik, mantıksal, kaydırma, döndürme, yükleme, saklama, koşullu ve koşulsuz dallanma, kesme ve yığın yönetim komutlarını içerir (Tablo 2'den Tablo 6'ya kadar).

MİB ADRESLEME MODLARI

MC6800; programcı tarafından kullanılabilen, komuttaki kodun ve komut tipinin fonksiyonu olan yedi adresleme modunu sahip, 8bitlik mikroişlemci birimidir. Belirli komutlara ilişkin adresleme modlarının özeti, makina saykılı olarak verilmiş komut yürütme süreleriyle birlikte Tablo 7'den bakılabilir. 1 MHz'lik saat frekansı ile bu süreler mikrosaniyeler düzeyinde olacaktır.

Akümülatör (ACCX) Adresleme - Yalnızca akümülatör adreslemesidir, akümülatör A ya da akümülatör B belirtilmiştir. Bunlar bir baytlık komutlardır.

Dolaysız Adresleme - Dolaysız adreslemede; işlenenin komutun ikinci ve üçüncü baytlarında belirtildiği LDS ve LDX komutları hariç, işlenen, komutun ikinci baytında yer alır. MİB, yürütülmek üzere bir dolaysız adresleme komutu getirildiğinde, bu konumu adresler. Bunlar iki veya üç baytlık komutlardır.

Doğrudan Adresleme - Doğrudan adreslemede, işlenenin adresi, komutun ikinci baytında belirtilir. Doğrudan adresleme, kullanıcıya, makinenin en alt-sıralı 256 baytını, yani 0'dan 255'e kadar olan konumunu doğrudan adresleyebilmesi olanağı tanır. Artırılmış yürütme süreleri, verinin bu konumlara saklanmasıyla başarılır. Birçok konfigürasyonda, bu konumların, rastgele erişilebilir bellek konumları olması gerekir. Bunlar, iki baytlık komutlardır.

Genişletilmiş Adresleme - Genişletilmiş adreslemede, komutun ikinci baytında içerilen adres, işlenen adresinin üst sıralı sekiz biti olarak kullanılır. Komutun üçüncü baytı, işlenen adresinin alt-sıralı sekiz biti olarak kullanılır. Bu adres, bellekteki mutlak adrestir. Üç baytlık komutlardır.

İndekslenmiş Adresleme - İndekslenmiş adreslemede, komutun ikinci baytının içerdiği adres, MİB'nin içindeki indeks kaydedicisinin en alt-sıralı sekiz bitine eklenir. Ardından Elde de, indeks kaydedicisinin üst-sıralı sekiz bitine eklenir. Elde edilen sonuç, belleği adreslemede kullanılır. Değiştirilmiş adres, geçici adres kaydedicisinde tutulduğundan, indeks kaydedicisinde değişiklik olmaz. Bunlar, iki baytlık komutlardır.

Örtük Adresleme - Örtük adresleme modunda, adresi (yani yığın işaretçisi, indeks kaydedicisi, vs.) komut vermektedir. Bir baytlık komutlardır.

Bağıl Adresleme - Bağıl adreslemede, komutun ikinci baytının içerdiği adres ile program sayıcısının en alt-sıralı bitlerinin iki fazlası toplanır. Üretilecek Elde veya Borç bitleri, program sayıcısının üst-sıralı sekiz bitine eklenir. Bu, kullanıcının, komutun bulunduğu yerden -125 ile +129 bayt aralığındaki verileri adresleyebilmesine imkan verir. Bunlar, iki baytlık komutlardır.

TABLO 2 - MİKROİŞLEMCİ KOMUT TAKIMI - ALFABETİK SIRA

ABA	Akümülatörleri topla	CLR	Sıfırla	PUL	Veriyl Çek
ADC	Eldeli toplama	CLV	Taşmayı Sıfırla	ROL	Sola Döndür
ADD	Toplama	CMP	Karşılaştırma	ROR	Sağa Döndür
AND	Mantıksal Ve	COM	Tümleme	RTI	Kesmeden Geri Dön
ASL	Aritmetik Sola Kaydır	CPX	İndeks Kaydedicisi nı Karşılaştırma	RTS	Altyordamdan Geri Dön
ASR	Aritmetik Sağa Kaydır	DAA	Ondalık Ayarı	SBA	Akümülatörleri Çıkart
BCC	Elde Sıfıra Dallan	DEC	1 Azaltma	SBC	Eldeli Çıkartma
BCS	Elde Bir ise Dallan	DES	Yığın İşaretçisini 1 Azalt	SEC	Eldeyi "1"e Kur
BEQ	Sıfıra Eşitse Dallan	DEX	İndeks Kaydedicisi nı 1 Azalt	SEI	Kesme Maskesini "1"e Kur
BGE	Sıfıra Eşit veya Daha Büyükse Dallan	EOR	Özel VEYA	SEV	Taşmayı "1"e Kur
BGT	Sıfırdan Büyükse Dallan	INC	1 Artırma	STA	Akümülatörü Depola
BHI	Daha Yüksekse Dallan	INS	Yığın İşaretçisini 1 Artır	STS	Yığın İşaretçisine Depola
BIT	Bit Testi	INX	İndeks Kaydedicisi nı 1 Artır	STX	İndeks Kaydedicisine Depola
BLE	Daha Küçükse veya Eşitse Dallan	JMP	Atla		Çıkartma
BLS	Daha Düşükse veya Aynıysa Dallan	JSR	Altyordama Atla	SUB	Yazılım Kesmesi
BLT	Sıfırdan Daha Küçükse Dallan	LDA	Akümülatöre Yükle	SWI	Akümülatörleri Aktar
BMI	Eksi ise Dallan	LDS	Yığın İşaretçisine Yükle	TAB	Akümülatörleri Koşul Kodu
BNE	Sıfıra Eşit Değilse Dallan	LDX	İndeks Kaydedicisi na Yükle	TAP	Kaydedicisi Aktar
BPL	Artı ise Dallan	LSR	Mantıksal Sağa Kaydır	TBA	Akümülatörleri Aktar
BRA	Her zaman Dallan	NEG	Negatif	TPA	Koşul Kodu Kaydedicisini
BSR	Altyordama Dallan	NOP	İşlem Yok		Akümülatöre Aktar
BVC	Taşma Sıfırsa Dallan	ORA	Akümülatör İçinde VEYA İşlemi Yap	TST	Test
BVS	Taşma bir ise Dallan	PSH	Veriyl İt	TSX	Yığın İşaretçisini İndeks
CBA	Akümülatörleri Karşılaştır				Kaydedicisine Aktar
CLC	Eldeyi Sıfırla			TXS	İndeks Kaydedicisini Yığın İşaretçisine
CLI	Kesme Maskesini Sıfırla				Aktar
				WAI	Kesme için Bekle

TABLO 3 - AKÜMÜLATÖR VE BELLEK KOMUTLARI

İŞLEMLER	ANIMSATICI	DOLAYSIZ OP	~	=	DOĞRUDAN OP	~	=	İNDEKSLİ OP	~	=	GENİŞ OP	~	=	ÖRTÜK OP	~	=	BOOLE ARİTMETİK İŞLEMLERİ (Tüm kaydedici etiketleri içerikleri göstermektedir)	KOŞ. KODU KAYD. 5 H	4 I	3 N	2 Z	1 V	0 C
Topla	ABDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	:	•	:	:	:	:
	ABDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	:	•	:	:	:	:
Akümülatörle topla	ABA													1B	2	1	A + B → A	:	•	:	:	:	:
EldeliTopla	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	:	•	:	:	:	:
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	:	•	:	:	:	:
VE'le	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	:	R	•	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	:	R	•	•
Bitl Test Et	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	:	R	•	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	:	R	•	•
Sil	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Karşılaştır	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	:	:	:	:
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	:	:	:	:
	COA													11	2	1	A − B	•	•	:	:	:	:
A küm.le karşılatrı	COM							63	7	2	73	6	3				M − M	•	•	:	R	S	S
1'e tümle.al	COMA													43	2	1	A − A	•	•	:	R	S	S
	COMB													53	2	1	B − B	•	•	:	R	S	S
	NEG							60	7	2	70	6	3				00 − M → M	•	•	:	:	①	①
2'ye tüm. al	NEGA													40	2	1	00 − A → A	•	•	:	:	①	①
(Negatif yap)	NEGB													50	2	1	00 − B → B	•	•	:	:	①	①
	DAA													19	2	1		•	•	:	:	:	③
Ondalık Ayar.	INC							6A	7	2	7A	6	3				M + 1 → M	•	•	:	:	④	•
	INCA													4A	2	1	A + 1 → A	•	•	:	:	④	•
	INCB													5A	2	1	B + 1 → B	•	•	:	:	④	•
Azalt	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	:	R	•	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	:	R	•	•
Özel VEYA'la	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	:	:	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	:	:	R	•
Artır	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Ven'yi İt	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Venyi Çek	ROL							69	7	2	79	6	3				M } A, B: C ← b7 ← b0	•	•	:	:	⑤	:
	ROLA													49	2	1		•	•	:	:	⑤	:
Sola Döndür	ROLB													59	2	1		•	•	:	:	⑤	:
	ROR							66	7	2	76	6	3				M } A, B: C → b7 → b0	•	•	:	:	⑤	:
Sağa Döndür	RORA													46	2	1		•	•	:	:	⑤	:
	RORB													56	2	1		•	•	:	:	⑤	:
Aritmetik Sola Kaydır	ASL							68	7	2	78	6	3				M } A, B: C ← b7 ← b0 ← 0	•	•	:	:	⑤	:
	ASLA													48	2	1		•	•	:	:	⑤	:
Aritmetik sağa Kaydır	ASLB													58	2	1		•	•	:	:	⑤	:
	ASR							67	7	2	77	6	3				M } A, B: b7 → b0 → C	•	•	:	:	⑤	:
Manlıksal Sağa Kaydır	ASRA													47	2	1		•	•	:	:	⑤	:
	ASRB													57	2	1		•	•	:	:	⑤	:
Aküm.'e Sakla	LSR							64	7	7	74	6	3				M } A, B: 0 → b7 → b0 → C	•	•	R	:	⑤	:
	LSRA													44	2	1		•	•	R	:	⑤	:
	LSRB													54	2	1		•	•	R	:	⑤	:
Çıkar	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	:	:	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	:	:	R	•
Aküm.'dan çıkar	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	:	:	:	:
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	:	:	:	:
Eldeli Çıkar	SBA													10	2	1	A − B → A	•	•	:	:	:	:
	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	:	:	:	:
Aküm.'ü Aktar	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	:	:	:	:
Sıfır veya Eksi Testi Yap	TAB													16	2	1	A → B	•	•	:	R	•	•
	TBA													17	2	1	B → A	•	•	:	R	•	•
	TST							6D	7	2	7D	6	3				M − 00	•	•	:	R	R	•
	TSTA													4D	2	1	A − 00	•	•	:	R	R	•
	TSTB													5D	2	1	B − 00	•	•	:	R	R	•
																		H	I	N	Z	V	C

KISALTMALAR

- OP İşlem Kodu (Onaltı)
- ~ MİB Saykıl Sayısı
- = Programın Bayt Sayısı
- + Aritmetik Artı
- − Aritmetik Eksi
- • Boole VE
- M_{SP} Yığın İşaretçisinin gösterdiği bellek konumunun içeriği
- + Boole VEYA
- • Bole Özel VEYA
- M M'nin tümleyeni
- → 'e Aktar
- 0 Bit = Sıfır
- 00 Byte = Sıfır

KOŞUL KODU SEMBOLLERİ

- H 3. bitten Yarım Elde
- I Kesme Maske
- N Negatif (İşaret bitli)
- Z Sıfır (bayt)
- V Taşma, 2'ye tümleyen
- C 7. bitten Elde
- R Dalma sıfırla
- S Dalma kur.
- .... Test et, doğruysa 1'e 0'a kur
- • Etkisi Yoktur

TABLO 4 - İNDEKS KAYDEDİCİSİ VE YIĞIN İŞLEME KOMUTLARI

İŞARETÇİ İŞLEMLERİ	KAM. KISILT.	DOLAYSIZ OP	~	=	DOĞRUDAN OP	~	=	İNDEKS OP	~	=	GENİŞL. OP	~	=	ÖRTÜR OP	~	=	BOLE/ARİTMETİK İŞLEM	KOŞUL KODU KAYD. 5 H	4 I	3 N	2 Z	1 V	0 C
İndeks Kayd. Karşılaştır	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				X_H − M, X_L − (M + 1)	•	•	⑦	:	⑧	•
İndeks Kayd. Azalt	DEX													09	4	1	X − 1 → X	•	•	•	:	:	•
Yığın İşriç. Azalat	DES													34	4	1	SP − 1 → SP	•	•	•	•	•	•
Yığın İşriç. Artır	INX													08	4	1	X + 1 → X	•	•	•	:	•	•
İndeks Kayd. Yükle	INS													31	4	1	SP + 1 → SP	•	•	•	•	•	•
Yığın İşriç. Yükle	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				M → X_H, (M + 1) → X_L	•	•	⑨	:	R	•
İndeks Kayd. Sakla	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				M → SP_H, (M + 1) → SP_L	•	•	⑨	:	R	•
Yığın İşriç. Sakla	STX				DF	5	2	EF	7	2	FF	6	3				X_H → M, X_L → (M + 1)	•	•	⑨	:	R	•
İndeks Kayd. → Yığın İşriç	STS				9F	5	2	AF	7	2	BF	6	3				SP_H → M, SP_L → (M + 1)	•	•	⑨	:	R	•
Yığın İşriç. → İndeks Kayd	TXS													35	4	1	X − 1 → SP	•	•	•	•	•	•
	TSX													30	4	1	SP + 1 → X	•	•	•	•	•	•

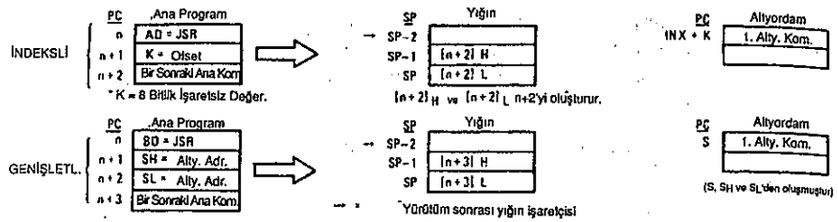
TABLO 5 - ATLAMA VE DALLANMA KOMUTLARI

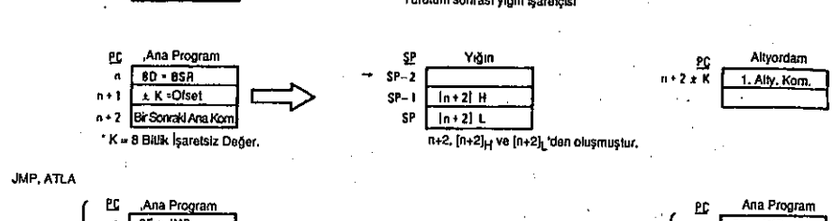
İŞLEMLER	KOY. KISILT.	BAĞIL OP	~	=	İNDEKS OP	~	=	GENİŞL. OP	~	=	ÖRTÜK OP	~	=	DALLN. TESTİ	KOŞUL KODU KAYD. 5 H	4 I	3 N	2 Z	1 V	0 C
Daima Dallan	BRA	20	4	2										None	•	•	•	•	•	•
Elde Sıfırsa Dallan	BCC	24	4	2										C = 0	•	•	•	•	•	•
Elde Kuruluysa Dallan	BCS	25	4	2										C = 1	•	•	•	•	•	•
= Sıfır ise Dallan	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
≥ Sıfır ise Dallan	BGE	2C	4	2										N ⊙ V = 0	•	•	•	•	•	•
> Sıfır ise Dallan	BGT	2E	4	2										Z + (N ⊙ V) = 0	•	•	•	•	•	•
≤ Sıfır ise Dallan	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Küçük veya eşit ise dallan	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
< Sıfır ise Dallan	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
- ise Dallan	BLT	2D	4	2										N ⊙ V = 1	•	•	•	•	•	•
Sıfıra eşit Değilse Dallan	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Taşma Sıfırsa Dallan	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Taşma Kuruluysa Dallan	BVC	28	4	2										V = 0	•	•	•	•	•	•
+ ise Dallan	BVS	29	4	2										V = 1	•	•	•	•	•	•
Alt Yordama Dallan	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Atla	BSR	8D	8	2											•	•	•	•	•	•
Al Yordama Atla	JMP				6E	4	2	7E	3	3				} Bkz. Özel İşlemler:	•	•	•	•	•	•
İşlem Yok	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
Kesmeden Geriye Dön	NOP										01	2	1	Yanlızca Prok. Say. İlerletir	•	•	•	•	•	•
Alt Yordamdan Geriye Dön	RTI										38	10	1		⑩					
Yazılımı kesmesi	RTS										39	5	1		•	•	•	•	•	•
Kesme İçin Bekle	SWI										3F	12	1	} Bkz. Özel İşlemler:	•	•	•	•	•	•
	WAI										3E	9	1		•	⑪	•	•	•	•

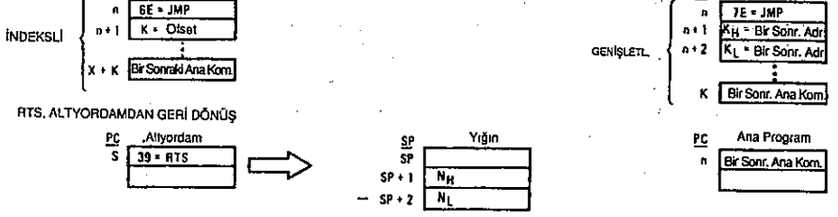
VMA ALÇAK düzeyde tutuluyorken, WAI; Adres Yolu, R/W ve Veri Yolunu 3 - durumlu Moda sokar

ÖZEL İŞLEMLER

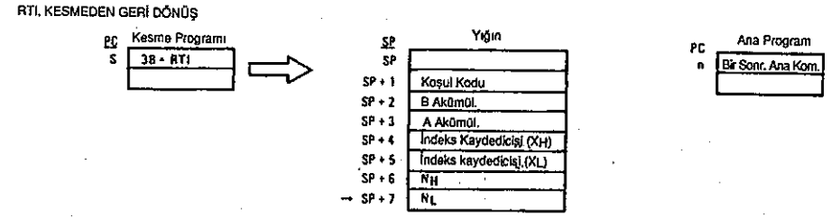
JSR, ALTYORDAMA ATLA



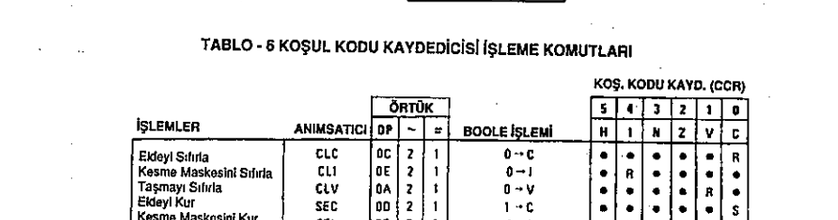




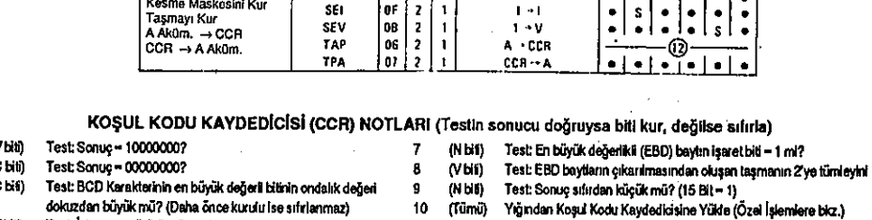
JMP, ATLA



RTS, ALTYORDAMDAN GERİ DÖNÜŞ



RTI, KESMEDEN GERİ DÖNÜŞ



TABLO - 6 KOŞUL KODU KAYDEDİCİSİ İŞLEME KOMUTLARI

İŞLEMLER	ANIMSATICI	ÖRTÜK OP	~	#	BOOLE İŞLEMİ	KOŞ. KODU KAYD. (CCR) 5 H	4 I	3 N	2 Z	1 V	0 C
Eldeyi Sıfırla	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Kesme Maskesini Sıfırla	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Taşmayı Sıfırla	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Eldeyi Kur	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Kesme Maskesini Kur	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Taşmayı Kur	SEV	0B	2	1	1 → V	•	•	•	•	S	•
A Aküm. → CCR	TAP	06	2	1	A → CCR	—	—	(12)	—	—	—
CCR → A Aküm.	TPA	07	2	1	CCR → A	•	•	•	•	•	•

KOŞUL KODU KAYDEDİCİSİ (CCR) NOTLARI (Testin sonucu doğruysa biti kur, değilse sıfırla)

1 (V biti) Test: Sonuç = 10000000?
2 (C biti) Test: Sonuç = 00000000?
3 (C biti) Test: BCD Karakterinin en büyük değerli bitinin ondalık değeri dokuzdan büyük mü? (Daha önce kurulu ise sıfırlanmaz)
4 (V biti) Test: İşlenen = yürütüm öncesi 10000000 mi?
5 (V biti) Test: İşlenen = yürütüm öncesi 01111111 mi?
6 (V biti) Test: Kaydırma ortaya çıktıktan sonra N ⊕ C'nin sonucuna eşitle.
7 (N biti) Test: En büyük değerlikli (EBD) baytın işaret biti = 1 mi?
8 (V biti) Test: EBD baytların çıkarılmasından oluşan taşmanın 2'ye tümleyini
9 (N biti) Test: Sonuç sıfırdan küçük mü? (15 Bit = 1)
10 (Tümü) Yığından Koşul Kodu Kaydedicisine Yükle (Özel İşlemlere bkz.)
11 (I biti) Kesme oluştuğunda kur. Eğer daha önce kurulu ise bekleme durumundan çıkmak için bir Maskelenemez Kesme gerekir.
12 (Tümü) A Akümülatörünün içeriğine bağlı olarak kur

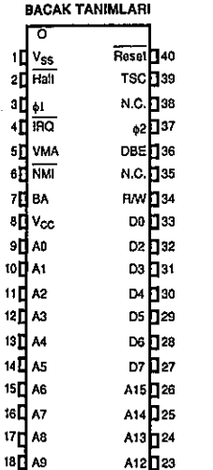
TABLO 7 - KOMUT ADRESLEME MODLARI VE İLGİLİ YÜRÜTME SÜRELERİ
(Süreler Makina Saykılıdır)

	Çift İşlenen	ACCX	Dolaysız	Doğrudan	Genişletilmiş	İndeksli	Örtük	Bağıl
ABA		•	•	•	•	•	2	•
ADC	x	•	2	3	4	5	•	•
ADD	x	•	2	3	4	5	•	•
AND	x	•	2	3	4	5	•	•
ASL		2	•	•	6	7	•	•
ASR		2	•	•	6	7	•	•
BCC		•	•	•	•	•	•	4
BCS		•	•	•	•	•	•	4
BEA		•	•	•	•	•	•	4
BGE		•	•	•	•	•	•	4
BGT		•	•	•	•	•	•	4
BHI		•	•	•	•	•	•	4
BIT	x	•	2	3	4	5	•	•
BLE		•	•	•	•	•	•	4
BLS		•	•	•	•	•	•	4
BLT		•	•	•	•	•	•	4
BMI		•	•	•	•	•	•	4
BNE		•	•	•	•	•	•	4
BPL		•	•	•	•	•	•	4
BRA		•	•	•	•	•	•	4
BSR		•	•	•	•	•	•	8
BVC		•	•	•	•	•	•	4
BVS		•	•	•	•	•	•	4
CBA		•	•	•	•	•	2	•
CLC		•	•	•	•	•	2	•
CLI		•	•	•	•	•	2	•
CLR		2	•	•	6	7	•	•
CLV		•	•	•	•	•	2	•
CMP	x	•	2	3	4	5	•	•
COM		2	•	•	6	7	•	•
CPX		•	3	4	5	6	•	•
DAA		•	•	•	•	•	2	•
DEC		2	•	•	6	7	•	•
DES		•	•	•	•	•	4	•
DEX		•	•	•	•	•	4	•
EOR	x	•	2	3	4	5	•	•

	Çift İşlenen	ACCX	Dolaysız	Doğrudan	Genişletilmiş	İndeksli	Örtük
INC		2	•	•	6	7	•
INS		•	•	•	•	•	4
INX		•	•	•	•	•	4
JMP		•	•	•	3	4	•
JSR		•	•	•	9	8	•
LDA	x	•	2	3	4	5	•
LDS		•	3	4	5	6	•
LDX		•	3	4	5	6	•
LSR		2	•	•	6	7	•
NEG		2	•	•	6	7	•
NOP		•	•	•	•	•	2
ORA	x	•	2	3	4	5	•
PSH		•	•	•	•	•	4
PUL		•	•	•	•	•	4
ROL		2	•	•	6	7	•
ROR		2	•	•	6	7	•
RTI		•	•	•	•	•	10
RTS		•	•	•	•	•	5
SBA		•	•	•	•	•	2
SBC	x	•	2	3	4	5	•
SEC		•	•	•	•	•	2
SEI		•	•	•	•	•	2
SEV		•	•	•	•	•	2
STA	x	•	•	4	5	6	•
STS		•	•	5	6	7	•
STX		•	•	5	6	7	•
SUB	x	•	2	3	4	5	•
SWI		•	•	•	•	•	12
TAB		•	•	•	•	•	2
TAP		•	•	•	•	•	2
TBA		•	•	•	•	•	2
TPA		•	•	•	•	•	2
TST		2	•	•	6	7	•
TSX		•	•	•	•	•	4
TSX		•	•	•	•	•	4
WAI		•	•	•	•	•	9

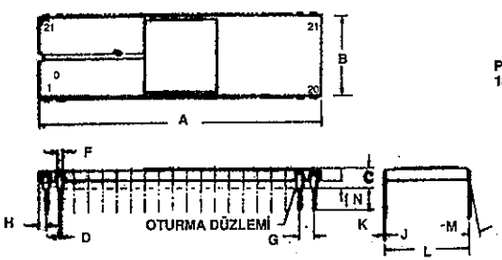
NOT: Kesme süresi, WAI komutu hariç, komut yürütümünün bitiminden itibaren 12 saykıldır

BACAK TANIMLARI



PAKET BOYUTLARI
KILIF 715-02
(CERAMIC)

Plastik Paket boyutları için 185. sayfaya bakın.



BOY.	MİLİMETRE MİN.	MİLİMETRE MAKS	İNÇ. MİN.	İNÇ. MAKS
A	50.29	51.31	1.980	2.020
B	14.86	15.62	0.585	0.615
C	2.54	4.19	0.100	0.165
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	14.60	15.37	0.575	0.605
M	–	10°	–	10°
N	0.51	1.52	0.020	0.060

NOT: BAŞLAR, 0.25 mm (010) ile DOĞRU POZİSYONLANMIŞTIR (YERLEŞİM DÜZLEMİNDE), EN BÜYÜK MAT'L KOŞULUNDA

ÇEVRİM İŞLEMLERİN ÖZETİ

Tablo 8, her bir komut için her saykılda adres yolunda, veri yolunda, Geçerli Bellek Adresi hattında (VMA) ve okuma/yazma (R/W) hattında hazır bulunan bilginin ayrıntılı açıklamasını verir.

Bu bilgi, denetim programının yürütülmesi gibi yazılım ve donanımın her ikisinin de hata ayıklama işlemleri esnasında beklenen sonuçlarla elde edilenler arasında karşılaştırma yapılmasında kullanılır. Bilgi, adresleme modu ve her komuttaki saykıl sayısına göre gruplar halinde sınıflandırılmıştır (Genel olarak, aynı adresleme modlu ve saykıl sayılı komutlar aynı yöntemle yürütülür, farklı olanlar tablodaki gibi tilmiştir).

TABLO 8 - İŞLEM ÖZETİ

Adres Modu ve Komutları	Saykıllar	Saykıl num.	VMA Hattı	Adres Yolu	R/W Hattı	Veri Yolu
DOLAYSIZ						
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1 2	1 1	Op Kod Adresi Op Kod Adresi + 1	1 1	Op Kod İşlenen Veri
CPX LDS LDX	3	1 2 3	1 1 1	Op Kod Adresi Op Kod Adresi + 1 Op Kod Adresi + 2	1 1 1	Op Kod İşlenen Veri (Üst-sıralı Bayt) İşlenen Veri (Alt-sıralı Bayt)
DOĞRUDAN						
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1 2 3	1 1 1	Op Kod Adresi Op Kod Adresi + 1 İşlenen Adresi	1 1 1	Op Kod İşlenen Adresu İşlenen Veri
CPX LDS LDX	4	1 2 3 4	1 1 1 1	Op Kod Adresi Op Kod Adresi + 1 İşlenen Adresi İşlenen Adresi + 1	1 1 1 1	Op Kod İşlenen Adres İşlenen Veri (Üst-sıralı Bayt) İşlenen Veri (Alt-sıralı Bayt)
STA	4	1 2 3 4	1 1 0 1	Op Kod Adresi Op Kod Adresi + 1 Varış yeri Adresi Varış yeri Adresi	1 1 1 0	Op Kod Varış yeri Adresi İlgisiz Veri (Not 1) Akümülatörden Veri
STS STX	5	1 2 3 4 5	1 1 0 1 1	Op Kod Adresi Op Kod Adres + 1 İşlenen Adres İşlenen Adres İşlenen Adres + 1	1 1 1 0 0	Op Kod İşlenen Adres İlgisiz Veri (Not 1) Kaydedici Verisi (Üst-sıralı Bayt) Kaydedici Verisi (Alt-sıralı Bayt)
İNDEKSLİ						
JMP	4	1 2 3 4	1 1 0 0	Op Kod Adresi Op Kod Adresi + 1 İndeks Kaydedicisi İndeks Kaydedicisi Artı Ofset (ve/ya Elde)	1 1 1 1	Op Kod Ofset İlgisiz Veri (Not 1) İlgisiz Veri (Not 1)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	5	1 2 3 4 5	1 1 0 0 1	Op Kod Adresi Op Kod Adresi + 1 İndeks Kaydedicisi İndeks Kaydedicisi Artı Ofset (ve/ya Elde) İndeks Kaydedicisi Artı Ofset	1 1 1 1 1	Op Kod Ofset İlgisiz Veri (Not 1) İlgisiz Veri (Not 1) İşlenen Veri
CPX LDS LDX	6	1 2 3 4 5 6	1 1 0 0 1 1	Op Kod Adresi Op Kod Adresi + 1 İndeks Kaydedicisi İndeks Kaydedicisi Artı Ofset (ve/ya Elde) İndeks Kaydedicisi Artı Ofset İndeks Kaydedicisi Artı Ofset + 1	1 1 1 1 1 1	Op Kod Offset İlgisiz Veri (Not 1) İlgisiz Veri (Not 1) İşlenen Veri (Üst-sıralı Bayt) İşlenen Veri (Alt-sıralı Bayt)

TABLO 8 - İŞLEM ÖZETİ (devamı)

Adres Modu ve Komutları	Saykıllar	Saykıl num.	VMA Hattı	Adres Yolu	R/W Hattı	Veri Yolu
İNDEKSLİ (devamı)						
STA	6	1 2 3 4 5 6	1 1 0 0 0 1	Op Kod Adresi Op Kod Adresi + 1 İndeks Kaydedicisi İndeks Kaydedicisi Artı Ofset (ve/ya Elde) İndeks Kaydedicisi Artı Ofset İndeks Kaydedicisi Artı Ofset	1 1 1 1 1 0	Op Kod Ofset İlgisiz Veri (Not 1) İlgisiz Veri (Not 1) İlgisiz Veri (Not 1) İşlemci Veri
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	7	1 2 3 4 5 6 7	1 1 0 0 1 0 1/0 (Not 3)	Op Kod Adresi Op Kod Adresi + 1 İndeks Kaydedicisi İndeks Kaydedicisi Artı Ofset (ve/ya Elde) İndeks Kaydedicisi Artı Ofset İndeks Kaydedicisi Artı Ofset İndeks Kaydedicisi Artı Ofset	1 1 1 1 1 0	Op Kod Ofset İlgisiz Veri (Not 1) İlgisiz Veri (Not 1) Yürürlükteki İşlenen Veri İlgisiz Veri (Not 1) Yeni İşlem Verisi (Not 3)
STS STX	7	1 2 3 4 5 6 7	1 1 0 0 0 1 1	Op Kod Adresi Op Kod Adresi + 1 İndeks Kaydedicisi İndeks Kaydedicisi Artı Ofset (ve/ya Elde) İndeks Kaydedicisi Artı Ofset İndeks Kaydedicisi Artı Ofset İndeks Kaydedicisi Artı Ofset + 1	1 1 1 1 1 0 0	Op Kod Ofset İlgisiz Veri (Not 1) İlgisiz Veri (Not 1) İlgisiz Veri (Not 1) İşlenen Veri (Üst-sıralı Bayt) İşlenen Veri (Alt-sıralı Bayt)
JSR	8	1 2 3 4 5 6 7 8	1 1 0 1 1 0 0 0	Op Kod Adresi Op Kod Adresi + 1 İndeks Kaydedicisi Yığın İşaretçisi Yığın İşaretçisi - 1 Yığın İşaretçisi - 2 İndeks Kaydedicisi İndeks Kaydedicisi Artı Ofset (ve/ya Elde)	1 1 1 0 0 1 1 1	Op Kod Ofset İlgisiz Veri (Not 1) Geri Dönüş Adresi (Alt-sıralı Bayt) Geri Dönüş Adresi (Üst-sıralı Bayt) İlgisiz Veri (Not 1) İlgisiz Veri (Not 1) İlgisiz Veri (Not 1)
GENİŞLETİLMİŞ						
JMP	3	1 2 3	1 1 1	Op Kod Adresi Op Kod Adresi + 1 Op Kod Adresi + 2	1 1 1	Op Kod Atlama Adresi (Üst-sıralı Bayt) Atlama Adresi (Alt-sıralı Bayt)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1 2 3 4	1 1 1 1	Op Kod Adresi Op Kod Adresi + 1 Op Kod Adresi + 2 İşlenen Adresi	1 1 1 1	Op Kod İşlenen Adresi (Üst-sıralı Bayt) İşlenen Adresi (Alt-sıralı Bayt) İşlenen Veri
CPX LDS LDX	5	1 2 3 4 5	1 1 1 1 1	Op Kod Adresi Op Kod Adresi + 1 Op Kod Adresi + 2 İşlenen Adresi İşlenen Adresi + 1	1 1 1 1 1	Op Kod İşlenen Adresi (Üst-sıralı Bayt) İşlenen Adresi (Alt-sıralı Bayt) İşlenen Veri (Üst-sıralı Bayt) İşlenen Veri (Alt-sıralı Bayt)
STA A STA B	5	1 2 3 4 5	1 1 1 0 1		1 1 1 1 0	Op Kod Varış yeri Adresi (Üst-sıralı Bayt) Varış yeri Adresi (Alt-sıralı Bayt) İlgisiz Veri (Not 1) Akümülatörden Veri
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	6	1 2 3 4 5 6	1 1 1 1 0 1/0 (Not 3)	Op Kod Adresi Op Kod Adresi + 1 Op Kod Adresi + 2 İşlenen Adresi İşlenen Adresi İşlenen Adresi	1 1 1 1 1 0	Op Kod İşlenen Adresi (Üst-sıralı Bayt) İşlenen Adresi (Alt-sıralı Bayt) Yürürlükteki İşlenen Veri İlgisiz Veri (Not 1) Yeni İşlenen Verisi (Not 3)

TABLO 8 - İŞLEM ÖZETİ (devamı)

Adres Modu ve Komutları	Say-kılar	Say-kıl num.	VMA Hattı	Adres Yolu	R/W Hattı	Veri Yolu
GENİŞLETİLMİŞ (devamı)						
STS STX	6	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	İşlenen Adresi (Üst-sıralı Bayt)
		3	1	Op Kod Adresi + 2	1	İşlenen Adresi (Alt-sıralı Bayt)
		4	0	İşlenen Adresi	1	İlgisiz Veri (Not 1)
		5	1	İşlenen Adresi	0	İşlenen Veri (Üst-sıralı Bayt)
		6	1	İşlenen Adresi + 1	0	İşlenen Veri (Alt-sıralı Bayt)
JSR	9	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Altyordam Adresi (Üst-sıralı Bayt)
		3	1	Op Kod Adresi + 2	1	Altyordam Adresi (Alt-sıralı Bayt)
		4	1	Altyordam Başlangıç Adresi	1	Bir Sonraki Komutun Op Kodu
		5	1	Yığın İşaretçisi	0	Geri Dönüş Adresi (Alt-sıralı Bayt)
		6	1	Yığın İşaretçisi - 1	0	Geri Dönüş Adresi (Üst-sıralı Bayt)
		7	0	Yığın İşaretçisi - 2	1	İlgisiz Veri (Not 1)
		8	0	Op Kod Adresi + 2	1	İlgisiz Veri (Not 2)
		9	1	Op Kod Adresi + 2	1	Altyordam Adresi (Alt-sıralı Bayt)
DOĞAL						
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Bir Sonraki Komutun Op Kodu
DES DEX INS INX	4	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Bir Sonraki Komutun Op Kodu
		3	0	Bir Önceki Kaydedici İçeriği	1	İlgisiz Veri (Not 1)
		4	0	Yeni Kaydedici İçeriği	1	İlgisiz Veri (Not 1)
PSH	4	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Bir Sonraki Komutun Op Kodu
		3	1	Yığın İşaretçisi	0	Akümülatör Verisi
		4	0	Yığın İşaretçisi - 1	1	Akümülatör Verisi
PUL	4	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Bir Sonraki Komutun Op Kodu
		3	0	Yığın İşaretçisi	1	İlgisiz Veri (Not 1)
		4	1	Yığın İşaretçisi + 1	1	Yığından Alınan İşlenen Veri
TSX	4	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Bir Sonraki Komutun Op Kodu
		3	0	Yığın İşaretçisi	1	İlgisiz Veri (Not 1)
		4	0	Yeni İndeks Kaydedicisi	1	İlgisiz Veri (Not 1)
TXS	4	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Bir Sonraki Komutun Op Kodu
		3	0	İndeks Kaydedicisi	1	İlgisiz Veri
		4	0	Yeni Yığın İşaretçisi	1	İlgisiz Veri
RTS	5	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	İlgisiz Veri (Not 2)
		3	0	Yığın İşaretçisi	1	İlgisiz Veri (Not 1)
		4	1	Yığın İşaretçisi + 1	1	Bir Sonraki Komutun Adresi (Üst-sıralı Bayt)
		5	1	Yığın İşaretçisi + 2	1	Bir Sonraki Komutun Adresi (Alt-sıralı Bayt)

TABLO 8 - İŞLEM ÖZETİ (devamı)

Adres Modu ve Komutları	Say-kılar	Say-kıl num.	VMA Hattı	Adres Yolu	R/W Hattı	Veri Yolu
GENİŞLETİLMİŞ (devamı)						
[illegible]	9	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Bir Sonraki Komutun Op Kodu
		3	1	Yığın İşaretçisi	0	Geri Dönüş Adresi (Alt-sıralı Bayt)
		4	1	Yığın İşaretçisi - 1	0	Geri Dönüş Adresi (Üst-sıralı Bayt)
		5	1	Yığın İşaretçisi - 2	0	İndeks Kaydedicisi (Alt-sıralı Bayt)
		6	1	Yığın İşaretçisi - 3	0	İndeks Kaydedicisi (Üst-sıralı Bayt)
		7	1	Yığın İşaretçisi - 4	0	A Akümülatörünün İçeriği
		8	1	Yığın İşaretçisi - 5	0	B Akümülatörünün İçeriği
		9	1	Yığın İşaretçisi - 6 (Not 4)	1	Koşul Kodu Kaydedicisinin İçeriği
[illegible]	10	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	İlgisiz Veri (Not 2)
		3	0	Yığın İşaretçisi	1	İlgisiz Veri (Not 1)
		4	1	Yığın İşaretçisi + 1	1	Yığından alınan Koşul Kodu Kaydedicisinin İçeriği
		5	1	Yığın İşaretçisi + 2	1	Yığından alınan B Akümülatörünün İçeriği
		6	1	Yığın İşaretçisi + 3	1	Yığından alınan A Akümülatörünün İçeriği
		7	1	Yığın İşaretçisi + 4	1	Yığından alınan İndeks Kaydedicisi (Üst-sıralı Bayt)
		8	1	Yığın İşaretçisi + 5	1	Yığından alınan İndeks Kaydedicisi (Alt-sıralı Bayt)
		9	1	Yığın İşaretçisi + 6	1	Yığından alınan Bir Sonraki Komutun Adresi (Üst-sıralı Bayt)
		10	1	Yığın İşaretçisi + 7	1	Yığından alınan Bir Sonraki Komutun Adresi (Alt-sıralı Bayt)
SWI	12	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	İlgisiz Veri (Not 1)
		3	1	Yığın İşaretçisi	0	Geri Dönüş Adresi (Alt-sıralı Bayt)
		4	1	Yığın İşaretçisi - 1	0	Geri Dönüş Adresi (Üst-sıralı Bayt)
		5	1	Yığın İşaretçisi - 2	0	İndeks Kaydedicisi (Alt-sıralı Bayt)
		6	1	Yığın İşaretçisi - 3	0	İndeks Kaydedicisi (Üst-sıralı Bayt)
		7	1	Yığın İşaretçisi - 4	0	A Akümülatörünün İçeriği
		8	1	Yığın İşaretçisi - 5	0	B Akümülatörünün İçeriği
		9	1	Yığın İşaretçisi - 6	0	Koşul Kodu Kaydedicisinin İçeriği
		10	0	Yığın İşaretçisi - 7	1	İlgisiz Veri (Not 1)
		11	1	FFFA Vektör Adresi	1	Altyordam Adresi (Üst-sıralı Bayt)
		12	1	FFFB Vektör Adresi	1	Altyordam Adresi (Alt-sıralı Bayt)
BAĞIL						
BCC BHI BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMI BVS	4	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Dallanma Offseti
		3	0	Op Kod Adresi + 2	1	İlgisiz Veri (Not 1)
		4	0	Dallanma Adresi	1	İlgisiz Veri (Not 1)
BSR	8	1	1	Op Kod Adresi	1	Op Kod
		2	1	Op Kod Adresi + 1	1	Dallanma Offseti
		3	0	Ana Programdan Geri Dönüş Adresi	1	İlgisiz Veri
		4	1	Yığın İşaretçisi	0	Geri Dönüş Adresi (Alt-sıralı Bayt)
		5	1	Yığın İşaretçisi - 1	0	Geri Dönüş Adresi (Üst-sıralı Bayt)
		6	0	Yığın İşaretçisi - 2	1	İlgisiz Veri (Not 1)
		7	0	Ana Programdan Geri Dönüş Adresi	1	İlgisiz Veri (Not 1)
		8	0	Altyordam Adresi	1	İlgisiz Veri (Not 1)

Not 1. Bu saykıl esnasında adreslenen birim VMA'yı kullanıyorsa, veri yolu yüksek empedans durumuna geçecektir. Yol kapasitesine göre, bir önceki saykıldan gelen veri, veri yolu üzerinde kalabilir.

Not 2. MİB, veriyi dikkate almaz.

Not 3. TST için, VMA = 0 ve İşlenen veri değişmez

Not 4. MİB kesmeyi beklerken, Yol Kullanılabilirliği, şu denetim hatlarının durumunu gösterecek şekilde yüksek duruma gidecektir: VMA alçak düzeydedir, Adres yolu, R/W ve Veri yolu yüksek empedans durumundadır.

EK D Gelişmiş Mikro Birimler
AmZ8002 16-bitlik mikroişlemci

ÖN BİLGİLER

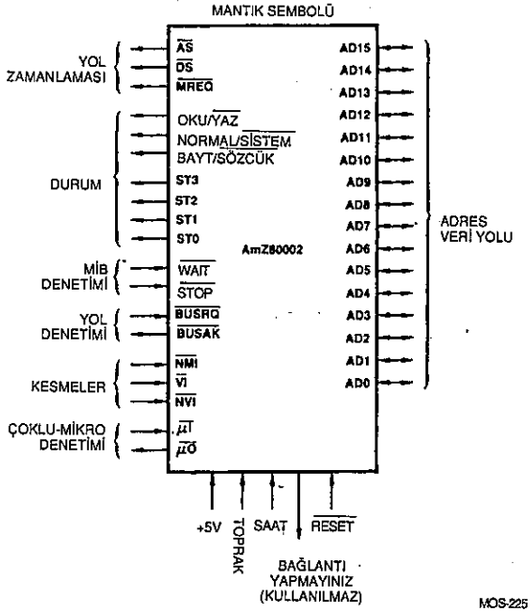
BAŞLICA ÖZELLİKLERİ

- 16 adet genel amaçlı kaydedici
- 64 KBayta kadar doğrudan adresleme
- AmZ8001 mikroişlemcisi ile yazılım uyumluluğu
- Esnek adresleme modları ve güçlü komutlar
- Ayrıcalıklı/ayrıcalıksız işlem modu
- Gelişkin kesme yapısı
- Yonga-içi bellek tazeleme yeteneği
- TTL uyumlu giriş ve çıkışlar
- Tek fazlı saat
- Tek +5V'luk güç kaynağı
- 40-bacaklık paket

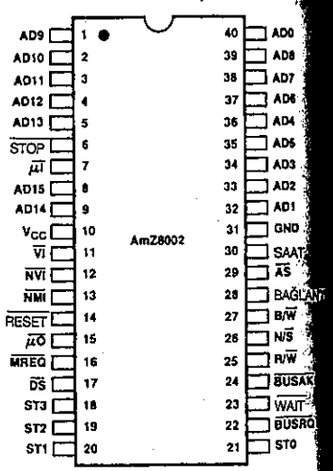
GENEL TANIMI

AmZ8002; AmZ8000 mikroişlemci ailesine ait olan 16-bitlik genel amaçlı MİB'dir. Mimarisi, 16-bitlik genel amaçlı kaydedicilerden oluşmıştır. 16-bitlik adres alanıyla çalışan MİB böylelikle 64 Kbaytlık adresi doğrudan adresleyebilir. Komutlar üç ayrı adresleme alanını (kod, veri ve yığın) kontrol üzere sağlanmıştır. AmZ8002 esnek adresleme modları ve güçlü komut takımını gerçekleştirir. Bu komutlar birçok tipi üzerinde çalışırlar: bit, bayt, sözcük (16 bit), uzun sözcük (32 bit), bayt ve sözcük dizisi. MİB, komutlarını sistem normal olmak üzere iki moddan birinde yürütülebilir. Bu modlar, sırasıyla ayrıcalıklı ve ayrıcalıksız modlar olarak da bilinirler. MİB, aynı zamanda, yonga-içi bellek tazeleme olanağı da içerir. AmZ8002, AmZ8001 mikroişlemcisiyle yazılım uyumludur. AmZ8002, silikon kapılı N-MOS teknolojisi kullanarak imal edilmiştir ve 40 bacaklı DIP kılıfında paketlenmiştir. AmZ8002, çalışması için yalnız +5 Volt güç kaynağına ve tek fazlı saate gereksinim duyar.

MANTIK SEMBOLÜ



BAĞLANTI DİYAGRAMI
Üstten Görünüş



Not: Bacak doğrultu için işaretlenmiştir.

SİPARİŞ BİLGİSİ

Paketleme Tipi	Ortam Sıcaklığı	Maksimum Saat Frekansı
		4MHz
DIP	$0°C \leq T_A \leq 70°C$	AmZ8002DC

BİRİM SİNYAL TANIMLARI

+5V Güç Kaynağı

Toprak

AD0-AD15: Adres/Veri Yolu (İki Yönlü, Üç-Durumlu)

Çoğullanmış adres/veri yolu bütün G/Ç ve bellek işlemleri için kullanılır. Yol üzerindeki YÜKSEK düzey, "1"e, ALÇAK düzey ise "0"a karşılıkı gelir. AD0, en küçük değerlikli, AD15 ise, en büyük değerlikli bit durumundadır. AS ve DS çıkışları yolun, adres için mi yoksa veri için mi kullanılacağını gösterir. Durum çıkış hatları ST0-ST3, işlemin bellek veya G/Ç türü işlem olduğunu belirtir.

AS: Adres Strobu (Çıkış, Üç-Durumlu)

Bu çıkıştaki ALÇAK düzey, AD0-AD15 yolunun adres bilgisi taşıdığını gösterir. Adres bilgisi, AS çıkışının ALÇAK'tan YÜKSEK'e geçişi süresince değişmez (zamanlama diyagramına bakınız). Durum çıkışları STO-ST3, yolun bellek adresi mi yoksa veri adresi mi içerdiğini gösterir.

DS: Veri Strobu (Çıkış, Üç-Durumlu)

Bu çıkıştaki ALÇAK düzey AD0-AD15 yolunun veri aktarımı için kullanıldığını gösterir. R/W çıkışı, veri aktarımının yönünü belirtir. - okuma (veya giriş), verinin MİB'e giriş yaptığını, yazma (veya çıkış) ise, verinin MİB'den çıkış yaptığı anlamındadır. Okuma işlemi süresince, DS, ALÇAK olduğunda veri, yol üzerinde kapılanabilir. DS çıkışının ALÇAK'tan YÜKSEK'e geçişi, MİB'nin veriyi kabul ettiğini belirtir (zamanlama diyagramına bakınız). Yazma işlemi süresince, DS çıkışındaki ALÇAK düzey, verinin yol üzerinde hazırlandığını gösterir. Veri bazen, DS çıkışının ALÇAK'tan YÜKSEK'e geçişinden sonra yoldan çekilebilir (zamanlama diyagramına bakınız).

R/W: Okuma/Yazma (Çıkış, Üç-Durumlu)

Bu çıkış AD0-AD15 yolunun üzerindeki verinün yönünü gösterir. YÜKSEK düzey okuma işlemini, yani verinin MİB'e doğru olduğunu, ALÇAK düzey ise yazma işlemini, yani verinin MİB'den dışarı doğru olduğunu belirtir. Bu çıkış aynı zamanda ALÇAK düzeye giden AS gibi davranır ve bütün işlemler boyunca kararlı kalır (zamanlama diyagramına bakınız).

B/W: Bayt/Sözcük (Çıkış, Üç-durumlu)

Bu çıkış, AD0-AD15 yolu üzerinde aktarılan verinin tipini belirtir. YÜKSEK düzey bayt (8 bit) ve ALÇAK düzey sözcük (16 bit) aktarımını gösterir. Bu çıkış, ALÇAK düzeye giden AS ile aynı aşamada etkinleşir davranır ve tüm işlem boyunca geçerli kalır (zamanlama diyagramına bakınız). MİB tarafından üretilen adres, daima bayt adresidir. Bununla beraber, bellek 16-bitlik sözcükler halinde düzenlenmiştir. Bütün komutlar ve sözcük işlenenleri sözcük esasına göre düzenlenmiştir ve çift adresler tarafından adreslenirler. Böylece bellek ile ilgili bütün sözcük işlemleri için, en küçük değerlikli adres biti sıfır olacaktır. Belleğin, bayt işlemleri için adreslenmesi halinde, en küçük değerlikli adres biti hangi bellek sözcüğü baytının neyi göstereceğini belirler, çift adresler en büyük değerlikli baytı, tek adresler en küçük değerlikli adres baytını belirtir. G/Ç işlemleri durumunda ise, AD0-AD15 yolundaki adres bilgisi, G/Ç portunu ve B/W de, işlenecek verinin, sözcük veya bayt olup olmadığını belirler. G/Ç bayt işlemleri süresince, en küçük değerlikli adres biti A0, AD0-AD15 yolunun hangi yarısının G/Ç işlemleri için kullanılacağını belirler. ST0-ST3 çıkışları, yürürlükteki işlemlerin bellek, normal G/Ç ya da özel G/Ç için mi olacağını gösterir.

ST0-ST3: Durum (Çıkış, Üç-Durumlu)

Bu dört çıkış yürürlükteki işlemlere bağlı olarak kodlanmış bilgi içerir. Durum hat kodları aşağıda tabloda gösterilmiştir.

ST3	ST2	ST1	ST0	
L	L	L	L	İç İşlemler
L	L	L	H	Bellek Tazeleme
L	L	H	L	Normal G/Ç İşlemleri
L	L	H	H	Özel G/Ç İşlemleri
L	H	L	L	Ayrılmıştır
L	H	L	H	Maskelenemez Kesme Alındısı
L	H	H	L	Vektörlendirilmemiş Kesme Alındısı
L	H	H	H	Vektörlenmiş Kesme Alındısı
H	L	L	L	İşlenen için Bellek İşlemleri
H	L	L	H	Yığın için Bellek İşlemleri
H	L	H	L	Ayrılmıştır
H	L	H	H	Ayrılmıştır
H	H	L	L	Komut Getirme için Bellek İşlemleri (Son Sözcük)
H	H	L	H	Komut Getirme için Bellek İşlemleri (İlk Sözcük)
H	H	H	L	Ayrılmıştır
H	H	H	H	Ayrılmıştır

WAIT: Bekle (Giriş)

Bu girişteki ALÇAK düzey, MİB'ne bellek veya G/Ç biriminin veri aktarımı için hazır olmadığını gösterir ve böylece yürürlükteki işlem uzatılmalıdır. İşlem süresince MİB, belirli sürelerde WAIT girişini örnekler (zamanlama diyagramına bakınız). Eğer bu anlarda WAIT girişi ALÇAK düzeydeyse, MİB işlemi uzatmak için bekleme durumuna gidecektir. Bekleme durumu, WAIT girişinin örnekleme YÜKSEK düzeye çıkmasına kadar tekrarlanacaktır.

N/S: Normal/Sistem Modu (Çıkış, Üç-Durumlu)

Bu çıkıştaki YÜKSEK düzey, MİB'nin normal modda, ALÇAK düzey ise, sistem modunda çalışmakta olduğunu gösterir. Bu çıkış, Bayrak Denetim Sözcüğü (FCW) kaydedicisinden türetilmiştir. FCW kaydedicisi, bu dökümanın program durum bilgisi bölümü altında açıklanmıştır.

$\overline{\text{MREQ}}$: Bellek İsteği (Çıkış, Üç-Durumlu)

Bu çıkıştaki ALÇAK yüzey, MİB'nin bellek ile ilgili bir işlem yapmakta olduğunu gösterir.

$\overline{\text{BUSRQ}}$: Yol İsteği (Giriş)

Bu giriştaki ALÇAK düzey, MİB'ne, (DMA gibi) diğer bir birimin yol denetimini almak istediğini belirtir. $\overline{\text{BUSRQ}}$ girişi herhangi bir zamanda ALÇAK düzeyle sürülebilir. MİB, bu girişi içerden eşzamanlaştırır. MİB, bu isteğe, yolun serbest bırakıldığını göstermek üzere $\overline{\text{BUSAK}}$ çıkışını ALÇAK düzeye çekerek cevap verir. Yolun serbest bırakılması, AD0-AD15, $\overline{\text{AS}}$, $\overline{\text{DS}}$, B/$\overline{\text{W}}$, R/$\overline{\text{W}}$, N/$\overline{\text{S}}$, ST0ST3 ve $\overline{\text{MREQ}}$ çıkışlarının yüksek empedans durumunda olacağı anlamına gelir. İstekte bulunan birim, işlemleri tamamlayabilmek için bu hatları MİB'ne benzer biçimde kullanmalıdır. $\overline{\text{BUSRQ}}$ girişi, bütün işlemler tamamlayıncaya kadar ALÇAK düzeyde kalmalıdır ve buna bağlı olarak MİB de, $\overline{\text{BUSAK}}$ çıkışını ALÇAK düzeyde tutacaktır. İşlemler tamamlandıktan sonra, aygıt, AD0-AD15, $\overline{\text{AS}}$, $\overline{\text{DS}}$, B/$\overline{\text{W}}$, R/$\overline{\text{W}}$, N/$\overline{\text{S}}$, ST0-ST3 ve $\overline{\text{MREQ}}$ çıkışlarında yüksek empedans durumunu yetkisizlenmelidir ve $\overline{\text{BUSRQ}}$ girişinin ALÇAK düzeyle sürülmesini durduru. MİB, tekrar yol denetimini aldıktan çok az bir süre sonra $\overline{\text{BUSAK}}$ çıkışını YÜKSEK düzeye çekecektir.

$\overline{\text{BUSAK}}$: Yol Alındısı (Çıkış)

Bu çıkıştaki ALÇAK düzey, yol isteğine cevap olarak MİB'nin yolu serbest bıraktığını gösterir.

$\overline{\text{NMI}}$: Maskelenemez Kesme (Giriş)

Bu giriştaki YÜKSEK'ten ALÇAK'a geçiş, maskelenemez kesme isteğini belirtir. MİB buna, ST0-ST3 çıkışları üzerinde maskelenemez kesme alındısı ile cevap verir ve kesme sırasına girer. $\overline{\text{NMI}}$'daki değişme herhangi bir anda olabilir. Mevcut olan ve çeşit kesmeden en yüksek öncelikli olanı maskelenemez kesmedir.

$\overline{\text{VI}}$: Vektörlenmiş Kesme (Giriş)

Bu giriştaki ALÇAK düzey, vektörlenmiş kesme isteğini belirtir. Vektörlenmiş kesme, öncelik sırasında maskelenemez kesmeden bir sonra gelir. Bayrak ve Denetim Sözcüğü kaydedicisindeki VIE biti, vektörlendirilmiş kesme durumuna girildiğini göstermek üzere mutlaka "1" olmalıdır. MİB buna, ST0-ST3 çıkışları üzerinde vektörlenmiş kesme alındısı kodu ile cevap verir ve kesme sırasına başlar. $\overline{\text{VI}}$ girişi, herhangi bir anda ALÇAK düzeyle sürülebilir ve alındı gelinceye kadar ALÇAK düzeyde tutulmalıdır.

$\overline{\text{NVI}}$: Vektörlenmemiş Kesme (giriş)

Bu giriştaki ALÇAK düzey, vektörlenmemiş, kesme isteğinde bulunulması durumunda ortay çıkar. Vektörlenmemiş kesme, üç tür kesmenin en düşük öncelikli olanıdır. Bayrak ve Denetim Sözcüğü kaydedicisindeki NVIE biti, bu istek durumuna girildiğini belirtmek için mutlaka "1" olmalıdır. MİB buna, ST0-ST3 çıkışları üzerinde vektörlenmemiş kesme alındısı kodu ile cevap verir ve kesme sırasına başlar. $\overline{\text{NVI}}$ girişi, herhangi bir anda ALÇAK düzeyle sürüle[illegible] gelinceye kadar ALÇAK düzeyde tutulmalıdır.

$\overline{\mu\text{I}}$: Mikro-Giriş (Giriş)

Bu giriş, papatya zinciri düzenindeki kaynak iste[illegible] munda ortaya çıkar. Bu konuda, bu dokümanın çoktu[illegible] işlemcili destek olanakları kısmına bakınız.

$\overline{\mu\text{O}}$: Mikro-Çıkış (Çıkış)

Bu çıkış, papatya zinciri düzenindeki kaynak iste[illegible] munda ortaya çıkar. Bu konuda, bu dokümanın çoktu[illegible] işlemcili destek olanakları kısmına bakınız.

$\overline{\text{RESET}}$: Sıfırlama (Giriş)

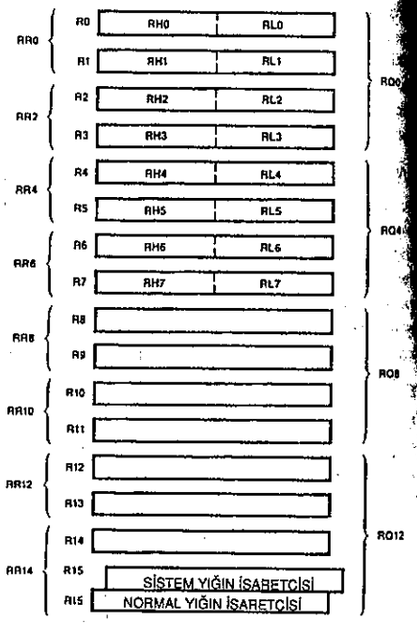
Bu giriştaki ALÇAK düzey, MİB içindeki sıfırlama [illegible] başlatır. Sıfırlama sırasındaki ayrıntılar için başlatma [illegible] na bakın.

CLK: Saat (Giriş)

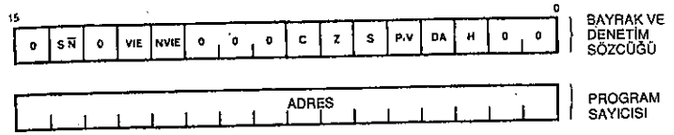
Bütün MİB işlemleri, bu girişe uygulanan sinyal tara[illegible] denetlenir. Saat gerilim düzey gerekleri için DC karakte[illegible] lerine bakınız.

DECOUPLE: (Çıkış)

Yonga-içi altkatman ön gerilimleme üretecinden çıkış. Kul[illegible] nılmaz.



Şekil 1. AmZ8002 Kaydedicileri



Şekil 2. AmZ8002 İşlemci Durumu

[illegible]OP: Dur (Giriş)

[illegible] ALÇAK'ta-etkin giriş, zamana ilişkin işlemlerde bir komutu [illegible]laştırır. Tek adımlama bölümüne bakın.

[illegible]LEMCİNİN ORGANİZASYONU

[illegible]ağıdakiler AmZ8002 MİB'nin kısa görüşleridir. Ayrıntılı [illegible] için, AmZ8001/AmZ8002 komut takımı el kitabına bakın ([illegible]ayın No. AM-PUB086)

Genel Amaçlı Kaydediciler

MİB, Şekil 1'de gösterildiği gibi RQ'dan R15'e kadar 16 adet 16bitlik genel amaçlı kaydedici etrafında düzenlenmiştir. Bayt işlemleri için ilk sekiz kaydedici (R0-R7); RL0, RH0 ve RL7, RH7 gibi tasarlanan onaltı adet sekiz-bitlik kaydedici gibi adreslenebilir. Bu onaltı kaydedici, aynı zamanda, RR0, RR2 ve RR14 gibi uzun sözcük (32 bit) kaydedicilerini oluşturmak için kaydedici çiftleri halinde gruplanabilirler. Benzer şekilde, onaltı kaydedici, 64-bitlik kaydediciler oluşturmak için RQ0, RQ4, RQ8 ve RQ12 gibi dörtlü gruplanabilirler.

YIĞIN İŞARETÇİSİ

AmZ8002 mimarisi, yığının bellekte saklanabilmesine olanak tanır. R0 dışındaki herhangi bir genel amaçlı kaydedici, PUSH ve POP gibi yığın yönetim komutlarında, yığın işaretçisi olarak kullanılabilir. Bununla beraber altyordam çağırma ve altyordamdan geriye dönüş gibi komutlar, yığın işaretçisi olarak kesinlikle, R15 kaydedicisinin kullanılmasını gerektirirler. İki örtük yığın korunmuştur. R15'i yığın işaretçisi olarak kullanılan normal ve R15'i sistem yığın işaretçisi olarak kullanan sistem yığını (Şekil 1'e bakınız). Eğer MİB normal modda çalışıyorsa, R15 etkindir, MİB eğer sistem modundaysa R15'in yerine R15' kullanılacaktır. Örtük yığın işaretçisi, genel kaydedicilerin bir parçasıdır ve böylece, kaydedici işlemleri için mevcut komutlar kullanılarak yönetilebilir.

İŞLEMCİ DURUMU

MİB durumu, 16-bitlik program sayıcısı (PC) ve 16-bitlik bayrak ve denetim sözcük kaydedicisinden (FCW) oluşmuştur (Şekil 2'ye bakınız). Aşağıda, FCW bitlerinin kısa tanımları verilmiştir:

S/$\overline{\text{N}}$: Sistem/Normal - "1" sistem modu ve "0" normal modu gösterir.

VIE: Vektörlenmiş Kesme Yetkilenmesi - "1", vektörlenmiş kesme isteğinin dikkate alınmaya hazır olduğunu gösterir.

NVIE: Vektörlenmemiş Kesme Yetkilenmesi - "1", vektörlenmemiş kesme isteğinin dikkate alınmaya hazır olduğunu gösterir.

C: Elde - "1", aritmetik işlemler yapıldığında, en büyük değerlikli bitinden bir elde üretildiğini gösterir.

Z: Sıfır - "1", işlemin sonucunun sıfır olduğunu gösterir.

S: İşaret - "1", işlemin sonucunun negatif, yani en büyük değerlikli bitin 1 olduğunu gösterir.

P/V: Eşlik/taşma - "1", aritmetik işlemlerinde bir taşmanın ortaya çıktığını gösterir. Mantık işlemleri içinse, bit sonucun eşliğini gösterir.

DA: Ondalık Ayar - Bayt aritmetik işlemlerini kaydeder.

H: Yarım elde - "1" bayt aritmetik işlemlerindeki alt basamağın en büyük değerlikli bitinden bir elde üretilmiş olduğunu gösterir.

VERİ TİPLERİ

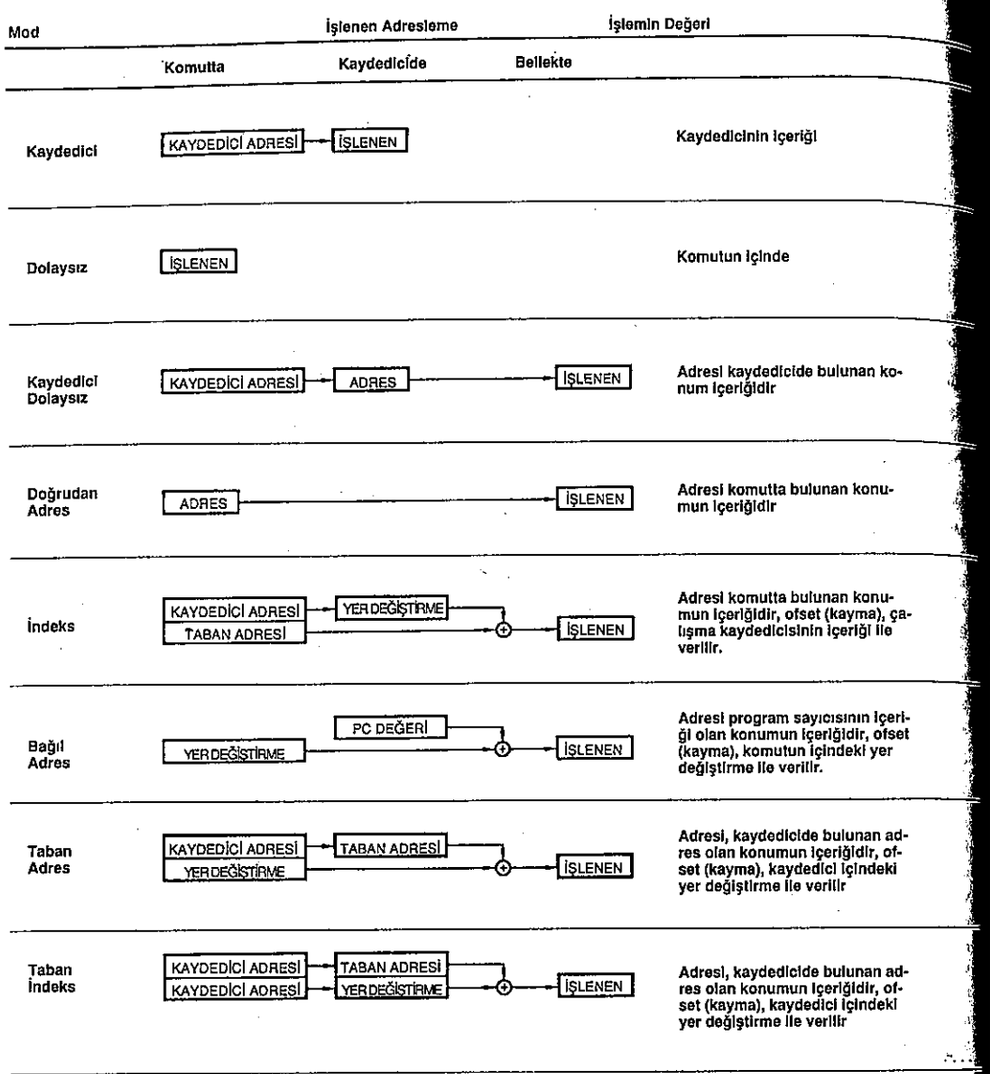
AmZ8002 komutları; bitler, basamaklar (4 bit), baytlar (8 bit), sözcükler (16 bit), uzun sözcükler (32 bit), bayt dizileri ve sözcük dizileri tipi işlenenler üzerinde işlem yaparlar. Bitler kurulabilir, sıfırlanabilir veya test edilebilir. Basamaklar (4 bit) BCD aritmetik işlemlerini kolaylaştırmak için kullanılır. Baytlar, karakterler ve küçük tamsayılar için kullanılır. Sözcükler, tamsayı değerleri ve adresleri için kullanılırken, uzun sözcükler, büyük tamsayı değerleri için kullanılır. Dizi hariç bütün işlenenler, bellek veya genel kaydedicilerde yerleşebilirler. Diziler yalnızca bellekte yerleşirler.

KESME VE TUZAK YAPISI

Kesmeler, programın kesilmesini gerektiren harici asenkron olaylar olarak tanımlanır. Örneğin, hizmet isteyen bir çevrebirimi, programın kesilmesine neden olur. Tuzaklar, belirli bazı koşullar altında, bazı komutların yürütülmesinin sonucu olarak ortaya çıkan eşzamanlı olaylardır. Kesmeler ve tuzakların her ikisi de, AmZ8002 tarafından benzer biçimde işlenir.

AmZ8002 yüksekte düşük öncelik düzeyine doğru sırasıyla üç çeşit kesmeyi destekler - maskelenmez, vektörlenmiş ve vektörlenmemiş. Vektörlenmiş ve vektörlenmemiş kesmeler FCW içindeki ilgili denetim bitleriyle yetkisizlenebilir. AmZ8002'de üç adet tuzak vardır. - Sistem çağrısı, tamamlanmamış opkod ve ayrıcalıklı komut. Tuzaklar, kesmelerden daha yüksek önceliğe sahiptirler.

Bir kesme veya tuzak ortaya çıktığında, yürürlükteki program durumu, otomatik olarak sistem yığınına itilir. Program duru-



Şekil 3. Adresleme Modları

mu, işlemci durumu (yani PC ve FCW) ve ayrıca 16-bitlik tanıtıcıdan oluşur. Tanıtıcı, kesmeye veya tuzağa ilişkin nedeni, kaynağı ve diğer kodlu bilgileri içerir.

Yürürlükteki program durumu saklandıktan sonra, yeni işlemci durumu, bellekte yerleşik yeni işlemci durumu alanından otomatik olarak yüklenir. Bu alan, Yeni Program Durumu Alan İşaretçisi kaydedicisi (NPSAP) tarafından düzenlenir. Daha fazla ayrıntı için AMPUBO86 yayınlarına bakınız.

ADRESLEME MODLARI

AmZ8002 komutları, yürütülmekte olan işlemi, işlenen tipini ve işlenenlerin konumlarını içeren bilgilerden oluşur. İşlenen konumları, genel kaydedici adresleri, bellek adresleri veya G/Ç adresleri tarafından oluşturulmuştur. Verilen bir komutun adresleme modu, adresleme alanına referans verilerek tanımlanır ve işlenen adresini hesaplamak için kullanılan yöntemdir. Adresleme modları, komutta açıkça belirtilmiş veya açıklanmıştır. Şekil 3'de, sekiz adet mutlak adresleme modu gösterilmiştir: Kaydedici (R), Dolaysız (M), Kaydedici Dolaylı (IR), Doğrudan Adres (DA) İndeksli (X), Bağıl Adres (RA), Taban Adres (BA) ve Taban İndeksli (BX).

GİRİŞ/ÇIKIŞ

G/Ç komutları takımı, G/Ç birimleri ile AmZ8002 arasında bayt veya sözcük aktarımını sağlar. G/Ç birimleri, 16-bitlik G/Ç port adreslerini kullanarak adreslenebilir ve G/Ç adres alanı, bellek adres alanının bir parçası değildir. Herbirinin kendi 16-bitlik adres alanı bulunan iki tip G/Ç komutu vardır. G/Ç komutları, giriş, çıkış ve blok aktarma komutlarının kapsamlı biçimini içerir.

MİB ZAMANLAMASI

AmZ8002, belleğe yazma, bellekten okuma gibi komut yürütümlerini, önceden-belirlenmiş makina saykılları sırasını tamamlayarak gerçekleştirir. Her makina saykılı, üç ile on saat saykılına gereksinim duyar. DMA birimleri tarafından gelen yol istekleri makina saykılı sınırlarında dikakte alınırlar. On saat saykılından büyük makina saykılı yoktur, böylece yol isteğine hızlı cevap verilmesi sağlanır (burada, fazladan araya eklenen bekleme durumları olmadığı varsayılmıştır). Makina saykılının başlangıcı, daima $\overline{AS}$ çıkışındaki ALÇAK düzeyli darbe tarafından belirtilir. Durum çıkış hatları ST0-ST3, kodlu biçimde yürürlükteki saykılın içeriğini gösterir.

DURUM HATTI KODLARI

Durum hattı kodları, Arabirim Sinyal tanımlarında ST0-ST3 çıkışları başlığındaki tabloda listelenmişti. Durum kodlarının ayrıntılı tanımları aşağıda verilmiştir:

İç İşlem:

Bu durum kodu, AmZ8002'nin iç işlemlerini makina saykılıyla yapacağını gösterir. Şekil 4'de, iç işlem saykılını açıklar. T1, T2 ve T3 gibi tanımlanan üç saat peryodundan oluşur. $\overline{AS}$ çıkışı, makina saykılının başlangıcını belirtmek için AmZ8002 tarafından ALÇAK düzeyli darbe ile etkin hale getirilecektir. ST0-ST3 çıkışları, iç işlemin koduna yansıtacaktır. $\overline{MREQ}$, $\overline{DS}$ ve $\overline{R/W}$ çıkışları YÜKSEK düzeyde olacaktır. $\overline{N/S}$ çıkışı, bir önceki makina saykılında olduğu gibi aynı düzeyde kalacaktır. AmZ8002, iç işlem saykılı süresince $\overline{WAIT}$ girişini dikkate almayacaktır. MİB, T1 süresince AD0-AD15 yolunu belirsiz bilgiyle sürecektir. Bununla beraber yol, T2 süresince yüksek empedans durumuna gidecektir ve saykılın kalan süresi için aynı durum kalacaktır. Aynı zamanda $\overline{B/W}$ çıkışı da belirsiz bilgi ile MİB tarafından etkin hale getirilir.

Bellek Tazeleme:

Bu durum kodu, AmZ8002'nin bellek tazeleme işlemine gireceğini gösterir. Tazeleme saykılı, Şekil 5'te gösterildiği gibi üç saat peryodundan oluşmuştur. MİB, makina saykılının başlangıcını belirtmek için $\overline{AS}$ çıkını ALÇAK düzeyli darbe ile etkin hale getirir ve ST0-ST3 çıkışları, tazeleme saykıl kodunu yansıtırlar. AD0-AD15 yolunun en küçük değerlikli 9 hattı tazeleme adresini içerir. Bellek, sözcük-düzenli olduğundan, AD0 daima ALÇAK düzeyde olacaktır. En büyük değerlikli yedi yol hattı belirtilmemiştir. $\overline{DS}$ çıkışı bütün saykıl için YÜKSEK düzeyde kalırken, $\overline{R/W}$, $\overline{B/W}$ ve $\overline{N/S}$ çıkışları, tazelemeden önceki makina saykılındaki gibi aynı düzeyde kalacaktır. AD0-AD15 yolu T2 süresince yüksek empedans durumuna gidecek ve saykılın kalan süresince bu düzeyinde kalacaktır.

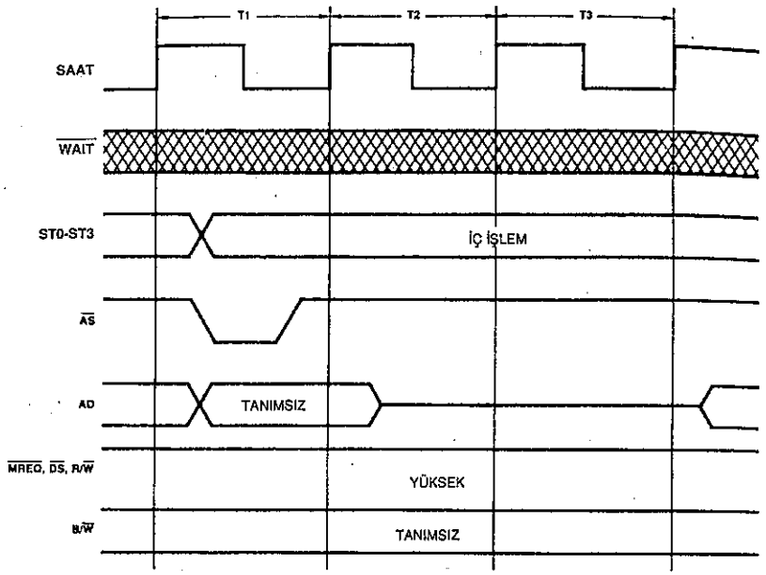
AmZ8002, tazeleme saykılı süresince $\overline{MREQ}$ çıkışını ALÇAK düzeyde etkinleştirecektir. Tazeleme işlemlerinde, $\overline{WAIT}$ girişinin MİB tarafından dikkate alınmadığına dikkat ediniz.

G/Ç İşlemleri:

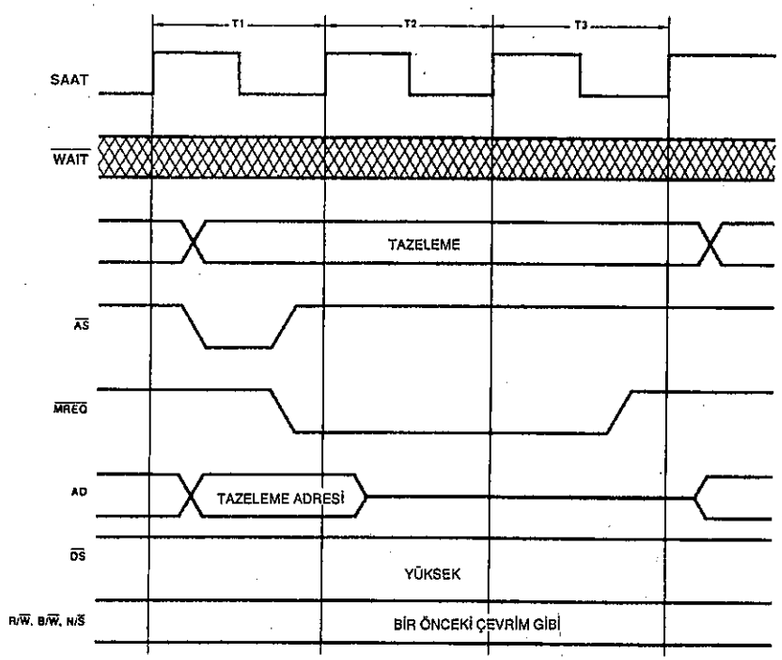
G/Ç işlem saykılı için kullanılan iki adet durum hattı kodu vardır. AmZ8002, iki farklı G/Ç alanı ve normal G/Ç ve özel G/Ç diye anılan iki tip komut sağlar. Her bir G/Ç alanı port adresi diye anılan 16-bitlik adres ile adreslenir. G/Ç işlemlerinin her iki tipi için zamanlama temelde birbirinin aynısıdır. Tipik G/Ç saykılı, Şekil 6'da gösterilen T1, T2, TWA ve T3 gibi dört saat peryodundan oluşur. TWA bekleme durumudur, G/Ç saykılına bir bekleme durumunun eklenmesi daima otomatik olarak gerçekleştirilir. Ek bekleme durumları, $\overline{WAIT}$ girişine ALÇAK düzey uygulanarak eklenebilir. $\overline{WAIT}$ girişi her TW durumu süresince örneklenir. Eğer bu giriş ALÇAK düzeydeyse, fazladan bir bekleme durumu daha eklenecektir. Bekleme durumlarının eklenmesi $\overline{WAIT}$ girişinin YÜKSEK düzeyde olmasına kadar devam eder. G/Ç saykılını tamamlamak için T3 durumu son bekleme durumunu takip edecektir.

G/Ç saykılı süresince ST0-ST3 çıkışları, yürütülen komut tipine bağlı ilgili kodu yansıtır (Normal G/Ç veya Özel G/Ç). Saykılın başlangıcını belirtmek için $\overline{AS}$ çıkışında ALÇAK düzeyli darbe olacaktır. MİB, AD0-AD15 yoluna, yürürlükteki komutta belirtilen 16-bitlik port adresini çıkarır. MİB sistem modunda çalışıyorken $\overline{N/S}$ çıkışı ALÇAK düzeyde olacaktır. $\overline{N/S}$ çıkışının, FCW kaydedicisindeki ilgili bitten türetilmiş olduğunu hatırlayınız. Bütün G/Ç komutları ayrıcalıklı komutlardır ve yalnızca FCW kaydedicisindeki uygun bitten türetilmiştir. Tüm G/Ç komutları ayrıcalıklı komutlardır ve yalnızca, FCW'nin sistem modu çalışmasını göstermesi halinde yürütülmelerine izin verilir. $\overline{MREQ}$ çıkışı YÜKSEK düzeyde olacaktır. AmZ8002 G/Ç komutları, hem sözcük ve hem de bayt işlemlerini destekler. $\overline{B/W}$ çıkışı, komutun bayt ya da sözcük aktarımını belirtip belirtmediğine bağlı olarak YÜKSEK veya ALÇAK düzeyde olacaktır.

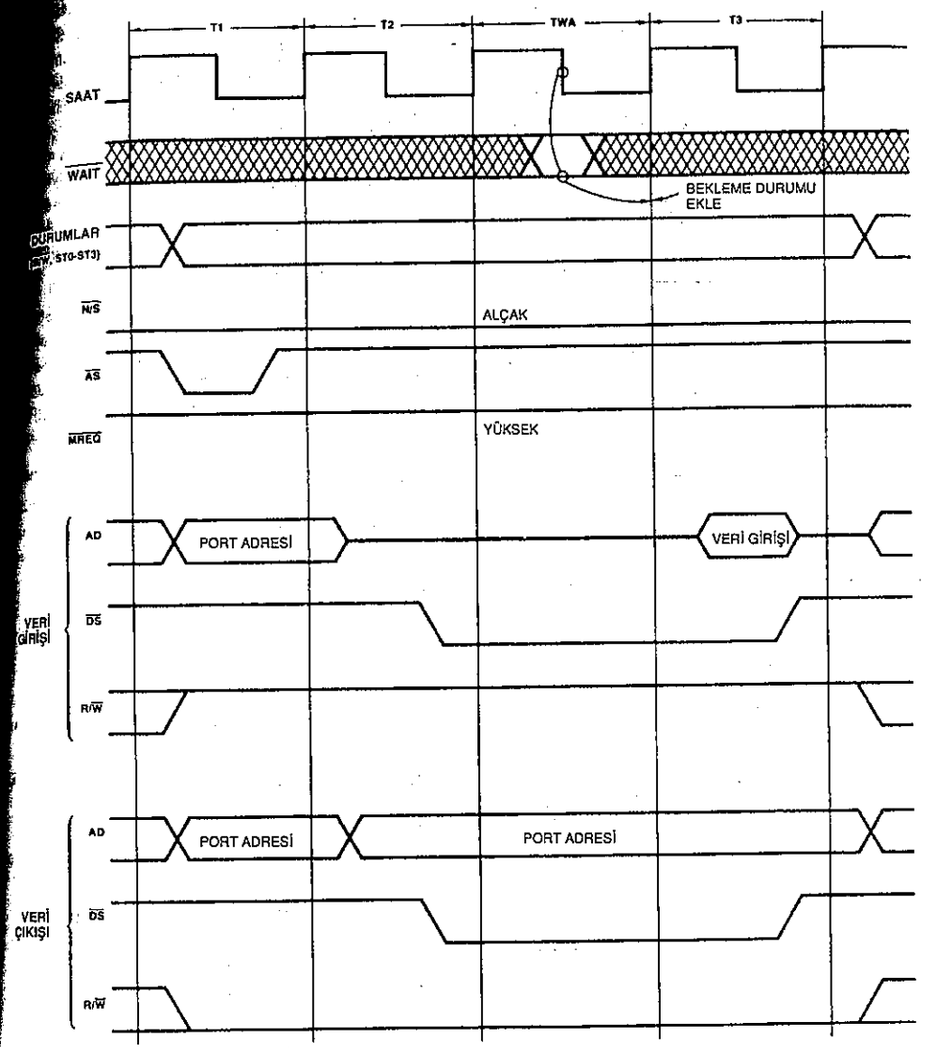
G/Ç aktarımının iki türü mümkündür: Veri girişi, birimin okunmasını ve veri çıkışı ise, birime yazmayı belirtir. Giriş işlemleri için, $\overline{R/W}$ çıkışı YÜKSEK düzeyde olacaktır. AD0-AD15 yolu T2 süresince yüksek empedans durumuna geçecektir. Bayt giriş komutları esnasında, port adresine bağlı olarak MİB veri yolunun çift yada tek yarısından birini okur. Eğer port adresi çift ise, veri yolunun en büyük değerlikli yarısı okunur. Eğer port adresi tek ise, veri yolunun en küçük değerlikli yarısı okunur. Sözcük giriş komutları esnasında, MİB veri yolunun bütün 16 bitini okur. AmZ8002, birime verinin yol üzerinde kapılandığını belirtmek için $\overline{DS}$ çıkışını ALÇAK düzeyle sü-



Şekil 4. İç İşlem Çevrimi



Şekil 5. Tazeleme Çevrimi



Şekil 6. AmZ8002 G/Ç Saykılı

recektir. MİB, veriyi T3 esnasında kabul edecek ve G/Ç işleminin sonunu belirtmek için $\overline{DS}$ çıkışı YÜKSEK düzeye gidecektir.

Veri Çıkışı için, R/$\overline{W}$ çıkışı ALÇAK düzeyde olmalıdır. AmZ8002, AD0-AD15 yoluna veri çıkaracak ve T2 sırasında $\overline{DS}$ çıkışını ALÇAK düzeye çekecektir. Bayt çıkış komutları sırasında, MİB, bayt verisini, veri yolunun yüksek ve düşük yarılarıyla dış mantık devresine kopyalar, A0'ı kullanarak uygun bayt portunu yetkiler. Sözcük çıkış komutları sırasında MİB, verileri, veri yolunun 16 bitinin tümüne çıkarır. T3 esnasında $\overline{DS}$ çıkışı YÜKSEK düzeye gider ve saykıl tamamlanır.

Bellek İşlemleri:

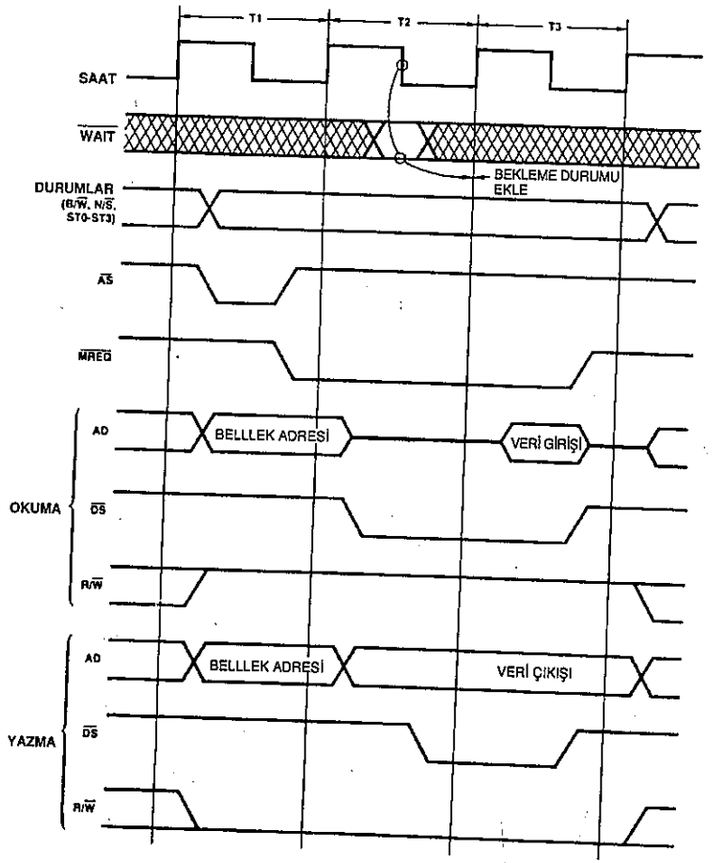
Bellek işlemlerini gösteren dört durum hattı kodu vardır:

a) İşlenene yazan veya işlenenden okuma yapan bellek işlemleri
b) Yığına yazan veya yığından okuma yapan bellek işlemleri

c) Komutun ilk sözcüğünü getiren bellek işlemleri (bazen IF1 diye anılır)
d) Komutun son sözcüğünü getiren belek işlemleri (bazen IFN diye anılır)

Yukarıdaki verilen tüm işlemler, temelde iki grupta değerlendirilebilir: bellek okuma ve belleğe yazma. IF1 ve IFN saykılları durumunda, bellek, program sayıcısı tarafından sağlanan adresten okunacaktır. Bütün AmZ8002 komutları, 16-bitlik sözcük çeşitleridir. Sözcükler daima çift adres ile adreslenir. Böylece IF1 ve IFN saykılları, sözcükler için bellek okuma işleminin yürütülmesine neden olur. Öte yandan işlenen ve yığın işlemlerini gerçekleştiren bellek işlemleri, okuma veya yazma yapabilir. Bundan başka, işlenen, sözcük veya bayt olabilir. Örtük yığın işaretçisini içeren yığın işlemleri için adres R15 (veya R15') tarafından sağlanacaktır. İşlenen işlemleri için, bellek adresi, kullanılan adresleme modu ve komuta göre çeşitli kaynaklardan sağlanır. Şekil 7'de bellek işlemleri saykıl zamanlaması gösterilmiştir. Tipik ola[illegible] T2 ve T3 diye üç adet saat periyodundan oluşur. B[illegible] durumu (TW), WAIT girişinin ALÇAK düzeyde tutu[illegible] T2 ve T3 arasına eklenebilir. WAIT girişi, T2 ve h[illegible] gelen bekleme durumu (TW) sırasında örneklene[illegible] ST3 çıkışları T1'in başında yürürlükteki saykıl için[illegible] sıtır ve AS çıkışı, saykılın başlangıcını belirtmek için A[illegible] düzeyle darbelenecektir. N/S çıkışı yürürlükteki say[illegible] kullanılan adres alanının, normal ya da sistem adre[illegible] olup olmadığını gösterecektir. Şekilden de görüldü[illegible] MREQ çıkışı, bellek işlemini göstermek için T1 sü[illegible] ALÇAK düzeye gidecektir.

Önce okuma işlemini ele alalım. R/W çıkışı YÜKSEK d[illegible] de olacaktır. AmZ8002, T1'in başında AD0-AD15'[illegible] adresle sürecektir. T2 süresince, yol yüksek empedan[illegible] rumuna geçecek ve DS çıkışı, MİB tarafından ALÇAK[illegible] ye çekilecektir. DS, ALÇAK düzeyde olduğunda veri[illegible]



Şekil 7. Bellek İşlemleri

[illegible] kapılanabilir. T1 esnasında B/W çıkışı da, bayt veya [illegible]leminin yapıldığını göstermek için etkin hale getirel[illegible] AmZ8002 belleği sözcük-düzenlidir ve sözcükler çift [illegible]le adreslenirler. Bununla beraber, bayt adresleme[illegible] bellek adresi tek veya çift olabilir, çift adres sözcüğün [illegible]yük değerlikli baytını ve bir sonraki tek adres, aynı [illegible]ün en küçük değerlikli baytını gösterir. Bellekten bayt [illegible]ğu zaman, en küçük değerlikli adres biti dikkate alın[illegible] istenen baytı içeren tüm sözcük yola kapılanır. MİB. [illegible] baytı otomatik olarak seçecektir. AmZ8002 veri kabu[illegible] gösteren DS çıkışını YÜKSEK düzeyle sürecektir.

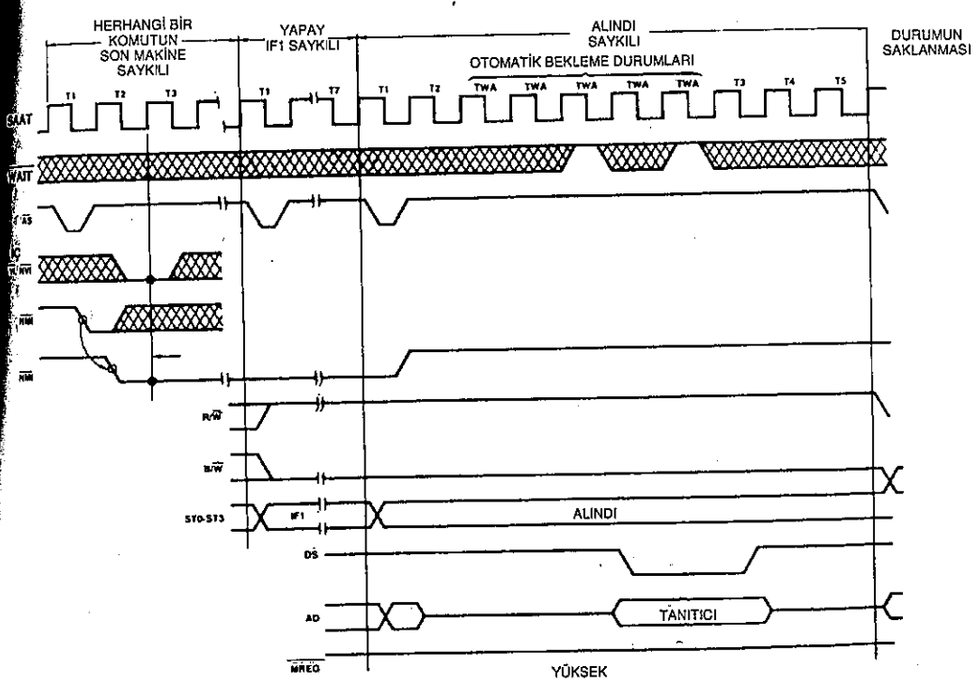
[illegible]de yazma işlemini ele alalım. R/W çıkışı ALÇAK [illegible]de olacaktır. AmZ8002, adresi yoldan çeker ve yola [illegible]miş veriyi dışarıya kapılayarak T2 süresince DS çıkışını [illegible]ÇAK düzeyde etkin hale getirir. Eğer veri baytı yazılmış [illegible] aynı bayt yolun her iki yarısında da ortaya çıkacaktır. DS [illegible]şi, saykılın bitimini belirten T3 süresince YÜKSEK düze[illegible]gidecektir.

Kesme Alındısı:

Kesme alındısına ayrılmış üç adet durum hattı kodu vardır. Bunlar; maskelenmeyen, vektörlenmiş ve vektörlenmemiş kesmeye karşılık gelir. Şekil 8'de kesme alındı saykılı gösterilmiştir. AmZ8002'nin NMI girişi kenara-duyarlıdır, yani giriş düzeyinin YÜKSEK'ten ALÇAK'a geçişi, iç mandala saklanır. Benzer iç saklama, VI ve NVI girişleri için sağlanmaz. Kesmeye neden olan VI ve NMI girişleri için FCW içindeki bu kesmeler karşılık gelen bitlerin "1" olacağı varsayılmıştır.

Şekilden de görüldüğü gibi VI girişi, NVI girişi ve iç NMI mandal çıkışı, komutun son makine saykılı olan T3 sırasında örneklenir. Bu sinyallerdeki ALÇAK düzey, bunlara karşılık gelen aşağıda sırayla açıklanmış kesme alındılarını tetikler. AmZ8002 gerçek alındı saykılından önce yapay IF1 saykılı esnasında belirtilen sistem yığın işaretçisinin (R15') azaltılmasına rağmen program sayıcısına yeni değeri yüklenmiştir. Yapay IF1 saykılını izleyen saykıl, gerçek kesme alındı saykılıdır.

Kesme alındı saykılı tipik olarak 10 saat periyodundan oluşmuştur; T1'den T5'e ve beş otomatik TW (bekleme) durumu. Her zaman olduğu gibi saykılın başlangıcını belirtmek için AS çıkışı, T1 esnasında ALÇAK düzey ile darbelenecektir. ST0-ST3 çıkışları uygun kesme alındı kodlarını yansıtacak, MREQ çıkışı YÜKSEK düzeyde olacak, N/S çıkışı bir önceki saykılda oldu gibi aynı düzeyde kalacak ve R/W çıkışı YÜKSEK ve B/W çıkışı ALÇAK düzeyde olacaktır. AmZ8002



Şekil 8. Kesme Alındı Saykılı

AD0-AD15 yolunu T1 süresince belirsiz bilgilerle sürecektir ve T2 esnasında yol, yüksek empendans durumuna geçecektir. Üç TWA durumu otomatik olarak T2'yi izleyecektir. Üçüncü TWA durumu esnasında WAIT girişi örneklenecektir.

Eğer WAIT girişi ALÇAK düzeydeyse, araya fazladan bir TW durumu eklenecektir ve WAIT girişi tekrar TW esnasında örneklenecektir. Bunun gibi TW durumlarının eklenmesi WAIT girişinin YÜKSEK düzeye çıkmasına kadar devam eder. Son TW durumundan sonra, DS çıkışı ALÇAK düzeye gidecek ve bunu iki adet daha otomatik bekleme durumu izleyecektir. DS çıkışı ALÇAK düzeyde olduğu zaman, kesmeye neden olan birim, yola 16-bitlik tanıtıcı kapılıyabilir. Son TWA drumunda WAIT girişi tekrar örneklenir. Eğer WAIT girişi ALÇAK düzeydeyse, bir TW durumu eklenecek ve WAIT girişi TW esnasında örneklenecektir. Bu gibi TW durumlarının eklenmesi WAIT girişinin YÜKSEK düzeye çıkmasına kadar devam eder. Son TW durumu tamamlandıktan sonra, T3'e girilecek ve DS çıkışı YÜKSEK düzeye gidecektir. Kesme birimi, tanıtıcıyı yoldan çekmeli ve yolu sürmeyi bırakmalıdır. Saykılı tamamlamak için T3'ün ardından T4 ve T5 durumları gelecektir. Kesme alındı saykılını takip edecek saykıl, durumu, yığında saklamak için kullanılacak olan bellek işlem saykılı olacaktır. Durum bilgisinin sakl[illegible] sında N/S çıkışının otomatik olarak ALÇAK düz[illegible] ğına dikkat ediniz.

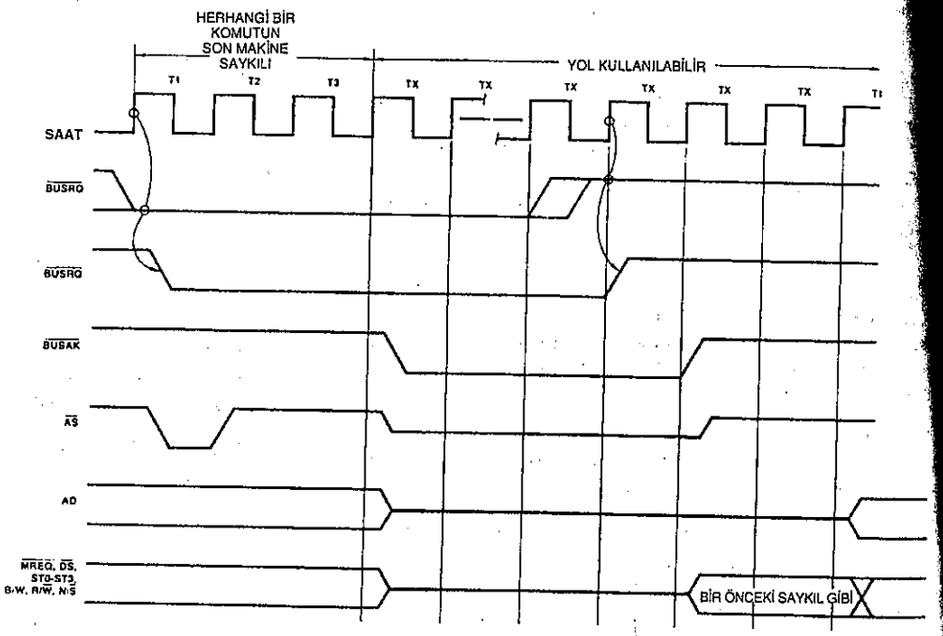
Kesme alındı saykılındaki AS çıkışının YÜKSEK[illegible] gittiği başlangıç durumunda, iç NMI mandalı sıfır[illegible] Bu zamana kadar VI ve NVI girişleri de ALÇAK [illegible] tutulmalıdır.

Durum Saklama Sırası:

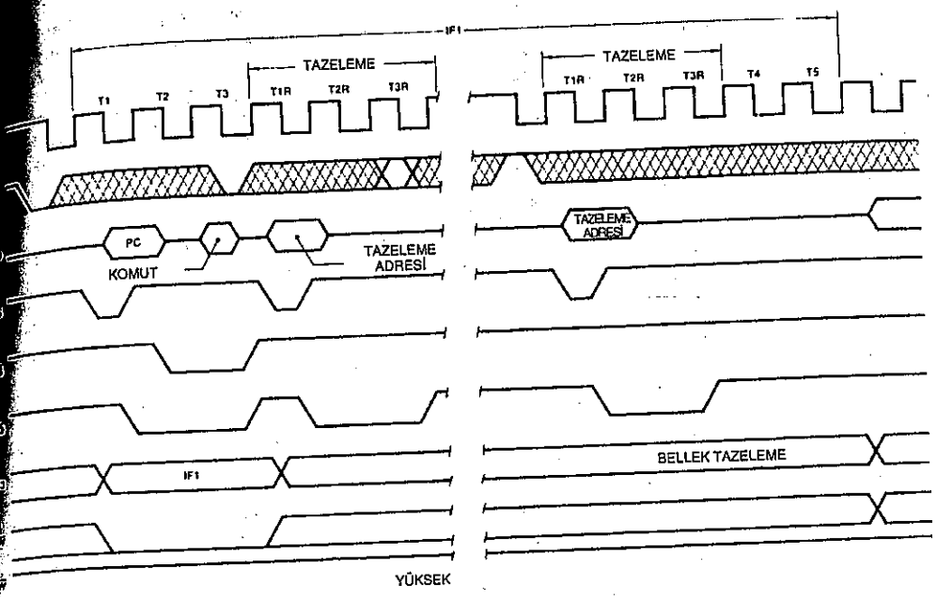
Kesme alındı saykılını takip eden makine saykılı, [illegible] bilgisini aşağıdaki sırada sistem yığının üzerine [illegible] program sayıcısı, bayrak ve denetim sözcüğü ve [illegible] da kesme/tuzak tanıtıcısı. Bir sonraki makine sayk[illegible] program durum alanından yeni program durumunu [illegible] ardından kesme/hizmet altyordamına dallanır.

YOL İSTEĞİ/YOL ALINDI ZAMANLAMASI:

BUSRQ girişindeki ALÇAK düzey, AmZ8002'ye (DMA [illegible] diğer birimlerin yolun denetimini istediklerini bildirir. BUS[illegible] girişi, herhangi bir makine saykılının T1 anında, dahili [illegible] eşzamanlanmıştır (farklılıklar için aşağıya bakın). BUS[illegible] çıkışı, makine saykılının son saat peryodundan sonra AL[illegible] ÇAK düzeye gidecektir. BUSAK çıkışındaki ALÇAK düz[illegible]



Şekil 9. Yol İstek/Alındı Saykılı



Şekil 10. Tek adım Zamanlaması

[illegible] alındıya işaret eder. BUSAK çıkışı ALÇAK olduğu zaman; AD0-ADM15, AS, DS, MREQ, ST0-ST3, B/W, R/W ve N/S çıkışları yüksek empedans durumuna geçer. Bütün işlemler tamamlanıncaya kadar BUSRQ girişi ALÇAK düzeyde tutulmalıdır. BUSRQ YÜKSEK düzeye gittiği zaman, dahili olarak eşzamanlanmıştır. BUSAK çıkışı YÜKSEK düzeye gidecek ve normal MİB işlemleri yeniden başlayacaktır. Şekil 9'da, BUSRQ /BUSRQ zamanlaması gösterilmiştir.

BUSRQ girişinin, biri dışında herhangi bir makine saykılında dikkate alınacağından söz edilmişti. Bu istisnai durum, TSET/TSETB komutlarının yürütülmesi dışında ortaya çıkar. Bu komutun yürütümü bir kez başladığında, BUSRQ girişi dikkate alınmayacaktır.

TEK ADIMLAMA

AmZ8002'nin STOP girişi, bir komutun tek bir anda veya tek adımda yürütülmesini sağlar. Şekil 10'da STOP girişinin zamanlaması gösterilmiştir. STOP girişi, IF1 saykılının hemen önündeki saatin YÜKSEK'ten ALÇAK'a geçişinde örneklenir. Eğer STOP girişinde ALÇAK düzey olarak algılanırsa, AmZ8002, T3'den sonra bellek tazeleme saykılı üretecektir. STOP girişi, ayrıca, tekrar T3'te örneklenecektir. Eğer STOP girişi ALÇAK düzeydeyse, bir veya daha fazla sayıda saykıl, daha önceki tazeleme saykılını izleyecektir. STOP, tazeleme saykılının T3 süresi boyunca tekrar örneklenecektir. STOP girişinde ALÇAK düzey örneklendiği süre-ce bu ek tazeleme saykılı her zaman için eklenecektir. STOP girişinin YÜKSEK düzeyinden sonra oluşacak son tazeleme saykılının tamamlanmasıyla, MİB, IF1 saykılını tamamlamak için iki adet sahte (dummy) T4 ve T5 durumlarını ekleyecektir ve komutu yürütmek için yeniden normal işlemlerine başlayacaktır. Bellek işlemleri ve bellek tazeleme konularındaki ilgili bölümlere bakın.

Tazeleme saykılının, tazeleme kaydedicisindeki tazeleme olanağı yetkisizlenmiş olsa bile STOP modunda gerçekleşeceğine dikakt ediniz.

ÇOKLU MİKROİŞLEMCİ OLANAKLARI

AmZ8002, çoklumikroişlemcili sistemleri desteklemek için donanım ve yazılım yetenekleriyle donatılmıştır. AmZ8002'nin μO ve μI sinyalleri, bu amaç için MBIT, MREQ, MRES ve MSET komutlarıyla birlikte kullanılır. μO çıkışı, AmZ8002'den, bir kaynak kullanımına ilişkin istekte bulunmak üzere, ilgili komut kullanılarak ALÇAK'ta etkin hale getirilebilir. μO çıkışı etkin hale gelmeden önce I girişi, AmZ8002 tarafından test edilir. Bu anda μI girişinde bulunan ALÇAK düzey, istek yapılan kaynağın meşgul olduğunu gösterir. AmZ8002, μO çıkışı ALÇAK düzeyde etkin hale geldikten sonra μI girişini kontrol eder. Eğer istek yapılan kaynak kullanılabilir hale gelmişse, μI girişi tekrar test edilecektir. Çoklumikroişlemci olanakları hakkında ayrıntılı bilgi için İşlemci Arabirim Elkitabına bakılmalıdır (Yayın No: AM-PUB089).

MİB'İN BAŞLATILMASI

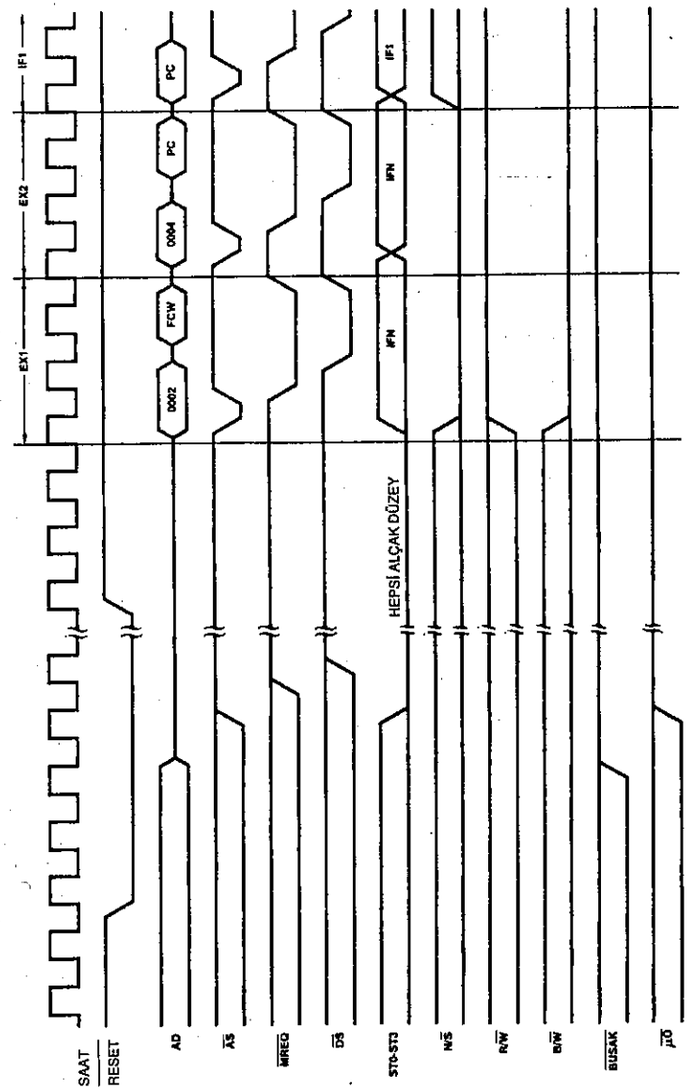
$\overline{\text{Reset}}$ girişindeki ALÇAK düzey, MİB'in başlatılmasına olanak verir. Başlatma sırası şekil 11'de gösterilmiştir. $\overline{\text{Reset}}$ girişinin YÜKSEK'ten ALÇAK'a geçişinden sonra beş saat peryodu boyunca aşağıdakiler ortaya çıkacaktır:

a) AD0-AD15 yolu YÜKSEK empedans durumuna geçecektir.
b) $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{MREQ}}$, $\overline{\text{BUSAK}}$ ve $\overline{\mu\text{O}}$ çıkışları YÜKSEK düzeyde olacaktır.
c) ST0-ST3 çıkışları ALÇAK düzeyde olacaktır.
d) Tazeleme yetkisizlenecektir.
e) R/$\overline{\text{W}}$, B/$\overline{\text{W}}$ ve N/$\overline{\text{S}}$ çıkışları etkilenmez açılışta-sıfırlama için bu çıkışların durumu belirtilmemiştir.

Reset girişi YÜKSEK düzeye döndükten ve üç saat peryodu boyunca bu düzeyde kaldıktan sonra, iki 16-bitlik bellek okuma işlemi aşağıdaki gibi yürütülecektir. N/S çıkışının ALÇAK düzeyde olacağına ve ST0-ST3 çıkışlarının da IFN kodunu yansıtacağına dikkate ediniz.

a) 0002 adresli bellek konumunun içeriği okunacaktır. Bu bilgi AmZ8002'nin program sayıcısına yüklenecektir.
b) 0004 adresli bellek konumunun içeriği okunacaktır. Bu bilgi AmZ8002'nin program sayıcısına yüklenecektir.

Bu işlemler başlatma sırasını tamamlar ve IF1 saykılı, programın yürütümüne başlanması için getirilen ilk komutu takip edecektir. Zamanlama için bellek işlemleri bölümüne bakın.



Şekil 11. Sıfırlama Sırası

AmZ8002 KOMUT TAKIMI

YÜKLEME VE DEĞİŞ-TOKUŞ

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
CLR CLRB	vrşyeri	R IR DA X	Sil vrşyeri ← 0
EX EXB	KYD., kayn	R IR DA X	Değiş-tokuş KYD. ← kayn.
LD LDB LDL	KYD., kayn	R IM IM IR DA X BA BX	Kaydediciye Yükle KYD. ← kayn.
LD LBD LDL	vrşyeri,KYD.	IR DA X BA BX	Belleğe Yükle (sakla) kayn. ← KYD.
LD LDB	vrşyeri, IM	IR DA X	Belleğe Dolaysız Yükle vrşyeri ← IM
LDA	KYD., kayn	DA X BA BX	Adresi yükle KYD. ← kaynak adresi
LDAR	KYD., kayn	RA	Adresi Bağıl Olarak Yükle KYD. ← kaynak adresi
LDK	KYD., kayn	IM	Sabiti Yükle KYD. ← n (n = 0 ... 15)
LDM	KYD., kayn, n	IR DA X	Çoklu Yükle KYD. ← kayn. (n tane ardışık sözcük) (n = 1 ... 16)
LDM	vrşyeri, KYD., n	IR DA X	Çoklu Yükle (Çoklu Sakla) vrşyeri ← KYD. (n tane ardışık sözcük) (n = 1 ... 16)
LDR LDRB LDRL	KYD., kayn	RA	Bağıl Yükle KYD. ← kayn. (aralık -32768 ... +32767)
LDR LDRB LDRL	vrşyeri, KYD.	RA	Bağıl Yükle (Bağıl Sakla) vrşyeri ← KYD. (aralık -32768 ... +32767)
POP POPL	vrşyeri, KYD.	R IR DA X	Çek vrşyeri ← Kom. KYD. KYD. içeriğinin otomatik artımı
PUSH PUSHL	IR, kayn	R IM IR DA X	İt KYD. içeriğinin otomatik azalımı Komut KYD. ← kayn.

Kullanılan kısaltmalar:
kayn. (kaynak): src (source)
KYD. (kaydedici): R (register)
kk (koşul kodu): CC (Condition Code)
vrşyeri (varış yeri): dst (destination)

ARİTMETİK

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
ADC ADCB	KYD., kayn	R	Eldeli Topla KYD ← KYD + kayn + elde
ADD ADDB ADDL	KYD., kayn	R IM IR DA X	Topla KYD ← KYD + kayn
CP CPB CPL	KYD., kayn	R IM IR DA X	Kaydediciyle Karşılaştır KYD – kayn
CP CPB	vrşyeri, IM	IR DA X	Dolaysız Karşılaştır vrşyeri - IM
DAB	vrşyeri	R	Ondalık ayar
DEC DECB	vrşyeri, n	R IR DA X	n kadar azalt vrşyeri ← vrşyeri - n (n = 1 ... 16)
DIV DIVL	KYD., kayn	R IM IR DA X	Böl (İşaretli) Sözcük: KYDn+1 ← KYDn,n+1 kayn KYDn ← kalan Uzun sözcük: KYDn+2,n+3 ← KYDn ... n+3 kayn KYDn,n+1 ← kalan
EXTS EXTSB EXTLS	vrşyeri	R	İşareti Uzat vrşyeri'nin alt-sıralı yarısının işaretini vrşyeri'nin üst-sıralı yarısına kadar uzat
INC INCB	vrşyeri, n	R IR DA X	n kadar azalt vrşyeri ← vrşyeri + n (n = 1 ... 16)
MULT MULTL	KYD., kayn	R IM IR DA X	Çarp (İşaretli) Sözcük: KYD n,n+1 ← KYDn+1 * kayn Uzun sözcük: KYDn...n+3 ← KYDn+2, n+3 * kayn * artı çarpılan içindeki her "1" için yedi saykıl
NEG NEGB	vrşyeri	R IR DA X	Negatif yap vrşyeri ← 0 - vrşyeri
SBC SBCB	KYD., kayn	R	Eldeli Çıkar KYD ← KYD - kayn - elde
SUB SUBB SUBL	KYD., kayn	R IM IR DA X	Çıkar KYD ← KYD - kayn

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
[illegible]	KYD., kayn	R IM IR DA X	VE'le KYD ← KYD VE kayn
[illegible]	vrşyeri	R IM IR DA X	Tümleyenini al vrşyeri ← DEĞİL vrşyeri
[illegible]	KYD., kayn	R IM IR DA X	VEYA'la KYD ← KYD VEYA kayn
[illegible]	vrşyeri	R IR DA X	TEST et vrşyeri VEYA 0
TCC TCCB	cc, vrşyeri	R	Test Koşul Kodu Eğer cc (koşul kodu) doğruysa EKDB'i kur
XOR XORB	KYD., kayn	R IM IR DA X	Özel VEYA'la KYD ← KYD özel VEYA kayn

PROGRAM DENETİMİ

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
CALL	vrşyeri	IR DA. X	Altyordamı Çağır SP otomatik azalımı @SP ← PC PC ← vrşyeri
CALR	vrşyeri	RA	Bağıl Çağır SP otomatik azalımı @SP ← PC PC ← PC + vrşyeri (aralık -4094'den +4096'ya)
DJNZ DBJNZ	KYD., vrşyeri	RA	Sıfırdan-Farklıysa Azalt ve Atla KYD ← KYD - 1 KYD = 0: PC ← PC vrşyeri (aralık -254'den 0'a)
IRET*	-	-	Kesmeden Geriye Dön PS ← @SP SP Otomatik Artımı
JP	kk, vrşyeri	IR IR DA X	Koşullu Atla Eğer cc (koşul kodu) doğruysa: PC ← vrşyeri
JR	kk, vrşyeri	RA	Koşullu Bağıla Atla Eğer cc (koşul kodu) doğruysa: PC ← PC + vrşyeri (aralık -256'dan +254'e)
RET	kk	-	Koşullu Geriye Dön Eğer cc (koşul kodu) doğruysa: PC ← @SP SP Otomatik Azalımı
SC	kayn	IM	Sistem Çağrısı SP Otomatik Azalımı P değeri Komutu it PS ← Sistem Çağrısı PS

* Ayrıcalıklı komutlar. Yalnızca sistem modunda yürütülür.

BİT İŞLEME

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
BIT BITB	vrşyeri, b	R IR DA X	Biti Statik Olarak Test Et Z bayrağı ← b ile belirtilen vrşyeri bitinin DEĞİLİ
BIT BITB	vrşyeri, R	R	Biti Dinamik Olarak Test Et Z bayrağı ← KYD'nin içeriği ile belirtilen vrşyeri bitinin DEĞİLİ
RES RESB	vrşyeri, b	R IR DA X	Biti Statik Olarak Sıfırla b ile belirtilen vrşyeri bitini sıfırla
RES RESB	vrşyeri, R	R	Biti Dinamik Olarak Sıfırla KYD'nin içeriği ile belirtilen vrşyeri bitini sıfırla
SET SETB	vrşyeri, b	R IR DA X	Biti Statik Olarak Kur. KYD'nin içeriği ile belirtilen vrşyeri bitini kur
SET SETB	vrşyeri, R	R	Biti Dinamik Olarak Kur. KYD'nin içeriği ile belirtilen vrşyeri bitini kur
TSET TSETB	vrşyeri	R IR DA X	Test Et ve Kur S bayrağı ← vrşyeri'nin EBDB'i vrşyeri ← tüm 1'ler

DÖNDÜRME VE KAYDIRMA

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
RLDB	KYD, kayn	R	Basamağı Sola Döndür
RRDB	KYD, kayn	R	Basamağı Sağa Döndür
RL RLB	vrşyeri, n	R R	n bit Sola Döndür (n = 1,2)
RLC RLCB	vrşyeri, n	R R	n biti elde ile birlikte Sola Döndür (n = 1,2)
RR RRB	vrşyeri, n	R R	n bit Sağa Döndür (n = 1, 2)
RRC RRCB	vrşyeri, n	R R	n biti eldeli sağa döndür (n = 1, 2)
SDA SDAB SDAL	vrşyeri, R	R	Aritmetik Olarak Dinamik Kaydır vrşyeri'nl KYD'nin içeriği kadar sola veya sağa kaydır
SDL SDLB SDLL	vrşyeri, R	R	Mantıksal Olarak Dinamik Kaydır vrşyeri'nl KYD'nin içeriği kadar sola veya sağa kaydır
SLA SLAB SLAL	vrşyeri, n	R	n bit kadar Aritmetik Sola Kaydır
SLL SLLB SLLL	vrşyeri, n	R	n bit kadar Mantıksal Sola Kaydır
SRA SRAB SRAL	vrşyeri, n	R	n bit kadar Aritmetik Sağa Kaydır
SRL SRLB SRLL	vrşyeri, n	R	n bit kadar Mantıksal Sağa Kaydır

BLOK AKTARMA VE DİZİ İŞLEME

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
CPD CPDB	KYD_X, kayn KYD_Y, kk	IR	Karşılaştır ve Azalt KYD_x – kayn kayn adresi otomatik olarak azalır $R_Y \leftarrow R_Y - 1$
CPDR CPDRB	KYD_X, kayn KYD_Y, kk	IR	Karşılaştır, Azalt ve Tekrarla R_x – kayn kayn adresi otomatik olarak azalır $KYD_x \leftarrow KYD_Y - 1$ cc (koşul kodu) doğru veya KYDY = 0 olana kadar tekrarla
CPI CPIB	KYD_X, kayn KYD_Y, kk	IR	Karşılaştır ve Artır KYD_x – kayn kayn adresi otomatik olarak artar $KYD_Y \leftarrow KYD_Y - 1$
CPIR CPIRB	KYD_X, kayn KYD_Y, kk	IR	Karşılaştır, Artır ve Tekrarla KYD_X - kayn kayn adresi otomatik olarak artar $KYD_Y \leftarrow KYD_Y - 1$ cc (koşul kodu) doğru veya KYDY = 0
CPSD CPSDB	vrşyeri, kayn KYD., kk	IR	Diziyi Karşılaştır ve Azalt vrşyeri - kayn vrşyeri ve kayn adresleri otomatik olarak azalır KYD ← KYD – 1
CPSDR CPSDRB	vrşyeri, kayn KYD, kk	IR	Diziyi Karşılaştır, Azalt ve Tekrarla vrşyeri - kayn vrşyeri ve kayn adresleri otomatik olarak azalır KYD ← KYD – 1 cc (koşul kodu) doğru veya $KYD_Y = 0$
CPSI CPSIB	vrşyeri, kayn KYD, kk	IR	Diziyi Karşılaştır ve Artır vrşyeri - kayn vrşyeri ve kayn adresleri otomatik olarak artar KYD ← KYD – 1
CPSIR CPSIRB	vrşyeri, kayn KYD, kk	IR	Diziyi karşılaştır, Artır ve Tekrarla vrşyeri - kayn vrşyeri ve kayn adresleri otomatik olarak artar KYD ← KYD – 1 cc (koşul kodu) doğru veya KYDY = 0
LDD LDDB	vrşyeri, kayn KYD.	IR	Yükle ve Azalt vrşyeri ← kayn vrşyeri ve kayn adresleri otomatik olarak azalır KYD ← KYD – 1
LDDR LDDRB	vrşyeri, kayn KYD.	IR	Yükle, Azalt ve Tekrarla vrşyeri ← kayn vrşyeri ve kayn adresleri otomatik olarak azalır KYD ← KYD – 1 KYD = 0 olana kadar tekrarla

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
LDI LDIB	vrşyeri, kayn KYD.	IR	Yükle ve Artır, vrşyeri ← kayn vrşyeri ve kayn adresleri otomatik olarak artar KYD ← KYD – 1
LDIR LDIRB	vrşyeri, kayn KYD.	IR	Yükle, Artır ve Tekrarla vrşyeri ← kayn vrşyeri ve kayn adresleri otomatik olarak artar KYD ← KYD – 1 KYD = 0 olana kadar tekrarla
TRDB	vrşyeri, kayn KYD.		Döndür ve Azalt vrşyeri ← kayn (vrşyeri) vrşyeri adresi otomatik olarak azalır KYD ← KYD – 1
TRDRB	vrşyeri, kayn KYD.	IR	Döndür, Azalt ve Tekrarla vrşyeri ← kayn (vrşyeri) vrşyeri adresi otomatik olarak azalır KYD ← KYD – 1 KYD = 0 olana kadar tekrarla
TRIB	vrşyeri, kayn KYD.	IR	Döndür ve Artır vrşyeri ← kayn (vrşyeri) vrşyeri adresi otomatik olarak artar KYD ← KYD – 1
TRIRB	vrşyeri, kayn KYD.	IR	Döndür, Artır ve Tekrarla vrşyeri ← kayn (vrşyeri) vrşyeri adresi otomatik olarak artar KYD ← KYD – 1 KYD = 0 olana kadar tekrarla
TRTDB	kayn 1, kayn 2, KYD.	IR	Döndür ve Test et, Azalt KYDH1 ← kayn 2 (kayn 1) kayn 1 adresi otomatik olarak azalır KYD ← KYD – 1
TRTDRB	kayn 1, kayn 2, KYD.	IR	Döndür veya KYDH1 = 0 olana kadar tekrarla KYDH1 ← kayn 2 (kayn 1) kayn 1 adresi otomatik olarak azalır KYD ← KYD – 1 KYD = 0 veya KYDH1 = 0 olana kadar tekrarla
TRTIB	kayn 1, kayn 2, KYD.	IR	Döndür ve Test Et, Artır KYDH1 ← kayn 2 (kayn 1) kayn 1 adresi otomatik olarak artar KYD ← KYD – 1
TRTIRB	kayn 1, kayn 2, KYD.	IR	Döndür ve Test Et, Artır ve Tekrarla KYDH1 ← kayn 2 (kayn 1) kayn 1 adresi otomatik olarak artar KYD ← KYD – 1 KYD = 0 veya KYDH1 = 0 olana kadar tekrarla

GİRİŞ/ÇIKIŞ

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
IN* INB*	KYD., kayn	IR	Giriş KYD ← kayn
IND* INDB*	vrşyeri, kayn KYD.,	IR	Giriş Yap ve Azalt vrşyeri ← kayn vrşyeri adresi otomatik olarak azalır KYD ← KYD – 1
INDR* INDRB*	vrşyeri, kayn KYD.,	IR	Giriş Yap, Azalt ve Tekrarla vrşyeri ← kayn vrşyeri adresi otomatik olarak azalır KYD ← KYD – 1 KYD = 0 olana kadar tekrarla
INI* INIB*	vrşyeri, kayn KYD.,	IR	Giriş Yap ve Artır vrşyeri ← kayn vrşyeri adresi otomatik olarak artar KYD ← KYD – 1
INIR* INIRB*	vrşyeri, kayn KYD.,	IR	Giriş Yap, Artır ve Tekrarla vrşyeri ← kayn vrşyeri adresi otomatik olarak artar KYD ← KYD – 1 KYD = 0 olana kadar tekrarla
OUT* OUTB*	vrşyeri, KYD.	IR	Çıkış vrşyeri ← KYD
OUTD* OUTDB*	vrşyeri, kayn KYD.,	IR	Çıkış Yap ve Azalt vrşyeri ← kayn kayn adresi otomatik olarak azalır KYD ← KYD – 1
OTDR* OTDRB*	vrşyeri, kayn KYD.,	IR	Çıkış Yap, Azalt ve Tekrarla vrşyeri ← kayn kayn adresi otomatik olarak azalır KYD ← KYD – 1 KYD=0 olana kadar tekrarla
OUTI* OUTIB*	vrşyeri, kayn KYD.,	IR	Çıkış Yap ve Artır vrşyeri ← kayn kayn adresi otomatik arten KYD ← KYD – 1
OTIR* OTIRB*	vrşyeri, kayn KYD.,	IR	Çıkış Yap, Artır ve Tekrarla vrşyeri ← kayn kayn adresi otomatik olarak artar KYD ← KYD – 1 KYD = 0 olana kadar tekrarla
SIN* SINB*	KYD., kayn	IR	Özel Giriş KYD ← kayn
SIND* SINDB*		IR	Özel Giriş ve Azalt vrşyeri ← kayn vrşyeri adresi otomatik olarak azalır KYD ← KYD - 1
SINDR* SINDRB*	vrşyeri, kayn KYD.,	IR	Özel giriş, Azalt ve Tekrarla vrşyeri ← kayn vrşyeri adresi otomatik olarak azalır KYD ← KYD – 1 KYD = 0 olana kadar tekrarla
SINI* SINIB*	vrşyeri, kayn KYD.,	IR	Özel Giriş ve Artır vrşyeri ← kayn vrşyeri adresi otomatik olarak artar KYD ← KYD – 1
SINIR* SINIRB*	vrşyeri, kayn KYD.,	IR	Özel giriş, Artır ve Tekrarla vrşyeri ← kayn vrşyeri adresi otomatik olarak artar KYD ← KYD – 1 KYD = 0 olana kadar tekrarla

* Ayrıcalıklı komutlar. Yalnızca sistem modunda yürütülür.

GİRİŞ/ÇIKIŞ (devamı)

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
SOUT* SOUTB*	vrşyeri, kayn	IR	Özel Çıkış vrşyeri ← kayn
SOUTD* SOUTDB*	vrşyeri, kayn KYD.	IR	Özel Çıkış ve Azalt vrşyeri ← kayn kayn adresi otomatik olarak azalır KYD ← KYD – 1
SOTDR* SOTDRB*	vrşyeri, kayn KYD.	IR	Özel Çıkış, Azalt ve Tekrarla vrşyeri ← kayn kayn adresi otomatik olarak azalır KYD ← KYD – 1 KYD = 0 olana kadar tekrarla
SOUTI* SOUTIB*	vrşyeri, kayn KYD.	IR	Özel Çıkış ve Artır vrşyeri ← kayn kayn adresi otomatik olarak artar KYD ← KYD – 1
SOTIR* SOTIRB*	vrşyeri, kayn KYD.	IR	Özel Çıkış, Artır ve Tekrarla vrşyeri ← kayn kayn adresi otomatik olarak artar KYD ← KYD – 1 KYD = 0 olana kadar tekrarla

MİB DENETİMİ

Komut Kısalt.	İşlenenler	Adresl. Modl.	İşlem
COMFLG	flags	-	Bayrakların Tümleyenini Al (C,Z,S, P/V'nin herhangi bileşimi)
DI*	int	-	Kesmeyi Yetkisizle (NVI, VI'nın herhangi bileşimi)
EI*	int	-	Kesmeyi Yetkile (NVI, VI'nın herhangi bileşimi)
HALT*	-	-	DURDURMA
LDCTL*	CTRL., kayn	R	Denetim kaydedicisine yükle DNTKYD ← kayn
LDCTL*	vrşyeri, CTRL.	R	Denetim kaydedicisinden yükle vrşyeri ← DNTKYD
LDCTLB	FLGR, kayn	R	Bayrak bayt kaydedicisine yükle BYRKYD ← kayn
LDCTLB	vrşyeri FLGR	R	Bayrak bayt kaydedicisinden yükle vrşyeri ← BYRKYD
LDPS*	kayn	IR DA X	Program Durumunu Yükle PS ← kayn
MBIT*	-	-	Çoklu-Mikro Bitini Test Et Eğer µI ALÇAK düzeyde ise S1 kur, µI YÜKSEK düzeyde ise S1 sıfırla.
MREQ*	vrşyeri	R	Çoklu-Mikro İsteği
MRES*	-	-	Çoklu-Mikro Sıfırlama
MSET*	-	-	Çoklu-Mikro Kurma
NOP	-	-	İşlem Yok
RESFLG	bayrak	-	Bayrağı Sıfırla (C,Z,S,P/V'nin herhangi bileşimi)
SETFLG	bayrak	-	Bayrağı Kur (C,Z,S,P/V'nin herhangi bileşimi

MAKSİMUM ANMA DEĞERLER'ni aşan değerlerde çalışma, kullanım ömrünün azalmasına neden olab

Toprağa göre bütün giriş ve çıkış gerilimleri	
Gerilim altında ortam sıcaklığı	
Saklama sıcaklığı	-65

Bu spesifikasyonlarla tanımlanan ürünler, statik giriş birimlerinin, yük birikimi sonucu zarar görmesini önleyecek biçim tasarlanmış iç devreler içerir. Bununla beraber, ortaya çıkacak aşırı gerilimlerden korunmak için deplomada ve kullan klasik önlemlerin alınması tavsiye edilmiştir.

ELEKTRİKSEL KARAKTERİSTİKLERİ çalışma aralığında (Not 1)

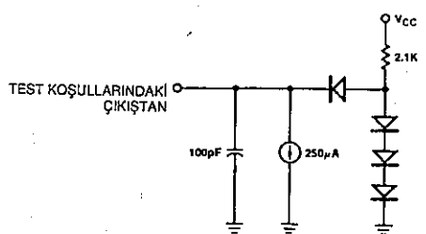
Parametre	Tanımı	Test Koşulları	AmZ8002DC Min.	Maks.	Birimi
V_{CH}	Saat Girişi Yüksek Düzey Gerilimi	Dış Saat Üreteci ile sürülür	V_{CC} - 0.4	V_{CC} + 0.3	V
V_{CL}	Saat Girişi Düşük Düzey Gerilimi	Dış Saat Üreteci ile sürülür	-0.3	0.45	V
V_{IH}	Giriş Yüksek Düzey Gerilimi		2.0	V_{CC} + 0.3	V
V_{IL}	Giriş Düşük Düzey Gerilimi		- 0.3	0.8	V
V_{OH}	Çıkış Yüksek Düzey Gerilimi	I_{OH} = - 250 µA	2.4		V
V_{OL}	Çıkış Düşük Düzey Gerilimi	I_{OL} = + 2.0 mA		0.4	V
I_{IL}	Giriş Sızıntı Akımı	$0.4 \le V_{GIR} \le +2.4V$		± 10	µA
I_{OL}	Çıkış Sızıntı Akımı	$0.4 \le V_{ÇIK} \le +2.4V$		± 10	µA
I_{CC}	V_{cc} Kaynak Akımı			300	mA

Not 1: Tipik Değerler T_A = 25 °C, nominal kaynak gerilimleri ve nominal işlem parametrelesi içindir.

Standart Test Koşulları

Karakteristikler, aksi belirtilmedikçe aşağıdaki standart test koşulları için uygulanır. Bütün gerilimler toprağı referans almıştır. Pozitif akım, referans alınan bacaktan içeri doğru akar. Standart koşullar aşağıdaki gibidir:

+4.75 V ≤ Vcc ≤ + 5.25 V
Toprak (GND) = 0 V
0 °C ≤ T_A ≤ + 70 °C



MOS-238

Bütün AC prametreler, maksimum 100 pF'lik yük kapasitesi olduğu varsayar. İki çıkış sinyali arasındaki zamanlama referansları maksimum 50 pF'lık yük kapasitesi olduğu varsayar.

ZAMANLAMA KARAKTERİSTİKLERİ çalışma aralığında

	Parametre	Tanımı	AmZ8002DC Min.	Maks.	Birimi
	TcC	Saat Saykıl Süresi	250	2000	nsn
	TwCh	Saat Genişliği (Yüksek düzey)	105	2000	nsn
	TwCl	Saat Genişliği (Alçak düzey)	105	2000	nsn
	TfC	Saat Düşme Süresi		20	nsn
	TrC	Saat Yükselme Süresi		20	nsn
	Tdc(Bz)	Saat ↑ 'dan - Yolun Yüzer Olma Durumuna		65	nsn
	TdC(A)	Saat ↑ 'dan - Adres Geçerli'ye		100	nsn
	TdC(Az)	Saat ↑ 'dan - Adresin Yüzer Olma Durumuna		65	nsn
11	TdA(DI)	Adres Geçerli'den - İstenen Veri Girişi Geçerli'ye	400		nsn
12	TsDI(C)	Giriş Verisinden - Saat ↓'a Kurulma Süresine	70		nsn
13	TdDS(A)	$\overline{DS}$ ↑ 'dan - Adresin Aktif Olmasına	80		nsn
14	TdC(DO)	Saat ↑ 'dan - Veri Çıkış Geçerli'ye		100	nsn
15	ThDI(DS)	Veri Girişten - $\overline{DS}$ ↑ 'a Kurulma Süresine	0		nsn
16	TdDO(DSW)	Veri Çıkış Geçerli'den - $\overline{DS}$ ↑ 'a Gecikme	230		nsn
17	TdA(MR)	Adres Geçerli'den - $\overline{MREQ}$ ↓'a Gecikme	55		nsn
18	TdC(MR)	Saat ↓ 'den - $\overline{MREQ}$ ↓'a Gecikme		80	nsn
19	TdC(MR)	$\overline{MREQ}$ Genişliği (Yüksek düzey)	190		nsn
20	TdMR(A)	$\overline{MREQ}$ ↓'dan - Adres Aktif Değil'e	70		nsn
21	TdDO(DSW)	Veri Çıkış Geçerli'den - $\overline{DS}$ ↓'a (Yazma) Gecikme	55		nsn
22	TdMR (DI)	$\overline{MREQ}$ ↓'dan - İstenen Veri Girişi Geçerli'ye	330		nsn
23	TdC(MR)	Saat ↓'dan - $\overline{MREQ}$ ↑'a Gecikme		80	nsn
24	TdC(ASf)	Saat ↓'dan - $\overline{AS}$ ↑'a Gecikme		80	nsn
25	TdA(AS)	Adres Geçerli'den - $\overline{AS}$ ↑'a Gecikme	55		nsn
26	TdC(ASr)	Saat ↓'dan - $\overline{AS}$ ↑'a Gecikme		90	nsn
27	TdAS(DI)	$\overline{AS}$ ↑'dan - İstenen Veri Girişi Geçerli'ye	290		nsn
28	TdDS(AS)	$\overline{DS}$ ↑'dan - AS ↓'a Gecikme	70		nsn
29	TwAS	$\overline{AS}$ Genişliği (Alçak düzey)	80		nsn
30	TDAS(A)	$\overline{AS}$ ↑'dan - Adres Aktif Değil'e Gecikme	60		nsn
31	TdAz(DSR)	Adresin Yüzer Olma Durumundan-DS (Okuma) ↓'a Gecikme	0		nsn
32	TdAS(DSR)	$\overline{AS}$ ↑'dan - $\overline{DS}$ (Okuma) ↓'a Gecikme	70		nsn
33	TdDSR(DI)	$\overline{DS}$ (Okuma) ↓'dan - İstenen Veri Giriş Geçerli'ye	155		nsn
34	TdC(DSr)	Saat ↓'dan - $\overline{DS}$ ↑'a Gecikme		70	nsn
35	TdDS(DO)	$\overline{DS}$ ↑'dan - Veri Çıkışı ve DURUM Geçerli Değil'e	80		nsn
36	TdA(DSR)	Adres Geçerli'den - $\overline{DS}$ (Okuma) ↓'a Gecikme	120		nsn
37	TdC(DSR)	Saat ↑'den - $\overline{DS}$'nin (Okuma) ↓'a Gecikme		120	nsn
38	TwDSR	$\overline{DS}$ (Okuma) Genişliği (Alçak düzey)	275		nsn
39	TdC(DSW)	Saat ↓'dan - $\overline{DS}$ (Yazma) ↓'a Gecikme		95	nsn
40	TwDSW	$\overline{DS}$ (Yazma) Genişliği (Alçak düzey)	160		nsn
41	TdDSI(DI)	$\overline{DS}$ (Giriş) ↓'den İstenen Veri Giriş Geçerliye	315		nsn

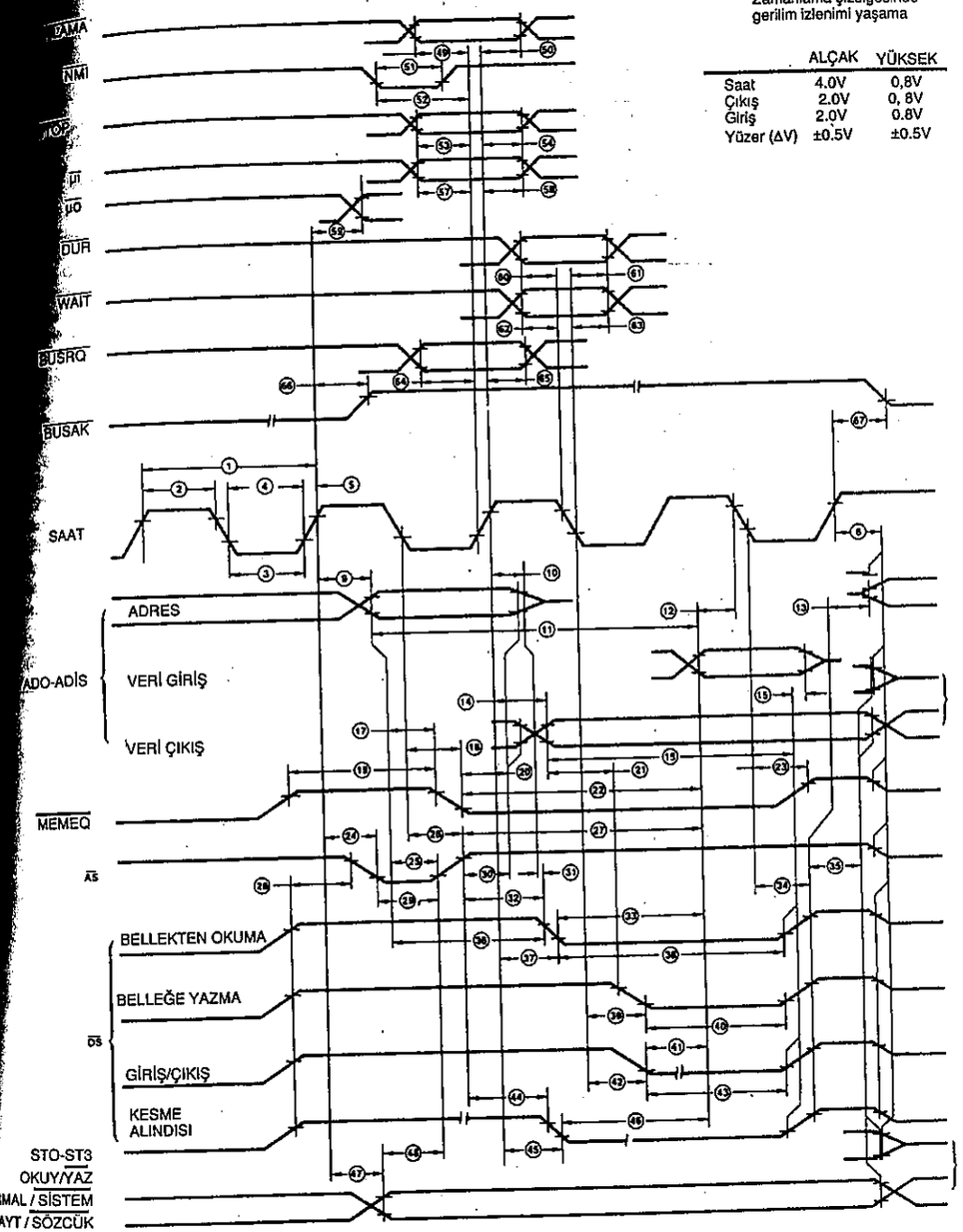
ANAHTARLAMA KARAKTERİSTİKLERİ çalışma aralığında (devamı)

Numara	Parametre	Tanımı	AmZ8002DC Min.	Maks.	
42	TdC(DSf)	Saat ↓'dan - $\overline{DS}$ (G/Ç) ↑'a Gecikme		120	[illegible]
43	TwDS	$\overline{DS}$ (G/Ç) Genişliği (Alçak düzey)	400		[illegible]
44	TdAS(DSA)	$\overline{AS}$ ↑'dan - $\overline{DS}$ (Alındı) ↓'a Gecikme	960		[illegible]
45	TdC(DSA)	Saat ↑'dan - $\overline{DS}$ (Alındı) ↓'a Gecikme		120	[illegible]
46	TdDSA(DI)	$\overline{DS}$ (Alındı) ↓'dan - İstenen Veri Girişine Gecikme	420		[illegible]
47	TdC(S)	Saat ↑'dan Durum Geçerli'ye Gecikme		110	[illegible]
48	TdS(AS)	Durum Geçerli'den - $\overline{AS}$ ↑'a Gecikme	40		[illegible]
49	TsR(C)	$\overline{RESET}$'den - Saat ↑'a Gecikme Süresi	180		[illegible]
50	ThR(C)	$\overline{RESET}$'den - Saat ↑'a Tutma Süresi	0		[illegible]
51	TwNMI	$\overline{NMI}$ Genişliği (Alçak düzey)	100		[illegible]
52	TsNMI(C)	$\overline{NMI}$'dan - Saat ↑'a Kurulma Süresi	140		[illegible]
53	TsVI(C)	$\overline{VI}$, - $\overline{NVI}$'dan Saat ↑ 'a Kurulma Süresi	110		[illegible]
54	ThVI(C)	$\overline{VI}$, $\overline{NVI}$'dan Saat ↑ 'a Tutma Süresi	0		[illegible]
55					
56					
57	Tsμi(C)	$\overline{\mu I}$'dan - Saat ↑'a Kurulma Süresi	180		nsn
58	Thμi(C)	$\overline{\mu I}$'dan - Saat ↑'a Tutma Süresi	0		nsn
59	TdC(μO)	Saat ↑'dan - $\overline{\mu O}$'a Gecikme		120	nsn
60	TsSTP(C)	$\overline{STOP}$'tan - Saat ↓'a Kurulma Süresi	140		nsn
61	ThSTP(C)	$\overline{STOP}$'tan - Saat ↓'a Tutma Süresi	0		nsn
62	TsWT(C)	$\overline{WAIT}$'den - Saat ↓'a Kurulma Süresi	70		nsn
63	ThWT(C)	$\overline{WAIT}$'den - Saat ↓'a Tutma Süresi	0		nsn
64	TsBRQ(C)	$\overline{BUSRQ}$'den - Saat ↑'a Kurulma Süresi	90		nsn
65	ThBRQ(C)	$\overline{BUSRQ}$'den - Saat ↑'a Kurulma Süresi	0		nsn
66	TdC(BAKr)	Saat ↑'dan - BUSAK ↑'a Gecikme		100	nsn
67	TdC(BAKf)	Saat ↑'dan - BUSAK ↓'a Gecikme		100	nsn

AC ZAMANLAMA DİYAGRAMI

Zamanlama çizelgesinde gerilim izlenimi yaşama

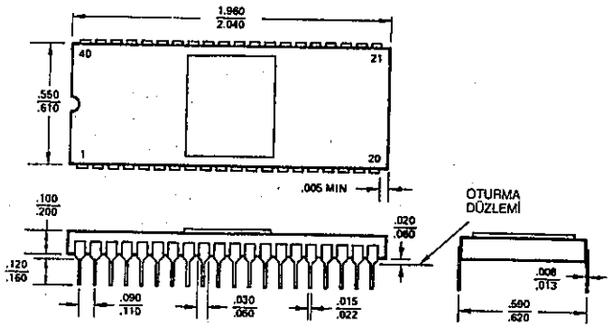
	ALÇAK	YÜKSEK
Saat	4.0V	0,8V
Çıkış	2.0V	0, 8V
Giriş	2.0V	0.8V
Yüzer (ΔV)	±0.5V	±0.5V



Bu birleşik zamanlama diyagramı, gerçek zamanlama sırasıyla gösterilmemiştir. Bu diyagram yalnızca her bir kenara ilgili ayrıntılı zamanlama ilişkileri için sunulmuştur. Çizimleri, çeşitli zamanlama sıralarını açıklamak için kullanılır.

FİZİKSEL BOYUTLAR
Çift Sıralı Paket

40-Bacaklı Seramik Paket



National Semiconductor tek-yongalı, N-kanallı mikrodenetleyiciler

ÖN BİLGİ
AĞUSTOS 1980

OP420L/COP421L ve COP320L/COP321L
k-Yongalı N-Kanallı Mikrodenetleyiciler

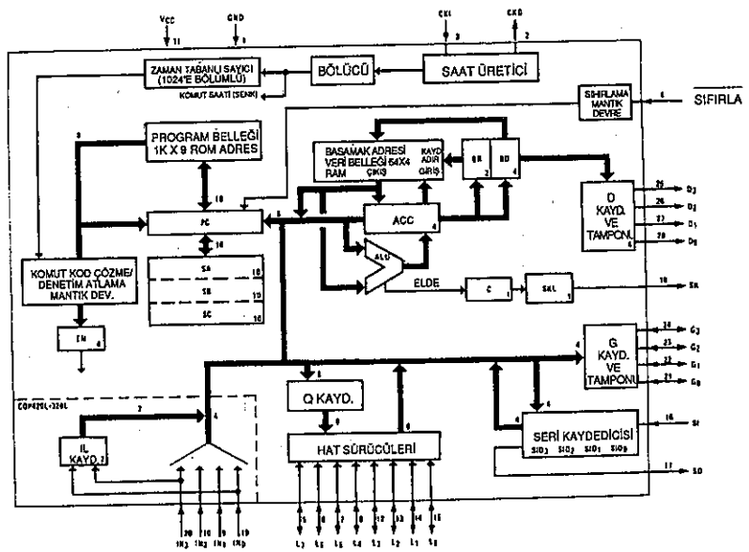
Tanımlama

20L, COP421L, COP320L ve COP321L tek-yonga N-ka-
ikrodenetleyicileri COPS™ ailesinin üyeleridir, N-kanal,
kapı MOS teknolojisi kullanarak üretilmiştir. Bu denetleyi-
elimli işlemciler, çeşitli uygulamalardaki özel denetim işlev-
yerine getirmek için gerekli olan G/Ç, RAM, ROM, iç mantık
leri ve tüm sistem zamanlamalarını içeren komple mikrobil-
arlardır. Özellikleri, tek besleme kaynağından çalışmayı,
çıkış konfigürasyon seçenekleri, komut takımı ile iç mimari
vye girişini, ekran çıkışını ve BCD veri işlemlerini kolaylaş-
üzere tasarlanmış G/Ç düzenlerini içerir. COP421L,
P420L'nin özdeşidir, ancak 23 yerine 19 G/Ç hattına sahiptir.
P421L ve COP420L, çok sayıda insanın arabirim denetim
amlarında kullanılmak üzere en uygun çözümdür. Standart
yöntemleri ve güvenilir yüksek-yoğunluklu fabrikasyon tek-
leri; uyarlanmış denetleyici-yönelimli işlemcilere sahip çok
yda müşteriye en düşük-üretim fiyatında, en uygun ortamı
ğlar.

OP320L, COP420L'nin genişletilmiş sıcaklık aralıklı versiyonu-
r. (Benzer şekilde COP321L, COP421L'nin sıcaklık aralığı
enişletilmiş tipidir). COP320L/COP321L, COP420L/COP-
21L'nin tam işlevsel eşdeğerdir.

Özellikleri

- Düşük maliyet
- Güçlü komut takımı
- 1K x 8 ROM, 64 x 4 RAM
- 23 G/Ç hattı (COP420L)
- Gerçek vektörlü kesme, ve yeniden başlatma olanağı
- Üç düzeyli altyordam yığını
- 16 µsn'lik komut süresi
- Çıkış aralığı geniş tek bir kaynak (4.5-9.5V)
- Düşük akım tüketimi (5V'ta maksimum 8mA)
- Gerçek-zamanlı işlemler için zaman-tabanlı iç sayıcı
- MICROWIRE™ uyumlu seri G/Ç'li iç İkili sayıcı/kaydedicisi
- Genel amaçlı ve ÜÇ-DURUMLU çıkışlar
- LSTTL/CMOS uyumlu giriş ve çıkış
- Doğrudan LED gösterge sürme
- COP400 ailesinin diğer üyeleriyle yazılım/donanım uyumluluğu
- Sıcaklık aralığı genişletilmiş birim COP320L/COP321L (-40 °C'den +85 °C'ye kadar)



Şekil 1. COP420L/COP421L, COP320L/COP321L Blok Diyagramı

COP420L/COP421L
Mutlak Maksimum Anma Değerleri

Toprağa Göre Bacak Gerilimleri	-0.5V ile +10V
Çalışma Ortamı Sıcaklığı	0°C ile +70 °C
Saklama Ortamı Sıcaklığı	-65°C ile +150°C
Bacak Sıcaklığı (Lehimlemede, 10 sn boyunca)	300°C
Güç Harcanımı	0.75 Wat - 25°C
	0.4 Wat - 70°C

Mutlak maksimum anma değerleri, üzerinde çalışıkması halinde elemanın zarar görebileceği sınır değerlerdir. Eleman mutlak maksimum anma değerlerinde çalışırken, DC ve AC elektriksel spesifikasyonları için garanti edilmemiştir.

DC Elektriksel Karakteristikleri 0°C ≤ T_A ≤ +70 °C, 4.5V ≤ V_{CC} ≤ 9.5V aksi belirtilmedikçe

Parametre	Koşullar	Min.	Maks.	Birimi
Standart Çalışma Gerilimi (V_{CC})	Not 1	4.5	6.3	V
İsteğe Bağlı Çalışma Gerilimi (V_{CC})		4.5	9.5	V
Çalışma Kaynak Akımı	V_{CC}=5V, (bütün giriş ve çıkışlar açık)		8	mA
Giriş Gerilim Düzeyleri				
CKI (saat) Giriş Düzeyleri				
Kristal Girişi				
Mantıksal Yüksek Düzey (V_{IH})		2.0		V
Mantıksal Alçak Düzey (V_{IL})		-0.3	0.4	V
Schmitt Tetikleyici Girişi				
Mantıksal Yüksek Düzey (V_{IH})		0.7 V_{CC}		V
Mantıksal Alçak Düzey (V_{IL})		-0.3	0.6	V
$\overline{RESET}$ Giriş Düzeyleri				
Mantıksal Yüksek Düzey		0.7 V_{CC}		V
Mantıksal Alçak Düzey		-0.3	0.6	V
$\overline{RESET}$ Histerisizliği		1.0		V
SO Giriş Düzeyi (Test Modu)		2.0	3.0	V
Diğer Bütün Girişler				
Mantıksal Yüksek Düzey	V_{CC} = Maksimum TTL trip düzeyi seçenekleri seçildiğinde, V_{CC} = 5V ± %10 yüksek tip düzey seçenekleri seçildiğinde	3.0		V
Mantıksal Yüksek Düzey		2.0		V
Mantıksal Alçak Düzey		-0.3	0.8	V
Mantıksal Yüksek Düzey		3.6		V
Mantıksal Alçak Düzey		-0.3	1.2	V
Giriş Kapasitesi			7	pF
Yüksek Emp. Giriş Kaçak Akımı		-1	+1	µA
Çıkış Gerilim Düzeyleri				
LSTTL Çalışma				
Mantıksal Yüksek Düzey (V_{OH})	V_{CC}=5V ± 5%, I_{OH}= -25 µA	2.7		V
Mantıksal Alçak Düzey (V_{OL})	I_{OL}= 0.36 mA		0.4	V
CMOS Çalışma				
Mantıksal Yüksek Düzey	I_{OH} = -10 µA	V_{CC} - 1		V
Mantıksal Alçak Düzey	I_{OL} = +10 µA		0.2	V

Not 1: Uygun çalışmayı sağlamak için V_{CC} geriliminin değişim oranı, 0.5V/msn'den daha az olmalıdır.

COP420L/COP421L
Elektriksel Karakteristikleri (devamı) 0°C ≤ T_A ≤ +70 °C, 4.5V ≤ V_{CC} ≤ 9.5V aksi belirtilmedikçe

Parametre	Koşullar	Min.	Maks.	Birimi
Akım Düzeyleri				
Çekme Akımı				
SK Çıkışları (I_{OL})	VCC = 9.5V, V_{OL} = 1.0V	4.5	22	mA
	V_{CC} = 4.5V, V_{OL} = 1.0V	2.2	11	mA
Çıkışları ve	V_{CC} = 9.5V, V_{OL} = 1.0V	2.0	9.0	mA
Standart Boyutlu G_0-G_3 ve	V_{CC} = 4.5V, V_{OL} = 1.0V	1.0	4.5	mA
D_0-D_3 Çıkışları (I_{OL})				
Yüksek Akım Seçenekli	V_{CC} = 9.5V, V_{OL} = 1.0V	15	75	mA
G_0-G_3 ve D_0-D_3 Çıkışları	V_{CC} = 4.5V, V_{OL} = 1.0V	7.0	35	mA
(I_{OL})				
Yüksek Akım Seçenekli	V_{CC} = 9.5V, V_{OL} = 1.0V	30	150	mA
G_0-G_3 ve D_0-D_3	V_{CC} = 4.5V, V_{OL} = 1.0V	15	75	mA
Çıkışları (I_{OL})				
Çıkış Kaynak Akımı				
Standart Konfigürasyon	V_{CC} = 9.5V, V_{OH} = 4.75V	-70	-450	µA
Tüm Çıkışlar (I_{OH})	V_{CC} = 4.5V, V_{OH} = 2.25V	-26	-190	µA
Push-Pull Konfigürasyonu	V_{CC} = 9.5V, V_{OH} = 4.75V	-1.45	-15.5	mA
SO ve SK Çıkışları (I_{OH})	V_{CC} = 4.5V, V_{OH} = 2.25V	-0.07	-2.8	mA
LED Konfigürasyonu	V_{CC} = 9.5V, V_{OH} = 2.0V	-1.5	-15	mA
Düşük Akım Sürücü Seçenekli	V_{CC} = 6.0V, V_{OH} = 2.0V	-1.5	-9.0	mA
L_0-L_7 Çıkışları				
(I_{OH})				
LED Konfigürasyonu	V_{CC} = 9.5V, V_{OH} = 2.0V	-3.0	-30	mA
Yüksek Akım Sürücü Seçenekli	V_{CC} = 6.0V, V_{OH} = 2.0V	-3.0	-20	mA
L_0-L_7 Çıkışları				
(I_{OH})				
Üç-Durumlu Konfigürasyon	V_{CC} = 9.5V, V_{OH} = 4.75V	-2.4	-24.5	mA
Düşük Akım Sürücü Seçenekli	V_{CC} = 4.5V, V_{OH} = 2.25V	-0.06	-3.8	mA
L_0-L_7 Çıkışları				
(I_{OH})				
Yüksek Akım Sürücü Seçenekli	V_{CC} = 9.5V, V_{OH} = 4.75V	-4.9	-47.5	mA
L_0-L_7 Çıkışları	V_{CC} = 4.5V, V_{OH} = 2.25V	-0.12	-8.1	mA
(I_{OH})				
CKO Çıkışı				
RAM Güç Kaynağı Seçeneği	V_R = 3.3V		3.0	mA
Güç Gereksinimi				
TRI-STATE (ÜÇ-DURUMLU) Çıkış Kaçak Akımı		-10	+10	µA

COP420L/COP421L

AC Elektriksel Karakteristikleri 0°C ≤ T_A ≤ +70 °C, 4.5V ≤ V_{CC} ≤ 9.5V aksi belirtilmedikçe

Parametre	Koşullar	Min.	Maks.	Birimi
Komut Saykıl Süresi - t_C		15	40	µsn
CKI				
Giriş Frekansı - f_I	÷ 32 modu	0.8	2.0	MHz
	÷ 16 modu	0.4	1.0	MHz
	÷ 8 modu	0.2	0.5	MHz
	÷ 4 modu	0.1	0.25	MHz
İş Saykılı		30	60	%
Yükselme Süresi	f_I = 2 MHZ		120	ns
Düşme Süresi			80	ns
RC (4) kullanan CKI (saat)	R = 51 kΩ ± 5 % C = 100pF ± 10 %			
Komut Saykıl Süresi		15	25	µsn
SYNC Girişi olarak CKO t_{SYNC}		400		µsn
GİRİŞLER:				
IN_3 - IN_0, G_3 - G_0, L_7 - L_0				
t_{KURMA}		8		µsn
t_{TUTMA}		600		µsn
SI				
t_{KURMA}		2		µsn
t_{TUTMA}		600		µsn
ÇIKIŞLAR:				
COP - CMOS YAYINIM GECİKMESİ	V_{OH} = 0.7 V_{CC}, V_{OL} = 0.3 V_{CC}, C_L = 50 pF			
Tüm Standart Çıkış Konfigürasyonları				
t_{PD1}			6.5	µsn
SO, SK Çıkışları				
t_{PD1} (push-pull)			4.0	µsn
t_{PD0}			1.2	µsn
D_3 - D_0, G_3 - G_0				
t_{PD0}			2.7	µsn
L_7 - L_0				
t_{PD0}			2.7	µsn
t_{PD1} (Standart Boyutlu push-pull)			3.0	µsn
t_{PD1} (Yüksek Akımlı push-pull)			2.5	µsn
L_7-L_0 LED Doğrudan Sürücü Çıkışları	6.0 ≤ V_{CC} ≤ 9.5 V, V_{OH} = 2.0V C_L = 50 pF			
t_{PD1} (standart boyutlu)			5.0	µsn
t_{PD1} (yüksek akım)			4.5	µsn
COP - LSTTL YAYINIM GECİKMESİ	VCC = 5V ± 5%, V_{OH} = 2.7V, V_{OL} = 0.4 V, C_L = 50 pF			
SO, SK Çıkışları				
t_{PD1} (standart)			5	µsn
t_{PD1} (push-pull)			3.5	µsn
t_{PD0}			3	µsn
L_7-L_0 Çıkışları				
t_{PD1} (push-pull)			1.5	µsn
L_7-L_0, G_3-G_0, D_3 - D_0 Çıkışları				
t_{PD1} (standart)			5.0	µsn
t_{PD0}			2.0	µsn
CKO (Şekil 3b)				
t_{PD1}			0.4	µsn
t_{PD0}			0.4	µsn

COP320L/COP321L

Mutlak Maksimum Anma Değerleri

Göre Bacak Gerilimleri	-0.5V ile +10V
Ortamı Sıcaklığı	-40 °C ile +85 °C
Ortamı Sıcaklığı	-65 °C ile +150 °C
Sıcaklığı (Lehimlemede, 10 sn boyunca)	300 °C
Harcanımı	0.75 Wat - 25 °C 0.25 Wat - 85 °C

Mutlak maksimum anma değerleri, üzerinde çalışılması halinde cihazın zarar görebileceği sınır değerleridir. Eleman mutlak maksimum anma değerlerinde çalışırken, DC ve AC elektriksel spesifikasyonları için garanti edilmemiştir.

DC Elektriksel Karakteristikleri - 40 °C ≤ T_A ≤ + 85 °C, 4.5V ≤ V_{CC} ≤ 7.5V aksi belirtilmedikçe

Parametre	Koşullar	Min.	Maks.	Birimi
Standart Çalışma Gerilimi (V_{CC})	Not 1	4.5	5.5	V
İsteğe Bağlı Çalışma Gerilimi (V_{CC})		4.5	7.5	V
Çalışma Kaynak Akımı	V_{CC}=5V, (bütün giriş ve çıkışlar açık)		10	mA
Giriş Gerilim Düzeyleri				
CKI (saat) Giriş Düzeyleri				
Kristal Girişi				
Mantıksal Yüksek Düzey (V_{IH})		2.2		V
Mantıksal Alçak Düzey (V_{IL})		-0.3	0.3	V
Schmitt Tetikleyici Girişi				
Mantıksal Yüksek Düzey (V_{IH})		0.7 V_{CC}		V
Mantıksal Alçak Düzey (V_{IL})		-0.3	0.4	V
RESET Giriş Düzeyleri				
Mantıksal Yüksek Düzey		0.7 V_{CC}		V
Mantıksal Alçak Düzey		-0.3	0.4	V
RESET Histeresizliği		1.0		V
SO Giriş Düzeyi (Test Modu)		2.2	3.0	V
Diğer Bütün Girişler				
Mantıksal Yüksek Düzey	V_{CC} = Maksimum TTL eşik düzeyi seçenekleri seçildiğinde,	3.0		V
Mantıksal Yüksek Düzey	V_{CC} = 5V ± %10 yüksek eşik düzey seçenekleri seçildiğinde	2.2		V
Mantıksal Alçak Düzey		-0.3	0.4	V
Mantıksal Yüksek Düzey		3.6		V
Mantıksal Alçak Düzey		-0.3	1.2	V
Giriş Kapasitesi			7	pF
Yüksek Emp. Giriş Kaçak Akımı		-1	+1	µA
Çıkış Gerilim Düzeyleri				
LSTTL Çalışma	V_{CC}=5V ± 5%			
Mantıksal Yüksek Düzey (V_{OH})	I_{OH}= -20 µA	2.7		V
Mantıksal Alçak Düzey (V_{OL})	I_{OL}= 0.36 mA		0.4	V
CMOS Çalışma				
Mantıksal Yüksek Düzey	I_{OH} = -10 µA	V_{CC} - 1		V
Mantıksal Alçak Düzey	I_{OL} = +10 µA		0.2	V

Not 1: Uygun çalışmayı sağlamak için V_{CC} geriliminin değişim oranı, 0.5V/msn'den daha az olmalıdır.

COP320L/COP321L

DC Elektriksel Karakteristikleri (devamı) - 40 °C ≤ T_A ≤ + 85 °C, 4.5V ≤ V_{CC} ≤ 7.5V aksi belirtilmedikçe

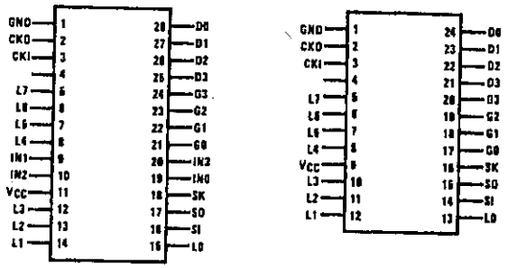
Parametre	Koşullar	Min.	Maks.	Birimi
Çıkış Akımı Düzeyleri				
Çıkış Çekme Akımı				
SO ve SK Çıkışları (I_{OL})	VCC = 7.5V, V_{OL} = 1.0V	4.0	22	mA
	V_{CC} = 4.5V, V_{OL} = 1.0V	2.1	12	mA
L_0-L_7 Çıkışları ve	V_{CC} = 7.5V, V_{OL} = 1.0V	1.7	8.0	mA
Standart Boyutlu G_0-G_3 ve	V_{CC} = 4.5V, V_{OL} = 1.0V	1.0	5.0	mA
D_0-D_3 Çıkışları (I_{OL})				
Yüksek Akım Seçenekli	V_{CC} = 7.5V, V_{OL} = 1.0V	14	60	mA
G_0-G_3 ve D_0-D_3 Çıkışları	V_{CC} = 4.5V, V_{OL} = 1.0V	7.0	40	mA
(I_{OL})				
Çok Yüksek Akım Seçenekli	V_{CC} = 7.5V, V_{OL} = 1.0V	28	120	mA
G_0-G_3 ve D_0-D_3	V_{CC} = 4.5V, V_{OL} = 1.0V	14	80	mA
Çıkışları (I_{OL})				
Çıkış Kaynak Akımı				
Standart Konfigürasyon	V_{CC} = 7.5V, V_{OH} = 3.75V	- 100	- 500	µA
Tüm Çıkışlar (I_{OH})	V_{CC} = 4.5V, V_{OH} = 2.25V	- 25	- 250	µA
Push-Pull Konfigürasyonu	V_{CC} = 7.5V, V_{OH} = 3.75V	- 1.3	- 9.0	mA
SO ve SK Çıkışları (I_{OH})	V_{CC} = 4.5V, V_{OH} = 2.25V	- 0.06	- 3.6	mA
LED Konfigürasyonu	V_{CC} = 7.5V, V_{OH} = 2.0V	- 1.4	- 25	mA
Düşük Akım Sürücü Seçenekli	V_{CC} = 6.0V, V_{OH} = 2.0V	- 1.4	- 13.0	mA
L_0-L_7 Çıkışları				
(I_{OH})				
LED Konfigürasyonu	V_{CC} = 7.5V, V_{OH} = 2.0V	- 2.7	- 51	mA
Yüksek Akım Sürücü Seçenekli	V_{CC} = 6.0V, V_{OH} = 2.0V	- 2.7	- 26	mA
L_0-L_7 Çıkışları				
(I_{OH})				
Üç-Durumlu Konfigürasyon	V_{CC} = 7.5V, V_{OH} = 3.75V	- 1.3	- 14.0	mA
Düşük Akım Sürücü Seçenekli	V_{CC} = 4.5V, V_{OH} = 2.25V	- 0.05	- 4.0	mA
L_0-L_7 Çıkışları				
(I_{OH})				
Yüksek Akım Sürücü Seçenekli	V_{CC} = 7.5V, V_{OH} = 3.75V	- 2.5	- 26.0	mA
L_0-L_7 Çıkışları	V_{CC} = 4.5V, V_{OH} = 2.25V	- 0.1	- 7.0	mA
(I_{OH})				
CKO Çıkışı				
RAM Güç Kaynağı Seçeneği	V_R = 3.3V		4.0	mA
Güç Gereksinimi				
TRI-STATE (ÜÇ-DURUMLU)		- 10	+ 10	µA
Çıkış				
Kaçak Akımı				

Not 2. Yüksek Sıcaklıkta, LED (veya benzer yükler) doğrudan sürüldüğünde, elemanın maksimum güç harcama sınırlarını aşmamak için çok dikkatli kullanılması gerekir.

COP320/COP321L

AC Elektriksel Karakteristikleri - 40 °C ≤ T_A ≤ + 85 °C, 4.5V ≤ V_{CC} ≤ 7.5V aksi belirtilmedikçe

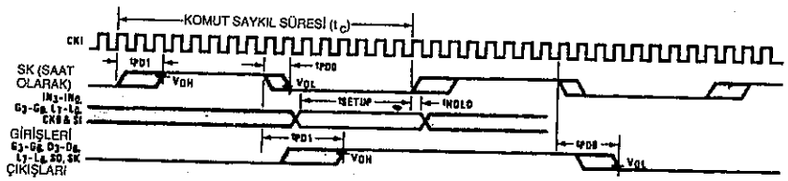
Parametre	Koşullar	Min.	Maks.	Birimi
Komut Saykıl Süresi - t_C		15	40	µsn
CKI				
Giriş Frekansı - f_I	÷ 32 modu	0.8	2.0	MHz
	÷ 16 modu	0.4	1.0	MHz
	÷ 8 modu	0.02	0.5	MHz
	÷ 4 modu	0.1	0.25	MHz
İş Saykılı		30	60	%
Yükselme Süresi	f_I = 2 MHz		120	nsn
Düşme Süresi			80	nsn
RC (4) kullanan CKI (saat)	R = 51 kΩ ± 5 %			
	C = 100pF ± 10 %			
Komut Saykıl Süresi		15	25	µsn
SYNC Girişi olarak CKO				
t_{SYNC}		400		µsn
GİRİŞLER:				
IN_3 - IN_0 , G_3 - G_0, L_7 - L_0				
t_{KURMA}		8		µsn
t_{TUTMA}		1		µsn
SI				
t_{KURMA}		2		µsn
t_{TUTMA}		1		µsn
ÇIKIŞLAR:				
COP - CMOS YAYINIM GECİKMESİ	V_{OH} = 0.7 V_{CC}, V_{OL} = 0.3 V_{CC}, C_L = 50 pF			
Tüm Standart Çıkış Düzenlemeleri				
t_{PD1}			7.0	µsn
SO, SK Çıkışları				
t_{PD1} (push-pull)			4.5	µsn
t_{PD0}			1.3	µsn
D_3 - D_0 , G_3 - G_0				
t_{PD0}			3.0	µsn
L_7 - L_0				
t_{PD0}			3.0	µsn
t_{PD1} (Standart Boyutlu push-pull)			3.3	µsn
t_{PD1} (Yüksek Akımlı push-pull)			2.8	µsn
L_7-L_0 LED Doğrudan Sürücü Çıkışları	6.0 ≤ V_{CC} ≤ 7.5 V, V_{OH} = 2.0V			
t_{PD1} (standart boyutlu)	C_L = 50 pF		5.5	µsn
t_{PD1} (yüksek akım)			5.0	µsn
COP - LSTTL YAYINIM GECİKMESİ	VCC = 5V ± 5%, V_{OH} = 2.7V, V_{OL} = 0.4 V, C_L = 50 pF			
SO, SK Çıkışları				
t_{PD1} (standart)			5.5	µsn
t_{PD1} (push-pull)			3.9	µsn
t_{PD0}			3.3	µsn
L_7-L_0 Çıkışları				
t_{PD1} (push-pull)			1.7	µsn
L_7-L_0, G_3-G_0, D_3 - D_0 Çıkışları				
t_{PD1} (standart)			5.5	µsn
t_{PD0}			2.2	µsn
CKO (Şekil 3b)				
t_{PD1}			0.45	µsn
t_{PD0}			0.45	µsn



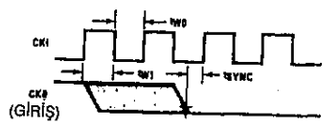
Şekil 2. Bağlantı Diyagramları

Bacak	Tanımı
L_7-L_0	8 adet TRI-STATE* (üç-durumlu) çift yönlü G/Ç portu
G_3-G_0	4 adet çift yönlü G/Ç portu
D_3-D_0	4 adet genel amaçlı çıkış
IN_3-IN_0	4 adet genel amaçlı giriş (yalnızca COP420L)
SI	Seri giriş (veya sayıcı girişi)
SO	Seri çıkış (veya genel amaçlı çıkış)

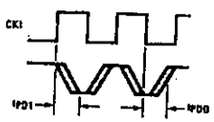
Bacak	Tanımı
SK	Mantık-denetimli saat (veya genel amaçlı çıkış)
CKI	Sistem osilatör girişi
CKO	Sistem osilatör çıkışı(veya genel amaçlı giriş)
$\overline{RESET}$	RAM güç kaynağı veya SYNC girişi)
Vcc	Sistem sıfırlama girişi
GND	Güç kaynağı
	Toprak



Şekil 3. Giriş/Çıkış Zamanlama Diyagramları (Kristal 16'ya Bölümlü modda çalışmaktadır)



Şekil 3a. Senkronizasyon Zamanlaması



Şekil 3b. CKO Çıkış Zamanlaması

Fiziksel Tanımlama

Tanımlamaların kolayca okunabilmesi için, yalnızca COP420L veya COP421L referans alınmıştır, bununla beraber verilen referanslar sırayla COP320 ve/veya COP321L'ye de uygulanabilir.

Şekil 1'de COP420'nin blok diyagramı verilmiştir. Veri yolları, komut takımının yürütülmesi yardımıyla mantık elemanları diğer elemanlarla nasıl iletişim kurduğunu göstermek için basitleştirilerek çizilmiştir. Pozitif mantık kullanılmıştır. Seçilen bu mantığın gereği olarak, bit kurulduğunda, bu; mantıksal 1 (2V'tan daha büyük), sıfırlandığında ise, mantıksal O (O,8 V'tan daha küçük) düzeyini gösterir.

Program Belleği

Program Belleği, 1024 baytlık ROM'dan oluşmuştur. COP420-/421L komut takımının incelenmesinden de görüleceği gibi bu sözcükler, program komutları, program verileri veya ROM adres verileri olabilir. JP, JSRP, JID ve LQID komutlarına ilişkin özel karakterler olduğu için, ROM'un, genelde her biri 64 sözcük içeren 16 sayfa şeklinde düzenlendiği düşünülmelidir.

ROM adreslemesi, 10-bitlik PC-kaydedicisiyle gerçekleştirilir. Kaydedicideki ikili değer, ROM'un içerdiği 1.024 adet 8-bitlik sözcükten birini seçer. Yeni adres, her komut saykılı sırasında PC kaydedicisine yüklenir. Komut, denetim aktarma komutu olmadığı sürece PC kaydedicisi, bir sonraki 10-bitlik sayım değeriyle yüklenir. Üç düzeyli alt yordamın oluşturulması; SA, SB ve SC gibi son giren ilk çıkar (LIFO) düzeninde donanımsal altyordam yığını sağlayan 10-bitlik altyordam saklama kaydedicileri tarafından gerçekletirilir.

ROM komut sözcüklerinin, getirilmesi, kodlarının çözülmesi ve yürütülmesi; Kod Çözme, Denetim ve Atlama Mantık devreleri tarafından gerçekleştirilir.

Veri Belleği

Veri belleği 16 adet 4-bitlik konumdan oluşan 4 adet veri kaydedicisi şeklinde düzenlenmiş, 256-bitlik RAM'dan oluşmuştur. RAM adreslemesi, üst-sıralı 2 bit (Br), 4 veri kaydedicisinden birini seçen, ve alt-sıralı 4 biti (Bd) de, seçilen veri kaydedicisindeki 16 adet 4-bitlik konumdan birini seçen, 6-bitlik B kaydedicisi tarafından yerine getirilir. Seçilmiş RAM gözününün (M) 4-bitlik içeriği, genellikle A kaydedicisine (Akümülatör'e) veya A kaydedicinden yüklenebilir ya da A kaydedicisinin içeriği ile değiş tokuş edilir, aynı zamanda, Q mandalına veya Q mandalından yükleme yapılabilir, yine benzer şekilde L portundan da yükleme yapmak mümkündür. RAM adreslemesi aynı zamanda, kendi işlenen alanlarının 6-bitlik içeriğine dayalı LDD ve XAD komutlarıyla doğrudan yürütülebilir. Bd kaydedicisi, D çıkışlarına doğrudan gönderilen 4-bitlik veri için kaynak kaydedicisi gibi de işlev görür.

İç Mantık Devreleri

4-bitlik A kaydedicisi (akümülatör), çeşitli G/Ç, aritmetik, mantık ve veri belleği erişim işlemleri için kaynak ve varışyeri kaydedicisidir. Bu kaydedici aynı zamanda, B kaydedicisinin Br ve Bd parçalarına yükleme yapmak, 8-bitlik Q mandalı verisinin 4 bitine yükleme ve giriş yapmak, 8-bitlik L G/Ç port verisinin 4bitini giriş yapmak ve SIO kaydedicisiyle veri değiş tokuşunu gerçekleştirmek için de kullanabilir.

4-bitlik toplayıcı, COP420/421L'nin aritmetik ve mantık işlemlerine yerine getirir ve sonuçları A'da saklar. Aynı zamanda, elde bitini, aritmetik işlemlerde ortaya çıkan taşmayı göstermede çok sıkça kullanılan 1-bitlik C kaydedicisine gönderir. XAS komutu ve EN kaydedicisiyle birlikte kullanılan C kaydedicisi, ayrıca SK çıkışının denetim işlevini de yerine getirir. C, doğrudan SK'ya çıkılabilir veya SK, her bir komut saykılı süreleri için bir senkronizasyon saatı olarak kullanılabilmek üzere yetkilendirilebilir. (Aşağıdaki XAS komutu ve EN kaydedicisi açıklamalarına bakın).

4 adet genel-amaçlı giriş için, IN_3-IN_0 girişleri mevcuttur.

D kaydedicisi, 4 genel amaçlı çıkış sağlar ve Bd'nin 4-bitlik içeriği için varış yeri kaydedicisi gibi kullanılır. D çıkışları, çoğullanmış LED göstergesinin basamaklarına doğrudan bağlanabilir.

G kaydedicisinin içeriği, 4-bitlik genel amaçlı iki yönlü G/Ç portuna gönderilir. G G/Ç portu, çoğullanmış LED göstergesinin basamaklarına doğrudan bağlanabilir.

Q kaydedicisi iç ve mandalı 8-bitlik kaydedicidir ve ROM'dan gelen 8-bitlik veri gibi, A'ya ve M'ye yüklenecek veya bu kaydedicilerden çıkışı yapılacak verileri tutmak için kullanılır. Bu kaydedicinin içeriği, L sürücüleri programın kontrolü altında yetkilendirildiğinde, L G/Ç portuna gönderilir (LEI komutuna bakın).

8-bitlik L sürücüsü, yetkilendirildiğinde, mandallanmış Q verisinin içeriğini L G/Ç portlarına gönderir. Aynı zamanda L'nin içeriği doğrudan A ve M'nin içine okunabilir. L G/Ç portları, çoğullanmış LED göstergesinin parçalarına doğrudan bağlanarak (LED doğrudan sürücü çıkış konfigürasyonu seçeneği kullanarak) Q verilerinin, ekranın ondalık nokta ve Sa-Sg parçalarına çıkışını sağlar.

SIO kaydedicisi, 4-bitlik seri giriş/seri çıkış kaydırma kaydedicisi veya EN kaydedicisinin içeriğine bağlı olarak sayan ikili sayıcı gibi işlev görür (Aşağıdaki EN kaydedicisi açıklamalarına bakınız). İçeriği A ile değiştirilebilir ve sürekli seri veri akışlarının girişine veya çıkışına izin verir. Aynı zamanda SIO, SO'nun dış seri giriş/paralel çıkış kaydırma kaydedicilerine bağlanmasıyla, paralel G/Ç'a artırmak için de kullanılır. Paralel çıkışın artırılması ile ilgili örnek için 2 no.lu uygulamaya bakınız.

XAS komutu C'yi SKL mandalına kopyalar. Sayıcı modunda, SK, SKL'nin çıkışıdır. Kaydırmalı kaydedici modunda, SK, SKL ile saatin VE'lenmiş çıkışıdır.

EN kaydedicisi, LEI komutuyla programın kontrolü altında yüklenen 4-bitlik iç kaydedicidir. Bu kaydedicinin herbir bitinin durumu, kendilerine ilişkin belirli özellikleri seçer veya seçmez (EN_2EN_0).

1. Yetkileme kaydedicisinin en düşük değerlikli biti EN0, SIO kaydedicisini, 4-bitlik kaydırma kaydedicisi veya 4-bitlik ikili sayıcı olarak seçer. EN0 biti kurulu iken ("1"), SIO asenkron ikili sayıcıdır. SI girişinde ortaya çıkan ve her düşük düzeye giden darbe ile ("1"den "0"a) değeri 1 azalır. Her darbenin süresi, en azından iki komut saykılı uzunluğunda olmalıdır. SK, SKL'nin değerini gösterir. SO çıkışı EN_3'ün değerine eşittir. EN0 biti sıfırlandığında ("0"), SIO, her komut saykılı süresince sola kaydırma yapan seri bir kaydırmalı kaydedicidir. SI girişinde hazır bulunan veri, SIO'nun en düşük değerlikli bitine gider. SO, her saykıl süresince SIO'nun en büyük değerlikli bitini çıkarmak üzere yetkilenebilir (Aşağıdaki 4. maddeye bakın) SK, bu aşamada mantık-denetimli saat olur.
2. EN_1 kurulu iken ("1") IN_1 girişi, kesme girişi olarak yetkilenir. Kesmenin ardından, diğer kesmelerin yetkilenmesini önlemek için EN1 biti sıfırlanır.
3. EN_2 kurulu iken ("1"), Q içindeki verilerin L G/Ç portlarına çıkması için L sürücüleri yetkilenir. EN_2 bitini sıfırlayarak L

sürücüleri yetkisizlenir ve L G/Ç portlarını yüksek empedanslı giriş durumuna sokar.

4. EN_0 ile birlikte EN_3, SO çıkışını etkiler. EN_0 kurulu iken ("1", ikili sayıcı seçeneği seçilmiştir) SO, EN_3 içine yüklenen değeri çıkarır. EN_0 sıfır iken, ("0", seri kaydırma yapan kaydedici seçeneği seçilmiştir) EN_3'ün kurulması ("1"), SO'nun, her komut süresince veriyi dışarıya seri olarak kaydıran SIO kaydırma kaydedicisinin çıkışı gibi yetkilenmesini, EN3'ün sıfırlanmasıyla seçilen olan seri kaydırmalı seçeneği, bu kaydedicinin çıkışı gibi devre dışı yetkisiz kılar; SIO içindeki kaydırılmış veri sırasıyla, komut yardımıyla A ile değiş-tokuş yapabilir fakat SO "0" düzeyinde kalır. Aşağıdaki tablo EN_3 ve EN_0 kullanılan modların bir özetini vermektedir.

Yetkileme Kaydedicisi Modları - EN_3 ve EN_0 Bitleri

	EN_3	EN_0	SIO	SI	SO	SK
0	0	0	Kaydırmalı Kaydedici	Kaydırmalı Kaydedicisi Girişi	0	Eğer SKL = 1, SK=1 Eğer SKL = 0, SK=1
	1	0	Kaydırmalı Kaydedici	Kaydırmalı Kaydedici Girişi	Seri Çıkış	Eğer SKL = 1, SK=1 Eğer SKL = 0, SK=0
	0	1	İkili Sayıcı	İkili Sayıcı Girişi	0	Eğer SKL = 1, SK=1 Eğer SKL = 0, SK=0
	1	1	İkili Sayıcı	İkili Sayıcı Girişi	1	Eğer SKL = 1, SK=1 Eğer SKL = 0, SK=0

Kesme

Aşağıda verilen özellikler; IN1 kesme yöntemi ve protokolune ilişkindir ve kesmeler kullanılırken, programcı tarafından göz önünde bulundurulmalıdır.

a. Kesme, aşağıda açıklandığı gibi önce kabul edilir, sonra bir sonraki program sayfası adresi (PC+1) yığına itilir, ardından diğer altyordam-saklama kaydedicilerinin içerikleri, bir sonraki kaydedicinin içeriği olacak şekilde döndürülür (PC+1 → SA → SB → SC). SC'nin bir önceki içeriği silinir. Program sayıcısı, onaltılı düzende OFF adresine (3. sayfanın en son sözcüğüne) kurulur ve EN1 biti sıfırlanır.

b. Kesmeler, aşağıdaki koşullarla karşılaşıldıktan sonra kabul edilecektir.
   1. EN1, kurulmuş ("1") olmalıdır.
   2. IN1 girişinde en azından iki komut saykılı boyunca alçak düzeye giden (1'den 0'a) darbe ortaya çıkmalıdır.
   3. O anda yürütülmekte olan komut tamamlanmış olmalıdır.
   4. Birbirini izleyen bütün denetim aktarma ve LBI komutları tamamlanmış olmalıdır (örneğin ana program; denetimini diğer JP komutunu aktaran bir JP komutunu yürütüyorsa kesme, ikinci JP komutu yürütülünceye kadar kabul edilmeyecektir).

c. Kesmenin kabulünde, atlama mantık durumu saklanır ve ardından yığından çekilme işleminde tekrar geriye alınır. Örneğin, eğer sonucu elde bitinde saklanan ASC (Eldeli Topla, Elde Atla) komutu yürütülürken bir kesme komutu ortaya çıkarsa, atlama mantık durumu saklanır ve programın denetimi, onaltılı OFF adresinde bulunan kesme hizmet altyordamına aktarılır. Kesme altyordamının sonunda yığına "itilmiş" verilerin geriye alınmaları için RET komutu yürütülür ve programını denetimi, orijinal ASC komutunun ardından gelen komuta geri döner. Bu anda, atlama mantığı yetkilenmiştir ve bir önceki ASC elde bitinden dolayı, bu komut atlanır. Altyordamlar ve LQID komutları, yığından çekilmeleri önceden saklanmış bulunan ana program atlamalarını yetkilendirecek ve kesme yordamının düzgün yürütülmesini aksatacağından, kesme yordamlarıyla birlikte iç içe kullanılmamalıdır.

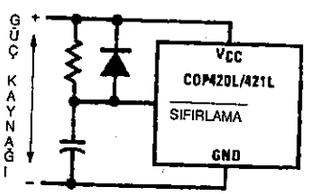
d. Onaltılı OFF adresinde bulunan kesme yordamının ilk komutu NOP olmalıdır.

e. Kesmelerin tekrar yetkilenebilmesi için RET komutundan hemen önce LEI komutu konulabilir.

Başlatma

Sıfırlama mantık devresi, eğer güç kaynağı yükselme süresi msn'den daha küçük ve 1 sn'den daha büyükse, birimi, her açılışında en baştan başlatacaktır (sıfırlayacaktır). Eğer güç kaynağı yükselme süresi 1 msn'den daha büyükse aşağıda görüldüğü gibi kullanıcı, dış RC devresini ve diyodu, mutlaka RESET bacağına bağlamalıdır. RESET bacağı, Schmitt tetikleyici girişi gibi düzenlenmiştir. Eğer kullanılmıyorsa, VCC'ye bağlanmalıdır. En azından üç komut saykılı süresi boyunca RESET girişine uygulanacak mantıksal "0" düzeyi, başlatmanın gerçekleşmesini sağlayacaktır.

Başlatma esnasında, PC kaydedicisi sıfırlanır (ROM'un 0. adresine alınır) ve A, B, C, d, EN ve G kaydedicileri de sıfırlanır. SK çıkışı, her komut saykılı süresince darbe üreten SYNC çıkışına yetkilenmiştir. Veri belleği (RAM), başlatma esnasında sıfırlanmaz. 0. adresteki ilk komut mutlaka CLRA olmalıdır.



RC ≥ 5 X GÜÇ KAYNAĞI YÜKS. SÜRESİ

AÇILIŞTA SIFIRLAMA DEVRESİ

gösterildiği gibi, saat osilatörünün 4 temel konfigürasyonu vardır.

Kristal Denetimli Osilatör. CKI ve CKO, dıştan takılan kristale bağlanır. Komut saykıl süresi, kristal frekansının 4'sine eşittir (isteğe bağlı olarak 1/16 veya 1/8 olabilir).

Dış Osilatör. CKI, dış saat giriş sinyalidir. Dış frekans, komut saykıl süresini düzenlemek üzere 32'ye bölünür (isteğe bağlı olarak 16 veya 8'e de bölünebilir). CKO bu sırada, RAM güç kaynağı (V_R), genel amaçlı giriş veya SYNC girişi gibi kullanılabilir.

RC denetimli Osilatör. CKI, RC denetimli Schmitt tetikleyici osilatörü olarak düzenlenmiş tek bacaktır. Komut saykıl süresi osilatör frekansının 4'e bölümüne eşittir. CKO, RAM güç kaynağı (V_R) veya genel amaçlı giriş gibi kullanılır.

Dıştan Eşzamanlanmış Osilatör. Çoklu-COP sistemlerinde kullanılmak üzere tasarlanmış bu konfigürasyonda CKO, görüldüğü gibi, aynı frekansta çalışan ve birbirlerine CKI ile bağlı diğer COP yongasının (L veya C sonekli COP yongası) SK çıkışına, giriş olarak bağlanacak şekilde programlanmıştır. Bu konfigürasyonda, CKO'a bağlı bulunan SK çıkışı, CKO'ya bir SYNC (komut saykılı) sağlamalıdır, böylece XAS komutuyla birlikte yalnızca SI ve SO seri G/Ç bacakları kullanılarak COP'lar arasında eşzamanlı veri aktarımı mümkün olur. Sistem açılışında SK'nın, SYNC çıkışı olacak şekilde otomatik olarak yetkilendiğine dikkat ediniz (Yukarıdaki İşlevsel Tanımlama ve Başlatma bölümlerine bakınız).

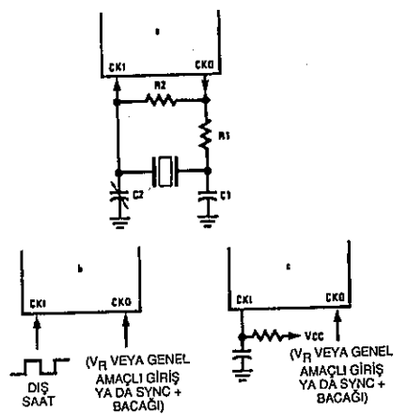
CKO Bacak Seçenekleri

Kristal denetimli osilatör sisteminde, CKO, kristal devresine çıkış olarak kullanılır. Yukarıda açıklandığı gibi, bir CKO seçeneği, CKO'nun, SYNC girişi olarak kullanılabilir. Diğer bir CKO seçeneği ise, INIL komutunun yürütümünde, A'nın (akümülatörün) 2. bitini okuyan genel amaçlı giriş olabilmesidir. Bir başka seçenek, ana besleme kaynağı çalışmıyorken veya güç korumak için kapatılmışken, RAM verisinin minimum güç tüketimiyle silinmeden tutulabilmesini sağlamak için yedek güç kaynağına bağlanmasına imkan veren bir RAM güç kaynağı ucu (V_R) olarak kullanılmasıdır. CKO bacağının kullanılmasının istenmediği COP420L/421L sistem zamanlama konfigürasyonu uygulamalarında bu iki seçeneği kullanmak uygundur.

RAM Canlı-Tutma Seçeneği

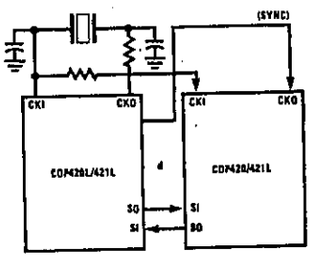
CKO'ın, RAM güç kaynağı (V_R) olarak seçilmesi, kullanıcıya, yonga güç kaynağının (V_{CC}) kapatılması durumunda bile, RAM içindeki verilerin korunabilmesi olanağını sağlar. RAM verisinin silinmeden bütünlük içinde tutulabilmesini sağlayabilmek için, aşağıdaki koşulların mutlaka yerine getirilmiş olması gerekir:

1. Güç-kapanması esnasında, V_{CC}, spesifikasyonda belirtilen değerlerin altına düşmeden önce RESET mutlaka ALÇAK düzeye gitmelidir. Güç uygulanması esnasında, RESET, YÜKSEK düzeye gitmeden önce V_{CC} mutlaka, spesifikasyonda belirtilen değerler içinde bulunmalıdır.
2. V_R, yonganın çalışma değerleri içerisinde ve normal çalışmada, V_{CC}, ± 1 V'a eşit olmalıdır.
3. V_{CC} kesildiği zaman V_R ≤ 3.3 V olmalıdır.



Kristalli Osilatör

Kristal değeri	Ekleme Değerleri			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	16k	1M	80	80
2.097 MHz	1k	1M	56	6-36



RC Denetimli Osilatör

R (kΩ)	C (pF)	Komut saykıl süresi (µsn)
51	100	19 ± 15%
82	56	19 ± 13%

200k > R > 25k
360 pF > C > 50 pF

Şekil 4. COP420/421L Osilatörü

G/Ç Seçenekleri

Aşağıdaki isteğe bağlı konfigürasyonlara sahip COP420/421L çıkışları, Şekil 5'te gösterilmiştir.

a. **Standart** - V_{CC}'ye bağlanabilen kanal-ayarlama modlu birimle birlikte, toprağa bağlanabilen kanal-geliştirme modundaki birim, LSTTL ve CMOS giriş gerekleriyle uyumludur. SO, SK ve bütün D ve G çıkışlarında kullanılabilir.

b. **Açık-Kanal** - Yalnızca toprağa bağlanabilen kanal-geliştirme modundaki birim, kullanıcı uygulamalarının gerektirdiği harici yukarı çekme dirençlerinin takılmasına olanak tanır. SO, SK ve tüm D ve G çıkışlarında kullanılabilir.

c. **Push-Pull** - Bu konfigürasyon; toprağa bağlanabilen kanal-geliştirme modlu birimle, V_{CC}'ye bağlı kanal-geliştime modlu bir birimle paralel kanal-ayarlama modlu birim'den oluşmuştur. Bu konfigürasyon, kapasitif yükler sürüldüğünde, hızlı yükselme ve düşme süreleri sağlar. Yalnızca SO ve SK çıkışlarında kullanılabilir.

d. **Standart L** - a şıkkında verilen konfigürasyon gibidir, fakat yetkisizlenebilme özelliği vardır. Yalnızca L çıkışlarında kullanılabilir.

e. **Açık-Kanallı L** - b şıkkında verilen konfigürasyon gibidir, fakat yetkisizlenebilme özelliği vardır. Yalnızca L çıkışlarında kullanılabilir.

f. **LED Doğrudan Sürme** - Toprağa ve V_{CC}'ye bağlanabilen kanalgeliştirme modunu ifade eder, LED göstergesindeki parçaların tipik akım gereksinimlerini sağlar. Kaynaklama birimi, limit akım değerine kenetlenmiştir. Programın denetimi altında kapatılabilen bu aygıtlar (EN kaydedicisi ve İşlevsel Tanımlamalara bakınız), çoğullamalı gösterge için gerekli LED parçasının karartılmasını sağlamak için çıkışları yüksek empedanslı duruma getirir. Yalnızca L çıkışlarında kullanılabilir.

g. **TRI-STATE * (Üç-Durumlu) Push-Pull** - Toprağa ve V_{CC}'ye bağlanabilen kanal-geliştirme modunu ifade eder. Diğer yol sürücüleri tarafından paylaşılmış veri yoluna bağlantı yapmaya izin veren bu çıkışlar, üç durumlu çıkışlardır. Yalnızca L çıkışlarında kullanılabilir.

COP420L/COP421L girişleri aşağıdaki isteğe bağlı konfi[illegible]yonlara sahiptir:

h. Yonga-içi boşaltılmış yüklü birim, V_{CC}'ye bağlanabilir.

i. Yüksek empedanslı girişler, dış elemanlar tarafından[illegible]ka "1" veya "0" ile sürülmelidir.

Yukarıdaki giriş ve çıkış konfigürasyonları, kanal-geliştirme [illegible]lu ve kanal-ayarlama modlu birimler tarafından paylaşılır. [illegible] olarak, tüm konfigürasyonlar, altı birimden biri veya bi[illegible] kullanabilir (sırayla 1-6 şeklinde numaralandırılmıştır). CO[illegible] L/421L sistemi tasarımında tasarımcıya, bu G/Ç konfigürasy[illegible]larını, etkin bir şekilde kullanabilmeye olanak tanıyan [illegible] birimlerin, minimum ve maksimum akım ($I_{çık}$ ve $V_{çık}$) eğ[illegible] Şekil 6'da verilmiştir.

SO, SK çıkışları, **a.**, **b.** veya **c.**'de gösterilen şekilde düzen[illegible] bilir. D ve G çıkışları a. veya b.'de gösterildiği gibi düzenlene[illegible] G portlarına veri girişi yapılıyorken, G çıkışlarının "1"e kurul[illegible] gerektiği unutulmamalıdır. L çıkışları **d.**, **e.**, **f.** veya **g.**'deki düzen[illegible]lenebilir.

Unutulmaması gereken önemli nokta **d.** veya **f.** konfigürasyo[illegible] L sürücüleri ile birlikte kullanılıyorsa, L sürücülerinin yetkisizlen[illegible]meleri durumunda bile, boşaltılmış yüklü birim, çok küçük [illegible] akım akıtacaktır (Şekil 6, birim 2'ye bakınız), bununla beraber [illegible] hatları giriş olarak kullanıldığında da, yetkisizlenmiş boşaltılm[illegible] birim, girişi, mantıksal "1" düzeyine çekmek için yeterli kaynak akımı üzerinde kuvvetlendirme yapmayabilir.

COP421L

Eğer COP420L, 24 bacaklı birim olarak üretilmişse, Şekil 2'de (COP420L/421L Bağlantı şemaları) gösterildiği gibi COP421L haline gelecektir. COP421L biriminin, dört genel amaçlı IN giriş[illegible]leri (IN_3-IN_0) içermediğine dikkat ediniz. Bu seçeneğin kullanım[illegible] tabii ki, IN seçeneklerinin ve kesme özelliklerinin kullanımı[illegible] engelleyecektir. Diğer bütün seçenekler COP421L içinde kul[illegible]nılabilir.

[illegible]420L/COP421L Komut Takımı

[illegible]'1'de, komut takımı tablosunda kullanılan komut işleneni, [illegible]sel sembolleri ve iç mimariyi gösteren sembol tablosu verilmiştir.

Tablo 2'de, anımsatıcı, işlenen, makina kodu, veri akışı, atlama koşulları ve COP420L/421 komut takımındaki her bir komuta ilişkin açıklamalar yer almaktadır.

Tablo 1. COP420L/421L Komut Takımı Tablo Sembolleri

Sembol	Tanımı
İÇ MİMARİ SEMBOLLERİ	
A	4-bitlik Akümülatör
B	6-bitlik RAM Adres Kaydedicisi
Br	B'nin üst 2 biti (kaydedici adresi)
Bd	B'nin alt 4 biti (sayı adresi)
C	1-bitlik Elde Kaydedicisi
D	4-bitlik Veri Çıkış
EN	4-bitlik Yetkileme Kaydedicisi
G	G/Ç Portu için Veriyi Mandallayan 4-bitlik Kaydedici
IL	IN_3 ve IN_0 Girişleriyle Birlikte Kullanılan İki adet 1-bitlik Mandal
IN	4-bitlik Giriş Portu
L	8-bitlik TRI-STATE (Üç-Durumlu) G/Ç Portu
M	B Kaydedicisinin gösterdiği RAM Belleğin 4-bitlik İçeriği
PC	10-bitlik ROM adres Kaydedicisi (program sayıcısı)
Q	Hat G/Ç Portu için Veriyi Mandallayan 8-bitlik Kaydedici
SA	10-bitlik Altyordam Saklama Kaydedicisi A
SB	10-bitlik Altyordam Saklama Kaydedicisi B
SC	10-bitlik Altyordam Saklama Kaydedicisi C
SIO	4-bitlik Kaydırma Kaydedicisi ve Sayıcı
SK	Mantık Denetimli Saat Çıkışı

Sembol	Tanımı
KOMUT İŞLENENİ SEMBOLLERİ	
d	4-bitlik İşlenen Alanı, 0-15 İkili (RAM Basamağı Seçimi)
r	2-bitlik İşlenen Alanı, 0-3 İkili (RAM Kaydedici Seçimi)
a	10-bitlik İşlenen Alanı, 0-1023 İkili (RAM adresi)
y	4-bitlik İşlenen Alanı, 0-15 İkili (Dolaysız veri)
RAM(s)	s ile Adreslenen RAM Konumunun İçeriği
ROM(t)	t ile Adreslenen RAM Konumunun İçeriği
İŞLEMSEL SEMBOLLER	
+	Artı
—	Eksi
→	Yerine yaz
↔	ile değiş-tokuş
=	Eşittir
$\overline{A}$	A'nın 1'e tümleyeni
⊕	Özel-Veya
:	Değerler aralığı

Tablo 2. COP420L/421L Komut Takımı

Kom.Kısıt.	İşlenen	On altılı Kod	Makina Dili Kodu (İkili)	Veri Akışı	Atlama Koşulları	Tanımı
ARİTMETİK KOMUTLAR						
ASC		30	00110000	A + C + RAM(B) → A Elde → C	Elde	Eldeli Topla, Elde Varsa Atla
ADD		31	00110001	A + RAM(B) → A	Yok	RAM'ı A'ya ekle
ADT		4A	01001010	A + 10_{10} → A	Yok	A'ya 10 Ekle
AISC	y	5-	0101 y	A + y → A	Elde	Dolaysız Topla, Elde Varsa Atla (y≠0)
CASC		10	00010000	$\overline{A}$ + RAM(B) + C → A Elde → C	Yok	Tümle ve Eldeli Topla, Elde varsa Atla
CLRA		00	00000000	0 → A	Yok	A'yı Sıfırla
COMP		40	01000000	$\overline{A}$ → A	Yok	A'nın 1'e Tümleyenini A'ya Yükle
NOP		44	01000100	Yok	Yok	Hiç İşlem Yapma
RC		32	00110010	"0" → C	Yok	C (Elde)'yi Sıfırla
SC		22	00100010	"1" → C	Yok	C (Elde)'yi Kur
XOR		02	00000010	A ⊕ RAM(B) → A	Yok	A ile RAM Arasında Özel-VEYA İşlemi Yap
DENETİM AKTARMA KOMUTLARI						
JID		FF	11111111	ROM(PC9:8, A, M) → $PC_{7:0}$	Yok	Dolaylı Atla (Not 3)
JMP	a	6-- --	011000 $a_{9:8}$ $a_{7:0}$	a → PC	Yok	Atla
JP	a	--	1 $a_{8:0}$ (pages 2,3 only) veya 1 1 $a_{5:0}$ (all other pages)	a → $PC_{8:0}$ a → $PC_{5:0}$	Yok	Sayfa içinde Atla (Not 4)
JSRP	a	--	1 0 $a_{5:0}$	PC + 1 → SA → SB → SC 0010 → $PC_{9:6}$ a → $PC_{5:0}$	Yok	Altyordam Sayfasına Atla (Not 5)
JSR	a	6- --	011010 $a_{9:8}$ $a_{7:0}$	PC + 1 → SA → SB → SC a → PC	Yok	Altyordama Atla
RET		48	01001000	SC → SB → SA → PC	Yok	Altyordamdan Geriye Dön
RETSK		49	01001001	SC → SB → SA → PC	Geriye Dönüşte Daima Atlar	Altyordamdan Geriye Dön, ardından Atla

Tablo 2. COP420L/421L Komut Takımı (Devamı)

Kom.Kısıt.	İşlenen	On altılı Kod	Makina Dili Kodu (İkili)	Veri Akışı	Atlama Koşulları	Tanımı
BELLEK REFERANS KOMUTLARI						
CAMQ		33 3c	00110011 00111100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	Yok	A ve RAM'ı Q'ya Kopyala
CQMA		33 2c	00110011 00101100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	Yok	Q'yu RAM'a ve A'ya Kopyala
LD	r	-5	00 r 0101	RAM(B) → A Br ⊕ R → Br	Yok	RAM'ı A'ya Yükle, r ile Br Arasında Özel-VEYA İşlemi Yap
LDD	r, d	23 --	00100011	RAM(r,d) → A	Yok	r, d ile tanımlanan RAM'ın Doğrudan Gösterdiği Yeri A'ya Yükle
LQID	0	BF	10111111	RAM($PC_{9:8}$, A, M) → Q SB → SC	Yok	Q'ya Dolaylı Yükle (Not 3)
RMB	0 1 2 3	4C 45 42 43	01001100 01000101 01000010 01000011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	Yok	RAM Bitini Sıfırla
SMB	0 1 2 3	4D 47 46 4B	01001101 01001101 01000110 01001011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	Yok	RAM Bitini Kur
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	Yok	Belleğe Dolaysız Sakla ve Bd'yi Artır
X	r	-6	00 r 0110	RAM(B) ↔ A Br ⊕ R → Br	Yok	RAM ile A'yı karşılıklı değiştir, Br ile r arasında Özel-VEYA İşlemi yap
XAD	r,d	23 --	00100011 10 r d	RAM(r,d) ↔A	Yok	r, d ile tanımlanan RAM'ın Doğrudan Gösterdiği Yer ile A'yı karşılıklı değiştir
XDS	r	-7	00 r 0111	RAM(B) ↔ A Bd-1 → Bd Br ⊕ r → Br	Bd, 0'dan geçene kadar azalır	RAM ile A'yı karşılıklı değiştir ve Bd'yi azalt, Br ile r arasında Özel-VEYA İşlemi yap
XIS	r	-4	00 r 0100	RAM(B) ↔ A Bd+1 → Bd Br ⊕ r → Br	Bd 15'ten geçene kadar artar	RAM ile A'yı karşılıklı değiştir ve Bd'yi Artır, Br ile r arasında Özel-VEYA İşlemi yap
KAYDEDİCİ REFERANS KOMUTLARI						
CAB		40	01010000	A → Bd	Yok	A'yı Bd'ye Kopyala
CBA		4E	01001110	Bd → A	Yok	Bd'yi A'ya Kopyala
LBI	r,d	--	00 r (d-1) (d=0, 9:15) veya 33 00110011 -- 10 r d (any d)	r,d → B	LBI Olmayana Kadar Atla	B'yi, r,d ile dolaysız Yükle
LEI		33 6-	00110011 0110 y	y → EN	Yok	EN'e dolaysız Yükle (Not 7)
XABR		12	00010010	A ↔ Br(0,0 → A_3,A_2)	Yok	A ile Br'yi karşılıklı değiştir

Tablo 2. COP420L/421L Komut Takımı (Devamı)

Kom.Kıslt.	İşlenen	On altılı Kod	Makina Dili Kodu (İkili)	Veri Akışı	Atlama Koşulları	Tanımı
TEST KOMUTLARI						
SKC		20	00100000		C="1"	C Doğru ise Atla
SKE		21	00100001		A=RAM(B)	A, RAM'a Eşitse Atla
SKGZ		33 21	00110011 00100001		$G_{3:0}$=0	G (tüm 4 bitl) Sıfırsa Atla
SKGBZ		33	00110011	1. bayt		G Bitl Sıfırsa Atla
	0	01	00000001		G_0=0	
	1	11	00010001		G_1=0	
	2	03	00000011	2. bayt	G_2=0	
	3	13	00010011		G_3=0	
SKMBZ	0	01	00000001		$RAM(B)_0$=0	RAM Bitl Sıfırsa Atla
	1	11	00010001		$RAM(B)_1$=0	
	2	03	00000011		$RAM(B)_2$=0	
	3	13	00010011		$RAM(B)_3$=0	
SKT		41	01000001		Zaman-tabanlı sayıcıdan çıkan elde son test sırasında üretilmiştir	Zamanlayıcıya Göre Atla (Not 3)
GİRİŞ/ÇIKIŞ KOMUTLARI						
ING		33 2A	00110011 00101010	G → A	Yok	G Portlarını A'ya gir
ININ		33 28	00110011 00101000	IN → A	Yok	IN Girişlerini A'ya Gir (Not 2)
INIL		33 29	00110011 00101110	IL_3, CKO, "0", IL_0 → A	Yok	IL Girişini A'ya Mandalla (Not 3)
INL		33 2E	00110011 00101110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	Yok	L Portlarını RAM ve A'ya Gir
OBD		33 3E	00110011 00111110	Bd → D	Yok	Bd'yi D Çıkışlarına Çık
OGI	y	33 5-	00110011 0101 y	y → G	Yok	G Portlarına Dolaysız Çıkış Yap
OMG		33 3A	00110011 00111010	RAM(B) → G	Yok	RAM verisini G Portlarına Çık
XAS		4F	01001111	A ↔ SIO, C → SKL	Yok	A'yı SIO ile karşılıklı değiştir (Not 3)

Not 1: Aksi belirtilmedikçe, alfabetik sembollere ilişkin tüm indisler, işaretler bit numaralarını gösterir (Örneğin burada Br ve Bd açık bir şekilde tanımlanmıştır). Bitler, en küçük değerlikli bitl gösteren 0 bitten (alt-sıralı, en-sağdaki bit), N. bite kadar numaralandırılır. Örneğin A3, 4-bitlik A kaydedicisinin en büyük değerlikli bitidir (en soldaki bit).
Not 2: 24 bacaklı COP421L, IN Girişlerini içermediğinden INI komutu bu birimde kullanılamaz.
Not 3: XAS, JID, LQID, INIL ve SKT komutlarının işleyişi hakkında ek bilgi için aşağıya bakın.
Not 4: JP komutu, 2. veya 3. altyordam sayfalarındayken, bu iki sayfanın sınırları içinde bulunan herhangi bir ROM konumuna atlamaya izin verir. Aksi halde, JP komutu, yürürlükteki 64sözcüklük sayfanın içinde bulunan bir ROM konumuna atlamaya izin verecektir. JP, sayfanın en son sözcüğüne atlamayabilir.
Not 5: JSPR komutu, program denetimini 2. altyordam sayfasına aktarır (P'nin üstteki dört bitine 0010 yüklenmiştir). JSPR, 2. ve 3. sayfalardayken kullanılamayabilir. JSPR, 2. sayfanın en son sözcüğüne atlama yapmayabilir.
Not 6: Eğer d = 0, 9, 10, 11, 12, 13, 14 ve 15 ise LBI tek baytlık komuttur. Son 4 bitl için makina kodu, "d" verisinin bir eksiğinin ikili tabandaki değerine eşittir. Örneğin, B'nin son dört bitini (Bd) 9 değeriyle (1001 2) yüklemek için, LBI komutunun son dört bitl 8'e (1000 2) eşittir. 0 yüklemek için, LBI komutunun son dört bitl 15'e (1111 2) eşit olmalıdır.
Not 7: LEI komutunun y işlenen alanının makina modu, EN içindeki mandallanmış ikili değere eşit olmalıdır. EN içindeki her bit ile birleştirilmiş özel işlevlerin seçilmesi veya seçilmemesi bunlara karşılık gelen bitlerinin "1" veya "0" yapılmasıyla sağlanır (İşlevsel Tanımlama, EN kaydedicisi'ne bakınız).

NEK LİSTESİ

COP421L'nin maske-programlanabilir seçeneklere, COP420L bacaklarına karşılık gelen numaralar verilmiştir.

Verilenler, COP420L'nin seçenek listesidir. COP421L yongası söz konusu olduğunda 9, 10, 19 ve 20. seçenekler sıfıra dir. Çok az veya hiç dış devre kullanmadan çeşitli G/Ç birimleriyle bağlantı kurmak üzere kullanıcılara donanım esnekliği n seçenekler aynı zamanda ROM biçimi olarak da programlanabilir.

nek 1 = 0: Toprak Bacağı - seçeneği yoktur

nek 2 = CKO Çıkışı
: kristal/rezonatöre giren saat üreteci çıkışı
(eğer seçenek 3=3 ise 0 geçersizdir değerdir)
: bacak RAM güç kaynağı (V_R) girişidir
: V_{CC}'ye bağlanabilen yük birimi olan genel amaçlı giriştir
: Genel amaçlı giriş, yüksek empedanslı
: Çoklu-COP SYNC girişi (CKI ÷ 32, CKI ÷ 16)
: Çoklu-COP SYNC girişi (CKI ÷ 8)

nek 3 = CKI Girişi
: osilatör girişi 32'ye bölünmüştür (maks. 2 MHz)
: osilatör girişi 16'ya bölünmüştür (maks. 1 MHz)
: osilatör girişi 8'e bölünmüştür (maks. 500 KHz)
: tek bacaklı RC denetimli osilatör (÷ 4)
: Schmitt tetikleyici saat girişi (÷ 4)

Seçenek 4 = RESET (sıfırlama) Girişi
= 0: yük V_{CC}'ye bağlanır
= 1: Yüksek-Empedanslı giriş

Seçenek 5 = L_7 Sürücüsü
= 0: Standart çıkış
= 1: Açık-kanallı çıkış
= 2: Yüksek akımlı LED doğrudan parça sürücü çıkışı
= 3: Yüksek akımlı TRI-STATE (üç-durumlu) push-pull çıkışı
= 4: Düşük akımlı LED doğrudan parça sürücü çıkışı
= 5: Düşük akımlı TRI-STATE (üç-durumlu) push-pull çıkışı

Seçenek 6: L_6 Sürücüsü
Seçenek 5 gibidir

Seçenek 7: L_5 Sürücüsü
Seçenek 5 gibidir

Seçenek 8: L_4 Sürücüsü
Seçenek 5 gibidir

Seçenek 9: IN_1 Girişi
= 0: yük birimi V CC'ye bağlanır
= 1: Yüksek Empedanslı giriş

Seçenek 10: IN_2 Girişi
Seçenek 9 gibidir

Seçenek 11: V_{CC} Bacağı
= 0: Standart V_{CC}
= 1: İsteğe bağlı daha yüksek gerilimli V_{CC}

Seçenek 12: L_3 Sürücüsü
Seçenek 5 gibidir

Seçenek 13: L_2 Sürücüsü
Seçenek 5 gibidir

Seçenek 14: L_1 Sürücüsü
Seçenek 5 gibidir

Seçenek 15: L_0 Sürücüsü
Seçenek 5 gibidir

Seçenek 16: SI Girişi
Seçenek 9 gibidir

Seçenek 17: SO Çıkışı
= 0: Standart çıkış
= 1: Açık kanallı çıkış
= 3: Push-pull çıkış

Seçenek 18: SK Sürücüsü
Seçenek 17 gibidir

Seçenek 19: IN_0 Girişi
Seçenek 9 gibidir

Seçenek 20: IN_3 Girişi
Seçenek 9 gibidir

Seçenek 21: G_0 G/Ç Portu
= 0: Çok yüksek akımlı standart çıkış
= 1: Çok yüksek akımlı açık-kanal çıkışı
= 2: Yüksek akımlı standart çıkış
= 3: Yüksek akımlı açık-kanal çıkışı
= 4: Standart LSTTL çıkışı (sürebileceği maksimum kapı sayısı = 1)
= 5: Açık kanallı LSTTL çıkışı (sürebileceği maksimum kapı sayısı = 1)

Seçenek 22: G_1 G/Ç Portu
Seçenek 21 gibidir

Seçenek 23: G_2 G/Ç Portu
Seçenek 21 gibidir

Seçenek 24: G_3 G/Ç Portu
Seçenek 21 gibidir

Seçenek 25: D_3 Çıkışı
Seçenek 21 gibidir

Seçenek 26: D_2 Çıkışı
Seçenek 21 gibidir

Seçenek 27: D_1 Çıkışı
Seçenek 21 gibidir

Seçenek 28: D_0 Çıkışı
Seçenek 21 gibidir

Seçenek 29: L Giriş Düzeyleri
= 0: Standart TTL giriş düzeyleri ("0" = 0.8V, "1" = 2.0V)
= 1: Yüksek gerilim giriş düzeyleri ("0" = 1.2V, "1" = 3.6V)

Seçenek 30: IN Giriş Düzeyleri
Seçenek 29 gibidir

Seçenek 31: G Giriş Düzeyleri
Seçenek 29 gibidir

Seçenek 32: SI Giriş Düzeyleri
Seçenek 29 gibidir

Seçenek 33: RESET (sıfırlama) Girişi
= 0: Schmitt tetikleyici girişi
= 1: Standart TTL giriş düzeyleri
= 2: Yüksek gerilim giriş düzeyleri

Seçenek 34: CKO Giriş Düzeyleri (CKO=giriş, Seçenek 2=2.3)
Seçenek 29 gibi

Seçenek 35: COP Kılıfı
= 0: COP420L (28 bacaklı birim)
= 1: COP421L (24 bacaklı birim)

TEST MODU (Standart-Dışı Çalışma)

Müşteri tarafından programlanabilen COP420L için SO çıkışı, standart test yöntemlerini sağlayacak şekilde konfigüre edilmiştir. SO çıkışı "1" düzeyine çekilerek ve SI'nın değerine bağlı olarak iki test modu sağlanmıştır.

a. RAM ve İç Mantık Test Modu (SI = 1)

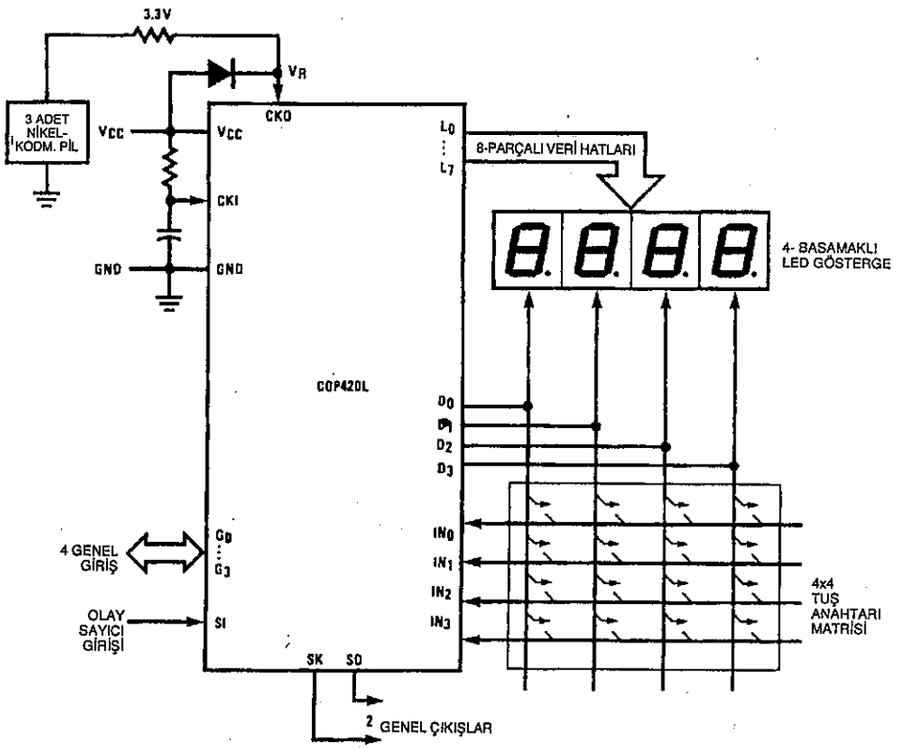
b. ROM Test Modu (SI = 0)

Bu özel test modları kullanıcılar tarafından çalıştırılmamalıdır. Bu modlar, yalnızca üretim testleri için kullanılabilir.

UYGULAMA 1: COP420L Genel Denetleyici

Şekil 9 COP420L'nin genel denetleyici olarak kullanılabilmesi için gerekli bağlantı şemasını gösterir. Sistemin çalışması aşağıdaki gibidir.

1. L_7-L_0 çıkışları, göstergenin parçalarına doğrudan bağlanabilen, LED Doğrudan Sürücü Çıkışları olarak konfigüre edilmiştir.
2. D_3-D_0 çıkışları, çoğullanmış göstergenin basamakla[illegible] rudan sürer ve 4x4 klavye matrisinin sütunlarını tara[illegible]
3. IN3-IN0 girişleri, klavye matrisinin 4 satırının giriş[illegible] kullanılır. D çıkışlarının yürürlükteki değeriyle birlikte [illegible] IN girişleri, 16 tuş anahtarından herhangi birinin bas[illegible] sılmadığının algılanmasına ve kodunun çözülmesine [illegible] verir.
4. CKI, tek uçlu RC devresi tarafından denetlenmek için [illegible] zamanlamalarına izin veren tek uçlu osilatör girişi[illegible] yüzden CKO, VR RAM güç kaynağı ucu gibi kullan[illegible] Böylelikle güç kaynağı kapatılsa bile RAM verisinin koru[illegible] sı sağlanmış olur (RAM'i Canlı-Tutma Seçeneği açıkla[illegible] rına bakın).
5. SI, ikili sayıcı girişi olarak seçilir. İkili sayıcı gibi kullanılan [illegible] ile birlikte, SO ve SK genel amaçlı çıkışlar olabilir.
6. 4 adet çift yönlü G G/Ç portları (G_3-G_0), kullanıcı uygula[illegible] sındaki ihtiyaçların karşılanması için kullanılır.



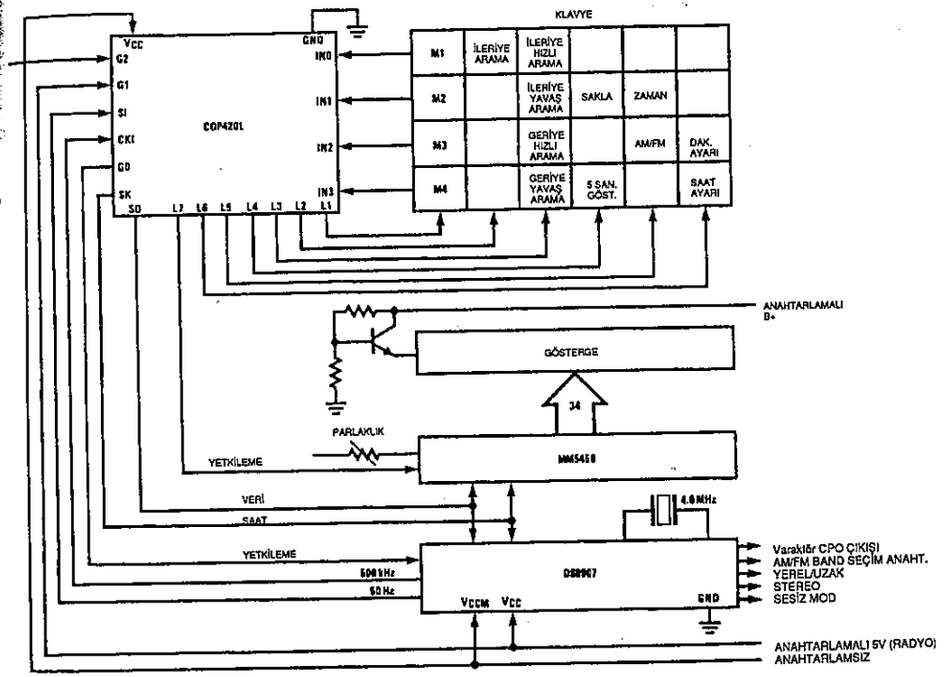
*SO, SI, SK, SERİ G/Ç İÇİN DE KULLANILABİLİR

Şekil 9. COP420L Klavye/Gösterge Arabirimi

[illegible]GULAMA 2: Sayısal Akortlu Otomobil Radyo [illegible]etleyicisi ve Saati

[illegible]10'de, LED gösterge, istasyonu saklamak ve tekrar çağır[illegible] [illegible]in 4x6 klavye, istasyonları ileriye veya geriye doğru arama [illegible]tarama özelliği, AM/FM seçimi ve zaman ayarlama yete[illegible]erine sahip sayısal saatli, sayısal olarak akort edilebilen [illegible]M otomobil radyosunun bağlantı diyagramını gösterir. Sis[illegible]in çalışması aşağıdaki gibidir.

1. DS8907, frekans sentezlemesine taban-zamanını sağlamak [illegible]in 4 MHz'lik kristal kullanılır ve COP 420L'nin çalışması için 500 KHz'lik taban-zamanı ve zamana ilişkin işlevleri için 50 [illegible]Hz'lik sinyal içerir.
2. Anahtarlanmasız 5V'luk kaynak, osilatörün çalışması ve bö[illegible]lünerek elde edilen 500 Khz ile 50 Hz'lik sinyaller için DS8907'nin V CCM bacağına bağlanır. Aynı zamanda bu gerilim, kontak kapandığı zaman, saat devreleri kanal bilgisinin saklanması ve son seçilmiş istasyon bilgisinin silinmesi için COP420L'nin beslemesini de (V CC) sağlamaktadır.
3. Radyo açıldığı zaman yüksek düzeye giden anahtarlamalı 5 V'luk kaynak, frekans üretim devresi için DS8907'nin V CC bacağına ve COP420L'nin G1 G/Ç bacağına bağlanır.
4. L1 ile L6 arasındaki çıkışlar klavyeyi taramak için çıkıştır (push-pull seçeneği seçilmiştir) ve IN0-IN3 girişleri, klavye girişleridir (V CC'ye bağlı yukarı-çekme ve yüksek turlu düzeyler seçilmiştir)
5. SK, seri girişli MM5450 gösterge sürücüsü ve DS8907 PLL Sentesizör saatini ve SO verisini sağlar. L7, MM5450 için yetkilendirme bacağıdır ve G0 (standart seçenek seçilmiştir) DS8907 için yetkilendirme hattıdır.
6. Yukarıya ve aşağıya arama işlemlerinde, istasyon algılandığı zaman G2, COP420L'yi bilgilendirir.



Şekil 10. Elektronik Akortlu Radyo Sistemi

EK F Intel 8080A MİB için Saat Üreteci ve Sürücüsü

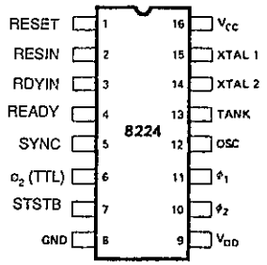
- 8080A MİB için Tek-Yongalı Saat Üreteci/Sürücüsü
- MİB için Açılışta-Sıfırlama
- Hazır Eşzamanlama Flip-Flobu
- Geliştirilmiş Durum Strobu
- Dış Sistem Zamanlaması İçin Osilatör Çıkışı
- Sistemin Kararlı Çalışması için Kristal Denetimi
- Sistem paket sayısını azaltır.

8224; 8080A MİB için tek yonga saat üreteci/sürücüsüdür. Sistemin çeşitli hız gereksinimlerini karşılamak için, tasarımcılarca seçilmiş kristal tarafından denetlenir.

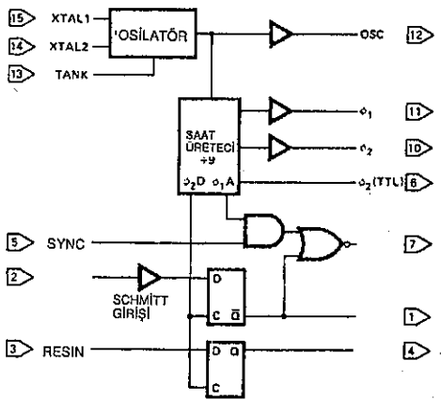
Ayrıca; açılışta-sıfırlama, ileri durum strobunu ve hazır durumu eşzamanlılığını sağlamak için devreler de içerir.

8080A zamanlamasının ve saatinin üretilmesinde kullanılan 8224, azaltılmış paket sayısı ile tasarımcılara kolaylık sağlar.

BACAK DİYAGRAMI



BLOK DİYAGRAMI



BACAK ADLARI

RESIN	SINIRLAMA GİRİŞİ
RESET	SINIRLAMA ÇIKIŞI
RDYIN	HAZIR GİRİŞİ
READY	HAZIR ÇIKIŞI
SYNC	SYNC GİRİŞİ
STSTB	DURUM STROBU (ALÇAKTA-ETKİN)
ϕ_1, ϕ_2	8080 SAATLERİ

XTAL 1, XTAL 2	KRİSTAL BAĞLANTILARI İÇİN
TANK	OVERTONE KRİSTALİ İLE BİRL. KULL.
OSC	OSİLATÖR ÇIKIŞI
ϕ_2 (TTL)	ϕ_2 CLK
V_{CC}	+5V
V_{DD}	+12V
GND	0V

İŞLEVSEL TANIMLAMA

Genel

8224, 8080A MİB için tek yonga saat üretici/sürücüsüdür. Kristal denetimli osilatör, 9'a bölünebilen sayıcı, iki yüksek düzeyli sürücü ve bazı yardımcı mantık işlevlerini içerir.

Osilatör

Osilatör devresi, kendi temel çalışma frekansını, dıştan takılan, seri rezononslı, temel mod kristalden üretir. Kristal bağlantıları için (XTAL1 ve XTAL2 olmak üzere) iki giriş sağlanmıştır.

Dış kristal frekansının seçimi, temel olarak 8080A'nın çalıştırılacağı hıza bağlıdır. Temel olarak osilatör, istenen işlemci hızının 9 misline çalışır.

Kristal seçim yöntemi için basit formül:

Kristal frekansı: $\frac{1}{t_{CY}} \times 9$

Örnek 1: (500 nsn t_{CY})
2MHz x 9 = 18MHz*

Örnek 2: (800 nsn t_{CY})
1.25 MHz x 9 = 11.25MHz

Osilatöre diğer giriş TANK'tır. Bu giriş aşırı ton modlu kristallerin kullanımına izin verir. Bu tip kristaller, temel tip kristallerden daha az kazanca sahip olduklarından, uygun osilatör işlemi için ek kazanç sağlayan dış LC devresi gereklidir. Dış LC devresi TANK girişine bağlanmıştır ve toprağa AC bağlantısı yapılmıştır. Bkz:Şekil 4.

LC devresi için formül:

$$F = \frac{1}{2\pi\sqrt{LC}}$$

Diğer sistem zamanlama sinyalleri bu kararlı, kristal denetimli kaynaktan türetilebilmesi için osilatör çıkışı tamponlanmıştır ve OSC çıkışına (12 no.lu bacak) getirilmiştir.

* 10 MHz'in üstünde kristaller kullanıldığında, kesin olarak istenen frekansı üretmek için, küçük bir miktar frekans "ayarlaması" gerekebilir. Bu, kristale seri olarak bağlanan küçük bir kapasite sayesinde yapılır.

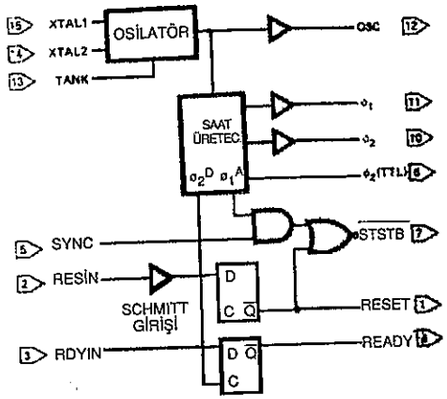
Saat Üreteci

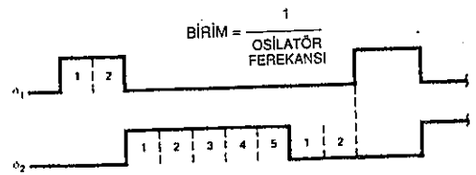
Saat üreteci, iki adet 8080A saati ve yardımcı zamanlama sinyallerini yaratmak üzere "dokuza bölünebilen" sayıcı ve bu sayıcıyla birlikte kullanılan kod çözücü kapılardan oluşur.

Kod çözücü kapıların ürettiği dalgabiçimleri, basit 2-5-2 sayısal desenini izler. Bkz:Şekil 2. Faz 1 ve Faz 2 olarak üretilmiş saat, osilatör frekansına bağımlı "birimler" gibi düşünülebilir. "Birimin" osilatör frekansının peryoduna eşit olduğunu varsayın. Darbe genişliğini veya gecikmesini içeren bu birimlerin sayısı çarpılarak, osilatör frekansının peryodunun katlarının süresi, nanosaniyeler cinsinden yaklaşık olarak bulunabilir.

Saat üretecinin çıkışları, 8080A MİB'ne doğrudan bağlantı için iki adet yüksek düzeyli sürücüye bağlanmıştır. TTL düzeyli faz 2, dış zamanlama isteklèri için ayrıca 2 (TTL) çıkışına getirilmiştir. Özellikle, DMA-tabanlı etkinliklerde yararlıdır. Bu sinyal, 8080A MİB bir kez Tutma Alındısını (HLDA) ürettiğinde, istekte bulunan birimi yol üzerinde kapılamak amacıyla kullanılır.

Yardımcı flip-flopların ve durum strobunun ($\overline{STSTB}$) optimum zamanlamasının sağlanması için diğer bazı sinyaller, devre içinden üretilmiştir.





ÖRNEK: (8080 t_{CY} = 500ns)
OSC = 18mHz/55ns
ϕ_1 = 110ns (2 x 55ns)
ϕ_2 = 275ns (5 x 55ns)
ϕ_2 ϕ_1 = 110ns (2 x 55ns)

$\overline{\text{STSTB}}$ (Durum Strobu)

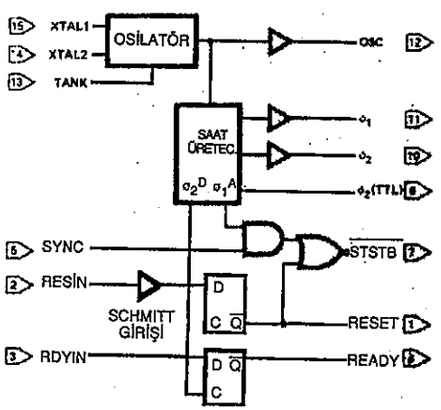
Her makina saykılının başında 8080A MİB, durum bilgisini kendi veri yoluna çıkarır. Bu bilgi, makina saykılı süresince hangi tip bilginin veri yolunda etkin hale geleceğini söyler. MİB'den gelen SYNC sinyali ve 8224 iç zamanlama sinyali (1A) ile kapılanması sonucu, her makina saykılının başında, mümkün olabildiğince başlarda durum verisini yol üzerinde kararlı şekilde ortaya çıkan alçakta-etkin strobe sinyali türetilebilir. $\overline{\text{STSTB}}$ sinyali doğrudan 8228 Sistem Denetleyicisine bağlanır.w

Açılışta-sıfırlama da $\overline{\text{STSTB}}$ sinyali üretir, fakat tabi ki bu, uzun peryotlu süreler içindir. Bu özellik, 8228'in bu işlev için ek bacaklar ayırmaksızın otomatik olarak sıfırlanmasına olanak sağlar.

Açılışta-Sıfırlama ve Hazır Flip Flopları

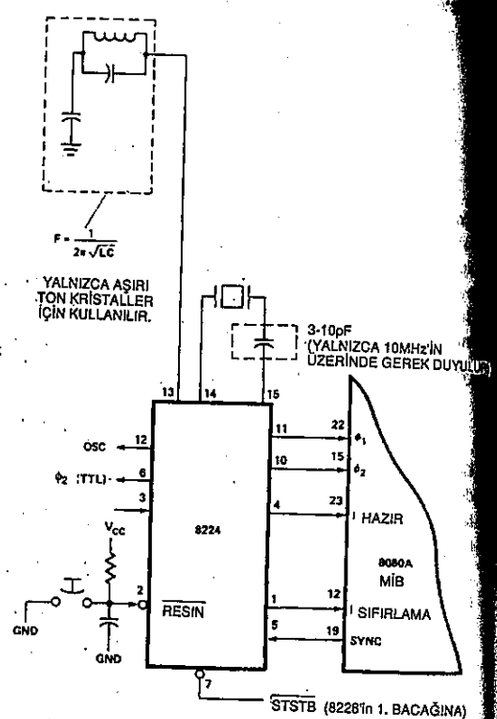
8080A Mikrobilgisayar sistemlerinin ortak işlevi, sistemin ilk açılışında otomatik olarak bir açılışta-sıfırlama üretmesi ve sonra çalışmaya başlamasıdır. 8224'da, bu işlemin yapılabilmesine olanak tanıyan yerleşik bir özellik mevcuttur.

Dış RC devresi $\overline{\text{RESIN}}$ girişine bağlanır. Güç kaynağının yavaş yükselme hareketi, iç Schmitt tetikleyici tarafından algılanır. Bu devre, giriş düzeyinin başta öngörülen değere erişmesi halinde, yavaş değişimi temiz ve hızlı kenar haline dönüştürür. Schmitt Tetikleyicisinin çıkışı, 2D (iç zamanlama sinyali) ile saatlenmiş D tipi bir flip-flopa bağlanmıştır. Flip-flop eşzamanlı olarak sıfırlanır ve 8080A'nın giriş özelliklerine uyan yüksek-etkin düzey üretilmiş olur. Elle anahtarlanabilen tipte sistem sıfırlama devreleri için, sistem açılış RC devresine ek olarak, alçakta-etkin bir anahtar, $\overline{\text{RESIN}}$ girişine bağlanabilir.



8080A MİB'ne HAZIR (READY) girişi, "kurma v... gibi kesin zamanlama özellikleri gerektirir, bu nede... dış eşzamanlama flip-flobuna gerek vardır. Bu ... 8224'da mevcuttur. RDYIN girişi asenkron "bek... isteği" olarak "D" tipi flip-floba uygulanır. ø2D ile ... lenmiş flip-flop, doğru giriş düzeyinde 8080A'ya doğ... dan bağlanabilen HAZIR (READY) sinyalini eşzaman...

"Bekleme" sinyali gerektiğinde, dış MOS mantık devre... lerinin karar vermesi esnasında, tasarımcıların yaklaş... 200 nsn'ni "çalan" bu uzun gecikmeli devrelerden dolayı, "bekleme isteğini" eşzamanlamak için gereken dış flip-flop, 8080A MİB'nin içine yerleştirilmiştir. Dış iki kutuplu devre, saat üretecinin içine yerleştirilerek, bu gecikmeni... büyük bir kısmı ortadan kaldırılır ve devre, eleman sayısının fazla oluşu nedeniyle olumsuz etkilenmez.



BK G Intel 8080A MİB için Sistem Denetleyicisi ve Yol Sürücüsü

- MCS-80 Sistemleri İçin Tek Yongalı Sistem Denetimi
- Veri Yolunun İzolasyonu İçin Yerleşik Çift-Yönlü Yol Sürücüsü
- Kesme Alındısı İçin, Çok Baytlı (örneğin CALL) Komutların Kullanılmasına İzin Verir.
- Kullanıcı Seçimli Tek Düzeyli Kesme Vektörü (RST 7)
- 28-bacaklı DIL Paketi
- Azaltılmış Sistem Paket Sayısı
- 8238, Büyük Sistemlerin Zamanlama Denetimi için, İleri $\overline{\text{IOW}}$/$\overline{\text{MEMW}}$ özelliklerine sahiptir.

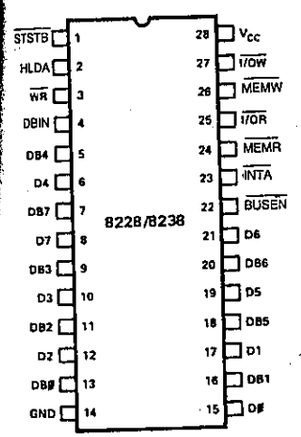
8228; MCS-80 için tek yongalı sistem denetleyicisi ve yol sürücüsüdür. MCS-80 ailesi RAM, ROM ve G/Ç birimlerle doğrudan bağlantı kurmak için gerekli tüm sinyalleri üretir.

İki yönlü yol sürücüsü, yüksek sistem TTL çıkış yelpazesini (fanout) sağlamak için dahil edilmiştir. Aynı zamanda 8080 veri yolunun bellekten ve G/Ç birimlerinden izolasyonunu da sağlar. Bu, tasarımcının, yavaş bellek ve G/Ç birimleri kullanabilmesini mümkün kılarak denetim sinyallerinin en uygun duruma getirilmesine imkan verir. Yol sürücüsünün yoldan izolasyonu, aynı zamanda, sistemin gürültü bağışıklığının artırılmasını da sağlar.

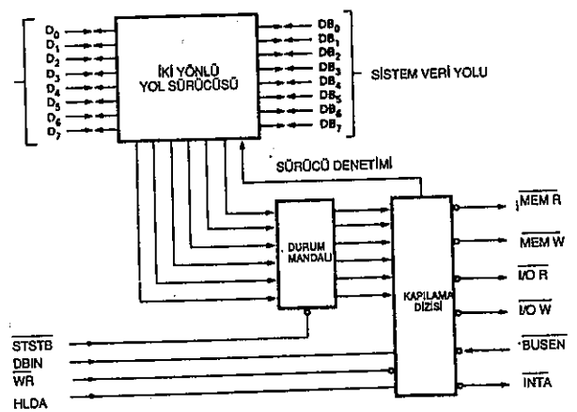
Kullanıcının seçebileceği tek düzeyli kesme vektörü (RST 7), gerçek-zamanlı çalışmayı, kesme sürücülü, küçük sistem gereklerini basitleştirmek için sağlanmıştır. Aynı zamanda 8228, 8080A'nın KESME ALINDISI'na cevap olarak kullanılan çok baytlı komutlara (örneğin CALL) izin vermek için doğru denetim sinyallerini de üretir. Bu özellik, büyük ve kesme sürücülü sistemlerin, sınırsız sayıda kesme düzeylerine sahip olmalarına olanak sağlar.

8228, çok çeşitli sistem yol yapılarını desteklemek için tasarlanmıştır ve MCS-80 sistemlerinin tasarımı, güvenilirliği ve gerçek fiyatları için sistem paketi sayısı azaltılmıştır.

BACAK TANIMLARI



8228/8238 BLOK DİYAGRAMI



BACAK ADLARI

D7-D0	VERİ YOLU (8080 TARAFI)	INTA	KESME ALINDISI
DB7-DB0	VERİ YOLU (SİSTEM TARAFI)	HLDA	HLDA (8080'DEN ÇIKAN)
I/OR	G/Ç OKUMA	WR	WR (8080'DEN ÇIKAN)
I/OW	G/Ç YAZMA	BUSEN	YOL YETKİLEME GİRİŞİ
MEMR	BELLEKTEN OKUMA	STSTB	DURUM STROBU (8224'DEN ÇIKAN
MEMW	BELLEĞE YAZMA	Vcc	+5V
DBIN	DBIN (8080'DEN ÇIKAN)	GND	0 VOLT

İŞLEVSEL TANIMLAMA

Genel

8228 ve 8238, 8080 Mikrobilgisayar sistemi için tek yongalı sistem denetleyicileri ve veri yolu sürücüleridir. MCS-80 ailesi RAM, ROM ve G/Ç birimleriyle doğrudan bağlantı kurmak için gereken bütün denetim sinyallerini üretir.

Küçük gecikme süreleri sağlamak ve küçük sistemlerle orta boy sistemleri denetlemek için, yüksek çıkış sürme yeteneği sağlayan Schottky iki-kutup teknolojisi kullanılmıştır.

İki-yönlü Yol Sürücüsü

8-bitlik, iki-yönlü yol sürücüsü; bellek ve G/Ç birimleri ile 8080 veri yoluna arasında tampon görevini görür. 8080A veri yolu 3.3 Volt (min) giriş gereksinimine sahiptir ve maksimum 1.9 mA'lik akım sürebilir. 8228/8238 veri yolu sürücüsü, yalnızca bu giriş gereksinimlerini karşılamakla kalmaz, artırılmış gürültü bağışıklığının da aşılmasını sağlar. Ayrıca, sürücünün sistem tarafındaki uygun sürme akımı (tipik olarak 10 mA), çok sayıdaki bellek ve G/Ç birimlerinin doğrudan yola bağlanabilmesi için kullanılabilir.

İki-yönlü Yol Sürücüsü, DMA işlemleri için kendi çıkışlarını diğer birimlerin yüksek empedanslı (3. durum) durumuna geçmeye zorlamak ve uygun yol akışını sağlamak için Kapılama Dizisi tarafından üretilmiş sinyallerle denetlenir.

Durum Mandalı

Her makina saykılının başında 8080 MİB, bu saykıl boyunca hangi tip işlemin gerçekleşeceğini gösteren "durum" bilgisini kendi veri yolu üzerine çıkarır. 8228/8238, $\overline{STSTB}$ girişi "alçak" düzeye gittiğinde, bu bilgiyi Durum Mandalına saklar. Kapılama Dizisine bağlı bulunan Durum Mandalının çıkışları, aynı zamanda Denetim Sinyali üretiminin bir parçasıdır.

Kapılama Dizisi

Kapılama Dizisi; Durum Mandalının çıkışları [illegible] MİB'den gelen sinyalleri (DBIN, $\overline{WR}$ ve HLDA) [illegible] rak denetim sinyalleri ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{I/OR}$, [illegible] $\overline{INTA}$) üretir.

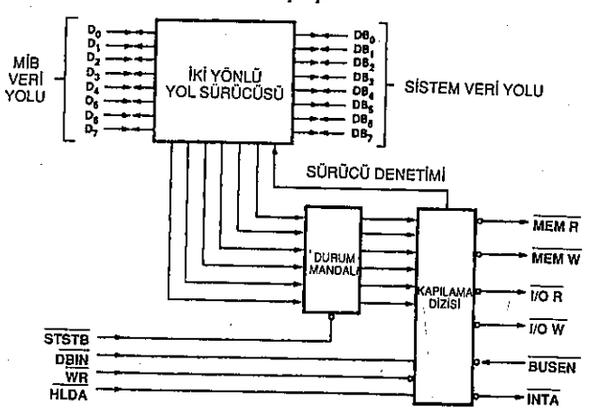
"Okuma" denetim sinyalleri ($\overline{MEMR}$, $\overline{I/OR}$ ve $\overline{IN}$[illegible]) gun durum biti veya bitleri ile 8080 MİB'den gelen [illegible] girişinin mantıksal birleşiminden türetilmiştir. [illegible] 8228'in "yazma" denetim sinyalleri ($\overline{MEMW}$, [illegible] uygun Durum Biti (veya bitleri) ile 8080 MİB'den [illegible] $\overline{WR}$ girişinin mantıksal birleşiminden türetilmiştir. [illegible] den gelen yazma sinyalleri, büyük sistemlerin zam[illegible] ma denetimi için geliştirilmiştir.

Bütün denetim sinyalleri "alçakta-etkindir" ve MC[illegible] ailesi; RAM, ROM ve G/Ç birimlerine doğrudan bağ[illegible] bilir.

$\overline{INTA}$ denetim sinyali normalde "kesme komut port[illegible] yol üzerine kapılamak için kullanılır. Bu, aynı zaman[illegible] 8228/38'e özel bir olanak sağlar. Küçük sistemler[illegible] olduğu gibi kesme yapısında yalnızca bir temel vek[illegible] ihtiyaç duyulursa, 8228/38 uygun zamanda, yola, otom[illegible] tik olarak RST 7 komutunu ekler. Bu seçeneğin kullan[illegible]ması için, 8228/38'in $\overline{INTA}$ çıkışı (23 no.lu bacak), bac[illegible] olarak 1 K Ohm'luk seri direnç yardımıyla +12 Volt[illegible] kaynağa bağlanır. Gerilim, 8228/38'in dahili olarak dol[illegible] algılanır ve DBIN girişi etkinken ve kesme kabul edil[illegible] ğinde, RST 7 komutunu yol üzerinde kapılamak için mantık devresi "ayarlanır". Bu olanak, kesme komut portu gibi, ek eleman ilave yapılmaksızın bir kesme vektörü sağlar.

Kesme komutu olarak CALL komutu kullanılacağı zaman, 8228/38 üç baytın her biri için $\overline{INTA}$ darbesi üretecektir.

Kapılama Dizisine $\overline{BUSEN}$ girişi (Bus Enable-Yol Yetkilendirme); mantıksal "bir" düzeyinde iken, veri yolu çıkış ve denetim sinyali tamponlarını yüksek empedans durumuna çeken asenkron bir giriştir. Eğer $\overline{BUSEN}$, "sıfır" ise, veri tamponları ve denetim sinyallerinin normal biçimde çalışır.



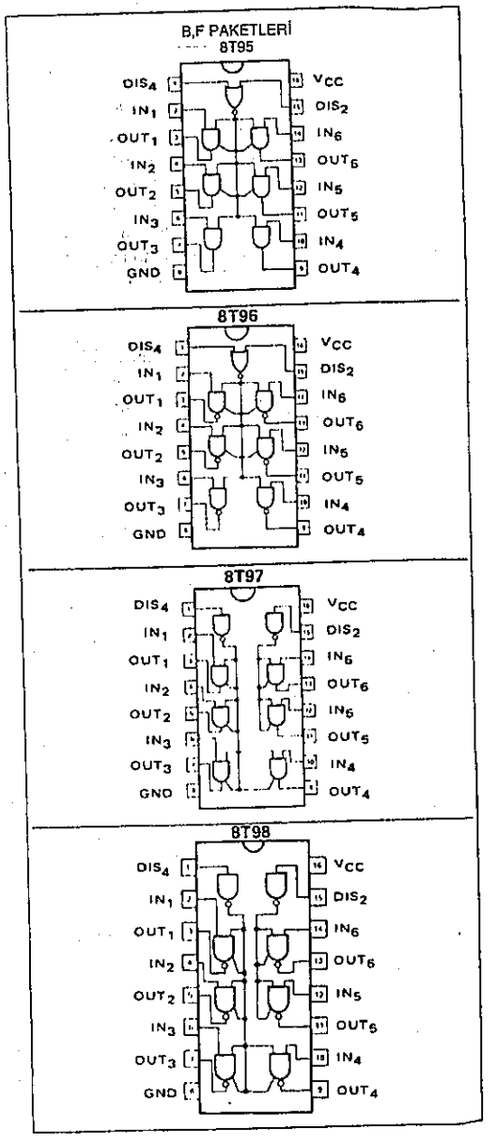
8228/38 BLOK DİYAGRAM 28/38 BLOK DİYAGRAM

Signetics Yüksek-Hızlı Üç-Durumlu Altı Tampon ve Tersleyici

SAYISAL 8T SERİSİ ARABİRİMİ TTL/MSI

ÜRÜN, YALNIZCA 0°C-75°C SICAKLIK ARALIĞINDA KULLANILABİLİR

BACAK KONFİGÜRASYONLARI (Üstten Görünüş)



[illegible]LAMA

[illegible] tanımlanmış Üç-Durumlu Yol Arabirimi Ele[illegible]rından her biri, düşük akımlı PNP girişlerine sahip[illegible] çok yüksek hız için Schottky TTL teknolojisi ile [illegible]nmıştır. Birimler, TTL/DTL veya MOS/CMOS dü[illegible]rini, üç-durumlu TTL yol düzeylerine dönüştürmek [illegible]kullanılır. Maksimum sistem esnekliği için 8T95 ve [illegible]7 mantıksal tersleme yapmaksızın çıkışları sağlar[illegible] 8T96 ve 8T98 girişlerin mantıksal tümleyenin çıkışta [illegible]ülmesini sağlar. 9T95 ve 8T96 bütün altı birim için [illegible]k denetim hattı içerirken, 8T97 ve 8T98, herbiri dört [illegible]nı denetleyen iki adet denetim girişine sahiptir.

ÖZELLİKLERİ

- DÜŞÜK AKIMLI PNP GİRİŞLERİ (400 A)
- YÜKSEK HIZLI SCHOTTKY TTL TASARIM (TİPİK 8 nsn)
- TTL/DTL, MOS/CMOS UYUMLULUĞU
- DÜŞÜK GÜÇ HARCANIMI
  - 8T95/97 TİPİK 325 mW
  - 8T96/98 TİPİK 295 mW
- YÜKSEK HIZ İÇİN AŞAĞIDAKİ TİPLERLE DEĞİŞTİRİLEBİLİR
  - DM8095 = 8T95
  - DM8096 = 8T96
  - DM8097 = 8T97
  - DM8098 = 8T98

DOĞRULUK TABLOLARI

8T95			
YETKİSİZLEME YETS1	GİRİŞ YETS2	GİRİŞ	ÇIKIŞ
0	0	0	0
0	0	1	1
0	1	x	Yük-Emp.
1	0	x	Yük-Emp.
1	1	x	Yük-Emp.

8T96			
YETKİSİZLEME YETS1	GİRİŞ YETS2	GİRİŞ	ÇIKIŞ
0	0	0	1
0	0	1	0
0	1	x	Yük-Emp.
1	0	x	Yük-Emp.
1	1	x	Yük-Emp.

DOĞRULUK TABLOLARI (devamı)

8T97			
YETKİSİZLEME YETS4	GİRİŞ YETS2	GİRİŞ	ÇIKIŞ
0	0	0	0
0	0	1	1
x	1	x	Yük-Emp. *
1	x	x	Yük-Emp. **

8T98			
YETKİSİZLEME YETS4	GİRİŞ YETS2	GİRİŞ	ÇIKIŞ
0	0	0	1
0	0	1	0
x	1	x	Yük-Emp. *
1	x	x	Yük-Emp. **

* Yalnızca 5-6 çıkışlar ** Yalnızca 1-4 çıkışlar x = İlgisiz

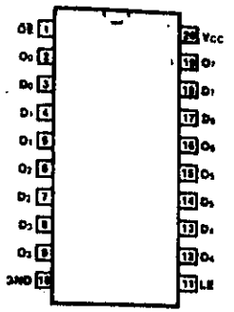
DC ELEKTRİKSEL KARAKTERİSTİKLER (Önerilen edilen gerilim ve sıcaklık değerlerinde)

KARAKTERİSTİKLER	SINIRLAR				TEST KOŞULLARI		NOTLAR
	MİN.	TİPİK	MAKS.	BİRİMİ			
Mantıksal "1" Giriş Gerilimi	2.0			V	V_{CC} = Min T_A = 25 °C		
Mantıksal "0" Giriş Gerilimi			0.8	V	V_{CC} = Min T_A = 25 °C		
Mantıksal "1" Çıkış Gerilimi	2.4			V	V_{CC} = Min I_O = -5.2 mA		6
Mantıksal "0" Çıkış Gerilimi			0.5	V	$I_{çık}$ = 48 mA		7
Üçüncü Durum Giriş Akımı			-40	µA	V_{CC} = Max	V_{gir} = 0.5 V DIS = 2.0 V	
Üçüncü Durum Çıkış Akımı			40 -40	µA µA	V_{CC} = Max	V_o = 2.4 V V_o = 0.5 V	
Mantıksal "1" Giriş Akımı			40	µA	V_{CC} = Max	V_{gir} = 2.4 V	
Mantıksal "0" Giriş Akımı			-400	µA	V_{CC} = Max	V_{gir} = 0.5 V DIS = 0.5 V	
Çıkış Kısa Devre Akımı	-40	-80	-115	mA	V_{CC} = Max	V_o = 0 V	9,10
Besleme Akımı (her birim) 8T95/97 8T96/98		65 59	98mA 89mA	mA	V_{CC} = Max		
Giriş Gerilimi Anma Değeri	5.5			V	I_{gir} = 1 mA		
Giriş Kenetleme Gerilimi				V	V_{CC} = Min	I_{gir} = -12 mA	
Çıkış V_{CC} Kenetleme Gerilimi				V	V_{CC} = 0V	I_o = 12 mA	
Çıkış Toprak Kenetleme Gerilimi				V	V_{CC} = 0V	I_{gir} = -12 mA	

Ek I Sekizli Saydam Mandal

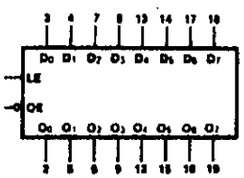
54F/74F373
SEKİZLİ SAYDAM MANDAL
(Üç-Durumlu Çıkışlara Sahip)

BAĞLANTI ŞEMASI



TANIMLAMA - 'F373, yol düzenli sistem uygulamaları için üçdurumlu çıkışa sahip sekiz adet mandaldan oluşur. Mandal Yetkileme (LE), YÜKSEK düzeyde olduğunda, flip-flopların girişlerindeki veri çıkışlarında görünür. LE, ALÇAK düzeyde olduğunda ise, girişlerdeki veri, "kurma süresinde" mandallanır. Çıkış Yetkileme ($\overline{OE}$), ALÇAK düzeyde olduğunda, veri yolda görünür. $\overline{OE}$, YÜKSEK düzeyde olduğunda, yol çıkışı, yüksek emdepans durumuna geçecektir.

MANTIKI ŞEMASI



V_{CC} = bacak 20
GND = bacak 10

- **TEK PAKETTE SEKİZ ADET MANDAL**
- **YOL ARABİRİMİ İÇİN ÜÇ-DURUMLU ÇIKIŞLAR**

SİPARİŞ KODU: 5. Bölüme bakın.

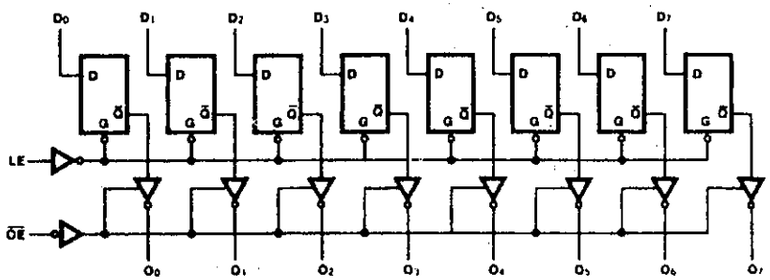
Paket Türü	TİCARİ SINIF	ASKERİ SINIF	PAKET TİPİ
	V_{CC} = +0.5V ±5% T_A = 0 °C to +70°C	V_{CC} = +0.5V ±10% T_A = -55 °C to +125°C	
Plastik DIP (P)	74F373PC		9Z
Seramik DIP (D)	74F373DC	54F373DM	4E
Flatpak (F)	74F373FC	54F373FM	4F

GİRİŞ YÜKLEMESİ/ÇIKIŞ-YELPAZESİ: U.L. tanımlamaları için 2. bölüme bakın.

BACAK ADLARI	AÇIKLAMA	54F/74F (U.L.) YÜKSEK/ALÇAK
D_0 - D_7	Veri Girişleri	0.5/0.375
LE	Mandal Yetkileme Girişi (YÜKSEK'te-Etkin)	0.5/0.375
OE	Çıkış Erkleme Girişi (ALÇAK'ta-Etkin)	0.5/0.375
O_0 - O_7	Üç-Durumlu Mandal Çıkışları	25/12.5

İŞLEVSEL TANIMLAMA - 'F373, 3-durumlu çıkış tamponlarına sahip sekiz adet D tipi mandal içerir. Mandal Yetkileme (LE) girişi YÜKSEK düzeyde olduğunda, Dn girişlerindeki veri, mandallara girer. Bu durumda mandallar saydamdır. Yani herhangi bir anda mandalın D girişi değiştiği zaman, mandalın çıkışı da değişecektir. LE, ALÇAK düzeyde olduğu zaman mandallar; LE'nin YÜKSEK'ten ALÇAK düzeye geçisinden hemen önce, mandalın D girişlerinde kurma süresinde mevcut bulunan bilgiyi saklarlar. Üç durumlu tamponlar, Çıkış Yetkileme ($\overline{OE}$) girişiyle denetlenir. $\overline{OE}$ ALÇAK düzeyde olduğu zaman, tamponlar iki durumlu çalışma modundadır. $\overline{OE}$ YÜKSEK düzeydeyken tamponlar, yüksek empedans moduna geçer, fakat bu durum, mandallara yeni veri girişine engel olmaz.

MANTIK DİYAGRAMI



Bu diyagramın, yalnızca mantıksal işlemlerin anlaşılmasını sağlamak üzere verilmiştir, yayılma gecikmelerinin hesaplanması için kullanılmamalıdır.

ÇALIŞMA SICAKLIK ARALIĞINDAKİ DC KARAKTERİSTİKLER (aksi belirtilmedikçe)

SEMBOL	PARAMETRE	54F/74F			BİRİMİ	KOŞULLAR
		Min.	Tipik	Maks.		
ICC	Güç Kaynağı Akımı (Tüm Çıkışlar KAPALI)		35	55	mA	VCC = Maks, OE = 4.5 V Dn, LE = Toprak

AC KARAKTERİSTİKLER: Yük konfigürasyonları ve dalgabiçimleri için 2. bölüme bakınız.

SEMBOL	PARAMETRE	54F/74F			54F		74F		BİRİMİ	ŞEK. NO.
		TA = +25°C VCC = +5.0V CL = 50pF			TA, VCC = ASK. CL = 50pF		TA, VCC = TİC. CL = 50pF			
		Min.	Tipik	Maks.	Min.	Maks.	Min.	Maks.		
tPLH	Dn'den On'ye	3.0	5.3	7.0			3.0	8.0	nsn	2-17
tPHL	Yayınım Gecikmesi	2.0	3.7	5.0			2.0	6.0		2-19
tPHL	LE'den On'ye	5.0	9.0	11.5			5.0	13.0	nsn	2-17
tPHL	Yayınım Gecikmesi	3.0	5.2	7.0			3.0	8.0		2-21
tPZH	Çıkış Yetkileme Süresi	2.0	5.0	11.0			2.0	12.0	nsn	2-25
tPZL		2.0	5.6	7.5			2.0	8.5		2-26 2-27
tPZH	Çıkış Yetkisizleme Süresi	2.0	4.5	6.5			2.0	7.5	nsn	2-25
tPZL		2.0	3.8	5.0			2.0	6.0		2-28 2-27

Ek J Motorola MC6820 çevresel arabirim adaptörü

ÇEVRESEL ARABİRİM ADAPTÖRÜ (PIA)

MC6820 Çevresel Arabirim Adaptörü, MC6800 mikroişlemci (MİB) birimiyle çevrebirimlerinin arabirim üzerinden bağlantı kurmasını sağlayan genel bir araçtır. Bu birim, iki adet 8 bitlik iki-yönlü çevrebirim veri yolu ve dört denetim hattı yardımıyla MİB ile çevrebirimlerini birbirine bağlar. Birçok çevrebirimine arabağlantı kurmak için dış mantık devrelerine gerek yoktur.

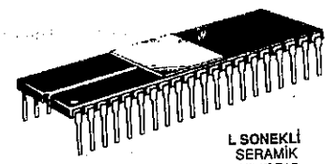
PIA'nın işlevsel konfigürasyonu, sistemin kullanıma hazırlanması sırasında MİB tarafından programlanır. Çevresel veri hatlarından her biri, giriş veya çıkış olarak ve dört denetim/kesme hattından her biri de, denetim modlarından birine programlanabilir. Bu, arabirimin genel amaçlı çalışmasında yüksek derecede esnekliğe olanak tanır.

- MİB ile Veri İletişimi için 8-bitlik İki-Yönlü Veri Yolu
- Çevrebirimlerine Bağlantı Kurmak için İki Adet İki-Yönlü 8-bitlik Veri Yolu
- İki Adet Programlanabilir Denetim Kaydedicisi
- İki Adet Programlanabilir Veri Yönü Kaydedicisi
- İkisi Çevrebirimi Denetim Çıkışı olarak kullanılabilen, Ayrı Ayrı Denetlenebilen Dört Adet Kesme Giriş Hattı
- Giriş ve Çıkış Çevrebirim İşlemleri için Onay Denetimi Mantık Devresi
- Yüksek-Empedanslı, 3-Durumlu ve Doğrudan Transistörle Sürülebilen Çevresel Hatları
- Program Denetimli Kesme ve Kesmeyi Yetkisizleme Olanağı
- A Tarafındaki Çevresel Hatların CMOS Sürme Yeteneği

MOS

(N-KANAL, SİLİKON-KAPI)

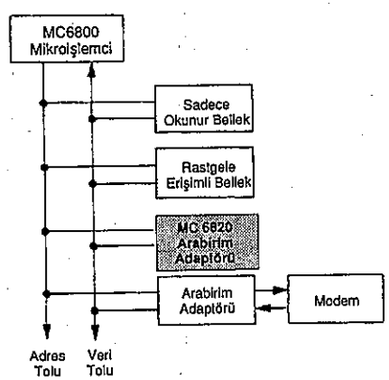
ÇEVRESEL BİRİM ADAPTÖRÜ



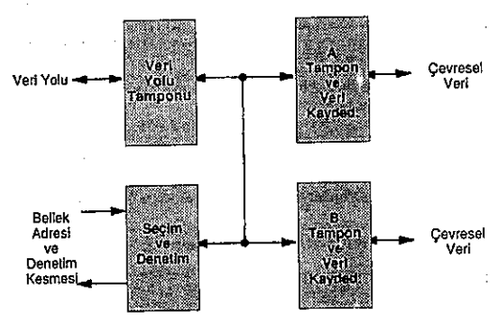
L SONEKLİ SERAMİK KILIF 715

GÖSTERİLMEYEN: P SONEKLİ PLASTİK PAKET KILIF 711

M6800 MİKROBİLGİSAYAR AİLESİ BLOK DİYAGRAMI



M6820 ÇEVRESEL ARABİRİM ADAPTÖRÜ BLOK DİYAGRAMI



ELEKTRİKSEL KARAKTERİSTİKLER (V_{CC}= 5.0 V ± 5%, V_{SS} = 0, aksi belirtilmedikçe T_A=0 - 70°C)

Karakteristik		Sembol	Min.	Tip.	Maks.	Birim
Yüksek Giriş Gerilimi	Yetkilendirme Diğer Girişler	V_{IH}	V_{SS} + 2.4 V_{SS} + 2.0	- -	V_{CC} V_{CC}	Vdc
Alçak Giriş Gerilimi	Yetkilendirme Diğer Girişler	V_{IL}	VSS - 0.3 V_{SS} - 0.3	- -	V_{SS} + 0.4 V_{SS} + 0.8	Vdc
Giriş Kaçak Akımı (V_{gir}=0 - 5.25 Vdc)	R/W, Sıfırlama, RS0, RS1, CS0, CS1, CA1 CB1, Yetkilendirme	I_{in}	-	1.0	2.5	µAdc
Üçüncü-Durum (KAPALI durum) Giriş Akımı (V_{gir}=0.4-2.4 Vdc)	D0-D7, PB0-PB7, CB2	I_{TSI}	-	2.0	10	µAdc
Yüksek Giriş Akımı (V_{IH} = 2.4 Vdc)	PA-PA7, CA2	I_{IH}	-100	-250	-	µAdc
Alçak Giriş Akımı (V_{IL} = 0.4 Vdc)	PA-PA7, CA2	I_{IL}	-	-1.0	-1.6	mAdc
Yüksek Çıkış Gerilimi ($I_{yük}$=-205 µAdc, Yetkileme Darbe Genişliği <25 µs) ($I_{yük}$=-100 µAdc, Yetkileme Darbe Genişliği <25 µs)	D0-D7 Diğer Çıkışlar	V_{OH}	VSS + 2.4 V_{SS} + 2.4	- -	- -	Vdc
Alçak Çıkış Gerilimi ($I_{yük}$=1.6 mAdc, Yetkileme Darbe Genişliği < 25 µs)		V_{OL}	-	-	VSS + 0.4	Vdc
Yüksek Çıkış Akımı (Kaynaklama) (V_{OH}=2,4 Vdc)	D0-D7 Diğer Çıkışlar	I_{OH}	-205 -100	- -	- -	µAdc
(V_O=1,5 Vdc, TTL dışındaki yükleri, örneğin Darlington devresini sürmek için akım)	PB0-PB7, CB2		-1.0	-2.5	-10	mAdc
Alçak Çıkış Akımı (Çekme) (V_{OH}=2,4 Vdc)		I_{OL}	1.6	-	-	mAdc
Çıkış Kaçak Akımı (KAPALI durum) (V_{OH}=2,4 Vdc)	IRQA, IRQB	V_{LOH}	-	1.0	10	µAdc
Güç Harcaması		P_D	-	-	650	mW
Giriş Kapasitansı (V_{gir} = 0, T_A = 25 °C, f = 1.0 MHZ)	Yetkilendirme D0-D7 PA0-PA7, PB0-PB7, CA2, CB2 R/W, Reset, RS0, RS1, CS0, CS1, CS2, CA1, CB1	C_{gir}	- - - -	- - - -	20 12.5 10 7.5	pF
Çıkış Kapasitansı (V_{gir} = 0, T_A = 25 °C, f = 1.0 MHZ)	IRQA, IRQB PB0-PB7	$C_{çık}$	- -	- -	5.0 10	pF
Çevresel Veri Kurma Süresi (Şekil 1)		t_{PDSU}	200	-	-	ns
Gecikme Süresi; Yetkilendirme'nin negatif geçişinden CA2'nin negatif geçişine (Şekil 2,3)		t_{CA2}	-	-	1.0	µs
Gecikme Süresi; Yetkilendirme'nin negatif geçişinden CA2'nin pozitif geçişine (Şekil 2)		t_{RS1}	-	-	1.0	µs
CA1 ve CA2 giriş sinyallerinin Yükselme ve Düşme Süreleri (Şekil 3)		t_r, t_f	-	-	1.0	µs
CA1'in aktif geçişinden CA2'nin pozitif geçişine Gecikme Süresi (Şekil 3)		t_{RS2}	-	-	2.0	µs
Gecikme Süresi; Yetkilendirme'nin negatif geçişinden Çevresel Veri geçerli'ye (Şekil 4.5)		t_{PDW}	-	-	1.0	µs
Gecikme Süresi; Yetkilendirme'nin negatif geçişinden Çevresel CMOS Veri Geçerli'ye (V_{CC}-30%V_{CC}, Şekil 4; Şekil 12, C Yükü)	PA0-PA7, CA2	t_{CMOS}	-	-	2.0	µs
Gecikme Süresi; Yetkilendirme'nin pozitif geçişinden CB2'nin negatif geçişine (Şekil 6, 7)		t_{CB2}	-	-	1.0	µs
Gecikme Süresi; Geçerli Çevresel Veri'den CB2'nin negatif geçişine (Şekil 5)		t_{DC}	20	-	-	µs
Gecikme Süresi; Yetkilendirme'nin pozitif geçişinden CB2'nin pozitif geçişine (Şekil 6)		t_{RS1}	-	-	1.0	µs
CB1 ve CB2 giriş sinyalleri için Yükselme ve Düşme Süreleri (Şekil 7)		t_r, t_f	-	-	1.0	µs
Gecikme Süresi; CB1'in aktif geçişinden CB2'nin pozitif geçişine (Şekil 7)		t_{RS2}	-	-	2.0	µs
Kesme'yi Bırakma Süresi, IRQA ve IRQB (Şekil 8)		t_{IR}	-	-	1.6	µs
Sıfırlama Alçak Düzey Süresi (Şekil 9)		t_{RL}	2.0	-	-	µs

* PIA adreslenmeden önce Sıfırlama (Reset) hattı mutlaka 1 sn süreyle YÜKSEK düzeyde olmalıdır.

AKSİMUM ANMA DEĞERLERİ

Anma değerleri	Sembol	Değeri	Birimi
leme Gerilimi	V_{CC}	-0.3 - +7.0	Vdc
iş Gerilimi	V_{gir}	-0.3 - +7.0	Vdc
şma Sıcaklık Aralığı	T_A	0 - +70	°C
klama Sıcaklık Aralığı	T_{sak}	-55 - +150	°C
direnç	θ JA	70	°C/W

Bu birimin içinde, girişleri, yüksek statik gerilim veya elektrik alanlarından kaynaklanan hasarlara karşı korumak için devreler bulunur. Bununla beraber, bu yüksek empedanslı devreye maksimum anma değerinden daha yüksek gerilimlerin uygulanmasını önlemek için normal önlemlerin alınması önerilir.

OL ZAMANLAMA KARAKTERİSTİKLERİ

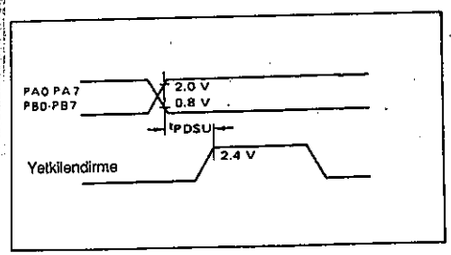
OKUMA (10 ve 12. Şekiller)

Karakteristikler	Sembol	Min	Tip.	Maks.	Birimi
Yetkilendirme Saykıl Süresi	t_{cycE}	1.0	-	-	µsn
Yetkilendirme Darbe Genişliği, YÜKSEK düzey	PW_{EL}	0.45	-	25	µsn
Yetkilendirme Darbe Genişliği, ALÇAK düzey	PW_{EL}	0.43	-	-	µsn
Kurma Süresi; Yetkilendirme'nin pozitif geçişinden, adres ve R/W geçerli'ye	t_{AS}	160	-	-	nsn
Veri Gecikme Süresi	t_{DDR}	-	-	320	nsn
Veri Tutma Süresi	t_H	10	-	-	nsn
Adres Tutma Süresi	t_{AH}	10	-	-	nsn
Yetkilendirme girişinin Yükselme ve Düşme Süresi	t_{ER}, t_{Ef}	-	-	25	nsn

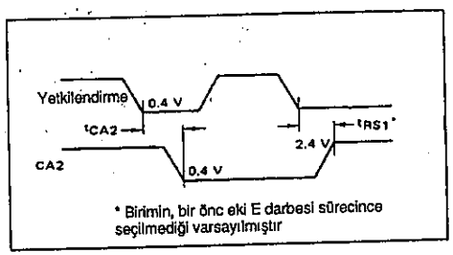
YAZMA (11 ve 12. Şekiller)

Yetkilendirme Saykıl Süresi	t_{cycE}	1.0	-	-	µsn
Yetkilendirme Darbe Genişliği, YÜKSEK düzey	PW_{EH}	0.45	-	25	µsn
Yetkilendirme Darbe Genişliği, ALÇAK düzey	PW_{EL}	0.43	-	-	µsn
Kurma Süresi; Yetkilendirme'nin pozitif geçişinden, adres ve R/W geçerli'ye	t_{AS}	160	-	-	nsn
Veri Gecikme Süresi	t_{DSW}	195	-	-	nsn
Veri Tutma Süresi	t_H	10	-	-	nsn
Adres Tutma Süresi	t_{AH}	10	-	-	nsn
Yetkilendirme girişinin Yükselme ve Düşme Süresi	t_{Er}	t_{Ef}	-	25	nsn

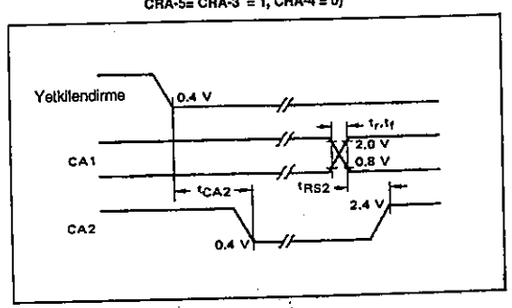
ŞEKİL 1- ÇEVRESEL VERİ KURMA ZAMANI
(Okuma Modu)



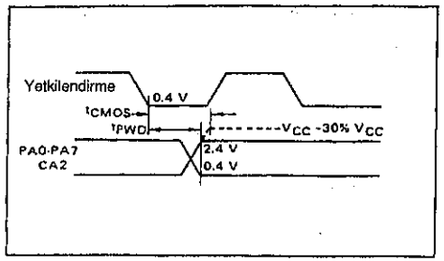
ŞEKİL 2- GAZ GECİKME SÜRESİ
CRA-5= CRA-3 = 1, CRA-4 = 0)



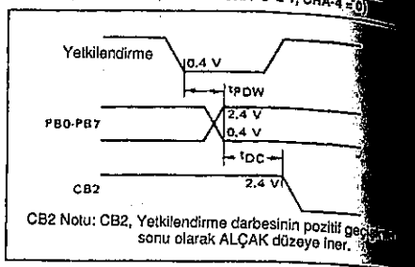
ŞEKİL 3- GAZ GECİKME SÜRESİ
CRA-5= CRA-3 = 1, CRA-4 = 0)



ŞEKİL 4- ÇEVRESEL CMOS VERİSİ GECİKME SÜRELERİ
(Yazma modu, CRA-5 = CRA -3 = 1, CRA-4 = 0)

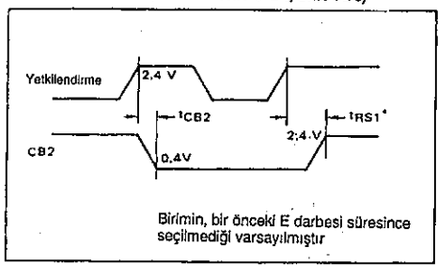


ŞEKİL 5 - ÇEVRESEL VERİ VE CB2 GECİKME SÜRELİ
(Yazma modu, CRA-5 = CRA -3 = 1, CRA-4 = 0)



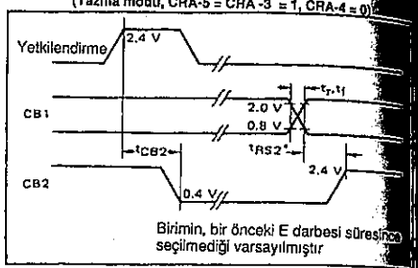
CB2 Notu: CB2, Yetkilendirme darbesinin pozitif geç... sonu olarak ALÇAK düzeye iner.

ŞEKİL 6 - CB2 GECİKME SÜRESİ
(Yazma modu, CRA-5 = CRA -3 = 1, CRA-4 = 0)



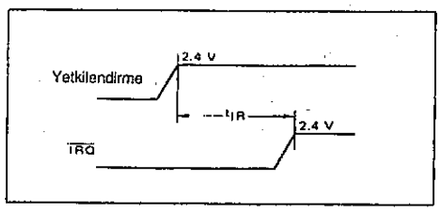
Birimin, bir önceki E darbesi süresince seçilmediği varsayılmıştır

ŞEKİL 7 - CB2 GECİKME SÜRESİ
(Yazma modu, CRA-5 = CRA -3 = 1, CRA-4 = 0)

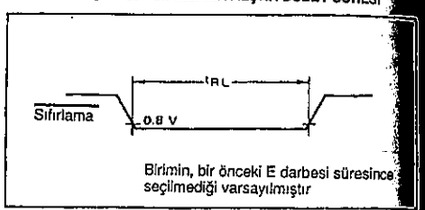


Birimin, bir önceki E darbesi süresince seçilmediği varsayılmıştır

ŞEKİL8 - IRQ BIRAKMA SÜRESİ

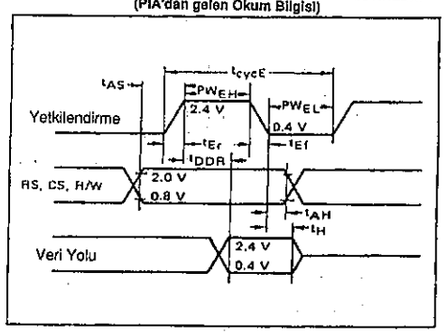


ŞEKİL9 - $\overline{\text{SIFIRLAMA}}$ ALÇAK DÜZEY SÜRESİ

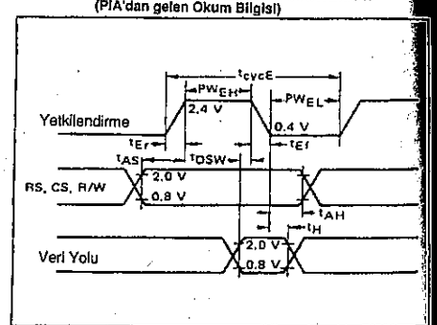


Birimin, bir önceki E darbesi süresince seçilmediği varsayılmıştır

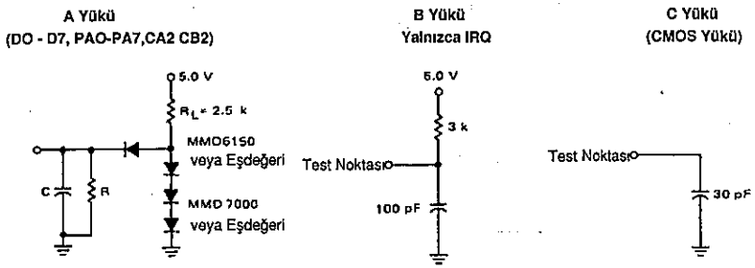
ŞEKİL10 - YOL OKUMA ZAMANLAMA KARAKTERİSTİKLERİ
(PIA'dan gelen Okum Bilgisi)



ŞEKİL11 - YOL OKUMA ZAMANLAMA KARAKTERİSTİ
(PIA'dan gelen Okum Bilgisi)



ŞEKİL 12 -YOL ZAMANLAMASI TEST YÜKLERİ



C = 130 pF, DO-D7 için
= 30 pF, PAO-PA7, PBO-PB7, CA2 ve CB2 için
R = 11.5 kΩ,DO-D7 için
= 24 kΩ , PAO-PA7, PBO-PB7, CA2 ve CB2 için

MİB İÇİN ARABİRİM SİNYALLERİ

PIA, MC6800 MİB'ne 8 bitlik iki yönül veri yolu, üç adet yonga seçim, iki adet kaydedici seçim, iki adet kesme istek, okuma/yazma, yetkilendirme ve sıfırlama hattıyla bağlanır. MC6800 VMA çıkışıyla birlikte bu sinyaller, MİB'nin PIA üzerinde tam denetim yapabilmesini sağlarlar. VMA, PIA yongasının seçiminde MİB adres hatlarıyla birlikte kullanılmalıdır.

PIA İki-Yönlü Veri (D0-D7) - İki-yönlü veri hatları (D0-D7), PIA ile MİB arasında bilgi aktarımına imkan verir. Veri yolu çıkış sürücüleri, MİB'nin PIA okuma işlemindeki durum dışında, yüksekempedans (KAPALI) durumunda kalabilen, üç durumlu sürücülerdir. PIA'nın, okuma işlemi için seçilmesi halinde, okuma/yazma hattı OKUMA (YÜKSEK düzey) durumundadır.

PIA Yetkilendirme (E) - Yetkilendirme darbesi, E; PIA'ya uygulanan zamanlama sinyalidir. Diğer bütün sinyallerin zamanlamaları, E darbesinin düşen veya yükselen kenarlarını referans alırlar. Bu sinyal normalde MC6800 Ø2 saatinin türevi olacaktır.

PIA Okuma/Yazma (R/W) - Bu sinyal MİB tarafından, veri yolu üzerindeki veri aktarımının yönünü denetlemek için üretilir. Eğer PIA seçilmişse, PIA okuma/yazma hattındaki ALÇAK düzey durumu, giriş tamponlarınının yetkilenmesine ve E sinyali boyunca MİB'den PIA'ya bilgi aktarımına izin verir. Okuma/yazma hattındaki YÜKSEK düzey durumu, PIA'yı veri yoluna bilgi aktarımı için düzenler. Uygun adresler ve Yetkilendirme (E) darbesi hazır olduğunda, PIA çıkış tamponları yetkilendirilmiş olur.

$\overline{\text{Sıfırlama}}$ - ALÇAK'ta ETKİN $\overline{\text{Sıfırlama}}$ hattı, PIA içindeki bütün kaydedici bitlerini mantıksal sıfır (ALÇAK) durumuna getirmek için kullanılır. Bu hat, sistemin çalışması sırasında, açılışta sıfırlama veya ana sıfırlama olarak kullanılabilir.

PIA Yonga Seçim (CS0, CS1 ve $\overline{\text{CS2}}$) - Bu üç sinyal PIA'yı seçmek için kullanılır. PIA'nın seçilebilmesi için CS0 ve CS1, YÜKSEK ve CS2 de ALÇAK durumda olmalıdır. Ardından, veri akışı, yetkilendirme ve okuma/yazma sinyallerinin denetimi altında sağlanabilir. Yonga seçim hatları E darbesi süresince kararlı olmalıdır. PIA seçim hatlarından herhangi biri aktif olmadığı sürece yonga seçilmez.

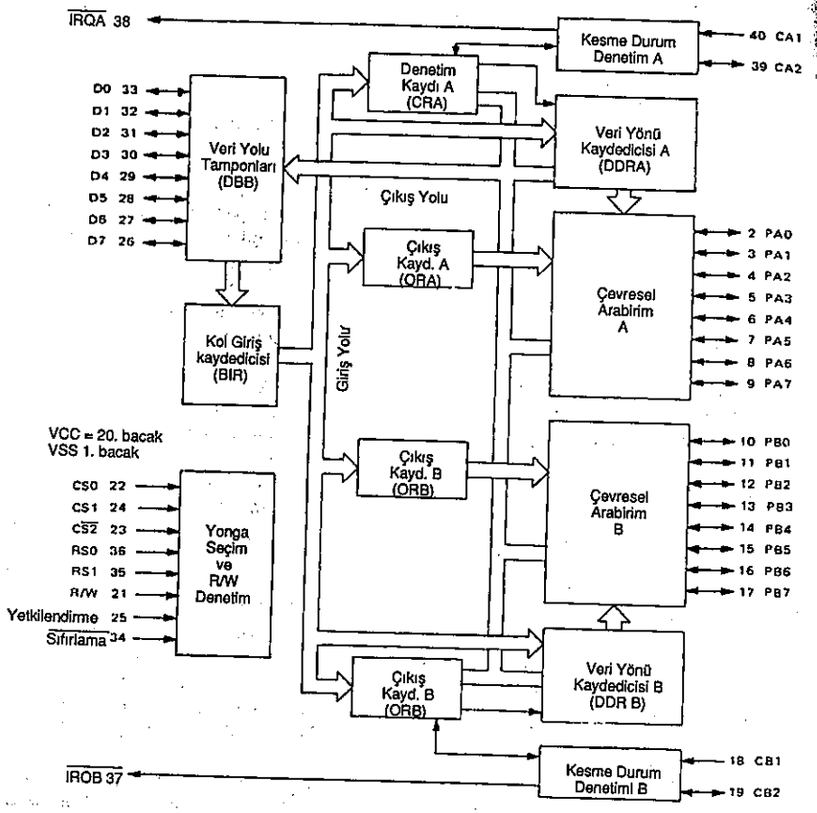
PIA Kaydedici Seçimi (RS0 ve RS1) - İki kaydedici seçim hattı, PIA içindeki çeşitli kaydedicileri seçmek için kullanılır. Bu hatlar iç denetim kaydedicileriyle birlikte yazılabilen veya okunabilen özel kaydedicileri seçmek için kullanılır.

Kaydedici ve yonga seçim hatları, okuma ve yazma çevriminde E darbesi boyunca kararlı olmalıdır.

$\overline{\text{Kesme İsteği}}$ ($\overline{\text{IRQA}}$ ve $\overline{\text{IRQB}}$) - ALÇAK'ta-ETKİN $\overline{\text{Kesme İstek}}$ ($\overline{\text{IRQA}}$ ve $\overline{\text{IRQB}}$) hatları, doğrudan veya kesme öncelik devreleri aracılığıyla MİB'ne kesmeyi bildirir. Bu hatlar "açık akaç" (yongaya bağlı yükün olmadığı durum) özelliğine sahiptir. Bu özellik, bütün kesme istek hatlarının birbiriyle telli-VEYA düzeninde bağlanmasına izin verir.

Her bir **$\overline{\text{Kesme İstek}}$** hattı, bu hattın alçak düzeye gitmesine neden olan iki adet iç kesme bayrak bitine sahiptir. Her bayrak biti, özel çevrebirimi kesme hattıyla bağlantılıdır. Ayrıca, PIA içinde, çevresel birimlerden gelen özel kesmeleri yasaklamak için kullanılabilen dört adet kesme yetkilendirme biti sağlanmıştır.

GENİŞLETİLMİŞ BLOK DİYAGRAM



MİB'nin kesmeye hizmet vermesi, her bir PIA içinde-ki iki adet denetim kaydedicisinde kurulu bulunan kesme bayraklarını sırayla okuyan ve test eden, öncelik esasına dayalı bir yazılım altyordamıyla gerçekleştirilebilir.

Kesme bayrakları; bunlara karşılık gelen veri kaydedicisinin, bir MİB Çevresel Veri Okuma İşlemi ile okunması sonucu sıfırlanır. Sıfırlama sonrasında, kesme bayrak bitlerinin, E darbesi boyunca PIA pasif duruma gelinceye kadar kurulmasına izin verilmez. E darbesi, kesme denetim hatlarının (CAI, CA2, CB1, CB2) durumu için de kullanılır. Bu hatlar, kesme girişleri olarak kullanılacağı zaman, kenara-duyarlı devrenin durumu için, kesme giriş sinyalinin aktif olmayan kenarından aktif olan kenarına kadar, en azından bir E darbesi mutlaka oluşmalıdır. Eğer kesme bayrağı yetkilendirilmişse ve kenara-duyarlı devre uygun durumdaysa, kesme bayrağı, kesme giriş bacağında ortaya çıkacak bir sonraki sinyalin aktif geçişinde kurulacaktır.

PIA ÇEVRESEL ARABİRİM HATLARI

PIA, çevresel birimlere arabağlantı kurmak için dört adet kesme/denetim hattı ve iki adet 8 bitlik iki-yönlü veri yolu içerir.

A Bölümü Çevresel Verisi (PA0-PA7) - Çevresel veri hatları giriş ya da çıkış olacak şekilde programlanabilir. Herhangi bir veri hattının çıkış olarak programlanabilmesi için, çevresel veri hatlarına karşılık gelen veri yönü kaydedici bitleri "1"e kurulur. Veri yönü kaydedici bitindeki "0" düzeyi, bu kaydediciye karşılık gelen çevresel veri hatlarının bir giriş gibi işlev görmesine neden olur. MİB Çevrebirimi Veri Okuma İşlemleri sırasında, giriş olarak programlanmış çevrebirimi hatlarındaki veri, doğrudan ilgili MİB Veri yolu hatlarında görünür. Giriş modunda, bu hatlardaki yukarıya çekme dirençleri maksimum bir standart TTL yükü olarak görülür.

A Çıkış kaydedicisindeki veriler, çıkış olarak programlanmış veri hatlarında görülecektir. Kaydediciye yazılan mantıksal "1", ilgili veri hattında "YÜKSEK" düzeye, bu kaydediciye yazılan "0" ise "ALÇAK" düzeye neden olacak. A Çıkış kaydedicisindeki veri, ilgili hatlar çıkış olarak programlandığında, MİB'nin "A Çevrebirimi Verisini Oku" işlemi yardımıyla okunabilir. Eğer çevrebirimi veri hatlarında mantıksal "1"e karşılık gelen çıkış, 2 volttan daha büyük ve mantıksal "0"a karşılık gelen çıkış da 0,8 volttan daha küçükse, veri uygun bir şekilde okunabilir. Çıkış hatlarını yüklemek, bu hatlardaki gerilim tam değerine ulaşamadığı için, okuma işlemi ile MİB'ne aktarılan verilerin, çıkış kaydedicisi A'nın sıralı bitlerinin içeriklerinden farklı olmasına neden olur.

B Bölümü Çevresel Verisi (PB0-PB7) - PIA'nın B bölümündeki veri hatları PA0-PA7'ye benzer şekilde giriş veya çıkış olarak programlanabilir. Bununla beraber, bu hatların çıkış tamponlarının sürme yeteneği PA0-PA7 hatlarınınkinden farklıdır. Çevrebirimi veri hatları giriş olarak kullanıldığında, üçdurumluluk yetenekleri, bu hatların yüksek empedans durumuna geçebilmelerini sağlar. Ayrıca, çıkış olarak programlanmış PB0PB7 çevrebirimi veri hatlarındaki bilgi, gerilimin değeri 2 voltun altında olsa bile "YÜKSEK" düzey olarak algılanacaktır. Çıkış olarak bu hatlar, standart TTL uyumludur ve doğrudan bir transisörü sürmek için 1.5 V'ta 1 miliamperlik akım akıtan bir kaynak gibi kullanılabilir.

Kesme Girişi (CA1 ve CB1) - CA1 ve CB1 çevrebirimi giriş hatları, işlevi; sadece denetim kaydedicisindeki kesme bayraklarını kurmak olan giriş hatlarıdır. Bu sinyallerin aktif geçişi iki adet denetim kaydedicisi tarafından programlanabilir.

Çevrebirimi Denetimi (CA2) - Çevrebirimi Denetim hattı CA2, kesme girişi veya çevrebirimi denetim çıkışı olarak programlanabilir. Çıkış olarak, bu hat standart TTL uyumludur, giriş olarak bu hattaki yukarı çekme dirençleri bir standart TTL yükü olarak görülür. Bu sinyalin işlevi, denetim kaydedicisi A ile programlanır.

Çevrebirim Denetimi (CB2) - Çevrebirimi Denetim hattı CB2'de, kesme girişi veya çevrebirimi denetim çıkışı olarak programlanabilir. Giriş olarak, bu hat yüksek giriş empedansına sahiptir ve standart TTL uyumludur. Çıkış olarak, standart TTL uyumludur ve doğrudan bir transistörü sürmek için 1,5 V'ta 1 miliamperlik akım akıtan bir kaynak gibi kullanılabilir. Bu hat, denetim kaydedicisi B tarafından programlanır.

NOT: $\overline{\text{Sıfırlama}}$, aktif olmayan duruma giderken, denetim kaydedicisinin içindeki ilgili kesme bayraklarının değerlerinin değişmesine engel olmak için, $\overline{\text{Sıfırlama}}$ aktif durumda iken denetim hatlarının (CA1, CA2, CB1, CB2) mantık "1" durumunda tutulmaları öğütlenir. Sonraki sıfırlama'nın aktif olmayan duruma gidişlerinde, veri kaydedicilerinden okuma, istenmeyen kesme bayraklarının sıfırlanması için kullanılabilir.

İÇ DENETİMLER

PIA içinde, PIA'nın MİB veri yoluna erişebileceği altı ayrı konum vardır: İki adet Çevrebirimi Kaydedicisi, iki adet Veri Yönü Kaydedicisi ve iki adet Denetim Kaydedicisi. Bu konumların seçimi, Tablo 1'de de gösterildiği gibi RS0 ve RS1 girişleriyle birlikte Denetim Kaydedicisinin ikinci biti tarafından kontrol edilir.

Tablo 1- İÇ ADRESLEME

RS1	RS0	Denetim Kaydedicisi Bit CRA-2	Denetim Kaydedicisi Bit CRB-2	Seçilen Konum
0	0	1	X	Çevrebirimi Kaydedicisi A
0	0	0	X	Veri Yönü Kaydedicisi A
0	1	X	X	Denetim Kaydedicisi A
1	0	X	1	Çevrebirimi Kaydedicisi B
1	0	X	0	Veri Yönü Kaydedicisi B
1	1	X	X	Denetim Kaydedicisi B

X= Dikkate alınmaz

HAZIRLAMA

Sıfırlama hattındaki alçak düzey, tüm PIA kaydedicilerinin sıfırlanmasına neden olur. Bu, PA0-PA7, PB0-PB7, CA2 ve CB2'nin giriş olarak düzenlenmesini ve bütün kesmelerin yetkisiz kılınmasını sağlayacaktır. PIA, sıfırlamayı takip eden yeniden başlatma programı esnasında mutlaka konfigüre edilmelidir.

Veri yönü ve denetim kaydedicilerinin olası konfigürasyon ayrıntıları aşağıdaki gibidir.

VERİ YÖNÜ KAYDEDİCİLERİ (DDRA ve DDRB)

MİB, iki adet Veri Yönü Kaydedicisi ile, bunlara karşılık gelen çevrebirimi veri hatlarından geçen verilerin yönünü denetler. Veri Yönü Kaydedicisinin bitini "0" olarak kurmak, bu bite karşılık gelen çevrebirimi veri hattını giriş, "1" olarak kurmak ise, çıkış olarak programlar.

DENETİM KAYDEDİCİLERİ (CRA ve CRB)

İki adet denetim kaydedicisi (CRA ve CRB), MİB'e dört çevrebirimi denetim hattı CA1, CA2, CB1 ve CB2 işlemlerini kontrol olanağı verir. Buna ek olarak, kesme hatlarını yetkilendirme ve kesme bayraklarının durumunu görme olanağı sağlar. Uygun yonga ve kaydedici seçim sinyalleri uygulandığı zaman bu iki kaydedicinin (5 dahilolmak üzere) 0'dan 5'e kadar olan bitleri, MİB tarafından yazılabilir veya okunabilir. İki kaydedicinin 6 ve 7. bitleri yalnızca okunabilir ve CA1, CA2, CB1 veya CB2 denetim hatlarında oluşan dış kesmeler tarafından değiştirilebilir. Denetim sözcüğünün formatı, Tablo 2'de gösterilmiştir.

Tablo 2- DENETİM SÖZCÜĞÜ FORMATI

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	CA2 Denetimi			DDRA Erişimi	CA1 Denetimi	
	7	6	5	4	3	2	1	0
CRB	IRQB1	IRQB2	CB2 Denetimi			DDRB Erişimi	CB1 Denetimi	

Veri Yönü Erişimi Denetim Biti (CRA-2 ve CRB-2) - Her bir Denetimi Kaydedicisinin 2. biti, RS0 ve RS1'le uygun kaydedici seçim sinyalleri uygulandığında, Çevresel Arabirim Kaydedicisi ya da Veri Yönü Kaydedicisinden birinin seçimine izin verir.

Kesme Bayrakları (CRA.6, CRA.7, CRB.6 ve CRB.7)- Dört adet kesme bayrak biti, giriş olarak programlanmış dört adet kesme ve çevrebirimi denetim hatlarındaki sinyalin aktif geçişiyle kurulabilir. Bu bitler, doğrudan MİB Veri Yoluyla kurulamayabilirler ve uygun bölümdeki Çevrebirimi Veri Okuma işlemi tarafından dolaylı olarak sıfırlanırlar.

TABLO 3- KESME GİRİŞLERİ CA1 VE CB1'İN DENETİMİ

CRA-1 (CRB-1)	CRA-0 (CRB-0)	Kesme Girişi CA1 (CB1)	Kesme Bayrağı CRA-7 (CRB-7)	MİB Kesme İsteği $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	↓ Aktif	CA1 (CB1)'nin ↓ geçişinde "yüksek"e kurulur.	Yetkisiz kılınmıştır - $\overline{IRQ}$ "yüksek"te kalır.
0	1	↓ Aktif	CA1 (CB1)'nin ↓ geçişinde "yüksek"e kurulur.	Kesme bayrak biti CRA-7 (CRB-7) "yüksek"e gittiği zaman "alçak" düzeye gider.
1	0	↑ Aktif	CA1 (CB1)'nin ↑ geçişinde "yüksek"e kurulur.	Yetkisiz kılınmıştır - $\overline{IRQ}$ "yüksek"te kalır.
1	1	↑ Aktif	CA1 (CB1)'nin ↑ geçişinde "yüksek"e kurulur.	Kesme bayrak biti CRA-7 (CRB-7) "yüksek"e gittiği zaman "alçak" düzeye gider.

Notlar: 1. ↑ pozitif (alçaktan yükseğe) geçişi gösterir.
2. ↓ negatif (yüksekten alçaka) geçişi gösterir.
3. CRA-7 kesme bayrak biti, A veri kaydedicisinin ve CRB-7 ise, B veri kaydedicisinin MİB tarafından okunmasıyla sıfırlanır.
4. Eğer CRA-0 (CRB-0), kesme (Kesmeler yetkisizlenmiş) durumu ortaya çıktığında, ALÇAK ve ardından YÜKSEK'e getirilirse, $\overline{IRQA}$ ($\overline{IRQB}$), CRA-0 (CRB-0) bitlerine "bir" yazılmasından sonra ortaya çıkacaktır.

CA1 ve CB1 Kesme Giriş Hatlarının Denetimi (CRA-0, CRB-0, CRA-1 ve CRB-1) - Denetim kaydedicisinin en alt sıralı iki biti, kesme giriş hatları CA1 ve CB1'in denetimi için kullanılır. CRA-0 ve CRB-0 bitleri sırasıyla, MİB kesme sinyalleri $\overline{IRQA}$ ve $\overline{IRQB}$'nin yetkilendirilmesi için kullanılır. CRA-1 ve CRB-1 bitleri, CA1 kesme girişlerinin aktif geçişini belirler (Tablo 3).

TABLO 4 - KESME GİRİŞLERİ OLARAK CA2 VE CB2'NİN DENETİMİ CRA-5 (CRB-5)=ALÇAK

CRA-5 (CRB-5)	CRA-4 (CRB-4)	CRA-3 (CRB-3)	Kesme Girişi CA2 (CB2)	Kesme Bayrağı CRA-6 (CRB-6)	MİB Kesme İsteği $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	0	↓ Aktif	CA2 (CB2)'nin ↓ geçişinde "yüksek"e kurulur.	Yetkisiz kılınmıştır - $\overline{IRQ}$ "yüksek"te kalır.
0	0	1	↓ Aktif	CA2 (CB2)'nin ↓ geçişinde "yüksek"e kurulur.	Kesme bayrak biti CRA-6 (CRB-6) "yüksek"e gittiği zaman "alçak" düzeye gider.
0	1	0	↑ Aktif	CA2 (CB2)'nin ↑ geçişinde "yüksek"e kurulur.	Yetkisiz kılınmıştır - $\overline{IRQ}$ "yüksek"te kalır.
0	1	1	↑ Aktif	CA2 (CB2)'nin ↑ geçişinde "yüksek"e kurulur.	Kesme bayrak biti CRA-6 (CRB-6) "yüksek"e gittiği zaman "alçak" düzeye gider.

Notlar: 1. ↑ pozitif (alçaktan yükseğe) geçişi gösterir.
2. ↓ negatif (yüksekten alçaka) geçişi gösterir.
3. CRA-6 kesme bayrak biti, A veri kaydedicisinin ve CRB-6 ise, B veri kaydedicisinin MİB tarafından okunmasıyla sıfırlanır.
4. Eğer CRA-3 (CRB-3), kesme (Kesmeler yetkisizlenmiş) durumu ortaya çıktığında, ALÇAK ve ardından YÜKSEK'e getirilirse, $\overline{IRQA}$ ($\overline{IRQB}$), CRA-3 (CRB-3) bitlerine "bir" yazılmasından sonra ortaya çıkacaktır.

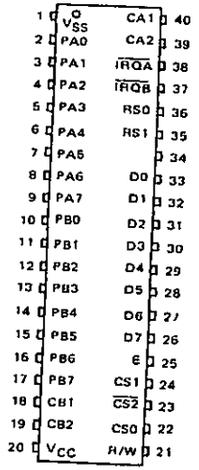
TABLO 5- ÇIKIŞ OLARAK CB2'NİN DENETİMİ CRA-5 (CRB-5) = YÜKSEK

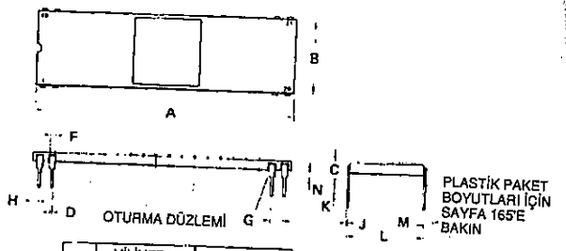
CRB-5	CRB-4	CRB-3	CB2 Sıfırlanır	CB2 Kurulur
1	0	0	MİB'nin, "B" Veri Kaydedicisine Yazma işlemini izleyen ilk E darbesinin pozitif geçişinde ALÇAK düzeye gider.	Kesme bayrak biti olan CRB-7'nin, CB1 sinyalinin aktif geçişiyle kurulması halinde YÜKSEK.
1	0	1	MİB'nin "B" Veri Kaydedicisine Yazma işleminden sonraki ilk E darbesinin pozitif geçişinde alçak düzeye gider.	İlgili parça seçilmediğinde ortaya çıkan E darbesini izleyen ilk E darbesinin pozitif kenarında YÜKSEK.
1	1	0	MİB'nin, B denetim kaydedicisine yazması nedeniyle CRB3'ün alçak düzeye gitmesi sonucunda ALÇAK.	CRB-3 alçak olduğu sürece ALÇAK. MİB'in, CRB-3'ün durumunun 1'e değişmesine neden olan, B Denetim kaydedicisine yazma yapması halinde YÜKSEK düzeye gidecektir.
1	1	1	CRB-3 yüksek olduğu sürece daima YÜKSEK. MİB'nin, CRB-3'ün sıfırlanması sonucunu doğuracak olan B denetim kaydedicisine yazma yapması halinde sıfırlanacaktır.	MİB'nin, B denetim kaydedicisine yazma yapması nedeniyle CRB3'ün yüksek düzeye gitmesi sonucu YÜKSEK.

...e CB2 Çevrebirimi Denetim Hatlarının De... (CRA-3, CRA-4, CRA-5, CRB-3, CRB-4 ve CRB-... denetim kaydedicisinin 3., 4. ve 5. bitleri, CA2 ve CB2 çevrebirimi denetim hatlarını denetlemek için kullanılır. Bu bitler, denetim hatlarının kesme girişi mi yoksa çıkış denetim sinyali mi olacağına karar verir. Eğer CRA-5 (CRB-5) biti "alçaksa", CA2 (CB2), CA1 (CB1) şekilde kesme giriş hattıdır. (Tablo 4). CRA-5 "yüksek" olduğu zaman CA2 (CB2); çevrebirimi tarımını denetlemek için kullanılan çıkış sinyali şür. Çıkış modunda CA2 ve CB2 biti karakteristiklere sahiptir. (Tablo 5 ve 6).

TABLO 6- ÇIKIŞ OLARAK CA2'NİN DENETİMİ
CRA.5= YÜKSEK

CRA-5	CRA-4	CRA-3	CA2 Sıfırlanır	CA2 Kurulur
1	0	0	MİB'nin "A" Verisini Okuma işleminden sonraki, E darbesinin negatif geçişinde ALÇAK.	Kesme bayrak biti olan CRA-7'nin, CA1 sinyalinin aktif geçişiyle kurulması halinde YÜKSEK.
1	0	1	MİB'nin "A" Verisini Okuma işleminden sonraki, E darbesinin negatif geçişinde ALÇAK.	Seçilmeme durumunda ortaya çıkan ilk "E" darbesinin pozitif kenarında YÜKSEK.
1	1	0	MİB'nin, A denetim kaydedicisine yazması nedeniyle CRA-3'ün alçak düzeye gitmesi sonucunda ALÇAK.	CRA-3 alçak olduğu sürece ALÇAK. MİB'in, CRA-3'ün durumunun 1'e değişmesine neden olan, A Denetim kaydedicisine yazma yapması halinde YÜKSEK düzeye gidecektir.
1	1	1	CRA-3 yüksek olduğu sürece daima YÜKSEK. MİB'nin, CRA-3'ün sıfırlanması sonucunu doğuracak olan "A" denetim kaydedicisine yazma yapması halinde sıfırlanacaktır.	MİB'nin, A denetim kaydedicisine yazma yapması nedeniyle CRA3'ün yüksek düzeye gitmesi sonucu YÜKSEK.





BOY	MİLİMETRE MİN.	MİLİMETRE MAKS.	İNÇ MİN.	İNÇ MAKS.
A	50.29	51.31	1.980	2.020
B	14.86	15.62	0.585	0.615
C	2.54	4.19	0.100	0.165
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	14.60	15.37	0.575	0.605
M	–	10°		10°
N	0.51	1.52	0.020	0.060

NOT:
BACAKLAR, (OTURMA DÜZLEMİNDE) 0,25 mm (0.010) ÇAPINDAKİ BİR ALANDA KONUMLANDIRILMIŞTIR, VERİLEN DEĞERLER MAKSİMUM KOŞULLAR ALTINDAKİ DEĞERLERDİR.

Intel 8251A programlanabilir iletişim arabirimi

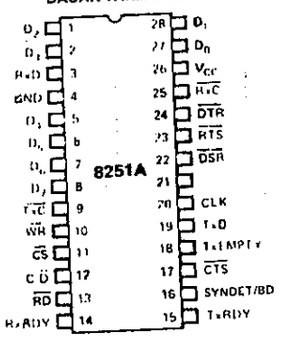
8251A
PROGRAMLANABİLİR İLETİŞİM ARABİRİMİ

- Senkron ve Asenkron Çalışma
- Senkron
  - 5-8 bit uzunlukta karakter
  - İç veya Dış Karakter Senkronizasyonu
  - Otomatik Senk. Eklemesi
- Asenkron
  - 5-8 bit uzunlukta karakter
  - Saat Hızı- Baud Hızının 1.16 veya 64 katı
  - Kesme karakteri üreteci
  - 1, 1 1/2 veya 2 Durdurma Biti
  - Hatalı Başlatma Bitinin Algılanması
  - Otomatik Kesme Algılama ve İşleme

- Baud hızı - DC'den 64 k Baud'a kadar
- Tam Dubleks, Çift Tamponlu, Verici ve Alıcı
- Hata Algılama - Eşlik, Ezme ve Çerçeve
- 8080/8085 MİB ile Tam Uyumluluk
- 28-bacaklı DIL paketi
- Bütün Giriş ve Çıkışlar TTL Uyumludur
- Tek bir 5 Volt'luk Kaynak
- Tek bir TTL Saati

Intel 8251 Genel-amaçlı Senkron/Asenkron Alıcı/Verici (USART)'nin geliştirilmiş endüstri standardı versiyonu olan 8251A, 8085 gibi Intel'in yeni yüksek performanslı mikroişlemci aileleriyle veri iletişimi için tasarlanmıştır. 8251A, çevresel birim olarak kullanılabilir ve günümüzde kullanımda olan (IBM Bi-sync tekniği de dahil) seri veri iletim tekniklerinden yararlanarak çalışabilecek şekilde MİB tarafından programlanabilir. USART, MİB'den gelen veri karakterlerini paralel biçimde alır ve bu karakterleri iletim için kesiksiz seri veri akışı biçimine çevirir. USART, aynı anda, seri veri akışını da alabilir ve MİB için paralel biçime çevirir. USART, iletim için yeni karakter kabul ettiğinde ya da MİB için bir karakter aldığında, MİB'le sinyal alış-verişinde bulunacaktır. MİB, herhangi bir anda USART'ın tüm durumunu okuyabilir. USART'ın durumları, veri iletim hataları ve SYNDET ve TxEMPTY gibi denetim sinyallerini içerir. Yonga N-kanallı silikon kapı teknoloji kullanılarak üretilmiştir.

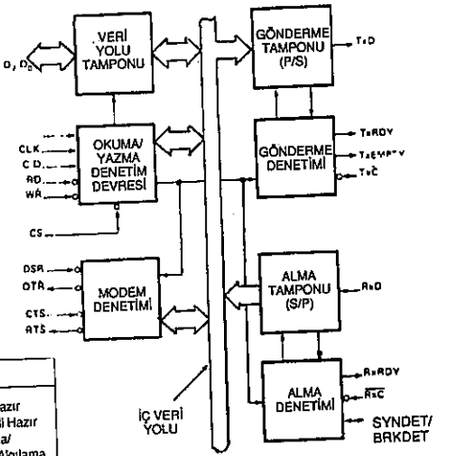
BACAK TANIMLARI



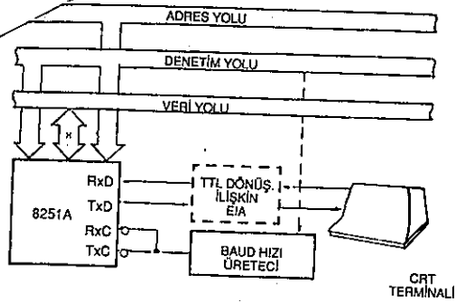
Bacak adı	İşlevi
D_0	Veri Yolu (8-bit)
D_1	Denetim veya Okunacak ya da Yazılacak Veri
RD	Veriyi Oku Komutu
WR	Veriyi Yaz veya Denetim Komutu
CS	Yonga Seçim
CLK	Saat Darbesi (TTL)
RESET	Sınırlama
TxD	Gönderici Saati
RxD	Gönderici Verisi
TxC	Alıcı Saati
RxD	Alıcı Verisi
RxRDY	Alıcı Hazır (MİB'e ilişkin karakter içeriyor)
TxRDY	Gönderici Hazır (MİB'den gelecek karakteri almaya hazır)

Bacak adı	İşlevi
DSR	Veri Grubu Hazır
DTR	Veri Terminali Hazır
SYNDET/BD	Sync Algılama/ Durdurma Algılama
RTS	Veri Gönderme İsteği
CTS	Veri Gönderme İzni
T x EMPTY	Gönderici Boş
Vcc	+ 5 Voltluk Kaynak
GND	Toplam

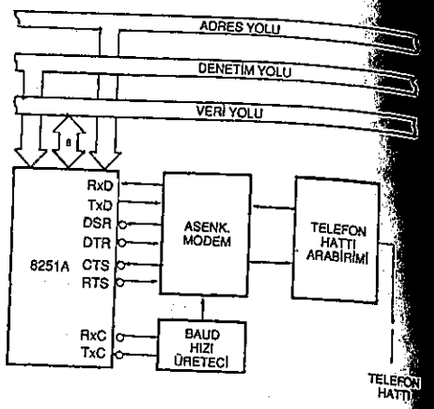
BLOK DİYAGRAM



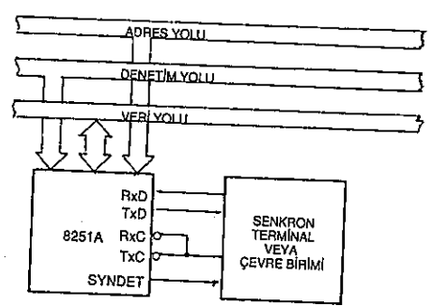
...ve CP
...(CRA-
...iki denet
...B2 çevre
...nılır. Bu
çıkış d



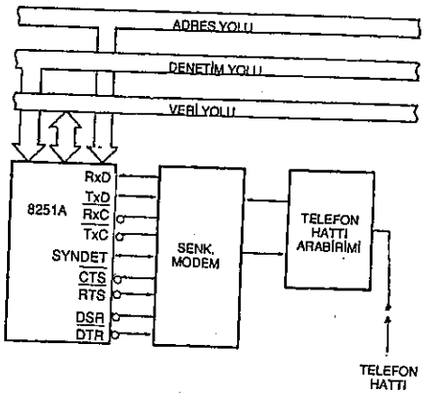
DC-9600 Baud hızında çalışan CRT Terminale Asenkron Seri Bağlantısı



Telefon Hatlarına Asenkron Bağlantı



Terminal veya Çevre Birimine Senkron Bağlantı



Telefon Hatlarına Senkron Bağlantı

Intel 2114 1.024 x 4-bitlik statik RAM

	2114-2	2114-3	2114	2114L3	2114L
Maks. Erişim Süresi (nsn)	200	300	450	300	370
Maks. Güç Harcanımı (mW)	710mw	710mw	710mw	370mw	370mw

- Yüksek Yoğunluklu 18-Bacaklı Paket
- Benzer Saykıl ve Erişim Süreleri
- Tek bir +5 Voltluk Kaynak
- Saat veya Zamanlama Strobu Gerektirmez
- Tümüyle Statik Bellek
- Bütün Giriş ve Çıkışlar: Doğrudan TTL Uyumlu
- Üç-Durumlu Çıkışlar Kullanarak Ortak Veri Giriş ve Çıkışı
- 3605 ve 3625 İkikutuplu PROM'larla Bacak-Çıkış Uyumluluğu

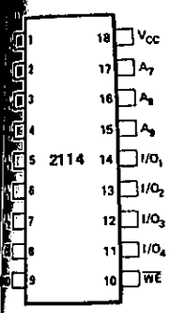
...2114, N-kanallı Silikon-Kapı MOS teknolojisi kullanılarak 4 bitlik 1024 sözcük olarak düzenlenmiş 4096 bitlik Statik Rastgele ...mli Bellektir. 2114, içindeki dizi ve kod çözme devrelerinde tümüyle DC kararlı (statik) devreler kullanır ve bu nedenle, ...smasında tazeleme veya saat sinyallerine gerek duymaz. Adres kurma süresi gerekmediğinden, veri erişimi özellikle ...illeşir. Veri bozulmadan okunur ve giriş verisi gibi aynı polariteye sahiptir. Ortak giriş/çıkış bacakları sağlanmıştır.

...2114; yüksek performans, düşük maliyet, büyük bit saklama ve basit arabağlantının, önemli tasarım amaçları olduğu belleklerin ...ygulamaları için tasarlanmıştır. 2114, mümkün olan en yüksek yoğunluk için 18 bacaklı paketin içine yerleştirilmiştir.

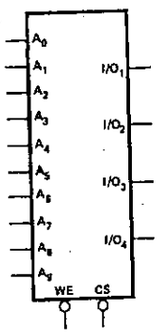
...2114, bütün yönleriyle doğrudan TTL uyumludur: Girişler, çıkışlar ve tek +5 V'luk kaynak. Çıkışlar, başka birimlerin çıkışlarına ...ağlı olduğu zaman, ayrı yonga seçim (CS) ucu, kendi paketinin kolay bir şekilde seçilmesine izin verir.

...2114, büyük maliyetli plastik paketlemenin kullanımında izin verilen kirliliğe karşı mükemmel koruma sağlayan Intel'in Nkanallı Silikon kapı teknolojisiyle üretilmiştir.

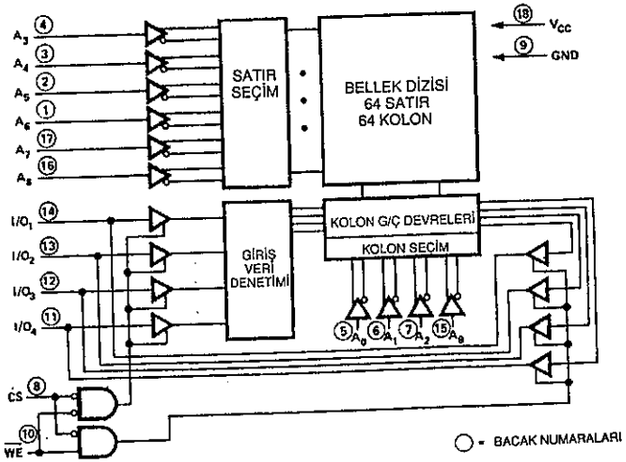
BACAK TANIMLARI



MANTIK SEMBOLÜ



BLOK DİYOGRAMI



BACAK ADLARI

A_0-A_9	ADRES GİRİŞLERİ	V_{CC} GÜCÜ (+5V)
$\overline{WE}$	YAZMA YETKİLEME	GND TOPRAK
$\overline{CS}$	YONGA SEÇİM	
I/O_1-I/O_4	VERİ GİRİŞ/ÇIKIŞ	

A.C. KARAKTERİSTİKLERİ $T_A = 0°C - 70°C$, $V_{CC} = 5V$ %5, aksi belirtilmedikçe

OKUMA SAYKILI [1]

SEMBOL	PARAMETRE	2114-2		2114-3, 2114I3		2114, 2114L	
		Min.	Maks.	Min.	Maks.	Min.	Maks.
t_{RC}	Okuma Saykıl Süresi	200		300		450	
t_W	Erişim Süresi		200	150	300		450
t_{CO}	Yonga Seçimden Çıkış Geçerli'ye		70	0	100		100
t_{CX}	Yonga Seçimden Çıkış Etkin'e	0		0		0	
t_{OTD}	Birim Seçilmediğinde Çıkışın 3. Duruma Gelmesine Kadar Süre	0	40	150	80	0	100
t_{OHA}	Adres Değişmesinden Çıkış Tutulu'ya	10		0		10	

YAZMA SAYKILI [2]

SEMBOL	PARAMETRE	2114-2		2114-3, 2114I3		2114, 2114L	
		Min.	Maks.	Min.	Maks.	Min.	Maks.
t_{WC}	Yazma Saykıl Süresi	200		300		450	
t_W	Yazma Süresi	100		150		200	
t_{WR}	Yazmayı Bırakma Süresi	20		0		0	
t_{OTW}	Yazmadan Çıkış 3-Duruma Geçmesine Kadar Geçen Süre	0	40	0	80	0	100
t_{DW}	Veriden, Yazma Süresi Örtüşmesi'ne	100		150		200	
t_{DH}	Yazma Süresinden Veri Tutma'ya	0		0		0	

NOT 1: ALÇAK CS ve YÜKSEK WE'nin üstüste binmesi süresinde bir Okuma işlemi gerçekleşir.
2: NOT 1: ALÇAK CS ve YÜKSEK WE'nin üstüste binmesi süresinde bir Yazma işlemi gerçekleşir.

A.C. TEST KOŞULLARI

Giriş Darbe Düzeyleri ..... 0.8 Volt - 2.4 Volt
Giriş Yükselme ve Düşme Süreleri ..... 10 nsn
Giriş ve Çıkış ..... 1.5 Volt
Çıkış Yükü ..... 1 TTL Kapı ve CL = 50 pF

Texas Instruments Dinamik RAM

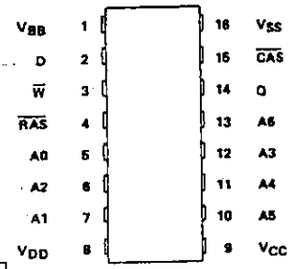
TMS 4116 - 30 JDH, NH

16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

ŞUBAT 1979

- 16.384 X 1 Düzeni
- Bütün Girişler, saat girişide dahil TTL uyumludur
- Tam TTL Uyumlu Mandallanmayan Üç - Durumlu Çıkış
- Erişim Süresi .... 300 nsn
- Saykıl Süresi .... 560 nsn
- "Erken Yazma" Özellikli Ortak G/Ç Yeteneği
- Düşük - Güç Harcanımı
  - Çalışmada .... 441 mW (maks.)
  - Beklemede .... 19 mW (maks.)
- 1-T Hücre Tasarımı, N - Kanallı Silikon - Kapı Teknolojisi
- 16-Bacaklı 300-Mil Paket Konfigürasyonu

16- BACAKLI SERAMİK VE PLASTİK DİL PAKET (ÜSTTEN GÖRÜNÜŞ)



BACAK ADLARI			
A0-A6	Adres Girişleri	$\overline{W}$	Yazıma Açma
$\overline{CAS}$	Kolon adresi strobu	V_{BB}	-5 V güç kaynağı
D	Veri Girişi	V_{CC}	+5 V güç kaynağı
Q	Veri Çıkışı	V_{DD}	+12 V güç kaynağı
$\overline{RAS}$	Satır adresi trobu	V_{SS}	0 V toprak

Tanımlama

TMS 4116 - 30 JH, NH; 16.384 bir-bitlik sözcükler halinde düzenlenmiş, monolitik, yüksek hızlı, dinamik 16.384 bitlik MOS rastgele - erişimli bellektir. Tek transistörlü saklama hücreleri ve N - kanallı silikon - kapı teknolojisi kullanır.

Bütün giriş ve çıkışlar, saat içeren 74 TTL serisi devrelerle uyumludur: Satır Adresi Strobu $\overline{RAS}$ (veya $\overline{R}$) ve Sütun Adres Strobu $\overline{CAS}$ (veya $\overline{C}$). Bütün adres hatları (AO'dan A6'ya) ve veri-girişi (D), sistem tasarımını basitleştirmek için yonga içine mandallanmıştır. Veri çıkışı (Q), daha büyük sistem esnekliğine imkan vermek için mandallanmamıştır.

Tipik güç harcanımı, çalışma durumunda 350 mW ve bekleme durumunda ise 6 mW'dan daha azdır (bekleme durumundayken V_{CC} gerekmez). Verinin bellekte tutulması için, belleğin içeriğini tazelemekte kullanılan güç de dahil olmak üzere yalnızca ortalama 10 mW güç gerekir.

TMS 4116 - 30 JDM, NM 16 bacaklı DIL kenarlı (JD) ve plastik (N) paketlerde sunulmuştur ve 0°C'den 55°C'ye kadar sıcaklık aralığında çalışabilmesi sağlanmıştır. Paket, 300-mil çapında takma-yuvası sıralarına ekleme yapılabilecek şekilde tasarlanmıştır.

Çalışma

adres (A0-A6)

1'den 16.384'e kadar saklama hücrelerinin kodunun çözülmesi için on dört adres biti gerekir. Yedi satır-adresi biti, AO'dan A6'ya kadar bacaklara bağlantılıdır ve satır adresi strobu ($\overline{RAS}$) tarafından yonga içinde mandallanmıştır. Ve yedi sütun adres biti de, AD'den A6'ya kadar yine bacaklara bağlantılıdır ve sütun adresi strobu ($\overline{CAS}$) tarafından yonga içinde mandallanmıştır. Tüm adresler, $\overline{RAS}$ ve $\overline{CAS}$'ın düşen kenarlarından önce veya bu esnada kararlı olmalıdır. RAS; Satır kod çözücüsü gibi algılama yükselteci gibi işlev gören yonga yetkileme sinyaline benzemektedir. $\overline{CAS}$, giriş ve çıkış tamponları ve sütun kod çözücüsü gibi işlev gören yonga yetkilemesi gibi kullanılır.

Okuma - yazma saykılı terimi kimi zaman *okuma değiştirmeli yazma saykılı* 'na alternatif olarak kullanılır.

TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

Yazıma Açma ($\overline{W}$)

Okuma veya yazma modu, yazıma açma ($\overline{W}$) girişinin aracılığıyla seçilir. Yazıma Açma ($\overline{W}$) girişindeki mantıksal YÜKSEK düzey, okuma modunu ve mantıksal ALÇAK düzey de, yazma modunu seçer. Yazıma açma ucu, yukarı çekme direnci kullanılmaksızın standart TTL devreler tarafından sürülebilir. Okuma modu seçildiğinde veri girişi yetkisizlenir. $\overline{W}$, $\overline{CAS}$'tan önce ALÇAK düzeye gittiğinde, ortak G/Ç işlemlerine izin veren saykılın tamamı boyunca veri çıkışı yüksek empedans durumda kalacaktır.

Veri-Girişi (D)

Veriler, yazma veya okuma değiştirmeli yazma saykılı süresinde yazılır. $\overline{CAS}$ veya $\overline{W}$'nin son düşen kenarı veriyi bellek içindeki veri mandalına stroblar. Bu mandal, yukarı çekme direnci kullanılmaksızın standart TTL devreler tarafından sürülebilir. Erken yazma saykılında $\overline{W}$, $\overline{CAS}$'tan önce ALÇAK düzeye çekilir ve veri, $\overline{CAS}$ sinyalini referans olarak kullanan kurma ve tutma süreleriyle birlikte bu sinyal tarafından stroblanır. Gecikmeli yazma veya okuma değiştirmeli yazma saykılında, $\overline{CAS}$ zaten ALÇAK düzeyde olacaktır, böylece veriler, $\overline{W}$ sinyalini referans alan kurma ve tutma süreleriyle birlikte bu sinyal tarafından mandalın içine stroblanacaktır.

Veri-Çıkışı (Q)

Üç durumlu çıkış tamponu, iki adet 74 TTL Serisi yükünü sürme yeteneğiyle, doğrudan TTL uyumluluk sağlar (yukarı çekme direnci gerekmez). Çıkan veri, giren veriyle aynı polaritededir. Çıkış, $\overline{CAS}$ ALÇAK düzeye getirilinceye kadar yüksek empedanslı (yüzer) durumdadır. Okuma saykılında çıkış, yetkileme süresi aralığından ta(C) - ki, $\overline{CAS}$'ın negatif geçişinden başlar, ta($\overline{R}$) 'nın istenen düzeye yükselmesine kadar devam eder - sonra etkin düzeye gider. $\overline{CAS}$, ALÇAK düzeyde iken, erişim süresi tamamlandıktan ve geçerli durumda kaldıktan sonra çıkış geçerli olur; $\overline{CAS}$'ın YÜKSEK düzeye gitmesi çıkışı yüksek empedanslı duruma geri döndürür. Erken yazma saykılında, çıkış her zaman yüksek empedans konumundadır. Gecikmeli yazma veya okuma değiştirmeli yazma saykılında, çıkış okuma saykılındaki sırayı takip edecektir.

Tazeleme

Tazeleme işlemi, verinin korunabilmesi için en azından iki milisaniyede bir yerine getirilmelidir. $\overline{CAS}$ uygulanmadığı sürece çıkış tamponu yüksek empedans durumunda kalacağından, çıkışı koruyacak olan, yalnızca $\overline{RAS}$ tazeleme sırasıdır. 128 satır adresinden (AO'dan A6'ya) herbirinin $\overline{RAS}$ ile stroblanması, her satırdaki bütün bitlerin tazelenmesini sağlayacaktır. $\overline{CAS}$, bu tazeleme sırası esnasında YÜKSEK (pasif) düzeyde kalacağından güç koruması sağlanır.

Güç-Uygulama

VBB, birime, ya diğer kaynaklardan önce veya aynı anda uygulanmalıdır ve en son olarak kesilmelidir. Bu işlemde yapılacak hata, ileri yöndeki iç öngerilim koşullarından dolayı mutlak maksimum anma değerlerinin aşılmasına neden olacaktır. Bu kaynak, aynı zamanda sistemin kullanımına da uygulanır. Sistem kullanımında VBB kaynağında ortaya çıkacak bir arıza, diğer kaynakları hemen kapatacaktır. Güç uygulandıktan sonra, birimin uygun biçimde çalışabilmesi için, sekiz bellek saykılı yerine getirilmelidir.

Serbest çalışma sıcaklık aralığındaki mutlak maksimum anma değerleri (aksi belirtilmedikçe)

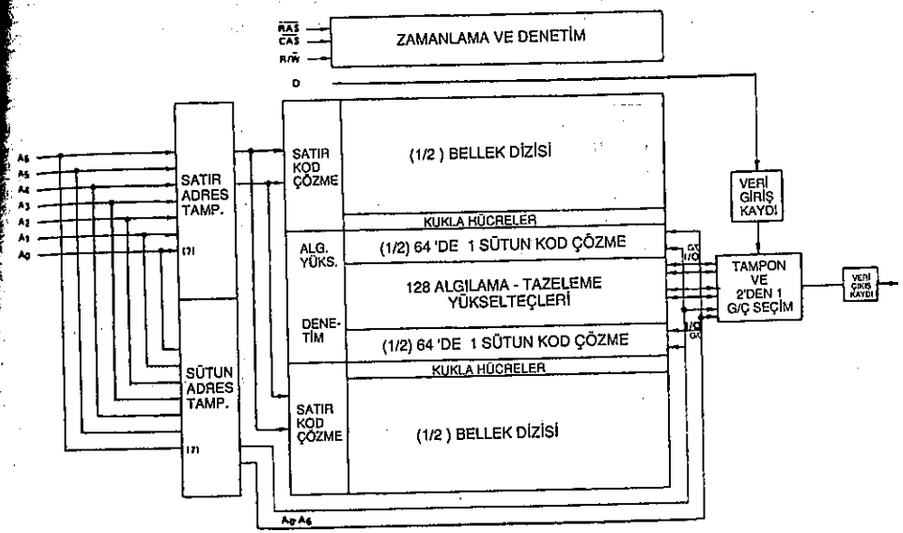
Herhangi bacaktaki gerilim (Bkz:Not 1) ..... -0.5 - 20 V
VSS'ye bağlı olarak VCC, VDD kaynaklarındaki gerilim ..... -1 - 15
Kısa devre çıkış akımı ..... 50 mA
Güç harcanımı ..... 1 W
Serbest çalışma sıcaklık aralığı ..... 0°C - 55°C
Saklama sıcaklık aralığı ..... -65°C - 150°C

NOT 1: Mutlak maksimum anma değerlerinde, gerilim değerleri aksi belirtilmedikçe, maksimum negatif besleme kaynağı, yani VBB gerilimlerine göre değerlendirilir. Bu veri tablosundaki geriye kalan değerlerin tümünün gerilim değeri Vss'ye göredir.

* "Mutlak Maksimum Anma Değerleri" başlığı altında sıralanan değerlerin üzerindeki kullanımlar, birimin kalıcı biçimde zarar görmesine neden olabilir. Bu, yalnızca kullanım aralığıdır ve birimin, bu veya "Önerilen Çalışma Koşulları" bölümünde belirtilen diğer değerlerin üzerinde çalıştırılması konusuna, bu tanımlama bölümde yer verilmemiştir. Daha uzun süreler boyunca mutlak maksimum anma değeri koşullarında çalışılması, birimin güvenilirliğini olumsuz yönde etkileyebilecektir.

TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

Fonksiyonel Blok Diyagram



Önerilen çalışma koşulları

PARAMETRE		MİN.	ANMA DEĞ.	MAK.	BİRİMİ
Besleme Gerilimi, VBB		-4.5	-5	-5.5	V
Besleme Gerilimi, VCC		4.5	5	5.5	V
Besleme Gerilimi, VDD		11.4	12	12.6	V
Besleme Gerilimi, VSS			0		V
Yüksek Düzey Giriş Gerilimi	$\overline{RAS}$, $\overline{CAS}$, $\overline{WRITE}$ dışında tüm Girişler	2.4		5.5	V
	$\overline{RAS}$, $\overline{CAS}$ $\overline{WRITE}$	2.7		5.5	V
Alçak-Düzey Giriş Gerilimi VIL (Bkz:Not 2)		-1	0	0.8	V
Tazeleme Süresi Aralığı, $t_{tazeleme}$				2	ms
Serbest Çalışma Sıcaklığı, TA		0		55	°C

NOT 2: Daha pozitif (veya daha negatif) sınır değerlerin maksimum değerler olarak belirlenmesine ilişkin cebirsel kurallar, bu veri tablosunda yalnızca mantıksal gerilim düzeyleri için kullanılmıştır.

TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

Önerilen çalışma koşullarının aralığındaki elektriksel karakteristikler (aksi belirtilmedikçe)

PARAMETRELER		TEST KOŞULLARI	MİN.	TİPİK†	MAKS.	BİRİM
V_{OH}	Yüksek-düzey çıkış gerilimi	I_{OH} = -5 mA	2.4			V
V_{OL}	Alçak-düzey çıkış gerilimi	I_{OL} = 4.2 mA			0.4	V
I_I	Giriş akımı (kaçak)	V_I = 0 V - 5.5 V V_{BB} = -5 V dışında diğer bütün çıkışlar			±10	μA
I_O	Çıkış akımı (kaçak)	V_O = 0 - 5.5 V, $\overline{CAS}$ yüksek			±10	μA
$I_{BB(1)}$	Yazma ve okuma saykılı	Minimum saykıl süresi		50	200	μA
$I_{CC(1)}$*	süresince ortalama				4**	mA
$I_{DD(1)}$	çalışma akımı			27	35	mA
$I_{BB(2)}$	Bekleme akımı	1 bellek saykılı sonrasında		10	100	μA
$I_{CC(2)}$		$\overline{RAS}$ ve $\overline{CAS}$			±10	μA
$I_{DD(2)}$		yüksek düzeydedir		0.5	1.5	mA
$I_{BB(3)}$	Ortalama tazeleme akımı	$\overline{CAS}$ yüksek düzeydeyken		50	200	μA
$I_{CC(3)}$		Minimum $\overline{RAS}$ saykılı			±10	μA
$I_{DD(3)}$		süresi		20	27	mA

*V CC yalnızca çıkış tamponuna uygulanır, böylece I CC çıkış yüküne bağlı olur.

** Çıkış yükü iki standart TTL yüküdür.

Önerilen kaynak gerilimi aralığı ve serbest çalışma sıcaklığı aralığında kapasitans, f= 1 MHz

PARAMETRE		MİN.	TİPİK†	MAKS.	BİRİM
$C_{I(A)}$	Giriş kapasitesi, adres girişleri		4	5	pF
$C_{I(D)}$	Giriş kapasitesi, veri girişi		4	5	pF
$C_{I(RC)}$	Giriş kapasitesi, strob girişleri		8	10	pF
$C_{I(W)}$	Giriş kapasitesi, yazıma açma girişi		8	10	pF
C_O	Çıkış kapasitesi		8	7	pF

†Bütün tipik değerler TA= 25°C ve norminal kaynak gerilimlerindedir.

Önerilen kaynak gerilimi aralığı ve serbest çalışma sıcaklığı aralığında anahtarlama karakteristikleri

PARAMETRE		TEST KOŞULLARI	ALTERNATİF SEMBOLÜ	MİN.	MAKS.	BİRİM
$t_a(C)$	Sütun adresi strobundan erişim süresi	C_L = 100 pF Yük = 2 adet 74 TTL Serisi kapı	t_{CAC}		190	nsn
$t_a(R)$	Satır adresi strobundan erişim süresi	t_{RLCL} = MAKS. C_L = 100 pF Yük = 2 adet 74 TTL Serisi kapı	t_{RAC}		300	nsn
t_{PXZ}	Çıkış yetkisizleme süresi	C_L = 100 pF Yük = 2 adet 74 TTL Serisi kapı	t_{OFF}	0	60	nsn

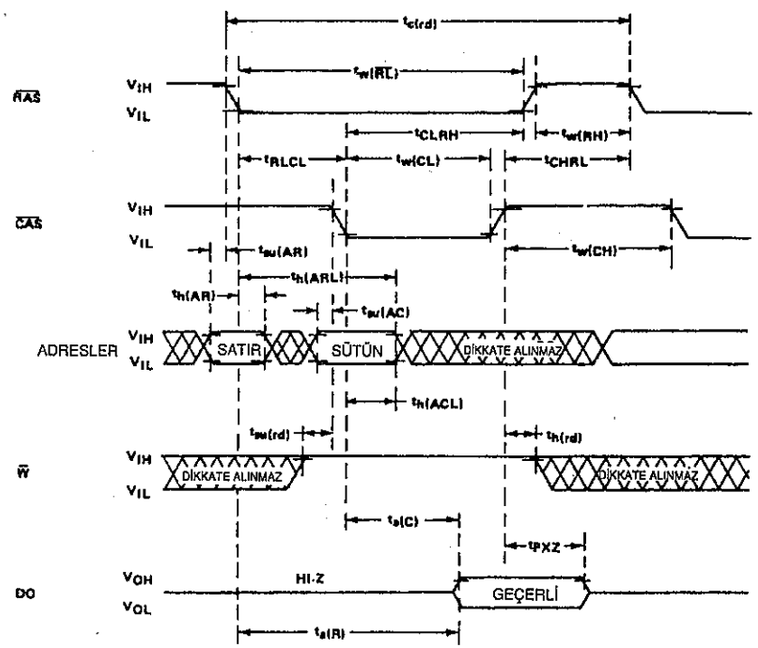
TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

Önerilen kaynak gerilimi ve serbest çalışma sıcaklığı aralığında zamanlama gerekleri

PARAMETRE	ALTERNATİF SEMBOLÜ	MİN.	MAKS.	BİRİM
Okuma saykılı süresi	t_{RC}	560		nsn
Yazma saykılı Süresi	t_{WC}	560		nsn
Okuma; değiştirmeli-yazma saykılı süresi	t_{RWC}	690		nsn
Darbe genişliği, sütun adresi strobu ALÇAK düzeyde	t_{CAS}	190	10.000	nsn
Darbe genişliği, satır adresi strobu YÜKSEK düzeyde (ön yük dolma süresi)	t_{RP}	250		nsn
Darbe genişliği, satır adresi strobu ALÇAK düzeyde.	t_{RAS}	300	10.000	nsn
Yazma darbesi genişliği	t_{WP}	100		nsn
$\overline{RAS}$ ve $\overline{CAS}$ için geçiş süreleri (yükselme ve düşme)	t_T	3	50	nsn
Sütun adresi kurma süresi	t_{ASC}	0		nsn
Satır adresi kurma süresi	t_{ASR}	0		nsn
Veri kurma süresi	t_{DS}	0		nsn
Okuma komutu kurma süresi	t_{RCS}	0		nsn
$\overline{CAS}$ YÜKSEK öncesi yazma komutu kurma süresi	t_{CWL}	125		nsn
$\overline{RAS}$ YÜKSEK öncesi yazma komutu kurma süresi	t_{RWL}	125		nsn
$\overline{CAS}$ ALÇAK sonrası sütun adresi tutma süresi		100		nsn
Satır adresi tutma süresi	t_{RAH}	60		nsn
$\overline{RAS}$ ALÇAK sonrası sütun adresi tutma süresi	t_{AR}	210		nsn
$\overline{RAS}$ ALÇAK sonrası $\overline{CAS}$ tutma süresi	t_{CSH}	300		nsn
$\overline{CAS}$ ALÇAK sonrası veri tutma süresi	t_{DH}	100		nsn
$\overline{RAS}$ ALÇAK sonrası veri tutma süresi	t_{DHR}	210		nsn
$\overline{W}$ ALÇAK sonrası veri tutma süresi	t_{DH}	100		nsn
Okuma komutu tutma süresi	t_{RCH}	0		nsn
$\overline{CAS}$ ALÇAK sonrası yazma komutu tutma süresi	t_{WCH}	100		nsn
$\overline{RAS}$ ALÇAK sonrası yazma komutu tutma süresi	t_{WCR}	210		nsn
Sütun adres strobu YÜKSEK'ten, satır adres strobu ALÇAK'a gecikme süresi	t_{CRP}	0		nsn
Sütun adres strobu ALÇAK'tan, satır adres strobu YÜKSEK'e gecikme süresi	t_{RSH}	190		nsn
Sütun adres strobu ALÇAK'tan, $\overline{W}$ ALÇAK'a gecikme süresi (yalnızca okuma, değiştirmeli yazma saykılı)	t_{CWD}	200		nsn
Tazeleme süre aralığı	t_{REF}		2	msn
Satır adres strobu ALÇAK'tan, sütun adres strobu ALÇAK'a gecikme süresi (yalnızca, erişim süresini sağlayan maksimum değer belirtilmiştir)	t_{RCD}	60	110	nsn
Satır adres strobu ALÇAK'tan, $\overline{W}$ ALÇAK'a gecikme süresi (yalnızca okuma-değiştirmeli yazma saykılı)	t_{RWD}	300		nsn
$\overline{W}$ ALÇAK'tan, sütun adres strobu ALÇAK'a gecikme süresi (erken yazma saykılı)	t_{WCS}	0		nsn

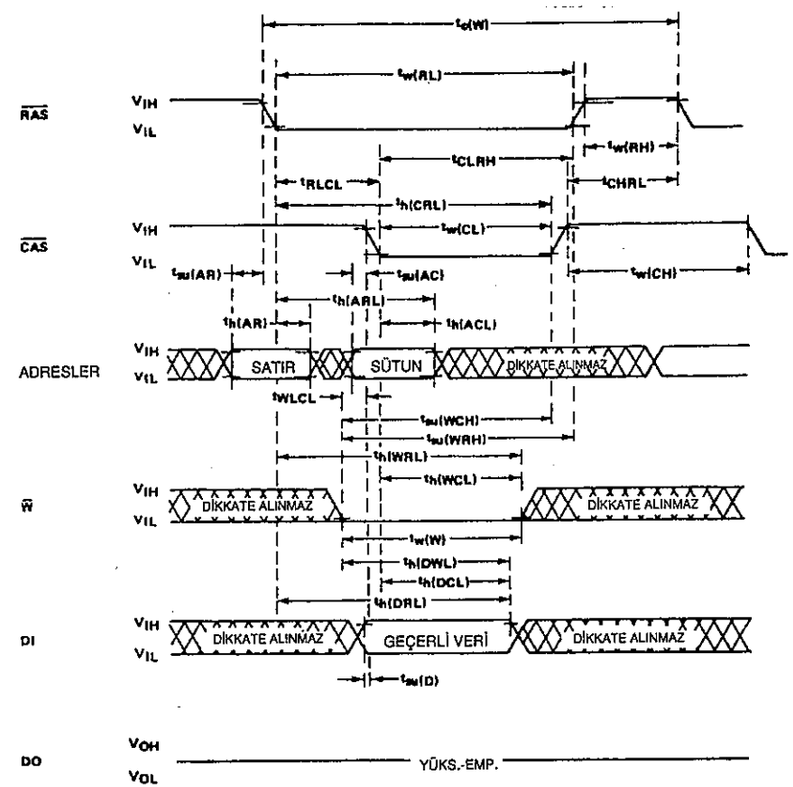
TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

okuma saykılı zamanlaması



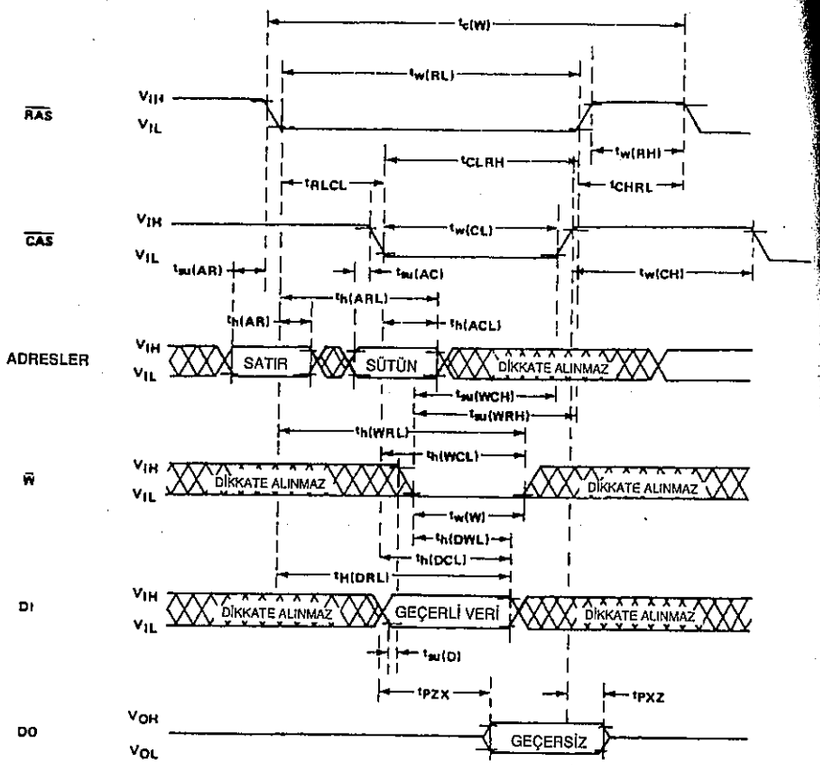
TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

Erken yazma saykılı zamanlaması



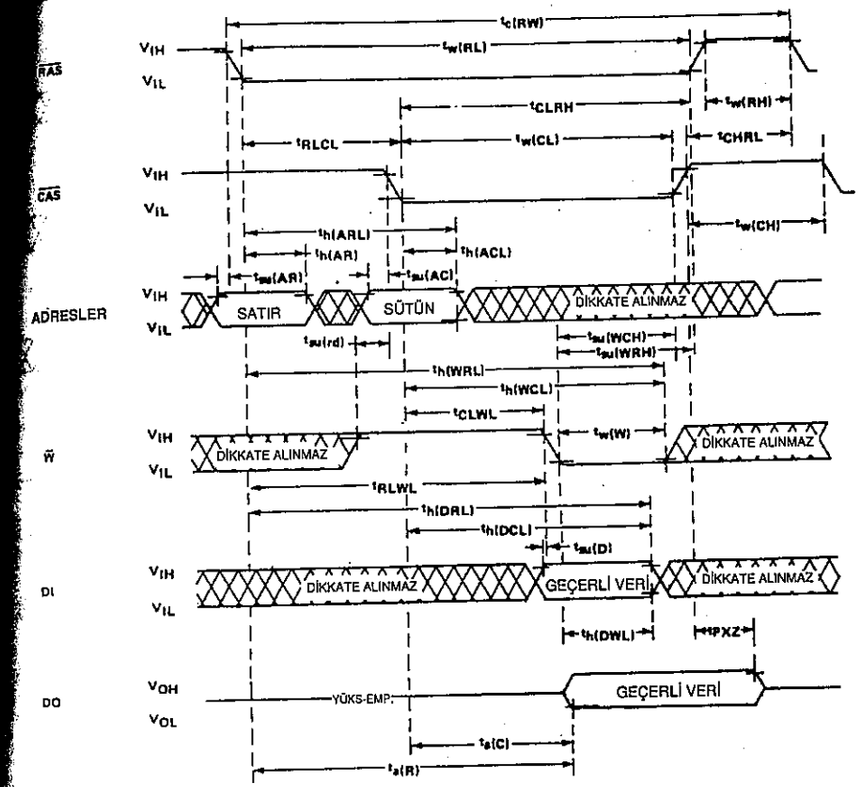
TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

yazma saykılı zamanlaması



TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

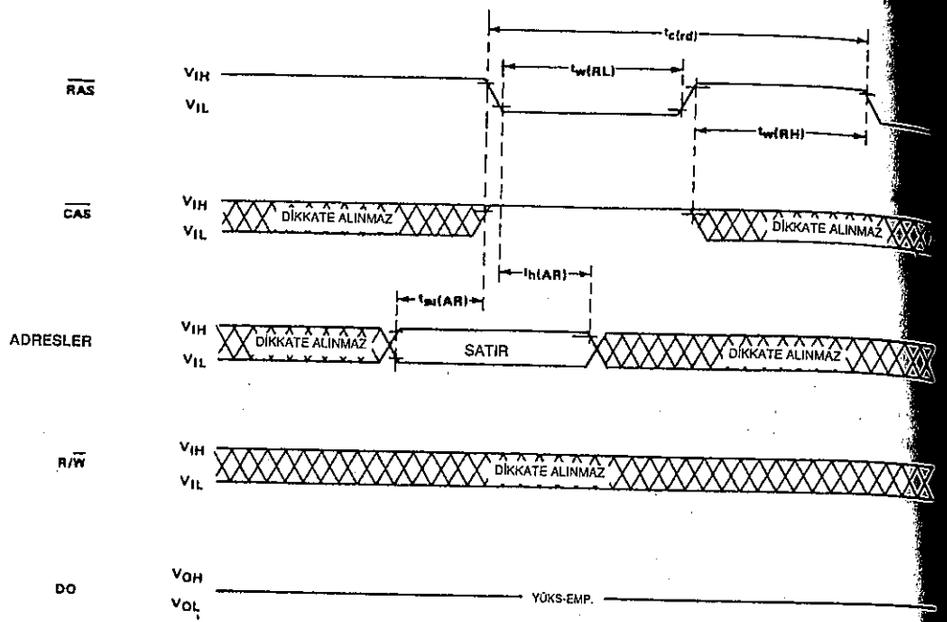
Yazma/okuma-değiştirmeli yazma saykılı zamanlaması



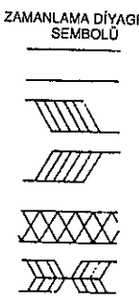
RAS - Yalnızca tazeleme zamanlaması

TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK

$\overline{RAS}$-yalnızca tazeleme zamanlaması

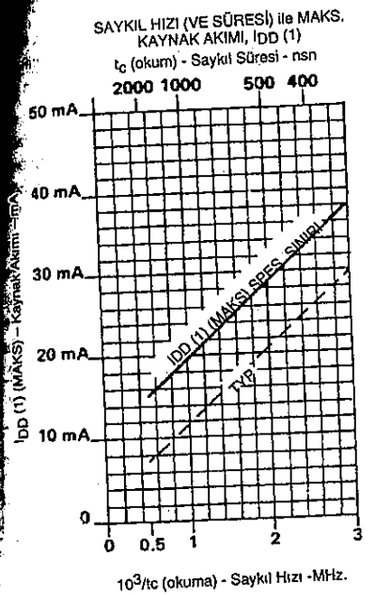


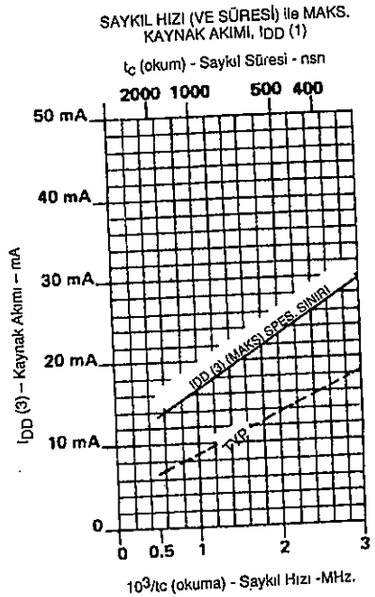
Zamanlama şeması kuralları



ZAMANLAMA DİYAGRAM SEMBOLÜ	ANLAMI: GİRİŞ ZORLAYAN FONKSİYONLAR	ANLAMI: ÇIKIŞ YANIT FONKSİYONLAR
	Yüksek veya Alçak düzeyde kararlı olmalı	Yüksek veya Alçak düzeyde kararlı olacak
	Yüksekten Alçak'a değişime izinli	Ayrılmış aralıkta bazen yüksekten-alçaka değişecektir.
	Alçaktan-Yüksek'e değişime izinli	Ayrılmış aralıkta bazen alçaktan - yüksek'e değişecektir
	Dikkate Alınmaz	Bilinmiyen veya değişen durum
	(Uygulanmaz)	Orta çizgi, yüksek- empedanslı KAPALI-durumu gösterir.

TMS 4116 - 30 JDH, NH
16.384-BİTLİK DİNAMİK RASTGELE-ERİŞİLEBİLİR BELLEK





Ek N Intel 2716 16K UVEPROM

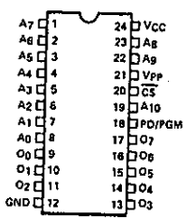
- Tek +5 V Güç Kaynağı
- Basit Programlama Gerekleri
  - Yalnızca 50 msn'lik Darbeyle
  - Tek Gözü Programlama Programları
- Düşük Güç Harcaması
  - Çalışmada maks. 525 mW
  - Beklemede maks. 132 mW
- Intel 2316E ROM'a Bacak Uyumlu
- Hızlı Erişim Süresi: Maks. 450 nsn
- Okuma ve Programlamada
  - Giriş ve Çıkışların
  - TTL Uyumluluğu

Intel 2716, mor ötesi ışık ile silinebilen ve elektriksel olarak programlanabilen 16.384 bitlik sadece-okunabilir (EPROM)'[illegible] Tek bir 5 Volt'luk güç kaynağı ile çalışan 2716, statik güç düşürme moduna ve hızlı tek adres gözünün programl[illegible] olanağına sahiptir. Bu özellik EPROM'un daha hızlı, daha kolay ve çok ekonomik bir biçimde düzenlenebilmesini sağl[illegible] ürün miktarlarını 2716 kullanıcısı hızlı bir şekilde Intel'in yeni, bire-bir bacak uyumlu 16K ROM'u olan 2316E'ye dönüştü[illegible]

450 nsn'lik 2716 tek 5 volt'luk kaynaktan beslendiğinden, Intel 8085 ve 8048 gibi yeni, yüksek performanslı +5 volt[illegible] işlemcilerle en uygun biçimde kullanılır. 2716, aynı zamanda, erişim süresini arttırmaksızın güç harcanımını azaltan [illegible] düşürme moduna sahip ilk EPROM'dur. En fazla etkin güç harcanımı 525 mW iken, bekleme durumundaki naksim[illegible] harcanımı yalnızca 132 mW'tır, %75 tasarruf sağlanır.

2716, EPROM'ların programlanması için şimdiye kadar bulunmuş en basit ve en hızlı yönteme sahiptir; tek darbe TT[illegible] programlama. Bütün programlama denetimleri TTL sinyallerle yönetildiğinden yüksek gerilim darbesi gerektirmez. Bu[illegible] çalışılan sahada, sistem içindeki elektronik kart üzerinde programlamak mümkün olur. Herhangi bir zamanda, herhang[illegible] tek tek, sırayla veya rastgele programlamak, 2716'nın tek adres gözü programlaması ile gerçekleştirilir. 16.384 bit için[illegible] programlama süresi yalnızca 100 saniyedir.

BACAK TANIMLARI



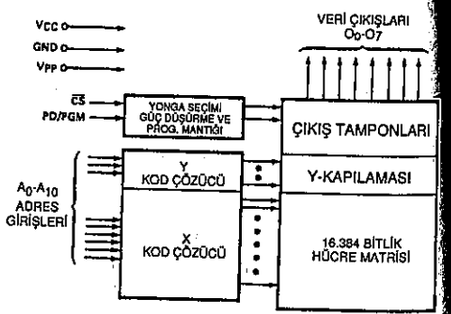
BACAK ADLARI

A_0–A_{10}	ADRESLER
PD/PGM	GÜÇ DÜŞÜRME/PROGRAM
$\overline{CS}$	YONGA SEÇİMİ
O_0–O_7	ÇIKIŞLAR

MOD SEÇİMİ

BACAK MOD	PD/PGM (18)	CS (20)	Vpp (21)	Vcc (24)	ÇIKIŞLAR (9-11,13-17)
Okuma	V_{IL}	V_{IL}	+5	+5	DÇIK
Seçilmeme	Dikkate Alınmaz	V_{IH}	+5	+5	Yüksek E[illegible]
Güç Düşürme	V_{IH}	Dikkate Alınmaz	+5	+5	Yüksek E[illegible]
Program	V_{IH}'ye darbelenen V_{IL}	V_{IH}	+25	+5	DGİR
Program Doğrulama	V_{IL}	V_{IL}	+25	+5	DÇIK
Program Yasaklama	V_{IL}	V_{IH}	+25	+5	Yüksek E[illegible]

BLOK DİYAGRAMI



[illegible]KARAKTERİSTİKLERİ

[illegible]me karakteristikleri, dalga boyu yaklaşık olarak [illegible]trom (Å)'dan daha kısa ışığa maruz bırakıldığında, [illegible]başlayacağını gösteren değerlerdir. Güneş ışığı ve [illegible]resan lambalar 3000 - 4000 Å değerlerinde dalga- [illegible] sahiptirler. Verilerden, tipik bir 2716'nın, odalarda [illegible]floresan ışığına maruz bırakıldığında yaklaşık 3 [illegible]bileceğini, doğrudan güneş ışığı altında ise bu [illegible] yaklaşık 1 hafta olacağını öğrenmekteyiz. Eğer [illegible]zun bir süre boyunca bu çeşit ışık kaynakları altında [illegible] birimin içerdiği verilerin, arzu edilmeyen bir biçim- [illegible]mesinin önüne geçebilmek için, elemanın üzerine [illegible]ndan kullanılan ışık geçirmez etiketler yapıştırılma-

[illegible]için önerilen silme yöntemleri; (bkz: sayfa-3-55), [illegible]2537 Angstroms (A) dalga boyuna sahip kısa-dalga [illegible]ışıka tutulmasıdır. Silme için toplam miktar: (yani, [illegible]yoğunluğu X kalış süresi), minimum 15 W - sani- [illegible]olmalıdır. Bu miktarda silme süresi, 12000 $\mu W/cm^2$ [illegible]değerine sahip morötesi lamba kullanarak yaklaşık 15 [illegible]dakika arasındadır. 2716, silme sırasında, lambanın [illegible]k 2.5 cm. uzağına konmalıdır. Bazı lambaların önün- [illegible]ütre olduğundan, silmeye başlamadan önce bunların [illegible]ması gerekir.

[illegible]İMİN ÇALIŞMASI

[illegible]'nin altı çalışma şekli, Tablo 1'de listelenmiştir. Altı [illegible]da kullanılan tüm girişlerin TTL düzeyli olduğuna dikkat [illegible] Gereken güç kaynakları +5V V_{CC} ve bir Vpp'dir. Vpp [illegible]aynağı, üç programlama modu sırasında 25 V, diğer üç [illegible]da ise 5 V üretmelidir.

Tablo 1. MOD SEÇİMİ

[illegible] BACAK [illegible]	PD/PGM (18)	CS (20)	Vpp (21)	Vcc (24)	ÇIKIŞLAR (9-11, 13-17)
[illegible]	V_{IL}	V_{IL}	+5	+5	DÇIK
[illegible]me	Dikkate Alınmaz	V_{IH}	+5	+5	Yüksek Emp.
[illegible] Düşürme	V_{IH}	Dikkate Alınmaz	+5	+5	Yüksek Emp.
[illegible]am	V_{IH}'ye darbelenen V_{IL}	V_{IH}	+25	+5	DGİR
[illegible]am Doğrulama	V_{IL}	V_{IL}	+25	+5	DÇIK
[illegible]am Yasaklama	V_{IL}	V_{IH}	+25	+5	Yüksek Emp.

[illegible]OKUMA MODU

[illegible]kuma modunda veri, çıkışlarda mevcuttur. Veriler, $\overline{CS}$; [illegible]ALÇAK düzeyde iken adreslerin kararlı hale gelmesinden [illegible]baren 450 nsn (t_{ACC}) içinde veya adresler kararlı halde iken, [illegible]ya da $\overline{CS}$, ALÇAK düzeye gelmesinden 120 nsn (tco) sonra [illegible]llanılabilir hale gelir.

SEÇİLMEME MODU

İki veya daha fazla 2716'nın çıkışları, aynı çıkışlar birbirinin üstüne gelecek şekilde aynı veri yoluna bağlanabilirler. Bu konfigürasyonda, 2716'ların veri yolunu kullanımı konusunda ortaya çıkabilecek bir karmaşaya engel olabilmek için, yalnızca bir 2716'nın çıkışları seçilmelidir ($\overline{CS}$ ALÇAK). $\overline{CS}$ girişlerine YÜKSEK TTL düzey uygulanarak, diğer 2716'ların çıkışlarının seçilmemiş olması sağlanmalıdır.

GÜÇ DÜŞÜRME MODU

2716, çalışmada harcanan gücü %75 oranında (525 mW'tan 132 mW'a) azaltan güç düşürme moduna sahiptir. Güç düşürme, PD/PGM girişine, yüksek TTL düzeyli sinyal uygulayarak gerçekleştirilir. Güç düşürme modunda çıkışlar, $\overline{CS}$ girişinden bağımsız olarak yüksek empedans durumunda bulunurlar.

PROGRAMLAMA

Başlangıçta ve her bir silmeden sonra 2716'nın içindeki bütün bitler "1" durumundadır. Veriler, seçilen "0" ları programlayarak istenen bit gözlerine yerleştirilirler. Yalnızca "0" lar programlanmasına rağmen, veri sözcüğünün içinde "1" ve "0" ların her ikisi de bulunabilir. "0"ları "1" lerle değiştirmenin tek yolu morötesi ışıkla silmektir.

Vpp güç kaynağı 25 V'ta ve $\overline{CS}$ VIH (Yüksek düzey)'de olduğu zaman 2716, programlama modundadır. Programlanacak veriler, 8-bitlik paralel şekilde veri çıkış bacaklarına uygulanır. Adres ve veri girişleri için gereken düzeyler, TTL düzeyidir.

Adres ve veriler kararlı olduğu zaman, PD/PGM girişine, 50 milisaniye süreli, YÜKSEK'te-ETKİN, TTL programlama darbesi uygulanır. Programlama darbesi, programlanacak her adres gözüne uygulanmalıdır. Herhangi bir anda, herhangi bir göz; tek tek, ardışık olarak ya da rastgele programlanabilir. Program darbesinin maksimum genişliği 55 milisaniyedir. 2716, PD/PGM girişine DC sinyal uygulanarak programlanmamalıdır.

Paralel biçimdeki farklı 2716'ların aynı veri ile programlanması, programlama gereklerinin basitliğinden dolayı kolaylıkla gerçekleştirilebilir. Aynı veri ile programlanacakları zaman, paralel 2716'ların girişleri birbirine bağlanabilir. PD/PGM girişine uygulanan YÜKSEK düzeyli TTL darbesi, paralel 2716'ları programlar.

PROGRAM YASAKLAMA

Paralel biçimdeki birden çok 2716'ın farklı verilerle programlanması da kolaylıkla gerçekleştirilebilir. PD/PGM hariç, tüm paralel 2716 girişleri, ($\overline{CS}$'de dahil) ortak olabilir. 2716'nın PD/PGM girişindeki ALÇAK düzey, diğer 2716'ları programlarken, söz konusu birimin programlanmasını yasaklar.

PROGRAM DOĞRULAMA

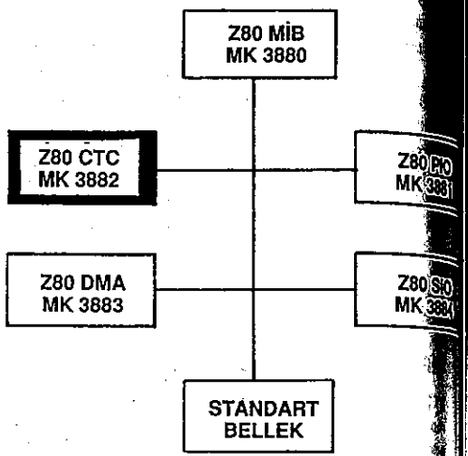
Doğrulama, programlanan bitlerin doğru programlanıp programlanmadığına karar vermek için bu bitlerin her biri üzerinde tek tek gerçekleştirilmelidir. Doğrulama, V_{PP} 25 V'ta iken yerine getirilebilir. V_{PP}; Programlama ve program doğrulama dışındaki durumlarda 5 V olmalıdır

Ek O Mostek Z80 sayıcı zamanlayıcı devresi

Z80 SAYICI ZAMANLAYICI DEVRESİ MK 3882

ÖZELLİKLERİ

- Her kanal, sayıcı ya da zamanlayıcı modunda çalışmak üzere seçilebilir.
- Sayıcı veya zamanlayıcı durumlarına göre programlanabilen kesmeler
- Geriye sayıcının, sıfır olana kadar saydığı sayılar okunabilir
- Her zamanlayıcı kanalı için seçilebilen 16 ya da 256 saatlik ön ölçekleyici
- Seçilebilen pozitif veya negatif tetikleme zamanlayıcının çalışmasını başlatır.
- Üç kanal da, Darlington tranzistörlerini sürebilen sıfır sayım/süre-aşımı çıkışlarına sahiptir.
- Dış mantık devreleri kullanılmaksızın otomatik kesme vektörlemesini sağlayan papatya zinciri düzeninde öncelikli kesme mantığı devreleri içerir.
- Bütün giriş ve çıkışlar tamamen TTL uyumludur.



AÇIKLAMA

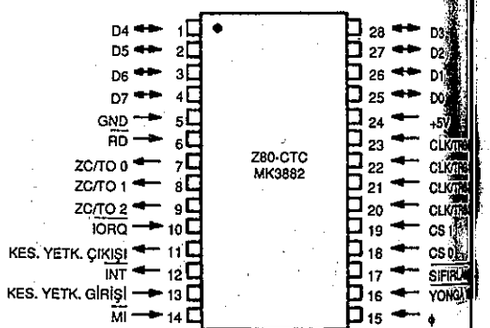
Mostek Z80 ürün ailesi, mikrobilgisayar elemanlarının, geliştirme sistemlerinin ve destek yazılımlarında kullanılan komple bir takımdır. Z80 mikrobilgisayar eleman takımı, en az sayıda düşük maliyetli standart bellek elemanlarıyla diğer mantık devreleri gerektirmeyen yüksek performanslı mikrobilgisayar sistemi kurabilmek için gerekli devrelerin tümünü içerir.

Z80 Sayıcı Zamanlayıcı Devresi (CTC), Z80 MİB için sayma ve zamanlama işlevlerini sağlayan, programlanabilir dört kanallı bir elemandır. Z80 MİB, Z80 - CTC'nin dört bağımsız kanalını, çeşitli modlar ve istenen koşullar altında çalıştırmak için düzenler.

YAPI

- N - Kanallı Silikon Kapı Boşaltılmış Yük Teknolojisi
- 28 bacaklı DIP
- Tek 5 V'luk kaynak
- Tek fazlı 5 V'luk saat
- Bağımsız programlanabilen dört adet 8-bitlik sayıcı/16-bitlik zamanlayıcı kanalları

BACAK BAĞLANTILARI



Ek P Zilog Z6132 yarı-statik RAM

Tanımlama

Zilog Z6132, 8-bitlik 4096 sözcük olarak düzenlenmiş, 5V'la çalışan akıllı MOS dinamik RAM'dır. Tek transistörlü dinamik saklama hücreleri kullanmasına rağmen, Z6132'in gerçekte statik RAM gibi davranır, çünkü kendi tazeleme işlevlerini kendi yerine getirir ve denetler. Bu, dıştan bir tazeleme devresi bağlanma ihtiyacını ortadan kaldırır ve statik RAM'ın olanaklanyla, normalde yüksek yoğunluklu ve düşük güç harcanımına sahip olan dinamik RAM'ın özelliklerini birleştirir.

Z6132, özellikle, bayt-genişlikli düzenlemelerin olduğu mikroişlemci ve mikrobilgisayar uygulamaları için uygundur ve kendi kendini tazeleme devresi ile tek besleme gerilimi, çevre elemanlarının sayısını azaltır ve tasarımı basitleştirir.

Z6132, küçük bellek hücresi, hızlı erişim ve düşük güç harcanımı sağlayan, statik ve dinamik devreler için yüksek performanslı, yük-oluşturmalı çift-kutuplu n-kanal silikon kapı MOS teknolojisi kullanır. Z6132, uygulamalarında maksimum esneklik sağlamak için ayrı adres ve ayrı veri G/Ç bacaklarına sahiptir.

Devre, endüstri standartlı 28-bacaklı DIP kılıfında paketlenmiştir ve öngörülen JEDEC standartı ile bacak uyumludur.

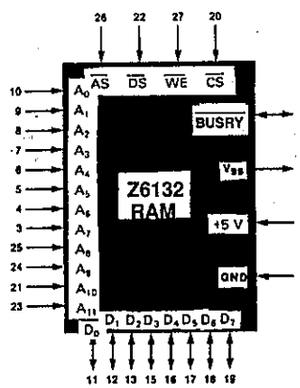
Z6132, Zilog mikroişlemcilerinin yeni kuşağı Z8 ve Z8000 tarafından kullanılan Z tipi yol özelliklerine uyar.

Özellikler

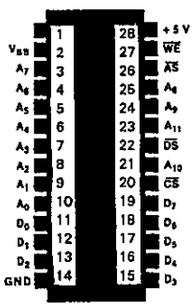
- Bayt - genişlikli düzenleme: 8-bitlik 4096 sözcük
- Belirtilen gerilim ve sıcaklık aralığında sağlanan erişim ve saykıl süreleri

Parça Numarası	Erişim Süresi	Saykıl süresi
Z6132-3	200 nsn	350 nsn
Z6132-4	250 nsn	375 nsn
Z6132-5	300 nsn	425 nsn

- Düşük Güç harcanımı: Çalışır durumda 200 mW, beklemede 125 mW
- JEDEC tarafından öngörülen bacak-çıkışlarına uyumlu, endüstri standartı 28 bacaklı-DIP kılıfı
- +5 V besleme geriliminde ± %10 tolerans
- Düşük ve hızlı saykıl modlu otomatik kendi kendini tazeleme devresi
- Bütün girişler ve çıkışlar TTL uyumludur.
- Yonga-içi altkatman öngerilim üretici
- Z8 ve Z8000 için hazır arabirimler



Şekil 1. Mantık Sembolü



Şekil 2. Bacak Tanımları

A_0 - A_{11}	Adres girişleri
D_0 - D_7	Veriş Giriş/Çıkışları (üç-durumlu)
AC	Adres Saat Girişi (yükselen kenar)
$\overline{DS}$	Veri Strob girişi (ALÇAK'ta-etkin)
$\overline{WE}$	Yazma Yetkileme Girişi (ALÇAK'ta-etkin)
$\overline{CS}$	Yonga Seçim Girişi (ALÇAK'ta-etkin)
$\overline{BUSY}$	Meşgul çıkışı (ALÇAK'ta-etkin, açık kanallı ve tazeleme modu denetim girişi)
V_{BB}	Negatif Taban Öngerilim çıkışı
V_{CC}	+ 5 V kaynak bağlantısı
V_{SS}	0 V = Toprak bağlantısı

Bacak Adları

İşlevsel Tanımlama

Z6132, 4Kx8'lık yan-statik RAM, 128 algılama yükseltecinin her iki kenarında 64 satır bulunun, iki gruba sahip, iki farklı blok olarak düzenlenmiştir. Her iki blok, ayrı ve birbirinden bağımsız satır adres tamponlarına ve kod çözücülere sahiptir, fakat sütun kod çözücülerini ve 8-bit genişlikte iç veri yolunu ortaklaşa kullanır. Satır adres kod çözücülerinin iki grubu da, ya A_1 - A_7 adres girişleri, ya da 7-bitlik iç tazeleme sayıcısı olarak adreslenirler. En küçük değerlikli adres girişi (A_0), dışarıya erişebilmek için iki bloktan birini seçer. Seçilen blok, okuma veya yazma işlemi yürütürken, diğer bellek bloğu, bir satır tazelemek için tazeleme sayıcısı adresini kullanır. Kendi kendini tazeleme mekanizmasının ayrıntıları daha sonra açıklanacaktır.

Bellek saykılı; Yonga Seçim ($\overline{CS}$), A_0 ve Yazma Yetkileme ($\overline{WE}$) sinyallerini saatleyen Adres Saat'inin (AC) yükselen kenarıyla başlar. Eğer yonga seçilmemişse ($\overline{CS}$ = YÜKSEK ise), diğer bütün girişler, saykılın geriye kalan (yani AC'nin bir sonraki yükselen kenarına kadar olan) süresi boyunca dikkate alınmaz. Her iki bellek bloku da, 7 bitlik tazeleme sayacı tarafından, kendi kendilerini tazelerler. Eğer yonga seçilmişse ($\overline{CS}$ = ALÇAK ise), 12 adres biti ve Yazma Yetkileme biti, kendi kaydedicilerine saatlenirler. A_0, hangi bloğun A_1-A_{11} bitleri tarafından adresleneceğine karar verir ve diğer blok ise, 7-bitlik tazeleme sayıcısı tarafından tazelenir.

Yonga Seçim ve Adres girişleri yalnızca, AC'nin yükselen kenarından sonra kısa tutma süresi boyunca geçerli olmalıdır. Bu, veri G/Ç'li farklı denetim girişi Veri strobu ($\overline{DS}$) tarafından denetlendiği için adres/veri çoğullamasına izin verir.

Okuma Saykılı

Okuma saykılı, Yonga Seçim ($\overline{CS}$) alçak ve Yazma Yetkileme ($\overline{WE}$) yüksek düzeydeyken adres saatinin yükselen kenarıyla başlar. Veri strobu ($\overline{DS}$), girişindeki alçak düzey, $\overline{DS}$'nin düşen kenarında olduğu gibi, AC'nin yükselen kenarından belirli bir gecikme süresi sonra veri çıkışlarını etkin hale getirir. Okuma işlemi esnasında, $\overline{DS}$, statik çıkış yetkileme sinyalidir.

Yazma Saykılı

Yazma saykılı, Yonga Seçim ($\overline{CS}$) ve Yazma Yetkileme ($\overline{WE}$) sinyalleri ALÇAK düzeyde olduğunda, Adres Saatinin (AC) yükselen kenarıyla başlar.

$\overline{WE}$ girişi, ($\overline{DS}$) veri strobunun başlangıçında (düşen kenarında) tekrar kontrol edilir. Eğer $\overline{WE}$ hala alçak düzeydeyse, DS'nin bu düşen kenarı D_0 - D_7 girişindeki veriyi tetikleyerek, adreslenebilir bellek gözüne gönderir. Veri, $\overline{DS}$'nin düşen kenarından sonrasında ortay çıkan kısa süre boyunca geçerli olmalıdır.

Yazma Yasaklama Saykılı

Yazma saykılı başlatıldıktan sonra, yürürlükteki yazma işlemi, $\overline{DS}$'nin düşen kenarından önce $\overline{WE}$'yi tekrar yüksek düzeye çekerek iptal edilebilir. Bu yazma yasaklama saykılı, AC süresinin hemen başında yazma saykılının başlamasına izin verir, fakat $\overline{DS}$ süresinde yazma işleminden sonra bile hala yasaklama seçeneğine izin veren özel bir olanaktır.

Not: *Yazma saykılı ne zaman başlarsa başlasın, bu işlem, veri strob girişindeki sinyalin yüksek'ten alçak düzeye geçişine eşlik etmelidir.*

Maksimum Saykıl Süresi

Maksimum okuma veya yazma saykıl süresi gerekleri (15.000 ve 800 nsn) herhangi tek saykılda uygulanmaz. Bunlar, 2 msn'lik peryotlarla bütün tazeleme işlemlerini sağlamak üzere belirlenmiştir.

Bk Q

TTL

MSI

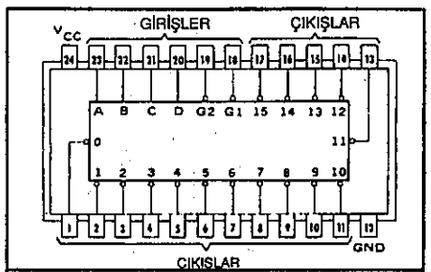
Texas Instruments kod çözücü/tekilleyicileri

SN54154, SN54L154, SN74154, SN74L154
4 HATTAN 16, HATTA KOD ÇÖZÜCÜ

BÜLTEN NO. DL - S 7211805, ARALIK 1973

- '154, Yüksek Performanslı Bellek Kod Çözme İçin İdealdır.
- 'L154, Kritik Güç Uygulamaları İçin Tasarlanmıştır.
- 4 adet İkili-Kodlanmış Girişlerin Kodunu, 16 adet Özel Çıkışlardan Birine Karşılık Gelecek Şekilde Çözer.
- Tekilleme İşlevini, Verileri Bir Giriş Hattından 16 Çıkışın Herhangi Birine Dağıtarak Yerine Getirir.
- Giriş Kenetleme Diyotları, Sistem Tasarımını Basitleştirir.
- Yüksek Çıkış Sürme Yeteneği, Düşük Empedans, Totem-Direği Çıkışları
- MOS TTL, DTL ve MSI Devreleriyle Tam Uyumluluk

SN54154... J VEYA W PAKETİ
SN54L154 ... J PAKETİ
N74154, SN74L154 ... J VEYA N PAKETİ
(ÜSTEN GÖRÜNÜŞ)



pozitif mantık: işlev tablosuna bakın

TİP	TİPİK ORTALAMA YAYINIM GECİKMESİ 3 DÜZEYLİ MANTIK	TİPİK ORTALAMA YAYINIM GECİKMESİ STROB	TİPİK GÜÇ HARCANIMI
'154	23 ns	19 ns	170 mW
'L154	46 ns	38 ns	85 nW

Tanımlama

Bu monolitik, 4 hattan 16 hatta kod çözücülerin her biri, G1 ve G2 strob girişlerinin her ikisi alçak düzeyde olduğu zaman 4 adet ikili kodlanmış girişlerin kodunu, 16 adet özel çıkışlardan birine karşılık gelecek şekilde çözmek için TTL devreleri kullanır. Tekilleme işlevi, iki strob girişinden biri alçak düzeyde iken, diğerinden verinin geçirilmesiyle ve 4 adet giriş hattının adres çıkış hatlarına bağlanmasıyla yerine getirilir. Strob girişlerinden herhangi biri yüksek düzeyde iken, bütün çıkışlar yüksek düzeydedir. Bu tetikleyiciler yüksek performanslı bellek kodu çözmelerinin gerçekleştirilmesinde çok uygundur. Çok çok yüksek hızlı sistemler için, SN54S138/SN74S138 ve SN545139/SN74S139 önerilmiştir.

Bu devreler, diğer birçok TTL ve DTL devreleriyle kullanılmak için tam uyumludur. Bütün girişler tamponlanmıştır ve iletim hattı etkilerini minimuma indirmek için giriş kenetleme diyotları eklenmiştir. Bu, sistem tasarımını basitleştirmektedir.

54 ve 54L serisi birimler, tam askeri sıcaklık değerleri aralığı olan -55°C'den 125°C'ye kadar sıcaklıklarda çalışmak için üretilmişlerdir.
74 ve 74L serisi elemanlar, 0°C'den 70°C'ye kadar olan sıcaklıklarda çalışmak için üretilmişlerdir.

SN54154, SN54L154, SN74154, SN74L154 TİPLERİ
4-HAT GİRİŞ/16-HAT ÇIKIŞ KOD ÇÖZÜCÜLER/ÇOĞULLAYICILAR

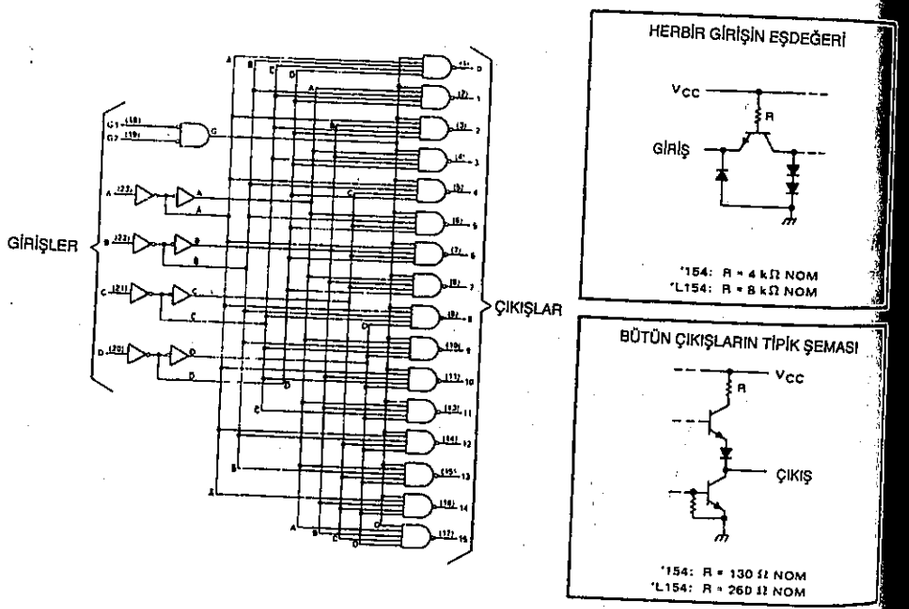
mantıksal

İŞLEV TABLOSU

GİRİŞLER						ÇIKIŞLAR																
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	A	A	A	A	A	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A	A	A	A	A	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A	A	A	A	Y	A	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A	A	A	A	Y	Y	Y	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A	A	A	Y	A	A	Y	Y	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A	A	A	Y	A	Y	Y	Y	Y	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A	A	A	Y	Y	A	Y	Y	Y	Y	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y
A	A	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	Y
A	A	Y	A	A	A	Y	Y	Y	Y	Y	Y	Y	Y	A	Y	Y	Y	Y	Y	Y	Y
A	A	Y	A	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	A	Y	Y	Y	Y	Y	Y
A	A	Y	A	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	A	Y	Y	Y	Y	Y
A	A	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	A	Y	Y	Y	Y
A	A	Y	Y	A	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	A	Y	Y	Y
A	A	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	A	Y	Y
A	A	Y	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	A	Y
A	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	A
A	Y	X	X	X	X	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Y	A	X	X	X	X	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Y	Y	X	X	X	X	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Y = yüksek düzey, A = alçak düzey, X = ilgisiz

İşlevsel blok diyagram ve giriş ve çıkışların şematik diyagramı



ASCII kodu

Tablonun karşısındakiler bütün kod takımını listeler. Mektup grupları yazılmayan denetim karakterleridir ve özellikle ilgilenilenler CR = satır başı (carriage return), LF = satır atlatma (line feed), SP = boşluk (space) ve DEL = silme (Delete)'dir. 7 bitlik kodla birlikte kullanılan eşlik biti, çift eşlik biti olarak gösterilmiştir ve her bir karakterin 8 bite tamamlanmasını sağlar.

Çift eşlik biti	7-bitlik sekizli kod	Karakter	Çift eşlik biti	7-bitlik sekizli kod	Karakter	Çift eşlik biti	7-bitlik sekizli kod	Karakter
0	000	NUL	0	053	+	0	126	V
1	001	SOH	1	054	,	1	127	W
1	002	STX	0	055	-	1	130	X
0	003	ETX	0	056	.	0	131	Y
1	004	EOT	1	057	/	0	132	Z
0	005	ENQ	0	060	Ø	1	133	[
0	006	ACK	1	061	1	0	134	\
1	007	BEL	1	062	2	1	135	]
1	010	BS	0	063	3	1	136	^
0	011	HT	1	064	4	0	137	_
0	012	LF	0	065	5	0	140	`
1	013	VT	0	066	6	1	141	a
0	014	FF	1	067	7	1	142	b
1	015	CR	1	070	8	0	143	c
1	016	SO	0	071	9	1	144	d
0	017	SI	0	072	:	0	145	e
1	020	DLE	1	073	;	0	146	f
0	021	DC1	0	074	<	1	147	g
0	022	DC2	1	075	=	1	150	h
1	023	DC3	1	076	>	0	151	i
0	024	DC4	0	077	?	0	152	j
1	025	NAK	1	100	@	1	153	k
1	026	SYN	0	101	A	0	154	l
0	027	ETB	0	102	B	1	155	m
0	030	CAN	1	103	C	1	156	n
1	031	EM	0	104	D	0	157	o
1	032	SUB	1	105	E	1	160	p
0	033	ESC	1	106	F	0	161	q
1	034	FS	0	107	G	0	162	r
0	035	GS	0	110	H	1	163	s
0	036	RS	1	111	I	0	164	t
1	037	US	1	112	J	1	165	u
1	040	SP	0	113	K	1	166	v
0	041	!	1	114	L	0	167	w
0	042	"	0	115	M	0	170	x
1	043	#	0	116	N	1	171	y
0	044	$	1	117	O	1	172	z
1	045	%	0	120	P	0	173	{
1	046	&	1	121	Q	1	174	|
0	047	'	1	122	R	0	175	}
0	050	(	0	123	S	0	176	~
1	051	)	1	124	T	1	177	DEL
1	052	*	0	125	U			

Soruların Cevapları

1.1 Kaynak program metninin karakterlerini gösteren kod biçiminde. Ayrıca bkz. Kısım 1.5.

1.2 Kaynak programın komut kısaltmalarını makine-kodu amaç programa dönüştürmek için. Ayrıca bkz. Kısım 1.5.

1.3 Düzenleyici.

1.4 Bkz. Şekil 1.2.

1.5 Programcının, programın çevrilme işlemini yönlendiren Çevirici programa bilgi girebilmesi için.

1.6 Sözde-iş kodları, yürütülebilir makine-kodu komutlarını göstermez.

1.7 GLBL, ORG, DS, END.

1.8 ORG.

1.9 Bir mutlak programda, programcı; amaç-programın, bellekte saklanacağı konumu, kaynak-programdaki ORG sözde-iş kodu aracılığıyla belirler. Yeri-değiştirilebilir bir programda amaç programın konumu, bir sonraki aşamada bir yerleştirici ya da muhtemelen bir bağlayıcı kullanarak belirlenir. Ayrıca bkz. Kısım 1.12, 1.13 ve 1.14.

1.10 Amaç programdaki tüm konuma-bağlı adresleri değiştirerek, yeniden yerleştirilebilir programın bellekte yerleşeceği adresi tesbit etmek için. Ayrıca bkz. Kısım 1.14.

1.11 Birbirinden ayrı program modüllerini, tek parça bir program halinde birbirine bağlamak için. Ayrıca bkz. Kısım 1.15.

2.1 Diğer birimler daha uygun olarak, MİB-tipi mikroişlemciler olarak adlandırılırmasına karşın, COPL420L, tek-yongalı bir mikrobilgisayar olarak adlandırılır.

2.2 *(a)* Denetim; *(b)* durum.

2.3 *(a)* 2; *(b)* 1; *(c)* 2; *(d)* 1.

2.4 Mikroişlemci paketi üzerindeki bacak sayısını ve dolayısıyla da üretim maliyetlerini en aza indirmek için.

2.5 8080, Z8000 (diğer mikroişlemciler de bu özelliğe sahiptirler).

2.6 Bkz. Kısım 2.3, "Z8000".

2.7 *(a)* 65.536; *(b)* 65.536; *(c)* 8.388.608.

2.8 INX B ve PUSH B. (Diğer birçok komut da 16-bitlik sözcükler üzerinde çalışır). Bkz. Ek A.

2.9 Bkz. Kısım 2.5.

2.11 Bkz. Kısım 2.6. "Kaydedici- ve bellek-yönelimli işlemciler".

2.12 *(a)* Bkz. Kısım 2.6, "Kaydedici- ve bellek-yönelimli işlemciler".
(b) Bkz. Kısım 2.7, "Doğrudan adresleme".
(c) Bkz. Kısım 2.7, "Dolaysız adresleme".
(d) Bkz. Kısım 2.7, "Kaydedici dolaylı adresleme".
(e) Bkz. Kısım 2.7, "İndeksli adresleme".

2.13 *(a)* İndeksli; *(b)* kaydedici dolaylı.

2.14 Birbiriyle karşılaştırılan bir veya daha çok sayıda bilgisayar üzerinde uygulanan bir veya daha çok sayıda hesaplama görevi. Farklı bilgisayarlar üzerinde uygulanan görevlerin verimlilikleri, bunların belirli bie uygulama için uygun olup olmadıklarına ilişkin bir kılavuz olarak kullanılabilir.

2.15

8080		*Saat saykılı*	*Bayt*
LDA	SAYI 1	13	3
MOV	B, A	5	1
LDA	SAYI 2	13	3
ADD	B	4	1
STA	SAYI 3	13	3
		48	11

Yürütme süresi = 48 mikrosaniye

Z80

LD	A, (SAYI 1)	13
LD	B, A	4
LD	A, (SAYI 2)	13
ADD	A, B	4
LD	(SAYI 3), A	13
		47

Yürütme süresi = 11.75 mikrosaniye

M6800

LDAA	SAYI 1	4	3
ADDA	SAYI 2	4	3
STAA	SAYI 3	5	3
		13	9

Yürütme süresi = 9 mikrosaniye
(Genişletilmiş adresleme kullanıldığı varsayılmıştır. Doğrudan adreslemenin kullanılması halinde yürütme süresi ve kullanılan bellek alanı azalacaktır).
Not: Diğer programlar, istenen hesaplama işlevini yerine getirecektir.

3.1 Bkz. Kısım 3.2.

3.2 Kesme ile başlatılan G/Ç işlemlerinde, işlemci, giriş ve çıkış işlemleri arasında kalan sürelerde diğer hesaplama görevlerini yerine getirebilir. Meşgul-tarama yönteminde ise, işlemci zamanının tamamı (yüzde 100'ü) giriş veya çıkış birimlerine hizmet vermekle geçecektir. Ayrıca bkz. Kısım 3.2.

3.3 Bkz. Şekil 3.10.

3.4 Bkz. Şekil 3.9.

3.5 Vektörlü kesme yönteminde, kesmenin kaynağını belirten kod MİB'e bildirilir. Vektörlenmemiş-kesme yönteminde ise, MİB'in, ilgili kesmeye hizmet vermeden önce kesmenin kaynağını mutlaka belirtmesi gerekir. Ayrıca bkz. Kısım 3.3.

3.6 Bkz. Kısım 3.3.

3.7 *(a)* Yerleşik G/Ç portları içeren tek-yongalı mikrobilgisayarlar.
(b) Ayrı G/Ç port yaklaşımı.
(c) Bellek-haritalama yaklaşımı.
Ayrıca bkz. Kısım 3.1.

3.8 Port G/Ç: 8080, Z80, Z8000 ve diğerleri. Bellek-haritalı G/Ç: M6800, 6502 ve diğerleri.
Ayrıca bkz. Kısım 3.1.

3.9 IN veya OUT komutu yürütüldüğünde alçak düzeye giden $\overline{IORQ}$ sinyali.

3.10 Bir kesme veya çıkış işlemi gerçekleştiğinde, ST0-ST3 durum hatları üzerinde 0010 ve 0011 Kodları gözükür. Ayrıca bkz. Kısım 3.1.

3.11 *(a)* 256; *(b)* 256; *(c)* 65.536 (ROM ve RAM bellek alanı); *(d)* 131.072.

3.12 Kesme, ki burada mikroişlemci denetimli sistemin güvenli bir biçimde kapatılması işlemini başlatır. Burada verilen yalnızca bir örnektir, başka durumların ortaya çıkması da mümkündür.

3.13

8080	3
M6800	1
Z80	1
Z8000	1
COPL420L	1

3.14 CMOS.

4.1 *Veri yolu* - verileri ve komutları, programın yürütülmesi sırasında MİB'ne ya da MİB'den diğer birimlere taşır.
Adres yolu - veri işlemleri sırasında, bellek konumlarının veya diğer işlevsel birimlerin adreslerini taşır.
Ayrıca bkz. Kısım 4.1.

4.2 IORQ, RD. Başka birçok cevap da kabul edilebilir ve A-F Ek'lerinde de başka cevaplar verilmiştir.

4.3 Bkz. Kısım 4.2.

4.4 Bu şekil, her bir birime bağlı olarak değişiklik gösterir, ancak bir TTL kapısı, bir NMOS çıkış kapısına oranla on veya yirmi kat daha fazla akım çeker. Ayrıca bkz. Kısım 4.2.

4.5 Bkz. Kısım 4.2.

4.6 Bkz. Kısım 4.2.

4.7 Yüksek, alçak veya yüksek empedans durumları. Ayrıca bkz. Kısım 4.2.

4.8 Bu diyagram, bir VE kapısı eklendiğinde Şekil 4.10'a benzemektedir.

4.9 9 MHz.

4.10 *(a)* 8228 yol denetleyicisine ait zamanlama strobu üretmek için.
(b) 8080 MİB'e ait eşzamanlı RESET sinyali üretmek için.
Ayrıca bkz. Kısım 4.3 ve Ek F.

4.11 Ana işlevi, veri ve durum sinyallerini, birleşik haldeki veri/durum yolu sinyalinden ayırmaktır. Ayrıca bkz. Kısım 4.3 ve Ek G.

4.12 Bkz. Kısım 4.14.

4.13 Çoğullanmış adres/veri yolunun, bir adres taşıdığını gösteren strob sinyalini üretmek için. Ayrıca bkz. Kısım 4.3.

4.14 Bu diyagram, dört-girişli VEYA kapısı yerine dört-girişli bir VEDEĞİL kapısı konulduğunda Şekil 4.17'ye benzemektedir. Ayrıca bkz. Kısım 4.4.

4.15 *(a)* Seri G/Ç; *(b)* bellek-haritalı paralel G/Ç. Ayrıca bkz. Kısım 4.5.

4.16 İlgili bit desenini veri yönü kaydedicileri (DDRA ve DDRB)'ne saklamak suretiyle. Ayrıca bkz. Kısım 4.5 ve Ek J.

4.17 Bkz. Kısım 4.5.

4.18 Senkron (eşzamanlı) iletimle olan temel farklılık, saat dalgabiçimini şeklindeki zamanlama bilgisinin veri ile birlikte iletilmesidir. Asenkron iletimde, zamanlama bilgisi, veri dalgabiçiminin içinde bulunmaktadır. Ayrıca bkz. Kısım 4.5.

4.19 Bkz. Şekil 4.22 ve 4.23.

4.20 Bkz. Kısım 4.5.

4.21 *(a)* Eşlik kontrolü; *(b)* ezme hatası kontrolü; *(c)* çerçeveleme hatası kontrolü. Ayrıca bkz. Kısım 4.5.

4.22 Avantajı - statik RAM, tazeleme gerektirmez. Dezantajı - statik RAM'da, tek bir entegre devre yongasında tutulabilecek bellek miktarı, dinamik RAM'a oranla daha azdır.
Ayrıca bkz. Kısım 4.6.

4.23 Bkz. Kısım 4.6.

4.24 Dinamik RAM'ın tazelenmesi için gereken devrenin büyük bölümü, MİB yongası içinde bulunmaktadır.

4.25 IC (Integrated Circuit:Entegre Devre) üzerindeki bacak sayısını en aza indirmek için. Ayrıca bkz. Kısım 4.6.

5.1 *(a)* Bilgi-işlem görevinin hacmi, tanımın genel-amaçlı olduğunu göstermektedir. Orta-büyüklükte ya da büyük boyutlu ticari işlemlere ilişkin genel-amaçlı bir bilgisayar, genelde bu türde uygulamalar için kullanılır.
(b) Uygulama, yarı-özel bir uygulama olmasına rağmen, muhtemelen genel-amaçlı bir bilgisayar üzerinde yapılacaktır.
(c) Özel.
(d) Özel.

5.2 *(a)* Küçük boyutlu üretim faaliyetleri açısından bakıldığında, geliştirme maliyetleri daha önemli olacaktır.
(b) Tüketim malzemeleri, çok sayıda üretilebilecek şekilde tasarlanmıştır. Bundan dolayı, bu türde bir sistem için, üretim maliyetleri normalde geliştirme maliyetlerinden daha önemli olacaktır.
(c) Bu problemin cevabı, kolayca verilemez. 5000 birimlik bir üretim miktarında, üretim maliyetleri, normalde geliştirme maliyetlerinin üzerine çıkar. Bununla birlikte, böyle bir durumda, işi karmaşık hale getiren etkenler mevcuttur. Denetleyicinin geliştirme ve üretim maliyeti, tüm sistemin toplam geliştirme ve üretim maliyetine oranla çok küçüktür. Uzun süre alması ve piyasanın rekabetçi yapısı da, denetleyicinin üretim süresinde, güncellenmesi ve muhtemelen yeniden tasarımının yapılmasını zorunlu hale getirecektir. Bu soruya kesin bir cevap verebilmek için, proje hakkında daha fazla bilgiye sahip olmak, piyasasının ve sistemi oluşturan diğer elemanların maliyetlerinin bilinmesi gerekir.

5.3 Programlar, özel-amaçlı sistemlerde ROM'da saklanır. Genel-amaçlı bir sistemde ise, yedek bellekte saklanır ve yürütülme öncesinde ana RAM belleğe aktarılır.

5.4 Özel-amaçlı bir sistemde bulunan entegre devrelerin sayısı, işlem güçleri karşılaştırılabilir düzeydeki genel-amaçlı bir sistemde bulunandan daha azdır.

5.5 Bkz. Kısım 5.2.

5.6 Bkz. Kısım 5.3.

5.7 Bkz. Şekil 5.3.

5.8 Bkz. 5.2, 5.3 ve 5.4 Kısımları.

6.1 Yazılımın tasarlanması. Ayrıca bkz. Kısım 6.1.

6.2 Bkz. Şekil 6.1.

6.3 Sistem yazılımının bir bölümü; yazıcılar, hareketlendiriciler, adım motorları vb. gibi fiziksel cihazların denetimi ile ilgili olacaktır. Bu yazılım, fiziksel cihazlar sisteme bağlanıp çalışılmadıkça tümüyle test edilmiş olmayacaktır. Ayrıca bkz. Kısım 6.2.

6.4 9.

6.5 0.0024 V.

6.6 60 birim.

6.7 75 sistem.

7.1 Bkz. Kısım 7.2, Kısım 7.3 ve Şekil 5.4.

7.2 Bir yazılım geliştirme istasyonunda, assembly ve yüksek-düzeyli d geliştirme çalışmaları yapabilmek için olanaklar bulunur. Tam- MDS'de ayrıca, devre-içi emülasyon, bellek yerini alma vb. destekd geliştirme olanakları da olacaktır. Ayrıca bkz. Kısım 7.1 ve 7.5.

7.3 Çeviriciler, derleyiciler vb. gibi standart destek programları için kalıcı saklama ortamları ve kullanıcı tarafından geliştirilen programların sak bilmesi için saklama alanı sağlamak için. Ayrıca bkz. Kısım 7.2.

7.4 *(a)* İşletim sistemi ve monitör.
(b) Düzenleyici.
(c) Çevirici.
(d) Derleyici(ler).
(e) Hata ayıklayıcı.
(f) Yükleyici.
(g) Bağlayıcı.
(h) Yerleştirici.
(i) Yazılım izleme.

7.5 Bkz. Kısım 7.3.

7.6 Bkz. Kısım 7.3.

7.7 Bkz. Kısım 7.3.

7.8 Bkz. Kısım 7.4.

7.9 Bkz. Kısım 7.4.

8.1 Yazılım.

8.2 Bkz. Kısım 8.2.

8.3 Programlama, tüm program tasarım işlemlerini tanımlamak için kullanılan bir terimdir. Bu işlem, program modüllerinin tanımlanması ve ardından da modül yapısının tasarlanması için programlama görevinin bölümlere ayrılmasından oluşmaktadır. Son aşama, program kodlama olarak adlandırılır ve programın yapısını oluşturmak için kaynak kodunun, assembly veya yüksek-düzeyli dilde yazılmasıdır.

8.4 Bkz. Kısım 8.3.

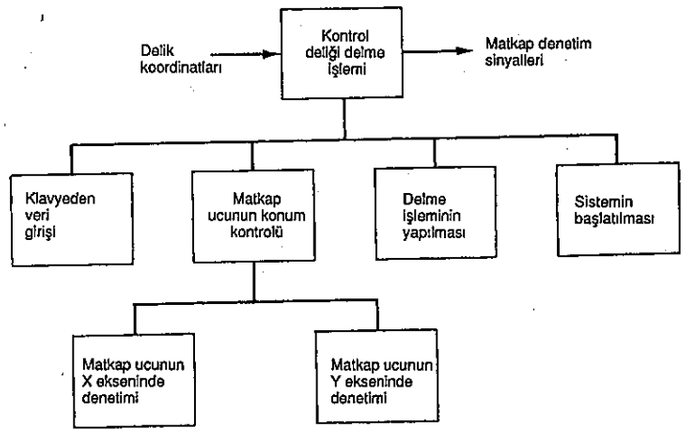
8.5 Bkz. Kısım 8.5.

8.6 Yüksek-düzeyli dil.

8.7 Programlar, örneğin yazıcı sürücü yordamları gibi çevresel donanımın yakından denetlenmesini gerektirir. Ayrıca, yürütme hızının başlıca öneme sahip olması halinde Assembly dili de tavsiye edilebilir.

8.8 Bkz. Kısım 8.7.

8.9 Bkz. Şekil A.



Şekil A Soru 8.9'a ait çözüm.

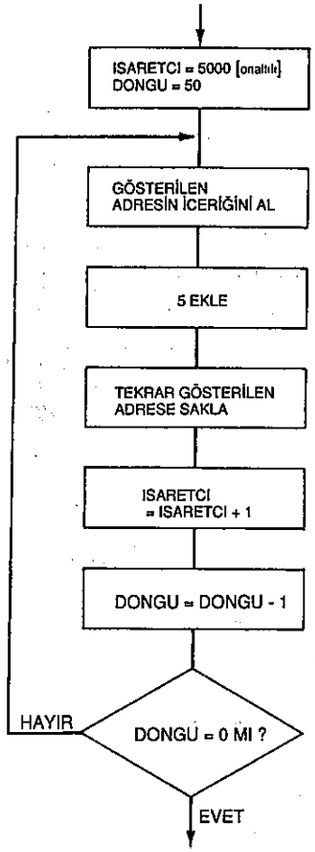
8.10 Program, DONGU=7 değeri dışında, Şekil 8.10 ve 8.11'de gösterilen programa benzemektedir.

8.11 Bkz. Şekil B.

8.12 Bkz. Şekil C.

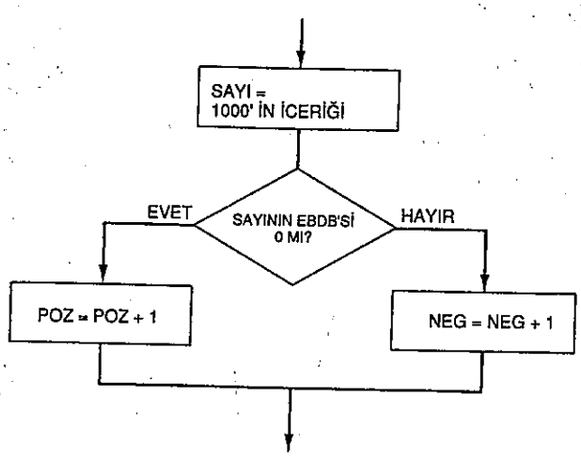
8.13 Bkz. Şekil D.

8.14 Bkz. Şekil E.



Şekil B Soru 8.11'in çözümü.

```
SAYI = 1000 ADRESİNİN İCERİĞİ
IF SAYININ EBDB'SI = 0 İSE
    THEN
        POZ = POZ + 1
    ELSE
        NEG = NEG + 1
END IF
```

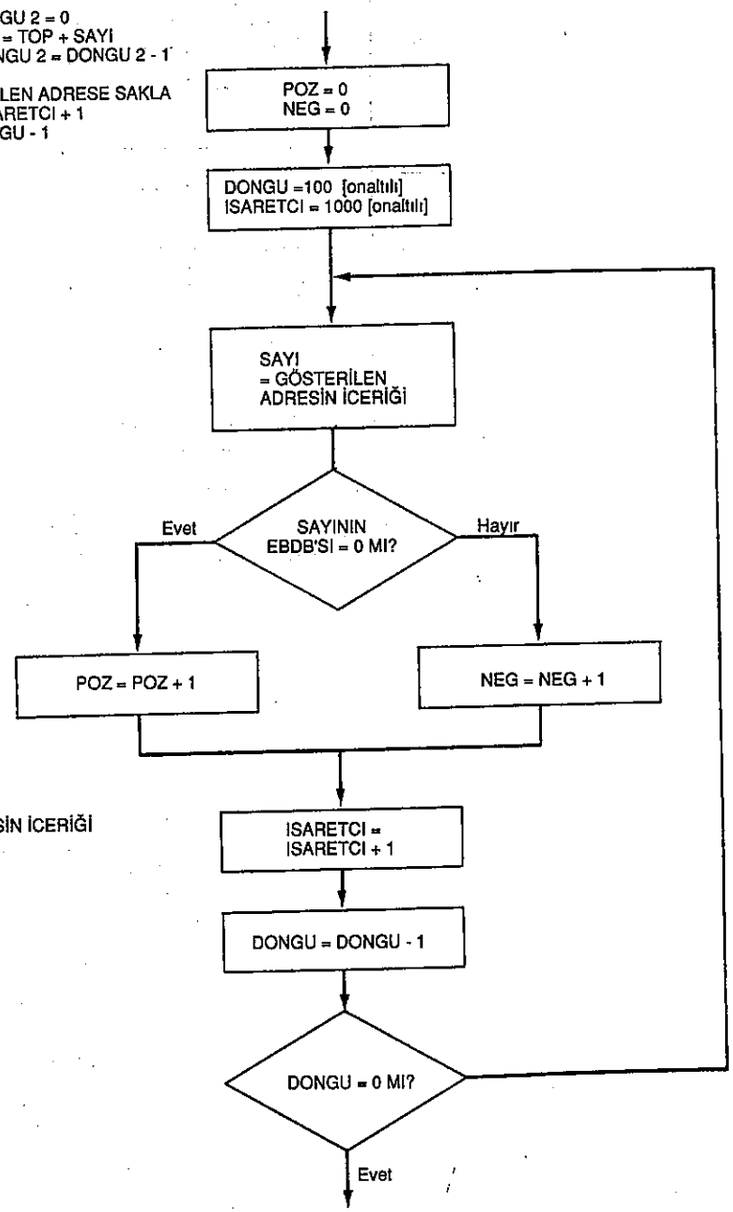


Şekil C Soru 8.12'nin çözümü.


```
DONGU 1 = 50
ISARETCI = 5000 [onaltılı]
DO UNTIL DONGU 1 = 0
        SAYI = GÖSTERİLEN ADRESİN İCERİĞİ
        TOP = 0
        DONGU 2 = 5
        DO UNTIL DONGU 2 = 0
                TOP = TOP + SAYI
                DONGU 2 = DONGU 2 - 1
        END DO
        TOP'I GÖSTERİLEN ADRESE SAKLA
        ISARETCI = ISARETCI + 1
        DONGU = DONGU - 1
END DO
```

Şekil D Soru 8.13'ün çözümü.

```
POZ = 0
NEG = 0
DONGU =100 [onaltılı]
ISARETCI = 1000 [onaltılı]
DO UNTIL DONGU = 0
    SAYI = GÖSTERİLEN ADRESİN İCERİĞİ
    IF SAYININ EBDB'SI = 0 İSE
        THEN
            POZ = POZ + 1
        ELSE
            NEG = NEG + 1
    END IF
    ISARETCI = ISARETCI + 1
    DONGU = DONGU - 1
END DO
```



Şekil E Soru 8.14'ün çözümü.

9.1 En azından her iki veya üç entegre devre başına 10-100 mikro Farad'lık seramik disk ayırma kondansatörü. Ayrıca, her beş veya on entegre devre başına 10-100 mikro Farad'lık elektrolitik veya tantal ayırma kondansatörü kullanılması da istenebilir.

9.2 Bkz. Kısım 10.1.

9.3 RAM'a ilişkin okuma/yazma test saykılının ROM'a uygulanamayacağı açıktır. ROM'a ilişkin test işlemi, bundan dolayı, içeriği bilinen ROM'un okunması şeklinde olacaktır. Ayrıca bkz. Kısım 9.3.

9.4 LED'ler, veri alan (akım çeken) birimlerin simülasyonu, Anahtarlar, veri ileten (kaynaklayan) birimlerin simülasyonuna ilişkindir.

9.5 Temel neden, karmaşıklığın artmasıyla birlikte, tasarımdan veya üretimden kaynaklanan hataların birden fazla olma olasılığının artmasıdır. Mevcutlar arasından bir hatanın bulunması daima zordur ve zorluk düzeyi, kabaca, mevcut hata sayısının kuvveti ile doğru orantılı olarak artar. Donanım ve yazılımın tasarımında modüler yaklaşım, herhangi bir anda test edilmekte olan modülde birden fazla hata bulunmamasını sağlayarak bu sorunun üstesinden gelmeye çalışır.

10.1 Bkz. Kısım 10.1.

10.2 *(a)* 8080 için mevcut bulunan destek.
(b) Tasarım süresine ilişkin daha önceki deneyim ve tasarlanacak sistemin üretim hızı.

10.3 Başlıca nedeni, bazı hesaplama işlemlerinin, saat darbeleri arasında belirli zaman aralıkları gerektirmesidir. Ayrıca bkz. Kısım 10.5.

zamanlama aralıkları süresince işlenmekte olduğundan, basit bir ...iktirme programı kullanılamaz. Ve, çeşitli hesaplama işlemleri ...ecikmelerin tahmini ve denetlenmesi zorlaşacaktır. Gerçek-...at, karmaşıklık düzeyi, incelenen bu durumdan çok daha ...ecektir. Ayrıca bkz. Kısım 10.5.

...ktirir. Bkz. Kısım 10.5.

```
SAYI = 1
IF SAYIN
  THEN
  ELSE
    NE
END IF
```

Şekil C Soru 8.12'nin ç

10.7 Yığının ve programın yürütülmesi sırasında üretilen geçici değişkenlerin tutulması için.

10.8 0000.

10.9 Bkz. Kısım 10.9.

11.1 Bkz. Kısım 11.2.

11.2 10 yıl.

11.3 Ayrıca bkz. Ek N.

11.4 525 mW. Ayrıca bkz. Ek N.

11.5 Maksimum 55, minimum 45 milisaniye.

11.6 *(a)* +25 V.; *(b)* +5 V.

11.7 2716'nın tipik erişim süresi, 300 - 500 nsn.
2114'ün tipik erişim süresi, 200 - 450 nsn.
Ayrıca bkz. Ek L ve N.

11.8 5.

11.9 Programda değişiklikler yapmak üzere EPROM'un içeriği birçok kez silinebilirken PROM silinemez özelliktedir, bu nedenle, program üzerinde değişiklik yapıldığı her seferinde yeni bir bellek birimi kullanılması zorunluluğundan kurtulunmuş olur.

11.10 *(a)* UVEPROM'ların programlanması, EAROM'lardan daha uzun sürede yapılabilir.
(b) UVEPROM'ların silinmesi, EAROM'lardan daha uzun süre alır.
Bu dezavantajların her ikisinden çıkarılacak bir sonuç; eğer özel-amaçlı bir sisteme ait destek yazılımın geliştirilmesinde tek yöntem olarak UVPROM kullanılıyorsa, test-silme-yeniden programlama çevriminin kabul edilemeyecek uzunlukta uzun süreler alacağıdır. EAROM kullanılması, bu süreyi azaltacaktır, ancak bunlarında da, Kısım 11.4'de anlatıldığı gibi kendine özgü problemleri bulunmaktadır.

12.1 Bkz. Kısım 12.3.

12.2 Bkz. Kısım 12.4.

12.3 Görsel inceleme. Ayrıca bkz. Kısım 12.4.

12.4 Osiloskop cihazı; d.c. gerilim kaynağının çıkış düzeylerindeki, mikroişlemci sistemleri için önemli olabilen, ancak normal test aletleriyle algılanamayan, hızlı değişimleri algılayabilme yeteneğindedir. Ayrıca bkz. Kısım 12.7.

12.5 Bkz. Kısım 12.6.

12.6 Bkz. Kısım 12.6.

12.7 Sistemin saatinden ayrı olarak, bir osiloskopla kolayca görülebilen diğer dalgabiçimleri, mikroişlemci sistemlerde çok sık olarak ortaya çıkmazlar. Ayrıca bkz. Kısım 12.7.

12.8 Osiloskop cihazı, yükselme süresi ve aşırı yükselme gibi mantık analizörünün ölçemeyeceği önemli parametreleri ölçebilir.

12.9 $\overline{CS}$, içinde A15=1 olduğu bir adresin okunması veya yazılması yapılırken etkinleşir. 8000-FFFF arasındaki tüm adresleri, bir çevrim boyunca sürekli olarak okuyan bir programın sözde-kodları, Şekil F'de gösterilmiştir. Bu program çalışırken, düzenli $\overline{CS}$ darbe dizisinin, bir osiloskop aracılığıyla gözlenmesi gerekir.

```
TEKRAR = 1
DO UNTIL TEKRAR = 0
        ISARETCI = 8000
        DONGU    = 8000
        DO UNTIL DONGU = 0
                GÖSTERİLEN ADRESİ OKU
                ISARETCI = ISARETCİ + 1
                DONGU = DONGU - 1
        END DO
END DO
```

Şekil F

Osiloskopun ikinci kanalından, $\overline{CS}$, ALÇAK düzeyde olduğunda daima YÜKSEK düzeyde olacak olan R/$\overline{W}$ sinyali izlenebilir. Okuma yerine yazma yapan benzer bir program da, yine bir $\overline{CS}$ darbe dizisi üretir, ancak bu durumda, $\overline{CS}$, ALÇAK olduğunda R/$\overline{W}$ YÜKSEK düzeyde olacaktır. Ayrıca bkz. Kısım 12.8.

12.10 Bkz. Kısım 12.10

13.1 Senkron örneklemede, örnekleme süreleri, test edilen sistemin saati nedeniyle sabittir. Asenkron örnekleme ise, örnekleme sürelerini belirlemek için mantık analizörü içindeki saati kullanır. Ayrıca bkz. Kısım 13.2.

13.2 Bkz. Kısım 13.2 ve Şekil 13.3.

13.3 Bkz. Kısım 13.2.

13.4 Bkz. Kısım 13.2 ve Şekil 13.5 ve 13.6.

13.5 Bkz. Kısım 13.3.

13.6 1101.

İNDEKS

Açık-kollektörlü devreler, 116
AD dönüşümü, 172
Adres: çoğullama, 138
dolaysız, 60
doğrudan, 60
indeksli, 62
kaydedici dolaylı, 61
kod çözme, 122
modları, 58
örtük, 59
Bağlayıcı, 35, 185
Bellek:
erişim süresi, 136
tasarımı, 240
testi, 224
türleri, 135
Bölüşüm kararı, 170
COP420L, 50
Çevirici 16, 184
listelemesi, 21
Çoğullanmış:
adres, 138
sinyal, 46
DDT, 187
Denetleyici:
çamaşır makinesi, 149
tek-yongalı, 44
Değerlendirme testi, 78
Devre-içi emülatör, 192
DO UNTIL yapısı, 207
DO WHILE yapısı, 213
Donanım birimlerinin seçimi, 234
Durum sözcüğü, 100
Düzenleyici, 182
EAROM, 136, 265
Emülatör, 190
EPROM, 136, 265
programlanması, 259, 265
Eşlik kontrolü, 133
Geliştirme sistemi, 157
G/Ç:
bellek-haritalı, 86
portu, 83, 240
seri, 128
senkronizasyonu, 88
tek-yongalı, 82
Hata ayıklayıcı, 186
Hata:
bulma, 274
tipik, 275
yazılım, 293
IF-THEN-ELSE yapısı, 216
İşaret analizörü, 308
İşlem gücü, 42
İşletim sistemi, 181
İzleme (Monitör) programı, 186
Kesme:
güç kesilmesi, 92
kullanımı, 242
maskelenebilir, 91
vektörlendirilmiş, 92
Kod çözücü, 122
Komut takımı, 55
M6800, 47
Maliyet hesaplama, 167
Mantık:
analizörü, 297
probu, 278
MDS:
destek yazılımı, 159, 181
düzenlemesi, 158, 180
Mikroişlemci:

mimarisi, 53
seçimi, 41
yonga takımı, 117
enerji gereksinimleri, 104
tek-yongalı, 44
Multimetre, 277
Osiloskop, 279
Örnekleme:
asenkron, 303
senkron, 302
Özel-amaçlı bilgisayar, 147
PIA/PIO, 127
Program:
bakımı, 197
bölümleme, 202, 245
modüler, 199
mutlak, 32
yapısal, 206, 246
yeniden-yerleştirilebilir, 34
PROM, 136
eriyebilir bağlantılı, 264
programlayıcı, 193
RAM:
dinamik, 137
statik, 137
ROM, 135
simülatörü, 270
Saat:
gerçek-zamanlı 236
gereksinimleri, 117
Sayıcı-zamanlayıcı, 142
Simülatör, 190
Sistem:
akış diyagramı, 163, 231
spesifikasyonu, 167, 230
Sıralı yapı, 207
Sözcük:
işlemcisi, 152
uzunluğu, 51
Sözde-opkod, 27
Sözdizim kuralları, 29
Tampon, 112
Tek sistemkartlı bilgisayar, 170
Test:
dalgabiçimleri, 281
kartı, 289
programı, 284
serbest çalışan, 286
UART, USART, 128
UVEPROM, 260
Üç-durumlu sürücüler, 114
Veri:
ekseni, 301
yolu, 45
Yerleştirici, 35, 185
Yol:
adres, 45
denetleyicisi, 120
tekilleyici, 120
sürücüsü, 114
veri, 45
Yukarıdan-aşağıya yöntemi, 205
Yükleyici, 35
Yüksek-düzeyli dil, 186
Z80, Z8000, 49
Zaman ekseni, 301

TERİMLER SÖZLÜĞÜ

Adres yolu (Address Bus): Üzerinden adres bilgilerinin taşındığı yol.

Akümülatör (Accumulator) : Aritmetik ve mantık işlemleri sırasında, üzerinde işlem yapılacak verinin tutulduğu yer.

Alçakta-etkin/ Yüksekte-etkin (Low active/-High active): Bir elektronik birimin, mantıksal 0'da (denetim darbesinin mantıksal 0 düzeyinde) etkinleşmesi alçakta-etkin, mantıksal 1'de etkinleşmesi ise yüksekte-etkin olma özelliğidir.

Altyordam (Subroutine): Bir programın içinde yer alan ve programda tekrarlanan belirli işlemleri yerine getirdikten sonra denetimi ana programa aktaran program parçalarıdır.

ALU (Arithmetic Logic Unit: Aritmetik ve Mantık Birimi): Bir merkezi işlem biriminde aritmetik ve mantık işlemlerinin yerine getirildiği birim.

Amaç program (Object program): Bilgisayar belleğine yüklenmeye hazır biçimde derlenmiş ya da çevrilmiş program.

Analog (Analog): Niceliklerin; gerilim, akım vb. gibi fiziksel değişkenlerle gösterimidir.

Analog-sayısal dönüştürücü (Analog to Digital Converter): Sürekli değişen analog sinyalleri, sayısal sinyallere dönüştüren elektronik devre.

Arabirim (Interface): Birimlerarası bağlantıyı ve bu bağlantıdaki uyumu sağlayan birim.

ASCII (American Standard Code for Information Interchange: Bilgi Değişimi için Amerikan Standart Kodu).

Assembly dili (Assembly language): Genellikle her bir komutu bir makine koduna karşılık gelen donanım-bağımlı sembolik dil.

Baud : Bit/sn olarak ifade edilen bir iletim hızı birimi.

Bellek (Memory) : Sayısal bilgisayarlarda bilgileri geçici veya kalıcı olarak saklamak üzere kullanılan ve genelde yarıiletken birimlerden oluşan kesim.

Bayrak (Flag): Belli bir koşulu ya da durumu göstermek üzere 1 ya da 0 değeri alan bir bitlik gösterge.

Bayrak işaretleri

S - Sign (İşaret)
NZ - Not Zero (Sıfır değil)
Z - Zero (Sıfır)
C - Carry (Elde)
NC - Not Carry (Elde Yok)
AC - Auxiliary Carry (Yardımcı Elde)
PO - Parity Odd (Tek Eşlik)
PE - Parity Even (Çift Eşlik)
P - Plus (Pozitif)
M - Minus (Negatif)

BCD (Binary Coded Decimal: İkili kodlanmış ondalık): Ondalık tabandaki sayıların, bir grup ikili sayı ile temsil edildiği sayı sistemi.

Bellek hücresi (Memory cell): Bir belleğin, bir bitlik bilgi saklanabilen ve okunabilen elemanı.

Bellek konumu (Memory location): Bir bilgisayar belleğinde, genelde bir baytlık bilginin saklandığı ve belirli bir adrese sahip saklama yeri.

Bit (Bit- BInary digiT): İkili basamak ifadesinden türetilmiş en küçük bilgi birimini ifade eden birim.

CP/M (Control Program for Microcomputers): Digital Research firması tarafından geliştirilmiş olan ve birçok kişisel bilgisayarda kullanılan, ancak günümüzde kullanımdan kalkmış bulunan işletim sistemi.

CRT (Cathode Ray Tube: Katot-ışınlı tüp): Elektron tabancasından çıkan elektron hüzmesinin, yatay ve düşey plakalarla saptırılarak ekrana yansıtılması ilkesine göre çalışan vakum tüpü.

Çevirici (Assembler): Sembolik komutları, bilgisayarda kullanılabilir komutlara çeviren program.

Çapraz-çevirici (Cross-assembler): Belli bir bilgisayar sistemi için yazılmış programları, farklı bir sistemde kullanılabilecek biçimde (örn. 6800'den

8080'e) çeviren çevirici program.

Çevrebirimi (Peripheral): Bilgisayarın kendi parçası olmayan, ancak bilgisayara arabirim üzerinden bağlanarak çalışabilen birimlerdir. Örn., yazıcılar, kart okuyucular, teyp birimleri, vb.

Çoğullayıcı (Multiplexer): Girişine uygulanan sinyallerden birini, denetim girişlerine göre çıkışa aktaran sayısal birim.

Denetim yolu (Control Bus): Bilgisayarı oluşturan birimlerin koordinasyon içinde çalışabilmesi için gereken denetim sinyallerinin iletildiği veya alındığı yol.

Denetleyici (Controller): Elektriksel cihazların çalışmasını denetleyen, denetim sinyallerinin alış-verişini yöneten donanım.

DIL paketi (Dual-In Line package): Çok kısa aralıklarla iki sıra bağlantı yapılabilmesini sağlayan entegre devre paketi.

Dinamik RAM (Dynamic RAM): İçerdiği bilgileri zaman içinde kaybedebilen ve bu nedenle belirli aralıklarla tazelenmesi gereken rastgele-erişimli bellek türü.

Donanım (Hardware): Bilgisayarı oluşturan fiziksel bileşenler ve devrelerin tümü.

Döngü (Loop): Bir bilgisayar programında, istenen sonuç elde edilinceye kadar tekrarlanan bir dizi komut.

Durdurma biti (Stop bit): Asenkron seri iletimde, gönderilen bir baytlık bir bit dizisinin sonunda yer alan ve 1 ya da 2 bit uzunlukta olabilen bit.

Durum (State): Bilgisayar içindeki herhangi bir birimin herhangi bir andaki durumu. Örn., wait state: bekleme durumu, vb.

Düzenleyici (Editor): Metinlerin yazılabilmesine ve çeşitli biçimlerde düzenlenebilmesine olanak tanıyan program.

Emülasyon (Emulation): Bir bilgisayar sisteminin, bir başka bilgisayar sistemi için hazırlanmış veri ve programları kullanabilmesine ve aynı sonuçları alabilmesine olanak tanıyan teknik.

Erişim Süresi (Access time): Verilerin bilgisayar belleğinden okunmak üzere çağrılmaları ile okunabilir ve kullanılabilir duruma geçmeleri arasındaki süre.

Eşlik biti (Parity bit): Verideki 1 veya 0 bitlerinin sayısının tek ya da çift olmasına göre 0 veya 1 olarak alarak veriye eklenen ve bu yolla, asenkron seri iletimde kontrol olanağı sağlayan bit.

Etiket (Label): 1. Bir adresin sembolik adı. 2. Bir bilgisayar programındaki bir deyimi veya bilgi bellek manını tanımlayan bir ya da birkaç karakter.

Ferrit-çekirdekli bellek (Ferrite-core memory): İçinden okumayazma iletkenleri geçen ve ferrit malzemeden yapılmış çok küçük sargılardan oluşan bellek türü. Birkaç özel uygulama dışında artık kullanılmamaktadır.

Firmware (Bellenim): Genelde bir ROM'a kaydedilen ve temel makine komutlarından oluşan yazılım; bu yazılımın tutulduğu birim. (Yerleşik program)

Flip-flop (Flip-flop): Kararlı iki durumdan birinde bulunan ve bir darbe uygulanmadıkça durum değiştirmeyen elektronik devre.

Format (Format: Biçim): Bilgilerin bir tablo, sayfa, dosya ya da mesaj üzerinde önceden belirlenmiş bulunan biçimi, düzeni.

Gerçek-zamanlı (Real-time): İlgili fiziksel olayın ortaya çıktığı anda işlem yapabilen bilgisayar sistemi ya da programı.

Getirme saykılı (Fetch cycle): Bilgilerin belleğden alınıp mikroişlemciye getirilmesi.

Hata ayıklama (Debug): Bir programın içerdiği hataların ayıklanarak giderilmesi.

Hata ayıklama programı (Debugger): Programlardaki hataları bularak gidermek üzere hazırlanmış özel yardımcı program.

İki-kararlılar (Bistables): Kararlı iki durumu bulunan flip-flop.

İki-yönlü (Duplex): Her iki yönde de iletimin mümkün olduğu iletişim sistemi.

İşlemkodu (Opcode): Bkz:İşlem kodu

İşlem kodu (Operation code): Makinenin belli bir işlemi yapmasını sağlayan ve bir veya birkaç bayt uzunlukta olabilen makine düzeyli komut.

İşlenen (Operand) : Bilgisayarda bir işleme giren ya da bir işlemin sonucu olarak ortaya çıkan nicelik.

Kalıcı (Non-volatile): Uygulanan besleme gerilimi kesildiğinde bile içerdiği bilgileri tutabilen bellek türü.

Kalıcı-olmayan (Volatile): Uygulanan besleme gerilimi kesildiğinde içerdiği bilgilerin silineceği bellek türü.

Karakter üreteci (Character generator): Bir bilgisayarda mevcut bulunan tüm kod ve sembollerin tutulduğu ve üretildiği genelde ROM veya EPROM şeklinde olan birim.

Kaydedici (Register): Genelde bilgisayarın sözcük boyuna eşit uzunlukta olan ve çeşitli amaçlarla kullanılan yerel bellek. Örn. A kaydedicisi.

Kaynak program (Source program): Bilgisayarda yürütülmeden önce derlenmesi, çevirilmesi ya da yorumlanması gereken program.

Kesme (Interrupt): Program akışının belli bir nedenle kesilmesi. Bu durumda, kesmeye neden olan işlem ya da olayla ilgilenir ve denetim, kesme hizmet yordamı adı verilen bir yordama aktarılır.

Klavye (Keyboard): Bilgisayara karakter girmek için kullanılan G/Ç birimidir.

Kod çözücü (Decoder): Bilgisayarlarda, girişine uygulanan sinyallere bağlı olarak bir ya da daha fazla çıkış veren devre ya da sistem.

Kodlayıcı (Encoder): Bilgisayarlarda, her seferinde sadece bir girişin uyarıldığı ve her girişin belli bir çıkış grubu çıkardığı devre ya da sistem.

Komut (Instruction): Bilgisayarda belli bir işlemi tanımlayan, bir veya birkaç adresle birlikte ya da adres olmaksızın verilen karakter takımı.

Komut saykılı (Instruction cycle): Bilgilerin belleğden alınıp getirilmesi (getirme saykılı) ve yürütülmesinden (yürütme saykılı) oluşan çevrim.

Komut takımı (Instruction set): Bir merkezi işlem birimi tarafından yürütülen temel işlemlere karşılık gelen komutlar topluluğu.

Koşullu atlama (Conditional jump): Bilgisayar programının komut yürütme sırasının belli bir koşula (örneğin, bir değişkenin değerinin sıfırdan büyük ya da küçük olması koşuluna) bağlı olarak değişmesi.

Koşulsuz atlama (Unconditional jump): Bilgisayar programının komut yürütme sırasının, ilgili Atlama komutuna gelindiğinde herhangi bir koşula bağlı olmaksızın değişmesi.

Kurma/Sıfırlama (Set/Reset): Bir bitlik bellek alanının içeriğini 1 yapmak, Kurma (Set); sıfır yapmak ise Sıfırlama (Reset) olarak adlandırılır.

Kütüphane (Library): Standart bilgisayar programları, modülleri ve yordamlarından oluşan modüller dizisi.

LED (Light-emitting diode: Işık Yayan Diyot): İleri öngerilim uygulandığında ışık yayan bir diyot türü.

Makro-çevirici (Macro-assembler): Kaynak programın her bir deyimine karşılık makine kodunda birkaç komut içeren yöntem üretmek üzere komut kısaltmalarının kullanıldığı çevirici program.

Mandal (Latch): Saat darbeleriyle değil de, giriş işaretinin değişmesiyle konum değiştiren flip-floplardır.

MİB-Merkezi İşlem Birimi (CPU-Central Processing Unit): Bir bilgisayarın aritmetik, mantık ve denetim devrelerini içinde bulunduran bölümü.

MODEM (MOdulator+DEModulator): Asenkron seri haberleşmede, bilgisayarın 0 ve +5 V düzeyleri şeklindeki mantıksal sinyallerini, iletim hatlarından iletilebilecek biçimde değişen frekanstaki alternatif sinyallere; alışta da, bu alternatif sinyalleri tekrar mantıksal 0 ve 1 sinyallerine dönüştüren cihaz.

MOS : Metal-oksit yarı iletken anlamında kısaltma. MOS teknolojisinde, bir yarı iletken alt katman üzerinde oksit yalıtkan tabaka ve en üstte de bir metal elektrot bulunur. MOS tekniğiyle üretilmiş transistörler MOSFET (MOS-Alan Etkili Transistör) olarak adlandırılır. P-MOS teknolojisinde yalnızca P-tipi, N-MOS teknolojisinde yalnızca N-tipi kanal ve CMOS (Complementary MOS) teknolojisinde ise hem P-tipi ve N-tipi kanal kullanılır.

NMI (Non-maskable interrupt : Maskelenemez kesme): Bir programın yürütümünü kesebilen ve Kesmeleri Yetkisizle (DI) komutu ile devre dışı bırakılamayacak öncelikte bulunan kesme türü. Güç kaynağı arızası, maskelenemez kesmeye bir örnektir.

Onay (Handshaking): Asenkron seri veri iletiminde, karşılıklı iki birimin iletişim kurmak üzere yaptıkları sinyal alışverişidir.

Osilatör (Oscillator): Elemanların değerleri ile belirlenen frekansta salınım yapan devre.

Paralel-seri dönüştürücüler (Parallel to Serial Converter): Girişine uygulanan paralel biçimdeki veriyi seri biçime (ardışık bit dizilerine) dönüştüren elektronik devre.

Polish form (Polish biçimi) : Polonyalı mantıkçı J. Lukasiewicz tarafından geliştirilen bir tekniktir. Bu gösterim biçiminin amacı, cebirsel ifadelerdeki operatörleri, işlenenlar arasında değil, işlenenlerin ardından (örn. a+b yerine ab+ gibi) yazabilmektir.

Port (Port): Giriş ya da çıkış amacıyla kullanılan fiziksel bağlantı birimi.

Saat (Clock): Bilgisayardaki birimlerin eşzamanlı biçimde çalışabilmesini sağlamak üzere bir üreteçten üretilen periyodik sinyaller.

Saat üreteci (Clock generator): Eşzamanlamayı sağlamak üzere periyodik sinyaller üreten ve genelde yüksek kararlılık sağlayabilmek için kristal içeren elektronik devre.

Sayısal-analog dönüştürücü (Digital to Analog Converter): Sürekli değişen analog sinyallerin sayısal sinyallere dönüştüren elektronik devre.

Sektör (Sector): 1.Dairesel veri saklama ortamlarının en küçük adres bölümü. 2.Disk/disket üzerindeki izlerin komşu bölümlerinden her biri. Sektör büyüklükleri genelde 128, 256 veya 512 baytlıktır.

Sembolik adres (Symbolic address): Bir programın içinde, bir sözcüğü, bir işlevi ya da bir bilgiyi, bilginin program içindeki yerinden bağımsız olarak tanıtmak üzere kullanılan etiket.

Senkron veri iletimi (Synchronous data transmission): Verici ve alıcı cihazların sürekli olarak aynı frekans ve faz ilişkisinde bulundukları iletim.

Seri-paralel dönüştürücüler (Serial to Parallel Converter): Girişine uygulanan seri (ardışık bit dizisi) biçimli veriyi paralel biçimdeki veriye dönüştüren elektronik devre.

Sinyal (Signal) : 1. Bilgi iletebilecek görülür, işitilir ya da başka biçimlerde elektriksel işaret. 2. Bilgisayar içindeki çeşitli işlemleri denetlemek ve/veya zamanlamak için kullanılan elektriksel işaret.

Simülasyon (Simulation: Benzetim): Fiziksel sistemler veya olguların, işlev ve özelliklerinin modeller, bilgisayarlar veya diğer donanımlar aracılığıyla oluşturulması.

Statik RAM (Static RAM): Uygulanan enerjinin sürdürülmesi halinde, Dinamik RAM'in tersine tazeleme gerektirmeden içerdiği bilgileri tutan rastgele-erişimli bellek türü.

Sürücü (Driver): Birden fazla sayıda TTL sinyalinin bulunması halinde bir ara hattaki sinyalleri yeniden biçimlendirmede kullanılan yükselteç devresi.

Tampon (Buffer): Sürücü bir devre ile sürülen devreler arasındaki sinyal alışverişini düzenleyen ara devre.

Tekilleyici (Demultiplexer): Bir çoğullayıcı (multiplexer) tarafından birleştirilen iki veya daha çok sinyali birbirinden ayıran cihaz.

Tek-yönlü (Simplex) : Her seferinde sadece tek bir yönde bilgi göndermeye izin veren haberleşme türü.

Teletayp (Teletype): Bir klavye ve yazıcı üniteden oluşan ve konsol veya terminal olarak kullanılan giriş/çıkış birimi. (Bu terimin orijinali, Teletype Co. firmasının ürettiği teleks cihazlarına vermiş olduğu isimdir).

Terminal (Terminal): 1.Bağlantı ucu, uç. 2. Bilgilerin bir iletim ağına girdikleri veya ağdan çıktıkları nokta; bu amaçla kullanılan birim.

Tersleyici (Inverter): Girişin tersini çıkış veren mantık devresi çıkış girişin her zaman DEĞİL'i olur. Örnek. Giriş 1 ise çıkışı 0 olur. Giriş 0 ise çıkışı 1 olur.

TTL-Transistör Transistör Mantık (Transistor--Transistor Logic): Yüksek hızda entegre mantık devreleri ailesi. Genelde girişleri çok emitörlü, çıkışları ise push-pull'dur.

UART: Asenkron seri veri haberleşmesinde kullanılan genel-amaçlı alıcı-verici entegre devresi (Universal Asynchronous Receiver-Transmitter'in kısaltması)

USART : Seri veri haberleşmesinde kullanılan, hem senkron (eşzamanlı) ve hem de asenkron çalışabilen alıcı-verici entegre devresi. (Universal Synchronous Asynchronous Receiver-Transmitter).

Üç-durumlu (Three-state) : Elektronikte, aynı hatta bağlı paralel birimlerin, giriş empedanslarının teorik olarak sonsuz olduğu, bu nedenle devreden akım çekmedikleri, devreye akım vermedikleri üçüncü durumu göstermek üzere kullanılan terim.

VDU (Video Display Unit : Görsel Görüntüleme Birimi): Bilgisayar ekranı anlamında İngiliz'lerin kullandığı bir kısaltma. Eş anlım. CRT-Display.

Veri yolu (Data Bus): Verilerin, MİB ile bellek ya da çevrebirimleri arasında alınıp verilmesinde kullanılan yol.

Yardımcı program (Utility) : Bir bilgisayarın çalışmasına yardımcı olmak üzere, kullanıcıya bilgisayarın işletim sistemi veya uygulama programıyla birlikte sunulan ve kullanıcının sık sık ihtiyaç duyacağı hizmetlerin karşılanması için parametrelerce yönetilen yardımcı hizmet programı.

Yarıiletken (Semiconductor): Mutlak sıcaklıkta akım geçirmeyen, ancak enerji (ışık, gerilim) uygulandığında, içerdiği valans elektronların yörüngelerinden koparak serbest hale gelmesiyle iletken duruma geçen malzeme, bu malzemeden yapılmış elektronik eleman.

Yarım-bayt (nibble): Bit gruplarının dörtlü gruplar halinde gösterim biçimi.

Yayınım (Propagation): Bir sinyalin devre ya da devreler dizisi içinden geçerek ilerlemesi.

Yazıcı (Printer): Gönderilen sayıların, kodların veya sembollerin basılı kopyasını çıkaran çıkış birimi. Basım tekniğine bağlı olarak satır, nokta-basışlı, mürekkep-püskürtmeli, tambur, vb. adlar alırlar.

Yazılım (Software): Bilgisayarın kaynaklarını kullanmak için gereken programlar, yordamlar, yardımcı programlar ve kütüphaneler gibi donanımın dışında kalan bölümü.

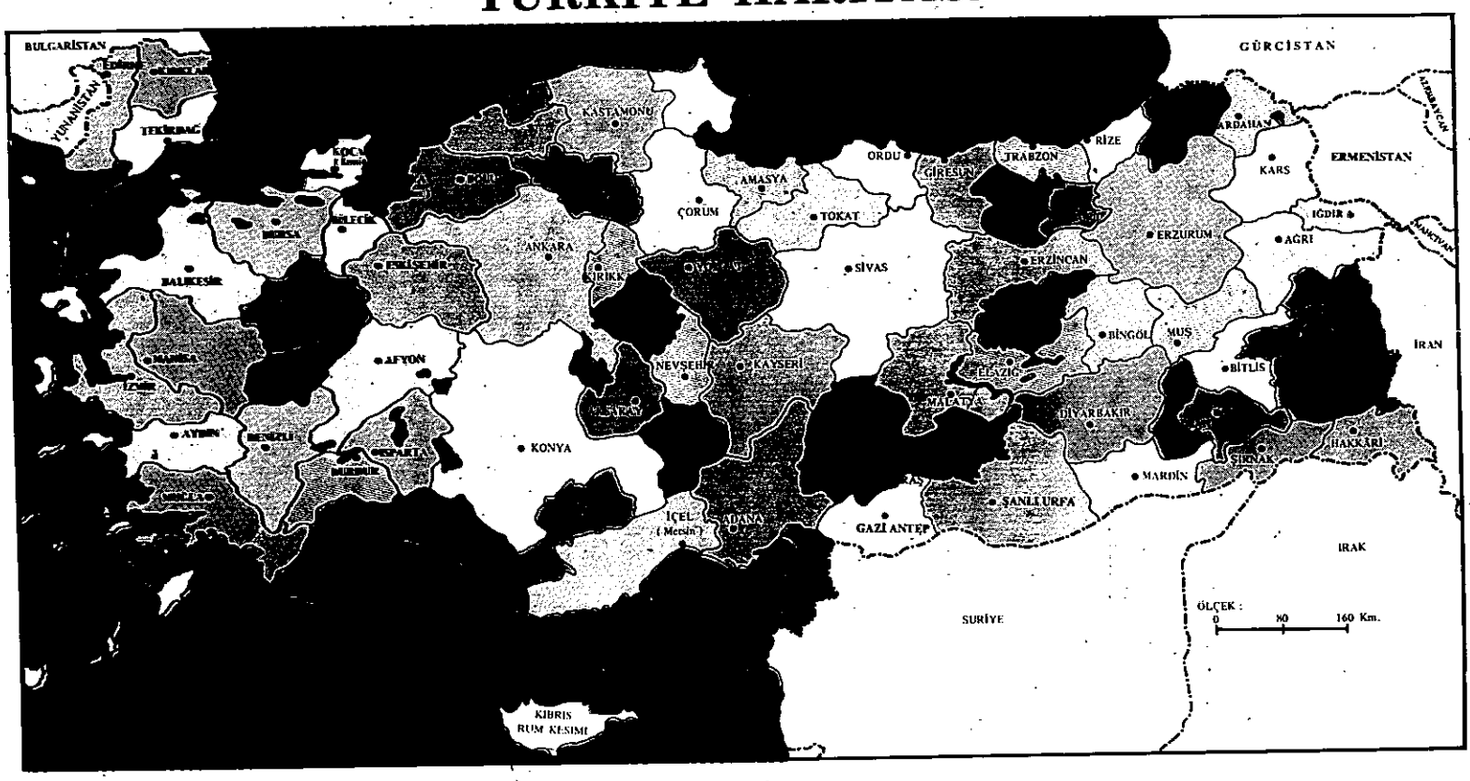
Yığın (Stack): Verileri geçici bir süreyle -örneğin, çıkışta geriye almak üzere bir altyordama girerken- saklandığı ve bilgiler, belirli bir sırada giriş-çıkış yapacak şekilde düzenlenmiş bellek alanı.

Yonga (Chip): Milimetrik boyutlarda yüzbinlerce elektronik devre elemanından oluşmuş yarıiletken devre elemanı.

Yordam (Routine) : Bir bilgisayar programı içinde yer alan ve işlevi, verilen değişkenlerle hep aynı işlemi gerçekleştirmek olan, ihtiyaç duyulduğunda ana program tarafından çağrılan program parçası.

Yedi-parçalı gösterge (Seven-segmented display): 0'dan 9'a kadar ondalık sayıları göstermek üzere çizgi biçimli yedi LED biriminden oluşan sayısal gösterge.

TÜRKİYE HARİTASI



ÖĞRETMEN MARŞI

Alnımızda bilgilerden bir çelenk,
Nura doğru can atan Türk genciyiz.
Yer yüzünde yoktur, olmaz Türk'e denk;
Korku bilmez soyumuz.

Şanlı yurdum, her bucağın şanla dolsun;
Yurdum, seni yüceltmeye antlar olsun.

Candan açtık cehle karşı bir savaş,
Ey bu yolda and içen genç arkadaş!
Öğren, öğret hakkı halka, gürle coş;
Durma durma koş.

Şanlı yurdum, her bucağın şanla dolsun;
Yurdum, seni yüceltmeye antlar olsun.

İsmail Hikmet ERTAYLAN