Satış fiyatı .....:
KDV .....:
KDV'li SATIŞ FİYATI .....: 130000- Lira

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

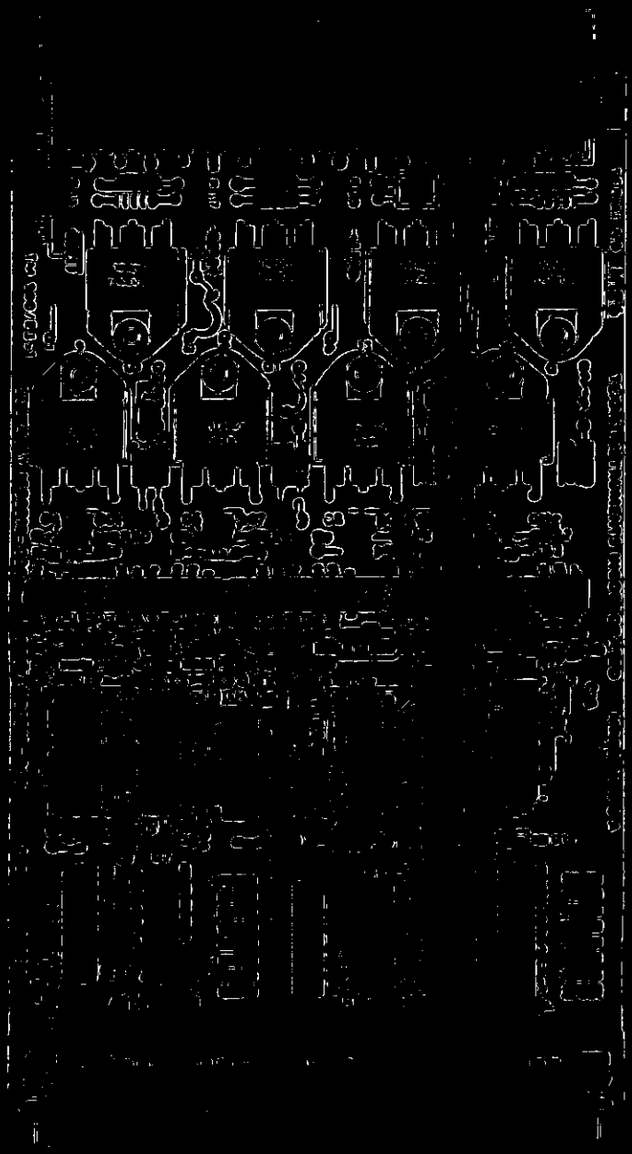
PERAKENDE SATIŞ
Millî Eğitim Yayınevleri ve Bakanlık yayınları satıcısı kitapçılar.

Endüstriyel Okullar İçin **Mikroelektronik Sistemler - Kitap III**

ENDÜSTRİYEL OKULLAR İÇİN

MİKROELEKTRONİK SİSTEMLER

KİTAP III



ENDÜSTRİYEL OKULLAR İÇİN

Mikroelektronik Sistemler

Kitap III

Dr. D.J. Woollons

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

Millî Eğitim Bakanlığı Yayınları ..... : 2694
Yardımcı ve Kaynak Kitaplar Dizisi ..... : 55

ISBN 975 - 11 - 0868 - 3

Hükûmetimiz ile Dünya Bankası arasında imzalanan Endüstriyel Okullar Projesi çerçevesinde hazırlanan "Mikroelektronik Sistemler Kitap III" Millî Eğitim Bakanlığı, Talim ve Terbiye Kurulu Başkanlığının 08/04/1994 gün ve 2995 sayılı kararı ile kaynak kitap olarak uygun bulunmuş ve 10.000 adet bastırılmıştır.

Çeviri - Dizgi - Mizampaj : Üniversal Dil Hizmetleri ve Yayıncılık A.Ş.
Çevirmen : Elekt. Müh. Erhan GÜNGÖR
Editör : Yrd. Doç. Ahmet Coşkun SÖNMEZ
Baskı Hazırlık - Baskı - Cilt : Evren Ofset Basım Sanayii ve Ticaret A.Ş.

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, Hak'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.

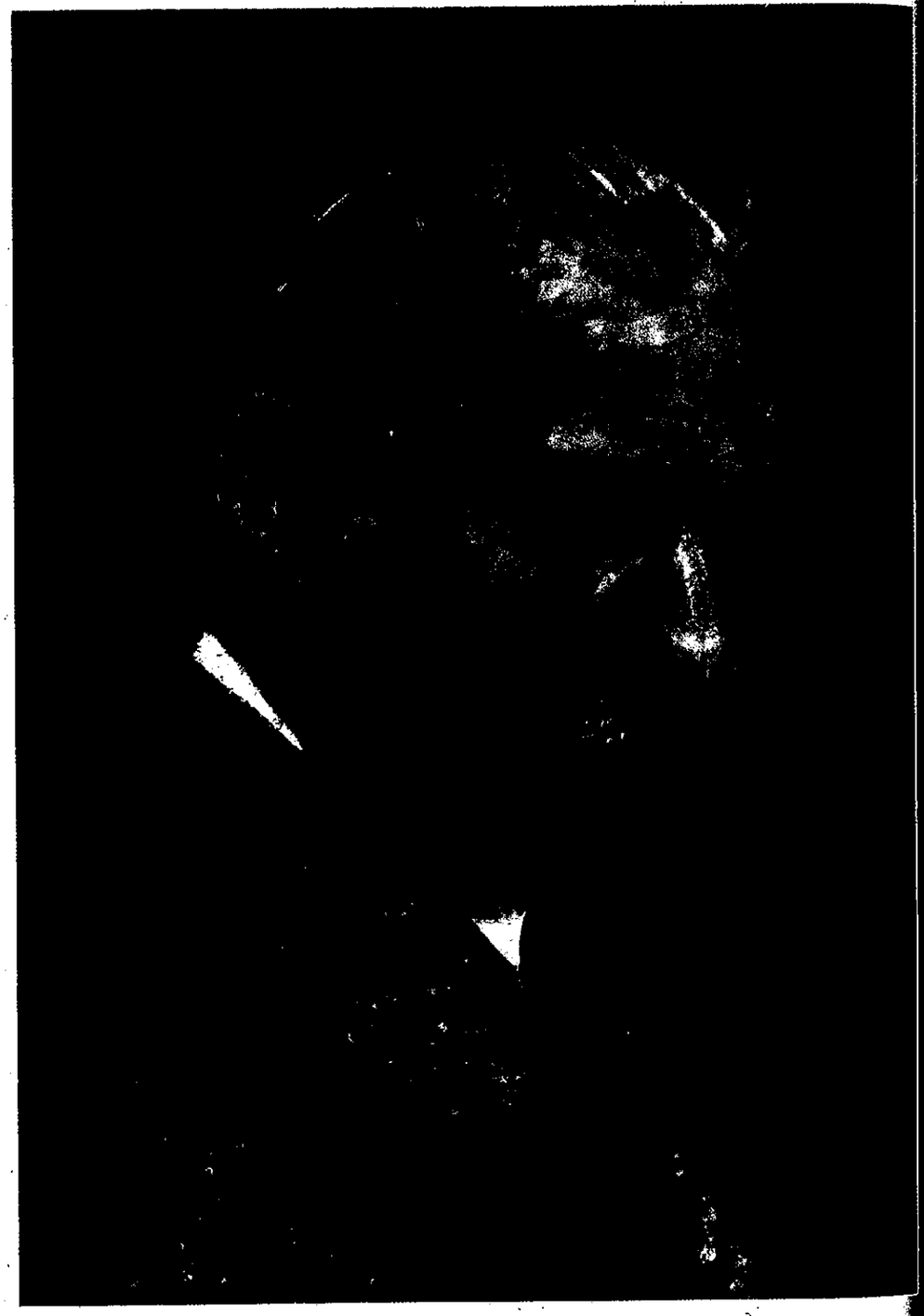
Kim bu cennet vatanın uğruna olmaz ki fedâ?
Şühedâ fışkıracak toprağı sıksan, şühedâ!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüdâ.

Ruhumun senden, İlâhî, 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, İlâhî, 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, bayrağının hürriyet;
Hakkıdır, Hak'a tapan, milletimin istiklâl!

Mehmet Âkif ERSOY

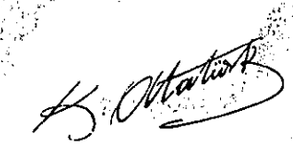


ATATÜRK'ÜN
GENÇLİĞE HİTABESİ

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!



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 Önsözü

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.

Eşim Jan'e,

Kitabın yazılmasındaki önemli çabaları ve bu kitabın hazırlanması esnasındaki teşvik ve daimi desteği nedeniyle teşekkür ederim.

DAVID WOOLLONS

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

Bu kitap birden fazla amaca yönelik olarak hazırlanmıştır. İlk olarak, mikroişlemci büyük ölçekli entegre devrelerinin donanım özelliklerini, bunların kullanım alanlarını açıklamak ve mikroişlemci ile birlikte kullanılan yan devreleri tanıtmak amaçlanmıştır. Bu nedenle, 2., 3. ve 4. Bölümlerde mikroişlemcinin kendisi tanımlanmış; bellek devreleri, giriş/çıkış ve diğer çevresel entegrelerle bağlantıların nasıl yapıldığı ve nihayet bütün bir mikrobilgisayar sisteminin nasıl oluşturulduğu anlatılmıştır.

Ardından, mikrobilgisayarlarda kullanılan birçok önemli kavram ve özellik açıklanmaktadır. Özellikle 5, 6, 7 ve 8. Bölümlerde alt yordamlar, yığınlar ve kesme sistemleri işlenmiştir. Son olarak 9. Bölümde, sırayla tarama ve mikrobilgisayar ile çevrebirimleri arasındaki seri veri aktarımı gibi arabirim teknikleri; 10. Bölümde ise, mikrobilgisayarlara uygun yazılım geliştirme özellikleri ele alınmıştır. (Kitap BTEC Birim Mikroelektronik Sistemleri U79/604'e uygun olarak yazılmıştır).

Mümkün olduğunca, bütün konular gerçek mikrobilgisayar sistemlerinden alınan örneklerle desteklenmiştir. Mevcut mikroelektronik devreler çoğaldıkça, doğal olarak bunların ayrıntıları da değişecektir. Bununla birlikte, sistem ve onu oluşturan yan birimlerin ana ilkelerinde bir değişiklik olmaz.

Kitap, belli tipteki bir mikroelektronik sistemin tasarlanmasında kullanılmak üzere hazırlanmamıştır. Örnekler, çoğunlukla Intel 8080, 8085 serisinin özelliklerine göre düzenlenmiştir; ancak bu, öğrencilerin kendilerini yalnızca bu sistemle sınırlandırması gerektiği anlamına gelmez. Bu kitabı bitirdikten sonra, genel kavramları diğer sistemlerle de bağdaştırabilecek ve aralarındaki ilişkiyi görebilecek durumda olacaklardır.

10. Bölümde ana çizgileriyle geliştirilen pratik tasarım, ilk bölümlerde anlatılan teorik konuları bir bütün içinde bir araya getirmektedir. Bu yüzden, okuyucunun ana amacı, bir bakıma kitabın hedefi olan bu bölümü çok iyi anlamak olmalıdır.

Bütünüyle bakıldığında, bu kitap okuyucuya mikrobilgisayar sistemlerinin değişkenliği ve pratikteki problemlere uygulanış kolaylığı hakkında iyi bir fikir verir.

Mikroelektronik Sistemler
Kitap III

Bölüm 1 Mikroişlemci-tabanlı sistemler

Bu Bölümün Amaçları: *Bu bölümü bitirdiğinizde:*

1 *Mikroişlemci ile mikrobilgisayar arasındaki farkı ayırt edebilmeli,*
2 *Çalışan bir mikrobilgisayar sisteminde yazılım ve donanıma niçin gerek duyulduğunu anlayabilmeli,*
3 *Bir mikrobilgisayar sisteminin,*
   (a) bir CPU yongası,
   (b) bir saat osilatörü,
   (c) sadece-okunur bellek ve
   (d) rastgele-erişimli bellekten meydana geldiğini,
   bunların birbirlerine veri, adres ve denetim anayolları ile bağlı bulunduğunu ve sistemin çalışması için bir güç kaynağı gerektiğini anlayabilmeli,
4 *MİB (Merkezi İşlem Birimi) ve buna bağlı çevresel birimler arasında hız farklılıkları olduğunu anlayabilmeli,*
5 *Mikrobilgisayarlarla çevrebirimlerinin birbirleriyle eşzamanlanması gerektiğini; bu eşzamanlama sırasında sinyal alışverişinin nasıl yapıldığını ve çevresel arabirim devrelerinin nasıl kullanıldığını değerlendirebilmeli,*
6 *İki çevresel iletişim tekniği olan*
   (a) yazılımsal tarama ve
   (b) kesme sistemlerinin, ana ilkelerini anlayabilmeli,
7 *Gerçek-zamanlı bir uygulamada, zaman gecikmesi üretmek için komut yürütüm süresinin kullanımına ilişkin ilkeleri anlayabilmeli,*
8 *(a) işlemci kaydedici takımı ve*
   (b) komut takımını da içerecek biçimde bir mikrobilgisayarın Merkezi İşlem Biriminin özelliklerini öğrenmiş olmalısınız.

1.1. Giriş

Mikroişlemci ya da daha doğru bir deyişle mikrobilgisayar, Cambridge'de Matematik Profesörü Charles Babbage'ın 19. yy.'ın başlarında başlattığı gelişmeler zincirinin en son halkasıdır.

Tüm dünya yüzeyindeki suların kabarma zamanlarına ilişkin tablolar örneğinin de gösterdiği gibi, büyük miktarlarda sayısal veriler içeren işlemlerle ilgilenen Bab-

bage, yaşamının büyük bir bölümünü bu tür işlemlerin, bir makine kullanarak otomatik ve hızlı bir biçimde nasıl gerçekleştirilebileceğini düşünerek geçirdi.

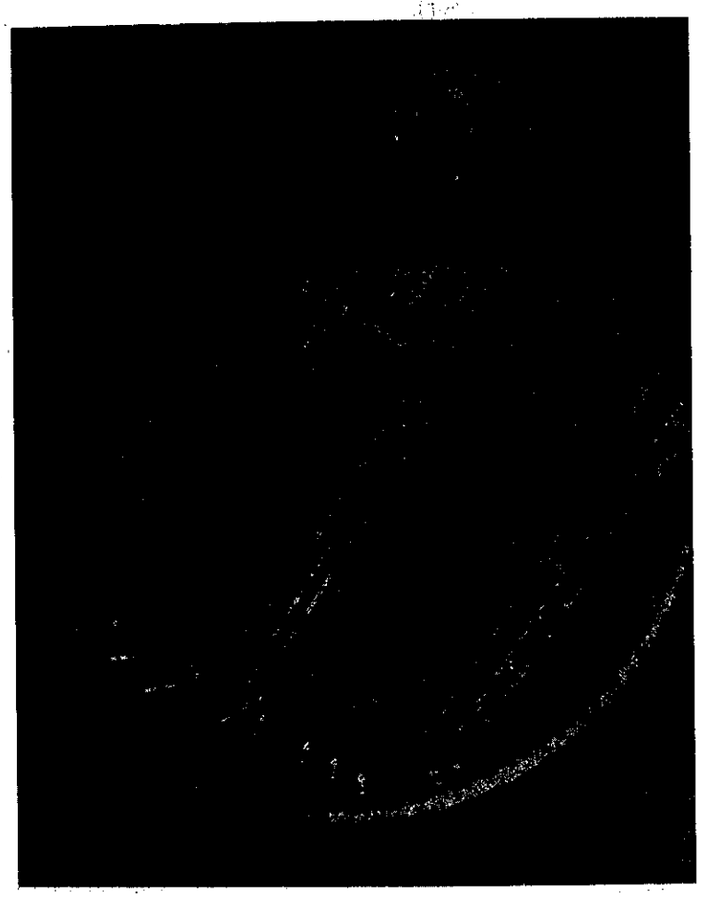
1833 yılında başladığı "Analitik Makine" tasarımında, Babbage modern bilgisayar sistemlerinin temeli olan birkaç öneri getirdi:

1 Makinenin yaptığı işlem bir *saklı program*, yani *bellekte* saklanmış bir komut listesi tarafından denetlenmeliydi. Her bir komut makineye; toplama, çıkarma veya bir sayının bir yerden diğer bir yere taşınması gibi bazı basit işlemleri yaptıracaktı. İşlemler, birbiri ardı sıra, otomatik ve (insanoğlunun kâğıt-kalem ile yapabileceğinden daha) yüksek bir hızda yürütülecek ve bu yolla karmaşık hesaplamaların yapılabilmesi mümkün olacaktı.
2 İkinci önerisi ise, makinenin yürüttüğü komutların bazı *koşullu* test komutlarını da içermesi gerektiğiydi. Bu komutlar, bilgisayarın program boyunca izleyeceği yolu belirleyen özel komutlar olacaktı. Bunlar, programcının bir hesaplamanın sonucunu incelemesi ve programın buna göre *dallanması*, yani incelemenin sonucuna bağlı olarak programın belli bölümlerine *atlaması* için kullanılabilirdi. Böylece program, bu tür bir test aracılığıyla, hangi aşamada hangi komutun işletileceğini kendisi belirleyecekti. Bu, daha sonra da göreceğiniz gibi çok önemli bir özelliktir.

Babbage, analitik makinenin teorik tasarımında çok önemli gelişmeler gerçekleştirmiş olmasına rağmen, pratikte bu tür bir makineyi oluşturmak için gerekli hassas mekanik parçaların bulunmaması nedeniyle tam olarak çalışan bir sistem oluşturamamıştır. Büyük ölçekli bilgisayar sistemlerinin üretimi, 2. Dünya Savaşı sırasında elektronik alanındaki gelişmeleri beklemek zorunda kaldı ve bu dönemi izleyen yıllarda bu tür sistemler ortaya çıkmaya başladı. İlk bilgisayarlar, lambalarla üretildi, ancak bunlar çok büyük miktarlarda güç harcıyordu.

Bu lambalı bilgisayarların güvenilirliği çok düşük düzeydeydi. 1950'li yılların sonlarında transistörün kullanıma girmesi, bilgisayar sistem tasarımlarına hız verdi. Transistör gibi aktif elemanların kullanımı sonucu güvenilirliğin artmasıyla birlikte, güç tüketimi ve fiziksel boyutlar da küçüldü. Böylece, bilgisayarlar ucuzladı, daha uygun ve daha geniş bir alanda kullanılabilir bir duruma geldiler.

Boyutların küçültülmesi eğilimi, 1960'lı yıllarda entegre devrelerin ortaya çıkışıyla ve 1970'lerde orta ve büyük kapasiteli entegreleştirmenin gelişimiyle devam etti. Entegreleştirmedeki gelişim, çok karmaşık mantık fonksiyonlarının küçük bir entegreye sığdırılmasına olanak sağladı. Bu kitabın yazıldığı günlerde artık, bir tek entegreye 29.000 tanesi transistör olmak üzere 260.000 bileşen yerleştirilebilmektedir.



Şekil 1.1 LSI *(Geniş ölçekli entegre)* yonga yaklaşık 5 mm genişliğinde

Ön sözdeki sorulara ilişkin nota bakınız **S 1.3**

Bunu mümkün kılan teknikler; P-MOS, N-MOS veya C-MOS teknikleri kullanılarak imal edilen, alan-etkili transistör temeline dayanır. N-MOS (N-kanal metal oksit yarı iletken), P-MOS'dan iki-üç kat daha yüksek anahtarlama hızına sahiptir ve bu yüzden kullanımı daha yaygındır. Çok az güç harcamasına karşın C-MOS, N-MOS'dan daha yavaş çalışır. Bununla birlikte, bazı uygulamalar için daha uygundur. MOS sistemler genelde, iki-kutup tekniğiyle düzenlenmiş sistemlere göre daha ileri üretim yöntemleriyle üretildiklerinden avantajlıdırlar. Bunun ötesinde, üretim hızının yüksek olmasından dolayı da "iyi" bir verim elde edilmektedir.

Mikroişlemci ve mikrobilgisayarlar

Bir önceki bölümde ana hatlarıyla gösterildiği gibi, bilgisayar sistemlerine güç ve esneklik veren iki temel özellik vardır:

1 Bellekte saklı bir program tarafından denetlenirler.
2 Mevcut program komut seti, yani *komut kodu* veya bilgisayarın *komut takımı*, koşullu dallanma komutlarını içerir.

Bu nedenle, en küçük bir bilgisayar sistemi bile bir işlemci ve bir bellek içermelidir. İşlemci, cihazın aritmetik ve mantıksal işlevlerini gerçekleştirmek ve sistemin genelinde denetimi sağlamak için gereklidir. Bellek ise, üzerinde işlem yapılacak verileri ve program komutlarını saklamak için gerekir.

Normal olarak mikrobilgisayarlar, kendilerine bağlanan dış birimlerle iletişime de gerek duyar. Bu *çevresel birimler*, mikrobilgisayar sistemine bilgi sağlar ve çıkışındaki veriler üzerinde işlem yapmak için kullanılır. Her biri özel bir uygulamaya yönelik pek çok çevre birimi vardır.

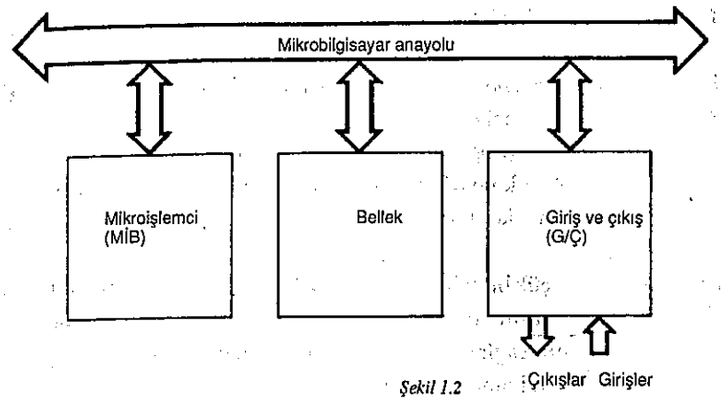
Bilgisayardaki sonuçları kağıda aktarabilmek için *satır yazıcı* çevrebirimleri kullanılır. Bilgisayara sayısal bilgiler girebilmek için de klavye kullanılır. Televizyon tipi bir katot ışınlı-tüplü ekran ile birlikte bir *klavye*, bilgisayara (klavye ile) veri girişi ve (görüntüleme ekranı ile de) bilgisayardan çıkış alabilmeye uygun bir *terminal* durumuna gelir.

Bilgisayarın daha büyük bir sistemin parçası olduğu uygulamalarda ise başka türde çevresel birimlere ihtiyaç duyulabilir. Örneğin, bir mikrobilgisayar, bir çamaşır makinesinin işlevlerini denetlemek amacıyla kullanılabilir. Bunu yapabilmek için, su sıcaklığı, kazandaki su düzeyi, yıkama süresinin ne olacağı vb. parametrelere ilişkin bilgilere gerek vardır. Bu durumda mikrobilgisayarların çevrebirimleri; termometreler, su düzeyi algılayıcıları, zamanlayıcılar ve benzeri aygıtlar olmaktadır.

Çevresel birimler bilgisayara, giriş-çıkış bilgilerini kullanan özel devreler aracılığıyla bağlanır. Bu devrelere ilerideki bölümlerde ayrıntılı olarak değinilecektir.

İşlemci, bellek ve giriş-çıkış devreleri bir araya geldiklerinde, Şekil 1.2'deki gibi tipik bir mikrobilgisayar sistemi ortaya çıkar. Bu üç ana birim, *sistem yolu* veya *sistem anayolu* adı verilen bir grup hat aracılığıyla birbirlerine bağlanır.

Burada, *mikrobilgisayar* ile *mikroişlemci* arasındaki farkları ele almak yerinde



Şekil 1.2

olacaktır. Sıkça kullanılan tanımlardan biri olan mikroişlemci, aritmetik işlevlerden ve sistem denetiminden sorumlu olan birimdir. Mikrobilgisayar ise, bellek ve giriş-çıkış (G/Ç) devrelerini de içeren tüm sistemdir.

Pratikte her ikisi de, birleşik halde tek-yongalı entegre bir devre olarak satın alınabilir. Bununla birlikte, bir mikrobilgisayar oluşturabilmek için, tek-yonga'dan ibaret mikroişlemcilerin, diğer büyük ölçekli entegre devreler (bellek ve giriş-çıkış devreleri) ile birlikte kullanılmaları gerekebilir.

Mikrobilgisayarların uygulama alanları

Mikrobilgisayarların çok ve çeşitli kullanım alanları vardır. En önemli özellikleri, saklı-program denetiminden gelen esneklikleri ve maliyetlerinin düşük olmasıdır.

Mikrobilgisayarın yürüttüğü işlevler, belleğindeki program tarafından yönetildiğinden, yalnızca programı değiştirerek mikrobilgisayarı yönlendirmek oldukça kolaydır. Dolayısıyla, evlerimizdeki bilgisayarlarımızı, aile bütçesini hesaplamada kullanabileceğimiz gibi, belleğine "Uzay Savaşçıları" gibi oyun programları yükleyerek oyun da oynayabiliriz.

Birbirinden oldukça farklı bu iki uygulamada, birinden diğerine geçerken *donanımı*, yani mikrobilgisayarın devresini değiştirmeye gerek yoktur. Sadece *yazılımı*, yani bellekteki programı değiştirmek yeterlidir. Böylece bilgisayara, tümüyle farklı bir işlev gördürmüş oluruz.

Mikrobilgisayarların maliyetinin düşük olmasında birkaç faktörün rolü vardır. Her

şeyden önce, imalatında kullanılan malzemeler, bildiğimiz deniz kumunun ana maddesi silikon olduğu için ucuz ve boldur. Tek bir üretim hattında bir defada milyonlarca yonga üretilebilir. Bundan başka, entegre devre imal eden firmalar arasındaki rekabet de fiyatların düşmesinde önemli bir rol oynar.

Esnek ve ucuz olmaları, mikrobilgisayarların, çeşitli ürünlerin içinde kullanılan amaca-özel devrelerin yerine kullanılabilmelerini mümkün kılmaktadır. Mikrobilgisayarların bu tür devrelerde kullanılması, ürünlerin işlevsel yeteneklerini büyük ölçüde artırmakta ve ürünlerin içindeki -örneğin transistör-transistör mantık (TTL) gibi- geleneksel entegre devrelerden mevcut on ya da yirmisinin yerine mikrobilgisayar kullanılması bu ürünleri daha ekonomik hale getirmektedir.

Geleneksel mantık-tasarım yöntemlerinin kullanılması, her uygulama için özel geliştirilmiş mantık sistemleri gerektirir. Bu sorun, mikrobilgisayarlar kullanıldığında donanım değiştirmeksizin, özel geliştirilmiş program tarafından denetlenen standart donanımla çözülebilir. Bu yüzden tasarımda ağırlık, donanımdan (yani devreler, yollar, arabağlantı hatları vb.), yazılıma (program komutlarına) kayar.

Bu özellikler, mikroişlemci-tabanlı sistemlere önceki rastgele-mantık eşdeğerlerine göre birkaç avantaj sağlar:

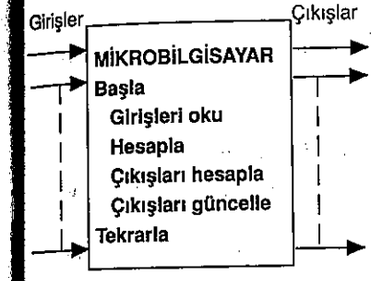
1 Yeni ürün geliştirmede gereken zaman ve masraflar büyük ölçüde azalmıştır.
2 Mikrobilgisayarların esnekliği, ürün üzerinde piyasanın gereksinmelerine uyacak tarzda değişiklik yapılabilmesine olanak tanımaktadır.
3 İşlevsel yeteneklerin büyük ölçüde artırılmış olması, daha iyi ve daha gelişmiş sistemler oluşturulabilmesine olanak tanımıştır. Örneğin, mikrobilgisayar denetimli ankesörlü telefon cihazları konuşma süresini ölçer ve artan jetonları iade eder.
4 Mikrobilgisayarlar, sistem içinde kullanılan bağlantı kablosu sayısını büyük ölçüde azaltmış ve böylelikle güvenilirlik de artmıştır.

Mikrobilgisayarların bazı kullanım alanları şunlardır:

1 *Aletlendirme* Karmaşık cihazlı sistemlerin denetim ve kalibrasyonu.
2 Bir fabrikanın işlemlerini düzene sokan *endüstriyel denetim sistemleri;* örneğin, bir kimyasal üretim. Bu tür sistemlerde, işlemler belli bir zaman sırasına göre yürütülmeli ve mekanik parçaların konumları çok iyi denetlenmelidir. Örneğin, bir çelik fabrikasındaki merdane konumlarının doğru biçimde belirlenmesi gerekir. Yukarıda kısaca açıklanan uygun çevrebirimiyle birlikte mikrobilgisayarlar bunu gerçekleştirmek için kullanılabilir.

3 *Bilgisayar sistemleri* Mikrobilgisayar, genel amaçlı bilgisayar tasarımında kullanılır. Bu açıdan, çoklu mikroişlemciler önemli yer tutar.
4 *Tüketici uygulamaları* Ev gereçlerinin, araçların ve elektronik olarak programlanabilir oyunların denetimi.
5 *Büro donanımı* Kelime-işlemci sistemleri.
6 *Askeri sistemler.*
7 *Tıbbi* teşhis ve izleme cihazları.
8 Bilgisayar terminalleri olarak kullanılmak amacıyla *akıllı sistemler.*
9 *Eğitim araçları.*
10 *İletişim sistemleri* İletişim devrelerindeki hataların algılanması, düzeltilmesi ve telefon anahtarlama şebekelerinin denetiminde kullanılır.

S 1.1



Şekil 1.3

1.2 Yazılım ve donanımın gerekliliği

Daha önceki bölümlerde de belirtildiği gibi, bir mikrobilgisayar sistemi, birbirleriyle iletişimde olan yazılım ve donanım modüllerinden oluşmaktadır. Böyle bir sistemin çalışması, Şekil 1.3'de görüldüğü gibi sembolik olarak gösterilebilir.

1 Mikrobilgisayarlar, dış dünyadan gelen bilgileri bir dizi giriş hattı üzerinden alır. Bu bilgiler, klavye ile girilen bir sayı ya da çamaşır makinesindeki suyun sıcaklık değeri vb. olabilir.
2 Çıkış hatlarına verilecek yeni değerleri hesaplar. Bu hesaplama, programın ana bölümünü oluşturur.
3 Hesaplanan bu yeni değerler çıkışa verilir. Bunlar, TV ekranında görüntülenecek ya da kağıda basılacak harfler, rakamlar ya da çamaşır makinesinin besleme gücünü artıracak denetim sinyalleri olabilir.

Üç aşamadan oluşan bu işlem, mikrobilgisayar programının çalışmasını durduracak bir komut (örneğin bir HALT komutu) ile karşılayıncaya kadar ya da dışarıdan bir etkiyle kesilinceye kadar sürekli tekrar edilir.

Daha önce de belirtildiği gibi, makineyi denetleyen program

bellekte tutulan bir basit komut dizisinden ibarettir. İşlemci bu komutlara, Şekil 1.4'de görüldüğü gibi bellekteki *adresleri* aracılığıyla başvurur.

Bellekler, her birinde program komutu ya da veri parçası gibi bilgileri tutmak için kullanılan ardışık konumlardan oluşmuştur.

Bellekte saklı bulunan tüm bilgiler, veri ya da program komutu olup olmadığına bakılmaksızın, ikili biçimde tutulurlar. Sayılar, büyüklük bitleri ve bir işaret biti ile komutlar ise kodlanmış ikili desenlerle gösterilir. Bu nedenle bellek büyüklükleri genelde, toplam konum sayısıyla birlikte her bir konumun tutabildiği bit sayısı üzerinden ifade edilir.

Mikrobilgisayar sistemlerinde, her bir bellek konumu normalde 8 bit, yani bir *bayt* uzunlukta veri veya komut ya da komut parçası tutar. Ancak, daha sonra da göreceğimiz gibi, bazı komutları belirtebilmek için 8 bitten daha fazlasına, 2 hatta 3 bayta gerek duyulur.

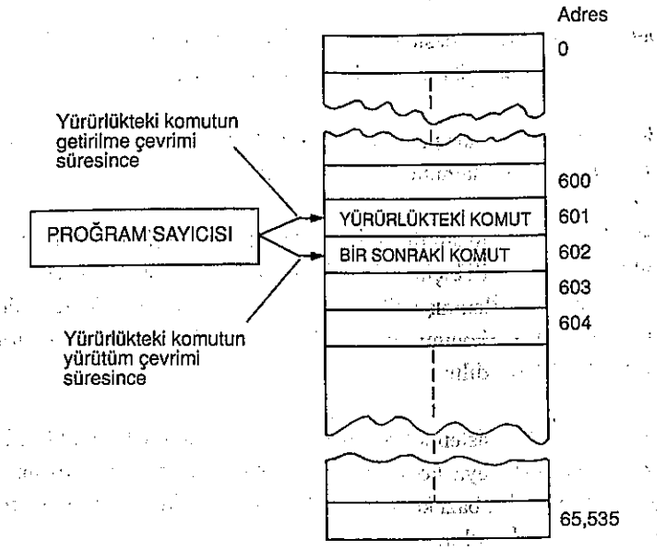
Bellekteki konum sayısı, yani toplam bellek büyüklüğü, bellek adreslerini belirtmek için kullanılan ikili digitlerin (*bitlerin*) sayısıyla ifade edilir. İşlemcinin, yalnızca ilgili konuma ulaşabilmesi için, her konumun adresi diğerlerinden farklıdır. Dolayısıyla, bir adresi ifade etmek için 16 bit'in kullanıldığı bir bellekte, 0 ile 65.535 arasında toplam 65.536 adet adres vardır. Böyle bir sistemde, (eğer her bir adres 8 bit'lik ise) *adres aralığı* 65.536 bayttır.

Programın bir parçası olarak bellekte saklı komutlar, genelde iki kısım olarak düşünülebilir:

1 *İşlem kodu* (veya *işkodu*)
2 *Adres*

Komutun işlem kodu kısmı, yürütülecek işlemin ne olduğunu belirtir (toplama, çıkarma, veri iletimi, vb.). Adres kısmı ise kullanılacak *işlenen*'in belleğin neresinde bulunduğunu gösterir.

Bilgisayar, bellekten bir komutu getirir, kodunu çözer ve işlem koduna göre bu komutu yürütür. Bu işlemi sürekli yineler. Eğer bir dallanma komutuna rastlanmazsa işlemci, program üzerinde, komutları art arda yürüterek ilerler. İşlemci, bunu yapmak için *program sayıcı* adı verilen bir kaydedici (içinde bilgi tutulabilen alan) kullanır.



Şekil 1.4

Program sayıcısı, programın yürütülmesinde işlemcinin hangi adıma geldiğini izlemek için kullanılır. Program sayıcısı, bellekten çağırılacak ve yürütülecek bir sonraki komutun adresini tutar.

Yürürlükteki komutun yürütülmesi tamamlandığında işlemci:

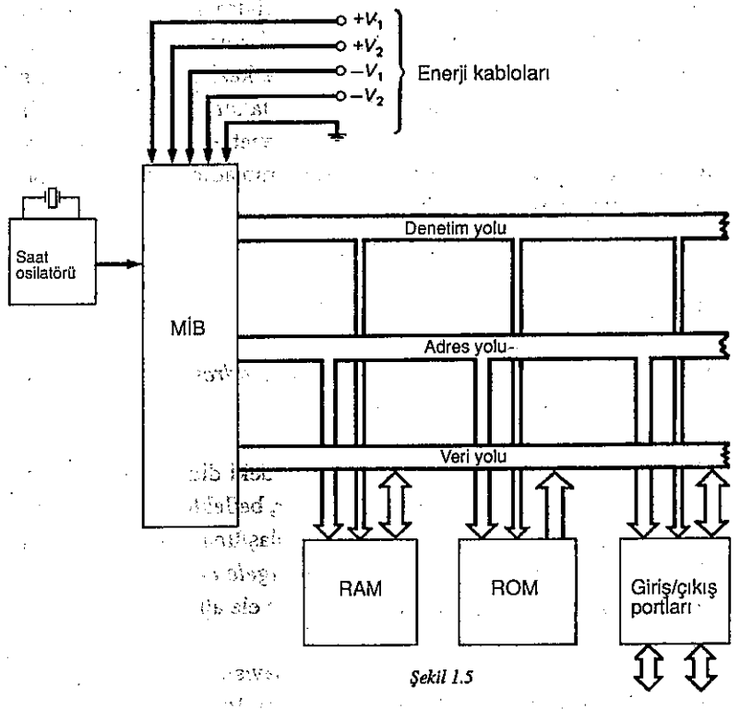
1 Program sayıcısında tutulan değeri bir adres olarak belleğe gönderir.
2 Bu adreste bulunan komutu bellekten işlemci yongasına getirir.
3 Komutun işlem kodunu inceler ve belirtilen işlemi gerçekleştirir.

Yukarıdaki 1. ve 2. aşamalar, işlemcinin *komut getirme saykılı*; 3. aşama ise *yürütme saykılıdır*.

Komut getiriminin gerçekleştirilmesinin hemen ardından, program sayıcısının içeriği bir artırılır. Böylece program sayıcısının içeriğinde artık bellekten getirilecek *bir sonraki* komutun adresi bulunmaktadır.

S 1.6

Dolayısıyla program sayıcısı, getirilecek bir sonraki komutun adresini 'gösteren' değerin tutulduğu bir birim olarak düşünülebilir. Bu işlemler Şekil 1.4'de şematik olarak gösterilmiştir.



Şekil 1.5

O halde, bir mikrobilgisayar sistemi donanımının program komutlarını yürütmek için tasarlandığı söylenebilir. Bu yüzden herhangi bir işlevin yerine getirilmesinden önce, bir takım komutların makinede mevcut bulunması gereklidir; bir uygulama programından okuma yapmak ve okunan değeri belleğe yerleştirmek için bile, bellekte işlemcinin program komutlarını ilgili çevrebirimlerinden almasını sağlayacak bir "yükleyici" programının bulunuyor olması gerekir. Bu nedenle sistem, yerleşik bir *monitör (izleme)* programı; yani program veya veri giriş-çıkışı, uygulama programının çalıştırılması, durdurulması ve benzeri temel makine işlevlerinin denetimine izin veren ve sürekli bellekte saklı bulunan bir program içerir.

S 1.2

1.3 Komple bir mikrobilgisayar sistemi

Bir mikrobilgisayar sisteminin birkaç modülden oluştuğunu daha önce görmüştük. Şimdi bu kavramı biraz daha genişletecek ve sistemi oluşturan bileşenleri daha ayrıntılı bir biçimde inceleyeceğiz. Şekil 1.5'de genel bir mikrobilgisayar sisteminin daha ayrıntılı bir çizimi verilmiştir.

Bölüm:1.1'de de izah edildiği gibi, mikrobilgisayarlar birbirlerine sistem anayolları veya yolları ile bağlanan büyük ölçekli entegre devre oluşur. Mikroişlemci, (MPU Mikroişlemci Birimi veya CPU-Merkezi İşlem Birimi) sistemin merkez entegresidir. Bu birim, makinenin komut takımını uygulamak, diğer birimlerin zamanlamasını ve eşzamanlı çalışmasını denetlemek, komut çağırma ve yürütme saykıllarını düzenlemek, komut işlenenlerinin adreslerini belirlemek, vb. amaçlar için kullanılan devreler içerir.

Sistem yolu

Sistem yolu, Şekil 1.5'de görüldüğü gibi, *adres, veri* ve *denetim yollarından* meydana gelen bir bağlantılar grubudur.

Adres yolu, işlemciden veya bazen sistemdeki diğer entegrelerden belleğe adres göndermek için kullanılır. Bu gibi adresler, bellekteki komutları ve verileri almak ya da belleğe bilgi kaydetmek amacıyla ulaşılması gereken yerleri belirtir. Şekil 1.5'de iki tip bellek gösterilmektedir: *rastgele erişimli bellek* ve *sadece okunur bellek*. Bunlar arasındaki farklar daha sonra ele alınacaktır.

S 1.5

Adres yolu için gerekli bağlantı hattının sayısı, mikrobilgisayarın adres alanının büyüklüğü ile belirlenir. Örneğin, sistemde 16 adres hattı varsa, belleğe 16 bit uzunlukta bir adres gönderilebilir ve dolayısıyla 65.536 (64K; 1K=1024) adreslenebilir konuma erişmek mümkün olur. Bu adres aralığı, Intel 8085, Zilog Z80 ve Motorola MC 6800 gibi 2. ve 3. kuşak (Mikrobilgisayar kuşak tanımları için Bölüm 2.2'ye bakınız) mikroişlemcilerinin tümünde ortaktır. Intel 8086, Zilog Z8000 ve Motorola 68000 gibi 4. kuşak makineler ise daha büyük adres alanlarına sahiptirler.

Dikkat edilmesi gereken önemli bir nokta, adreslerin her zaman mikrobilgisayarlardan belleğe gönderiliyor olmasıdır. Bu nedenle, Şekil 1.5'deki RAM ve ROM ile adres yolu arasındaki bağlantılar *tek yönlüdür.*

Veri yolu ise, bilgilerin belli bir bellek konuma aktarılmasında ya da çevresel giriş/çıkış işlemlerinin sistem düzeyinde dağıtılmasında kullanılır. Bu yolun *genişliği,* diğer bir deyişle, bağlantı hatları sayısı, kullanılan veri biriminin büyüklüğüne bağlıdır. 2. ve 3. kuşak mikrobilgisayarlarda genellikle 1 baytlık veri birimi kullanılır, dolayısıyla veri yolu 8 hattan oluşur. Daha gelişmiş olan 4. kuşak sistemler ise 16 bitlik paralel veri kullanır, dolayısıyla bunlar 2 bayt genişlikte bir veri yoluna sahiptirler.

Veri yolu *çift yönlüdür*. Bu yüzden, bellekten veya çevre birimlerinden işlemciye olduğu gibi, işlemciden de bellek ya da çevre birimlerine veri aktarımı yapmak mümkün olur. Dahası, veri aktarımı yalnızca işlemci üzerinden yapılmaz. Bazen işlemcinin müdahalesi olmaksızın, çevre birimleri ile bellek arasında doğrudan bilgi alışverişi yapılabilir.

Giriş ve çıkış için ayrı ayrı bağlantılar tahsis etmeye gerek olmadığı için, veri yolunun iki yönlü olarak kullanımı, işlemcinin tümleşik devrelerindeki bacak sayısını azaltır. Bu da maliyeti düşürdüğü ve pratikte kolaylık getirdiği için çok aranılan bir özelliktir.

Denetim yolu, işlemci ve diğer birimlerce üretilen sinyalleri taşıyan bağlantı hatlarından oluşmuştur. Bu sinyaller, farklı modüllerin çalışmalarının sistem tarafından senkronize edilmesi (eşzamanlanması) için kullanılır. İlerki bölümlerde bu konu daha ayrıntılı olarak incelenecektir.

Saat osilatörü

Bir mikrobilgisayar sisteminde gerçekleşen tüm işlemler mutlaka doğru biçimde eşzamanlanmalıdır. Örneğin, bellekten bir veri alınırken yapılacak işlemler, Kısım 3.1'de verildiği gibi, olması gereken sırada gerçekleştirilmelidir; bunu sağlamak üzere mikrobilgisayarlar, bir *saat osilatörü* veya bazen adlandırıldığı biçimiyle bir *saat üreteci* içerirler (Şekil 1.5).

Saat osilatörü, sistemin çeşitli birimlerinin çalışmalarını eşzamanlamak için gerekli olan zamanlama sinyallerini üretir. Bu sinyaller, bir veya ardışık iki kare dalga darbeden oluşur. Bunlar, işlemci yongasındaki bağlantı uçlarına gider ve bir komutun yürütülmesindeki aşamaların her birininin zamanlamasında kullanılır. Çoğu komutu yürütmek birden fazla basamak gerektirdiğinden, her biri için birkaç saat periyotluk süreye gerek vardır. Bu konu Bölüm 3'de geniş olarak incelenecektir.

Saat osilatörü de sistemdeki entegre devrelerden birisidir ve entegrede ek bir dış kristale veya bazı tiplerde entegrenin dışında bir frekans-belirleyici devreye gerek duyulur. Ele alınan mikrobilgisayar sistemine bağlı olarak, 2 ile 8 MHz ya da daha yüksek saat frekansları kullanılabilir.

Güç kaynakları

Mikrobilgisayar sistemleri, çeşitli entegre devreleri tarafından gerek duyulan gerilim ve akımları üretmek üzere bir güç kaynağı içerirler. Gereken gerilim aralığı

sistemden sisteme değişir. Örneğin, bir Intel 8080 mikrobilgisayar yongası +12, -12, +5 ve -5 voltluk dört gerilime gerek duyarken, Zilog Z80 veya Intel 8085 aygıtları için sadece +5 volt'luk bir kaynak yeterlidir. Genellikle imalatçıların eğilimi, kullanılan kaynak çıkışlarını azaltmak yönündedir.

Bellekler

Mikrobilgisayar iki tip belleğe sahiptir: RAM ve ROM. Sadece-okunur bellek, standart liste ve tablolar gibi değişmeyen verileri ve program komutlarını tutar; rastgele- erişimli bellek ise, veriler üzerinde değişiklik yapılması veya ara sonuçların geçici olarak saklanması gerektiğinde kullanılır.

Her iki tip bellek, adres, veri ve denetim anayollarına bağlıdır ve bu yollar üzerinden diğer birimlerle bilgi alışverişinde bulunurlar.

ROM'lar (sadece-okunur bellekler), adından da anlaşıldığı gibi, yalnızca *okunabilirler*. Bu tür bir bellekte veriler, bellek imal edilirken kaydedilir. İşlemci RAM'leri, daha sonra hatırlaması gereken sonuçları yazmak için kullanabilir.

Diğer bir bellek ise, bir bakıma RAM ile ROM arasında bir özelliğe sahip olan PROM, *Programlanabilir Sadece Okunur Bellek*'tir. Normalde PROM da tıpkı ROM gibi davranır, ancak bir defaya mahsus olmak üzere PROM'a veri veya program komutları yazmak mümkündür. Bunun için *PROM Programlayıcısı* denilen özel bir gereç kullanılır.

Bir kez programlandıktan sonra PROM'ların içeriğini değiştirmek mümkün değildir. Fakat diğer bir bellek tipi olan EPROM (*Silinir Programlanır Sadece Okunur Bellek*) tekrar tekrar programlanabilir. EPROM'ların içeriği, normal olarak entegre devre paketinin üzerindeki pencereden morötesi ışın göndererek silinebilir ve PROM'da olduğu gibi yeni veriler yazılabilir. Bu işlem defalarca yinelenebilir. Belleklere ilişkin daha ayrıntılı bilgiler 2. Bölümde yer almaktadır.

Giriş/Çıkış yongaları

Mikroişlemci ile dış dünya arasındaki iletişimin önemine daha önce değinilmişti. Herhangi bir bilgisayar sistemi ne kadar güçlü olursa olsun, kendisinin de bir parçası olduğu dış dünya ile bilgi alışverişi olmaksızın ondan gereği gibi yararlanmak mümkün olmayacaktır.

Bu nedenle, daha sonraki bölümlerde tartışılacağı gibi, piyasada çevrebirimlerle

mikrobilgisayar arasında kolayca bağlantı kurabilecek biçimde tasarlanmış çok sayıda özel entegre devre bulunmaktadır. Bu devrelerden en çok bilinenleri; Çevresel Arabirim Adaptörü (Peripheral interface adaptor, PIA), Çevresel Arabirim Devresi (Peripheral interface circuit, PIC), Çevresel Giriş-Çıkış (Peripheral input-output, PIO) devreleridir.

Aslında bu devrelerin hepsi aynı işlevi görür. Bu devreler, bir bölümü doğrudan mikrobilgisayar anayoluna bağlı, diğer bir grubu da dış donanımların takılabileceği *portlar* oluşturan uçlar içerecek biçimde tasarlanmıştır. Mikrobilgisayar portları, tıpkı bir ülkedeki mal ithalat/ihracatının yapıldığı limanlarda olduğu gibi, sisteme bilgi girişine veya sistemden dışarıya bilgi çıkışına olanak tanırlar.

Çevresel arabirim devrelerinin sağladığı olanaklar, yongadan yongaya değişir; ancak, genelde her biri 8 giriş veya çıkış hattı içeren iki ya da üç adet porta sahiptir. Portlar, çıkış veya giriş olarak kullanılabilirler.

Bir portu giriş ya da çıkış olarak kullanmaya programcı karar verir. Çevresel arabirim devreleri *yazılımla düzenlenecek* şekilde tasarlanmıştır; yani programcı, hangi portun giriş, hangisinin çıkış olarak kullanılacağına karar verebilir ve programındaki komutları kullanarak ve denetim verileri göndererek PIC'ni ayarlayabilir.

Bu yüzden, programın başında programcının, gerek duyduğu biçimde çalıştırabilmesi için, mikrobilgisayar sistemindeki tüm çevresel arabirim devrelerini *kullanıma hazırlaması* gerekir. Daha sonra program içinde gerek duyarsa programcı, herhangi bir portun kullanım biçimini değiştirebilir, yani giriş olarak kullanılan portu bir çıkış portuna ya da çıkış portunu girişe çevirebilir. PIC'de (Çevresel Arabirim Devreleri) donanım değişikliği yapmaya gerek yoktur.

Seri iletişim

Çevresel arabirim devreleri, yukarıda özetlendiği gibi, giriş ve çıkış bilgilerini *paralel* biçimde verir. Bu nedenle, her bir port bir defada 8 bitlik bir veri alır ya da gönderir.

Bu tür paralel veri iletimi mikrobilgisayarlarla çevresel birimler arasında sıkça kullanılsa da, özellikle iletim hızının önemli olduğu durumlarda en ekonomik bağlantı tipi, *seri arabirimler* üzerinden yapılan bağlantılar olmaktadır.

Seri arabirimlerde, birimler arasındaki veri alışverişi, bir kerede 1 bit olacak

biçimdedir. Bir baytlık bir bilgi iletilmesi gerektiğinde de art arda 8 bit gönderilir. Bu şekilde işlem biraz daha yavaş gerçekleşir, ancak bağlantı kurmak için gereken hat sayısı çok daha az olacaktır.

Çevresel birimlerin bilgisayardan uzakta bulunduğu durumlarda ya da ikisi arasında telefon santralleri aracılığıyla bağlantı kurulması gerektiğinde, daha az arabağlantı gerektirmesi açısından seri iletişim daha avantajlıdır.

Seri veri iletiminde kullanılan büyük-kapasiteli entegre devreler uzun yıllardan beri üretilmekte ve piyasadan sağlanabilmektedir. Bunlar bir yandan işlemciyi veri, adres ve denetim anayollarına bağlarken, diğer yandan da seri veri aktarım hatları sağlamaktadır. Bunlar genelde, UART (Genel-Amaçlı Asenkron Alıcı ve Verici) ya da USART (Genel-Amaçlı Senkron/Asenkron Alıcı Verici) olarak adlandırılır-
S 1.7 lar. Bu devreler, 9. bölümde ayrıntılı olarak ele alınmıştır.

1.4 Bilgisayar ve Çevrebirimlerin hızları

Bilgisayar, işlevi gereği dış dünya ile sürekli bilgi alışverişinde bulunduğundan, pek çok değişik çevresel birim tipi geliştirilmiştir. İşlemci ve çevresel birimler arasındaki bilgi alışverişi ile ilgili olarak genelde karşılaşılan sorun, aralarında büyük bir hız farklılığı olmasıdır. İşlemci oldukça hızlı çalışan bir cihazdır; buna karşın, çevresel birimler genelde mekanik olarak çalıştıklarından veya bilgilere seri yoldan (yani, paralel çalışmadaki çok sayıda bit yerine bir kerede tek bir bit gönderecek biçimde) eriştiklerinden seri iletim çok daha yavaş olmaktadır.

Genelde kullanılan hız aralıklarına ilişkin bir fikir edinmek için, bazı tipik çevrebirimlerini inceleyelim. Bilgisayardan saniyede 10 karakter alıp yazabilen bir cihaz olan teletype en yavaş olanlardandır. Her bir karakter 11 bit ile ifade edilir, dolayısıyla iletim hızı 110 *baud*'dur (1 baud = 1 bit/sn). Mekanik işlemler de içeren ve biraz daha hızlı olanlar, saniyede 1000 adet 8 bitlik (8000 baud) karakter okuyabilen kağıt-bant okuyucuları ile satır yazıcılarıdır. Satır yazıcı, dakikada her biri 160 adet 8 bitlik karakter içeren 1000 satır yazabilir (yaklaşık 20 Kbaud'luk bir aktarım hızı). Diğer taraftan, sabit kafalı diskler saniyede 500.000 sözcüklük bir hıza sahiptir. 16 bitlik bir sözcük uzunluğu için bu, 8 Mbaud'luk bir hız demektir.

Mikrobilgisayar komut yürütme hızı, bir önceki bölümde de izah edildiği gibi, makinenin temel saat frekansı ve her bir komutun yürütümü için gerekli saat periyotlarının sayısı ile belirlenir. Bu ikinci parametre değişken olduğundan yürütme zamanı da komuttan komuta değişir.

Tablo:1.1'de, 2.048 MHz'lik bir saat frekansı (0.488 µsn saat peryodu) kullanan bir Intel 8080 işlemcisi için bazı örnek komut süreleri verilmiştir. Tablodan da görüldüğü gibi, bu mikrobilgisayar ve saat frekansı değeri için, bir komut 2 ile 10 µsn arasında bir zaman alır. Daha yüksek saat frekanslarında bu süre oldukça azalır ve 1 µs'den daha az bir yürütme zamanı elde etmek mümkün olur.

Tablo 1.1 *Komut yürütme süreleri*

Komut	*Kullanılan saat periyodu*	*Yürütme süresi/µs*
LDA (bellek adresi)	13	6.344
ORA (kaydedici)	4	1.952
JMP (adres)	10	4.880
MVI B (veri)	7	3.416
NOP	4	1.952

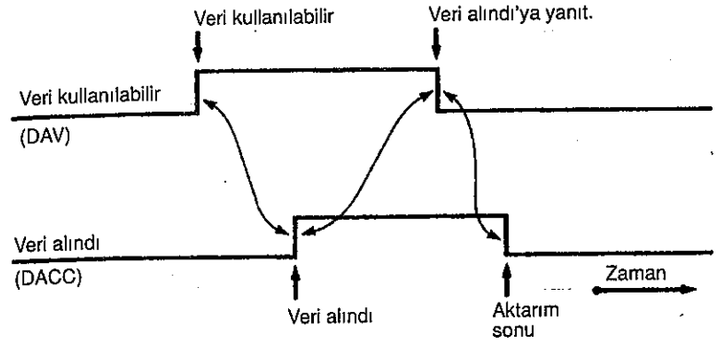
1.5 Çevrebirimleri ile eşzamanlama (senkronizasyon)

İşlemci ve çevrebirimleri arasındaki veri aktarım işlemlerinin tam zamanında senkronize edilmeleri gerekir. İşlemci komut takımı iki özel G/Ç komutuna sahiptir: IN ve OUT. İşlemci, programın yürütülmesi sırasında IN komutu ile karşılaştığında, bir çevrebirim portundan işlemciye veri alınmasını, OUT ile karşılaştığında da, işlemciden çevrebirim portuna veri gönderilmesini sağlar. Bu konular 2. Bölümde ayrıntılı bir biçimde tanımlanmıştır.

İşlemciden veri giriş-çıkışı belirli zaman dilimlerinde, yani programın yürütülmesi sırasında IN veya OUT komutları ile karşılaşıldığında gerçekleşir. Diğer yandan çevresel birimler de, belirli zaman dilimlerinde, yani veri alışverişine hazır olduklarında, veri göndermek veya almak gereği duyarlar. Diğer bir deyişle, işlemci ve çevrebirimleri kendi hızlarında çalışırlar; ancak, bilgi alışverişi gerektiğinde kısa süreler için ikisinin çalışmalarının bir biçimde birbirine kilitlenmesi gerekir. İşte *Onay (handshaking)* bunun için kullanılır.

Onay (handshaking)

Bir çevrebirimi, genellikle bir çevresel arabirim devresi aracılığıyla, mikrobilgisayara bağlandığında, işlemci ve dış aygıtı eşzamanlayacak (senkronize edecek)



Şekil 1.6 Onay

sinyalleri taşıyan iki bağlantı hattı kullanılır. Bu iki *denetim hattına* çeşitli adlar verilir. Biz de burada bu denetim sinyallerini DAV (veri kullanılabilir) ve DACC (veri alındı) şeklinde adlandıracağız.

Şekil 1.6'da, çevrebirimi/işlemci veri aktarımı sırasında bu hatlardaki sinyal desenleri gösterilmiştir.

İşlemcinin bir çevrebirimine veri göndermekte olduğunu düşünelim. Program içinde OUT komutuna gelindiğinde veriler, çevresel birime gönderilir ve DAV denetim hattı da mantıksal 1'e getirilir. Bu, sinyal çevrebirimine, verinin alınmaya hazır olduğunu gösterir. Çevrebirimi, DAV'ın '1' olduğunu algıladıktan sonra veriyi alır ve DACC'ı "mantıksal 1" değerine getirir. Bu sinyal ise işlemciye, verinin alındığını söyler.

İşlemci DACC "veri alındı" sinyalinin 1'e çekilmesine DAV'ı sıfırlayarak yanıt verir; çevrebirimi de buna karşılık DACC'ı sıfırlar ve böylece veri iletimi tamamlanmış olur.

Veri iletimini her zaman işlemcinin başlatması gerekmez. İşlemciye veri göndermek istediğinde, çevrebirimi de DAV'ı 1 değerine getirerek veri aktarımı yapabilir. İşlemci de buna yanıt verir ve aktarım yukarıda anlatılan biçimde devam eder.

Anlaşma-denetimli aktarım genellikle şöyle gerçekleştirilir; veriyi gönderecek taraf (bu, mikrobilgisayara bilgi girişi yapacak bir çevrebirimi ya da çıkış verisini çevrebirimine gönderecek olan mikrobilgisayarın kendisi olabilir), veriyi çıkış

hatlarına sürer ve DAV (veri kullanılabilir) sinyali gönderir, yani DAV'a 1 değ verir. Alıcı birim (bilgisayar veya çevrebirimi) veriyi alır ve DACC (veri alındı sinyalini mantıksal 1'e çeker. Gönderici, DACC'ın 1'e çekildiğini algılar ve bun DAV'ı sıfırlayarak yanıt verir ve bunun üzerine alıcı, DAV'ın çekildiği değ algılayarak bir sonraki veri aktarımına hazırlamak için DACC'ı sıfırlar.

Çevrebirim ve kesme taramaları

Anlaşma işlemi, işlemci ile çevrebirimi arasındaki veri aktarımını eşzamanlaması na rağmen, tüm işlemci/çevrebirimi iletişim problemlerinin üstesinden gelmek için tek başına yeterli değildir.

Bu problemler, Kısım 1.4'de sözü edilen hız farklılığı ile ilgilidir. Örneğin bi çevresel birim, işlemcinin binde biri oranında bir hızda (sözgelimi çevresel aygı her milisaniyede bir, işlemci ise her mikrosaniyede bir veri aktarımı yapabilecek biçimde) çalışıyorsa, işlemci veriyi ne zaman alabileceğini ya da gönderebileceğini nasıl bilebilir?

Bu konuya ilişkin iki önemli teknik geliştirilmiştir:

1 Yazılım kullanarak çevresel birimlerin taranması
2 Dışsal kesmeler

Kesmeler 8. bölümde, yazılım taraması konusu da 9. bölümde ayrıntılı olarak ele alınmıştır. Bu aşamada özet bir bilgi vermek yeterli olacaktır.

Yazılımsal tarama, işlemciye ait çevresel aygıtların (veri alışverişine) hazır olup olmadıklarının sorgulandığı bir işlemdir ve çevresel birimleri aşağıdaki örnekte olduğu gibi sırayla sorgulayan komut listesinden oluşur:

'Çevrebirimi 1, veri var mı?'
'Çevrebirimi 2, veri var mı?'
'Çevrebirimi 3, veri var mı?' vb...

Eğer yukarıdaki sorulardan herhangi birine çevresel birimlerden gelen yanıt 'Evet' ise, veri aktarımı başlatılır. 'Hayır' cevabı alındığı sürece aynı soru yinelenir.

Kesmeler, bir çevresel birimin, işlemcinin dikkatini çekmek istediğinde ürettiği sinyallerdir. Bu sinyaller işlemcinin, programın kesmeye ilişkin özel bir bölümüne atlanmasına neden olur. Örneğin, bir çevrebirimi, işlemciye göndereceği bir veri

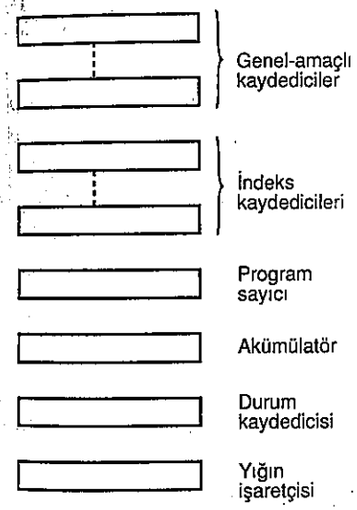
içeriyorsa, bir kesme gönderir. Bu kesme sinyalinin alınmasının ardından işlemci, veriyi alacak olan program parçasına atlar.

1.6 İşlemci olanakları ve komut takımı

İlerki bölümlerde, mikrobilgisayarların çeşitli yönleri ele alınacaktır. Kısım 1.2'de görüldüğü gibi, bunların tümü birbiriyle yakından ilişkili yazılım ve donanım konularını içermektedir. Bu nedenle burada, bilgisayar sistemi içerisindeki mikroişlemci yongasının olanaklarını kısaca gözden geçirmek yararlı olacaktır.

Bu nedenle aşağıdaki paragraflar, ilerideki bölümler için zorunlu bir ön bilgi olarak buraya alınmıştır. Burada verilen konuların çoğu, daha sonraki bölümlerde açıklanmaktadır. Bunun yanında, bu kitabın içinde baştan sona kullanılmış olan, Intel 8085 mikroişlemcisi komut takımı, Ek 1'de verilmiştir.

İşlemci yongasının olanakları



Şekil 1.7 İşlemci kaydediciler

Bir mikrobilgisayar sisteminin mikroişlemci yongası birçok kaydedici içerir. Kaydedici kavramı, özel-amaçlı bir kaydedici olan program sayıcısının anlatıldığı Kısım 1.2'de ele alınmıştı. Hatırlanacağı üzere, bu kaydedici, yürütülecek bir sonraki program komutunun adresini tutmak için kullanılmaktaydı.

İşlemci entegresinde bulunan diğer kaydediciler belli sınıflara ayrılabilir. Farklı üreticilerin yapmış olduğu farklı sistemlerin sunduğu olanaklar tabii ki farklılık gösterir, ancak genelde tüm kaydedici tipleri için Şekil 1.7'de verilen sınıflama kullanılır.

1 *Genel-amaçlı kaydediciler* Bunlar, bir hesaplama işleminde gerekli olan verileri tutmak amacıyla kullanılır. Hem aritmetik işlenenler hem de adresler bu kaydedicilerde saklanabilir.

2 *Akümülatörler* Mikroişlemcide işleme koyulan aritmetik ve mantıksal işlemler akümülatörde yapılır. Bu kaydedici genel-

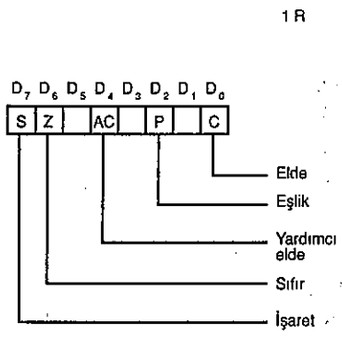
de, hem toplama veya çıkarma gibi iki-işlenenli komuta ilişkin işlenenleri, hem de işlemin sonucunda elde edilen değeri tutar. Mantıksal fonksiyonlar, kaydırma veya döndürme gibi yalnızca tek bir iş gerektiren komutlar da yine akümülatörde tutulan veriler üzerinde işlem yaparlar.

3 *İndeks kaydedicileri* İndeks kaydediciler, bir komutun adres parçasının oluşturulmasında kullanılır. Yani, bir komutun referans aldığı işlenenin bellek adresi, mikroişlemcilerin çoğunda, birisi indeks kaydedicisi olmak üzere, birkaç parçadan oluşur. Bu konu, Kısım 2.2'de ele alınmıştır.

İndeks-değiştirilmiş bir adreste, indeks kaydedicisinin içeriği, işlenenin tam adresini üretmek için komutta belirtilen *ofset* (kayma) adresine eklenir. Bu işlem karmaşık görünebilir, ancak birçok yararlı olanaklar sunar. İndeks kaydedicisinin içeriğini, programın denetimi altında değiştirmek yoluyla, bir programda bir veri listesi veya dizisi boyunca ilerlemek mümkündür. Ayrıca, bazı mikrobilgisayarlar, kaydedici içeriğini; adres oluşturulmasında kullanılmalarının sonrasında, otomatik olarak artırma (otomatik-adres artırma modu) veya adres oluşturulmasında kullanılmalarının öncesinde otomatik olarak azaltma (otomatik-azaltma modu) olanağına sahiptir. Buna alternatif olarak, adresin diğer parçaları, örneğin ofset (kayma) değeri değiştirilirken, kaydedici içeriği sabit tutulabilir. Bu durumda kaydedicinin, veri blokunun taban adresini gösteren bir işaretçi olarak işlev gördüğü düşünülebilir.

4 *Program sayıcısı* Daha önce de izah edildiği gibi, bu kaydedici, belli bir anda yürütülen komutun ardından gelen komutun adresini tutar.

5 *Durum kaydedicisi* Durum kaydedicisi, mikrobilgisayarın durumunu kaydetmek için kullanılan belli sayıdaki bayrak bitlerini içerir. Örneğin, Intel 8085 mikroişlemcisinde durum bitleri, Şekil 1.8'de gösterilen biçimde kullanılır.



Şekil 1.8 Zilog Z80 içindeki bayraklar

Herhangi bir aritmetik işlemin sonucunun işaret biti, D7 bayrağına yerleştirilir. Böylece, eğer işlemin sonucu negatifse D7 1, pozitifse 0 olur. Bu bit, sonunda bir koşullu dallanma

komutu ile test edilebilir. Örneğin, JP (pozitif olduğunda atla) komutu, D7'yi test eden bir komuttur. Testin sonucunda D7 bitinin sıfır olduğu görülürse, atlama gerçekleşir. Sıfır sonucunu üreten herhangi bir aritmetik işlem, D6'nın 1 değerine kurulmasına neden olur. Bu bit, JZ (sıfır olduğunda atla) komutu ile test edilebilir. Bu komut yürütüldüğünde D6, 1 değerine kurulu ise belirtilen atlama gerçekleşir. Diğer bayrak bitleri de aşağıda verildiği gibi kullanılır:

D_4 Toplayıcının 3 bitinden gelen elde çıkışını kaydetmek için kullanılır. Bundan, ikili-kodlanmış ondalık işlemler yapılırken yararlanılır.
D_2 *Eşlik bayrağıdır*. Tüm mantık komutları bu bayrağı etkiler. Eğer eşlik (bir işlemin sonucundaki birlerin sayısı) tek ise, eşlik bayrağı 0'a kurulur (sıfırlanır). Eğer işlemin sonucundaki birlerin sayısı çift ise bayrak 1'e kurulur.
D_0 Toplayıcının en ağırlıklı bitinden bir kalan değerin olduğunu gösterir.

6 *Yığın işaretçisi* Yığın işaretçisi, ileride daha ayrıntılı olarak tanımlanacağı gibi, rastgele-erişimli bellekte bir "son giren ilk çıkar" yığını oluşturmak için kullanılır.

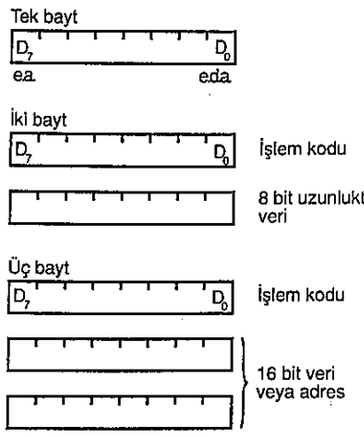
Komut takımı

Mikroişlemcilerde komut takımını oluşturan komutları sınıflara ayırmak uygun olacaktır.

Veri aktarım komutları Verilerin, bir yerden bir başka yere taşınması, bilgisayar programlarında çok sık rastlanan bir işlemdir. Bilgiler, temelde, bilgisayar belleğinde ya da işlemci kaydedicilerinde saklanır. Bu nedenle, bir mikroişlemcide bulunan veri aktarım komutlarının çoğu, bu kaydedicilerin içerdikleri değerlerin birbirleriyle değiş-tokuşu ya da kaydedicilerle bellek konumları arasındaki bilgi aktarımlarıyla ilgilidir. Tipik örnekler aşağıda verilmiştir:

Komut		*İşlem*
MOV	M,C	(kaydedici)'den → belleğe aktar
MOV	D,M	(bellek)'ten → kaydediciye aktar
MOV	C,H	(kaydedici)'den → kaydediciye aktar
LDA	adres	Adresteki bellek konumunun içeriğini → A'ya yükle
MVI	A,bayt	Veriyi A'ya dolaysız taşı

Bu komutlar Intel 8080/8085 komut setinden alınmıştır, ancak genelde mikrobilgisayar sistemlerinde kullanılan işlem tiplerini temsil ederler. Kaydedicinin parantez içinde yazılmış olması, 'kaydedicinin içeriği' anlamına gelmektedir, yani (B) 'B'nin içeriği' demektir.



Şekil 1.9 Komut formatları

Bu örneklerde kullanılan ve C, D, L ve H olarak belirtile kaydediciler, 8080 ve 8085 mikroişlemcilerinin kaydedici t kımında yer almaktadır. Bu birimler, B, C, D, E, H ve L olara adlandırılan altı adet genel-amaçlı kaydedici içerirler.

8080 ve 8085 mikroişlemcilerindeki komutlar, Şekil 1.9'd gösterildiği gibi, bellekte bir, iki ya da üç baytlık yer kaplaya bilirler. Belirli bir komutu kaydetmek için gerekli bayt sayısı o komut içinde ne kadar bilgi ifade edilmesi gerektiğine bağ lıdır.

A kaydedicisine (yani Akümülatör'e) bellekten bir değer yük lendiğinde, ilgili değeri almak için kullanılacak bellek adres 3 bayt uzunluktaki 'LDA adres' komutunun 2. ve 3. baytların da belirtilir.

'MVI A, bayt' komutu, 'bayt' ile belirtilen veriyi akümülatör taşır. Örneğin, MVI A,37 komutu, +37 değerini A akümülatö rüne yerleştirir. Bu komutu saklamak için, A'ya yerleştirilece olan veri ikinci baytta olacak şekilde, iki bayt uzunlukta bi bellek alanına gerek duyulur.

Herhangi bir komutta kullanılan bellek adresi, az önce sözü edilen "LDA adres" komutunda olduğu gibi, doğrudan baz komutların içine yazılabilir. Bazen de, bellek adresinin, H v L kaydedicilerinde tutulan değer olacağı kabul edilir.

Bu nedenle, MOV M,C komutu, C kaydedicisinde tutula sayıyı, adresi H ve L'de tutulan bellek alanına aktarır. H kaydedicisi, adresin en ağırlıklı 8 bitini, L kaydedicisi ise e az ağırlıklı 8 bitini tutar. Verinin varış yeri (burada M, belle konumu) komutta, verinin kaynağından (yani C'den) önc yazılır.

'MOV M,C' ve benzeri komutlar bellekte bir baytlık ye tutarlar.

Aritmetik ve mantıksal komutlar

İşlemciler, genelde aritmetik işlemler yapmak ve bit desenle

rini işlemek için, oldukça kapsamlı olanaklara sahiptirler. Bazı örnekler aşağıda verilmiştir:

Komut		*İşlem*
ADD	E	(kaydedici)'sini A ile topla
SUB	H	(kaydedici)'sini A'dan çıkar
INR	B	(kaydedici)'sini 1 arttır
DCR	L	(kaydedici)'sini 1 azalt
ADI	bayt	Veriyi A'ya dolaysız ekle
RLC		(A)'yı bir konum sola döndür
RRC		(A)'yı bir konum sağa döndür
ANI	bayt	(A)'yı dolaysız veri ile mantıksal VE'le

Program akış denetimi

Komut		*İşlem*
JMP	adres	'adres'e koşulsuz atla
JZ	adres	'Sıfır'sa 'adres'e atla
JP	adres	'Pozitif'se 'adres'e atla
JPO	adres	'Tek eşlik' ise 'adres'e atla
CALL	adres	Altyordamı çağır
RET		Altyordamdan dön
CZ	adres	'Sıfır'sa alt programı çağır
CPE	adres	'Çift eşlik' ise alt programı çağır
RZ		'Sıfır'sa geriye dön

Yukarıda sıralanan komutlar içinde sözü edilen *altyordam* kavramı 5., 6. ve 7. Bölümlerde tüm ayrıntılarıyla ele alınacaktır.

Yukarıdaki atlama komutları, bu bölümün girişinde sözü edilen koşullu dallanma komutlarıdır. Bu komutlar durum kaydedici bitlerinin değerlerinin test edilebilmesine ve programdaki denetim akışının bu testin sonuçlarına göre gözden geçirilebilmesine olanak tanır.

Giriş-çıkış

Komut	*İşlem*
IN adres	'adres' portundan A'ya değer gir
OUT adres	A'dan 'adres' portuna değer çıkart

EI	kesmeleri yetkile
DI	kesmeleri yetkisizle

Çeşitli yardımcı işlemler

Komut	*İşlem*
NOP	Hiç işlem yapma
HLT	Durdur
RST	Yeniden başlat

Modern bir mikrobilgisayar sisteminin tüm komut takımları, yukarıda verilmemiş olan daha başka bir çok komut içerirler. Bununla birlikte, yukarıda verilenler programcının kullanabileceği tipik komutlarıdır.

1.7 Assembly (simgesel) dili ve makine kodu

Bir önceki bölümde komutlar, *komut kısaltmaları* biçiminde yazılmıştı. Bu gösterim biçimi, burada ve ilerideki program örneklerinde, okunması ve anlaşılması kolay olduğu için kullanılmıştır. Gerçekte, bir program çalıştırıldığında mikrobilgisayar belleğinde saklı bulunan komutlar, ikili sisteme göre kodlanmıştır. Buna göre, MOV M,C komutunun kodu:

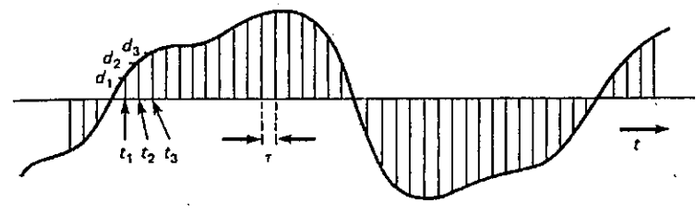
01110001'dir.

Bu, bellekte 1 bayt uzunlukta bir yer işgal eder. Bazen komutu *on altılı kodlar* şeklinde yazmak gerekebilir. Bu durumda 01110001 ifadesi, on altılı biçimde 71 (on altılı) olarak yazılır.

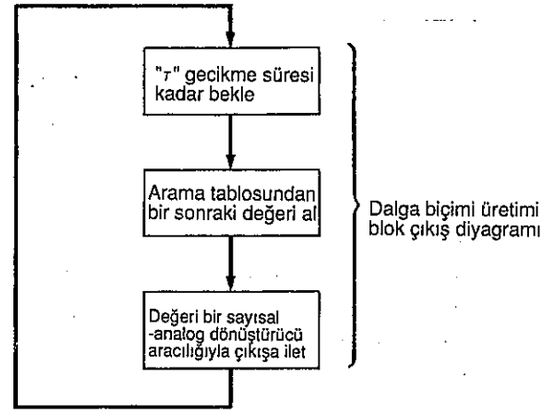
İkili ve on altılı kodların ikisi de assembly dili kadar elverişli değildir. Tabii ki assembly dilinde yazılan bir program bilgisayarda saklanıp yürütülmeden önce makine koduna çevrilmelidir; ancak bu, *Çeviriciler* adı verilen çevirici bir program aracılığıyla mikrobilgisayarın kendisi tarafından kolaylıkla yapılabilir.

Çeviriciler, giriş verisini, yani programcı tarafından assembly dilinde yazılan *kaynak* programı alır, bunu makine diline çevirir ve bilgisayarda saklar.

Assembly dili, bir komutun işlem kod kısmını göstermek için komut kısaltmaları kullanımına; adres ve verileri belirtmek için de *etiket* ve *ad* kullanımına olanak tanır. Assembly dili komutları, (assembler aracılığıyla) makine dili komutlarına çevrilir.



Dalga biçimi



Şekil 1.10

S 1.8 Assembly dili ile programlamaya ilişkin kavramlar basit bir örnekle gösterilebilir.

1.8 Zaman Gecikmeleri üretmek için program kullanımı

Bu bölümde özetlenen kavramları da içeren ilginç bir programlama örneği, gecikme üretmek için kullanılan programlardır. Bu tür gecikmelere uzun programlarda gerek duyulabilir. Gerçekten de, bir mikrobilgisayar darbe dizisi üretecindeki gecikme altyordamı örneği, 7. bölümde verilmiştir.

Belirlenmiş hassas zaman aralıklarının gerektiği durumlara ilişkin pek çok örnek verilebilir. Dalgabiçimleri üretmek için mikrobilgisayar kullanımında, örneğin, hassas zaman aralıklarında çıkarılacak dalgabiçiminin genliğini tanımlayan değerlere gerek duyulur. Eğer dalga biçimi ardışık biçimde devam ediyorsa bu değerler, bir arama tablosu olarak bellekte tutulabilir. Bu işlem süreci Şekil 1.10'da ana hatlarıyla verilmiştir. t_1, t_2, t_3 ... anlarında dalga biçiminin genlikleri, sırasıyla d_1,

d_2, d_3 ... değerleriyle belirtilir. Bu değerler, birbirinden ayrı τ saniye süreli zaman aralıklarında, dalgabiçiminin örneklem lerini gösterir. Dalgabiçimi üreten program, uygun bir çevresel arabirim devresi aracılığıyla, bir sayısal-analog dönüştürücü ye art arda değerler gönderir. Bunu yapmak için de, sistemin ölçümünü yapacak bir gereç içermesi gerekir.

Burada iki temel olanak vardır: birincisi, ileride tartışılacağı gibi, 'aralık zamanlayıcısı' denen özel bir devre eklemek; diğeri ise, gerekli gecikmeyi üretmek için, bilgisayarın komut yürütme süresini kullanmaktır.

Mikrobilgisayar komut takımında hiçbir işlem yapmayan bir NOP komutu vardır. Diğer bir deyişle NOP komutu, ne bellekteki veya işlemci kaydedicilerindeki bir değeri değiştirir, ne de bir başka işlevi yürütür. Ancak bu komutun yürütülmesi, işlemci zamanının bir bölümünü alır ki, bu da istenilen zaman gecikmesini üretmek için kullanılır.

Tablo:1.1'de verilen değerler göz önüne alındığında, her bir NOP komutu 1.952 µsn zaman almaktadır. 19 µsn'lik bir gecikme elde etmek için, 10 NOP komutu programda arka arkaya eklenmelidir. Daha iyi bir çözüm ise, içinde NOP komutu olan bir program döngüsü yazmaktır. Böyle bir döngü örneği Şekil 1.11'de verilmiştir.

Şekil 1.11 Gecikme akış diyagramı

Program oldukça basittir. Önce, üretilecek gecikmenin süresini belirleyecek bir sabit sayı, A akümülatörüne (A kaydedicisine) yüklenir. NOP komutunun ardından A kaydedicisinin değerini 1 azaltacak (bir çıkaracak) bir komut işleme sokulur. Bu işlemin hemen ardından ise, Akümülatördeki değerin 0 olup olmadığını öğrenebilmek için bir test yapılır. İstenen gecikme süresi geçildiğinde, program bir sonraki işlemle çalışmasına devam eder.

Akümülatördeki değer 0 değilse program, istenen gecikme süresine erişilmediği için NOP komutunu tekrar yürütmek üzere başa döner. A'nın içeriği '0'a eşit oluncaya kadar bu döngü sürer.

Bu işlem adımlarını yürütmek için yazılmış (soldan sağa doğru, 'etiket', 'komut', 'işlenen' ve 'açıklama' bölümlerinden oluşan) assembly dili program şöyledir:

```
            MVI   A,0A        ;A'ya +10 yerleştir
TEKRARLA:   NOP               ;Gecikme
            DCR   A           ;[A]'yı 1 azalt
            JNZ   TEKRARLA    ;[A], 0'a eşit değilse TEKRARLA'ya git
                              ;TEKRARLA
```

Programın başında A'ya yüklenen sabit sayı 10'dur. Eğer daha uzun süreli bir gecikme istenirse, verilen bu değer daha büyük bir değerle değiştirilebilir. Komutun içinde bu değer on altılı kodda belirtilmiştir [10 (ondalık) = 0A (on altılı)].

Döngü üç komuttan oluşmaktadır: NOP, DCR A ve JNZ REPEAT. Son ikisi sırasıyla 2.441 ve 4.882 sn'lik zaman almaktadır. Bu nedenle döngü on kez tekrarlandığında geçecek toplam süre :

10 (1.952 + 2.441 + 4.882) = 92.75 µsn olacaktır.

Programda TEKRARLA etiketinin kullanılması, atlama komutunun yazılmasını ve anlaşılmasını kolaylaştırmaktadır.

Programın ikili ve on altılı biçimdeki listesi Tablo:1.2'de gösterilmiştir. Makine kodu programı bellekte, 00AF (on altılı) ile 00B5 (on altılı) adresleri arasında yerleştiği varsayılmıştır. Tabloda yer alan komutlar, her birinin bellekte kapladığı bayt sayısını göstermek için çizgilerle ayrılmıştır.

S 1.9-1.11

Tablo 1.2

Adres (on altılı)	*Komut*		*Açıklama*
	İkili	*On altılı*	
00AF	00111110	3E	OA (on altılı) değerini
00B0	00001010	0A	A'ya yükle
00B1	00000000	00	NOP (hiç işlem yapma)
00B2	00111101	3D	(A)'yı 1 azalt
00B3	11000010	C2	(A) ≠ 0 ise
00B4	10110001	B1	00B1'e atla
00B5	00000000	00	

Sorular

1.1 Mikrobilgisayar sistemlerinin benzer uygulama alanlarını tartışın. Mikro gisayarların bu tür uygulamalarda tercih edilmelerini gerektiren özelli nelerdir?

1.2 *Donanım* ve *yazılım* terimlerinin anlamları nelerdir? Donanım ve yazılı hangi biçimde, bir mikrobilgisayar sistemi meydana getirdiğini açıklayın ikisi arasındaki ilişkiyi kısaca özetleyin.

1.3 Bilgisayar sistemlerinin gelişim tarihine ilişkin bir yazı yazın. Bu sistemle temel ilkelerini belirtin ve bunların, mikrobilgisayarların ortaya çıkışıyla gibi değişiklikler geçirdiğini izah edin. (Bunun için bir miktar daha bi edinmeniz gerekir.)

1.4 Aşağıda verilenleri kısaca tanımlayın ve her birinin kullanım yerlerini beli tin:

(a) Sadece okunur bellek.
(b) Rastgele erişimli bellek.
(c) İşlemci durum kaydedicisi.
(d) Assembly dili.

1.5 24-bitlik bir adres yoluna sahip bir mikrobilgisayarın maksimum adreslem alanı ne kadardır? Cevabı Kbayt ve Mbayt cinsinden ifade edin.

1.6 Bir bilgisayarın program içindeki ardışık komutların yürütülmesiyle ilgil olarak yapması gereken işlemleri özetleyin. Program sayacının, bu işlemler sırasındaki rolü nedir? JUMP komutu ile karşılaşıldığında ne gibi işlemle olduğunu belirtin.

1.7 Bir mikrobilgisayar sistemini oluşturan modülleri tanımlayın, kullanım amaç larını ve işlevlerini kısaca açıklayın.

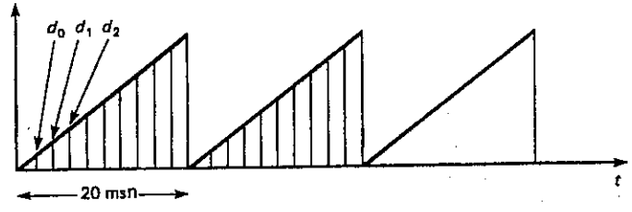
1.8 Aşağıdaki işlem adımlarını yapacak basit bir assembly dili program yazın Program şu işlemleri gerçekleştirecektir:

(a) C kaydedicisine dolaysız 92 (on altılı) değerini yükleyecek.
(b) D kaydedicisine dolaysız 64 (on altılı) değerini yükleyecek.
(c) C'deki değeri A'ya aktaracak.
(d) D'deki değeri A'dakine ekleyecek.
(e) A'daki sonucu bellekteki 3130 (on altılı) adresine yazacaktır.

1.9 Bir mikrobilgisayarın sekiz çıkış hattına sahip bir çıkış portu vardır. Bu hatlardan en küçük değerlikle olanı, bir mantık devresi üzerinden hoparlöre bağlanmıştır ve hattaki sinyal düzeyinin her 0'dan 1'e geçişinde, hoparlöre bir kare darbe uygulanmaktadır. 1'den 0'a geçişler hoparlörü etkilememektedir. Darbe süresinin 100 μsn olduğunu varsayarak, değişik nota sesleri üretmek için bilgisayarın nasıl programlanması gerektiğini gösteren bir akış diyagramı çizin.

1.10 Ek 1'de verilen Intel 8085 komut takımını kullanarak, 9. sorudaki akış diyagramını gerçekleyecek bir assembly programı yazın. Çıkış portunun adresinin 32 (on altılı) olduğunu varsayın.

1.11 Bir mikrobilgisayar sisteminden testere dişi dalga biçimi üretecek bir assembly programı yazın. Çıkış portunun adresini, 10. soruda verildiği gibi, 32 olarak alın. Çıkıştaki dalga biçimi, Şekil 1.12'deki gibi olacaktır. Testere dişi dalganın her bir dilimi, d_0, d_1, d_2, ... ,d_9, şeklinde on çıkış değeriyle ifade edilmektedir ve bu dilimler arasında 2 msn'lik zaman aralıkları vardır. İşlemi kolaylaştırmak için mikrobilgisayardaki her bir komutun 2 μs sürdüğünü varsayın.



Şekil 1.12

Bölüm 2 Mikroişlemci-tabanlı sistemlerin donanımı

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

1 *Bir mikrobilgisayar sisteminin donanım konfigürasyonunu tanımlayabilmeli,*
2 *Mikrobilgisayar yolu kavramı ile diğer standart yolların kullanılabilirliğini anlayabilmeli,*
3 *(a) komut takımı,*
(b) işlemci kaydedici olanakları,
(c) adres modu ve adres aralığı,
(d) işlenebilecek veri tipleri ve pratikteki sistemlerde bu özelliklerin kullanılabilirliklerinin değerlendirilmesi konularını da içine alacak biçimde, mikroişlemcinin önemli parametrelerini tanımlayabilmeli,
4 *Mikrobilgisayar sistemlerinde kullanılan bellek tiplerini, ve çeşitli tümleşik devrelerin parametre ve kullanılabilirliklerini tanımlayabilmeli,*
5 *(a) veri giriş kaydedicisi,*
(b) veri çıkış kaydedicisi,
(c) durum bayrakları ve
(d) denetim kaydedici birimlerini içine alacak biçimde bir PPI (paralel programlanabilir arabirimi) bileşenlerini ve özelliklerini tanımlayabilmeli,
6 *PPI bileşenlerinin adreslenmesinde ve tüm sistem içinden arabirim devrelerinin ya da bellek yongalarının seçiminde kullanılan yöntemi tanımlayabilmeli,*
7 *Pratikteki programlanabilir arabirim devrelerinin çalışma modlarını tanımlayabilmeli,*
8 *Bir çevrebirim elemanı olarak kullanılan aralık zamanlayıcının uygulama alanlarını, makinenin senkronizasyonunu sağlayan saat osilatörü ile bu eleman arasındaki farkları tanımlayabilmeli,*
9 *Bir aralık zamanlayıcı yongasında bulunan olanakları tanımlayabilmeli,*
10 *Bir aralık zamanlayıcısının, bir olay sayıcı ve gerçek-zamanlı saat olarak kullanımını tanımlayabilmeli,*
11 *Mikrobilgisayar sistemlerinde kullanılmak üzere neden bir dizi destek entegresine gerek duyulduğunu değerlendirebilmeli ve özellikle, bilgisayarın önemli elemanlarından olan tek ve çift yönlü yol tamponları, sürücüler ve adres kod çözücüleri tanıyabilmelisiniz.*

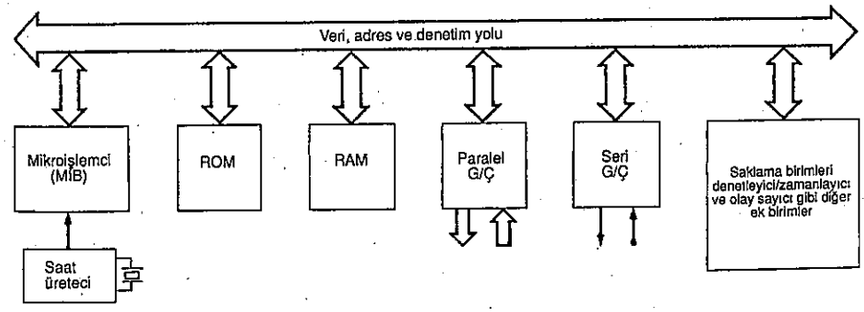
2.1 Giriş

Bir önceki bölümde mikrobilgisayar sistemi kavramı, bir modüller toplamı olarak tanımlanmıştı. Her bir modül, örneğin işlemci ya da rastgele erişimli bellek, bir veya daha fazla entegre devreden oluşur. İşlemci çoğu kez tek bir VLSI (Çok Büyük Ölçekli Entegre) devreden ibarettir. Bellek sistemi ise, normalde bir kaç büyük Ölçekli Entegre devreden oluşur. Bu bölümde, bu tür mikrobilgisayar sistemlerinin donanım özellikleri tartışılmakta ve içerdikleri mevcut olanaklar ve yongalar gösterilmektedir.

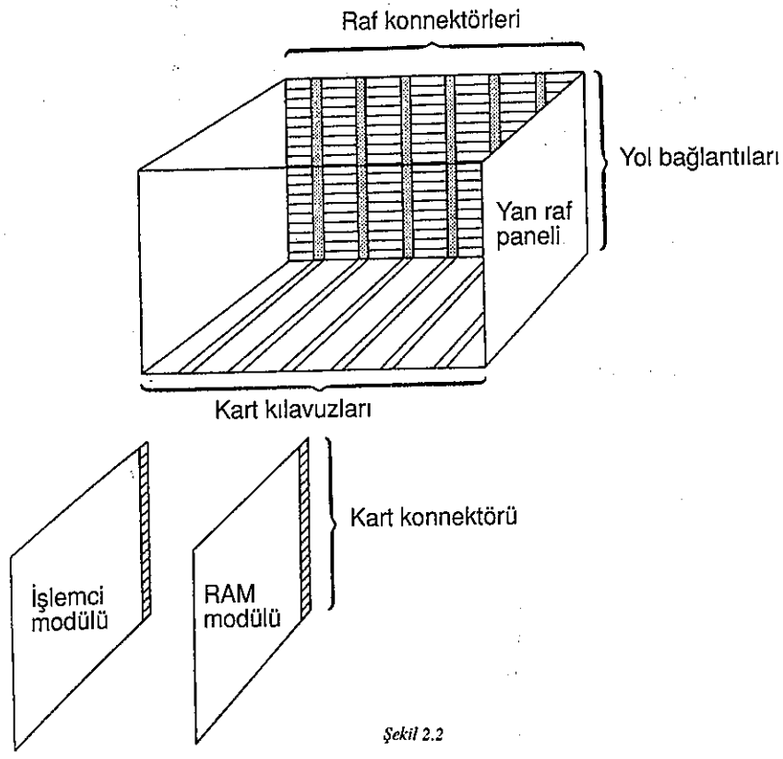
Şekil 2.1'de, tipik bir mikrobilgisayar sisteminin blok şeması verilmiştir. Görüleceği gibi, tüm sistemi oluşturan modüller; veri, adres ve denetim yollarıyla birbirine bağlanmıştır. Bu yollar, sistemin fiziksel özelliklerinin belirlenmesinde önemli rol oynar.

Sistem modülleri aşağıda verildiği gibi sınıflandırılabilir:

1 İşlemci modülü.
2 Bellek modülü.
3 Paralel giriş/çıkış modülleri.
4 Seri giriş/çıkış modülleri.
5 Çeşitli diğer modüllerle destek modülleri. Örneğin, bir mikroişlemci birimi, sistem zamanlama darbeleri üretmek için, bir *saat üretecine* ihtiyaç duyabilir. Disket sürücü gibi saklama birimleri, onları mikrobilgisayar yoluna bağlayacak bir *denetleyiciye* gerek duyarlar. Özellikle, mikrobilgisayarın gerçek-zamanlı uygulamalarında, *zamanlayıcı/olay sayıcı* yongası çok sık kullanılır. Yol sürücüleri, veri seçicileri ve çoğullayıcılar, adreslenebilir mandallar ve diğerleri gibi destek devreleri de, sistemin oluşturulmasında kullanılır.



Şekil 2.1 Bir mikrobilgisayar sistemi



Şekil 2.2

Donanım konfigürasyonu ve standart yollar

Mikrobilgisayarı oluşturan modüller, baskılı devre kartları üzerinde birleştirilir. Bu kartlar, arkalarındaki konnektörler üzerinde bulunan ve veri, adres ve denetim yollarını oluşturan bağlantı tellerini taşıyan bir raf sistemine takılırlar. Bu yapı, Şekil 2.2'de gösterilmektedir. Kartların uçlarında erkek, raflarda ise dişi konnektörler bulunmaktadır.

Burada en önemli olan nokta, raf üzerindeki yol bağlantı tellerinin her birinin işlevini ve konumunu tanımlamaktır. Farklı imalatçıların kartlar üzerinde yer alan modülleri geliştirebilmeleri ve üzerlerinde herhangi bir değişiklik yapmaya gerek kalmadan, bu kartların diğerleriyle birlikte sisteme takılabilmesi için, standart bir tanımlama gerekmektedir.

Mikrobilgisayarlarda kullanılan ortak bir standart S100 yoludur. Bu, 1976'da ALTAIR mikrobilgisayarlarının üreticileri olan MITS tarafından, hobi amaçlı

sistemlerde kullanılmak amacıyla geliştirilmiş, ancak daha sonra değiştirilmiş ve 'IEEE S100' standardı olarak yeniden tanımlanmıştır. Bu standart çok geniş çapta kabul görmüştür ve piyasada bu standartla uyumlu pek çok modül bulunmaktadır. Modüller, ilerki bölümlerde tek tek ve daha ayrıntılı olarak incelenecektir. Sistemdeki ana modül mikroişlemcinin kendisidir.

2.2 İmalatçı bilgi sayfalarının yorumlanması- işlemcilerin olanak ve özellikleri

Piyasada, çeşitli biçimlerde paketlenmiş VLSI devrelerinden oluşan birçok mikroişlemci mevcuttur. En yaygın paketlenmiş tipleri, sağladığı olanaklara ve makinenin mimarisine bağlı olarak değişen 40, 48 ve 64 ayaklı plastik ya da seramik DIL (çift taraflı) paketlerdir.

Mikroişlemcinin bazı önemli parametreleri aşağıda verilmiştir:

1 Teknoloji.
2 Komut takımı.
3 Komut yürütme hızı (saat hızı).
4 İşlemci kaydedicileri.
5 Adresleme aralığı (adres yolunun genişliği).
6 Adresleme modları.
7 Veri tipleri ve veri yolunun genişliği.
8 G/Ç olanakları; bellek-haritalı G/Ç.
9 Veri ve adres yolu organizasyonu.
10 Güç kaynağı gerekleri; güç tüketimi.
11 Diğer yonga-yerleşik olanakları.

Şekil 2.3'de, tipik bir işlemci yerleşim düzeni görülmektedir.

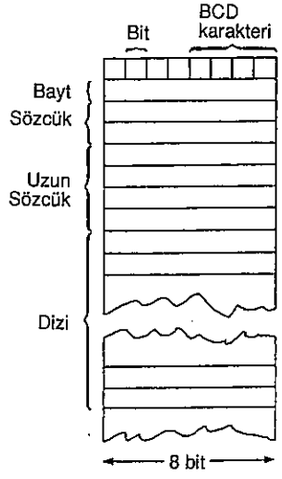
İşlemcinin ana bileşenleri; iç kaydediciler, aritmetik mantık birimi ve durum yazmacı, komut kaydedicisi ve denetim devreleri, veri ve adres yolu tamponları ve denetim devreleridir.

Komut takımı ve veri tipleri

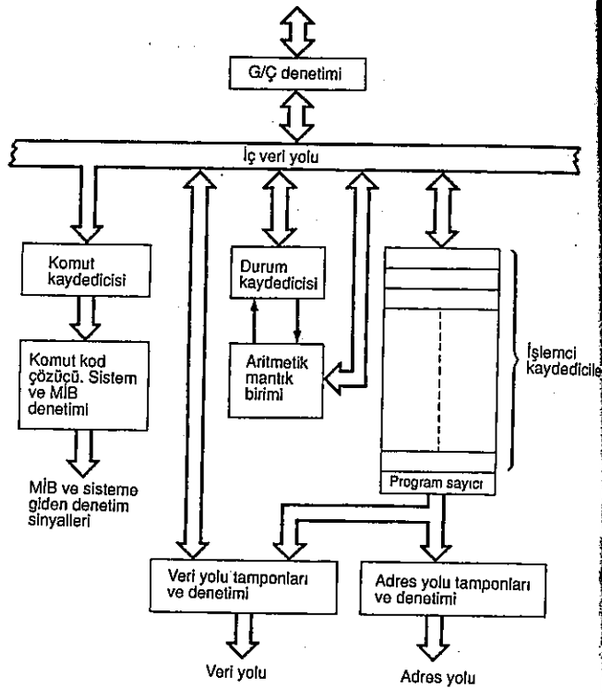
1. Bölümde, tipik bir mikroişlemcinin komut takımı ana hatlarıyla verilmiştir. Ek 1'de, Intel 8085 komut listesini bulabilirsiniz. Çeşitli sistemlerdeki komutların kullanım alanı, o sistemin üretim kuşağına bağlı olarak değişir.

Bu kitaptaki mikroişlemcilerin kuşak sınıfları aşağıda verile biçimde alınmıştır:

1. Kuşak - Intel 4004/4040 serisi gibi ilk mikroişlemciler
2. Kuşak - ilk 8-bitlik makineler: Intel 8080, Motorol MC6800, vb.
3. Kuşak - Intel 8085, Zilog Z80, vb.
4. Kuşak - Motorola MC68000 , Zilog Z8000, Intel 8086, vb



Şekil 2.4 Veri tipleri



Şekil 2.3 İşlemci blok diyagramı

Elbette, bu örneklerde sözü edilmeyen daha birçok mikroişlemci bulunmaktadır. 2. ve 3. kuşak makineler, veriyi aktarmak ve işlemek için oldukça iyi komutlara ve ayrıca elverişli nitelikte koşullu dallanma ve giriş/çıkış komut serisine sahiptirler. 8-bit uzunlukta veri işleyebilecek biçimde düzenlenmişlerdir ve 8-telli veri yolu içerirler. Bazen bu mikroişlemciler, 16-bit uzunlukta işlenenler üzerinde aritmetik işlemler yapabilirler. Örneğin, Zilog Z80 mikroişlemcisi, 16-bitlik değerlerin toplama-çıkarma işlemlerini gerçekleştirebilir.

4. kuşak birimler çok daha esnektir. Özellikle, bit, karakter, bayt, sözcük, uzun sözcük ve diziler de dahil olmak üzere veri tipleri ailesi üzerinde işlem yapabilirler. Bunlar, Şekil 2.4'de gösterilmektedir. Bu makinelerin veri yolları genelde 16-bağlantı teli içerir.

Bilgiler, bellekte bayt olarak yani 8-bitlik gruplar halinde tutulur. Yarım-bayt (4 bit), ikili kodlanmış ondalık bir *karakteri* gösterebilir. İki bayt (16 bit), bir *sözcük;* dört bayt, bir *uzun-sözcük*; daha fazla sayıda bayt ise bir *dizi* oluşturur. Mikroişlemci, bu tip verilerin hepsini ya da bir bölümünü işleyebilecek komutlar içerebilir. Örneğin, Zilog Z8000 bunların tümüyle işlem yapabilir.

Bazen mikroişlemci, yalnızca bir komutla bir veri dizisinin tümünü bir yerden bir yere taşıyabilir. Büyük miktarlardaki bilgileri ekonomik bir şekilde, yani çok kısa programlar kullanarak işleme tabi tutabildiğinden, bu tür komutlar çok güçlüdür.

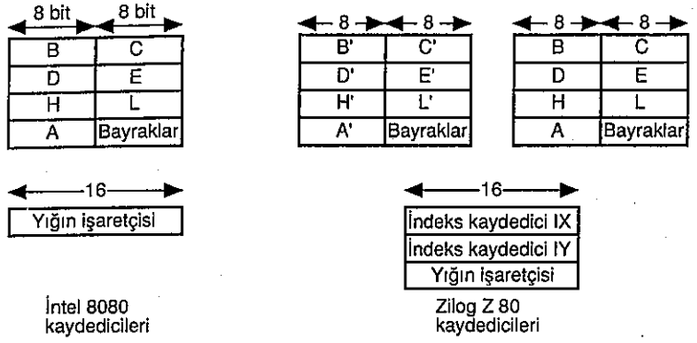
Dördüncü kuşak makinelerdeki diğer komutlar ise, çarpma ve bölme işlemlerini içerir. Daha önceki sistemlerde bu işlemler, işlemcinin toplama ve çıkarma komutları kullanılarak yazılan yazılım yongaları yardımıyla gerçekleştirilmek zorundaydı. Bu türde yongalar çok yavaş çalışmaktaydı.

İşlemci kaydedicileri

Mikroişlemcilerin gelişmesiyle, bir işlemci yongasında kullanılabilecek kaydedici sayısı ve bu kaydedicilerin kullanım kolaylığı artmıştır. Sonuçları belleğe yazarak saklama ihtiyacını azalttığı için, kaydedici sayısının fazla olması programlamayı kolaylaştıran bir etkendir.

Şekil 2.5, 2. kuşak bir işlemci (Intel 8080) ve 3. kuşak bir birim (Zilog Z80) içindeki kaydediciler arasındaki bir karşılaştırmayı göstermektedir. 8080 (Bölüm 1), 6 genel amaçlı kaydedici B, C, D, E, H ve L ile bir akümülatör (A) ve bir bayrak kaydedicisine sahiptir. Bundan başka bir de 16-bitlik yığın işaretçisi vardır. Bunun anlamı ilerki bölümlerde açıklanacaktır. Bütün genel amaçlı kaydediciler ve akümülatör 8-bit genişliğindedir, yani 8-bitlik bir sayı tutabilirler.

Z80; 8080'de bulunan normal bir kaydedici takımına ve buna ek olarak alternatif bir gruba, A', B', C', D', E', H' ve L' kaydedicilerine sahiptir. Makinenin komut takımı, programcının, normal kaydedici takımını ya da alternatif komut takımından



Şekil 2.5

birini kullanabilmesine olanak tanıyan bir değiş-tokuş komutu içerir. Bu düzenleme, işlemci entegresinde mevcut çalışma alanını iki katına çıkarmakta ve daha sonra da belirtileceği gibi, örneğin 'kesme yönetimi'nde çok yararlı olmaktadır.

4. kuşak mikroişlemci olan, Motorola MC68000'in kaydedici takımı Şekil 2.6'da gösterilmiştir.

İşlemci, her biri 32-bit genişliğinde 17 kaydediciye, yine 32-bit'lik bir program sayacına ve bir durum kaydedicisine sahiptir. İlk 8 kaydedici (Şekil 2.6'daki R_0-R_7) işlenenleri tutmak için, kalanlar (R_8-R_{14}) ise, adresleri oluşturmak için kullanılır.
S 2.3 Bu, çok esnek ve kullanışlı bir programlama yapabilme olanağı verir.

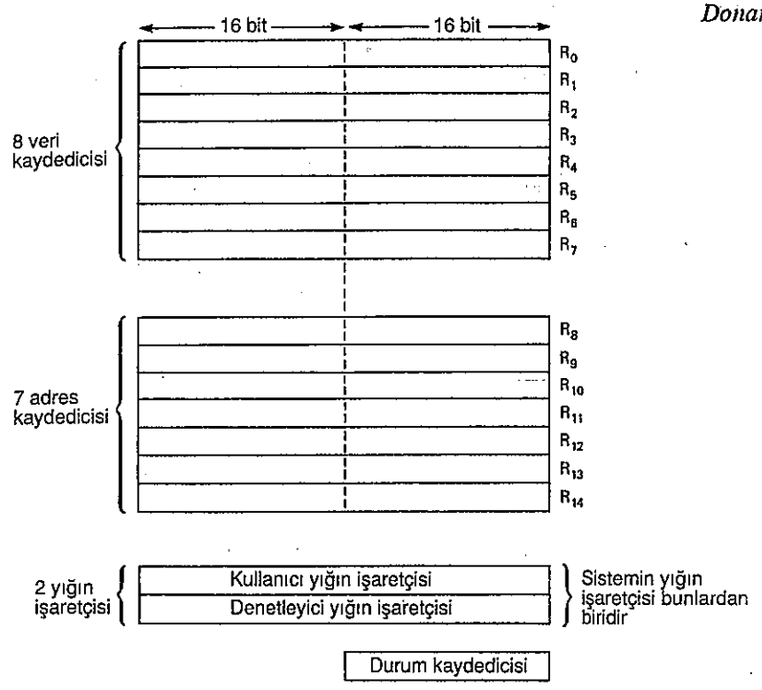
Adresleme modları

Bir ya da iki işlenene karşılık gelen her bir komut, adresleri de içermelidir. Örneğin:

ADD (1. işlenenin adresi) (2. işlenenin adresi)

komutu, 1. ve 2. işlenenlerin birbiriyle toplanması işlemini gerçekleştirir. '1. işlenenin adresi' ve '2. işlenenin adresi' değerleri, kaydedici numaraları -eğer 1. ve 2. işlenenler kaydedicilerde saklanıyorsa- veya bellek adresleri olabilir.

Adreslerin belirlenebildikleri çeşitli yöntemler, mikroişlemcinin *adresleme modlarını* oluşturur.
Genelde bir komut adresi birkaç parçadan meydana gelmiş olabilir. 1. Bölümde de görüldüğü gibi, örneğin indeks kaydedicileri, bir bileşik adresin parçalarından birini verebilir. Şekil 2.7'de, Zilog Z8000'in adresleme modlarını gösterilmektedir.



Şekil 2.6 Motorola MC68000 mikroişlemcisi kaydedicileri

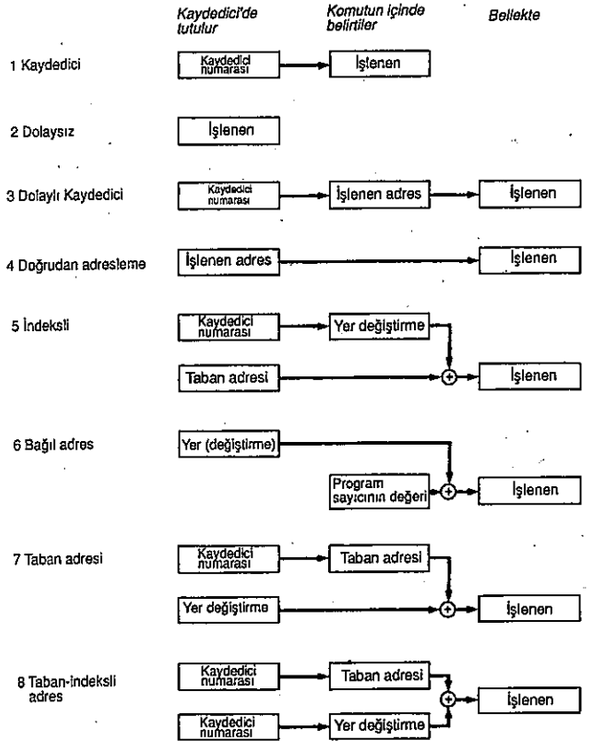
Diyagram, her bir adresleme modu için, komutta ve kaydedicilerde neler tutulduğunu gösterecek biçimde düzenlenmiştir. Örneğin, *kaydedici* modunda işlenen, işlemcideki bir kaydedicide saklanır. Bu durumda, komutun adres kısmı, yalnızca kullanılan kaydedicinin numarasını belirtir.

Dolaysız modunda (Immediate mode), işlenenin değeri doğrudan komutun içinde tutulur.

Dolaylı mod (Indirect mode), işlenenin adresini tutmak için kaydedicinin nasıl kullanıldığını göstermektedir. Kaydedici numarası komutun içine yazılır ve bu kaydedicinin içeriği, işlenenin adresi olarak kullanılır.

Doğrudan modda (Direct mode), komutun içinde işlenenin adresi bulunur.

Son dört modun hepsi, iki veya daha fazla parçadan oluşan bileşik adresler kullanır. Örneğin *indeks* modunda, işlenenin adresini oluşturmak için, komutun içinde tutulan taban adresi, kaydedicide tutulan yerdeğiştirme değerine eklenir. Mod 6, 7 ve 8 birbirinin benzeridir.



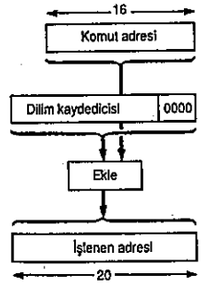
Şekil 2.7 Z8000 adresleme modları

Bu modların mevcut olması esnek adresleme yapısına olanak tanımakta ve programlamada, listelerden ya da veri dizilerinden bilgi çekilmesine yardımcı olmaktadır.

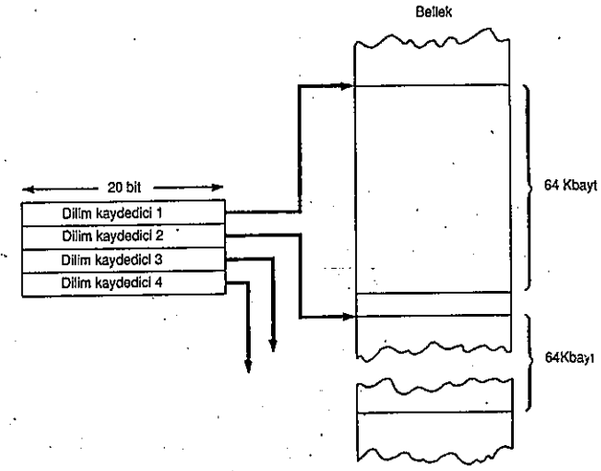
S 2.4, 2.5

Adresleme Aralığı

Açıklandığı üzere, makinenin adresleme modlarının kullanılması belli büyüklüklerde adres üretilebilmesine olanak tanır. Örneğin, işlemcideki kaydediciler, Zilog Z8000 veya Intel 8086'da olduğu gibi, 16 bit uzunluktaysa, adres de 16 bit uzunlukta olur. Bundan dolayı, adresleme aralığı da 64 Kbayttır. İşlemcinin 64 Kbayttan daha büyük bellek alanına erişebilmesini sağlamak için, 16 bitten daha fazla adres büyüklüğüne gerek olduğundan, bir adım daha gereklidir. Bu nedenle, işlemci tarafından üretilen adreslerin büyüklüğü 20 ya da daha fazla bite kadar artırılmalıdır.



Şekil 2.9



Şekil 2.8

4. kuşak mikrobilgisayarlarda adres genişlemesinin bir örneği olarak, Şekil 2.8'de 8086 sistemi gösterilmektedir. 8086, sözü edildiği gibi, 1 Mbaytlık bir bellek alanına erişimi sağlayan, 20 bağlantı telli bir adres yoluna sahiptir. Genel-amaçlı kaydedicilerin yanında 4 tane de *dilim kaydedicisi* bulunmaktadır. Bu kaydediciler, 20 bitliktir ve dolayısıyla 0 ile 1 Mbayt bellek aralığındaki herhangi bir adresi gösteren adres değeri tutabilirler. Uygulamada, bir dilim kaydedicisinin küçük değerlikli dört biti daima sıfır değerde olduğundan, içerdiği adresler 16'nın katları olmaktadır. Bu nedenle, bir dilim kaydedici 0 ile 1 Mbayt aralığında, 16'nın katları olan adresleri gösterebilir.

Dilim kaydedicileri, (segment register) Şekil 2.9' da gösterildiği gibi kullanılır.

Komutun içinde bulunan 16-bitlik adres, mevcut adresleme modu grubundan herhangi birini kullanmak suretiyle, 20 bit uzunlukta bir işlenen adresi üretmek için, işlemci donanımı tarafından otomatik olarak ilgili dilim kaydedicisi içeriğine eklenir. Kullanılan dilim kaydedicisi, yani bir önceki cümledeki 'ilgili' kaydedici, oluşturulan adresin tipine bağlıdır. Örneğin:

1 Program komutlarını bellekten getirmek için kullanılan tüm adresler *kod* dili kaydedicisine eklenir.
2 Program veri referanslarında kullanılan adresler, *veri* dilim kaydedicisini kullanır.
3 Tüm yığın referansları *yığın* dilim kaydedicisini kullanır vb.

Adres ve Veri Yolları

Bu bölümün başında da belirtildiği gibi, mikroişlemciler 40, 48 ve 64-pinli paketler halinde bulunur. Bir işlemci entegre devresinde gerek duyulan baca sayısını belirleyen önemli bir parametre, veri ve adres anayollarının düzenlenm biçimidir. Bazen, bunlar birbirinden tamamen bağımsızdır, yani her iki anayolu her bir biti için ayrı bir bacak ayrılmıştır, bazen de bazı bacaklar ortak kullanılı Örneğin, Motorola MC68000 işlemcisi 23 tanesi adres, 16 tanesi de veri yolu iç ayrılmış 64-bacaklı bir paket kullanır. 40-bacaklı bir pakete sahip olan Zilog Z80'd ise 16 bacak adres, 8 bacak veri yolu için ayrılmıştır. Diğer taraftan, Intel 8086, k tanesi veri yolu ile adres yolunun en küçük değerlikli 16 biti arasında paylaşılan v 4 tanesi de adres yolunun en büyük değerlikli dört biti tarafından kullanılan 40-bacaklı bir paket biçimindedir. Zilog Z8000 işlemci yongasında, ortak kullanı lan adres veri bağlantıları da bulunmaktadır.

Bir mikrobilgisayar sistemindeki bellek ve çevresel denetim devreleri, birbirinde ayrı adres ve veri giriş/çıkış hatlarına gerek duyarlar. Bu nedenle, *ortak* anayoldak adres bilgisi ve işlemciden çıkan veriler bölünmeli ve ayrı ayrı yollara yerleştiril melidir. Bu, aynı zamanda mikrobilgisayar sisteminin bir parçası olan özel devrel aracılığıyla gerçekleştirilir.

Ortak yoldaki adres ve veri bilgileri birbirlerinden, yola yerleştirildikleri *zamanla* açısından ayrılırlar. Diğer bir deyişle yol, *zaman-çoğullanır*. Örneğin, işlemci belleğe veri yazmak istediğinde ilk olarak, veriyi almak üzere adresi yola yerleş tirmelidir. Verinin, işlemciden belleğin veri giriş uçlarına aktarılabilmesi için yolun boşaltılması amacıyla, bu adres herhangi bir yere, örneğin bellek yongasına sak lanmalıdır. Daha sonra, bu veri de yol üzerine yerleştirilmelidir.

Eğer yol üzerindeki adres bilgisi geçerliyse, işlemci bunu dış aygıtlara bildirmek için, çıkış denetim hattında ALE Adres mandalını yetkilendirme olarak adlandırılan özel bir denetim sinyali üretir. Bu sinyal, dış birimlere adreslerin ne zaman inceleneceğini ve kullanılacağını gösterir. Bu konu, 3. Bölümde tüm ayrıntılarıyla anlatılmaktadır.

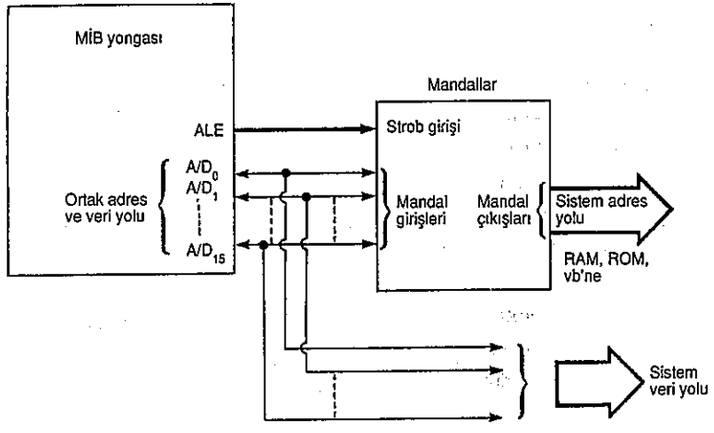
Adres bilgisini ortak yoldan ayırmak için gereken, o yolu, mandal (mandal, bir bit uzunlukta bellektir) grubunun girişine vermektir. Bazen bunlar, bellek yongasının içinde olacak şekilde üretilir ve daha önce belirtildiği gibi, adres o bellek yongasının içine saklanır. Alternatif olarak, dış mandallar, Şekil 2.10'da gösterildiği gibi, daha sonra mikrobilgisayar sistemindeki tüm modüller tarafından kullanılabilecek biçimde, birbirinden ayrı veri ve adres yolları sağlamak üzere kullanılabilir.

16-bitlik bir adres yolunda 16 mandal bulunması gerekir; ancak, bir tek yongada 8 mandal elde edilebildiğinden, bu topu topu birkaç entegre devreyi gerekli kılar. ALE denetim hattı, *strob* sinyali sağlamak için kullanılır. Bu, mandalların girişlerindeki verileri içeri almasına imkan verir ve böylece mandalların adres bilgisini 'dondurduğu' zaman parçasını tanımlar. Mandal çıkışları mikrobilgisayar adres yolu alarak kullanılabilir.

2.3 Bellekler

1. Bölümde özetlendiği gibi, mikrobilgisayar sistemindeki temel bellek tipleri, 'sabit' program belleği ile 'değiştirilebilir' veri belleğidir. Elbette, burada sözü edilen 'sabit' ve 'değiştirilebilir' ifadeleri, normal program yürütümü sırasındaki bellek işlemlerine karşılık gelmektedir. Program belleği, ROM ya da PROM'u, veri belleği de RAM'ı kullanır.

Bu kısımda amaç, üreticinin literatüründen bazı bellek örnekleri tanıtmaktır. Bu-



Şekil 2.10

nunla birlikte, ilk olarak bellek sistemlerinin bazı önemli parametrelerini incelem daha uygun olacaktır. Mikrobilgisayar sistemlerinde yarı-iletken bellekler kulla lır. En çok kullanılan teknolojiler, iki kutuplu transistör (azınlık ve çoğunl taşıyıcılarının her ikisinin bulunduğu transistörler) kullanan *iki kutuplu* teknol ile tek kutuplu alan etkili transistörler kullanan MOS teknolojisidir. En sık kul nılan teknoloji ise NMOS (N-kanal MOS)'dur.

Rastgele erişimli bellekler içinde saklama, *kalıcı-olmayan* özelliktedir, yani g kaynağı kapatıldığında, bellekte tutulan tüm bilgiler silinir. İki tip RAM vard *Statik* RAM'lar; iki kutuplu teknoloji ya da MOS teknolojisi kullanırlar, RAM' statik temelde flip-flop dizilerinden oluşur. Veriler, ancak devreye enerji verild sürece bellekte tutulurlar. Enerji kesildikten sonra güç kaynağı tekrar açılır flip-floplar gelişigüzel bir durum alacaklardır.

Dinamik RAM'lar verileri, devrede bulunan kondansatörlerdeki yük olarak tut lar. Bu tip bellekler yalnızca MOS teknolojisini kullanır ve kondansatörler depolanan yük zamanla azaldığından, bu belleklerin periyodik olarak tazelenme gerekir. Besleme gücü kesildiğinde ise tüm bilgiler silinir. Dinamik RAM' da v tazelenmesi her birkaç milisaniyede bir yapılır ve çoğunlukla, bunun otomat olarak gerçekleşmesi için gerekli devreler, bellek entegresinin içinde olacak biçi de üretilir.

Bir belleğin *erişim süresi*, verilerin o bellekten alınması için geçen süredir. *Say süresi* ise, bütünüyle bir oku-değiştir-yaz saykılını gerçekleştirmek, yani bellek bilgiyi alıp belleğe yeni bilgiler yerleştirmek için gereken süredir.

Bellek entegresindeki önemli bir faktör, uygulamadaki *paketleme yoğunluğud* çünkü bu, yongaya yerleştirilebilen saklama kapasitesinin miktarını belirler. Din mik RAM'larda paketleme yoğunluğu, statik tiplere göre daha yüksektir.

Dinamik RAM'larda *çalışma gücü* ile *beklemede harcanan güç* birbirinden fark dır; yani işlemci veya diğer birimler tarafından erişim yapılmadığı durumlard belleğin yalnızca veriyi tutmak için harcadığı güç, erişim yapılırken harcanand çok daha azdır. Bu nitelik, dinamik RAM'ların statik tiplere göre önemli b avantajıdır. Bu nedenle, her iki açıdan da, yani saklama kapasitesi ve güç tüketim açılarından bakıldığında dinamik RAM'lar; büyük bellek hacmi, düşük çalıştırm maliyeti ve bit başına saklama maliyetinin önemli olduğu uygulamalarda daha ço tercih edilir. Statik RAM'lar ise özellikle, daha düşük saklama kapasitesi ile erişi hızının başta geldiği uygulamalarda tercih edilir.

Dinamik RAM'lar

Bir dinamik RAM yongasından elde edilebilecek bellek kapasitesi oldukça çeşitlilik gösterir. 4.096 (4K), 8.192 (8K) ve 16.384 (16K) bitlik saklama alanı sağlayan bellekler, uzunca bir süredir piyasada bulunmaktadır, şu sıralar 65.536 (64K) kapasiteli bellekler piyasaya sürülmek üzeredir.

Bazı dinamik RAM örnekleri Tablo 2.1'de verilmiştir. Buradaki belleklerin büyüklükleri, "*N*x1 bit" olarak belirtilmiştir. Bu ifade, bellek tümleşik devresinin, her biri 1 bit uzunlukta *N* tane sözcüğü tutabildiği anlamına gelir. *N* sözcükten daha büyük uzunlukta, örneğin her biri 8-bitlik *N* tane sözcük, yani *N* bayt içinse, her biri sözcükteki bir bit için kullanılacak 8 tane bellek entegresine gerek duyulacaktır. Bu konu 4. Bölümde işlenecektir.

Tablo 2.1'de önerilen güç tüketimi değerleri (4.164 hariç) maksimum değerlerdir. Tipik değerler genellikle daha düşüktür.

Tablo 2.1

Bellek tipi	*Kapasite/ bit*	*Erişim süresi/nsn*	*Saykıl süresi/nsn*	*Güç kaynakları**	*Paket*
Intel 2104A	4.096 x 1	350	500	+12, +5, -5 V	16 bacak
Texas TMS 4108	8.192 x 1	100-250	373-515	+12, +5, -5 V 462 mW (Ç) 20 mW (B)	16 bacak
Intel 2109	8.192 x 1	200-250	375-475	+12, +5, -5 V 462 mW (Ç) 20 mW (B)	16 bacak
Texas TMS 4116	16.384 x 1	150-250	375-515	+12, +5, -5 V 462 mW (Ç) 20 mW (B)	16 bacak
Intel 2117	16.384 x 1	150-250	330-475	+12, +5, -5 V 462 mW (Ç) 20 mW (B)	16 bacak
Texas TMS 4164	65.536 x 1	150-326	256-326	+5 V 125 mW (tipik) 17.5 mW (tipik)	16 bacak

* (Ç) = çalışmada; (B) = beklemede

Burada dikkati çeken bir nokta da, tabloda sıralanan entegre devrelerin tümü iç
16 bacaklı paketler kullanılıyor olmasıdır. Paketlerdeki bacak sayısını 16'y
sınırlamak, bazı bacakların ortak kullanımını zorunlu kılacaktır. Örneğin 4116 tip
16K sözcüklük bir bellekte, bellek belirtimi için 14 bit kullanmak gerektiğinde
veri çıkışı için 1, veri girişi için 1, besleme girişleri için 3 ve çeşitli denetim
sinyalleri için de 3 veya 4 bacağa gereksinim vardır. Eğer her bir adres biti için b
bacak kullanılacağını varsayarsak, (14+1+1+3+4) = 23 bacak gerekecektir. B
sayıyı azaltmak için adres, her biri yedişer bit olacak biçimde iki parça olara
gönderilir. Böylece paketin adres girişleri için sadece 7 bacak kullanılır ve bunl
her bir erişim süresinde iki kez kullanılırlar. Bunu gerçekleştirmek için, bellek yo
ve adres girişleri arasında ek bir kaç yongaya daha ihtiyaç vardır.

Statik RAM'lar

Piyasada çok çeşitli değerlerde statik RAM'lar bulunmaktadır. Bunlar arasında
küçük, 64-bit kapasitede yüksek hızlı ve 35 nsn saykıl süresine sahip belleklerden
450 nsn saykıl süresine sahip 8.192 bit kapasiteli belleklere kadar pek çok stati
RAM mevcuttur. Bu bellekler genellikle sadece +5 voltluk bir besleme gerilimi
gerektirirler ve harcadıkları güç de, bellek hızına bağlı olarak oldukça değişen
değerlerde olmaktadır.

Bellek organizasyonu ve entegreler için kullanılan paketleme tipleri de değişiklik
gösterir. Örneğin, 4.096-bit kapasiteli bellekler, 1 bitlik 4.096 sözcük (4.096 x 1
veya 4 bitlik 1.024 sözcük (1.024 x 4) halinde bulunabilirler. 1.024 sözcüklü
bellekler, 1,4 veya 8 bitlik sözcük uzunlukları halinde de satın alınabilir. Paketler
16, 18, 20, 22 ve 24 bacak kullanır. Değerleri çok geniş aralıklarda değişen bazı
örnekler Tablo 2.2 de gösterilmiştir.

Tablo 2.2

Bellek tipi	*Kapasite/ bit*	*Erişim süresi/nsn*	*Güç kaynakları*	*Paket*
Texas TMS 4008	1.024 x 8	140-450	+5 V, 450 mW	24 bacak
Intel 2101A	256 x 4	350	+5 V, 300 mW	22 bacak
Texas TMS 4036-2	64 x 8	450-1000	+5 V, 450 mW	20 bacak
Intel 2114	1.024 x 4	450	+5 V, 525 mW	18 bacak
Texas TMS 4033	1.024 x 1	450	+5 V, 368 mW	16 bacak
Intel 3101A	16 x 4	35	+5 V, 525 mW	16 bacak

Tablo 2.3

Bellek tipi	*Kapasite/ bit*	*Erişim süresi/nsn*	*Güç kaynakları*	*Paket*
Texas TMS 4700 ROM	1.024 x 8	450	+5, -5, +12 V 580 mW maks.	24 bacak
Intel 2364A ROM	8.192 x 8	—	+5 V	28 bacak
Intel/Texas 2708 EPROM	1.024 x 8	450	+5, -5, +12 V 750 mW	24 bacak
Intel/Texas 2716 EPROM	2.048 x 8	450	+5, -5, +12 V 550 mW	24 bacak
Intel 2732 EPROM	4.096 x 8	450	+5 V 780 mW (Ç) 160 mW (B)	24 bacak

ROM ve PROMlar

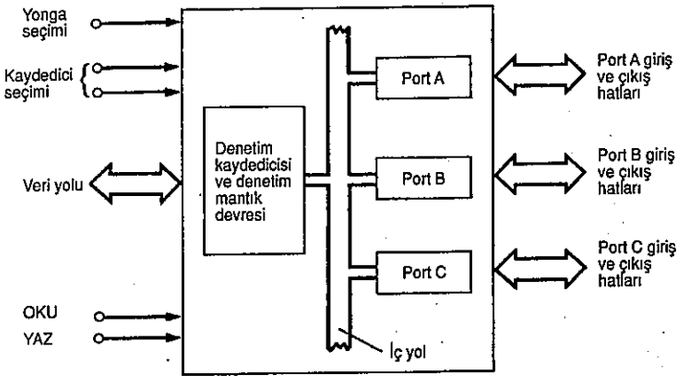
ROM ve PROM'lar kalıcı türde olduklarından özellikle program saklamak için çok uygundur. Bir ürünün büyük kısmının, tasarım test edilip onaylandıktan sonra satışa sunulacağı mikrobilgisayar uygulamalarında *maske-programlı* bir ROM kullanılır. Bu türde bir cihazın programlanması, üretimi sırasında yapılır. Bu, belleğin bit başına maliyetini azaltır; ancak tabii ki daha sonra yapılacak bir program değişikliği olasılığını da ortadan kaldırır.

Programlanabilir sadece okunur bellekler (PROM'lar), kullanıcı tarafından programlanabildikleri için bir miktar daha esnektirler. Silinebilir-programlanabilir sadece-okunur bellekler (EPROM'lar) ise, tekrar tekrar kullanılabildikleri için en esnek olanıdır. Tablo 2.3'de, bazı ROM ve EPROM örnekleri liste halinde verilmiştir.

Intel 2708, 2716 ve 2732 EPROM serisi bir endüstri standard olmuştur ve bu devreler çeşitli imalatçılarda mevcuttur.

Söz edilmeye değer diğer bir nokta ise, bazı ROM, EPROM ve PROM'lar arasındaki *soket uyumluluğu*dur, yani bacak bağlantıları bu tip yongalar çıkartılıp, diğerinin aynı yere takılmasına imkan verecek şekildedir. Bu özellik, ROM veya EPROM ile değiştirmeden önce, program yazmak için RAM kullanılarak sistem tasarımına imkan verir; bu nedenle, TMS 4008 statik RAM'ı, 2708 EPROM ve TMS 4700 ROM ile soket-uyumludur.

S 2.6, 2.7



Şekil 2.11 Bir çevresel arabirim devresi

2.4 Arabirim Elemanları

1. Bölümde, özellikle çevresel birimlerle mikrobilgisayar sistemi arasındaki bağlantıları sağlamak amacıyla, tasarlanan özel entegre devreler tanıtıldı. Şekil 2.1'de görüldüğü gibi, bu birimler, *seri* bilgi aktarım ve *paralel* bilgi giriş-çıkış işlevlerini gerçekleştiren birimler olarak ikiye ayrılır. Bunlardan ilki, daha önce kısaca incelenmişti ve daha geniş biçimde 9. Bölümde ele alınmıştır. İkincisi, yani paralel veri giriş-çıkış işlevi ise bu bölümde tartışılacaktır.

Bir çevresel arabirim devresindeki tipik olanaklar, Şekil 2.11'de blok diyagram halinde verilmiştir. Devrede:

1 Üç giriş/çıkış portu: A,B,C,
2 Bir *denetim kaydedicisi,*
3 PIC'nin (Çevresel Arabirim Devresi) doğrudan mikrobilgisayar yolu ve denetim hatlarına bağlanmasına olanak tanıyan arabirim devreleri bulunmaktadır. Bu devreler, yol sürüş ve yükleme gerekleriyle uyumludur. Bu nedenle, örneğin PIC'nin veri yolu hatlarında, üç-durumlu sürücüler kullanılır.

G/Ç Portları

Portlar, ya giriş ya da çıkış olarak kullanılırlar. Bu nedenle, bir çevresel birime bağlanabilir, bu birimden veri alabilir ya da bu birimlere veri gönderebilirler. Çevrebirimlerinden alınan veriler, portta bulunan bir veri kaydedicisinde saklanır (mandallanır). Bir çevrebirimine gönderilecek veriler de, yine porttaki tampon yükselteçlerinin çıkışından alınır.

Denetim kaydedicisi

Denetim kaydedicisi, A, B ve C portlarının işlevlerini denetleyen bir bit desenini tutar. Örneğin, kaydedicideki bir bit, A portunun giriş ya da çıkış olarak kullanılacağını belirler. Denetim kaydedicisi, (daha sonra da tanımlanacağı) gibi, mikrobilgisayar tarafından bu kaydedici içine kaydedilen 8 biti tutar.

Çevresel Arabirim Devresinin Adreslenmesi

Mikrobilgisayar sisteminde birden fazla çevresel arabirim devresi bulunabilir. Ayrıca, içinde her bir PIC'nin (Çevresel Arabirim Devresi) ayrı ayrı adreslenebileceği dört yer vardır; bunlar, A, B ve C portları ile denetim kaydedicisidir.

Mikrobilgisayar, her bir PIC ile, IN ve OUT (bkz: Bölüm 1) gibi normal giriş-çıkış komutlarını kullanarak iletişim kurar. Bu komutlar içindeki adres kısmı, gereken PIC'i seçmek ve PIC'in içindeki adresi belirtmek için kullanılır.

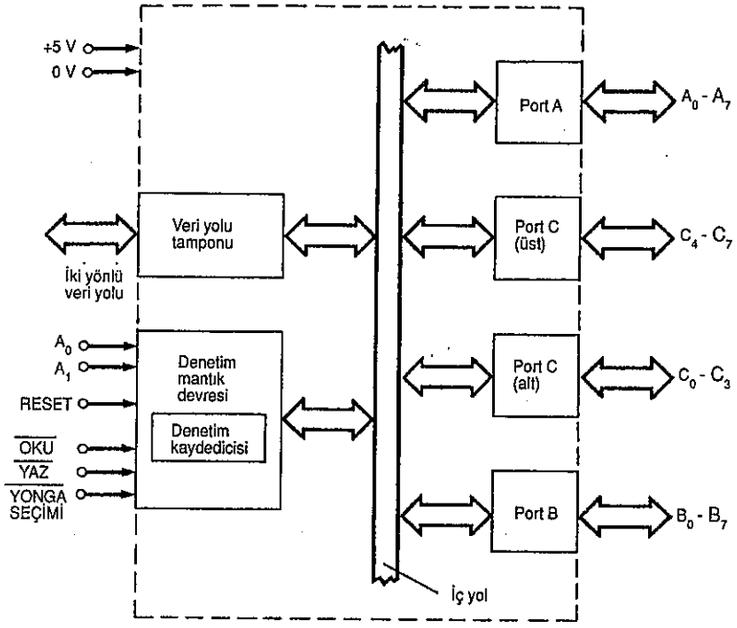
Belirli bir PIC'in seçimi, mikrobilgisayarın adres anayolundaki sinyallerinin kodlarını çözerek ve kod çözücü devrenin çıkışlarını, seçilen PIC'in 'yonga seçim' girişini yetkilendirmede kullanarak gerçekleştirilir. (Bkz: Şekil 2.11 ve 2.16). PIC içinde yer alan belirli bir adresin seçimi, 'kaydedici seçimi denetim girişleri' kullanılarak yapılır (Şekil 2.11).

Örneğin adresler, aşağıdaki gibi düzenlenebilir:

Yonga seçim hatları	*Seçilen adresler*
00	Port A
01	Port B
10	Port C
11	Denetim kaydedicisi

'Denetim kaydedicisi' sinyalleri de adres yolundan türetilirler.

Intel 8085 mikroişlemcisi gibi bir sistem için, IN veya OUT komutunun adres parçası 8 bitten ibarettir. Bu adres, adres yolunun en düşük ağırlıklı 8 hattına, yani A0-A7 hatlarına yerleştirilir.



Şekil 2.12

Birçok çevresel arabirim devresi içeren bir sistemde, bir kod çözücü, birbirinde farklı yol adreslerini algılamak ve buna göre seçim hatlarını sürmek için kullanıl bilir. Ancak genelde, en alttaki iki adres yolu hattı, doğrudan PIC'in kaydedi seçim hatlarına bağlanır. Bu, farklı PIC'leri seçmek için 6 adet hattı yedek bırak ve böylece sistemde olabilecek bu tür devrelerin toplam sayısını 64 ile sınırla PIC'lerin mikrobilgisayara bağlantısı, ana çizgileriyle Şekil 2.16'da gösterilmişti

PIC'lere ilişkin 'oku' ve 'yaz' denetim girişleri, veri aktarımının yönünü belirle "Oku", PIC'ten çıkan bilgilerin bilgisayar tarafından okunmasını sağlar, "yaz" i bilgisayardan PIC'e veri gönderir.

Bazen, bir mikrobilgisayar sistemindeki çevresel G/Ç olanakları, bir ya da dah fazla özel PIC yongası tarafından sağlanır. Diğer yandan bazı sistemlerde de, G/Ç olanakları diğer yongaların içerisinde, özellikle sistemin bellek yongalarında bulu nur. Örneğin, Intel 8155 entegre devresi, bir 2.048 bitlik statik RAM artı 2 tan programlanabilir 8-bitlik G/Ç portu ve bir programlanabilir 6 bitlik G/Ç portu içeri Bunlara ek olarak, bir de zamanlayıcıya sahiptir. Bu türde çok-amaçlı devrelerin

kullanılması, komple bir sistem oluşturmak için gerekli entegre devre paketlerinin toplam sayısını ve dolayısıyla maliyeti azaltır.

Çevresel Arabirim Devrelerinin Olanakları

Bu aşamada, pratik bir çevresel arabirim devresinde mevcut bazı olanakları incelemek yerinde olacaktır. Intel 8255A Programlanabilir Çevresel Arabirimi (PPI) bu tür birimlere bir örnek olarak kullanılmıştır. Bu birimin temel olanakları, Şekil 2.12'deki blok diyagramda gösterilmiştir.

Yonga, 40 bacaklı bir paketten oluşmuştur. Mikrobilgisayar sistemine bağlantı, üç-durumlu sürücüler kullanan 8 bitlik iki yönlü bir veri yolu üzerinden yapılmıştır. Mikrobilgisayara gönderilecek olan veri, örneğin daha önceki bir örnekte kullanılan bant okuyucusudan gelen veriler, iletime hazır durumdaki *veri yolu tamponu*nda tutulabilir. Benzer biçimde, bilgisayardan alınan veriler de tampona yazılır. Bu nedenle, tüm veri akışı bu tampon üzerinden gerçekleştirilir.

Denetim sinyalleri

Şekilde, çevresel arabirimlerle mikrobilgisayar arasında bağlantı kuran denetim sinyalleri de gösterilmiştir. Bu sinyallerin üçü, yani READ, WRITE ve CHIP SELECT (OKU, YAZ ve YONGA SEÇİM) sinyalleri, *alçak* durumda etkindir, yani bilgisayar bu hatlar tarafından denetlenen herhangi bir işlevi yerine getireceği zaman, bu hatlardan herhangi biri üzerindeki sinyali alçak düzeyine çeker. Alışılageldiği üzere, bu tür alçakta-etkin sinyaller $\overline{\text{READ}}$, $\overline{\text{WRITE}}$ vb. şeklinde yazılır. Sinyal adlarının üzerlerindeki çizgiler, o sinyalin alçakta-etkin olduğunu gösterir.

A_0 ve A_1, daha önce anlatılan kaydedici seçim hatlarıdır. $\overline{\text{CHIP SELECT}}$ sinyali ise, birden fazla çevresel arabirim devresi içeren sistemlerde belirli bir arabirim devresini yetkilemek için kullanılır. RESET denetim sinyali ise, PPI (Paralel Çevresel Arabirim) sıfırlamak için kullanılır. Bu hatta bulunan sinyal, denetim kaydedicisinin içeriğini sıfıra getirir ve böylece, daha sonra da gösterileceği gibi, tüm çevresel portları giriş moduna kurar.

Modlar ve G/Ç Portları

Arabirim devresi, A, B ve C olarak tartışılan her biri 8-bitlik üç G/Ç portuna

sahiptir. Biraz sonra izah edileceği gibi devre, mod 0, mod 1 ve mod 2 olmak üz
üç moddan birinde çalışabilir. Seçilen mod ve her bir portun nasıl çalışac
denetim kaydedicisindeki bit deseni tarafından belirlenir.

Şekil 2.12'de gösterildiği gibi, C portu; Port C (üst) ve Port C (alt) olmak üzere
biri 4 bitten oluşan iki kısma bölünmüştür. Çoğu işlemlerde Port C (üst), Por
ile, Port C (alt) ise Port B ile birlikte çalışır.

Denetim sözcüğü formatı

Denetim kaydedicisine yerleştirilen denetim sözcüğünün formatı, Şekil 2.13
gösterilmiştir.

0., 1. ve 2. bitler ('B grubu denetim' bitleri) port B ve port C (alt)'ın işlevler
denetler.

3., 4., 5. ve 6. bitler ('A grubu denetim' bitleri) port A ve Port C (üst)'ün işlevler
denetler.

7. bit, özel 'bit kurma' modu işleminde kullanılır. Burada, buna ilişkin daha faz
ayrıntıya girilmeyecektir.

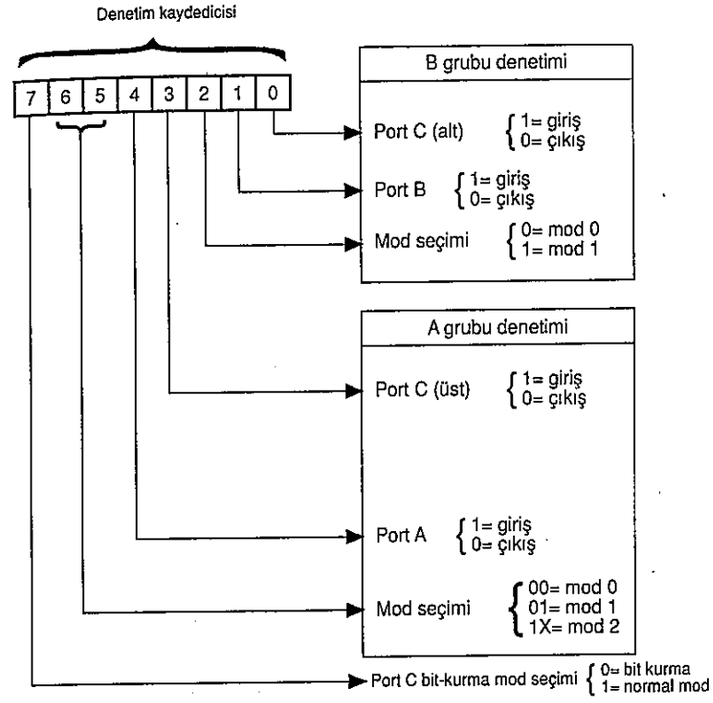
Tüm portlar, yani A, B, C (üst) ve C (alt) portları; 0., 1., 3. ve 4. bitler tarafınd
belirlenecek biçimde, giriş ya da çıkış olarak kullanılabilir.

Modlar, B ve C (alt) portu için 2. bit tarafından; A ve C (üst) portu için de 5. ve
bitler tarafından denetlenir.

Mod 0 Bu temel G/Ç modudur. Sistem, sonuçta dört port, yani 8-bitlik A port
8-bitlik B portu ve 2 tane de dörder bitlik C (alt) ve C (üst) portlarını sağlar.

Bir veri çıkışı yapmak için, bilgisayar seçilen porta veriyi (uygun adresle birlik
bir OUT komutunu) gönderir; bir çevrebiriminden veri almak için de, bir IN komu
kullanarak ilgili portu okur. Bu aşamada anlaşma olmaz. Çevresel birimin veri
kabul edebileceği hızın kritik değerde olmaması için, PPI'dan çıkan veri, portta
bir kaydedici içinde tutulur.

Mod 1 Şekil 2.13'de de belirtildiği gibi, G/Ç portları mod 1'de çalışmada, A ve
olmak üzere iki grupta işlev görürler. A portu, port C (üst) ile, B portu da port



Şekil 2.13 Denetim kaydedicisi formatı

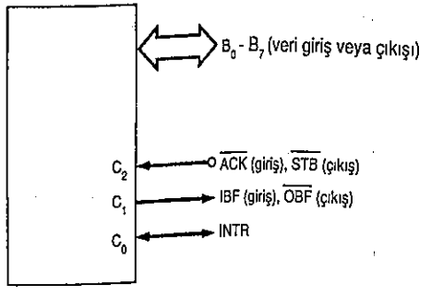
(alt) ile birer grup oluştururlar. 4 bitlik C portları, her biri onayı (handshake) sağlamada kullanılacak denetim sinyallerini sağlarlarken, A ve B portları 8 bitlik giriş ve çıkış olarak çalışırlar.

Bu modda, bir çevresel birimden A ya da B portuna gönderilen veriler, porttaki bir giriş kaydedicisinde saklanır. Benzer biçimde, bir çevrebirimine gönderilecek veri de, birim tarafından kabul edilene kadar bir çıkış kaydedicisinde saklanır.

PPI'nın grup B kısmının mod 1'deki çalışma biçimini gösteren bir diyagram, Şekil 2.14'de verilmiştir. Grup A kısmı da benzer biçimde çalışır.

Verilerin B portu üzerinden çıkışı için, mod 1'de işlemler, aşağıda verilen sırada gerçekleşmelidir:

1 Mikrobilgisayar, veri yolu üzerinden PPI'ya veri gönderir.
2 PPI veriyi, B portunun çıkış kaydedicisine yerleştirir ve çıkış $\overline{\text{OBF}}$ denetim hattını (C portunun C_1 bitini) alçak düzeye çeker.



Şekil 2.14 Port B, mod 1 biçimi

3 B portuna bağlı birim, yani çevrebirimi, veriyi alır ve PPI'nin $\overline{ACK}$ [
denetim hattını, veriyi aldığını göstermek için alçak düzeye çeker.
4 $\overline{OBF}$ tekrar yüksek düzeye döner. $\overline{OBF}$ sinyali, C portundaki 4-b
kaydedicinin 1. bitinden çıkar. Bu kaydedici, veri port B'ye gönderildiğ
(yukarıdaki 2. aşama) sıfırlanır ve $\overline{ACK}$ sinyali ile 1'e kurulur.

$\overline{OBF}$ ve $\overline{ACK}$ denetim hatları onay sistemini oluştururlar.

Veri, A portu aracılığıyla mikrobilgisayara girildiğinde benzer işlemler yürütü
Bu durumda kullanılan denetim sinyallerine $\overline{STB}$ ve IBF denir.

Verinin B portu üzerinden giriş-çıkışında, C port kaydedicisinin 1. biti; bi
aktarım işleminde gelinen durumu göstermek için kullanılır. Örneğin, veri
mikrobilgisayara girişinde, 1. bit; çevrebiriminden çıkan verinin B portund
kaydediciye yerleştiğini göstermek üzere *durum bit*'i olarak kullanılabilir. C p
tunun içeriğini okumak için kullanılan normal bir IN komutu, mikrobilgisayarın
biti gözden geçirmesine olanak tanır.

Port C kaydedicisinin diğer bitleri, mod 1 çalışmada, diğer G/Ç aktarımların
durumunu göstermek ve PPI'ın kesme olanaklarını denetlemek için kullanılır. B
ikinci nokta, bu kitabın kesmeler ile ilgili bölümünde daha geniş bir şekild
anlatılacaktır. Şu aşamada belirtilmesi gereken, port C kaydedicisinin, mod 1 (v
S 2.14 mod 2) çalışması için bir *durum kaydedicisi* olarak işlev gördüğüdür.

Mod 2 Mod 0 ve mod 1 çalışmada, programlanabilir çevresel arabirimleri
kullanımı, özetlendiği üzere, mikrobilgisayar için G/Ç olanakları sağlamaktadır
Bu modların yalnızca bir kısıtlaması vardır. Portlar, ya giriş ya da çıkış olarak
davranabilir, fakat *iki yönlü* olarak çalışamazlar, yani iki yönlü port gerçekleştirmek

mümkün değildir. Ancak, mod 2'de bu olanak vardır, bu modda A portu, yukarıda açıklandığı gibi, C portunu kullanan onay sinyalleri ile birlikte iki yönlü olarak çalışabilir.

Arabirim portları 8 ya da 4 bit genişlikte olmalarına rağmen, gönderilen (ya da alınan) veriler, söz konusu sınırlar içinde kalmak koşuluyla, herhangi bir genişlikte olabilir. Bunun en basit örneği, yalnızca 6 bitlik bir veriyi bir çevrebirimine göndermek için, 8 bitlik bir portun, sözgelimi PPI'nin A portunun, kullanılmasıdır. Portun diğer iki çıkış hattı dikkate alınmaz.

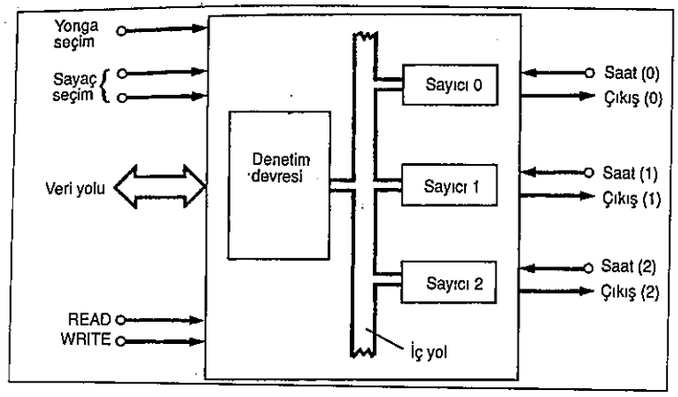
Benzer biçimde, PPI, bir çevrebiriminden -örneğin 16 bit uzunlukta- bir veri almak için kullanılabilir. Giriş bilgisinin 8 biti A portuna, 8 biti de B portuna yerleştirilebilir. Bu durumda mikrobilgisayar programı, birimin A ve ardından da B portunu okuyabilmesi için, iki IN komutuna ihtiyaç duyacaktır. Bu nedenle, bilgisayar veri yolundaki bağlantı hattı sayısının, çevresel birimle olan bağlantı hattı sayısına eşit olma zorunluluğu yoktur; programlanabilir çevresel arabirimi bir biçimden diğerine dönüşümü gerçekleştirmek için kullanılabilir.

S, 2.8, 2.9, 2.11 Son olarak, PPI'ın karma modlarla birlikte kullanılabileceği unutulmamalıdır. Böylece, grup A kısmı mod 2'de çalışırken, grup B kısmı da mod 1'de çalışacak biçimde düzenlenebilir. Veya grup A mod 1'deyken, grup B de mod 0'da olabilir.

2.5 Gerçek-zamanlı saatler ve aralık zamanlayıcıları

Kısım 1.3'de, bir mikrobilgisayar sisteminin çalışmasının, kristal denetimli bir *saat osilatörü* tarafından senkronize edildiğinden bahsedilmiş ve Kısım 1.8'de zaman gecikmesi üretiminde mikroişlemci komut yürütme süresinin kullanımı ile ilgili bir örnek verilmişti. Bu türde bir sistemin bazı dezavantajları bulunmaktadır. İlk olarak, mikrobilgisayar zamanının çoğunu bu zamanlama döngülerinin yürütümünde harcamaktadır. Diğer fonksiyonları gerçekleştirmek için araya program parçaları sokmak mümkündür, fakat program parçalarının alacağı sürenin kesin olarak hesaplanması gerektiğinden, bunu yapabilmek biraz güç olmaktadır. İkinci olarak, programın normal bir biçimde yürütülmesine dışardan karışan olayların -örneğin kesmelerin- zamanlama işlevini olumsuz yönde etkilemesine izin verilmemelidir.

Yazılımın denetimi altında duyarlıklı zaman aralıkları üretmenin daha iyi bir yolu, aralık zamanlayıcısı denilen özel bir yonga kullanmaktır. Bu birimin blok diyagramı Şekil 2.15'de görülmektedir.



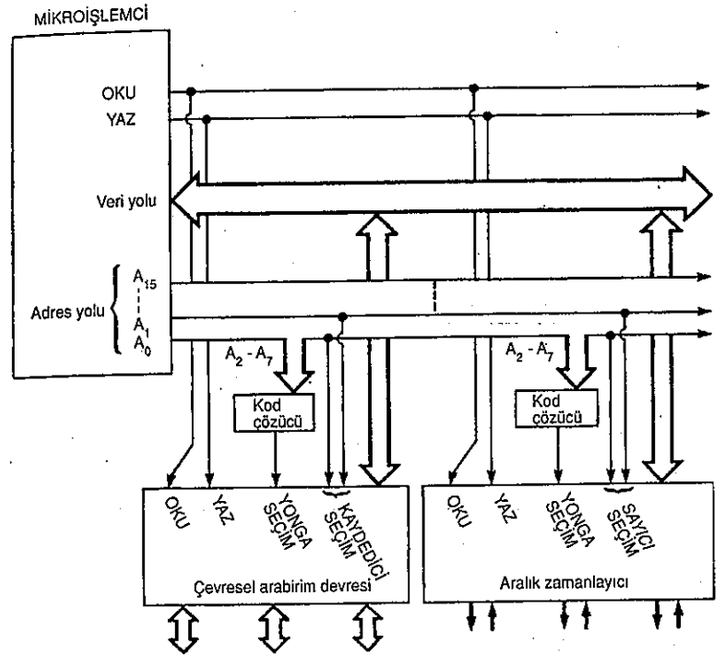
Şekil 2.15 Aralık zamanlayıcıları

Temel olarak aralık zamanlayıcısı bir ya da daha fazla sayıda sayıcı içerir. Şe
2.15'de, tümü benzer biçimde çalışan üç sayıcı gösterilmektedir. Sayıcı 0'ı
alalım. Mikrobilgisayar, programın denetimi altında sayıcıya bir değer yükleye
lir. Bir dış zaman referans kaynağından CLOCK (0) girişine darbe uygulanma
sayıcı içeriğinin azalması sonucunu doğurur. Uygulanan her bir zamanlama darb
ile, saklanan değer bir azaltılır.

Sayıcının içeriği sıfıra ulaştığında, yongadaki mantık devresi OUT(0) hattından
çıkış sinyali üretir. Normalde bu sinyal mikroişlemciyi kesmek için kullanıl
Böylece işlemci, zamanlayıcıyı, hassas biçimde tanımlanmış aralıklarla kesi
sinyalleri üretecek şekilde ayarlayabilir. Bu aralıklar, referans kaynağının frek
sına ve başlangıçta sayıcıya yerleştirilen değere bağlıdır ve dolayısıyla progra
denetimi altındadır.

Aralık zamanlayıcısı, Şekil 2.16'da görüldüğü gibi, çevresel arabirim üzerind
işlemcinin veri adres ve denetim yolları ile bağlanmıştır. Görülebileceği gibi,
birimin bilgisayar arabirimi ile çevresel arabirim devresi arasında yakın bir pa
lellik vardır.

Aralık zamanlayıcısı, mikrobilgisayar veri yoluna, üç-durumlu sürücüler ve alıcı
kullanılarak bağlanmıştır. Belirli bir zamanlayıcının seçimi, adres yolundan ko
çözülerek elde edilen bir sinyalle, birimin 'yonga seçim' girişi kilitlenerek yapılı
Zamanlayıcı yongasındaki iç adreslerin seçimi ise, (örneğin sayıcı 0, sayıcı 1 ve
sayıcı 2'ye bir başlangıç değeri atamak için) 'sayıcı seçim' yonga giriş hatla
kullanılarak yapılır. Şekil 2.16'da, bunların doğrudan A0 ve A1 adres yolu hatlarını
bağlandığı varsayılmıştır.



Şekil 2.16

Daha önce anlatıldığı şekilde ilk kullanıma hazırlanmaları için veriler, sayıcı yerleştirilebilir ya da sayaçtaki değerler bilgisayara okutulabilir. Aktarımın yönü 'oku' ve 'yaz' denetim hatları tarafından belirlenir ve bunlar da sırasıyla programdaki IN veya OUT komutlarıyla işleme sokulurlar. Birden fazla kaydedicinin bulunması, aynı anda ayrı ayrı aralıkların zamanlanmasına olanak tanır.

Bilgisayarın zamanlayıcı-sayıcılarda tutulan değerleri okuyabilmesi, devrenin bir *olay sayıcısı* olarak kullanılabilmesine olanak tanır. Bunda, sayılacak olan olayların ardışık tekrarlarını gösteren sinyal, sayıcılardan birinin saat girişine verilir. Hassas biçimde bir aralık tanımlanabilecek şekilde diğer sayıcı osilatörden alınan bir referans saat girişi ile beslenirse, mikroişlemci, ilgili zaman aralığında tekrarlanan olay sayısını okuyabilecektir. Periyot-tanımlayıcı sayıcıdan her kesme gönderildiğinde, işlemci olay sayıcıyı okur; bir hesaplamanın ardından herhangi bir periyottaki sayıyı bulur.

Yukarıdan da görülebileceği gibi, aralık zamanlayıcısı, bir mikrobilgisayar sistemi için çok önemli bir yardımcıdır. Temel özelliği olan aralık zamanlamasının yanın-

S 2.1, 2.12, 2.13 da, yukarıda tanımlandığı gibi, bir olay sayıcı ya da gerçek-zamanlı saat olara kullanılabilir. Bu sonuncusu, yani *gerçek-zamanlı saatler*, temelde bilgisa gerçek zamanlı olaylarla senkronize etmek üzere zamanlama darbeleri üretir

2.6 Destek Yongaları

Mikrobilgisayar sistemlerinde; disket sürücüleri, görsel görüntüleme birimi y iletişim ağı protokolünde kullanılan denetleyiciler gibi karmaşık birimlerden sürücüleri, kod çözücüler, saat darbe üreteçleri, sayısal görüntüleme birimleri yedi-parçalı kod çözücüler, yol tamponları gibi nisbeten daha basit devrelere k kullanılabilecek pek çok destek yongası bulunmaktadır.

Bunların çoğu, mikrobilgisayarın diğer sistemlere ya da bilgisayarın denetle donanımlara bağlanmasında yardımcı olmak üzere kullanılırlar. Ne var ki, d leri, mikrobilgisayarın tamamlayıcı bir parçasını oluştururlar. Yol sürücüleri tamponları, kod çözücüler ve saat üreteçleri bu ikinci kategoriye girer.

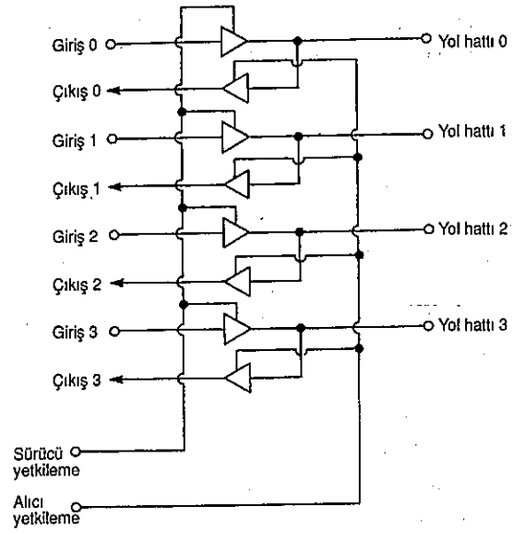
Yol sürücüleri ve yol tamponları

Mikrobilgisayar veri yolu, görmüş olduğumuz gibi, sistem modüllerinin o vasıtasıyla birbirleriyle iletişim kurduğu, iki yönlü bir bilgi yoludur. Üzerine b yerleştirmek için bu yolla bağlantı kurulması, üç-durumlu sürücülerle yapılır.

Yoldan veri almak için de, yine üç-durumlu devreler kullanılır. Mikrobilgisa yolu ile daha kolay bir bağlantı kurulmasına imkan vermesi amacıyla özel ola tasarlanmış ve bir pakette birden fazla (genellikle dört) sürücü/alıcı içeren *alıcı-verici*'leri mevcuttur. Bu türde bir sürücü/alıcının diyagramı Şekil 2.17 görülmektedir.

Bu türde iki yönlü alıcı-vericiler, bir çevresel birim ya da belleği mikrobilgisa yoluna bağlamak için, yol sürücüleri olarak kullanılabilir ya da yol kapasite genişletmek için hatlarına seri olarak yerleştirilebilirler. Mikrobilgisayar yol de tim çıkışlarının sürebileceği kapı sayısı sınırlıdır ve bu nedenle, sistem, daha f bellek ya da çevrebirimi ile genişletileceğinde, genellikle tampon sürücü/alıcı eklemek gerekir.

Şekil 2.17'de görülen sürücü/alıcı, şeklin sol tarafındaki giriş ve çıkışlarla bir sürücü olarak çalışabilir. Böylece, bir birim giriş 0, giriş 1 vb. hatları üzerine



Şekil 2.17

yerleştirebilir ve çıkış 0, çıkış 1 vb.'den verileri alabilir. Şeklin sağ tarafında, mikrobilgisayar anayoluna bağlamak için kullanılan ortak iki yönlü yol görülmektedir.

Anayol üzerindeki verilerin kapılanması, 'sürücü yetkileme' denetim girişi kullanılarak yapılır ve anayoldan alınması ise, 'alıcı yetkileme' denetim hattı tarafından denetlenir.

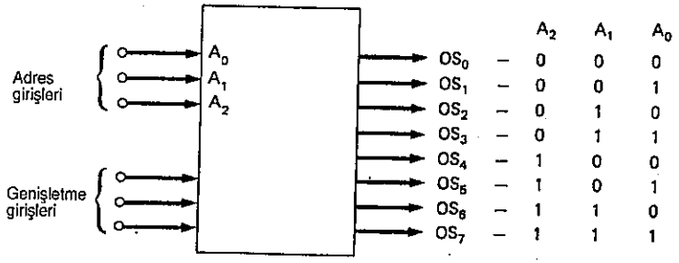
Alıcı ve vericilerin her ikisi de üç durumlu olduğundan, şeklin solundaki giriş ve çıkışlar; giriş 0, çıkış 0'a; giriş 1, çıkış 1'e, vb. biçimde bağlanabilir. Bu, yonganın, yola seri gelecek biçimde, iki yönlü bir yol tamponu olarak kullanılabilmesine imkan verir. Bu tip yol alıcı-vericilerine örnek olarak Intel 8216, Texas Ins. SN 75136 ve Signetics N8T26 birimlerini verebiliriz.

Kod Çözücüler

Adres kod çözücüleri, mikrobilgisayarda birçok yerde gereklidir. Şekil 2.16'da, bunların arabirim devreleri ve aralık zamanlayıcıları için "yonga seçim" sinyalini türetmeleri gösterilmiştir. Kod çözücülere (4. Bölümde de ele alınacağı gibi) bellek sistemlerinin geliştirilmesinde de gerek duyulmaktadır.

Tipik bir kod çözücü yongasına, belli sayıda ikili kodlanmış giriş hattı bağlan
ve yonga, bellek ve G/Ç devrelerin 'yonga seçim' girişlerini sürmek için ay
çıkış 'seçim' hatları üretir. Bu, Şekil 2.18'de şematik olarak gösterilmiştir. Üç
hattı olan A_0, A_1 ve A_2'nin kod çözümü şekilde gösterildiği gibidir. Örneğin,
$A_1 = A_2 = 0$ olduğunda, OS_0 hattı etkinleşir; $A_0 = 1$, $A_1 = 0$ ve $A_2 = 1$ olduğ
da OS_5 etkinleşir, vb. Genişletme girişleri ise, üçten daha fazla sayıda adres gir
kodu çözülmek gerektiğinde, birkaç kod çözücü yongasının birbirine bağlan
mesi için kullanılır.

Birçok sistemde, bellek ve G/Ç devrelerinin 'yonga-seçim' hatları, kendil
etkinleştirmek için alçak düzeye çeken bir sinyale ihtiyaç duyarlar (Bölüm 2.4)
ihtiyacı karşılamak için, mevcut devrelerin kod çözücü çıkışları, etkin olduğu
alçak düzeyine çekilir. Tanımlanan kod çözücü tipi, Intel 8205 entegre devres



Şekil 2.18

Sorular

2.1 Bir mikrobilgisayar sisteminin temel modüllerini sıralayın ve her birin
fonksiyonunu özetleyin. Üretici literatürünü gözden geçirerek, iki ya da dah
fazla modülün gördüğü işlevi bir entegre devrede birleştirip birleştiremeyc
ceğinizi tartışın.

2.2 Mikrobilgisayar sistemleri için bir standart yol seçimiyle elde edilen avanta
ları açıklayın. Bunun herhangi bir dezavantajı var mıdır?

2.3 Bir mikrobilgisayar sisteminin işlemci yongasında sunulan önemli olanaklar
nelerdir? Intel 8085, Zilog Z80 ve Motorola 6809 mikroişlemcilerini, sağla
dıkları olanaklar açısından karşılaştırarak, aralarındaki farkları inceleyin. (Bu
soruyu cevaplamak için, üretici literatürünü incelemeniz gerekmektedir.)

2.4 4. kuşak mikroişlemcileri, 3. kuşak mikroişlemcilerden üstün kılan ek olanaklar nelerdir? Bu olanakların, 4. kuşak entegrelerin olası uygulama alanlarını nasıl etkilediğini tartışın.

2.5 Tipik bir 4. kuşak mikroişlemcide bulunan adres modlarını, her bir modun programda nasıl kullanıldığına da belirterek kısaca açıklayın.

2.6 Aşağıdaki terimleri açıklayın: *a)* dilim kaydedicisi; *b)* statik RAM; *c)* dinamik RAM; *d)* erişim süresi; *e)* kalıcı-olmayan bellekler.

2.7 Mikrobilgisayar sistemlerinde kullanılan çeşitli saklama birimlerinin temel özellikleri nelerdir? Okunur/yazılır ve sadece okunur bellek erişimi arasındaki farkı belirtin ve mevcut bellek yongalarının harcadıkları gücü ve paketleme yoğunluğunu kısaca tartışın.

2.8 Bir arabirim elemanının temel bileşenleri nelerdir? Aşağıda verilen birimlerin işlevlerini açıklayın:
(a) Veri giriş kaydedicisi.
(b) Veri çıkış kaydedicisi.
(c) Durum bayrakları.
(d) Denetim kaydedicisi.
Mikroişlemcinin, yukarıda verilen birimlerle ayrı ayrı nasıl iletişim kurabildiğini açıklayın.

2.9 Programlanabilir çevresel arabirim, çevresel birimleri bir bilgisayara bağlamak için kullanılır. Bu birimin iç kaydedicileri, Bölüm 2.4'de gösterilen kodlara göre yongadaki 'kaydedici seçim' hatları tarafından adreslenirse ve PPI'ın adresi 08 (on altılı) ise istenen işlevi seçmek için, sistemin adres yolunun en düşük değerlikli sekiz hattının nasıl kullanılacağını gösterin. Ek 1'de verilen komut formatını kullanarak, B portu giriş, A portu çıkış ve C portu da anlaşma için kullanılacak şekilde, PPI'ı konfigüre edecek kısa bir program yazın. Şekil 2.13'de gösterilen denetim sözcüğü formatını kullanabilirsiniz.

2.10 Bir mikrobilgisayar sistemi, (00-1F) (on altılı) adres aralığını kaplayan 6 adet PPI ve 2 adet zamanlayıcı/sayıcı'ya sahiptir. Ya bir PPI ya da herbir birim, bir sayıcı-zamanlayıcı olan, ayrı ayrı dört tane adreslenebilir iç kaydedici veya sayıcıya sahiptir. Her bir birimin seçiminin nasıl yapıldığını gösteren bir blok diyagram çizin ve kullanılan adres yapısını açıklayın.

2.11 Programlanabilir bir çevresel arabirim yongasında, veri kaydedicisini *m*'den *n*'e dönüşümü gerçekleştirmek için nasıl kullanılacağını açıklay
Burada:
m: mikrobilgisayar veri yolunun hat sayısı
n: bir çevresel birimle kurulan bağlantının hat sayısıdır.

2.12 Bir aralık zamanlayıcı entegre devresinin yapısını tartışın ve bu tip devrenin, gerçek-zamanlı saat ya da olay sayıcı olarak nasıl kullanılabileceğini gösterin. Bir saat devresi olarak kullanılan bu tür bir çevresel birim mikrobilgisayardaki kristal-denetimli ana sistem saati arasındaki farklılıkl belirtin.

2.13 Yüksek-enerji fiziğindeki bir deneyde, 5 saniyelik aralıklarla meydana gel nükleer 'olayları' saymak gerekmektedir. Bu sayının 100 ile 120 arasınd olacağı beklenmektedir. Sayım, basit bir zamanlayıcı-sayıcı içeren bir mik robilgisayar sistemi tarafından gerçekleştirilecektir. Bu birim, yongadaki bi 'sayıcı seçim' bağlantısınca seçilen iki tane iç 8-bit sayım kaydedicisin sahiptir. Bu hattaki 'yüksek' düzey, sayıcılardan birini, 'alçak' düzey is diğerini seçecektir. Sistemdeki zamanlayıcı-sayıcı adresi 02 (on altılı)'dır v devrede 25 Hz frekanslı bir dış saat darbe üreteci de mevcuttur. İstenen sayım fonksiyonunu gerçekleştirmek için bir program yazın. Sayıların sayıcılar OUT komutu ile yazıldığını ve sayıcılardaki değerlerin IN komutu ile okun duğunu varsayın.

2.14 Bir bant delgi birimi, bir 8255A programlanabilir çevresel arabirim kullanı larak, bilgisayara bağlanacaktır. Delgi, kağıt bandın üzerine 8-bitlik karakter ler yerleştirmektedir. Bir karakteri delmek için hazır olduğunda, birim, 'hazır' olarak adlandırılan çıkış denetim hattına, 'yüksek' mantık düzeyi çıkarmaktadır. Bu durumda, sekiz veri giriş hattına bir karakter çıkarılmalı ve 'veri kullanılabilir' giriş denetim hattı 'yüksek' düzeye çekilmelidir. Delgi, veriyi giriş tamponuna alır ve 'hazır' hattını 'alçak' düzeye çekerek sıfırlar. Eğer 'veri kullanılabilir' hattı üzerinde yüksek düzey varsa, 'hazır' hattı 'yüksek' düzeye çekildiğinde, delgi doğrudan veriyi alır.

Bu arabirimi gerçekleştirmek için bir PPI'ın nasıl kullanılabileceğini tartışın ve PPI'ı konfigüre etmek ve veriyi delgiye aktarmak için kod parçaları halinde bir program yazın.

Bölüm 3 Zamanlama diyagramları ve gerekleri

Bu Bölümün Amaçları: *Bu bölümü bitirdiğinizde:*

1 *Bir mikrobilgisayar sisteminin, bilgi alışverişini merkezi sistem anayolu üzerinden gerçekleştiren çok sayıda karmaşık büyük ölçekli entegre devreden oluştuğunu değerlendirebilmeli,*
2 *(a) işlemci ile bellek yongası ve*
(b) işlemci ile çevresel arabirim elemanları arasındaki veri alışverişinin birkaç aşamada gerçekleştiğini değerlendirebilmeli,
3 *Bir mikrobilgisayarın getir-yürüt komut çevriminin, 'durum' denilen ve her birinde, çevrimin tanımlanan bölümlerinin gerçekleştiği kısa periyotlara bölündüğünü anlayabilmeli,*
4 *Zamanlama diyagramlarının bilginin geçerli, geçersiz ve belirsiz olduğu aralıkları tanımlamak için kullanılabileceği konusuda dahil olmak üzere, zamanlama diyagramları ile zamanlama sinyallerinin kullanımını anlayabilmeli,*
5 *Mikrobilgisayarlarda meydana gelen tüm olayların, kristal denetimli bir osilatör olan sistem saati tarafından denetlendiğini ve farklı mikrobilgisayar sistemlerinin farklı saat gereksinimleri olacağını anlayabilmeli,*
6 *Bellekten okuma veya belleğe yazma saykılı süresince oluşan olayları denetlemek için sistem içinde bulunan çeşitli denetim hatlarının kullanımı da dahil olmak üzere,*
(a) verinin belleğe yazılacağı bir aktarımı ve
(b) verinin bellekten okunacağı bir aktarımı için,
işlemci ve bellek arasındaki veri aktarımının zamanlama gereklerini belirtebilmeli,
7 *İşlemcinin, verinin kullanılabilir hale gelmesini beklemek durumunda olduğu yavaş bellek birimlerine bağlanabilmesi için, mikroişlemci entegresine bağlanan WAIT ya da READY denetim girişlerinin kullanımını anlayabilmeli,*
8 *Bellekten okuma, belleğe yazma ve çevresel aktarım işlemlerine ilişkin zamanlama diyagramları çizebilmeli,*
9 *Adres ya da verinin geçerli olduğu ve daha sonraki kullanımlar için kilitlenebileceği noktaları tanımlamak amacıyla, zamanlama sinyallerinin strob olarak kullanımını anlayabilmeli,*
10 *Tamponlama veya diğer mantıksal işlemlerin, zorunlu olarak, sinyal yolunda bir zaman gecikmesine neden olduğunu ve sistem zamanlaması söz konusu olduğunda, bu tür gecikmelerin hesaba katılması gerektiğini değerlendirebilmelisiniz.*

3.1 Niçin Saat Darbeleri Kullanılır?

Saat sinyallerine, mikrobilgisayarlarda, sistemi oluşturan çeşitli birimlerin (iş
ci, bellekler, G/Ç, vb.) faaliyetlerini senkronize edecek şekilde, sistem zamanl
sını denetlemek için gerek duyulur. Böyle bir senkronizasyona gerek duyulm
birkaç nedeni vardır.

Sistemin yapısı ve çalışması

Bir mikrobilgisayar sisteminin, birbirine anayollar ya da yollarla bağlı belli sa
büyük-ölçekli entegre devreden meydana geldiğini görmüştük. Bu devreler, k
içlerinde çeşitli karmaşık mantıksal fonksiyonlara ve bilgi aktarımı için iç
yollarına sahiptir. Örneğin, işlemci yongası, program yürütülmesi esnasında yo
nin aralarında sürekli gidip geldiği, iç kaydedicilere ve bir aritmetik mantık biri
sahiptir (Şekil 2.3). Bunlardan başka, işlemci, yürütülen komuta ilişkin işl
kodunun, o komuttaki işlenenlerin ve bir sonraki komutun bellek adresinin, işlem
durumunun, vb. tutulduğu kaydediciler içerir.

Bu nedenle, mikroişlemci yongası, bir iç yol üzerinden bilgi alışverişi yapa
birbirine bağlı mantıksal birimlerden oluşmaktadır ve mikrobilgisayar, bilgisay
anayolu üzerinden bilgi alışverişi yapan bu tür bir yongalar topluluğudur.

Makine içindeki bir komutu yürütmek için, işlemcideki denetim devreleri şu
işlevleri yerine getirmek için gereklidir:

1 Komutu program belleğinden almak.
2 Bu komutu, komut kaydedicisine aktarmak.
3 Gerçekleştirilecek işlemi belirlemek için komutun kodunu çözmek.
4 Komutun işlenen ya da işlenenlerini elde etmek ve
5 Komutu yürütmek.

Bu işlemlerin her biri birkaç adımdan oluşur. Örneğin, bellekten bir işlenen almak
belleğe bir adres göndermek, belleği okuma konumuna almak ve çıkış verisini
anayol aracılığıyla, bellekten işlemciye aktarmak gerekir.

Bu ifadeden de anlaşıldığı gibi, mikrobilgisayar, program yürütmesi süresinin
büyük bir bölümünü veri aktarma işlemlerinde harcar. Bununla beraber, herhangi
bir programda var olan mantık, aritmetik ve test fonksiyonlarını da gerçekleştirir.
Bunların çoğu, birkaç ara adımın yürütülmesinde işlemcinin müdahalesini ge-
rektirir ve tüm bu işlemler, doğru zaman sırasında gerçekleştirilmelidir.

Kaydediciden-akümülatöre ADD komutunu gerçekleştirmek için, örneğin (Kısım
6) işlemci:

1 İşlenenleri, seçilen kaydedici ve akümülatörden, toplayıcının girişine uygula-
malı,
2 Toplayıcıya, toplama işlemini gerçekleştirmesi için bir sinyal göndermeli,
3 Toplayıcı çıkışındaki sonucun, tekrar akümülatöre yazılmasını sağlamalıdır.

Mikrobilgisayardaki işlemlerin doğru sırada oluşması, ister bu türde işlemler sistem
modülleri arasındaki veri aktarımları, isterse işlemcinin kendi içindeki faaliyetlerin
bir denetimi olsun, sistemde, *zamanlama* ya da *saat* darbelerinin kullanılmasını
gerektirir.

Anayollar ve Zamanlama

Bilgisayar sisteminde bilgi iletimi için, veri yolları ile daha önce sözü geçen
anayolların hazır olması gerekir. Bu anayolların gerçekleştirilebileceği, sistemin
çalışması üzerinde oldukça etkisi bulunan birkaç yol vardır.

Örneğin, Kısım 2.2'de ana hatlarıyla verildiği gibi, ortak bir veri ve adres anayolu,
tek bir komutun yürütümü esnasında, ilgili veri ve adresin *her ikisini* iletmek için
yolun kullanılmasını gerektirir ve bu da, söz konusu komutu işleme koymak için
gereken süreyi artırır. Benzer biçimde, 16-bitlik sistemlerde 8-bit genişliğindeki
bir veri yolunun kullanımı, tek bir 16 bitlik sözcüğü iletmek için, bellek ile işlemci
arasında, art arda iki donanımsal aktarım gerektirir. Yine bu da, komut yürütme
süresini artırır.

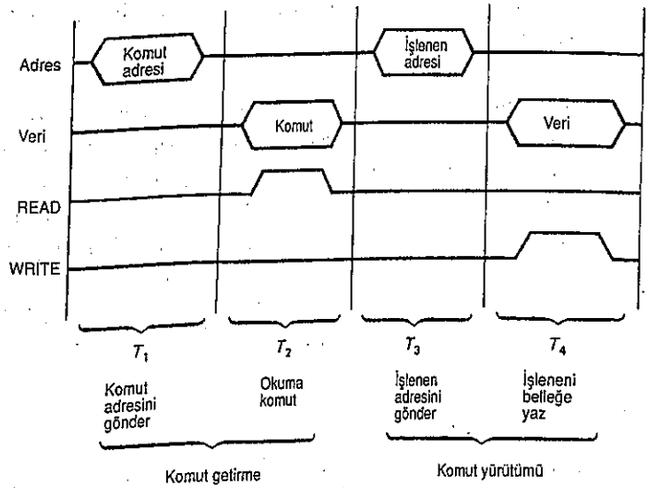
8-bitlik bir mikrobilgisayar sisteminde, tek-baytlık komut, komut getirme fazı
sırasında bellekten işlemciye tek bir aktarım gerektirir. Aynı şekilde, 2-baytlık bir
komut 2 aktarıma, 3-baytlık komut da 3 aktarıma gerek duyar. Bunların da doğru
bir şekilde sıralanması gerekir ve her biri, özellikle sistem yolu ortaksa, sistem
yolunda bir kaç periyot sürer.

Böylece ortaya sistemin genel görünümü, yani sistemde ne olup bittiğini tanımla-
mak ve işlemleri doğru sırasında sıralamak üzere kullanılan ayrık zaman birimle-
rinde çalışan mikrobilgisayar çıkar. CPU yongasının dışındaki bir bellekten alınan
her mikrobilgisayar getirme komutu, her bir komutun yürütülmesi için bu zaman
aralıklarından birkaçına ihtiyaç duyar. Bu sayının tam olarak değeri değişkendir
bazıları önceki örneklerde bahis konusu sözü edilmiş olan, çok sayıda parametreye
S 3.1, 3.2 bağlıdır.

Durumlar

Mikroişlemci sistemlerde, temel komut getir-yürüt süresini, *durum* olarak adlandırılan daha küçük periyotlara bölmek uygun olacaktır. Buradaki periyotlarından her biri bir adresin belleğe ya da bir çevresel arabirim devresine iletimi, yoldaki veri bellek ya da çevrebirimlerine iletimi ve benzeri tanımlanmış işlemlerden birinin gerçekleşmesine imkan sağlar. Gereken durum sayısı, farklı mimarileri nedeniyle bir mikrobilgisayardan diğerine ve bir mikrobilgisayar içerisinde de bir komuttan diğerine değişmektedir.

Durumların kullanımına ilişkin bir örnek, Şekil 3.1.'deki *zamanlama diyagramı* görülmektedir. Diyagram, mikrobilgisayar sistemindeki çeşitli hatlardaki sinyalleri ve özellikle, sinyallerin ne zaman *geçerli* olduğunu, yani sistem tarafından ne zaman kullanılabileceklerini gösterir. Şeklin yatay ekseni zamana göre değişimi gösterir. Bu konu, bir sonraki kısımda daha ayrıntılı olarak tartışılacaktır.



Şekil 3.1.

Şekil 3.1'de, bir işleneni belleğe yazmak için gereken bir baytlık bir komutun yürütülmesi gösterilmektedir. T_1, T_2, T_3 ve T_4 olarak adlandırılan dört faz vardır. Bu fazlarda meydana gelen işlemler aşağıdaki gibidir:

T_1 süresince, komut adresi belleğe gönderilir.
T_2 süresince, komut getirilir.
T_3 süresince, işlenen adresi belleğe gönderilir.
T_4 süresince, işlenen o adresteki bellek konumuna yazılır.

Şekilde ayrıca, bir önceki bölümde de anlatıldığı gibi, veri iletiminin yönünü tanımlayan READ (OKU) ve WRITE (YAZ) sinyalleri de gösterilmiştir. Bu sinyaller işlemci tarafından üretilir. Bir işlemci girişi yapıldığında, READ hattı, çıkış yapıldığında WRITE hattı mantıksal 1'e gider.

Adres ve veri hatlarındaki sinyaller, adres ve verinin her ikisi de sıfır ve birler içerdiği için, ya 1 ya da 0'dır. Bu nedenle, yani sinyaller hem mantıksal 1 ve hem de 0 değerlerini alabildiklerinden, şekilde bu sinyallerin gösterimi kutu biçiminde verilmiştir.

3.2 Zamanlama diyagramları ve zamanlama sinyalleri

Bu aşamada, (Şekil 3.1'deki gibi) zamanlama diyagramlarının anlamını biraz daha ayrıntılı bir biçimde tartışmak yararlı olacaktır. Bu tür diyagramların, mikrobilgisayar sistemindeki sinyalleri arasındaki zamanlama ilişkilerini göstermek ve bir sinyalin ne zaman geçerli olduğunu tanımlamak için kullanıldığından daha önce söz edilmişti.

Şimdi bu konu biraz daha ayrıntılı bir biçimde ele alınacaktır. Komut adresinin geçerli olduğu tam aralık, "komut adresi" kutusunun uzunluğu ile gösterilir. Adres anayollarındaki adresleri gösteren sinyallerin yükselme ve düşme zamanları da benzer biçimde, bu kutunun eğimli kenarlarıyla gösterilir. Bu nedenle, adres bilgisini doğru olarak algılayabilmek için, o bilgiye ihtiyaç duyan herhangi bir birim, yükselme periyodu ile düşme periyodu arasındaki bir anda, yani Şekil 3.1'deki "komut adresi" kutusunun merkezine yakın bir yerde, adres hatlarını gözden geçirmelidir. Benzer yolla, komutu gösteren veri, veri anayolunda Şekil 3.1'deki "komut" kutusu ile işaretlenmiş periyot süresince geçerlidir, yani kullanılabilir durumdadır.

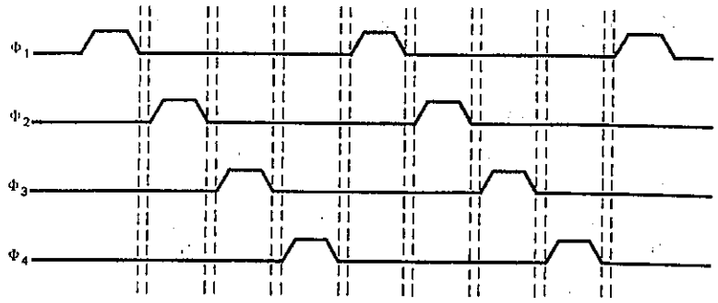
Mikrobilgisayardaki veri iletiminin gerçekleştirileceği hassas zaman aralıkları, sistemdeki *zamanlama sinyalleri* tarafından belirlenir. Buna ilişkin örnekleri ilerki bölümlerde göreceksiniz.

3.3 Saat Darbe Üreteci

Mikrobilgisayarlarda gerçekleşen tüm işlemler, bir ana *saat osilatörü*nce senkronize edilir ve zamanlanırlar. Bu aygıt, CPU yongasının bacaklarına gidecek çıkış darbeleri üretir ve işlemci yongasının adı geçen temel periyotları oluşturmasında kullanılır.

Saat osilatörü ya da saat *üreteci*, genellikle kristal-denetimli olup, Şekil 1.5 2.1'de gösterildiği gibi, mikrobilgisayar sistemlerinde ayrı bir yonga olarak bul nabilir. Alternatif olarak, işlemci yongasının kendisi de saat devreleri içerebil Örneğin, Z80, 8080 ve TMS 9900 mikroişlemci yongalarının tümü, dış bir osil törden sağlanacak saat sinyallerine gerek duyarlar. Diğer taraftan ise, 8085 ve TM 9940, işlemci içinde saat üretim devrelerine sahiptir.

Saat üreteci içeren işlemciler, yalnızca, işlemci yongasındaki bacaklara bağlan uygun bir frekans belirleyici devreye ihtiyaç duyarlar. Bu amaçla, bir kristal ya bir bobin/kondansatör devresi kullanılabilir. Bu sistemlerdeki işlemci entegre ayrıca, dış donanımla birlikte kullanmak üzere senkronizasyon sinyalleri sağlay cak bir saat çıkışına da sahiptir.

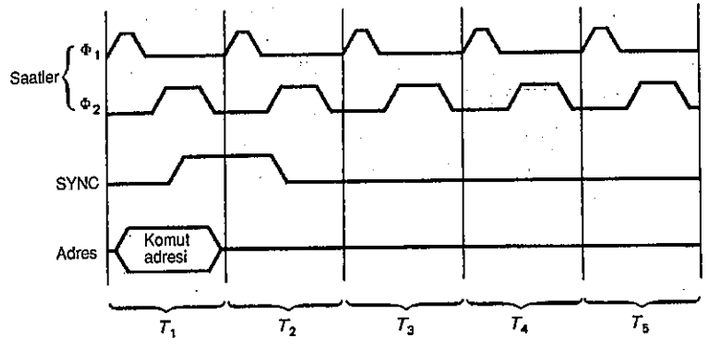


Şekil 3.2

Farklı mikroişlemcilerin saat darbesi gerekleri farklıdır. Örneğin, Texas TMS 9900 Şekil 3.2'de gösterildiği gibi, dört-fazlı bir saat sinyaline gerek duyar. Diyagramdan da görülebileceği gibi, zaman ekseninde çakışmayacak bir şekilde düzenlenmiş dört ayrı darbe dizisi vardır. TMS 9900 işlemcisi bir saat üretici içermediğinden Φ_1, Φ_2, Φ_3 ve Φ_4 darbe dizileri, işlemci yongasındaki dört ayrı bacağa verilir. Darbeler üst üste binmemelidir; bu nedenle, istenen darbe genişliği ile yükselme ve düşme zamanlarını belirten tanımlamalar bulunmaktadır. Saat frekansı 2 ile 3 MHz arasında değişebilir. Bir TMS 9900 sisteminde bu darbelerin üretimi, bir TIM 9904 tek yongalı saat sinyali üreteci ve sürücü kullanılarak gerçekleştirilebilir.

Intel 8080A'nın iki fazlı bir saat sinyaline gereksinimi vardır. İşlemci yongasında, bu fazların girişi ve saat üretici devresi için iki bacak bulunmaktadır. 8224 saat

üreteci ve sürücü, gereken sinyalleri üretmek için kullanılır. Şekil 3.3'de, bu sinyallerle işlemcinin durumları arasındaki ilişki gösterilmiştir. Her bir durumun, her bir saat darbesinin baştan sona tümünü işgal ettiği görülebilir. İşlemci, bir çıkış bacağından, diyagramda da gösterilen, her bir çevrimin başlangıcını tanımlamakta kullanılan bir SYNC sinyali çıkarır.



Şekil 3.3

Bazı mikroişlemcilerin yalnızca tek fazlı bir giriş saatine gereksinimi vardır, yani saat üreteci sadece bir tek darbe dizisi üretir. Zilog Z80 ve Z8000, Intel 8086 ve Motorola 68000, bunlara örnektir.

3.4 Bellek Erişim-Süresi Gerekleri

Bir bellekten veri almak için harcanan süre olarak tanımlanan bellek erişim-süresini gösteren bazı şekiller Kısım 2.3'de verilmişti. Daha doğru bir tanımla erişim-süresi, bilgileri almak için bellekten istek yapıldıktan sonra, bilgi kullanılır hale gelene kadar geçen süredir.

Kısım 2.3'deki tablolarda da gösterildiği gibi, çeşitli belleklerin erişim süreleri geniş bir aralıkta değişir. Örneğin, küçük bir statik RAM 35 nsn'lik bir erişim hızı ile çalışırken, dinamik RAM'lar yaklaşık 500 nsn'ye kadar bir erişim süresine sahiptirler. EPROM'ların erişim süreleri de genellikle 450 nsn civarındadır.

Uzun bellek erişim sürelerinin etkisi

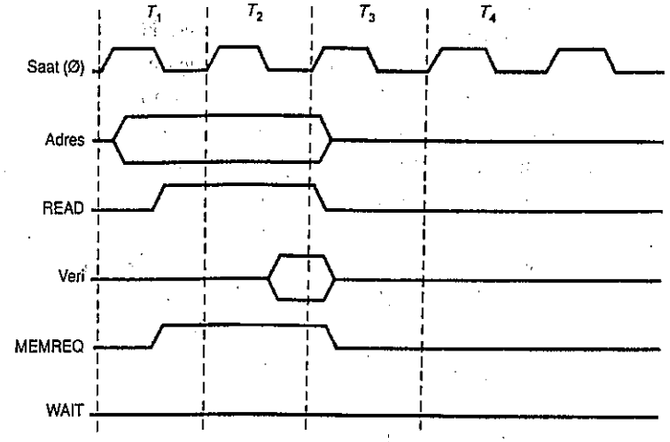
İşlemci yongasının farklı hızlara sahip belleklerle birlikte çalışabilme yeter
sahip olması gerekir ve dolayısıyla, işlemcinin *yavaş* belleklerde de çalışabi
bir yöntem belirlemek gerekmektedir. Erişim süresinin, işlemcinin saat çevr
den az olduğu hızlı bellekler genellikle herhangi bir sorun oluşturmazlar.

Yavaş belleklerin kullanımında bazı zorluklar ortaya çıkar, çünkü işlemci
işlemci durumu süresinde, örneğin T_1 süresinde, veriyi okumak için belleğe
gönderir ve bir süre sonra, örneğin Şekil 3.1'de gösterildiği gibi, T_2 süresinde ve
okur. Bellek, işlemci tarafından ihtiyaç duyulmadan önce mevcut veriyi al
kadar hızlı olduğu sürece sorun yoktur. Belleğin erişim süresi uzun ise ve
nedenle, veriye erişim süresinde birkaç işlemci fazının geçmesi gerekiyorsa, b
özel işlem adımları gerekecektir.

Bekleme durumları (Wait states)

Temelde işlemci, böyle bir durumda, veri kullanılabilir olana kadar bekletilmeli
Bu, işlemci saykıl içine WAIT durumlarının yerleştirilmesiyle gerçekleştirilir.

Bir örnek bunu daha iyi açıklayacaktır. *Ayrı* bir 16-bitlik adres ve 8-bitlik veri yo
olan bir mikrobilgisayar için, normal bir komut getirme saykılını ele alalım. Bir
tek fazlı bir saat sistemi kullanıyorsa, durum, Şekil 3.4 deki gibi olabilecektir.



Şekil 3.4

Şekildeki zamanlama sinyalleri üzerinde küçük bir değişiklik yapılmıştır. Daha
önce (Şekil 3.3) olduğu gibi, istenen bellek konumunun adresi, T_1 süresinde,
bilgisayar anayolunun adres hatlarına kapılanır. MEMREQ (bellek isteği) sinyali-
nin yükselen kenarı, adresin ne zaman geçerli olduğunu gösterir. Bu kenar, adres
anayola çıkarıldıktan ve yerleşmesi için bir zaman tanındıktan sonra yaklaşık bir
yarım saat saykılı sürer.

T_1'in ardından, işlemci, T_2 süresince, bellekten alınacak verinin kullanılabilir hale
gelmesini *bekler*. Bu periyot süresince, işlemci bellekten veri almaya çalışmaz;
ancak, veri yolunda geçerli bilginin kullanılabilir durumda olup olmadığını işlem-
ciye göstermek için kullanılan WAIT (BEKLE) giriş denetim hattını izler.

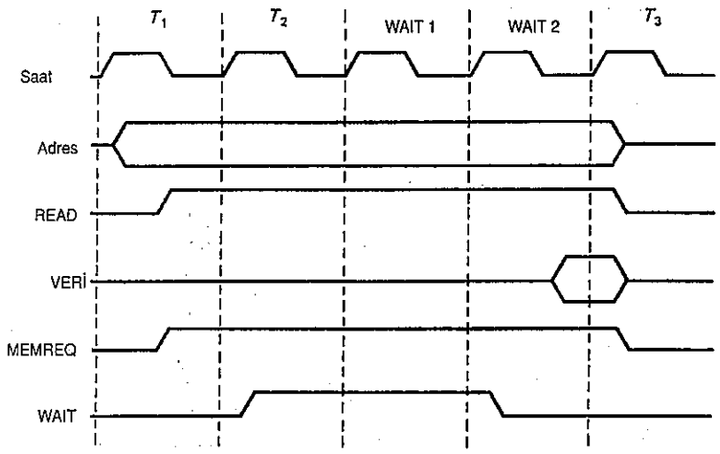
Eğer işlemci hattı gözden geçirirken WAIT etkin değilse, yani şekil 3.4'de olduğu
gibi, 'alçak' düzeyde ise, belleğin veriye erişim hızı yeterlidir. Ancak, hattın gözden
geçirilmesi sırasında WAIT hattı etkinse, işlemci, belleğe işlemini tamamlayabil-
mesi için gereken zamanı vermek üzere, kendi işlemine bir süre ara vermelidir.

Sistemin tam olarak çalışması şöyledir. İşlemci, T_2 periyodunda, saat darbesinin
düşen kenarınca tanımlanan anda, WAIT hattındaki sinyali algılar. Bu anda (Şekil
3.4 deki gibi) WAIT *etkin değilse*, belleğin hızlı olduğu ve T_2 aralığı süresince,
istenen veriye erişip onu veri anayoluna koyabileceği varsayılır. Ardından, T_3
aralığında, saat darbesinin yükselen kenarında, yani T_3 durumunun başlangıcında,
veri işlemciye alınır. İşlemcinin içinde, bir komutu gösteren veri, kodunun çözül-
düğü yer olan ve yürütme saykılını başlatmak için kullanılan komut kaydedicisine
yerleştirilir.

Burada tanımlanan işlem sırası, bu sistemle gerçekleştirilebilecek en hızlı komut
saykılını gösterir. Komutu bellekten getirmek için en az üç faza gereksinim vardır.
Bununla birlikte, eğer bellek yavaş ise saykıl, araya WAIT fazlarının konulmasıyla
uzatılabilir. Şekil 3.5, bir başka komut getirimin saykılının zamanlama diyagramını
göstermektedir, fakat bu kez belleğin, T_2 süresinde veriye erişebilmesi mümkün
olmayan yavaş bir bellek olduğu varsayılmıştır.

Yine, istenen bilginin, yani erişilecek komutun adresi, T_1 fazı aralığında adres
anayoluna yerleştirilmiştir. Daha önce olduğu gibi, MEMREQ, bu adresin geçerli
olduğunu göstermek için işlemci tarafından etkin düzeye çekilir. Ve yine işlemci,
WAIT hattını, T_2 aralığında, saat darbesinin düşen kenarının meydana geldiği anda
denetler.

Şekil 3.5'de, T_2 periyodunda, saatin düşen kenarından önce, WAIT hattının etkin



Şekil 3.5

olduğu varsayılmıştır. Nitekim bakılırsa hattın, etkin (mantıksal 1) düzeyde oldu görülür. Eğer böyleyse, işlemci saykla fazladan bir *wait fazı* ekler (WAIT 1, Şek 3.5), yani daha önce yaptığı gibi, veri hatlarını bir sonraki saat darbesinin yüksel kenarında okumayıp bunun yerine, WAIT 1 saat darbesinin düşen kenarına kad hiçbir şey yapmayarak bekler.

WAIT 1'in düşen kenarı süresince işlemci yine WAIT sinyalini denetler. Verile örnekte, bu sinyal hâlâ etkin düzeydedir. İşlemci bu duruma karşılık olarak, ara 2. bir WAIT fazı koyar (WAIT 2, Şekil 3.5).

WAIT 2'nin düşen kenarı boyunca, WAIT sinyali bir kere daha gözden geçiril Bu sefer, WAIT 2'nin ilk yarısında mantıksal 0'a döndüğünden etkin değildir. B durumda, işlemci bir sonraki (T_3) duruma geçer ve veriyi, daha önce olduğu gib T_3 saat darbesinin yükselen kenarında okur.

Buradan, WAIT sinyalinin etkisi kolayca anlaşılabilmektedir. İşlemcinin WA sinyalinin etkin olduğunu her algılayışında, ele alınan komut okuma saykılı, ya d daha doğrusu, sözgelimi, bir komutu veya işleneni elde etmek için herhangi b bellek erişimi sırasında, işlemci araya, belleğin çalışmasını tamamlamasını bekle yeceği bir saat saykılı yerleştirir. Bu saat saykılı esnasında, WAIT sinyalini gözde geçirir ve eğer etkin ise, araya tekrar bir WAIT fazı ekler. Bu faz boyunca, WA

sinyalini gözden geçirir ve işlem böylece sürer. İşlemcinin çok yavaş bir bellekle çalışması gerekirse, kullanılan bekleme saat saykllarının sayısı süresiz olarak artırılabilir. Bununla birlikte, çok fazla WAIT aralığı kullanıldığında, işlemcinin komut yürütme hızı oldukça düşeceğinden, bu işlem, sistemi çalıştırmak için istenen bir durum değildir.

Uygulamada, işlemciye gelen WAIT denetim girişlerinin kullanımı yalnızca bellek aktarımlarıyla sınırlı değildir. Bu bölümde daha önce, işlemci ve bellek arasındaki senkronizasyon kapsamında tanımlanmış olmasına karşın bunlar çevresel aktarımlarda da kullanılabilir. Bu nedenle, eğer belirli bir uygulamada böyle bir uygulama yapılması gerekecekse, Bölüm 2'de tanımlanan G/Ç yonga tipleriyle bağlanmasına izin vermek için işlemcinin çalışmasını yavaşlatmak mümkündür.

Sözü edilmeye değer diğer bir nokta ise, mikrobilgisayar sistemleriyle ilgili pek çok konuda olduğu gibi burada da, farklı üreticilerin benzer fonksiyonlar için farklı adlar kullanıyor olmasıdır. Özellikle, WAIT işlemci-denetim girişi yerine, buna eşdeğer başka bir alternatif konabilir. Örneğin, bazı mikroişlemciler READY denilen bir denetim girişine sahiptirler. Bunun gördüğü işlev aslında WAIT'e çok benzer; tek fark, yalnızca tersi olmasıdır (işlemcinin beklemesi (WAIT) gerekiyorsa, bu bellek ya da çevrebiriminin *hazır olmadığı* içindir). Gördüğü işlev aynıdır, yani işlemciyi, kendisine bağlı daha yavaş devrelerle senkronize etmektedir.

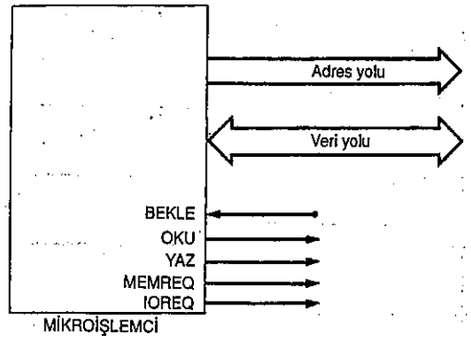
3.5 Bellek aktarım işlemlerinde kullanılan denetim sinyalleri

Önceki bölümlerde işlemcinin, mikrobilgisayar belleğinden veri alabildiği görülmüş ve mikrobilgisayar sisteminde yer alan bu iki birimin eşzamanlanmasına ilişkin bazı ayrıntılar ele alınmıştı. Şimdi, bellekten bilgi okuma ve belleğe bilgi yazma işlemlerini daha geniş bir biçimde ele alacak ve şu ana kadar bu konularla ilgili ayrı ayrı verilmiş olan çeşitli bilgileri bir araya getireceğiz.

Mikroişlemci yongasının blok diyagramı Şekil 3.6'da verilmiştir. Bu diyagram, bellek aktarımında kullanılan denetim sinyallerini belirtir.

Bu sinyallerin çoğu daha önce açıklanmıştı, ancak işlevlerini bir kez daha özetlemek uygun olacaktır:

1 Adres yolu, bilgi yazmak ya da okumak üzere erişilecek bellek konumlarının adreslerini taşır.
2 Veri yolu, belleğe yerleştirilecek ya da bellekten alınacak olan verileri taşır.



Şekil 3.6

3 READ ve WRITE sinyalleri, daha önce tanımlandığı gibi, bir aktarımın yönü göstermek için kullanılırlar.
4 MEMREQ ve IOREQ sinyalleri, istenen aktarım tipini göstermek için işlem tarafından kullanılır. Aktarım; bellekle bir veri değiş-tokusu ise, MEMREQ bellek adresinin geçerli olduğu zamanı (Kısım 3.4), bir G/Ç değiş-tokusu is IOREQ (G/Ç isteği) (genellikle en küçük değerlikli 8 adres hattındaki) çevre birimi adresinin geçerli olduğu zamanı tanımlar.

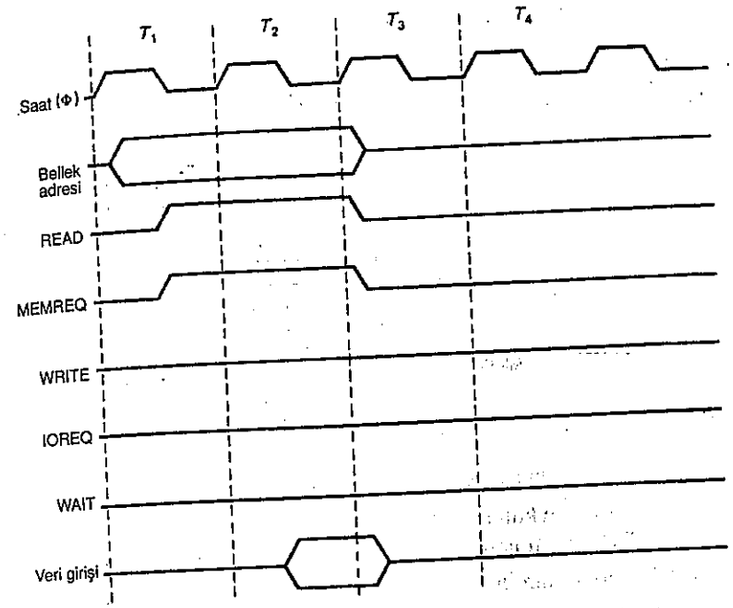
Bu nedenle bir bellek konumu; adresin adres yoluna çıkarılması, MEMREQ (Bellek isteği) geçerli sinyalini taşıması ve READ sinyalinin geçerli olması durumunda okunur.

Bir bellek konumuna veri; adres, MEMREQ ve WRITE sinyallerinin geçerli olması durumunda yazılır.

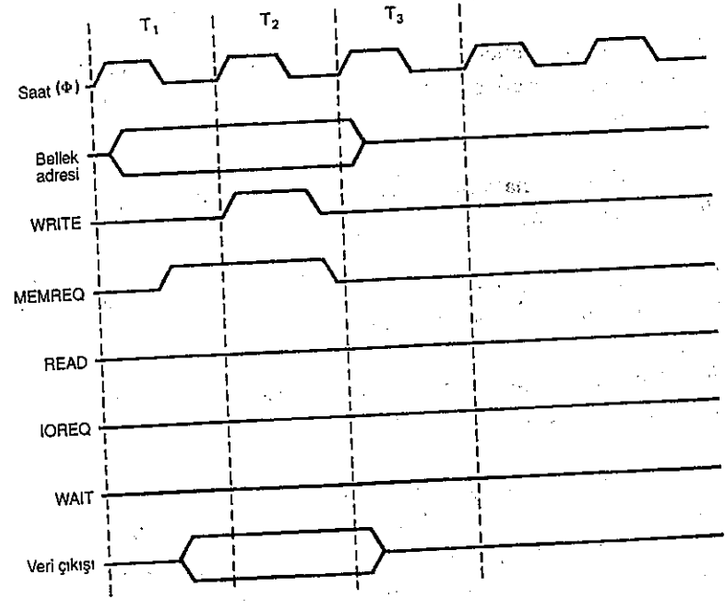
Bir çevresel birimden işlemciye bilgi; o birimin adresi, IOREQ ve READ sinyallerinin geçerli olması durumunda yazılır.

Son olarak da, veriler çevresel birimlere; birim adresi, IOREQ ve WRITE sinyallerinin tümünün geçerli olması durumunda yazılır.

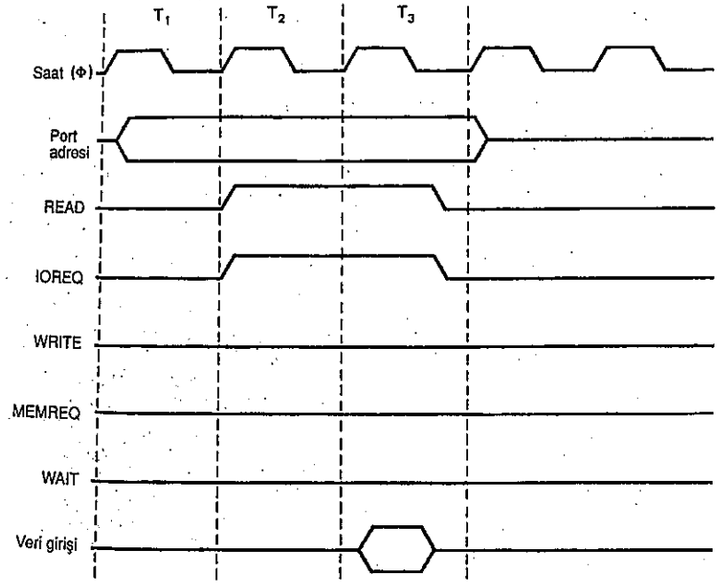
Bellekten okuma, belleğe yazma ve G/Ç işlemlerini gösteren zamanlama diyagramları sırasıyla Şekil 3.7, 3.8 ve 3.9'da verilmiştir. Bu diyagramlarda, belleğin (Şekil 3.7 ve 3.8) ve çevrebiriminin (Şekil 3.9) araya fazladan WAIT durumları yerleştirmeksizin veri almada ya da iletmede yeterince hızlı olduğu varsayılmıştır. Ancak WAIT durumları, daha önce de anlatıldığı gibi, gerektiğinde araya eklenirler.



Şekil 3.7 Bellekten okuma işlemi



Şekil 3.8 Belleğe yazma işlemi



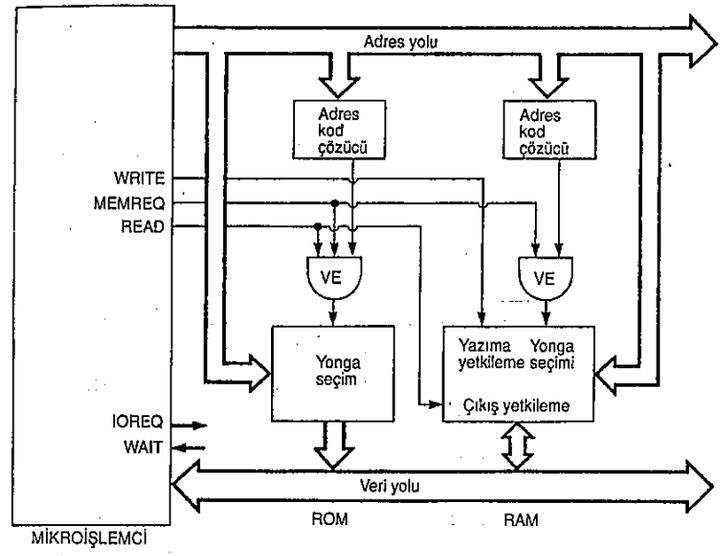
Şekil 3.9 Bir G/Ç işlemi

Belleğe yazma işleminde, belleğe yerleştirilecek veri, WRITE darbesi etkinleştiril diğinde, zaman açısından kararlı olmasını sağlamak üzere, işlemci tarafında mümkün olduğunca saykılın başında ayarlanır. Bu darbe süresince veri belleğ alınır.

Çevresel aktarım işleminin (Şekil 3.9), önceki şekillerde verilen bellek işlemlerin den bir saat periyodu daha uzun sürdüğü gösterilmiştir. Yükselen kenarının, geçer saykıl adresinin adres yolunda olduğunu belirttiği IOREQ sinyali, bir önceki belle işlemlerinde, MEMREQ sinyalinden yarım saykıl sonra ortaya çıkar. Bu, gere duyulduğunda, adres sinyallerinin kararlılığını sağlamak üzere daha uzun sü tanır.

Sistem blok diyagramı

Mikroişlemci yongasıyla RAM ve ROM arasındaki bağlantılar, Şekil 3.10'd gösterilmiştir. En düşük değerlikli adres-yolu hatları, bir bellek bloğundaki eriş lecek konumları seçmek için kullanılır. Örneğin, şekildeki ROM 1.024 baytlık bi saklama alanına sahip ise, en küçük değerlikli 10 adres-yolu hattı (A_0-A_9 hatları) bellek adres girişlerine bağlanacaktır (burada adreslerin, çoğullanmamış belle



Şekil 3.10

adresleri olduğu varsayılmıştır, bkz: Kısım 2.2). ROM yongasını seçmek için, MEMREQ ve READ sinyalleri etkin (yüksek) olduğunda, sıralı adres bitlerinin kodu çözülür ve bu bitler kullanılır. READ, yüksek düzeyde olmadığında, yani, örneğin belleğe yazma işlemi yapılacağı bir durumda, ROM seçilmez.

RAM' deki konumlar; yüksek sıralı adres bitleri, adres çözücüden bir çıkış üretmeye uygun olduklarında ve MEMREQ 'yüksek' düzeyde olduğunda seçilirler. Bu nedenle, bir 'okuma' ya da 'yazma' işleminin hangisinin yapılacağına bakılmaksızın RAM seçilir. O zaman WRITE sinyali, verinin seçilen adrese yerleştirilmesine izin verecek olan, RAM yongasındaki ayrı bir 'Write enable' (Yazma yetkileme) girişini etkinleştirmek için kullanılır.

İster okuma ve ister yazma işlemi için olsun, bir RAM yongası yetkilendiğinde, seçilen konumdaki veri okunur. Bununla birlikte, bu veri, 'yazma' işlemi sırasında değil, sadece bir 'okuma' işlemi gerçekleştirileceğinde, mikroişlemcinin veri yoluna yerleştirilmelidir. 'Yazma' işleminde yol, belleğe saklanacak veriyi aktarmak için zaten kullanılmaktadır ve bu bilgi zarar görmemelidir. Bu nedenle, işlemcideki READ sinyali RAM'daki 'çıkış yetkileme' denetim girişini etkinleştirmek için kullanılır (Şekil 3.10). Bu girişe, yüksek düzeyli READ sinyali verildiğinde ve yonga seçildiğinde, RAM çıkış verisi veri yoluna bağlanır. Ancak, bu giriş alçak

düzeyde olduğunda, bir bellekten okuma işleminin
daki herhangi bir işlem sırasında, RAM çıkışı yoldan
edilir.

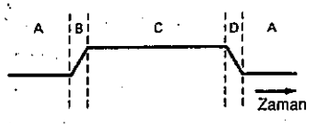
Çoğu sistemlerde, dış mantık devresine, yani şekil 3.
gösterilen iki 'VE' kapısına ihtiyaç duyulmaz. Bununy
bellek yongalarında, iç mantık devreleri ve yonganın
sinde 'VE'lenen iki ya da daha fazla 'yonga seçim'
vardır.

Denetim hatlarını ifade etmek için kullanılan adlar siste
sisteme değişir. Genellikle, Kısım 2.4'deki arab
cihazlarında tanımlanmış olan, alçak düzeyde etkin olan
yaller kullanılır. Daha önce geçen adlar kullanılarak, der
sinyalleri $\overline{\text{READ}}$, $\overline{\text{WRITE}}$, $\overline{\text{MEMREQ}}$, $\overline{\text{IOREQ}}$ vb. ol
adlandırılır. Bu değişiklikler, sistemin çalışma ilkelerini al
te değiştirmez, ancak zamanlama diyagramlarındaki sinya
rin alçakta-etkin niteliği tersine döner. Bir sinyalin geçerl
periyodunu göstermek için, negatife-giden bir kenar değiş
S 3.9 kullanılır.

3.6 Aktarım periyodunu tanımlamak için stroblama (Işıldaklama)

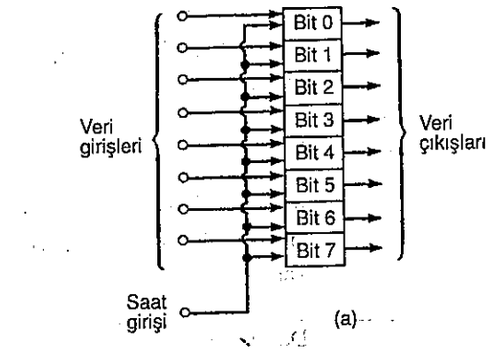
Bellekten okuma ve belleğe yazma işlemlerinde, denetim
yalleri kullanıldığı için, bu sinyaller arasındaki zamansal il
kileri ve zamanlama diyagramlarını kısaca bir gözd
geçirmek uygun olacaktır.

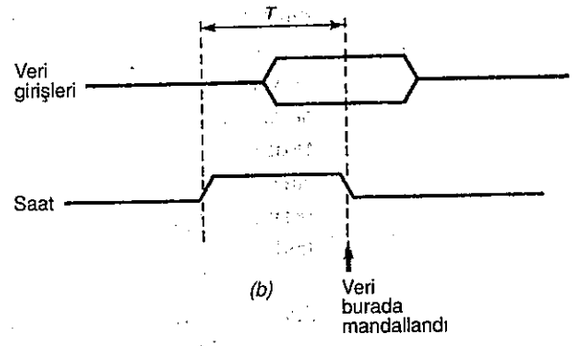
Genelde bir zamanlama diyagramı, sistemdeki bir sinyalin
fazını tanımlar. Sinyal; *geçerli*, *geçersiz* ya da sinyalin geçe
veya geçersiz olup olmadığına bağlı olarak *belirsiz* olabil
Bu, bir örnek yardımıyla daha iyi açıklanabilir.



Şekil 3.11 Bir G/Ç işlemi

Şekil 3.11'de, daha önce kullanılan tipik bir sinyal gösteri
mektedir. A olarak adlandırılan aralıklarda, sinyal *alçak* d
rumdadır. C aralığında ise, *yüksek* düzeydedir. B ve
periyotlarında ise, alçak ile yüksek arasında bir ara düzeydedi
B ve D aralıklarında ise sinyal belirsizdir. Belli olan tek şey
aralığı bir fazda başlatıp bir diğerinde bitirdiğidir.





Şekil 3.12

Bu nedenle, bilgisayar sinyalleri yalnızca düzeyi kesin olarak tanımlanabilen aralıklarda gözden geçirmelidir. Bu, daha önce gördüğümüz gibi, veri hatlarındaki bilginin doğru olduğu zamanı bildirmek üzere zamanlama sinyallerinin yükselen ve düşen kenarlarını kullanarak gerçekleştirilir. Buna bir örnek, bellek adres bilgisinin doğru olduğu zamanı belirten MEMREQ'in kullanımıdır.

Bu tür sinyallerin bir sistemde uygulanabileceği pek çok yol vardır. Örneğin MEMREQ sinyali, Kısım 2.2'de sözü edilen yolla, adres verisini belli sayıda *mandalı* stroblamak için kullanılabilir. Bu işlem yapılır ve sinyaller Şekil 3.7 ve 3.8'de verilen kutuplamaya sahip olurlarsa, MEMREQ sinyali, saat darbesinin pozitife giden kenarında veriyi alıp saklayan bir mandal grubuna saat girişi olarak kullanılır. Bu işlem, *veriyi sabitleştirme* olarak bilinir. Yükseliş periyodu süresince saatin tam olarak bilinmiyor olması, tüm bu periyot boyunca adres geçerli olduğu
S. 3.3 sürece önemli değildir.

Veriyi mandallara stroblamak için daha sık kullanılan alternatif bir yöntem
saat darbesinin negatife-giden kenarının veriyi sabitleştirdiği bir sistem kull
tır. Şekil 3.12 (a)'da, 8 kilit içeren ve dolayısıyla 8 bit saklayabilen (bkz:Kıs
bir tümleşik devrenin blok diyagramı verilmiştir. Her bir bitin bir veri girişi ve
vardır ve tüm bitler ortak bir saat giriş hattını paylaşırlar. Şeklin (b) bölüm
mandalların zamanlama diyagramı gösterilmektedir. Saat darbesinin yükseke
ğu aralıkta (T aralığında), her bir mandal *saydam* olarak adlandırılır, yani gi
çıkış daima aynı düzeydedir. Bu zaman zarfında bir mandalın girişi değişirse
da aynı şekilde onu izler.

T periyodunun sonunda, saat sinyali alçak düzeye düştüğünde, mandal girişi
deki veriler sabitlenmiş olur. Verinin sabitlendiği an, saat darbesinin düşen
rının yüksekten-alçağa eşiğinden geçtiği andır. Bununla beraber, daha önce de
edildiği gibi, tüm saat darbesinin düşüş periyodu süresince, veri girişleri k
olduğu sürece bu çok kritik bir parametre değildir.

Bu tip mandallar mikrobilgisayar sistemlerinde sıklıkla kullanılır. Bunlar
zaman ayrı bir parça halinde bulunmazlar, genellikle diğer modüllerin içler
mevcutturlar. Örneğin, bir bellek yongasına giren giriş hatları o yonga iç
mandallanır, (sabitlenir). Bundan başka, bir önceki bölümde de görüldüğü gibi,
sıklıkla programlanabilir çevresel arabirim portlarında sabitleştirilir ve işle
yongalarının kendileri de giriş ve çıkış mandalları içerirler.

3.7 Tamponlama ve Zamanlama

Bir mikrobilgisayar sisteminde, ya da daha doğrusu herhangi bir sayısal ma
sisteminde, bir sinyal üzerinde gerçekleştirilen herhangi bir mantıksal işle
sinyal yolunda mutlaka bir gecikmeye neden olduğunu bilmek önemlidir. Bu
nedeni, işleme ilişkin devrenin yol üzerine yerleştirilmesinin gerekmesi ve bu
da işlevini görebilmesinin belli bir zaman almasıdır. Örneğin, NAND (VE DEĞ
kapısına giren girişler birbirine bağlanır ve mantıksal 0'dan mantıksal 1'e giden
sinyal uygulanırsa, kapının *yayınım gecikmesinin* ardından çıkış mantıksal 1'
0'a gider. Yayınım gecikmesinin gerçek değeri pek çok faktöre, ama temel
kullanılan mantık devresinin tipine bağlıdır.

Sistem tasarımında, tamponlamanın bilgi iletim yolunda yol açacağı gecikmeler
dikkate alınmalıdır. Örneğin, Kısım 2.6'da özetlendiği gibi, bilgisayarın veri
adres yolu hatları tamponlandığında, bu hatlarda ortaya çıkacak gecikmeler un
tulmalıdır.

Adres yolunun tamponlanması halinde, adres bilgisinin geçerli olduğu aralık, zamanlama diyagramı üzerinde sağa kayar (Şekil 3.7 veya 3.8). Bu, MEMREQ sinyali (kendisi de tamponlanmadıkça) büyük oranda gecikmeye uğramadığı için, bilginin bellek mandalına stroblandığı noktanın adres bilgisinin hatta çıkarıldığı noktaya daha da yaklaşacağı anlamına gelmektedir. Bu nedenle, veri veya adres hatlarındaki çok büyük gecikmeler sistemin zamanlama toleransını düşürür. Verinin hatta çıkarılması zamanına çok yakın olan strob noktası, veri sinyallerinin yerleşmesi için yeterli süreye imkan vermeyebilir ve böylece hata oluşmasına neden olabilir.

Tamponlanmaya, çevresel birimlerin bilgisayar sistemine belli mesafe uzağında kullanılacağı durumlarda sık sık ihtiyaç duyulur. Bu tür uygulamalarda, çevrebirimine giren bağlantı hatlarını denetimli bir şekilde sürebilmek için gereken akım ve gerilimleri artırmak üzere özel *hat-sürücü* devreleri kullanmak gerekebilir. Programlanabilir çevresel arabirimler buna uygun devrelere sahip olmadıklarından, bu işlevi görebilecek kapasitede değillerdir. Yine buna ilişkin olarak, sürücüler ve uzun bağlantı hatlarının neden olacağı gecikmeler de dikkate alınmalıdır.

Sorular

3.1 Bir mikrobilgisayar sistemindeki tek bir program komutunun yürütülmesine ilişkin adımları sıralayın. Bu adımların her birini oluşturan işlemlerin fonksiyonlarını açıklayın ve tek baytlık bir komut ile iki-üç baytlık komutların yürütülmelerindeki farklılıkları özetleyin.

3.2 'Bir mikrobilgisayar sistemi, paralel veri yolları kullanarak aritmetrik devreler, bellek ve kaydediciler arasındaki karmaşık bağlantılardan oluşmuştur.' Yukarıdaki ifadeyi tartışın.

3.3 Bir sayısal mantık sisteminin çalışmasını açıklamada yararlanılan zamanlama diyagramları nasıldır? Bu tür diyagramların, sinyallerin ne zaman:
(a) geçerli
(b) geçersiz
(c) belirsiz
olduğunu göstermek için nasıl kullanıldığını tartışın ve yalnızca geçerli durumda olduğunda kullanılmasını sağlamak için, zamanlama denetim sinyallerinin kullanılabileceği yöntemi özetleyin.

3.4 Bir mikrobilgisayarın çalışma saykılının nasıl birkaç farklı zaman aralığına

bölünebildiğini tartışın ve
(a) bir komut-getirme saykılı ve
(b) bir bellek konumuna verinin yazılacağı bir yürütme saykılı boyunca aralıklarda yer alan adımları tanımlayın.

3.5 Bir mikrobilgisayardaki saat osilatörünün işlevini anlatan kısa bir ta yazısı yazın.

3.6 Bir mikrobilgisayarda tamponlamaya niçin gerek duyulduğunu tartış tamponlamanın sistem içerisinde zamanlama toleranslarını nasıl değiştir diğini özetleyin.

3.7 Eğer elinizde üretici kataloğu varsa, bazı ticari saat-üreteci devrele karakteristiklerini inceleyin ve:
(a) Kaç tane saat fazı üretebildiğini,
(b) Her bir saat fazının darbe genişliğini,
(c) Farklı fazlardaki darbeler arasındaki minimum gecikmeyi,
(d) Çıkış saat darbelerinin yükselme ve düşme zamanlarını sıralayın.
Cevapta minimum, tipik veya maksimum değerler ya da mevcutsa üçü bi verilebilir. Cevabınızda, Intel 8284, 8224 ve Texas Instruments 9904 üreteçlerinin tümünün de yer alması tavsiye edilir.

3.8 Bazı bellek erişim işlemlerinde mikroişlemci saykılının uzatılmasına n gerek duyulur? Bunun nasıl yapıldığını özetleyin. 3 MHz frekanslı tek f bir saat kullanan bir mikroişlemci:
(a) 400 nsn, lik bir erişim süresine sahip bir ROM'a
(b) 550 nsn, lik bir okuma-yazma saykıl süresine sahip bir RAM'a bağl caktır.

Saat darbesi işaret/boşluk oranın bir olduğu kabul edilmektedir.
(a) Her iki bellekten işlemciye okuma yapma,
(b) Veriyi RAM'a yazma
işlemlerini gösteren zamanlama diyagramlarını çizin.

3.9 Mikrobilgisayar sistemince:
(a) Bellek erişim işlemlerinde,
(b) Çevresel G/Ç işlemlerinde kullanılan denetim hatlarını sıralayın.
Sistem:
(a) bir bellek konumuna veri yazan ya da
(b) Programlanabilir bir çevresel arabirim devresine veri gönderen işle

çıkış işlemlerini birbirinden nasıl ayırt eder?

3.10 Farklı tipteki mantık devreleri farklı yayınım gecikmelerine sahiptir. Üretici literatürünü inceleyerek:
(a) Normal transistör-transistör mantık (TTL)
(b) Düşük-güçlü TTL
(c) Schottky-kenetlemeli TTL kullanan tampon yükselteçleri, TERSLEYİCİLER, VEDEĞİL kapıları için tipik yayınım gecikmelerini gösteren bir tablo düzenleyin. En en hızlı mantık tipi hangisidir?

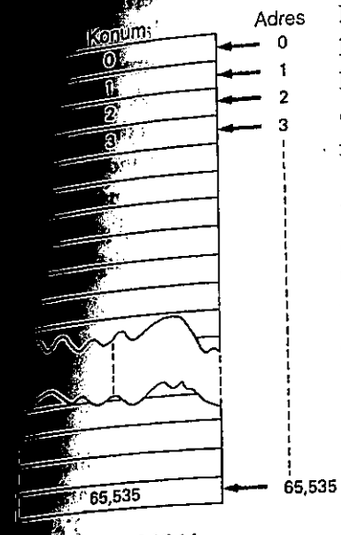
Bölüm 4 Bellek haritalama ve bellek organizasyonu

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

1 *Mikrobilgisayar belleğin organizasyonunu anlayabilmeli,*
2 *Bellek adresleri yazımında on altılı gösterimi kullanabilmeli ve ikili ve or... biçimler arasında sayı dönüştürmesi yapabilmeli,*
3 *Mikrobilgisayarın adres alanı kavramını anlayabilmeli,*
4 *Toplam adres alanının farklı tipteki belleklerce işgal edilebileceğini ve ... alanının tümünün bellek yongalarıyla doldurulması gerekmediğini anla... meli,*
5 *Bellek kompozisyonunu göstermek için bellek haritalarını kullanabilmeli,*
6 *(a) Özel IN ve OUT komutlarının kullanımı ve*
(b) Bellek-haritalı G/Ç dahil olmak üzere
çevresel veri aktarımı ile bellek veri aktarımı arasındaki farkları ayırt ed... lmeli,
7 *Çevresel birimlerin adresleme alanlarının, bellek adresleme alanlarına har... landırılması ilkelerini anlayabilmeli,*
8 *Mikrobilgisayar sistemlerinde kod çözmenin gerekliliğini kavrayabilmeli,*
9 *Kod çözme işleminin ilkelerini anlayabilmeli,*
10 *Kodlanmış bilgi kullanımının getirdiği avantajlar -örneğin, adreslerde, bir... ler arasında gereken arabağlantı sayısını azaltmak gibi- sıralayabilmeli,*
11 *Üç bitle kodlanmış bir adres bilgisinden 8 bellek konumundan birini seçm... için kod çözme yöntemini kullanabilmeli,*
12 *Pratik kod çözme yongalarının kullanımını anlayabilmeli,*
13 m *bit genişliğinde bellek entegrelerini,* n *bit genişliğinde saklama alanı oluş... racak biçimde (burada* n *değeri* m *'den büyüktür) birleştirme yöntemini açıkl... yabilmeli,*
14 *Kullanılabilir saklanan sözcük sayısını artırmak için kullanılan kod çözücüler... birlikte, birkaç bellek yongası grubunu birleştirme yöntemini açıklayabilmeli,*
15 *Bir adres aralığını seçmek için bir kod çözücünün nasıl kullanılabileceğini... örneğin, 1K'lık bir adres aralığının belleğe nasıl yerleştirilebileceğini açıkla... yabilmeli,*
16 *Eğer mikrobilgisayar yoluna çok sayıda bellek yongası bağlanacaksa, olası... yükleme problemlerini azaltabilmek için yol-sürücüleri gerekeceğini kav... rayabilmeli,*
17 *Belli sayıda iki-durumlu anahtarlama birimlerinin, merkezi işlemci birimi... tarafından sorgulanma yöntemini açıklayabilmelisiniz.*

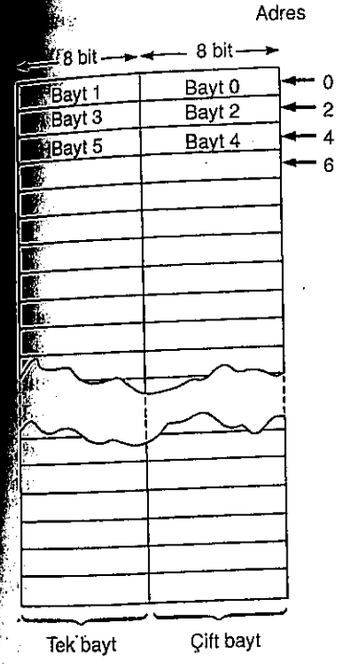
4.1. Mikrobilgisayar Belleği

Daha önceki bölümlerde, mikrobilgisayar belleği konusuna kısaca değinilmiş ve üretiminde kullanılan entegre devre tiplerinin ayrıntıları, işlemci ile bellek arasındaki işlemleri eşzamanlı yapmak için kullanılan denetim sinyalleri ve zamanlama yöntemleri ana hatlarıyla sunulmuştu.



Şekil 4.1

Bellekler, görülebileceği gibi, çok sayıda *konum* veya *sözcük*ten oluşmuştur. Bu, Şekil 4.1'de verilmiştir. Bu diyagram, belleğin sıfır (0) düzeyli yani 0, 1, 2 vd. gibi adreslerinin bulunduğu bölüm üstte ve bir (1) düzeyli adres bölümü ise altta olacak biçimde çizilmiştir. Şu noktaya dikkat edilmelidir ki, bu kitapta kullanılan yöntem bu olmasına karşın, diğer kitapların tümünde aynı yöntem kullanılmamış olabilir. Bazen diyagramlar, bir (1) düzeyli adresler üstte bulunacak şekilde çizilir. Her iki yöntem de yeterince kullanışlıdır, seçim kişinin tercihine kalmıştır.



Şekil 4.2

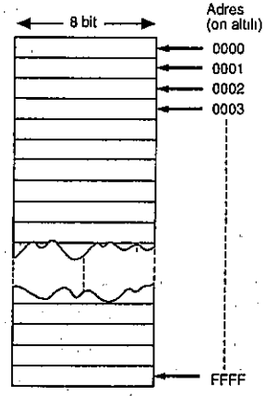
Bellek Genişliği

Her bir bellek konumu, veriyi veya bir komutun tümünü ya da bir bölümünü tutabilme yeteneğine sahiptir ve bellek konumları, içerdikleri bit sayısı, yani *genişliği* ile nitelendirilir. Mikrobilgisayar sistemlerinde bir sözcük, genellikle 8 bit uzunluğundadır. Bu nedenle de bellekler, 8 bitlik bir birim için yaygın olarak kullanılan bir ad olan 'bayt' dizilerinden oluşmuştur. Bu tür sistemlerde işlemci komutları, genellikle 8 bitlik büyüklükler üzerinde işlem yaparlar. Örneğin, LDA (adres), yani akümülatörü 'adres' kısmında verilen değerle yükle komutu, bellekten 8-bitlik bir veri değerini getirir ve bunu akümülatöre yerleştirir.

Bununla birlikte, bazı makineler bundan daha esnektir. Farklı veri tipleri -bayt, karakter, bit, sözcük, uzun sözcük- üzerinde yapılan işlemlerden Kısım 2.2'de söz edilmişti. Daha az tercih edilen, ancak hâlâ çok kullanışlı olan diğer bir yaklaşım da, komutları baytlarla ya da 16 bitlik sözcüklerle işlem yapabilen işlemcinin içine yerleştirmektir. Bu tür bir sistemde belleğin,

Tablo 4.1 *On altılı sayı gösterimi*

İkili desen	*On altılı gösterimi*
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F



Şekil 4.3 On altılı bellek adresleri

Şekil 4.2'deki gibi düzenleneceği düşünülebilir. Her bir
cük iki bayttan oluşmuştur ve sözcük adresleri daima çi
Bu nedenle, bunlara karşılık gelen komutlar da çift a
içerirler. Baytlara karşılık gelen komutlar doğrudan çift y
tek adresli baytlara ulaşabilir. Örneğin, LDA 2 komut
adresindeki çift baytı, LDA 3 komutu ise 3 adresindeki
baytı alacaktır. Bu, konunun bütünlüğü açısından tanı
belirtilmiştir, ancak burada üzerinde daha fazla durulmaya
tır. İlerideki kısımlarda, Şekil 4.1'de gösterildiği gibi, ard
baytlardan oluşan bir saklama alanının kullanıldığı varsa
mıştır.

Varsayımlar

Belleğin adres aralığı daha önceki bölümlerde anlatılmış v
kuşak mikroişlemcilerde, 8Mbayt'a kadar bellek aralığ
olduğu belirtilmişti. Bu sistemlerde, bellek adresleri 23'e (2
8.388.608) kadar bit içerebilirler. Ancak, bu bölümün amac
uygun olarak, bellek alanının, 16-bit adres uzunluğu gere
ren 64K (65.536)'lık olduğu varsayılmıştır. Bu varsayım,
tışmanın genelliğini azaltmaz, yalnızca böylesi uygun olu
için tercih edilmiştir. Burada verilecek tanımlar ve tartışıla
konular, diğer farklı büyüklükteki belleklere de uygulanab

On altılı Adresler

Şu ana kadar verilen örneklerde, bellek adresleri ondalık s
lar olarak ifade edilmişti. Tabii, uygulamada program adre
mikrobilgisayarda ikili sayılar olarak saklanır ve programc
bu adresleri, ikili forma kolayca çevrilebilecek biçi
yazması en uygundur. Adresleri ondalık sayı olarak yazm
ondalık ve ikili form arasında karmaşık bir çevirme iş
gerektireceğinden bu kriteri karşılamaz.

Mikrobilgisayar bellek adreslerinin on altılı gösterimi ise b
uygun düşmektedir. On altılı sayılar üzerinde biraz alıştı
yapıldığında yazması ve yorumlaması kolay oldu
görülecektir ve ikili sayılarla on altılılar arasında doğrudan

çevirme yapılabilir. Çünkü her bir on altılı basamak, Tablo 4.1'de gösterildiği gibi, doğrudan doğruya 4-bitlik bir ikili deseni gösterir.

Bu ikili sayıyı on altılı biçimde yazmak için sayı, en küçük değerlikli bitlerden başlayarak dört bitlik gruplara ayrılır ve her grup on altılı gösterme altına yazılır. Böylece 16 bitlik ikili sayıları örnek olarak kullanarak ve on altılı sayıları (on altılı), ikili sayıları (ikili) ve ondalık sayıları (ondalık) ekleriyle gösterirsek:

0000000000000000 (ikili)=0 (ondalık)=0000 (on altılı)
0000000011110000 (ikili)=240 (ondalık)=00F0 (on altılı)
1110000000000001 (ikili)=57.345 (ondalık)=E001 (on altılı)
0000101101101010 (ikili)=2.922 (ondalık)=0B6A (on altılı)
1111111111111111 (ikili)=65.535 (ondalık)=FFFF (on altılı)

elde ederiz.

Üzerinde biraz çalışılınca, on altılı ve ikili sayılar arasındaki dönüşümün son derece kolay ve açık olduğu görülecektir.

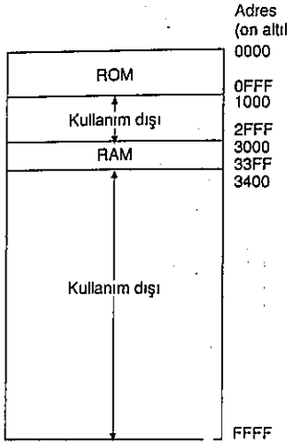
Tablo 1.2'deki örnek programda sıralandığı gibi, genellikle tercih edilen yol, bellek adreslerini belirtmek için on altılı sayıları kullanmaktır. Makine dilinde, program sabitleri de bu biçimde belirtilebilir. Bu gösterme biçimini kullanarak, mikrobilgisayar belleği Şekil 4.3'deki gibi tekrar çizilebilir. Daha öncede belirtildiği gibi, 8 bitlik bir genişlik varsayılmıştır. Olabilecek adres aralığı 0'dan 65535 (ondalık)'a ya da FFFF (onaltılı)'ya değişmektedir.

Bellek alanının doldurulması

Pratikte, mikrobilgisayar sistemi tüm bu adreslerde, saklama alanı sağlamaya yetecek sayıda bellek yongası içermeyebilir, yani bellekteki baytların sayısı 16-bit genişlikte adres kullanıldığında 65.536'ya kadar çıkabilmesine rağmen, gerçek bir sistemdeki, özellikle yalnızca kısa program gerektiren bir uygulamada kullanılmak amacıyla kurulan bir sistemdeki *gerçek* sayı, bundan çok daha az olabilir. Bu nedenle, mevcut bellek alanının yalnızca bir bölümü gerçek bellek yongalarıyla 'doldurulabilir'.

Bu tür bir sistemde, toplam adres sayısının ancak bir bölümü bir program içinde kullanılabilir, geriye kalan kısmı, bu bölümde ileride tartışılacak olan durum hariç olmak üzere, geçersizdir.

Görülebileceği gibi, mikrobilgisayar belleği, çeşitli bell
lerinin bir karışımı olabilir. Belleğin bir bölümü, ö
sadece okunur (ROM) ve bir bölümü de okunur-yazılır (
olabilir. Belleğin kompozisyonunu, yani ne kadar RA
kadar ROM ve kullanılmayacak adres alanının ne büyü
bulunduğunu gösteren bir yöntemin gerekli olduğu ç
gerekli ortadadır.



Şekil 4.4 Bir bellek haritası

4.2 Bellek haritası kavramı

Bir bellek haritası, Şekil 4.4'de çizildiği gibi, bu nok
şematik olarak göstermek için kullanılır. Harita, bir mik
gisayar sistemindeki kullanılabilir toplam adres alanı ile
likte, herhangi belirli bir adresin yerleştirileceği kull
alanını gösterir. Burada gösterilen sistem, 4Kbaytlık RO
1Kbaytlık bir RAM içermektedir. ROM, 0000 (on altılı)
33FF (on altılı) arasındaki adres alanını, RAM ise, 3000
altılı) ile 33FF (on altılı) arasındaki adres alanını kaplar. 1
(on altılı) ile 2FFF (on altılı) ve 3400 (on altılı) ile FFFF
altılı) arasındaki adresler kullanılmaz.

Bunu, daha bilinen bir şekilde, ondalık biçimde göste
istersek, ROM; 0 ile 4.095 arasındaki adresleri işgal e
4.096 ile 12.287 arasında bir boş aralık vardır; RAM
12.288 ile 13.311 arasındaki adresleri işgal eder. Geriye ka
13.312'den 65.535'e kadar adres alanı boştur.

Bellek haritasının kullanımı

Bellek haritası, sistem kullanıcısının sistemde kullanılab
bellek kapasitesinin ne kadar olduğunu ve ne kadar kulla
mayan adres bulunduğunu bir bakışta görebilmesini sağ
Yukarıda verilen örnekte kullanılan en yüksek adres 33FF(
altılı) 'dır.

Eğer sistem adres yolu hatları, A_0'dan (en küçük değerlikli h
A_{15}'e (en büyük değerlikli hat) kadar etiketlendirilmişse; A
ve A_{15} hatları daima sıfır olduklarından, bu hatlara ge
duyulmaz. Diğer tüm hatlar kullanılır ve ileride açıklanac

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
	32.768	16.384	8.192	4.096	2.048	1.024	512	256	128	64	32	16	8	4	2	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	}*
	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	}*
	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	}*
	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	On altılı sayı 3				On altılı sayı 2				On altılı sayı 1				On altılı sayı 0				

gibi içerdikleri kodlar çözülmelidir. Bellek adres hatları ile on altılı adres arasındaki karşılıklı ilişki, Tablo 4.2'de gösterilmiştir.

Haritanın incelenmesi sonucu çıkarılabilecek bazı genel sonuçlar vardır. Örneğin, 1 Kbaytlık bir bellek bloğunu adreslemek için, A_0'dan A_9'a kadar alt sıradaki 10 adres hattına gerek vardır. Bu 1 Kbaytlık blok bellekte nereye yerleştirilirse yerleştirilsin, yani nerede başlayıp nerede son bulursa bulsun, doğrudur ve bu on adres bitinin tüm kombinasyonları kullanılmalıdır.

Eğer 1K'lık blok 1.024'ün çift katları olan sözgelimi 2.048, 4.096, 8.192, vb. bir adresten başlarsa, en alttaki on adres hattına ek olarak, bu aralık içindeki herhangi bir adresi seçebilmek için bir bite daha gerek duyulacaktır. Örneğin, bellek adresleri 1.024 ile 2.047 arasında ise, A_0-A_9'a ek olarak yalnızca A_{10} hattı kullanılır; bellek adresleri 8.192 ile 9.215 arasında ise, sefer ek olarak yalnızca A_{13} hattı kullanılır (A_{10}, A_{11} ve A_{12}' ye gerek yoktur).

Blok eğer bu türde adreslerden başlamazsa, daha fazla adres hattına gerek olacaktır. 1 Kbaytlık bir blok, örneğin 8.200 ile 9.223 adresleri arasında ise, (A_{10}, A_{11}, A_{12} dahil olmak üzere) A_{13}'e kadar *tüm* adres hatlarını kullanır.

Yukarıdaki açıklamalarda yer alan öneriler, ilgili değerlerde uygun değişiklikler yapılmak suretiyle, diğer blok büyüklüklerine de uygulanabilir. Örneğin, 2 Kbaytlık bir bellek bloğunun adreslenebilmesi, A_0-A_{10} hatlarınının kullanılmasını gerektirir ve bu bloğun başlangıç adresi, örneğin 4.096, 8.192 gibi 2.048'in çift katları şeklindeyse, blok seçimi için bir hatta daha gerek duyulacaktır.

Yukarıdaki anlatımdan da görüleceği gibi, eğer bir mikrobilgisayar sisteminin tüm

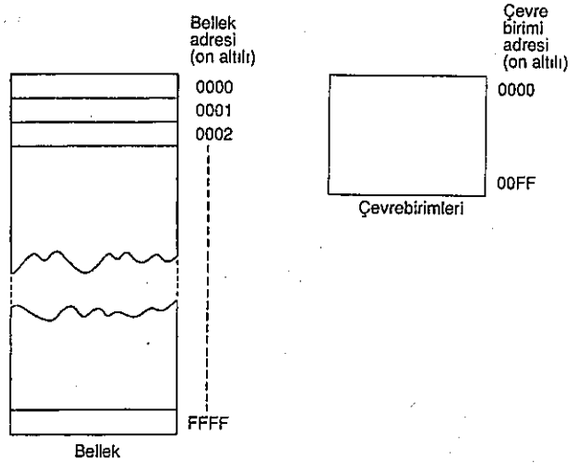
adres alanı, sözgelimi bir RAM bloğunun yer aldığı adres alanında alte
konumlar olabilmesi için doldurulmuyorsa, RAM başlangıç adresini dikk
seçmek suretiyle donanımda tasarruf sağlayabiliriz. Buradaki amaç, RAM'i
alanında yerleştirmek üzere, blok seçmek için adres kod çözme gereğini minim
S 4.1 indirmektir. Bu konu Kısım 4.5'de tekrar ele alınacaktır.

4.3 Bellek aktarımı ile G/Ç arasındaki farklılıklar

Şimdiye kadar G/Ç olanaklarının tartışıldığı konularda, bu olanakların mikroiş
ci komut takımında yer alan IN ve OUT gibi bazı özel çevresel aktarım kom
rınca sağlandığı varsayılmıştı. Bu, *programlı* G/Ç'dır.

Mikroişlemci, bu G/Ç komutlarının yürütülmesi sırasında, sistemin adres yo
bağlı çevresel birimlere, kendilerine istek gönderildiğini ve adres hatların
bilgiyi çözmesi gerektiğini bildiren bir çıkış denetim sinyali IOREQ (Kısım
üretir. Diğer taraftan, bir bellek aktarım komutu esnasında, IOREQ etkin olm
düzeyde kalır, denetim hattı MEMREQ (Kısım 3.5) ise, bellek sistemine kendi
bir adres bilgisi yöneltildiğini bildirmek üzere etkin duruma gelir.

Bu nedenle, çevresel adresleme ile bellek adresleme arasındaki farkı, IOREQ
MEMREQ sinyalleri oluşturur ve adres yolu kullanılarak elde edilebilen tüm



Şekil 4.5

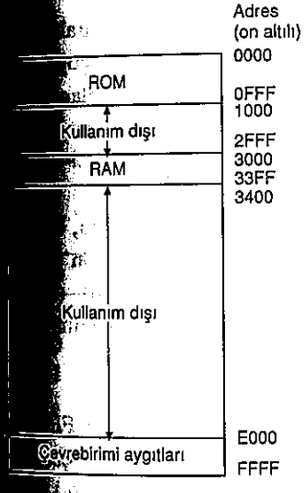
aralığı, çevresel birimlere ya da belleğe uygulanabilir. Bu nedenle, 16 bitlik bir adres yoluyla 65.536 olası bellek adresi ve 65.536 olası çevrebirimi adresi mevcut olmaktadır.

Genelde, bu kadar fazla çevrebirimi adres alanına gerek yoktur (ancak birkaç bilgisayar birbirinden ayrı 65.536 çevrebirimine sahiptir), dolayısıyla çevresel aktarım komutları ile birlikte yalnızca en alt-sıralı sekiz adres hattı (A_0-A_7) kullanılır.

Dikkat edilmesi gereken diğer bir nokta ise, Şekil 4.5'de görüldüğü gibi, MEMREQ ve IOREQ sinyallerinin etkisinin çevrebirimleri adres alan ile bellek adresleme alanını birbirinden ayırmasıdır. Bu şekilde, yukarıda izah edildiği gibi, çevrebirimi adresinin 8-bitlik olduğu varsayılmıştır.

4.4 Giriş/Çıkış için bellek alanı ayrılması

Eğer bir bellek adres haritasında kullanılabilir boş alan varsa, G/Ç için alternatif bir yaklaşım seçilebilir. Bu yaklaşım, kullanılabilir bellek adresleme alanını; birini, daha önce olduğu gibi bellek konumlarına erişmek, diğerini ise, çevresel birimleri adreslemede kullanmak üzere ikiye ayırmak şeklindedir. Bu yöntemi kullanarak, Şekil 4.6'da gösterilen biçimde, Şekil 4.4'e ilişkin bellek haritası tekrar çizilebilir.



Şekil 4.6 Bellek-haritalı G/Ç

Bu diyagramda, daha önce kullanılmayan adres alanının, E000 (on altılı)'dan FFFF (on altılı)'ya, yani 57.344'den 65.535'e kadar olan kısmı, çevresel birimler için ayrılmıştır. Adres yolunda, bu aralığa düşen bir adres algılandığında, bu adres, bir çevresel işlemi seçmek için kullanılır.

IOREQ ve MEMREQ denetim sinyalleri, daha önce olduğu gibi fonksiyonlarına devam ederler, ancak şimdi çevrebirimi seçimi, E000 ile FFFF (on altılı) arasındaki bir adres ile birlikte, etkin duruma gelen MEMREQ tarafından belirlenmektedir. Bu nedenle, ister belleğe isterse G/Ç birimlerine olsun, veri aktarımlarının tümü, MEMREQ sinyalinin etkin duruma gelmesiyle gerçekleşir ve G/Ç işlemlerinin bellek işlemlerinden ayrılması, bütünüyle aktarıma eşlik eden adres

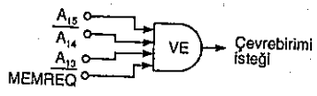
aracılığıyla yapılır.

G/Ç adresleri, bellek adres alanının bir parçası oldu
yani bellek alanına *eşlendiğinden* bu, *bellek-haritalı G*
rak bilinir. E000 ile FFFF (on altılı) arasındaki adres
G/Ç için kullanıldığından artık bellek erişimleri için
lamaz. Bu yolla mikrobilgisayarın kullanılabilir bellek
leme aralığı azalır, bu örnekte 64K'dan 56K'ya
Olabilecek maksimum bellek büyüklüğünü azalttığın
bellek-haritalı G/Ç'ın bir dezavantajıdır.

Bellek-haritalı G/Ç'ın avantajı, çevrebirimlerine ya da
birimlerinden yapılan veri aktarımının, belleğe ya da bel
yapılan veri aktarımı ile aynı olmasıdır. Makinenin
kodlarındaki bellek aktarımına ilişkin tüm komutlar, G
lemleri için de kullanılabilir. Bundan başka, bellek eriş
için mevcut tüm adresleme modları, G/Ç'a da uygulan
Böylece, işlemci ve çevrebirimleri arasındaki veri değiş
şu, artık basit IN ve OUTPUT komutlarıyla sınırlanmam
ve işlemleri, öncekinden daha esnek biçimde yapmak
kün hale gelir. Bir G/Ç örneği, bu bölümün sonundaki K
4.8'de verilmiştir. Bu aşamada, adresleme konusuyla da
etmek daha yararlı olacaktır.

Şekil 4.6'daki 8K baytlık bir çevrebirimi adresleme aralığ
seçimi kuşkusuz gelişigüzel yapılmıştır. Bu seçim, çoğu
temde çevrebirimleri için 8.192 adet adresin yeterli olaca
aynı zamanda da, çok fazla bellek alanı kaybedilmeyc
düşüncesiyle yapılmıştır.

Eğer yalnızca bir kaç çevrebirimine gereksinim varsa, bu
için daha küçük, örneğin F000'dan FFFF'e (on alt
(4.096 adres) ya da F800'den FFFF'e (on altılı) (2
adres) adres aralığı ayrılabilir, bellek büyüklüğü de buna g
artırılır.



Şekil 4.7

Çevrebirimince kullanılmak üzere ayrılmış E000'dan FFF
(on altılı) kadar 8.192 adresli orijinal bellek-haritalı G/Ç ör
ğine dönersek, en büyük değerlikli üç biti, yani A_{13}, A_{14}, A
bitleri '1' olan her adresin çevrebirimlerine karşılık gel

Tablo 4.3

			Birleşim Numarası
A	B	C	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

görülebilir. Diğer tüm adresler ise belleği ifade ederler. Bu nedenle, Şekil 4.7'de gösterildiği gibi, 4-girişli bir VE kapısı bir çevresel aktarımı bellek aktarımından ayırmak için kullanılabilir. Şu noktaya dikkat edilmelidir ki, bellek ve çevrebirimleri arasındaki ayrımın basitçe yapılabilmesi, yalnızca çevresel adresleme alanının, adres haritasının bir ucunda olacak şekilde seçilmesiyle mümkün olmaktadır. Adres haritasının içinde yer alan çevresel adresleme aralığının ayırt edilmesi biraz daha karmaşık olacaktır, yani daha fazla *kod çözme* işlemini gerektirecektir.

4.5 Kod Çözme

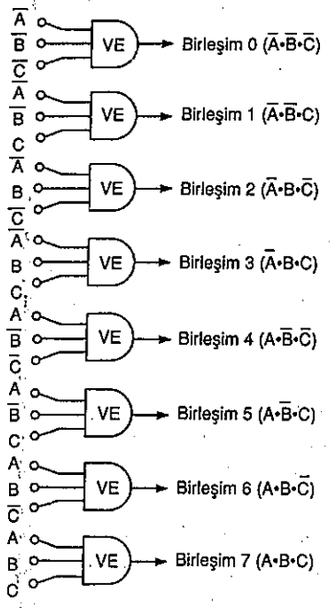
Mikrobilgisayar sistemlerinde kod çözme, ikili sayılara karşılık gelen ve biri diğerine benzemeyen birleşimleri algılamak için bir mantık devre grubunun kullanımı anlamına gelir. Bellek düzenlemesi konusuna dönmeden önce, bu konuyu anlamak için biraz konu dışına çıkmakta yarar vardır.

3-bitlik bir ikili sayıyı ele alalım. Tablo 4.3'de görüldüğü gibi bu sayı 2^3, yani 8 olası birleşimi gösterir. İkili basamaklar A, B ve C olarak adlandırılır.

Her bir olası birleşim, A, B ve C'nin bir diğerine benzemeyen tek bir grubunu ifade eder. A=1 olduğu durumlarda A, A=0 olduğu durumlarda $\overline{A}$ yazarsak:

Birleşim 0 = $\overline{A}$ ve $\overline{B}$ ve $\overline{C}$
Birleşim 1 = $\overline{A}$ ve $\overline{B}$ ve C
Birleşim 2 = $\overline{A}$ ve B ve $\overline{C}$
ve benzer şekilde devam edildiğinde
Birleşim 7 = A ve B ve C

Yukarıdaki listede bulunan gruplamaları gerçekleştirmek için, tek tek birleşimlere karşılık gelen sinyaller, VE kapıları kullanarak elde edilebilir. Şekil 4.8'de gösterildiği gibi, sekiz VE kapısı kullanılarak, her bir ikili sayı birleşimi için tek bir sinyal elde edilebilir. Verilen bu şekilde, VE işlemi, aradaki bir noktayla gösterilmiştir, böylece A *ve* B *ve* C ifadesi A.B.C



Şekil 4.8 Bir üç-bit giriş/sekiz-çıkışlı kod çözücü

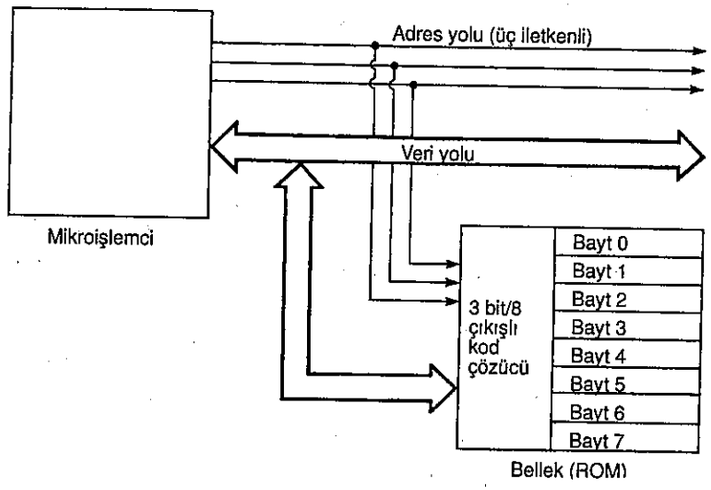
olarak yazılmıştır. Bu şekil, komple bir üç-bit giriş/se kışlı kod çözücüyü gösterir.

Tabii ki bu tür bir kod çözücünün boyutunu geniş mümkündür. Örneğin, dört basamaklı bir ikili sayı gru ayrı birleşime sahiptir. Bu nedenle, 4-bitlik bir kod çözü VE kapısına ve dört giriş ile 16 tane de çıkış hattına duyar. Bu kod çözücüdeki her bir VE kapısının dört vardır.

Kod çözmenin avantajları

Bir mikrobilgisayar sisteminde kod çözücüsü kullanı getireceği en büyük avantaj, birimler arasındaki araba sayısını azaltmasıdır.

Şekil 4.9'u ele alalım. Bu şekilde, bir mikroişlemci ile sa 8 baytlık bir bilgi içeren bir belleğe (ROM) sahip çok basit mikrobilgisayar sistemi gösterilmektedir. Bu, örneğin Ş 1.2 veya 1.5'deki gibi mikrobilgisayarlarda daha önce gör ğümüzle aynı yapıya sahiptir, ancak adres sayısı çok a azdır.



Şekil 4.9

Belleğin, üç-bit/sekiz-çıkışlı bir kod çözücü içerdiği varsayılmıştır. Bu devre, adres yoluna 3 bitlik bir adres çıkarıldığında, belleğin tek tek baytlarını seçmek için kullanılır. Dolayısıyla adresler, aşağıda verildiği gibi olmaktadır:

000 adresi Bayt 0'ı,
001 adresi Bayt 1'i
002 adresi Bayt 2'yi, vb. seçer.

Bellek, bu kod çözücüyü içerdiğinden, yukarıda da ifade edildiği gibi, yalnızca *üç* adres yolu bağlantısı yeterlidir. Bu hatlarla gönderilen her adres, üç bit ile elde edilebilen 8 kombinasyondan biri olarak iletilir ve baytların her birini seçmek üzere kodları *bellekte* çözülür.

Eğer bellek kod çözücü içermeseydi, neyin gerekeceğini düşünmek hayli ilginç olacaktır. O zaman sekiz baytın her birini seçmek için ayrı bir sinyal gönderilmesi, yani adres yolunda üç yerine sekiz hat bulunması gerekecekti.

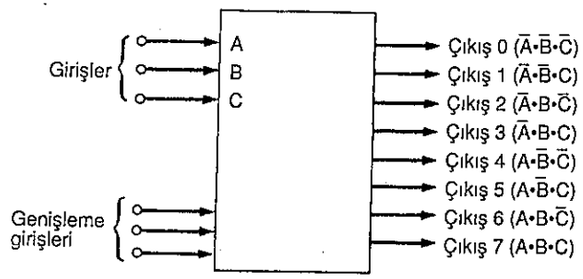
Bu durum, bu türde çok küçük bellekli bir örnek için büyük bir sorun olarak görülmeyebilir. Ne var ki, belleğin pratikteki boyutu, söz gelimi 1.024 veya 2.048 bayt büyüklükte ise, bir kod çözücünün gerekli olacağı açıktır. 2.048 baytlık bir bellek ve *kodlanmış* bir adresle, adres iletimi için 11 hatta gerek olacaktır. Ancak kodlama yapılmazsa, 2.048 hat kullanılması gerekecektir; açıkça görüldüğü gibi, bu pratik değildir.

Pratik Kod Çözücü Yongalar

Tek bir entegre üzerinde, üç-girişten sekiz-çıkışa kod çözen mantık devresine sahip özel kod çözücü entegre devreler, mikrobilgisayar sistemlerinde destek devreleri olarak kullanılırlar. Bunlardan daha önce Bölüm 2'de söz edilmişti. Bu devreler, şematik olarak Şekil 4.10'da gösterilmiştir.

Kodu çözülecek ikili desen, A, B, C girişlerine yerleştirilir ve ÇIKIŞ 0' dan ÇIKIŞ 7' ye kadar sekiz çıkış hattından birini etkinleştirir; sonra bu hatlardan biri üzerinde '1' sinyali görülür.

Genişleme girişleri, eğer belirli bir uygulamada bunlardan birkaçı kullanılacaksa, bunların arasından bir tane kod çözücüyü seçmek için kullanılır. Örneğin, 5 bitlik bir ikili desenin kodunun çözülmesi istenebilir. 2^5=32 olduğundan, bu 32 çıkış hattı demektir. Olası bir sistem Şekil 4.11(a)'da gösterilmektedir.



Şekil 4.10

Dört adet üç-bit giriş/sekiz-hatlı kod çözücü, D_1, D_2, D_3 ve D_4 kullanılır. Bu örnek bunlardan her biri yalnızca tek bir genişleme girişine sahiptir. Eğer bu giriş sinyal etkin (yani '1') ise, yonganın çıkışları daha önce açıklandığı gibi olacak. Eğer bu giriş etkin değilse, A, B, C sinyallerinin düzeyi *ne olursa olsun* entegre çıkışlarından hiçbiri etkin değildir. Bu nedenle, çıkışların hepsi "sıfır" olur.

Şekil 4.11(a)'daki kod çözücüye giren ikili veri deseni, B_0, B_1 ve B_2 en az ağırlıklı ve B_3 ve B_4 en ağırlıklı bitler olmak üzere, Şekil 4.11 (b)'deki gibidir.

B_0,B_1,B_2 bitleri, dört kod çözücünün A, B, C girişlerine parelel olarak girer. B_3 ve B_4 bitleri ayrı bir iki-bit giriş dört-çıkışlı kod çözücüye (D_0) bağlanır ve bunun çıkışları diğer dört yonganın genişleme girişlerini besler, böylece:

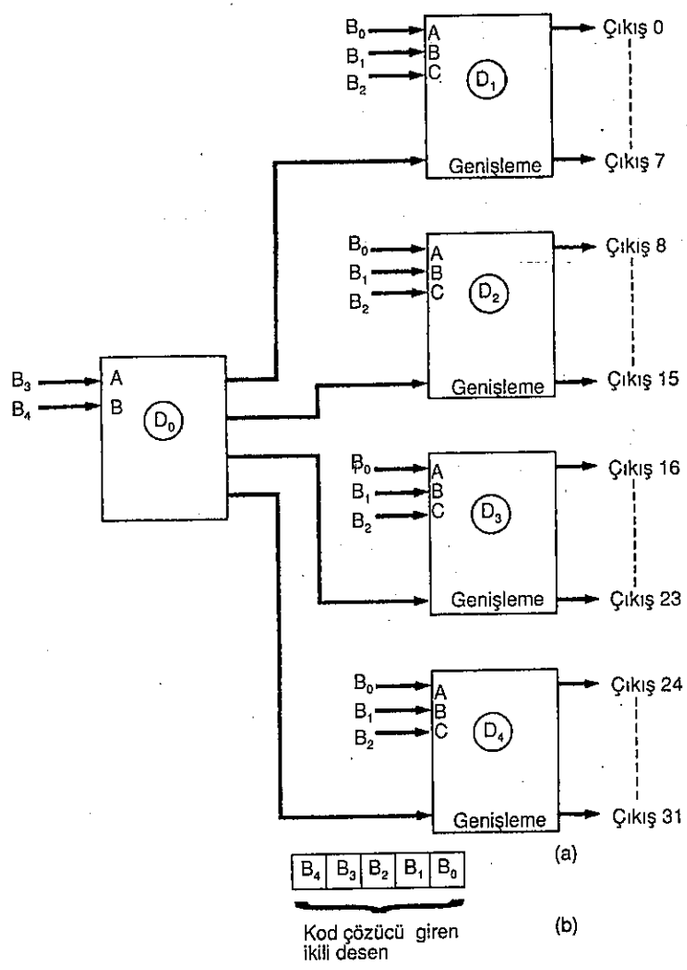
(a) B_3=0 ve B_4=0 olduğunda, D_1 yongası seçilir (0-7 çıkışları)
(b) B_3=1 ve B_4=0 olduğunda, D_2 yongası seçilir (8-15 çıkışları)
(c) B_3=0 ve B_4=1 olduğunda, D_3 yongası seçilir (16-23 çıkışları)
(d) B_1=1 ve B_4=1 olduğunda, D_4 yongası seçilir (24-31 çıkışları).

Diğer bir deyişle tüm devre, beş-bit giriş/otuz iki-hat çıkışlı kod çözücü gibi davranır.

Pratik bir kod çözücüde bulunan üç genişleme girişi, yonganın kendi içindeki kod çözücünün genişletilmesinde kullanılmak üzere mantıksal kombinasyonlar sağlar
S 4.9 ve bu yolla büyük bir esneklik olanağı verir.

Mikrobilgisayar sistemlerinde kod çözme

Kod çözme işleminin kullanımı, bu bölümde 8-bit bellekli basit bir örnek üzerinde



Şekil 4.11

ele alınmıştır. Tek bir baytın seçimi, bellek yongası içerisindeki üç-bit giriş/sekiz-çıkışlı bir kod çözücü ile gerçekleştirilir.

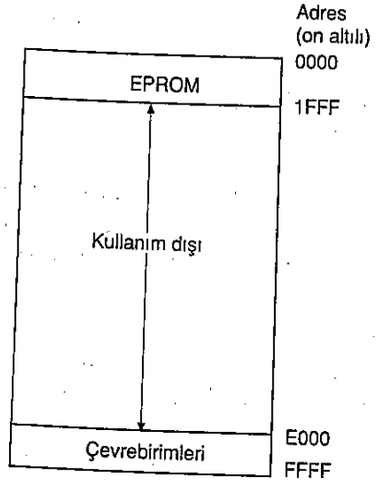
Pratik bellek yongaları genelde 8 bayttan çok daha fazlasını içerirler, ancak daha önce özetlenen ilkeler burada da geçerlidir. Bu nedenle, sözgelimi, 1.024 bayt kapasiteli bir bellek yongasında herhangi bir konuma erişmek için 10-bitlik ikili-kodlanmış adresler kullanılmalıdır. Bellek yongasının kendisi de, istenen baytı

seçmek üzere 10 adres biti kullanır, yani yonga, 10 bit girişi kullanarak i
konumdan birini gösteren kodu çözecektir.

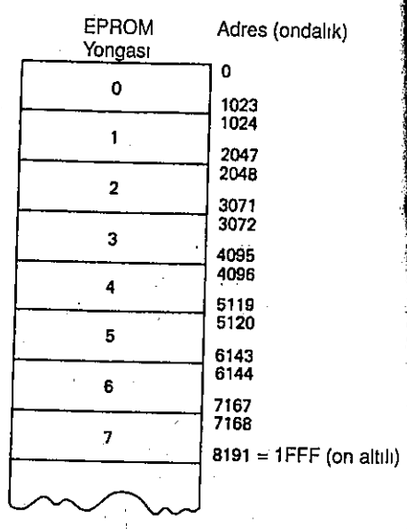
Bir mikrobilgisayarın toplam belleği, 64 Kbayt ya da daha fazla kapasiteyi c
turmak üzere bu türde bir çok yongadan oluşmuştur. Bu sistemlerin bir bil
olarak kod çözücü yongalarının işlevi; Bölüm 2'de özetlendiği gibi, sistemi c
turan bellek veya G/Ç devrelerinden birini seçmektir. Bunu, adres yolu
istenen bellek ya da G/Ç yongasını adresini gösteren ikili veri desenini algıla
ve belleği veya G/Ç devresini gözden geçirip belleğin veya G/Ç devresinin 'y
seçim' denetim girişini sürmek için bu desen tarafından üretilen çıkış sinyı
S 4.6, 4.7 kullanarak gerçekleştirirler.

4.6 Bellek Organizasyonu

8 tane 2.708 tipi EPROM yongasından oluşan bir mikrobilgisayar belleğini
alalım. Her bir yonga (Tablo 2.3) 1.024 bayt içerir. Dolayısıyla, bellek büyükl
8.192 bayttır ve bellek haritası Şekil 4.12'de verildiği gibidir. Verilen bu şekil
toplam bellek adresleme alanının 64 kbayt olduğu ve üst 8Kbayt adreslerin, d
önce açıklandığı gibi bellek-haritalı G/Ç olarak kullanılacağı varsayılmıştır.



Şekil 4.12



Şekil 4.13

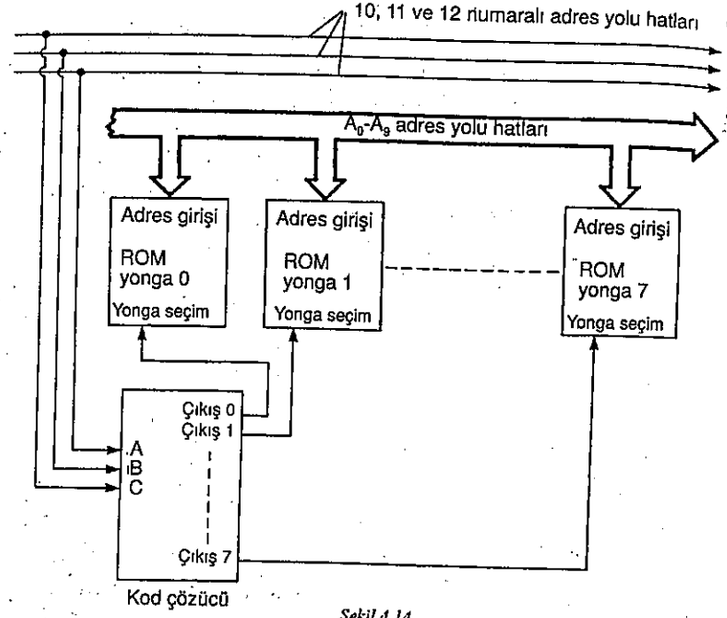
Adres (ondalık)	Adres (on altılı)	Adres (ikili) 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1,023	03FF	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1,024	0400	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2,047	07FF	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
2,048	0800	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
3,071	0BFF	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1
3,072	0C00	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
4,095	0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
4,096	1000	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5,119	13FF	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1
5,120	1400	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
6,143	17FF	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
6,144	1800	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
7,167	1BFF	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1
7,168	1C00	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
8,191	1FFF	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
					Yongayı seçer			Her bir EPROM içindeki adresi seçer.									

EPROM'u oluşturan sekiz yongadan her birinin, Şekil 4.13'de görüldüğü gibi, ayrı ayrı başlangıç ve bitiş adresleri vardır. Kolayca anlaşılması için, bu adresler ondalık biçimde yazılmıştır. Bunlar Tablo 4.4'de on altılı ve ondalık biçimde gösterilmiştir. Bu tablodan da görülebileceği gibi, en alttaki 10 adres biti (0-9 bitleri) yongalar *içindeki* adresleri seçmek üzere kullanılmaktadır. Örneğin, 5. yongadaki baytlar 5.120 (ondalık), yani 1400 (on altılı) adresinden başlar ve 6.143 (ondalık), yani 17FF (on altılı) adresinde sona erer. Bu bellek adres bloğunun başında, 0-9 bitlerinin tümü sıfırdır (5.120 için Tablo 4.4'deki değerlere bakınız) ve bloğun sonunda hepsi "1" dir (6.143 için Tablo 4.4'deki değerlere bakınız).

Yonganın kendi adresi, bir sonraki en ağırlıklı üç bit ile yani 10, 11 ve 12 bitleri ile verilir. 5. yonga için bitler 101(5); 7. yonga içinse 111(7)'dir.

Bu bellek, adres yoluna Şekil 4.14'de gösterildiği gibi bağlıdır. 0-9 adres bitleri, tüm bellek yongalarının adres girişlerine girerler. 10, 11, 12 adres bitlerinin kodu, 3-bit giriş/8-hat çıkışlı kod çözücü birimi tarafından çözülür, çıkış hatları ise istenen yongayı seçmek için kullanılır.

Daha önce de izah edildiği gibi, bilgisayarın bellek alanının, gerçek bellek ile doldurulması gerekmez, yani, Şekil 4.6'da görüldüğü gibi, arada bazı aralıklar bırakılabilir. Böyle yapılırsa, kod çözücünün, bellek yongalarını seçmek için kullanılan tüm çıkışlarının kullanılmasına gerek yoktur. Yalnızca istenen gerçek ROM, RAM veya EPROM'un bulunduğu adreslere karşılık gelen çıkışların kullanılması yeterli olacaktır.



Şekil 4.14

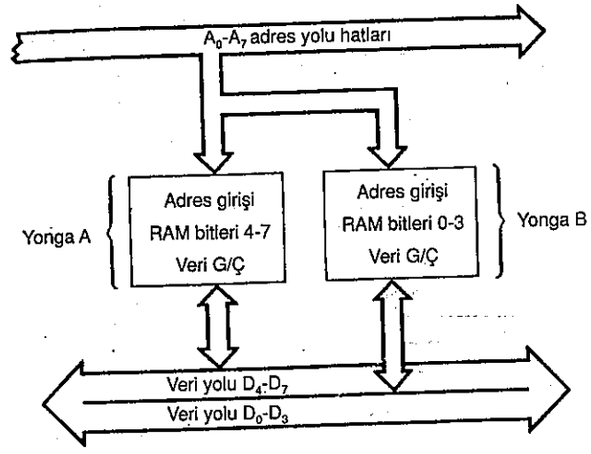
4.7 Bellek yongası büyüklüğü

Bellek yongaları, ileride daha açıkça görülebileceği gibi, çok değişik büyüklüklerde olabilir. Yongalar, hem adreslenebilecek her bir konumun *genişliği* (Bkz: Kısım 4.1) hem de yonga başına kullanılabilir toplam adres sayısı açısından değişkenlik gösterirler.

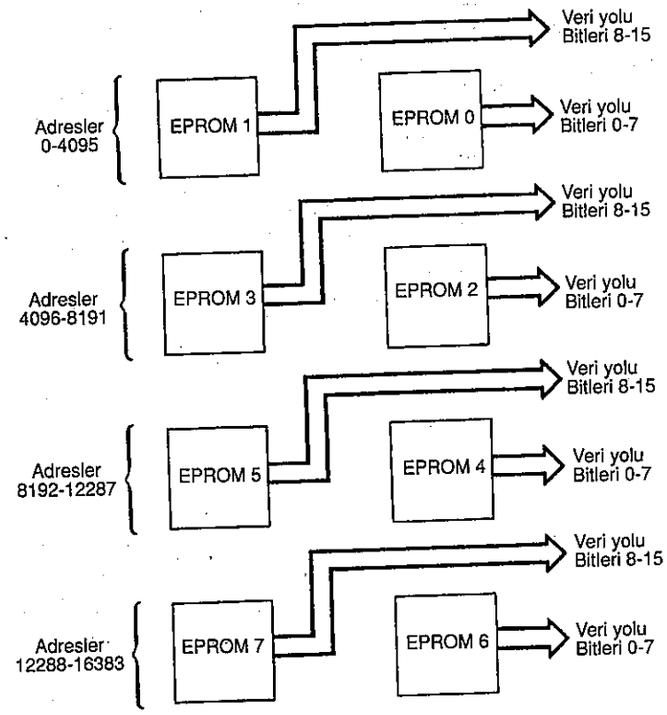
Bir önceki kısımda, saklama alanı sözcüklerinin, yani bir saklama alanındaki adres sayısının, ek adres kod çözme devreleriyle birlikte bir çok bellek yongası kullanılarak nasıl artırılabileceği incelenmişti. Bu kısımda ise, kısaca bellek genişliğinin nasıl artırılabileceği tartışılacaktır.

Bellek genişliğinin arttırılması

Tipik bellek yongası büyüklüklerinin örnekleri, Kısım 2.3'de verilmişti. Intel 2101A statik RAM belleğini ele alalım (Tablo 2.2). Bu yonga, her biri 4 bitten oluşan 256 sözcüklük bir kapasiteye sahiptir, yani her bir yonga konumunun genişliği 4 bittir. Sözgelimi 256 bayt kapasitede bir bellek oluşturabilmek için, Şekil 4.15'de görüldüğü gibi bağlanacak 2 yongaya ihtiyaç vardır.



Şekil 4.15



Şekil 4.16 2732 EPROM'ları kullanarak oluşturulan 16K-sözcük ve 16-bitlik bir bellek

İki yonga birbirine paralel olarak bağlanmıştır. Bir yonga (Şekildeki A yongası) bir baytın en ağırlıklı bitlerini yani (4-7 bitlerini, de içine alacak biçimde) 4 bitini, diğer yonga (B yongası) ise, en az küçük değerlikli bitlerini, yani (0-3 bitlerini içine alacak biçimde) diğer dört bitini saklar. Dolayısıyla, yongaların veri giriş ve çıkışları mikrobilgisayar veri yoluna şekilde görüldüğü gibi bağlanmıştır.

İki entegrenin adres girişleri ise, paralel olarak bağlanır. Her bir yonga 256 adres içerdiğinden, A_0-A_7 adres yolu hatları kullanılır.

'Yonga seçim' bağlantıları Şekil 4.15'de gösterilmemiştir, ancak daha önce de belirtildiği gibi, gerçekte mevcutturlar. Örneğin, 2101A RAM'lar kullanılarak 512 baytlık saklama alanına sahip bir bellek oluşturulacaksa, iki çift, toplam dört bellek yongasına gerek olacaktır. Şekil 4.15'de gösterildiği gibi, bir çift yonga 0-255 arasındaki, diğer çift de 256-511 arasındaki adresler için kullanılır. S 4.5

Bayt sayısı açısından daha büyük bir belleğe ihtiyaç duyulduğunda, daha fazla sayıda yonga çifti kullanılır. Dolayısıyla, üç çift halinde düzenlenmiş altı yonga, 768 baytlık, sekiz yonga, 1.024 baytlık vb. bellek oluştururlar. Bellek adresleme aralığını genişletmek için yonga çifti eklenmesi, Şekil 4.14'de verilen örnektekine benzer biçimde gerçekleşir ve buna uygun ek kod çözücülere ve 'yonga seçim' bellek denetim girişlerinin kullanımına gerek olacaktır. Her bir çiftteki belleklerin yonga seçim girişleri birbirine bağlanır. Son olarak, yukarıda ifade edildiği gibi, 8 bit genişliğinde bir bellek oluşturmak için 4 bitlik yongalar kullanılması yalnızca bir örnek olmak üzere verilmiştir. Diğer bellek genişliklerini kullanmak da benzer biçimde olmaktadır. Örneğin, 8.192 baytlık bir dinamik RAM, sekiz adet Texas TMS 4108 dinamik RAM yongası kullanılarak meydana getirilebilir (Tablo 2.1). Benzer biçimde, her biri 16 bit içeren 16.384 konumluk bir EPROM, her biri iki yongalık 4 grup halinde düzenlenmiş sekiz tane 2732 EPROM yongası (Tablo 2.3) kullanılarak meydana getirilebilir (Bkz: Şekil 4.16). S 4.3, 4.4

Yol sürücüleri

Görüldüğü gibi, bellek genişliğinin artırılması veya yeni adreslerin eklenmesi ile bellek büyüklüğünün artırılması, bellekteki yonga sayısının artırılmasını gerektirmektedir. Bu yongaların mikrobilgisayarın adres ve veri yollarına bağlanmaları gerekir; bu da, yollar üzerinde bir *yük* oluşturacaktır. *Yol sürücüleri*ne, bu yüklenmenin oluşturacağı sorunun çözümünde gerek duyulabilir.

Entegre devre yongalarının yük oluşturma nedeni, yongaların giriş ve çıkışların-

daki akım akışıdır. Yongaya bağlantılı devreler, bu akım akışını karşılayabilmelidirler.

Mikroişlemcide ve sistemin adres ve veri anayollarındaki sinyalleri alan ve gönderen diğer yongalar içinde yer alan devrelerin, yol bağlantılarından akan akımı taşıma yetenekleri sınırlıdır. Bu nedenle, bellek genişletilmesinde sık sık yapıldığı gibi, eğer bir yola çok sayıda yonga bağlanacaksa, yol ile bellek arasına yol sürücüleri eklenir.

Yol sürücüleri yalnızca birer yükselteçtirler. Bu türde bir yol sürücünün girişine bağlı bir yol hattının, (sürücü girişince ihtiyaç duyulan) küçük akımları taşıyabilmesi gerekir. Bununla birlikte, sürücü çıkışının diğer birçok yonganın gereksindiği akımları da verebilmesi gerekir. Bu nedenle yol sürücüleri, yolu, denetlediği diğer yongaların etkilerinden koruyan bir tampon gibi işlev görmektedir.

4.8 Basit çevresel giriş işlemlerine bir örnek

Bellek-haritalı G/Ç ilkelerinin bir özeti, bu bölümün başlarında verilmişti. Şimdi ise bununla ilgili basit bir pratik örneği ele alacağız.

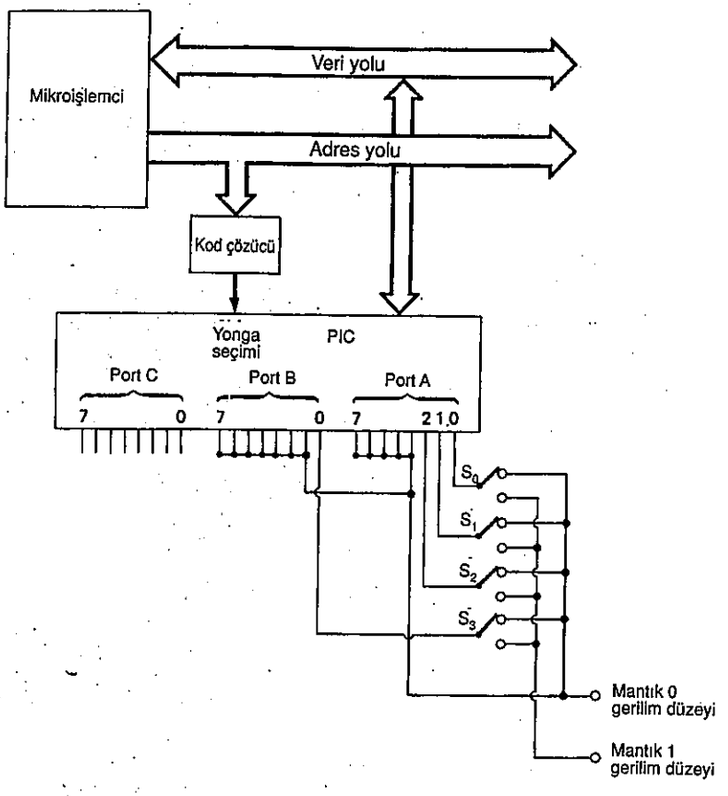
Çamaşır makinası gibi bir ev gerecinin içinde, sözgelimi ısı denetleyicisi veya sayısal saat gibi bir denetim birimi olarak kullanılacak bir mikrobilgisayar sistemini ele alalım. Hangi alanda kullanılırsa kullanılsın, mikrobilgisayar genelde, çok benzer işlemleri yapacaktır. Yani:

1 Dış bir birimden bazı giriş değerlerini okuyacak,
2 Bir hesaplama yapacak,
3 Dış birime gönderilecek çıkış bilgisini güncelleştirecektir.

Bu işlemleri oluşturan komutların ayrıntıları, uygulamaya göre değişir. Bir saat programının, çamaşır makinasını denetleyen bir programdan farklı olacağı açıktır; fakat, yukarıda verilen üç temel adım genelde olmak zorundadır.

Mikrobilgisayarın, değişik uygulamalardaki kullanım örnekleri ilerideki bölümlerde verilecektir. Burada, bu türde uygulamaların küçük bir altbölümü, yani bilgilerin mikrobilgisayara girişi tanımlanmıştır. Bundan başka, incelenecek girişlerin tümüyle iki durumlu anahtarlama elemanları olduğu varsayılmıştır. Bu elemanlar; röleler, anahtarlar ya da mandalların çıkışları olabilir.

Sistemin blok diyagramı Şekil 4.17'de gösterilmektedir.

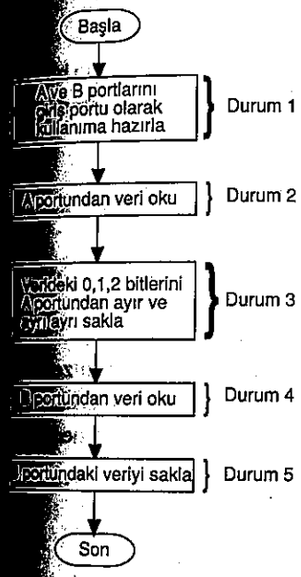


Şekil 4.17

Basitleştirmek için, mikrobilgisayara veri iletecek giriş birimlerinin tümü bi anahtarlar olarak seçilmiştir. Bunlar S_0, S_1, S_2 ve S_3 olarak adlandırılmıştır ve Bölümde tanımlanan tipte bir çevresel arabirim devresi kullanılarak mikrobilgi yar sistemine bağlanmışlardır. Her biri, mantıksal 0 veya 1 değeri; alacak biçimde açılıp kapatılabilirler.

Arabirim devresi, A, B ve C olarak adlandırılan, her biri 8-bitlik üç port içe Anahtarların, A portunun 0, 1 ve 2, ve B portunun da 0 bitine bağlandığı ka edilmiştir. Portların diğer bitleri -A portunun (3-7 bitleri) ile B portunun (1 bitleri)- sürekli olarak mantıksal 1 düzeyine bağlıdırlar.

Çevresel arabirim devresi, daha önce de izah edildiği gibi, üç portun yanı sıra de denetim kaydedicisi içerir. Birimdeki adresler, daha önce olduğu gibi (Kıs 2.4):



Şekil 4.18

Port A	0, yani 00 (on altılı)
Port B	1, yani 01 (on altılı)
Port C	2, yani 02 (on altılı)
Denetim kaydedicisi	3, yani 03 (on altılı)

olarak alınacaklardır.

Bu adresler, PIC (Çevresel Arabirim Devresi)'nin 'kaydedici seçim' girişleri kullanılarak seçilir. Ne var ki, karışıklığa yol açmamak için bunlar diyagramda gösterilmemiştir. Bu girişler, Şekil 2.16'da gösterildiği gibi bağlanmıştır.

IN ve OUT komutlarının kullanımı

Veriler, çevresel arabirim devresi üzerinden, IN ve OUT komutları kullanılarak iletilebilir. Anahtarları okumak için yalnızca IN komutu yeterlidir.

Bu komutlar kullanıldığında, görüldüğü gibi, devrenin adresi, normalde yalnızca en alttaki 8 adres hattı kullanılıyor olsa bile (Kısım 4.3), makinenin adres aralığının herhangi bir yerinde bulunabilir. Dolayısıyla, bu örnek için çevresel arabirim devresinin adresinin 08 (on altılı) olduğunu varsayacağız. Bu adres, Şekil 4.17'de kod çözücü tarafından algılanır ve Çevresel Arabirim Devresini yetkilendirmek için kullanılır.

Bu nedenle port adresleri aşağıdaki gibidir:

Port A	08 + 00 = 08 (on altılı)
Port B	08 + 01 = 09 (on altılı)
Port C	08 + 02 = 0A (on altılı)
Denetim kaydedicisi	08 + 03 = 0B (on altılı)

Anahtarların durumunu okumak için kullanılan programın akış diyagramı Şekil 4.18'de verilmiştir.

Programın 1. aşaması, A ve B portlarının giriş olarak kullanılması için Çevresel Arabirim Devresi'nin (PIC) kurulması, yani *kullanıma hazırlanmasıdır*. Bu işlem, PIC denetim kaydedicisinin adresi ve A'da tutulan denetim kaydedicisine

yerleştirilecek bit deseni ile birlikte, bir OUT komutu kullanılarak gerçekleş
Böylece:

```
MVI    A, 92
OUT    0B
```

Çevresel Arabirim Devresi'nin (PIC) denetim kaydedicisinde bulunan bit d
Şekil 2.13'de gösterildiği gibi, A ve B portlarının her ikisini "0" modunda
yapan 92 (on altılı), yani 10010010 (ikili)'dir.

Programın 2. aşaması, A portundan veriyi okur. Böylece bu aşamada şu k
bulunur:

```
IN     08
```

Okunan veri A kaydedicisine girer. Bu komutun ardından, A kaydedicisini
küçük değerlikli üç biti, S_0, S_1 ve S_2 anahtarlarının durumlarını gösterecek biç
"bir"ler ve "sıfır"lar içerir. A'daki diğer tüm bitler sıfırdır.

3. aşamada 0,1 ve 2 bitlerini ayırmak ve değerlerini teker teker saklamak için bi
program komutuna ihtiyaç vardır. Bunu gerçekleştiren kod parçası burada
alınmayacak, fakat benzer örnekler ilerki bölümlerde verilecektir.

4. aşamada, B portu üzerinden S_3 anahtarının durumu okunur. Böylece ko
aşağıda verilmiştir:

```
IN     09
```

Veri, yeniden A kaydedicisine girer. Ne var ki veri, bu sefer B portuna yalnızca
anahtar bağlı olduğu için, doğrudan saklanabilir. Bu tür bir saklamayı gerçekleş
menin uygun bir yolu, A'daki değeri diğer bir kaydediciye aktarmaktır. Bu nede
5. aşamada:

```
MOV    E, A
```

yani, "A kaydedicisindeki değeri E kaydedicisine aktar" komutu yer alır.

Bellek haritalı G/Ç

Bellek-haritalı G/Ç, verileri, IN ve OUT komutları yerine, PIC üzerinden aktarm
için kullanılabilir.

Bu yöntemde, PIC'in adresi çevresel birim adresleme alanı içerisinde olacak şekilde düzenlenmelidir. Şekil 4.6'da gösterilen adres alanı ayırmayı kullanarak, PIC; E000 (on altılı) ile FFFF (on altılı) arasında bir adrese sahip olmalıdır. E000 (on altılık) olduğunu varsayalım, o zaman:

A portunun adresi E000 (on altılı)
B portunun adresi E001 (on altılı)
C portunun adresi E002 (on altılı)
Denetim kaydedicisinin adresi E003 (on altılı) olacaktır.

Kısım 4.4'de de açıklandığı gibi, bu giriş-çıkış sisteminde, herhangi bir bellek aktarım komutu, çevresel veri aktarımı için kullanılabilir. Böylece programın başında Çevresel Arabirim Devresi'nin (PIC) kullanıma hazırlanmasında şu komutlar kullanılabilir:

```
MVI    A, 92
STA    E003
```

İkinci komut, A kaydedicisinin 92 (on altılı) içeriğini, doğrudan E003 adresine, yani PIC'in denetim kaydedicisine yazar.

Benzer biçimde, veri A ve B portları aracılığıyla, A'nın dışında diğer kaydedicilere de okunabilir. Örneğin:

```
LXI    H, E001
MOV    E, M
```

komutları, H, L kaydedici çiftini, E001 (on altılı) ile, yani B portunun adresi ile yükler ve ardından, o adresteki bilgiyi doğrudan E kaydedicisine alır.

§4.8;4.10, 4.11 Bu olanaklar, çevresel veri aktarımı için daha ekonomik programlar yazılabilmesine izin verir.

Sorular

4.1 Bellek haritası kavramını açıklayın ve bir mikrobilgisayar sistemi için bellek sistemi tasarımına yardımcı olabilmesi için nasıl kullanılabileceğini gösterin.

4.2 *(a)* Bellek genişliği ve

(*b*) Adres aralığı

kavramlarını açıklayarak, bir mikrobilgisayar sistemindeki bellek organ
yonunu tanımlayın.

Bir mikrobilgisayar, 32 Kbaytlık bir adres alanına sahiptir. Makine, bel
sıfır (0) seviyeli adres bitiminden itibaren 0000 (on altılı) adresinden bas
12 Kbaytlık bir ROM'a ve belleğin bir (1) sıralı adres ucunu kapla
Kbaytlık bir RAM'a sahip olduğuna göre, her bir bloğun başlangıç ve
adreslerini on altılı gösterimde ifade edin.

4.3 n-bitten daha az genişliğe sahip özel entegre devreleri kullanarak, bir n-bi
belleğin nasıl oluşturulabileceğini açıklayın. 2. Bölümde verilen tabl
kullanarak, aşağıdaki bellekleri oluşturabileceğiniz yollar önerin:

(*a*) 100 nsn.'den daha az erişim süresine sahip 64 baytlık bir RAM.

(*b*) 16.384 bayt'lık bir EPROM.

(*c*) 16.384 sözcüklük bir ROM. Her bir sözcük 16 bitliktir.

(*a*), (*b*) ve (*c*) şıklarının her birindeki yongalara adres bağlantılarının n
yapılacağını gösterin. Bu sistemlerde adres kod çözücülerine ihtiyaç
mıdır?

4.4 8.192 bayt'lık bir bellek oluşturmak üzere 16 tane Intel 2114 RAM yon
kullanılacaktır. Bu bellekteki herhangi bir adresi seçmek üzere kullanıla
A0-A12 adres hatlarını yetkilendirmek için bir adres kod çözücü yongası
nasıl kullanılabileceğini gösterin.

4.5 Yüksek kapasiteli bir bellek oluşturmak için, çok sayıda bellek yon
kullanıldığında, fazladan ne gibi ek devrelere ihtiyaç duyulur. Her bir
sürücü yongasının dört sürücü içerdiği varsayılırsa, 4.4 numaralı soru
bellek için gerekecek toplam yonga sayısını bulun.

4.6 1 Kbaytlık bir bellek bloğunun, adres kod çözücü kullanılarak, daha geniş
Kbayt) bir bellek içerisine nasıl yerleştirilebileceğini tartışın. 1 Kbay
bloğun yerleştirilme biçiminin, gerekli kod çözücü devrelerin karmaşıklığı
nasıl en aza indireceğini gösterin. Sözü geçen tüm adresleri onaltılı for
ifade edin.

4.7 Toplam 64 K'lık adresleme aralığına sahip bir mikrobilgisayar, çevres
birimlere erişmek için, 62K ile 64K aralığındaki adreslerin kullanılacağ
bellek-haritalı G/Ç kullanacaktır. Bu sistemde kullanılacak adres kod çözü
gereklerinin neler olacağını tartışın. Çevresel adreslerin bu türde seçilme
uygun mudur?

4.8 IN ve OUT gibi özel çevresel aktarım komutları kullanan programlı G/Ç'ı, bellek haritalı G/Ç ile karşılaştırarak aralarındaki farklılıkları belirtin. Bu yöntemlerin avantaj ve dezavantajlarını sıralayın.

4.9 Mikrobilgisayar sistemlerinde, kod çözme niçin gerekmektedir? 6 bit/64 hat'lı bir kod çözücünün:

(*a*) Basit mantık kapıları ve

(*b*) Tek bir genişleme girişli, 3 bit/8 hat'lı kod çözücü yongalar kullanılarak, nasıl oluşturabileceğini açıklayın.

4.10 Bir mikrobilgisayar, bir çamaşır makinesinde denetleyici olarak kullanılacak, yıkama çevriminin durumunu gösteren iki durumlu sinyalleri sorgulayacaktır. Bu nedenle:

Sinyal 1 = 1 olduğunda, yıkama kazanı doludur.
Sinyal 2 = 1 olduğunda, su ısıtıcısı açıktır.
Sinyal 3 = 1 olduğunda, yıkama kazanı boştur.
Sinyal 4 = 1 olduğunda, suyun sıcaklığı yeterli düzeye ulaşmıştır.

Bu sinyallerin mikrobilgisayara nasıl bağlanabileceğini gösterin. Bunları sorgulayan program için bir akış diyagramı çizin ve E 1'deki kodu kullanarak, çamaşır makinesinin durumunu incelemek için gerekli komut listesini yazın.

4.11 Bir mikrobilgisayar, kare dalga sinyali üretmek için kullanılacaktır.

(*a*) Kare dalgaların 1 ya da 0 olup olmadığını (Eğer 0 ise, mikrobilgisayar çıkışı mantıksal 0 düzeyinde kalmalıdır) ve,

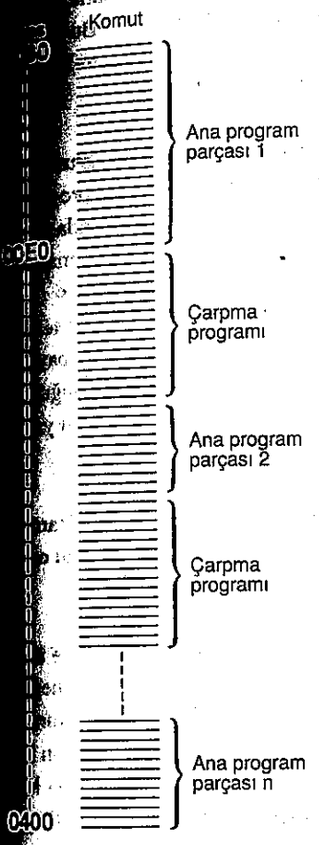
(*b*) Kare dalganın süresini denetlemek için, mikrobilgisayara dört anahtar bağlanacaktır. Süre, T'nin katlarında ayarlanacaktır. Dolayısıyla, T, 2T, 3T.....8T'ye kadar olabilir.

Anahtarların, mikrobilgisayarlara nasıl bağlanabileceğini gösterin ve onları denetleyen programın akış diyagramını çizin.

Bölüm 5 Altyordam ve yığınlara giriş

Bu bölümün amaçları : *Bu bölümü bitirdiğinizde:*

1 *Bilgisayar programlarının, sıklıkla, birbirinin tekrarı olan komut dizileri[n]*
ibaret olduğunu anlayabilmeli,
2 *Bu tür komut tekrarlarının bellek israf ve programcı için boşuna çaba sarfe[t]*
anlamına geldiğini kavrayabilmeli,
3 *Altyordam kullanımının, daha büyük bir programda, pek çok kez kullanıla[cak]*
cek, tek bir program kodu parçasına olanak tanıdığını anlayabilmeli,
4 *Altyordamların temel ilkelerini açıklayabilmeli,*
5 *Altyordamların:*
(a) Program amaç kodunu daha kısaltmaları,
(b) Programların modüler biçimde yazılmasını teşvik ederek, program y[a]
sını geliştirmeleri,
(c) Programın okunabilirliğini artırmaları,
(d) Program çalışma süresini artırmaları,
(e) Ana programda kullanılan kaydedicilerin içeriklerini değiştirebilme[si]
gibi önemli olanaklarını açıklayabilmeli,
6 *Bir ana program ile bir altyordam arasında, parametre ya da argüman geçi[şi]*
gerekliliği, ve bunu gerçekleştirmek için:
(a) işlemci kaydedicilerini kullanmak; ya da
(b) parametreleri belleğe yerleştirmek ve gerekli bellek alanını göste[ren]
işaretçiyi tutmak için bir ya da daha fazla işlemci kaydedicisi kullanm[ak]
şeklinde iki yöntem bulunduğunu kavrayabilmeli,
7 *Bir altyordama giriş ve altyordamdan çıkış için, özel CALL ve RETU[RN]*
komutlarına gerek olduğunu kavrayabilmeli,
8 *Yukarıdaki 5. maddenin (e) şıkkında belirtilen nedenle, işlemci kaydedici iç[e]*
riklerini, bir altyordam girilmeden önce saklamak gerektiğini kavrayabilme[li]
9 *Bir altyordama doğru biçimde giriş ve altyordamdan çıkışın, program sayac[ı]*
nın değerinin girişte saklanması ve bu değerin çıkışta geriye alınması gerek[ti]
ğini anlayabilmeli,
10 *Yığının, son-giren ilk-çıkar (LIFO) düzeninde bir bellek olduğunu kavrayab[il]*
meli,
11 *Verilerin yığına atılışı ve oradan çıkartılış ya da çekilişini anlayabilmelisini[z]*



Şekil 5.1

5.1 Altyordamlar niçin önemlidir ?

Bir bilgisayar programında, özel bir program parçasını birçok kez çalıştırmak sıklıkla gerekebilir. Bu durumda, bir *altyordam* kullanılabilir. Bir altyordam, daha büyük bir program içerisinde tekrar tekrar çalıştırılabilen bir program kodu parçası, yani komut dizisidir.

Altyordam gereksinimi

Şekil 5.1'de gösterilen durumu ele alalım. Bu diyagramda, 00B0 (on altılı) adresinde başlayıp 0400 (on altılı) adresine kadar devam eden bir makine kodu komut dizisinden oluşan bir program gösterilmektedir. Program, 848 baytlık bir bellek alanını kaplar. Programdaki komutlar, şekilde, kısa yatay çizgilerle gösterilmiştir. Bu gösterme biçimi, diyagramı çizerken kolay anlaşılabilir olması için seçilmiştir.

Program, herhangi bir işi gerçekleştiren, ('ana program bölümü 1' şeklinde adlandırılan), bir komut dizisiyle başlar. Bunların, bilimsel hesaplamanın bir parçası olduğunu varsayalım.

Bu tür bir hesaplamada, büyük bir olasılıkla sayıların çarpımını oluşturmak, yani sayıları birbirleriyle çarpmak gerekeceği açıktır. Bununla birlikte, eğer Intel 8085 gibi bir mikrobilgisayar kullanılıyorsa, komut takımı içinde 'çarpma' komutu yoktur. İşlemci içinde, çarpma işlemini gerçekleştirecek hiçbir devre mevcut değildir. Bu arada , 'bölme' komutunun da bulunmadığını söylemekte yarar vardır.

Çarpma, gerçekte, bir toplama işleminin tekrarlamasından ibarettir. Örneğin:

$$4 \times 24 = 24 + 24 + 24 + 24$$

Benzer biçimde bölme de, tekrarlanan bir dizi çıkartma işleminden oluşmaktadır. Bu nedenle, çarpma yapmak için, işlemcinin toplama komutunu kullanarak kısa bir program yazmak ya da çıkartma komutunu kullanarak bir bölme program yazmak (Bkz:Ek 1) göreceli olarak basit bir iştir. Bu tür program-

lara genellikle *yordam* denir.

Şekil 5.1'e dönelim ve ana program parçasının, iki sayının değerlerini hesapl
ve ardından bunların birbiriyle çarpılacağını varsayalım. Çarpma işlemi, yu
önerildiği gibi, ana program parçası 1'i izleyen kısa bir program biçimind
çarpma programı ile gerçekleştirilir.

Bu çarpımın ardından, 'ana program parçası 2' kod dizisi tarafından daha
işlem yapılır ve bunun ardından bir başka çarpma işlemi gerçekleştirilir. Bu
çarpma programları arasına serpiştirilmiş ana program parçaları kullanılarak
tekrar sürdürülür.

Şekil 5.1'de gösterildiği gibi, komple bir program, kolay anlaşılır doğru
komut dizisi olarak yazılırsa, şekilde yalnızca ilk ikisi gösterilen, çarpma pro
larını oluşturan komutları da eklemek gerekecektir. Bu bölümün başında
edildiği gibi, özel bir işlem (bu örnekte çarpma işlemi) pek çok kez tekrarlanır

Açıkça görülmektedir ki, çarpma programı komutlarını, bu şekilde tekrar
yazmak savurganlık olacaktır. Örneğin, program, her biri bir ya da iki b
saklama alanı gerektiren 40 veya 50 komut içerirse, programa her konulduğ
bu program, en az 40 baytlık bir bellek alanını tüketecektir. Bu tür en çok bir
iki eklemeye izin verilebilir, daha fazlası uygun olmayacaktır.

Altyordamların kullanılması, pek çok kez kullanılacak olan çarpma progr
oluşturan komutların bir kopyasının alınmasına olanak tanıyarak, bu sorunu
dan kaldırır.

Yukarıdaki açıklamalarda, programın gerektirdiği saklama alanını küçültm
rinde durulmasına karşın, diğer bazı faktörler de önem taşımaktadır. Ö
çarpma yordamı program içinde defalarca tekrarlanacak olursa, programcı
bu programı ve dolayısıyla program bilgisayara girilirken, programı oluştur
komutu defalarca yazması gerekecektir. Dolayısıyla, yalnızca tekrar eden b
parçalarının yazılması bile, çok fazla çaba harcanmasına neden olacaktır. D
olarak, bu arzu edilen bir durum değildir.

5.2 Altyordamlara giriş

Şekil 5.1'dekine alternatif bir program yapısı, Şekil 5.2'de gösterilmektedir.
da, ana program parçası ile çarpım yordamı birbirinden ayrılmıştır.

Çarpım yordamı yalnızca bir kez yazılır ve daha sonra görüleceği üzere, çarpma *altyordamı* olarak adlandırılır. Bu örneğin amacına uygun olarak, bu alt yordamın, ilk komutu 0301 adresinde olacak şekilde belleğe yerleştirileceği varsayılmıştır. Alt yordamın içindeki çarpma işlemini gerçekleştiren komutlar, daha öncekilerle tümüyle aynıdır.

Ana programı oluşturan kod parçaları da bu aşamada, birbiri ardı sıra sıralanır. Böylece, 'ana program parçası 1', 00E0 adresinde biter ve 'ana program parçası 2', bir sonraki adreste yani 00E1'de başlar. Kuşkusuz çarpma altyordamı ve ana program, belleğe, bellekte işgal ettikleri adresler birbirleriyle çakışmayacak şekilde, aynı anda yerleştirilmelidir. Bu yüzden, altyordamın başlangıç adresi, ana program içerisinde serpiştirilmiş bir çok alt yordam kodu bulunmadığından, ana programın bitiş adresi olarak kabul edilen 0300 adresinden sonraki, yani 0301 adresi olarak gösterilmiştir.

Bu şekilde altyordam, altyordama giriş ve alt yordamdan çıkış yönteminin açıkça gösterilebilmesi için, ana programdan ayrı olarak çizilmiştir.

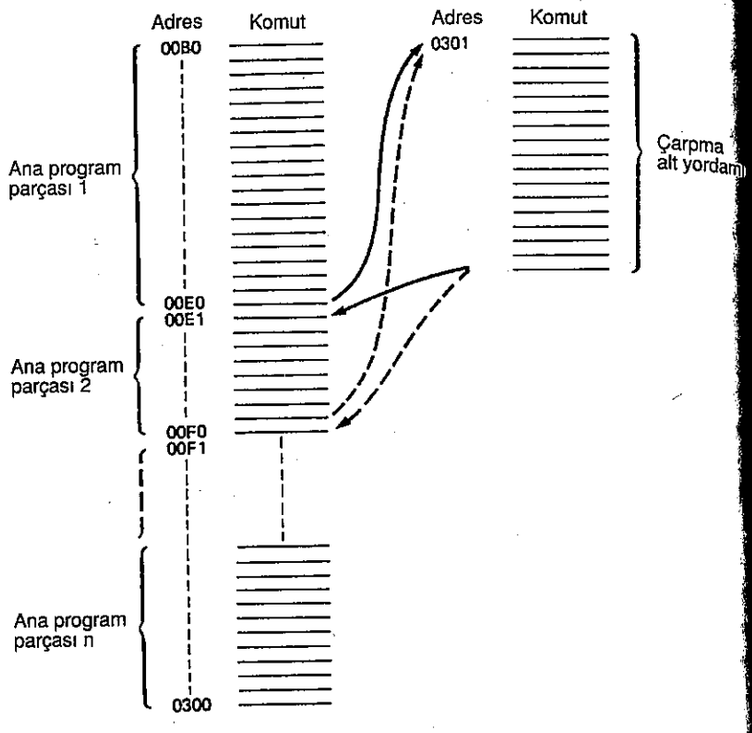
Altyordam kullanarak program çalıştırılması

Çarpma alt yordamını oluşturan program kodu, bilgisayar belleğine sadece bir kez yerleştirilmesine rağmen, gerçekleştirdiği çarpma işlemi, tıpkı program içerisine defalarca yazılması gibi, yine aynı sayıda tekrarlama gerektirecektir. Bu nedenle, programın yürütme sırası, Şekil 5.2'de olduğu gibi oklarla gösterilmiştir.

'Ana program parçası 1', daha önce olduğu gibi, iki sayının değerlerini hesaplar. Bu sayıları birbirleriyle çarpmak için, çarpma alt yordamının yürütülmesi, yani, alt yordam komutlarının, 'ana program parçası 2' başlamadan önce çalıştırılması gerekir. Bu nedenle, program yürütümünün, 00E0 adresinden, alt yordamın başlangıç adresi olan 0301'e aktarılması gerekir. Bu, şekil 5.2'de, üstteki kesiksiz çizgili okla ile gösterilir ve 'altyordama *giriş* ya da *çağırma*' olarak adlandırılır.

Bunun için, ileride 7. bölümde ayrıntılı olarak izah edilecek olan özel bir 'çağırma' komutu, 00E0 adresine yerleştirilir.

Alt yordamın sonunda, istenen çarpım işleminin ardından, program yürütülmesi, 'ana program bölümü 2'yi devam ettirmek için 00E1 adresine döndürülmelidir. Bu nedenle, alt yordam kodunun sonuna yerleştirilen, özel bir '(geriye) dön' komutu olmalıdır. Bu da yine, 7. bölümde ele alınacaktır.



Şekil 5.2

Program yürütmesinin, alttaki kesiksiz çizgili okla gösterildiği gibi, 00E1 adres dönmesinin ardından, bir sonra gelen iki değer, 'ana program parçası 2' tarafın hesaplanır ve çarpma alt yordamı, tekrar 00F0 adresinden girilir. Bu da Şekil 5.2 üst taraftaki kesikli ok ile gösterilmiştir.

Çarpma işleminin sonunda, yürütme bir kez daha, altta kesikli ok ile gösteril gibi, ana programa, ancak bu sefer 00F1 adresine döndürülür.

Çarpma altyordamı, gerek duyuldukça tekrar tekrar kullanılabilir. Gerekli olduğ da ana programdan çağrıldığından ve program akışı her zaman alt programlar ana programa doğru döndüğünden, altyordamlar, bir anlamda, ana programa ğımlıdır. Bu nedenle, ana programın bir *alt programı* olarak adlandırılırlar.

Pratikte büyük programlar, yalnızca tek bir altyordam kullanımıyla sınırlandırılmaz. Çoğu kez programlar, bir hesaplama sırasında her biri birkaç kez kullanılan, birçok alt yordam içerirler. Dahası alt yordamlar, pek çok farklı amaç için kullanılabilirler. Sadece aritmetik işlemler gerçekleştirmekle kalmazlar. İleride verilen örneklerde diğer uygulamalarından bazılarını göreceğiz.

İç içe altyordamlar

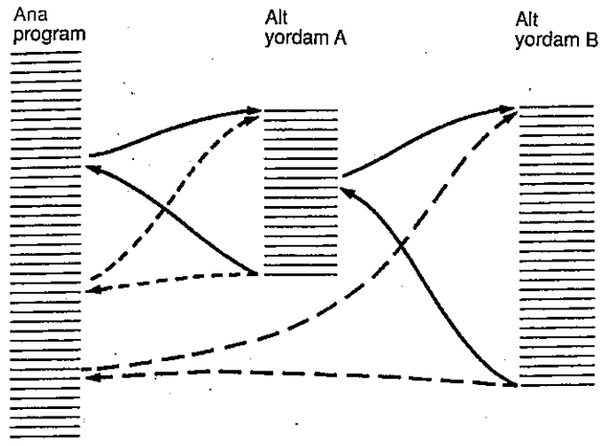
Şekil 5.2'de, tek *düzeyli* bir alt yordamlama, yani bir ana programın, her birinin görevlerini tamamladıktan sonra denetimi ana programa aktardığı, bir ya da daha fazla altyordamı çağırdığı durum gösterilmiştir. Ana program, programın en üst düzeyi ve bağımlı alt yordamlar da onun bir düzey altı olarak düşünülebilir.

Ancak genelde, daha fazla altyordamlama düzeyine ihtiyaç duyulur. Şekil 5.3'de, ana programın, kendisi de diğer bir altyordamı çağıran, bir altyordamı çağırdığı bir program yapısı görülüyor. Bu nedenle, burada iki düzeyli altyordam kullanılmıştır. Ana program, A altyordamını her çağırdığında, o da B altyordamını çağıracaktır. Bu nedenle, program akışı, ana programdan A altyordamına, oradan da B altyordamına biçiminde olur.

B altyordam işlemini tamamlar ve B'ye giriş yapılmasına neden olan komutun ardından gelen komutta, denetimi A alt yordamına aktarır. Ardından, A altyordamı (B işleminden elde edilen sonuçları kullanarak) işlemini tamamlar ve daha önce olduğu gibi denetimi yine ana programa aktarır.

Alt yordamların diğer alt yordamları çağırdığı bu işleme, alt yordamları *iç içe geçirme (nesting)* denir. Genel olarak, bu herhangi bir *derinlikte* olabilir. Yani, A altyordamı B'yi, B altyordamı C'yi, C ise D'yi.... çağırabilir. Dönüş ise, daha önce olduğu gibi, D'den C'ye, C'den B'ye ve B'den de A'ya şeklindedir.

Şekilde görüldüğü gibi, B altyordamının sadece A altyordamından çağrılması gerekmez. Doğrudan ana programdan da çağrılabilir. Bu tür bir durum, daha önce de tartışıldığı gibi, B bir çarpma altyordamı ve A da, sözgelimi, karekök alan bir altyordam olduğunda ortaya çıkabilir. B altyordamı ise, bir sayının karekökünü almaya ilişkin çarpanlar da dahil olmak üzere, çarpma işlemlerine gerek duyulan herhangi bir anda, genel bir uygulanabilirliğe sahiptir.



Şekil 5.3

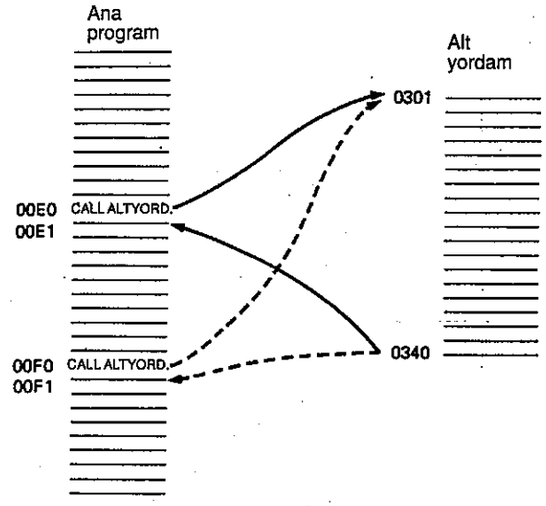
Alt yordama giriş ve çıkış

Yukarıdaki tanımlamalar, çok esnek bir biçimde hem ana programdan hem birbirlerinden giriş yapabilen bir altyordam grubu olduğunu ortaya koymak Bunun nasıl gerçekleştirileceği, 7. bölümde daha ayrıntılı olarak tartışılaca Burada dikkat edilmesi gereken nokta, bunun için özel bir mekanizmaya ihti duyulacağıdır. Basit dallanma komutlarının kullanımı, gereken denetim aktarım olanak tanımaz. Aşağıdaki örnek, bu noktayı açıklığa kavuşturacaktır.

Şekil 5.4'ü ele alalım. Bu diyagram, Şekil 5.2'dekine çok benzemektedir. A programın, 00E0 ve 00F0 adreslerindeki komutları kullanmak suretiyle iki kez yaptığı bir altyordamı kullandığı varsayılmıştır. Alt yordamın, denetimi, 00E ilk girişten ve 00F1'e ikinci girişten sonra aktarması istenmektedir.

Altyordama 00E0 adresinden giriş yapma, 00E0 adresine bir '0301'e dall komutu yerleştirerek yapılabilir. 00E1'e geri dönüş ise, altyordamı 0340 adres deki bitimine "00E1'e dallan" komutu yerleştirerek gerçekleştirilebilir.

Altyordamlama, 00F0 adresinden ikinci giriş ise, 00F0 adresine '0301'e dall komutu yerleştirerek gerçekleştirilebilir. Fakat, altyordamdan dönüş, 0340 adres de zaten '00E1'e dallan' komutu bulunduğu için mümkün değildir. Ve kuşkusu 00E1 adresi, alt yordamdan ikinci çıkış için, doğru dönüş noktası değildir. Doğ nokta, gerçekte, 00F1 adresidir.



Şekil 5.4

Bu nedenle, dallanma komutlarının, altyordama giriş ve altyordamdan çıkışlar için kullanımı, bu örnek için seçilen tek düzeyli altyordamlama gibi basit bir durumda bile hataya yol açar. Dallanma, daha karmaşık altyordam yapıları içinse tümüyle yetersiz kalır.

S 5.3 Bu bölümde daha önce sözü geçen özel altyordam giriş komutu "CALL" ve dönüş komutu "RETURN", bu sorunun tümüyle üstesinden gelir.

Altyordam Kullanmanın Avantajları

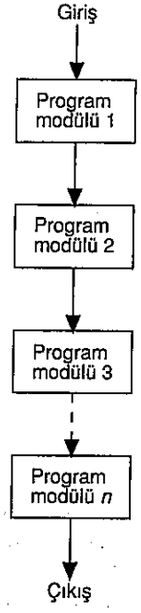
Altyordamların bir program içinde kullanılması pek çok avantaj sağlar. Her şeyden önce, altyordamı oluşturan komutlar, programda sadece bir kez yazıldığından, programın uzunluğu azalacaktır. Dolayısıyla, *amaç program* için gerekli bellek alanı miktarı en alt düzeye inmiş olur. (Amaç kodu, programı oluşturan makine kodu komutlarının listesidir.) İkinci olarak, program yapısı geliştirilmiş olur. Hatadan kolaylıkla arındırılacak ve düzeltilecek programlar yapma yöntemlerinden birinin, onları *modüler* yapmak olduğu genelde kabul edilmiştir. Bu nedenle, bir programı modüller dizisi olarak yazmak iyi bir programlama yöntemidir. Her bir modül, bir program kodu bloğundan ibarettir ve tercihan, Şekil 5.5'de görüldüğü gibi, tek bir giriş ve tek bir çıkış noktasına sahip olmalıdır.

Altyordam kullanımı, modüler programların gelişimi eder. Her altyordam bir modül oluşturur ve genell birinin sadece birer giriş ve çıkış noktaları vardır.

Bu türde bir altyordam dizisi halinde komple bir p yazmak mümkündür. Bu yapılırsa, ana program, sad yordamların çağrılacağı sırayı denetleyen bir çağırma l ları listesine indirgenmiş olur.

Hatalardan ayıklama, yani bu tür bir programdaki h bulmak ve düzeltmek, program boyunca denetim akışı ifade edilmiş olduğu için, çok daha kolaylaşmıştır.

Altyordam kullanımının üçüncü avantajı, programın o *bilirliğinin* artırılmış olmasıdır. Bir programcı, program sini inceleyebilir ve eğer program bir altyordam dizisi şe yazılmışsa, ne yaptığını kolaylıkla anlayabilir.



Şekil 5.5

Genellikle, altyordamlara, işlevlerini ifade eden isimler ve Böylece, daha önce sözü edilen çarpım altyordamı, CARP ve zaman gecikmesi altyordamı GECIK olarak adlandırıl lir. Bu altyordamları bir çağırma dizisinden ibaret olan programla okuduğumuzda:

```
CALL CARPIM
CALL KAREKOK
CALL GECIK
vb.
```

programın ne yapmak istediğini anlamak çok kolaydır. kesinlikle, altyordamlar kullanılmayıp yerine, çağrı kullan dığında ortaya çıkacak olan ayrıntılı komut dizilerini okum tan çok daha kolay olacaktır.

Özel bir altyordam modülünün çalışmasını ayrıntılı bir biçim de denetlemek gerektiğinde, o modülün kodunu incelemek zaman mümkündür. Bu nedenle, programı modüllere ayırma programı ayrıntılı bir şekilde inceleyebilme olanağını kay betmemize neden olmaz.

Kütüphaneler

Altyordam kullanımı bir önemli avantaj daha getirir; programın önemli bölümleri *kütüphanelerden* oluşturulabilir.

Bilgisayar yazılımının geliştirilmesi sürecinde, bir bilgisayar için yararlı yordamların, yalnızca onları üreten programcılar tarafından değil, aynı zamanda bilgisayar sisteminin diğer kullanıcıları tarafından da kullanılabilmesi için, saklanmasının çok faydalı olacağını anlaşılmıştı. Örneğin, daha önce sözü edilen bir çarpım yordamı, pek çok uygulamada ihtiyaç duyulabilecek bir kod parçasıdır.

Altyordamlar, bu tür genel uygulamalar için çok uygundur. Az önce de ana hatlarıyla gösterildiği gibi, programa herhangi bir yerden girilebilir veya herhangi bir yerden çıkılabilir. Eğer bilgiler (parametreler), bir sonraki kısımda tanımlandığı gibi, uygun ve esnek bir biçimde değiş-tokuş edilebilirse, ana programa giriş daha kolay olacaktır.

Bu nedenle, bir sistemde, *kütüphane*de pek çok altyordam grubu bulundurulması mümkün olacaktır. Kullanıcılara, mevcut yordamların bir listesi verilir ve onlar da bunlardan bazılarını istedikleri gibi kullanırlar.

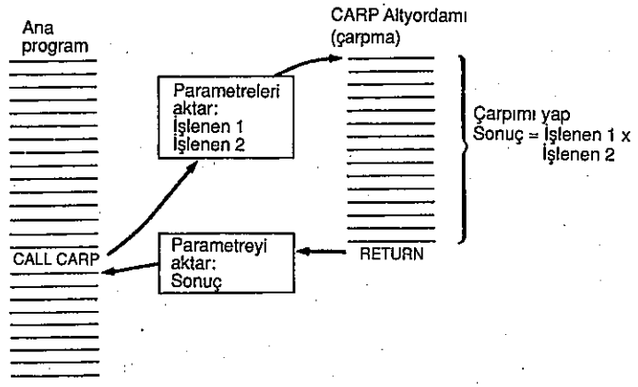
5.3 Parametre aktarma

Parametre aktarma gereği

Altyordam kullanımının şimdiye kadar ele alınmamış bir yönü, parametre aktarma sorunudur. Çağrıldığı zaman bir altyordama *parametre* aktarmak ve işlemini tamamladığında da ondan bilgiyi geri almak gerekir.

Bir önceki kısımda ele alınan çarpma yordamı, iki giriş değerini (çarpımları bulunacak olan iki sayıyı) alır ve bir sonuç değeri üretmek üzere bunları çarpar. Bu nedenle, ana programdan altyordama girişte, o program için, iki işlenenin değerlerini altyordama aktarmak gereklidir.

Altyordam, RETURN komutunu kullanarak denetimi, ana programa döndürdüğünde, işlemin sonucunu da, yani giriş değerlerinin çarpımını da aktarmak gerekir. Bu nedenle, bu altyordamın iki giriş ve bir de çıkış parametresi vardır. Bazen, bu parametreler, altyordam *argümanları* olarak adlandırılır.



Şekil 5.6 : Parametre aktarma

Parametre-aktarma işlemi, Şekil 5.6'da gösterilmektedir. Altyordamlara ş
altyordamlarda ters yönde aktarılan parametrelerin tipi ve sayısı çok çeşitli ol
Yukarıdaki örnekte de ifade edildiği gibi, genellikle sadece birkaç veri
değiş-tokuş edilir. Bununla birlikte, bazen, matris işleyen bir altyordamın
munda olduğu gibi, çok sayıda parametreler de kullanılabilir.

Bazen, aktarılacak parametreler sadece sayısal değerler olmayabilir. Çevre
aktarımı gerçekleştirecek bir altyordam, örneğin, kullanılacak birimi ve bel
birimin işlem modunu ifade eden bilginin aktarılmasına gerek duyabilir. Bu
metreler, alt programın (ya da denetlediği birimlerin) nasıl çalıştığını tanıml

Parametre aktarma yöntemleri

Parametrelerin, bir altyordama aktarılabileceği çok çeşitli metotlar vardır.

Bunların ilki, değiş-tokuş edilecek parametreyi (parametreleri) tutmak için
da daha fazla işlemci kaydedicisi kullanmaktır. Intel 8085 mikroişlemcisi
alalım. Daha önce görüldüğü gibi, bu işlemci A, B, C, D, E, H ve L kaydedici
sahiptir. Bir altyordama aktarılacak bir parametreyi saklamak için bunlardan
hangi biri kullanılabilir.

Bu nedenle, çarpma altyordamını çağırmak için, bir program:

(a) İşlenen 1'i, B kaydedicisine ve
(b) İşlenen 2'yi, C kaydedicisine yerleştirir.
(c) Altyordamı çağırır.

Dönüşte, alt program:
(a) Sonucun üst sıralı 8 bitini B kaydedicisine,
(b) Sonucun alt sıralı 8 bitini C kaydedicisine koyar
(c) Ana programa geri döner.

Sonucu ana programa göndermek için, 2 kaydedici (B ve C) kullanılmıştır, çünkü 8-bitlik iki sayının çarpım sonucu, genellikle 16 bitlik bir sayıdır.

Yukarıda, B ve C kaydedicilerinin kullanılmasının özel bir nedeni yoktur. Bunların yerlerine, A ve B, veya D ve E de kullanılabilir.

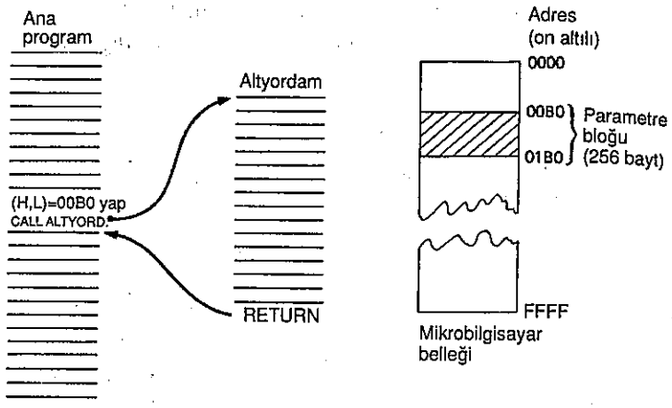
Bir ana program ile altyordam arasında, işlemci kaydedicileri kullanılarak parametre değiş-tokuşu yapılacaksa çok dikkatli olunmalıdır. Çünkü kaydediciler, hesaplama aşamasında, hem ana programda hem de altyordamda ara sonuçları tutmak için kullanılabilir. Bu nedenle, bir altyordama aktarılan her parametre, altyordamdaki kaydediciler kullanılmadan ve üzerine bir başka değer yazılmadan önce, bellekte saklanmalıdır.

Parametreleri altyordama aktarmak için, işlemci kaydedicilerini kullanmak, sadece küçük miktarda bir bilgi içerdiği sürece, yeterli bir yöntemdir. Bununla beraber, eğer 64 x 64 (4.096) değeri içeren bir matris gibi, çok fazla bilgi işlemek için altyordam kullanılacaksa, başka bir yönteme ihtiyaç olacaktır.

Bu durumda, altyordama aktarılacak bilgi, mikrobilgisayar belleğine saklanır. İlgili bellek alanının ilk konumunun adresini işaret eden bir *işaretçi*, kaydedicideki altyordama aktarılır. Şekil 5.7'de bu düzenleme gösterilmiştir.

8085 mikroişlemcisinde, H ve L kaydedicileri, normalde bellek adreslerini tutmak için kullanılır. Bu nedenle, altyordama aktarılacak parametreler 00B0-01B0 (256 baytı gösterir) bellek konumlarına saklanırsa, H ve L kaydedicilerine, sırasıyla 00 (on altılı) ve B0 (on altılı) değerleri atanır.

Diğer bir deyişle, H, parametre bloğunun ilk baytının adresinin en büyük değerlikli yarısını, L ise bu adresin en küçük değerlikli diğer yarısını tutar. Bu genellikle, "H,L çifti, parametre bloğunun işaretçisini tutmak için kullanılır" şeklinde ifade edilir.



Şekil 5.7 Parametre geçişi

Altyordam kolaylıkla parametrelere erişebilir. Örneğin, 8085'deki

```
MOV     (kaydedici), M
```

komutu, (burada (kaydedici), A, B, C, D veya E olabilir), verilerin, adresi H ve L kaydedicilerinde saklı olan bellek baytından alınmasını sağlar ve veriler, A veya C, vb. kaydedicilerden birine yerleştirir. H,L kaydedici çiftini artıran (veya ekleyen) bir,

```
INX     H
```

komutu da vardır. Bu nedenle, 00B0'den 01B0'ye kadar parametreler boyunca parametre değerlerini birer birer bir kaydediciye koyarak ilerlemek oldukça kolay olacaktır.

Altyordam, büyük miktarlardaki bilgileri, ana programa geriye göndermek için aynı yöntemi kullanılır. Alt yordam:

1 Bellekte hesaplamış olduğu sonuçları uygun bir konuma yerleştirir.
2 H ve L'nin içeriklerini, bu bellek alanının ilk adresini gösterecek şekilde ayarlar.
3 Denetimi ana programa aktarır.

Belleğin, ana program ile altyordam arasındaki parametre değiş-tokuşunda kullanımı, yalnızca büyük miktarlarda veri aktarımı ile sınırlı değildir. Birkaç bayt

aktarılacağı zaman da kullanılabilir. Ancak bu, parametreleri saklamak, H ve L kaydedici çiftine değerler atamak ve parametreleri geriye getirmek zaman alıcı işlemler olduğundan, programın çalışırken harcadığı sürenin artmasına neden olmak gibi bir sonucu getirir.

5.4 Verilerin saklanması ve geriye alınması

İşlemci kaydedicileri

Bir hesaplama işlemi süresince, hem ana program hem de altyordamlar, büyük bir olasılıkla işlemci kaydedicilerini kullanırlar. Bu nedenle denetimin, programın bir bölümünden diğer bölümüne aktarımı, örneğin ana program tarafından bir altyordam çağrısı yapıldığında, çağrı yürütülmeden önce bu kaydedicilerde saklı bulunan değerlerin korunması önemlidir. Ardından, altyordam çalıştıktan sonra, denetim ana programa aktarıldığında, kaydedici içeriği tekrar eski değerine getirilebilir ve program, çalışmaya tam olarak kaldığı yerden devam edebilir.

Kuşkusuz, eğer altyordam, kaydedicileri hiç kullanmazsa, yukarıda belirtildiği gibi, içeriklerinin saklanmasına da gerek yoktur. Özellikle altyordam belli bir karmaşıklık düzeyinde hesaplamalar yapıyorsa, hiç kaydedici kullanılmaması söz konusu olmaz.

Program sayıcısı ve altyordamlar

Program sayıcısı, daha önceki bölümlerde (Kısım 2.1) izah edildiği gibi, işlemci içerisinde, özel kullanım amacı olan bir kaydedicidir. Program sayıcısı, daima, yürütülecek *bir sonraki* komutun adresini tutar. Bu nedenle, bilgisayarın komut-getirme saykılı süresince, elde edilecek komutun bellek adresi olarak kullanılır.

Altyordam kavramı açısından program sayıcısının özel bir önemi vardır. Şekil 5.4'den de görülebileceği gibi, bir altyordamın denetimi aktarması gereken adres, altyordama giriş yapılmasını sağlayan CALL komutunu izleyen adrestir.

Örneğin, 00E0 adresinde bulunan CALL komutu yürütüldüğünde, dönüş adresi 00E1 adresidir ve bu değer, program sayıcısının çalışma mantığından dolayı, içeriğinde tutulan sayıdır. Benzer biçimde, 00F0 adresindeki CALL komutu yürütüldüğünde de program sayıcının içeriği 00F1 adresi olacaktır. Yine, bu değer de, altyordamın denetimi aktaracağı adrestir.

Bu nedenle, eğer program sayıcısındaki değer, her altyordam çağrısı yürütmes
saklanırsa, doğru dönüş adresi otomatik olarak korunmuş olacaktır. O hal
altyordamdan bu adrese dönüş, saklanan adresin, altyordam kodunun sonun
RETURN komutu tarafından program sayıcısına yüklenmesiyle gerçekleşt
Bu işleme ilişkin örnekler ileride verilecektir.

Özet

Yukarıdaki kısımları kısaca özetlersek, bir altyordamın bir ana programdan (v
herhangi bir programdan) çağrılması durumunda, çağrı yapan program, ger
CALL komutunun yürütülmesi öncesinde, program sayıcı da dahil olmak üz
işlemci kaydedicilerinin içeriklerini saklamalıdır.

Ana programa dönüş, programın, çalışması nerede kesilmişse tam olarak o no
dan itibaren devam edebilmesi için, saklanmış bulunan kaydedici değerler
altyordamın sonunda yeniden yüklenmesiyle gerçekleştirilebilir.

Altyordamların dezavantajları

Altyordam kullanmanın bazı avantajları daha önce ele alınmıştı. Birçok şey
olduğu gibi bunda da bazı dezavantajlar vardır.

İlk olası dezavantajından, Kısım 5.3'de söz edilmişti: altyordam kullanımı prog
mın çalışma süresini uzatır, ya da diğer bir deyişle, yordamın çalışma hızını azal
Bunun nedeni gayet açıktır. Altyordama giriş yapılmadan önce, çağıran prog
tarafından denetlenen işlemci, kaydedici değerlerini saklamalıdır. Bu değe
üzerinde değişiklik yapmış olabilen altyordamdan geriye dönüş yapılmadan
de, tekrar geriye yüklenmelidir.

Bu nedenle, altyordam kullanımının getirdiği kaçınılmaz *yükler* olacaktır. Sakl
ve tekrar geriye yükleme işlemleri zaman kaybına ve programın çalışma süres
artmasına neden olur. Programın çalışma süresinin artması, tabii ki her zaman
olmayabilir. Programların tümünde, süre konusunda bir sınırlama yoktur. Ge
likle, sadece *gerçek-zamanlı* sistemlerde, yani mikrobilgisayar dışında herhang
biçimde zamanla ilgili sınırlamaları bulunan bir başka işlemle karşılıklı işlem ya
programlarda bu tür kısıtlamalar bulunur. Bu tür bir işleme güzel bir örnek,
mikrobilgisayarın zamana göre değişen analog sinyali işlemek için kullanılması

İkinci dezavantaj ise; altyordam kullanımının fazladan bellek alanı gerektirmesidir. Kaydedici içerikleri ve belki, denetimin bir altyordama aktarılması esnasında saklanması gereken diğer değerler de, belli bir yerde saklanmalıdır. Genellikle bunlar da belirli bir bellek alanını işgal ederler.

5.5 Yığınlar konusuna giriş

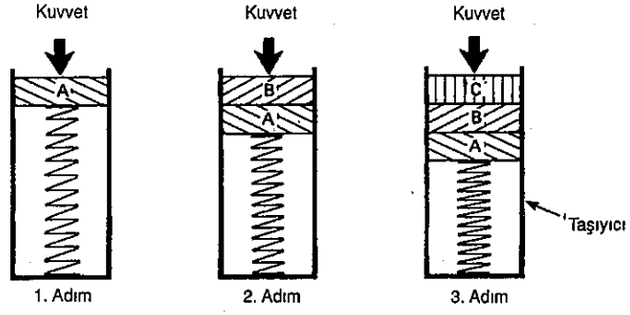
Şu ana kadar, kaydedici içeriklerinin, bir altyordama giriş yapılmadan önce saklanması gerektiği belirtilmiş; fakat, bu saklamanın nasıl gerçekleştirileceği açıklanmamıştı. İşte bu bağlamda, *yığın*lar önemli olmaktadır.

Bir mekanik yığın

Bir yığını gözümüzde canlandırmanın iyi bir yolu Şekil 5.8'de gösterilmektedir. Bu diyagramda yığın, bir kap şeklinde temsil edilmiştir. Diyagramın solundaki birinci aşamada, yığının yani kabın içine bir nesne konur. Bu, A nesnesidir ve kabın üst kısmında, kendisiyle kabın tabanı arasındaki bir yay aracılığıyla tutulmaktadır.

2. aşamada ise, diğer bir B nesnesi yığına konmuştur. B nesnesini eklemek basit bir işlemdir; B, A'nın üzerine konur ve her ikisi (A ve B birlikte), yay, B'nin üst yüzeyi kabın üst kenarlarıyla aynı düzeye gelinceye kadar aşağıya bastırılmak suretiyle kabın içine itilir.

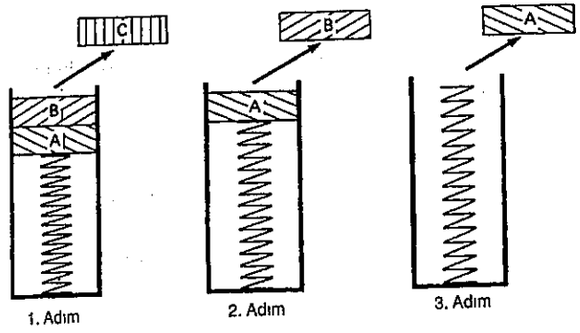
3. aşamada, diğer bir C nesnesi, aynı biçimde yığının üzerine eklenir. Böylece, yığına en son eklenen nesne, daima en üstte, ilk konulan nesne de daima en altta



Şekil 5.8 Mekanik bir yığın

kalmaktadır. Tutulabilecek nesne sayısı, yığının büyüklüğü ile, bu örnekte, ... sığabilecek nesne sayısı ile sınırlı olmaktadır.

Nesnelerin yığına konulma şekli nedeniyle, bir nesneyi saklama işlemi, yığına ... olarak bilinir.



Şekil 5.9 Yığından çıkarma

Nesnelerin yığından geriye alınması ise, yığından (dışarı) *çıkartılarak* ya da ... *lerek* yapılır. Şekil 5.9'da, bu işlem gösterilmiştir. Bu şekil, Şekil 5.8'in ters... Yığının tepesindeki kuvvet kaldırıldığında, üstteki nesne (C) kabın dışına çıkar... geri alınmış olur. B yığının en üstüne gelecek biçimde, B ve A bir kademe yu... çıkarlar. Bir sonraki adım olarak, A yığının tepesine ilerlerken, B yığından çık... labilir; son olarak A da yığından çıkarılabilir ve yığın boşalır. Çekme ve çıkar... yığından veriyi çıkarma işleminin alternatif isimleridir.

Yukarıdaki açıklamalar ve diyagramlardan da anlaşılacağı gibi, yığın *son-g... ilk-çıkar düzeninde* çalışan *bir bellektir*. Kaba itilecek olan en son nesne, yani Ş... 5.8'in 3. adımındaki C, Şekil 5.9'un 1. adımında ilk geriye getirilecek olandır. S 5.5 tür bir birime, kısaca LIFO (son-giren ilk-çıkar) bellek denir.

Mikroelektronik yığınlar

Bir mikrobilgisayar sisteminde, kuşkusuz yukarıda tanımlanan türde bir y... kullanmak uygun değildir. Pratik mikroelektronik yığınlar çeşitli biçimlerde ... çekleştirilebilir. Bunlardan biri, mikroişlemci yongası içinde, yonga fabrik... üretilirken LIFO bellek olarak işlev görecek özel bir bellek alanı ayırmaktır.

başkası ise, normal RAM'in bir bölümünü, bir *yığın işaretçisi* olarak bilinen özel bir kaydedici ile birlikte bir yığın olarak kullanmaktır.

Bu olasılıklar, bir sonraki bölümde daha ayrıntılı bir biçimde ele alınacaktır.

Yığın uygulamaları

Yığınlar, bir altyordamdan dönüşe olanak tanıyabilmek üzere gereken program sayıcısı değerlerini saklamak için çok uygundurlar.

Bir programda CALL komutu ile karşılaşıldığında, program sayıcısındaki değer, o komutun yürütümü sırasında otomatik olarak yığına atılır. Altyordamın bitiminde bir RETURN komutu ile karşılaşıldığında da, yığının üstündeki değer dışarı çıkarılır ve program sayıcısına yüklenir. Bu, altyordam çağrısını izleyen komutun ardından, tekrar ana programa giriş yapılabilmesini sağlar.

Kesme yönetimi ve iç içe altyordam program yapılarında yığın kullanımı ilerki bölümlerde ele alınacaktır. S 5.4, 5.9, 5.11

Sorular

5.1 Uzun bilgisayar porgramlarının gelişiminde altyordamların neden önemli olduğunu açıklayın. Bir ya da daha fazla altyordam kullanmanın uygun olacağı durumlara örnekler verin.

5.2 Altyordam kullanımının temel avantaj ve dezavantajlarını sıralayın. Her birini, uygun örnekler vererek gösterin.

5.3 CALL ve RETURN komutları kullanarak, bir altyordama nasıl giriş ve çıkış yapılabileceğini tartışın. Yalnızca mikrobilgisayar komut takımındaki normal BRANCH ve JUMP komutlarını kullanarak, altyordama giriş ya da çıkış yapmanın mümkün olmadığını gösterin.

5.4 Altyordamların "iç içe geçmesi"nden ne anladığınızı açıklayın. İç içe geçmiş altyordamlara giriş ve çıkışların, bir CALL komutunun yürütümünde, program sayıcısının içeriğini saklamak suretiyle gerçekleştirilebileceğini gösterin. Pratikte bu tür bir saklama nasıl gerçekleştirilebilir?

5.5 Yığın ne demektir? Altyordama giriş ve altyordamdan çıkışa olanak tan
bilmek için, bir yığının, program sayıcısının içeriğini saklamak üzere n
kullanılabileceğini gösterin. PUSH ve POP terimlerinin anlamlarını, yığ
içerikleri üzerinde işlem yapmak için kullanımları açısından kısaca açıkla

5.6 Bir altyordama girilmeden önce, işlemci kaydedicilerinin içeriklerinin n
saklanması gerektiğini açıklayın. Cevabı örneklerle gösterin.

5.7 $y = 1 + 3x + 4x^2 + 7x^3$

ifadesinin, 0 ile 20 arasındaki x değerleri için nasıl hesaplanacağını göst
bir akış diyagramı çizin. Elinizde bir çarpım altyordamın olduğunu varsa
ve yukarıdaki hesaplamada bunun nasıl kullanılabileceğini gösterin.

5.8 100 sayılık bir sayı kümesi, 0B00 (on altılı) ile 0B63 (on altılı) adres
arasındaki bir mikrobilgisayar belleğine yerleştirilmiştir:

$$\text{Ortalama} = \frac{1}{100}\sum_{n=1}^{100} x_n$$

$$\text{Değişim} = \frac{1}{100}\sum_{n=1}^{100} x_n^2$$

formüllerini kullanarak, bu sayıların ortalamalarını ve değişimlerini hesap
mak için nasıl bir altyordam yazılabileceğini açıklayın.
Giriş değerlerinin altyordama ve çıkış değerlerinin ana programa hangi yö
temle aktarılacağını ana noktalarıyla gösterin.

5.9 Ek 1'de verilen Intel 8085 komut takımını kullanarak, Soru 5.8'de sözü edi
parametrelerin aktarılmasında gereken program kod parçalarını yazın.

5.10 Soru 5.8'deki altyordam için bir akış diyagramı çizin. Eğer bu altyordam, In
8085 mikroişlemcisini çalıştırmak için yazılırsa, bu altyordamın başka he
hangi bir altyordamı çağırmasına gerek olacak mıdır?

5.11 1000 sayılık bir sayı kümesi bir mikrobilgisayar belleğine yerleştirilmişti
Bu sayıların en büyük ve en küçüğünü bulacak bir programın akış diyagram
çizin. Bu programın, altyordam olarak yazılmaya uygun herhangi bir parç
var mıdır?

olüm 6 Yığın mekanizması

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

1 *Bir yığının mikrobilgisayar organizasyonunu tanımlayabilmeli,*
2 *Sağladığı saklama düzeyi sayısına göre, bir yığının kapasitesini tanımlayabilmeli,*
3 *Yığının esnek bir veri saklama birimi olduğunu ve tek tek değerleri ya da veri bloklarını tutmak için kullanılabileceğini kavrayabilmeli,*
4 *Bir altyordamdan dönüş adresini saklamak üzere yığının nasıl kullanıldığını tanımlayabilmeli,*
5 *Bir altyordamı çağırmadan önce, altyordamdan çıkışta bu değerleri tekrar geriye yükleyebilmek için işlemci kaydedicisinin içeriğini saklamak üzere yığının nasıl kullanılabildiğini anlayabilmeli,*
6 *İç içe altyordamları desteklemek için kaydedici değerlerini ve dönüş adreslerini tutmada yığın kullanımını anlayabilmeli,*
7 *Yığınların, elverişli bir geçici veri saklama alanı olduğunu kavrayabilmeli,*
8 *Yığın içerikleri üzerinde işlem yapmak için kullanılan PUSH ve POP komutlarının işleyişlerini anlayabilmeli,*
9 *Yığınların, ya*
   (a) mikrobilgisayar sisteminin işlemci yongasındaki kaydediciler, veya
   (b) sistemin rastgele-erişimli belleğindeki kaydedicilerle gerçekleştirilebileceğini kavrayabilmeli,
10 *İşlemci yongasındaki yığınların sınırlı, ancak RAM içindeki yığınların, çok geniş bir kapasitede olabileceklerini kavrayabilmeli,*
11 *Bir yığın işaretçisinin,*
   (a) bir PUSH komutu ve
   (b) bir POP komutunun yürütülmesi sırasındaki
   işlevini ve işleyişini açıklayabilmeli,
12 *İşlemci yongasındaki yığının tersine, RAM'de yığın kullanmanın avantajları ve dezavantajlarını değerlendirebilmeli,*
13 *Yığın içeriklerinin üzerinde işlem yapmak üzere kullanılabilen tipik komutlar ve bunların kullanım biçimlerini değerlendirebilmelisiniz.*

6.1 Yığın, son-giren-ilk-çıkar (LIFO) düzeninde bir bellektir.

5. Bölümde, yığın kavramına, mekanik bir benzeştirme kullanarak kısa bir giriş yapılmıştır. Yığına yerleştirilecek olan nesnelerin üstten içeri doğru nasıl itildiği

ve geri alınacak nesnelerin, yine üstten, dışarıya doğru nasıl çıkartıldığı aç
mıştı.

Yığın işlemi için kullanılabilecek diğer bir benzeştirme ise kağıtlardan oluş
demet ya da yığındır. Yığına konulacak bir bilgi, sözgelimi bir yaprak, ya
üzerine yerleştirilir. Bilgi geri alınacağı zaman, yapraklar yığının üzerinden sı
toplanır.

Kısım 5.5'de olduğu gibi, bu örnekten, yığının son-giren ilk-çıkar düzenin
bellek olduğu görülebilir. Yığına en son konan yaprak en üsttekidir ve dolay
bir geriye alma işleminde ilk çıkartılacak olan odur. Böylece, her zaman içe
dışarı çıkartılacak olan, yığına en son konan bilgi (son yaprak) olacaktır.

Yığının büyüklüğü, sahip olduğu *düzey* sayısı ile ifade edilir. Kağıt yığını örn
de, düzey sayısı, destedeki yaprakların sayısıdır; bir mikrobilgisayar yığınında
yığında tutulabilecek bilgi parçalarının sayısına karşılık gelir.

Örneğin, bir yığın 8-bitlik değerler tutacak şekilde tasarlanmışsa ve her biri 8
genişliğinde 16-bitlik bellek konumundan oluşmuşsa, yani 16 baytlıksa, bu
16-düzeyli yığın denir.

Veri blokları

Yığınlar sadece tek parça verileri tutmak için kullanılmaz. Çoğu zaman, mikro
gisayar sistemlerinde, birçok bilgi parçası bir araya getirilir ve sistemde bir gr
olarak işlenir.

Örneğin, ana programdan bir altyordam çağrıldığında, program sayıcı da de
olmak üzere, işlemci kaydedici içeriklerinin, altyordamın bitiminde çağıran pro
rama dönüşe imkan verecek biçimde saklanması gerektiğini görmüştük. İşlem
kaydedicileri, bu örnekte, bir bütün olarak saklanması ve geri getirilmesi gere
bir bilgi bloğu olarak ele alınabilir.

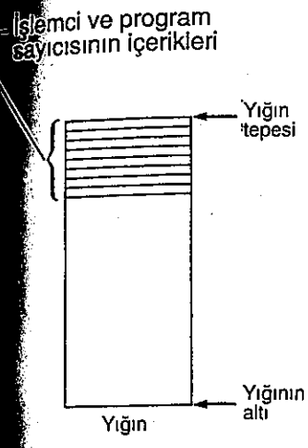
Blok, her biri bir işlemci kaydedicisinin içeriğini gösteren birçok bilgi baytını içe
(8085 ya da 6809 gibi 8-bitlik bir sistem kullanıldığı varsayılmıştır). 16-bitlik b
kaydedici olan program sayıcısının 2 bayta gereksinimi vardır.

Bu nedenle, bir altyordama girişte, sözgelimi, 10 baytlık bir veriyi saklam
gerekebilir. Bunlar :

1 bayt (A)
1 bayt (B)
1 bayt (C)
1 bayt (D)
1 bayt (E)
1 bayt (H)
1 bayt (L)
1 bayt (İşlemci Durum Kaydedicisi)
2 bayt (Program Sayıcısı) içindir.

Daha önce de belirtildiği gibi, (A), "A kaydedicisinin içeriği" anlamına gelmektedir.

Çoğu zaman, elbette, tüm işlemci kaydedicilerinin içeriğini saklamak gerekmez, çünkü ana program bunların hepsini kullanmıyor olabilir. Bu nedenle, içlerinden yalnızca birkaç tanesi geçerli veri içerebilir. Bununla birlikte, genellikle, birkaç kaydedicinin içeriğini bir blok içinde birlikte saklamak gereklidir.



Şekil 6.1

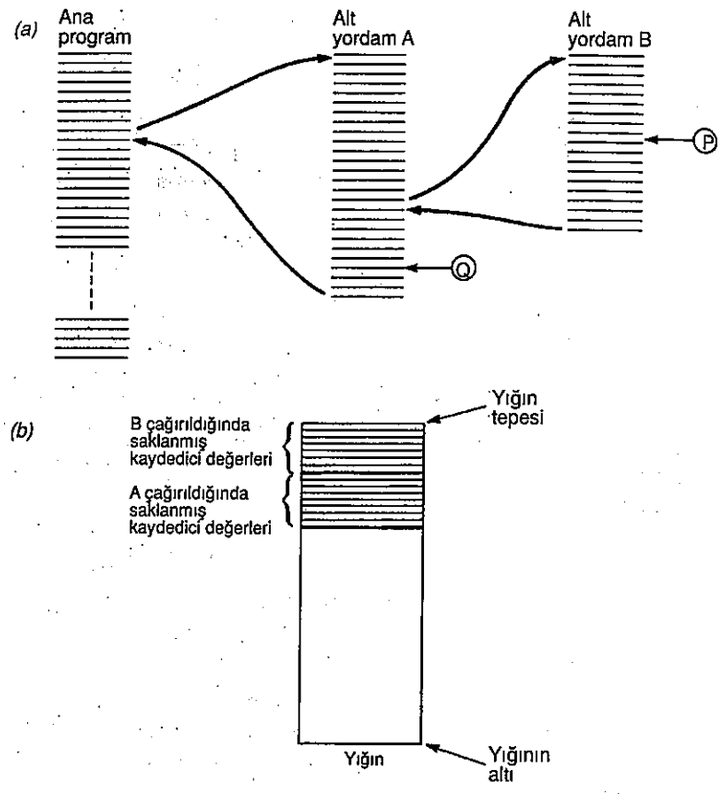
Bunu yapmak için, Şekil 6.1'de gösterildiği gibi, bilgi bloğu yığına birlikte yerleştirilebilir.

Bilgiyi saklamak için, daha sonra açıklanacağı gibi, bilgisayar komut kodunda bulunan komutlar kullanılarak, ayrı ayrı parçalar, yığına birbiri ardı sıra atılır. Veriyi dışarı almak için, yine mevcut komutları kullanarak, parçalar tek tek yığından dışarı alınır.

İç içe altyordamlarda yığın kullanımı

5. Bölümde, bir altyordamın diğer bir altyordamı çağırdığı iç içe altyordam kavramı, ana hatlarıyla anlatılmıştı. Yığının, son-giren ilk-çıkar düzenindeki işleyişi, bu tip işlemlerde çok kullanışlıdır.

Şekil 6.2 (a)'yı ele alalım. Burada, kendisi de bir başka altyordamı (B altyordamını) çağırmakta olan bir altyordamı (A altyordamını) çağıran bir ana program gösterilmektedir.



Şekil 6.2

Görüldüğü gibi, A altyordamı çağırıldığında, A'nın sonunda ana programa dönüş için, program sayıcısındaki değeri saklamak gereklidir. Benzer biçimde, B çağırıldığında da, B'nin bitiminde A altyordamına geri dönüş için, program sayıcısındaki değeri saklamak gerekecektir. Hem A hem de B çağırıldığında, işlemci kaydedicilerinden bazılarının içeriklerini de saklamak gerekebilir. Şekil 6.2 (b)'de yığın kullanılarak bunun nasıl yapılabileceği gösterilmiştir.

Ana program, A altyordamını çağırdığında içindeki komutlar, işlemci kaydedicisinin, yığında saklanması gereken içeriklerini yığına itmek için kullanılır. Altyordam çağrısı, program sayıcısındaki değerin saklanmasına neden olur. Böylece, denetimin A altyordamından ana programa geçmesi ile ilgili veri bloğu, yığının üzerine yerleşmiş olur.

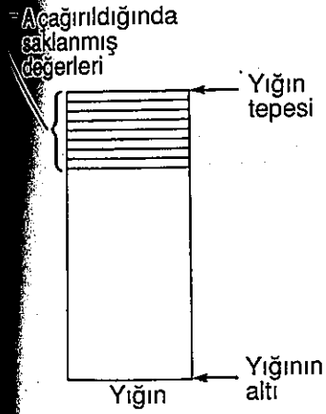
A'dan B altyordamı çağırıldığında da, benzer bir yöntem izlenir. A altyordamındaki komutlar, yığında saklanması gereken kaydedici değerlerini yerleştirmek için kullanılır. Ardından, B çağrısı, B'den A'ya dönüş için uygun olan program sayıcısı değerini yığına koyar.

Şekil 6.2 (b)'de, B altyordamına girildikten sonra, örneğin, program yürütmesinin, B içerisindeki bir P komutunda olduğu bir andaki yığın içeriği gösterilmektedir.

Yığının tepesindeki değerler, B'ye giriş yapılırken konulmuş değerlerdir, yani B sona erdikten sonra, A altyordamının kaldığı yerden devam etmesi için gerekli kaydedici içerikleri ve dönüş adresleridir.

A altyordamının bitiminde, ana programın kaldığı yerden devam etmesi için gereken kaydedici değerleri ve dönüş adresi yığının altına itilmiştir ve B'den A'ya dönüş için gerekli değerlerin altında bulunmaktadır.

B altyordamı yürütmesi tamamladığında, yığının tepesindeki değerler dışarı alınır. B'nin sonunda yer alan RETURN komutu, geriye alınan program sayıcısı değerinin (ki bu değer yığının en üstündeki konumlarda tutulmaktadır), program sayıcısına yerleştirilmesine neden olur. Bu, programın yürütülmesini, A altyordamında bulunan B çağrısının ardından gelen komuta aktarır. Sonra A altyordamındaki komutlar, saklanmış kaydedici değerlerini yığından çıkarmak ve bunları uygun kaydedicilere yerleştirmek için kullanılırlar. Bunun ardından da, A altyordamı çalışmasına, kaldığı yerden devam edebilir.



Şekil 6.3

Yığından her değer alınışında, altta saklı bulunan bilgi, Kısım 5.5'de de belirtildiği gibi, bir konum yukarı taşınır. Bu nedenle, programın yürütülmesi B altyordamından geriye dönüldükten sonra, A altyordamının içinde bir yerde, örneğin Q komutunda [Şekil 6.2 (a)] iken, yığının içerdikleri Şekil 6.3'de gösterildiği gibi olacaktır.

Yığının şu anda en üstünde bulunan değer, A'dan ana progra-

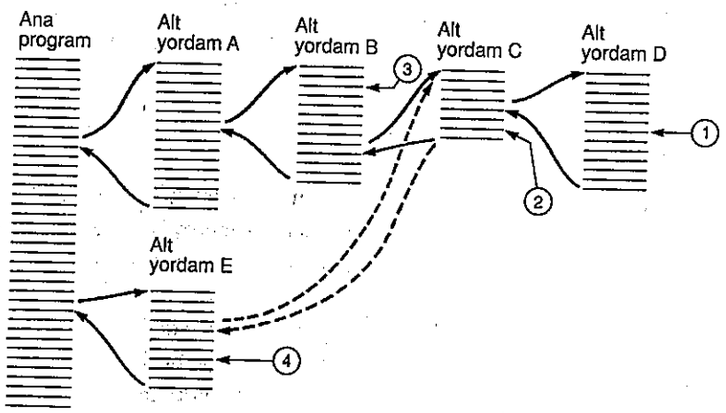
ma dönüş adresidir. Bu nedenle, A altyordamının sonundaki RETURN kom
bu değerin program sayıcısına yüklenmesini sağlar ve böylece denetim ana p
rama aktarılır. Bu programdaki komutlar, gerekli kaydedici değerlerini çıka
ve işlem (program) kaldığı yerden devam ettirilir.

Yığının, bir LIFO bellek biçiminde davranmasından dolayı, denetimin, ana p
ramdan A altyordamına, oradan B'ye ve sonra B'den A'ya ve oradan da
programa düzenli bir biçimde aktarımı otomatik olarak gerçekleştirilir.

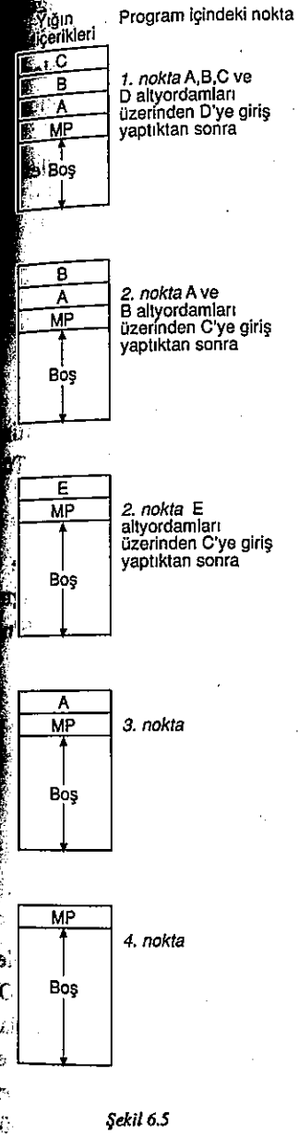
Sistem daha karmaşık yapılarda da doğru olarak çalışır. Örneğin, Şekil 6.4'de
ana program ve A, B, C, D, E olarak adlandırılmış 5 altyordamı gösterilmekte
Program yürütmesi esnasında çeşitli noktalarda yığının durumu Şekil 6.5'de ve
miştir. Verilen bu şekilde, denetimin belirli bir altyordama aktarılması ile il
parametreler, içinde harfler bulunan bir kutu ile gösterilmiştir. Bu nedenle:

MP	denetimin, ana programdan bir altyordama aktarılmasında saklanan p rametreleri gösterir.
A	denetimin, A altyordamından (ana program ya da altyordam B gibi) di bir altyordama aktarılmasında saklanan parametreleri gösterir.

Açıktır ki, altyordamların herhangi bir derinlikte iç içe geçirilmesi, yığında her
düzey için dönüş değerlerini saklamaya yeterli alan olduğu sürece mümkün ol
S 6.1, 6.3 caktır.



Şekil 6.4



Şekil 6.5

Veri saklama alanı olarak yığın

Yığınlar, şimdiye kadar genellikle, altyordamlara giriş ve çıkışlardaki kullanımları açısından ele alındı. Bu, oldukça önem taşımasına karşın, kullanım yerlerinden yalnızca biridir. Yığınlar birçok uygulama için çok uygun birer geçici veri saklama alanı olarak işlev görürler. Son-giren-ilk-çıkar biçimindeki işleyişleri, veriler uygun biçimde saklanıp geri alınabildikleri sürece bir sorun oluşturmaz. Üstelik, verilere bu yolla erişim, bazı uygulamalar için özellikle uygundur.

Bunun bir örneği, aritmetik ifadelerin, yüksek düzeyli dil derleyicisiyle işlenmesinde ortaya çıkar.

6.2 PUSH ve POP işlemleri

İtme işlemi, Bölüm 5.5'de ana hatlarıyla verildiği gibi, yığının üzerine yeni bir değer yerleştirmek için kullanılır. Aslında, bu değer yığında zaten bulunan diğer sayıların üzerine yerleştirilir ve tüm yığın, yeni bilginin, yığının en üst (ilk) konumuna yerleştirilebilmesi için, birer konum aşağıya itilir.

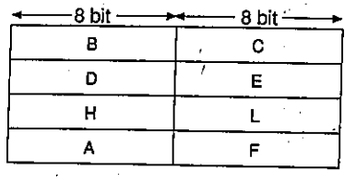
Çıkartma veya çekme işlemi, bir veri değerini, yığının en üst konumundan dışarı almak için kullanılır. Dışarı çıkartılan bir değerin altındaki tüm değerler, yani yığının ikinci, üçüncü, vb. konumundaki değerler otomatik olarak bir konum yukarı ilerlerler.

Tipik komutlar

Yığın içeriklerini işlemek için kullanılan birkaç komut örneği aşağıda verilmiştir. Bunlar, Ek-1'de sıralanan Intel 8085 mikroişlemci komut kodundan alınmıştır.

Daha önceki bölümlerden de hatırlanabileceği gibi, 8085, bir bayrak (F) kaydedicisi ile birlikte A (akümülatör), B, C, D, E, H, L işlemci kaydedicilerini içerir. Bazen, *durum* kaydedicisi olarak adlandırılan -ve örneğin, bir önceki aritmetik işlemin,

pozitif, negatif ya da sıfır olarak hangi sonucu verdiğ
teren- bayrak kaydedicisi, işlemci durumunu kaydet
kullanılır. Bu kaydediciler, Şekil 6.6'da gösterildiği
zenlenirler.



Şekil 6.6

Her bir kaydedici 8 bit içerir, ancak şekilde görüldü
kaydediciler çift çift gruplandırılabilir. Böylece, B C il
ile; H L ile; ve akümülatör, bayrak kaydedicisi ile grupl
lir. Son çift, PSW (işlemci durum sözcüğü) olarak adla

Makinenin komut takımında bulunan itme ve çekme işl
şunları içerir:

```
PUSH   B
PUSH   D
PUSH   H
PUSH   PSW
```

Bu komutlarda B, B ve C kaydedici çiftine; D, D ve E çift
H, H ve L çiftine; PSW ise A ve F çiftine karşılık gelir. Bu
komutlar benzer biçimde çalışır.

Bu nedenle, PUSH B komutu yürütüldüğünde B ve
kaydedicileri, iki bellek konumu işgal edecek şekilde yığ
atılır. PUSH PSW komutu yürütüldüğünde ise, A ve F'
içerikleri yığına atılır.

Mevcut POP işlemleri şunlardır :

```
POP    B
POP    D
POP    H
POP    PSW
```

ve bunların her biri, aşağıda belirtildiği gibi, karşılık gelen
PUSH komutunun tersini gerçekleştirir. Bu nedenle, POP
PSW komutu, iki yığın konumunun içeriklerini alır ve bunları
A ve F'ye yerleştirir. Daha açık olarak söylemek gerekirse, en
üstteki değer alınır ve F'ye yerleştirilir, daha sonra A'ya yer-
leştirilerek elde edilen veri ile ikinci bir POP işlemi yürütülür.

PUSH B - B'yi ve ardından C'yi iter
PUSH D - D'yi ve ardından E'yi iter
PUSH H - H'yi ve ardından L'yi iter
PUSH PSW - A'yı ve ardından F'yi iter.

POP işlemleri ise bunların tersidir:

POP B - C'yi ve ardından B'yi çıkartır.
POP D - E'yi ve ardından D'yi çıkartır.
POP H - L'yi ve ardından H'yi çıkartır.
POP PSW - F'yi ve ardından A'yı çıkartır.

Herhangi bir kaydedici çiftinde PUSH ve POP işlemleri, yukarıda olduğu gibi, diğerinin tam olarak tersi ise, programcının, hangi yığın konumunun A ve hangisinin F için (veya B ve C, ya da D ve E için) kullanılacağını bilmesine gerek yoktur.

Bu nedenle, bir hesaplama sırasında kullanılan kaydediciler, bir ya da daha fazla PUSH komutu yürüterek bir altyordama giriş yapıldığında, yığına saklanabilir ve karşılık gelen POP komutlarını yürütmek suretiyle de geri alınabilirler.

H ve L kaydedicilerinin içeriklerini, yığının en üstteki iki konumunun içerikleriyle değiştiren bir

```
XTHL
```

komutu da mevcuttur. Bu nedenle, yığındaki en üstteki iki konumun içerikleri çekilir ve aynı üst konumdaki iki yeri işgal edecek şekilde, H ve L'in bir önceki içerikleri yığına atılırken, bu içerikler H ve L'ye yerleştirilir.

Bu komutların herhangi biri, daha önce sözü edildiği gibi, yığının geçici bir veri belleği olarak kullanılmasına imkan vermek üzere bir program kodunun herhangi bir yerine yerleştirilebilir.

6.3 Mikroişlemcilerde yığın oluşturulması

5. Bölümde de özetlendiği gibi, mikroişlemci sistemlerinde yığınların gerçekleştirilebileceği iki yol vardır. Bunların ilki, sistemin imalatçısı için, işlemci yongasında bir LIFO belleği gibi davranan bir bellek alanı oluşturmaktır. İkincisi ise, işlemcideki *yığın işaretçisi* ile birlikte kullanılacak RAM'dir.

İşlemci yongasındaki yığınlar

Bazı mikrobilgisayar sistemleri, mikroişlemci yongası içinde, bir yığın gibi nacak şekilde düzenlenmiş kaydediciler içerirler. Bunlar, bir altyordam yapıldığında, program sayıcısının içeriklerini saklamak için kullanılır.

Genellikle, bu tür yığınların büyüklükleri sınırlıdır. Örneğin:

1 Rockwell PPS4 mikroişlemci yongası iki-düzeyli bir yığına sahiptir.
2 Texas TMS 1000 mikroişlemci yongası tek düzeyli bir yığına sahiptir.
3 General Instruments PIC 1645/1650 mikroişlemcisi iki-düzeyli bir yığ hiptir.
4 Intel 4040 mikroişlemci yongası yedi-düzeyli bir yığın içerir.
5 Signetics 2560 mikroişlemci yongası onbeş-düzeyli bir yığın içerir.

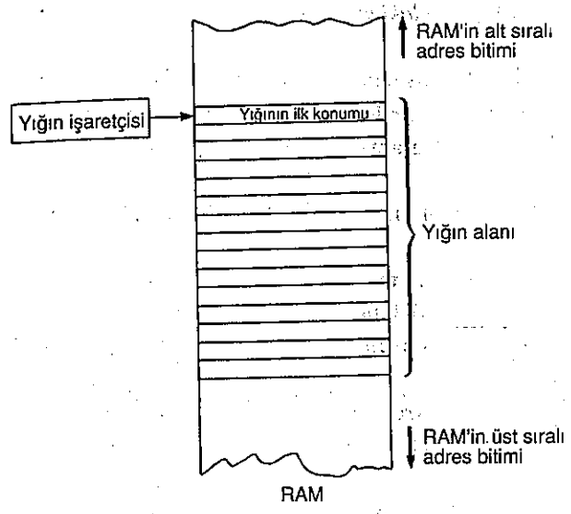
Bu birimlerdeki yığınların kapasiteleri sınırlı olduğundan, altyordamlara ((kesme hizmet altyordamlarına) giriş yapıldığında bunlar, normal olarak yal program sayıcısının içeriğini saklamak için kullanılır. Diğer herhangi bir bil saklanmasında kullanılmaz.

Bu nedenle, örneğin, Intel 4040 mikroişlemcisi, yedi-düzeyli altyordamlamaya verir. Texas TMS 1000 mikroişlemcisi, yalnızca, içine program sayıcısı içeri saklanabileceği bir kaydediciye sahiptir (gerçekte, buna yığın bile denemez dolayısıyla, sadece tek düzeyli bir altyordam çağrısına izin verir.

Sınırlandırılmış yığın, gösterildiği gibi, sadece program sayıcısının içeriğini, altyordam dönüş adreslerini tutmak için kullanıldığından, korunması ger herhangi bir işlemci kaydedicisi içeriği bir başka yere saklanmalıdır. Altyo dönüş adresleri, belleğe normal değişkenler olarak yazılabildiğinden, bu tam geçerlidir.

Yukarıda sıralanan mikroişlemciler, genellikle, mevcut birimler arasında en geliştirilmiş olanlarıdır. Çok karmaşık işlemleri gerektirmeyen, basit uygulama da kullanılmak içindirler ve bunlarda, muhtemel altyordamlama düzey sayıs daki sınırlama çok da önemli değildir. Bununla birlikte, daha gelişkin ve daha kullanılan üçüncü ve dördüncü kuşak sistemler için, elde edilebilen sınırlı y boyutu, eğer yığın, işlemci kaydedicilerinden imal ediliyorsa, ciddi bir sınır getirebilir. Bu sistemler için, RAM'in bir parçasını, yığın olarak kullanma alte tifi, cazip bir yöntemdir.

S 6.5



Şekil 6.7

Yığın olarak kullanılan rastgele erişimli bellek

Son kuşak mikroişlemcilerde, yığını oluşturmak için en sık kullanılan yöntem, yığın işaretçisi ile birlikte RAM kullanmaktır. Yığın işaretçisi, mikrobilgisayarın işlemcisinde daha özel bir şekilde kullanılan, basitçe bir kaydedicidir. Bu işaretçi, RAM'de bulunan yığının üstündeki konumun adresini tutar. Dolayısıyla, Şekil 6.7'de gösterildiği gibi, yığının üstünü *işaret eder.*

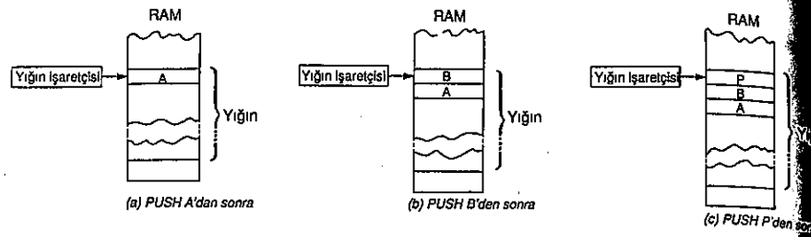
Yığın, bir RAM alanından ibarettir. Bu alanın boyutu, programcı tarafından istenildiği kadar küçük ya da büyük olabilir ve programcı, yığını bellek adresleme alanı içerisinde herhangi bir yere yerleştirebilir. Bu noktalar, kısaca açıklanacaktır.

Mikrobilgisayarda bir PUSH komutu yürütüldüğünde, şu işlemler meydana gelir:

1 Yığın işaretçisinin içeriği bir azaltılır.
2 Yığına yerleştirilecek veri, yığın işaretçisinde tutulan adreste saklanır.

Bu işlem Şekil 6.8'de gösterilmektedir. İlk olarak, şeklin (a) bölümünde gösterildiği gibi A değeri, yığının en üstündedir. Yığın işaretçi kaydedicisi, A'nın bulunduğu konumun adresini içerir.

Yığına B değerini yerleştirmek üzere bir PUSH komutu yürütüldüğünde, işaretçi,



Şekil 6.8

belleğin daha sıfır (0) düzeyli adreslerine doğru bir adım ilerlemek suretiyle azaltılır ve B, tutmakta olduğu adrese, yani A'nın hemen üzerindeki konuma saklanır.

P değerini yığına yerleştirecek bir PUSH komutu, benzer biçimde B'nin üzerine P'yi yerleştirerek işlem yapar [şeklin (c) bölümü]. Bu nedenle, yığının, belleğin sıfır (0) düzeyli adreslerine doğru 'genişlediği' görülebilir.

Yığının bellekteki mutlak konumu, yığın işlemleri başladığında, işaretçi kaydedicisindeki değer ile yönlendirilir. Kaydedici, bu konumun ayarlanmasına olanak tanımak üzere programın denetimi altında yüklenebilir.

Mikrobilgisayarda bir POP komutu yürütüldüğünde :

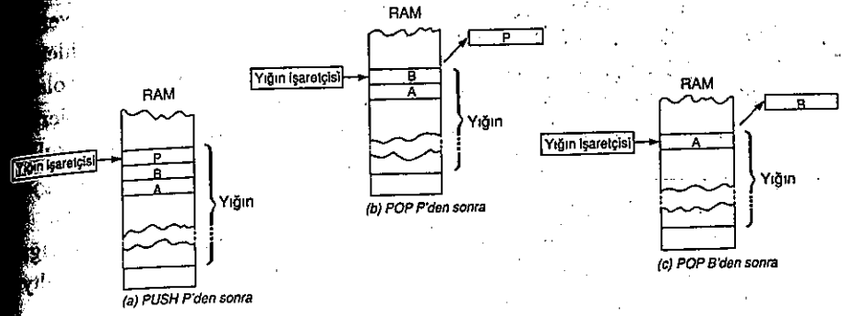
1 Yığın işaretçisinin içerdiği adresteki veri, bellekten okunur.
2 Yığın işaretçisinin içeriği bir artırılır.

Dikkat edilmesi gereken önemli bir nokta, işaretçi üzerinde işlem yapılması RAM'e erişimin, PUSH ve POP işlemlerinde ters sırada meydana geldiğidir.

Şekil 6.9'da, bir POP işlemleri dizisi gösterilmektedir. Yığının, Şekil 6.8 (c)
S 6.4 gösterilen konumdan itibaren başladığı varsayılmıştır.

Uygulamada yığın işaretçisinin işleyişi

Intel 8085 mikroişlemci sisteminde mevcut PUSH ve POP işlemleri Kısım 6.2'de tanımlanmıştı. Bunlar, uygulamada, RAM ile birlikte bir yığın işaretçi kaydedicisi kullanılarak yürütülür. Bu nedenle, program yürütmesi sırasında PUSH B komutu ile karşılaşıldığında, işlemci şu işlem adımlarını gerçekleştirir:



Şekil 6.9

1 Yığın işaretçisini 1 azaltır.
2 B kaydedicisinin içeriğini, yığın işaretçisinde tutulan adreste saklar.
3 Yığın işaretçisini 1 azaltır.
4 C kaydedicisinin içeriğini, yığın işaretçisinde tutulan adreste saklar.

Genelde, bu komutun yürütülmesi sırasında, yığın işaretçisi iki azaltılır.

Bir POP B işlemi ile karşılaşıldığında, işlemci :

1 Adresi yığın işaretçisinde bulunan bellek konumundaki veriyi okur. Bu veri, C kaydedicisine yerleştirilir.
2 Yığın işaretçisinin içeriğini 1 artırır.
3 Adresi yığın işaretçisinde bulunan bellek konumundaki veriyi okur. Bu veri B kaydedicisine yerleştirilir.
4 Yığın işaretçisinin içeriğini 1 artırır.

Görüldüğü gibi, sonuçta yığın işaretçisi 2 artırılır.

Böylece, yığının RAM'deki konumunu tanımlaması için, yığın işaretçisine bir değer atanır. Bu amaçla komut takımında

SPHL

komutu bulunur. Bu komut, H,L kaydedici çiftindeki 16-bitlik değeri, yığın işaretçisine aktarır. H ve L, makinenin normal genel-amaçlı işlemci kaydedicileri oldukları
S 6.6 ndan, bu kaydedicilere bellekten ya da diğer kaydedicilerden değer atanabilir.

Bu değerler, gerektiğinde, yığına yerleştirilebilir.

Yığının RAM'de olmasının avantaj ve dezavantajları

RAM'in bir parçasının, bir LIFO yığını olarak kullanılmasına imkan verm
bir yığın işaretçisi kaydedicisi kullanmanın temel avantajı, elde edilen
boyutunun, RAM'in toplam sınırları içinde kalmak koşuluyla, sınırsız ola
dir. Bu nedenle yığınlar, yalnızca altyordam çağrıları için dönüş adresleri
kesme yönetim programlarını tutmak için değil, başka amaçlar için de kull
lir.

Bir altyordama girilirken, işlemci kayedici içeriklerinin saklanması, yığ
türdeki uygulamalarından biridir. Diğer bir tanesi de, bir program ile bir alty
arasında parametre aktarımında kullanılmasıdır.

Yığının RAM'de bulunmasının getireceği dezavantaj ise, kullanımının, bilg
rın anayolları boyunca veri aktarımı gerektirmesidir. RAM'de bilgi sakl
işlemci yongası içerisindeki doğrudan saklama yönteminden daha yavaş
nedenle, bir altyordama girmek için geçen süre, yığın RAM'de olduğunda, m
melen daha uzun olacaktır.

6.4 Yığın kullanımına giriş

Parametre aktarma

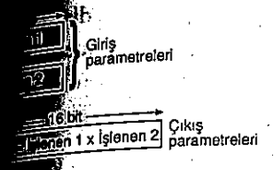
Bir ana program ile altyordam arasında parametre aktarmanın yapılabileceği
yöntemi ele aldık. Bu yöntemler:

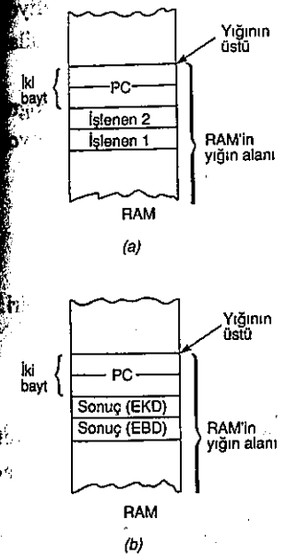
1 İşlemci kaydedicilerini kullanmak;

2 Parametreleri tutmak üzere bir bellek alanı kullanmak ve işlemci kaydedi
lerindeki bu alana bir işaretçi vermekti.

Diğer bir olanak, iki program arasındaki parametre aktarımını, yığın kullanar
gerçekleştirmektir.

Yığın genelde RAM içinde yer aldığından, bu yöntem, bir anlamda yukarıda
(2)'ye benzer. Fakat, RAM'in adreslenme yöntemi oldukça farklıdır. Bir durumda
belleğe, adres işaretçisi kullanmak suretiyle normal MOV komutları ile erişili
Diğerinde ise PUSH ve POP komutları kullanılır.





Şekil 6.11

Kısım 5.2 ve 5.3'de bir örnek olarak kullanılan çarpma yordamını ele alalım. Bu altyordamın iki giriş, bir de çıkış işleneni vardır. Giriş işlenenlerinin her birinin 8 bitlik olduğunu varsayalım. O zaman çıkış işleneni de 16 bit içerir. Çünkü, *n* bitlik iki sayının çarpımı, 2*n* bitlik bir sayıdır. Bu nedenle, çarpma altyordamı ile onu çağıran program arasında değiş-tokuş yapılacak parametreler, Şekil 6.10'da gösterildiği gibidir.

Altyordama girmek için, çağıran program şu işlemleri gerçekleştirir:

1. İşleneni iter (yığına atar)
2. İşleneni iter (yığına atar)

Çarpma altyordamını çağırır.

Bu, gerçek komutlarla, daha önce belirtilen kaydediciye PUSH (itme) komutları kullanılarak gerçekleştirilebilir.

Örneğin, çağıran programın, bir çarpma işlemi gerçekleştirmesine gerek duyması halinde, işlenenler uygun bir kaydedici çiftine, sözgelimi B ve C'ye yerleştirilebilir. 1. işlenen B'ye, 2. işlenen de C'ye yerleştirilebilir. O zaman,

```
PUSH B
```

komutu, B'yi (yani, 1. işleneni) ve ardından C'yi (yani, 2. işleneni) yığına saklar. Ardından programda, çarpma altyordamını çağırmak üzere bir CALL komutu bulunacaktır. Bu komut, program sayıcısının (PC) içeriklerini yığına yerleştirir. Bu nedenle, altyordama girildiğinde, yığının üstü Şekil 6.11 (a)'da görüldüğü gibi olur.

Altyordam, yığının 2. ve 3. konumlarında bulunan işlenenleri kullanmak isteyecektir. Bu değerleri almak için, ilk olarak dönüş adresini (yani program sayıcısı içeriğini) dışarı çıkartmalı (POP) ve ileride kullanmak üzere bir yere saklamalıdır. Buna uygun bir komut dizisi (bunu gerçekleştirmek için birçok yol vardır) aşağıda verildiği gibi olabilir:

```
Komut       Açıklama
POP   B   ;[PC]'yi, B ve C kaydedicilerine yerleştir.
POP   D   ;2. işleneni E'ye, 1. işleneni de D'ye yerleştir.
```

Altyordamda, bu işlemin ardından, E ile D'yi çarpacak bir komut dizisi bul
lıdır. Bu komut dizisinin bulunduğunu ve elde edildikten sonra sonucun, en
değerlikli yarısı (8 bit) H'ye ve en az ağırlıklı diğer yarısı (8 bit) L'ye olacak
H ve L kaydedicilerine yerleştirildiğini varsayalım. Daha sonra, çağıran pro
geri dönmek için, altyordamdaki en son komutlar :

```
Komut       Açıklama
PUSH  H   ;Sonucu yığına at
PUSH  B   ;Yığındaki [PC] 'yi değiştir.
RETURN    ;Çağıran programa geri dön.
```

PUSH ve PUSH B işlemlerinden sonra, yığının üstü, Şekil 6.11 (b)'deki gibi

RETURN komutu, program sayıcısının, yığının üstündeki değer ile yüklenm
neden olur. Ardından, çağıran program, iki baytlık sonucu çıkarır ve kullanı

Bu örnek, yığınların parametre aktarmada kullanımını göstermek için verilm
Normalde, bu durumda olduğu gibi, sadece birkaç parametrenin altyord
aktarılması gerektiğinde, işlemci kaydedicileri yığına başvurmaksızın kullanı

Bununla birlikte, eğer birçok parametre gerekirse, yığın, bunları bir progra
diğerine aktarmak için uygun bir yöntem sağlar. Bir sonraki bölümde, altyord
S 6.1, 7.1 giriş ve çıkışla ilgili daha başka örnekler verilmiştir.

Aritmetik ifadelerin hesaplaması

Bir yüksek düzey dil derleyicinin gerçekleştirmesi gereken görevlerden biri

Sonuç = A + B x C + (D + E) x F

gibi aritmetik ifadeleri çözmek için gerekli olan makine kodu komutlarını üret
tir. Makine kodlu program, bu ifadeleri, bu tür toplama işlemlerinde ge
kurallara uygun olarak sonuçlandırmalıdır. Sözgelimi yukarıdaki örnekte,
toplama yapmadan önce BxC işlemini yapmalı, yani ifadenin ilk bölümü için

(A + B) x C

işlemini değil,

(A + B) x C

işlemini yapmalıdır.

Bunu gerçekleştirmenin bir yolu, ifadeyi *son-ek* ya da *Polish* (J. Lukasiewicz tarafından sayısal bilgisayar mantıkları için geliştirilen bir gösterim, bkz. Terimler sözlüğü) biçimine getirmektir. Bu biçimde ifade, operatörler, yani +, -, x, ÷, aralarda yazılmak yerine değişkenlerden, yani A, B, C, D, E ve F'den sonra gelecek biçimde yeniden yazılarak değiştirilir. Böylece ifade şu hale gelir:

Sonuç : ABC x + DE + F x +

İlk bakışta bu ifade biraz garip gelebilir, ancak birtakım tanımlanmış kurallar kullanarak orijinal (*orta-ek*) ifadeden kolaylıkla çıkarılabilir.

İfade son-ek biçimine getirildiğinde, bir yığın kullanılarak hesaplanabilir.

Son-ek biçimindeki ifadenin RAM'de bir bayt dizisi olarak tutulduğunu ve ifadenin terimlerini birbiri ardı sıra okumak için bir program yazıldığını varsayalım. İzlenmesi gereken iki kural vardır :

1 Bellekten her ne zaman bir değişken okunursa, yığına atılır.
2 Bellekten her ne zaman bir matematik işlemci okunursa, en son iki değer yığından çıkartılır, gereken işlem yapılır ve sonuç tekrar yığına atılır.

Az önce kullanılan ifadeyi ele alalım :

ABC x + DE + F x +

Komut dizisi aşağıdaki gibidir :

a A değişkeni, RAM'den okunur ve yukarıdaki 1. kurala göre, yığına atılır.
b B değişkeni okunur ve yığına saklanır.
c C değişkeni okunur ve yığına saklanır. Bu noktada yığın, Şekil 6.12 (a)'da görüldüğü gibidir.
d x işlemi okunur. Yukarıdaki 2. kural gereğince:

C, yığından çıkartılır.

B, yığından çıkartılır.
B x C işlemi gerçekleştirilir.
B x C yığına saklanır.
Bu noktada, yığının üstü, Şekil 6.12 (b)'deki gibidir.

e + işlemi okunur. Bunun sonucu olarak:
B x C yığından çıkartılır.
A yığından çıkartılır.
(B x C) + A işlemi gerçekleştirilir.
(B x C) + A yığına atılır.
Bu noktada, yığın, Şekil 6.12 (c)'deki gibidir.

f D değişkeni okunur ve yığına saklanır.

g E değişkeni okunur ve yığına saklanır.
Bu noktada, yığın, 6.12 (d)'deki gibidir.

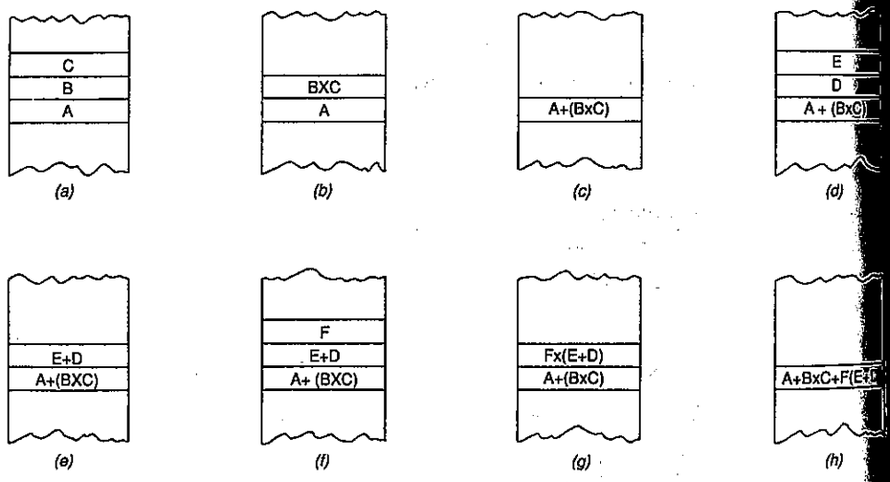
h + işlemcisi okunur. Bu, yukarıda (e) şıkkında tanımlanan biçimde işleminin yapılmasına ve sonucun yığına atılmasına neden olur. Bu noktada yığın, Şekil 6.12(e)'deki gibidir.

i F değişkeni okunur ve yığına atılır. Bkz: Şekil 6.12 (f).

(j) x işlemcisi okunur. Fx(D+E) işlemi gerçekleştirilir ve sonuç yığına atılır. Şekil 6.12 (g).

k Son olarak + işlemcisi okunur. Tüm işlem sonuçlandırılır ve Şekil 6.12 (h)'de görüldüğü gibi yığının en üstüne yerleştirilir.

Böylece, bu işlemleri gerçekleştirecek makine kodu dizisi; RAM'den veriyi al



Şekil 6.12

gerekli işlemleri yapmak ve yığın ve aritmetik komutlar üzerinde işlem yapmak üzere, PUSH ve POP komutları ile serpiştirilmiş, bir MOV komutları dizisinden ibarettir.

Yığının çalışma şekli bu türde problemler için çok uygundur.

Sorular

6.1 'Yığınlar ve uygulamaları' başlıklı kısa bir yazı yazın. Yazı aşağıdaki her bir şık için yığınların nasıl kullanıldığını gösteren bilgiler içermelidir.
(a) Kaydedici içeriklerini saklama ve geriye alma.
(b) Parametre aktarma.
(c) Altyordama giriş ve çıkış.

6.2 Yığının, son-giren ilk-çıkar düzenindeki işleyişinin, özellikle iç içe altyordama çağrılarında gerekli işlemler için nasıl uygun olduğunu gösterin. İç içe altyordamların derinliği, nelerle sınırlıdır?

6.3 Bir ana programda 9 ayrı altyordama çağrı yapılmaktadır. Programın çalışması esnasında, bunların her biri 6 kere kullanılır. Altyordamlar, A, B, C, D, ... I olarak adlandırılır. A, C'yi; B, F'yi; C, G'yi; D, A'yı; E hem D'yi hem de A'yı ve son olarak da G, H'yi çağırıyorsa, ana programı, altyordamları ve birbirleriyle bağlantılarını göstererek, program yapısını çizin. Bu yapıda ne kadar büyüklükte bir yığına ihtiyaç vardır? Her bir altyordam çağrısının (ikisi, işlemci kaydedicisini; biri program sayıcısını saklamak için) üç yığın konumu kullanacağını varsayın.

6.4 Mikrobilgisayarın normal RAM'indeki bir alanın, yığın olarak nasıl kullanılabileceğini tanımlayın ve PUSH ve POP komutları işlemci tarafından yürütüldüğünde, yığın işaretçisi kaydedicisinin içeriklerinin nasıl etkileneceğini açıklayın.

6.5 Bir mikrobilgisayar, işlemci yongası içinde 8-baytlık bir yığına sahiptir. İşlemci kaydedicilerinin her biri 8-bit, program sayıcısı ise 16-bit genişliğindedir. Her bir altyordam çağrısı yapıldığında, üç işlemci kaydedicisinin içeriklerinin saklanması gerekmektedir. Kısım 6.4'de özetlenen türde bir çarpım altyordamı ile argümanları değiş-tokuş etmek için yığın kullanılabilir mi? Cevabınızın gerekçesini açıklayın. Bu uygulamada, argümanların değiştirilebileceği en uygun alternatif yöntem ne olacaktır?

6.6 Yığın işaretçisinin bir PUSH komutu sırasında ilk olarak 1 azaltıldığı ve POP komutu sırasında son olarak 1 artırıldığı normal bir yığın işleminde yığın RAM'in alt-sıralı adres ucuna doğru genişler. Yığın işaretçisinin çalışma şeklini değiştirmek suretiyle, RAM'deki bir yığının, belleğin bir üst düzeyli adres ucuna doğru genişleyeceği bir yöntem bulmak mümkün müdür? Bir PUSH ve POP işlemleri dizisi gösterin ve diyagramlar çizmek suretiyle nedenlerinizi açıklayın.

6.7 B, C, D, E ve F değerleri, argüman olarak altyordama aktarılacak ve sonuç (A) değeri tekrar, altyordamdan çağıran programa gönderilecek olan:

$$A = (B+C) - (D-E) + F$$

ifadesini sonuçlandıracak kısa bir altyordam yazın.
Bu argümanların:
(a) işlemci kaydedicileri ve
(b) yığın
kullanmak suretiyle nasıl değiş-tokuş edilebileceğini gösterin. Son bölümde anlatılan PUSH ve POP işlemlerini ve Ek-1'de verilen 8085 komut takımını kullanın.

6.8 Bu bölümde anlatıldığı gibi, bir yığın kullanarak, aşağıdaki ifadeleri sonuçlandırmak için bir akış diyagramı çizin.
1 (A + B) . RAM'de AB + olarak saklı.
2 (A + B) x (C - D) . RAM'de AB + CD - X olarak saklı.
3 A(A + B) - C(B + D) . RAM'de AB+AxBD+Cx - olarak saklı.
4 A + (Bx ((C+D/E+F)/C) . RAM'de ABCDE/+F+C/x+ olarak saklı.
Çarpma ve bölme komutlarının mevcut olduğunu varsayın.

Bölüm 7 Altyordamlar

Bu bölümün amaçları : *Bu bölümü bitirdiğinizde:*

1 *Bir altyordamın temel mekanizmalarını değerlendirebilmeli,*
2 *Bir CALL komutunun:*
   (a) program sayıcısının içeriklerini uygun biçimde sakladığını ve ardından,
   (b) alt programlar başlangıç adresini program sayıcısına yüklediğini değerlendirebilmeli,
3 *Bir RETURN komutunun, program sayıcısını, altyordama en son girildiğinde saklı bulunan değerle yeniden yüklediğini değerlendirebilmeli,*
4 *Pratik bir mikroişlemcideki (Intel 8085) CALL ve RETURN komutlarının çalışmasının ayrıntılarını ve sistemdeki CALL komutunun üç baytlık, RETURN komutunun ise, bir baytlık bellek alanı işgal ettiğini anlayabilmeli,*
5 *Bir altyordama girişte program sayıcısındaki değerin saklanmasının,*
   (a) Mikroişlemci yongasındaki kaydedici ya da kaydediciler
   (b) Mikrobilgisayarın saklama alanlarındaki konumlarının kullanılmasıyla gerçekleştirilebileceğini değerlendirebilmeli,
6 *Yığın işaretçisinin işleyişini ve yığınların, altyordam çağrılarında nasıl kullanıldığını anlayabilmeli,*
7 *Bir altyordama giriş ve altyordamdan çıkışlarda makinenin durumunu saklamak ya da geriye almaktan sorumlu komutların:*
   (a) Ya çağıran programa, ya da
   (b) Altyordamın kendisine yerleştirilebileceğini değerlendirebilmeli,
8 *Elektriksel dalga biçimleri üretmek üzere kullanılan bir gecikme üretebilmeli,*
9 *Hesaplanacak bir pozitif sayı grubunun ortalamasını alabilmelisiniz.*

7.1 Giriş

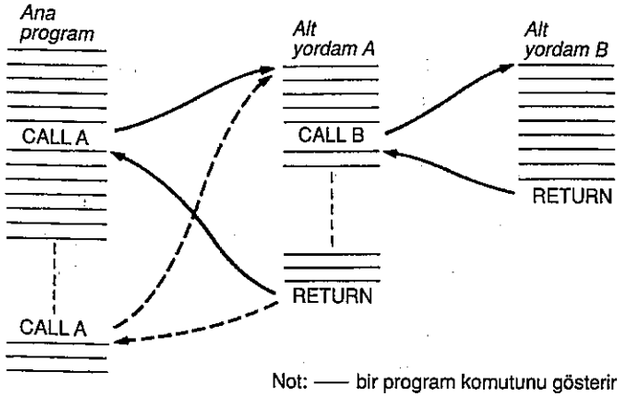
Bu bölüm, altyordamın özellikleri ve mikrobilgisayar sisteminde kullanımı ile ilgili incelememizi tamamlayacaktır. Bununla birlikte, bunun ayrıntılarına girmeden önce, şimdiye kadar ele alınan konuları kısaca hatırlamak ve gözden geçirmek yararlı olacaktır.

5 ve 6. Bölümlerde alt programlarla ilgili giriş bilgileri verilmişti. Örneğin, Kısım 5.2'de, bir alt programa giriş ve çıkış yapabilmek için özel 'call' ve 'return' komutlarının gerekli olduğu belirtilmişti.

Kısım 5.4'de bu komutların, bir CALL komutu yürütüldüğünde saklanacak ve RETURN komutu ile karşılaşıldığında tekrar yüklenecek, program kaydedicisi tutulan değere gerek duyduğu gösterilmiş, Kısım 5.5 ve 6.1'de ise, bu tür işlemle yığının oynadığı rol belirtilmişti.

Bir altyordama giriş ve ayrılışla ilgili aşamaları tekrar kısaca gözden geçirel Şekil 7.1'de, söz konusu durum gösterilmektedir. Ana program, yürütümü sıra da, A altyordamını, bir kez şekilde kesiksiz ok, bir kez de kesikli ok ile gösteril gibi, toplam iki kez çağırmaya gerek duyar. Alt programdan dönüş ise, yine kes ve kesiksiz oklarla gösterildiği gibi, ilgili CALL komutunun ardından gelen kom yapılır.

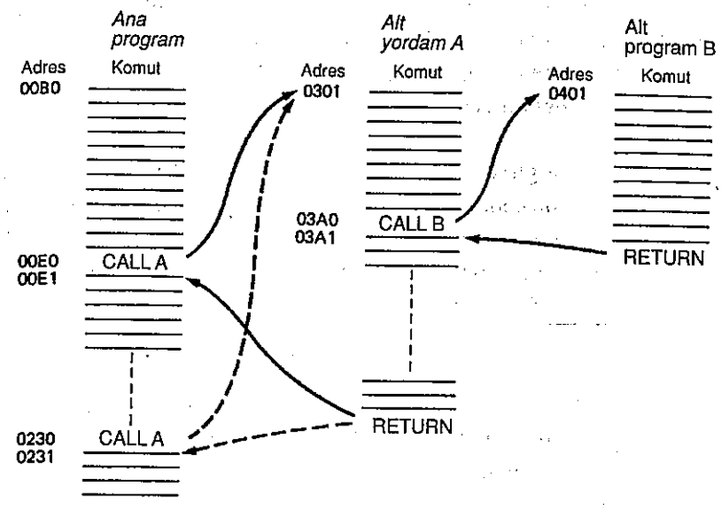
A altyordamı, A'nın içerisindeki çağır komutundan, B'yi oluşturan komut dizisi ilk sırasına uzanan kesiksiz ok ile gösterildiği gibi, B alt programını çağırır. Dön şekilde gösterildiği gibidir.



Şekil 7.1

7.2 CALL ve RETURN komutları

CALL ve RETURN komutlarının ayrıntılı çalışma biçimi, bir örnek aracılığı gösterilmiştir. Ana program ve altyordamların, Şekil 7.2'de gösterilen adres saklandığını varsayalım. Ana programdaki ilk komut 00B0 adresindedir, A tyordamı 0301, B altyordamı ise 0401 adresinden başlar. Bu adresler, yalnızca örneği göstermek için seçilmiş gelişigüzel adreslerdir. İki CALL A komutu sıyla 00E0 ve 0230; CALL B komutu ise A altyordamı kodu içerisindeki adresindedir.



Şekil 7.2

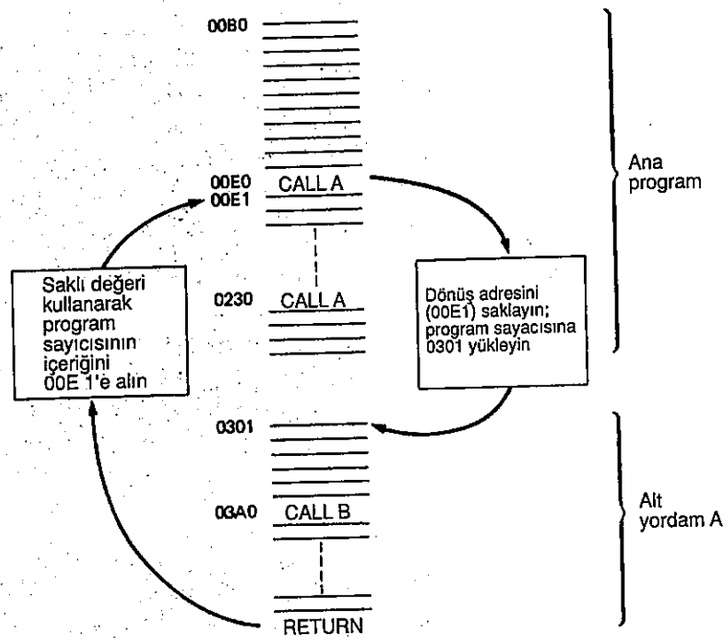
CALL A, CALL B ve RETURN komutları şöyle işler. Ana programın 00E0 adresindeki ilk CALL A komutu ile karşılaşıldığında, bilgisayar:

1 Program sayıcısının içeriklerini uygun biçimde saklar. Daha önce gösterildiği gibi, yığın bu amaçla kullanılabilir, fakat ileride anlatılacak olan daha başka alternatifler de vardır. CALL komutunu okuyup kodunu çözdükten sonra bilgisayarın, bir sonraki komutun adresini (bu örnekte 00E1) içerecek şekilde, program sayıcısının değerini otomatik olarak 1 artırdığına ve bunun, saklanan adres olduğuna dikkat edin.
2 Altyordamının başlangıç adresini (0301) program sayıcısına yükler. Bu adres, bir sonraki komut-getirme saykılında, yürütülecek komutu almak için kullanılır bu nedenle, ve programın işleyişi, alt programdaki ilk komuta atlar.

A Altyordamın sonundaki RETURN komutu yürütüldüğünde, saklı değer (00E1), komut yürütümü geriye, ana programa aktarılarak, program kaydedicisine yeniden yüklenebilir.

Bu işlemler, şematik olarak Şekil 7.3'de verildiği gibi gösterilebilir. Bu şekilde, altyordam gerçekte de olduğu gibi, bellekte ana programdan sonra gösterilmektedir. Karmaşıklığa yol açmamak için şekilde B altyordam gösterilmemiştir.

Ana programdaki ikinci CALL A komutu, ilgili dönüş adresinin (0231) saklanmasına neden olur ve program sayıcısını, daha önce olduğu gibi, 0301 değeri ile yükler.



Şekil 7.3

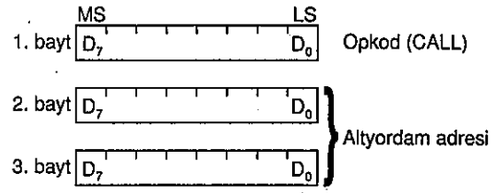
Daha sonra, A altyordamın bitiminde yürütülen RETURN komutu, progra sayıcısının içeriğinin yeniden 0231 ile yüklenmesini sağlar.

A altyordamındaki CALL B komutu, dönüş adresi (03A1)'nin saklanmasını sağ ve program sayıcısını, B altyordamındaki ilk komutun adresi, yani (0401) yükler. Ve son olarak, B altyordamının sonundaki RETURN komutu, progr sayıcısı içeriklerine yeniden 03A1 ile yükletir.

Altyordamlar, bu örnekte olduğu gibi, iç içe geçirilirse, dönüş adreslerinin sakl şekli önemli olacaktır. Bu daha sonra kısaca tekrar ele alınacaktır.

Intel 8085 CALL ve RETURN komutları

Yukarıdaki örneklerde, CALL ve RETURN komutları assembly dili biçimi yazılmıştır. Bu nedenle, komutlar anımsatıcılarla, altyordam ise sembolik ad gösterilmiştir.



Şekil 7.4

Intel 8085 mikrobilgisayarında, CALL komutunun makine kodlu komut biçimi, Şekil 7.4'de gösterildiği gibidir. Bu tür bir komuta ilişkin pratik bir örnek burada verilmiştir. Bu komut, 3 bayt işgal eden bir komuttur. İlk bayt, gerçekte CD (on altılı), yani ikili sistemde 11001101 (ikili) olan işlem kodunu; 2. ve 3. baytlar ise çağrılan altyordamın ilk komutunun adresini içerirler. Örneğin, A altyordamını (bkz: Şekil 7.2 ve 7.3) çağırmak için üç baytlık komut şöyledir :

	On altılı	*İkili*	
Bayt 1	CD	11001101	
Bayt 2	01	00000001	CALL A
Bayt 3	03	00000011	

Bu mikroişlemcinin imalatçıları tarafından belirlenen şu genel kurula dikkat edilmelidir: Altyordam adresinin en küçük değerlikli 8 biti, komutun ikinci baytını; en ağırlıklı 8 biti ise üçüncü baytını işgal eder.

Toplam adres, 16-bitlik bir yer işgal ettiğinden, 64Kbaytlık bir adresleme alanı içerisinde herhangi bir yerdeki bir altyordama çağrı (CALL) yapılabilir.

RETURN komutu birleşik bir adrese gerek duymaz (sadece, program sayıcısı içeriklerinin geriye alınmasını sağlar) ve bu nedenle bir baytlık bir komuttur. 8085'deki işlem kodu C9 (on altılı), yani 11001001 (ikili) dir.

Bir başka ayrıntının daha açıklığa kavuşturulması gerekir. Şu ana kadar verilen örneklerde, program sayıcısının içeriklerinin, bellekten bir komut getirilmesinin sonrasında bir sonraki komutu işaret edecek şekilde, 1 artırıldığı varsayılmıştı. Pratikte, durum her zaman tam olarak böyle değildir.

Program sayıcısı, bir sonraki komutun adresini içerecek şekilde artırılır - bu her zaman doğrudur. Bununla birlikte, artırılan miktar 1, 2 ya da 3 olabilir. Örneğin, CALL gibi üç baytlık bir komutla, belleğe saklandığında, durum aşağıda gösterildiği gibi olacaktır:

Adres	*Komut*
0000	CD
0001	01
0002	03
0003	Bir sonraki komut

Komutun ilk bölümü, yani işlem kodu bellekten getirildiğinde, (komut, yuk
gösterilen adreslerde saklanırsa) program sayıcısının içeriği 0000'dır. Daha so
içeriği 1 artırılır (0001'e) ve komutun ikinci baytı getirtilir. Ardından, prog
sayıcısı tekrar 1 artırılır (0002'ye) ve üçüncü bayt getirtilir. Son olarak, bir sonr
komut-getirme çevrimi için hazır olacak şekilde bir kez daha 1 artırılır.

Böylece, CALL komutunun yürütülmesi sırasında, program sayıcısı toplam ola
"üç" artırılmış olur. Bu durum, daha önceki açıklamaların hiçbirini değiştirmez. B
istisna dışında bir altyordama girişte saklanan dönüş adresi çağır (CALL) komut
nun ilk baytının adresi artı "1" değil, artı "3" dür. Bu adres de, yine CALL
komutunun hemen ardından gelen komutun adresidir.

8085 gibi pratik bir mikroişlemcide, basit CALL ve RETURN komutlarının birçok
varyasyonu vardır. Bu nedenle işlemci:

CNZ adres	Sıfır değilse çağır
CZ adres	Sıfırsa çağır
CNC adres	Elde yoksa çağır
CP0 adres	Tek eşlikse çağır
CP adres	Pozitifse çağır
CM adres	Negatifse çağır

gibi *koşullu* dönüş komutlarına sahiptir.

Bu aşamada, bunların her birinin tam olarak ne anlama geldiği önemli değildir.
Belirli koşullarla karşılaşıldığında, bir altyordamın çağırılabileceğini belirtmek
yeterlidir.

RNZ	Sıfır değilse geri dön
RZ	Sıfırsa geri dön
RNC	Elde yoksa geri dön
RPO	Tek eşlikse geri dön
RP	Pozitifse geri dön
RM	Negatifse geri dön

gibi, bir önceki CALL komutlarına karşılık gelen *koşullu dönüş* komutları vardır.

Bu komutlar, belirtilen koşullar ile karşılaşıldığı sürece, bir altyordamdan ana programa dönüşe izin verirler.

7.3 Program sayıcısının saklanması

Bir alt programdan dönüş adresinin saklanabileceği birçok yol vardır. Bunlar genel olarak:

1 Mikroişlemci yongasındaki kaydedici ya da kaydedicilerin içerisinde
2 Mikrobilgisayar belleği içinde

olmak üzere sınıflandırılabilir.

İşlemci yongasındaki kaydediciler

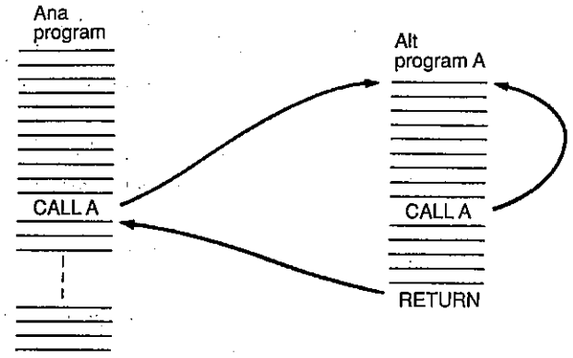
Bazı mikroişlemciler, işlemci yongasında, özellikle altyordam dönüş adreslerini saklamak üzere bir ya da birkaç kaydediciye sahiptirler. Bunlara örnekler, Kısım 6.3'de verilmiş ve sınırlamaları dile getirilmişti.

Bu sistemlerde, her altyordam çağrı komutu yürütümünde, program sayıcısının içeriği, kaydedici ya da kaydedicilere otomatik olarak yerleştirilir. Bu türde bir çağrı sırasında, sistemin ana belleğine erişim yapılmasına gerek yoktur.

Program sayıcısının mikrobilgisayar belleğinde saklanması

Bir altyordam dönüş adresini, mikrobilgisayar belleği içerisinde saklamanın iki yolu vardır. Birincisi, belleğin bir bölümü, 6. Bölümde belirtildiği gibi, bir yığın işaretçi kaydedicisi ile birlikte, son-giren ilk-çıkar düzeninde bir yığın olarak düzenlenebilir. Alternatif olarak dönüş adresi, altyordamın kendisinin içinde saklanabilir. İlk olarak bu yöntem ele alınacaktır.

Kullanılan bir teknik, altyordamdaki ilk ya da ilk iki bellek konumunu, dönüş adresini tutmak üzere ayırmaktır. Ardından altyordamın program kodu, bunun ardından, ikinci ya da üçüncü ve ardından gelen bellek konumlarında yer alır.



Şekil 7.5

Altyordama giriş yapılmasına neden olan komut, daha sonra, program sayıcısının içindeki değeri bu ayrılmış bellek konumuna (konumlarına) yazar ve programın yürütmesini, ilgili altyordamın ilk komutuna aktarır. Altyordamın bitiminde, komut, bu saklanmış değeri kullanarak, program sayıcısı içeriğini yeniden eski değerine alır.

Bu teknik, her bir altyordamlama düzeyinin dönüş adresini, o düzeyde saklayarak, iç içe altyordamlamaya izin verir. Bununla birlikte, *kendi kendini tekrarlayan (recursive)* altyordam kullanmaya izin vermez. (Kendi kendini tekrarlayan altyordamlar, kendi kendini çağıran altyordam-lardır).

Bir altyordam her çağırıldığında, dönüş adresini saklamak için kullanılan bellek konumları doldurulur. Dönüş komutu yürütüldüğünde, bu konumlar, içlerindeki bilgiye artık ihtiyaç olmadığından boşaltılır.

Bir altyordamın, -ilkin gerçekleşmesi pek mümkün olmayan bir teknikte ortaya çıkan- özyineli kullanımı, bazı tip problemlerin çözümünde çok faydalıdır. Bununla birlikte, karşılık gelen RETURN komutlarını yürütmeksizin (Şekil 7.5), pek çok altyordam çağrısı (CALL) yürütmesini içerir.

Bu tip bir işlemde, dönüş komutunun sayısı kadar altyordam çağrısı saklanmalıdır. Bu da, dönüş adreslerini tutmak için bir yığın kullanılarak gerçekleştirilebilir, fakat az önce anlatılan yöntemle mümkün değildir.

Yığın kullanımı

Yığın kullanımı, mikrobilgisayar sistemlerinde, en yaygın program sayıcısı sakla-

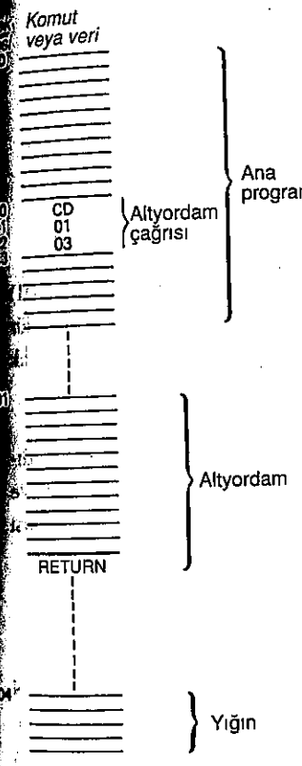
ma yöntemidir ve bu uygulamadaki işleyiş biçimi, Kısım 5.5 ve 6.1'de özetlenmiştir. 8085, bu şekilde yığın kullanan bir mikrobilgisayardır:
CALL komutunun işlevi şu şekilde ifade edilebilir.

1 Program sayıcısının içeriklerini yığına at.
2 CALL komutunda belirtilen adresi, program sayıcısına yükle

RETURN komutunun işlevi ise şöyledir :

1 Yığının tepesindeki değeri dışarı çıkar.
2 Bu değeri program sayıcısına yükle.

Şimdi, yığın işaretçisi kullanımı, RAM'in yığın olarak kullanımı ve altyordama giriş ve çıkış tanımlarını birleştirmek mümkündür. Yine bu da, en iyi biçimde bir örnekle gösterilebilir.



Şekil 7.6

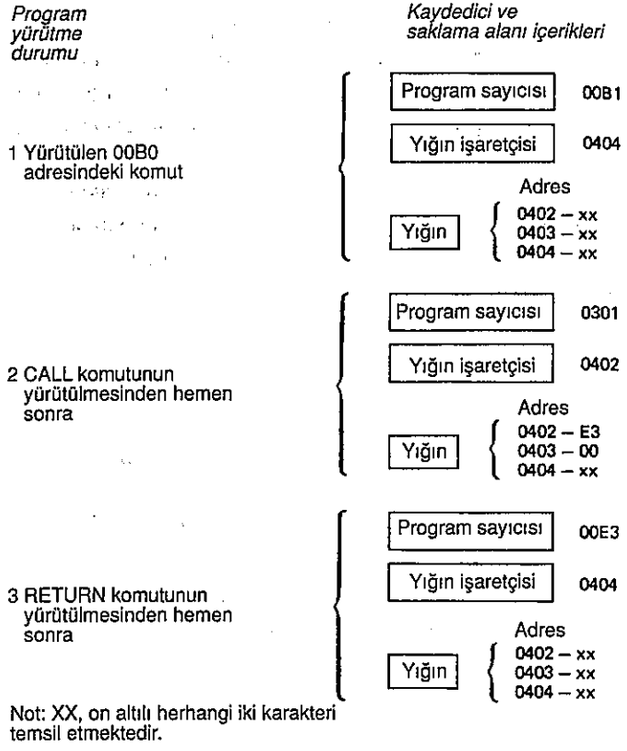
Şekil 7.6'da gösterildiği gibi, 00B0 adresinden itibaren saklanan bir ana programın, 0301 adresinden başlayan bir altyordamı çağırdığını varsayalım. CALL komutu, 00E0, 00E1 ve 00E2 (on altılı) adreslerinde üç baytlık yer işgal eder.

Bundan başka, yığının RAM'de olduğunu ve başlangıç adresinin, yani yığın işaretçisinde tutulan adresin, 0404 (on altılı) olduğunu varsayalım.

Şekil 7.7'de, programın yürütülmesi sırasında, yığının, yığın işaretçisinin ve program sayıcısının içerikleri gibi çeşitli noktalar gösterilmiştir. Bütün değerler, on altılı biçimde verilmiştir.

Başlangıçta, 00B0 adresindeki komut yürütüldüğünde, program sayıcısında 00B1 değeri (bir sonraki komutun adresi) vardır, yığın işaretçisi 0404'ü tutar, yığındaki değerler ise ilgisiz değerlerdir. Her bir yığın konumu tek bir baytlıktır ve bu nedenle, iki tane onaltılı 'dikkate alınmaz' değeri gösteren XX ile temsil edilir.

00E0'dan 00E2'ye kadar olan adreslerdeki CALL komutunun yürütülmesinin hemen ardından, durum şeklin 2. adımında



Şekil 7.7

gösterildiği gibidir. Program sayıcısı, altyordamdaki iki komutun adresiyle (0301) yüklenmiştir ve dönüş adresi yığının tepesine yerleştirilmiştir. Bu adres 00E3'dü ve yığın işaretçisi 0402 değerine, yani yığının o anki başlangıç noktasına ulaşacak şekilde, iki kez azaltılarak yığının iki baytına saklanır.

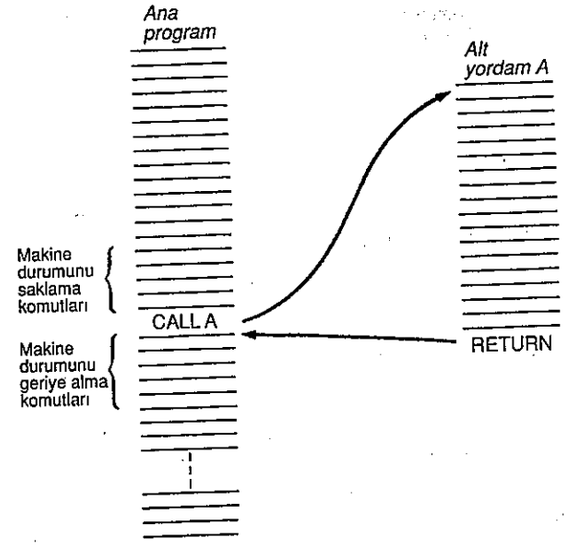
Altyordamda RETURN komutunun yürütülmesinden sonra, Şekil 7.7'nin 3. adımına varılır. Program sayıcısı değeri tekrar yığından alınarak 00E3'e getirilir ve
S 7.1, 7.2 yığın işaretçisi iki kez artırılarak 0404'e getirilir. Böylece, yığının başlangıç noktası tekrar 0404 adres olmuş olur.

7.4 Makine durumunun saklanması ve geriye alınması

Daha önce (Kısım 5.4'te), bir alt yordama girişin, çağıran programın RETURN komutunun ardından yürütmenin devam edebilmesi için, mikroişlemcinin durumunun genellikle bir yığında saklanmasını gerektirebileceğini görmüştük. Kısım 6.1'de, bunun gerçekleştirilebileceği bir yol anlatılmıştı. Makine durumunu sakla-

mak ve tekrar eski konumuna getirmekle yükümlü komutlar, çağıran programda yerleştirilmiştir. Bu nedenle, bu programdaki komutlar, işlemci kaydedicilerinin içeriklerini, bir altyordama giriş yapmadan önce yığına atar; diğer komutlar ise, altyordamdan çıkıştan sonra, bu kaydedicileri tekrar eski konumlarına getirir.

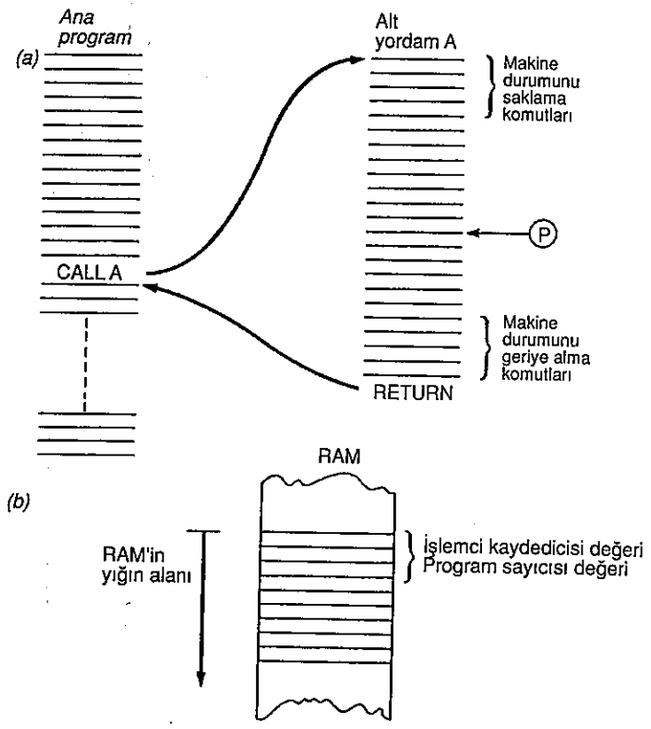
Bu, şematik olarak Şekil 7.8'de gösterilmiştir. Saklama komutları, bir PUSH işlemleri dizisi; geriye alma ise, bir PULL işlemleri dizisidir. Bu sistemde program sayıcısı değeri, yığına yerleştirilecek en son ve yığından dışarı çıkartılacak ilk veri parçasıdır (Bkz: Kısım 6.1).



Şekil 7.8

Alternatif bir olasılık ise, saklama ve geriye alma komutlarını, altyordamın kendi içine yerleştirmektir. Bu, şematik olarak Şekil 7.9 (a)'da gösterilmiştir. Bu şekilde, CALL A komutu, program sayıcısı değerinin yığına atılmasını ve program yürütümünün, A altyordamının başlangıç noktasına aktarılmasını sağlar. Altyordamın başındaki komutlar (PUSH işlemleri) ise doğrudan işlemci kaydedici içeriklerini saklayacaktır.

Altyordamın sonunda, saklanan değerler yığından çıkartılır ve tekrar kaydedici-lere yerleştirilir. Ardından, RETURN komutu yürütülür.



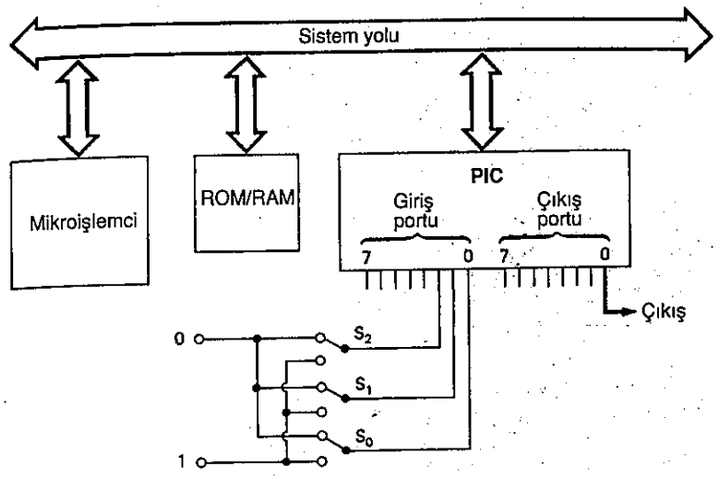
Şekil 7.9

Bu sistemde, program sayıcısının içeriği, Şekil 7.9 (b)'de gösterildiği gibi, yığındaki kaydedici içeriklerinin altında tutulur. Diyagram, altyordamın bünyesinde herhangi bir noktada, sözgelimi P'de, yığının durumunu göstermektedir.

Saklama ve geriye alma komutlarını, bu şekilde altyordamın içine yerleştirmenin avantajı, altyordamın kendisi çok uzun olmasına rağmen, toplam program uzunluğu azaltılıyor olmasıdır. Bu düzenlemeyle, tüm programda, sadece bir saklama ve (altyordamdaki) geriye alma komut takımı vardır.

Bununla birlikte, eğer bu komutlar ana programa yerleştirilirse, her altyordam çağrılışında bu komutların o noktada bulunuyor olması gerekecek, bu nedenle de program içinde birçok yerde kopyası bulunabilecektir.

S 7.3



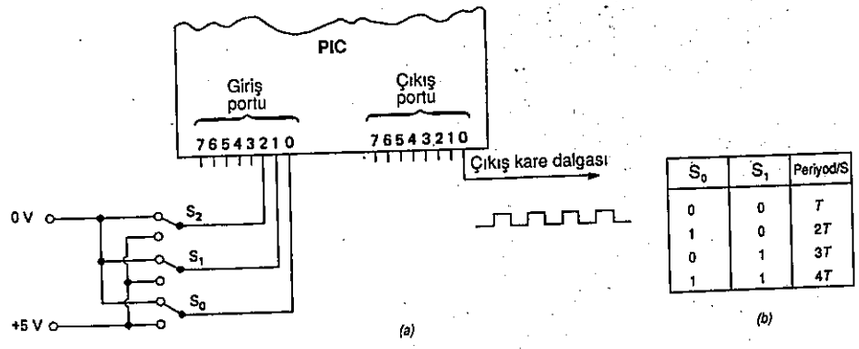
Şekil 7.10

7.5 Altyordam Örnekleri

Bir zamanlama örneği

Bir mikrobilgisayarın, bir karedalga sinyali üretmek için kullanılacağını varsayalım. Buna ilişkin bir sistem Şekil 7.10'da gösterilmektedir.

Mikrobilgisayar; birbirlerine normal sistem anayolu ile bağlanan, bir işlemci, bir ROM, bir RAM ve bir çevresel arabirim devresi içerir.

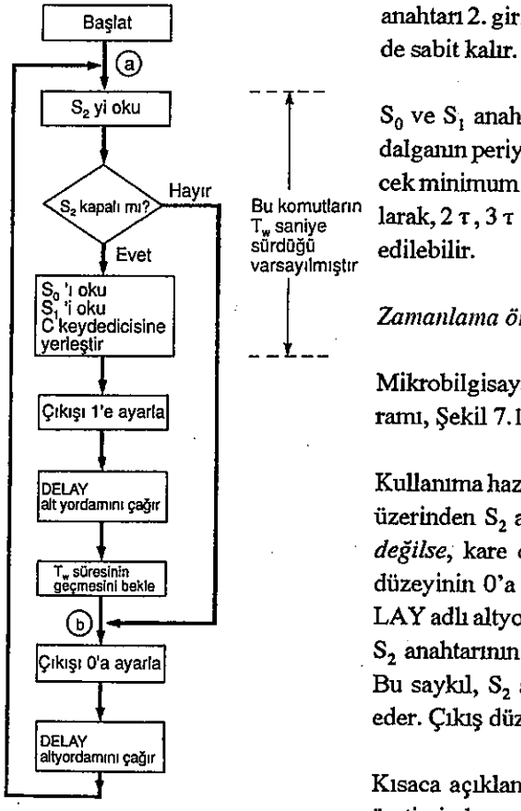


Şekil 7.11

Bu örnekte, biri giriş diğeri çıkış olarak düzenlenmiş, iki[t] çevresel arabirim devre portu kullanılır. Daha önce 2. Böl[üm] de anlatıldığı gibi, her iki port da paralel bilgi aktarabil[ir] yeteneğine sahip olduğu halde, bu uygulama için, çıkış po[rtu] nun yalnızca bir biti ve giriş portunun üç bitine ihtiyaç var[dır]

Çıkış karedalga sinyalini taşımak için ise, Şekil 7.11 (a)'[da] gösterildiği gibi, tek çıkış hattı kullanılır. Üç giriş hattı, S_0[,] ve S_2 anahtarlarına bağlanır. Bu anahtarların her biri, denet[le]dikleri giriş portu bitini ya mantıksal 1 (+5 V düzeyi) ya[da] mantıksal 0'a (0 V düzeyi) bağlanabilir.

S_2 anahtarı, karedalga sinyalinin üretilip üretilmeyeceğini de[ğiş]netlemek için kullanılacaktır. S_2 anahtarı 2. giriş bitine 1 vermesi durumunda, çıkışta kare dalga ortaya çıkmalıdır; S_2 anahtarı 2. giriş bitine 0 verdiğinde, çıkış mantıksal 0 düzeyin[de] de sabit kalır.



Şekil 7.12

S_0 ve S_1 anahtarları, Şekil 7.11 (b)'de gösterildiği gibi, kare dalganın periyodunu denetlemek için kulanılır. Elde edilebilecek minimum periyot τ saniyedir. S_0 ve S_1 anahtarları kullanılarak, 2τ, 3τ veya 4τ saniye süren daha uzun periyotlar elde edilebilir.

Zamanlama örneği ana program akış diyagramı

Mikrobilgisayar programını denetleyen programın akış diyagramı, Şekil 7.12'de gösterilmektedir.

Kullanıma hazırlanmasından sonra program, giriş portu 2. biti üzerinden S_2 anahtarının durumunu okur. Bu anahtar kapalı *değilse*, kare dalga üretilmez. Bu durumda, program, çıkış düzeyinin 0'a ayarlandığı (b) noktasına dallanır. Sonra DELAY adlı altyordama girer ve bu altyordamdan dönüşte, tekrar S_2 anahtarının durumunu okuduğu (a) noktasına geri döner. Bu saykıl, S_2 anahtarının konumu değiştirilene kadar tekrar eder. Çıkış düzeyi bu nedenle 0'da kalır.

Kısaca açıklanan DELAY altyordamı, kare dalga sinyalinin üretiminde gerekli τ, 2τ, vb... zaman aralıklarını üretmekle

sorumlu program parçasıdır.

S_2 anahtarı, (a) noktasının ardından okunup test edildiğinde, kapalı durumda görülürse, program (b)'ye dallanmaz, test kutucuğundan 'Evet' kolu yönünde aşağıya doğru devam eder (Şekil 7.12).

Bu program parçası, kare dalganın periyodunu öğrenmek için, S_0 ve S_1 anahtarlarını okur, çıkış hattını mantıksal 1'e çeker ve ardından DELAY altyordamını çağırır. Bu altyordamdan dönüşte, bir miktar 'T_w süresi harcar' ve sonra daha önce anlatılan program parçasına giriş yapar. Programın bu parçasında, daha önce olduğu gibi, çıkışı 0'a çeker, DELAY'i çağırır ve sonra (a)'ya döner.

Bu nedenle, S_2 anahtarının kapalı olmasının sonuçtaki etkisi, çıkış hattının sırayla 1 ile 0 arasında değişmesidir. Kare dalgalar bu yolla üretilmiş olur.

Bu kare dalganın mantıksal 0 düzeyinde olduğu aralık :

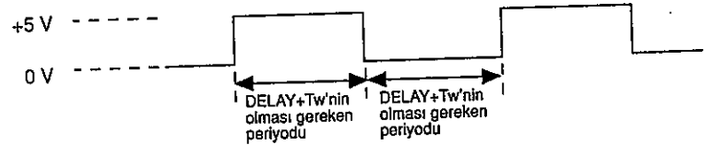
1 DELAY altyordamının harcadığı zaman aralığı, artı
2 'S_2'yi oku' komutunun aldığı süre, *artı*
3 'S_2 kapalı mı?' komutunun aldığı süre, *artı*
4 'S_0'ı oku', 'S_1'i oku' ve 'C kaydedicisine yerleştir'

komutlarının aldığı süreye (bu sürelerin toplamına) eşittir.

(2), (3) ve (4) nolu maddelerde belirtilen sürelerin, Şekil 7.12'de gösterildiği gibi, toplam T_w saniye tuttuğu varsayılır.

Gerçek bir kare dalga üretileceği zaman, bu periyoda, çıkışın mantıksal 1 düzeyinde tutulduğu zaman aralığının da eklenmesi gerekir. Bu nedenle, akış diyagramına (b) noktasından hemen önce, 'Boşta bekleme süresi Tw' kutucuğu yerleştirilir.

Böylece, çıkışta üretilen kare dalga, Şekil 7.13'de gösterildiği gibi olur.



Şekil 7.13

DELAY altyordamı, DELAY altyordamı τ , 2τ, 3τ ya da 4τ saniyelik bir z aralığı üreten kısa bir program parçasıdır.

İstenen zaman aralığı, daha öncede açıklandığı gibi, ana programdaki S_0 anahtarlarından okunan değer tarafından denetlenir. Bu nedenle, altyordam ç dığında, bu değerlerin parametre olarak altyordama aktarılması gerekir. gerçekleştirmek için seçilmiş olan yöntem, değerleri işlemcinin C kaydedic yerleştirmektir (Bkz: Şekil 7.12).

Altyordamın akış diyagramı Şekil 7.14'de verilmektedir. D ve E kaydedic altyordamın içinde kullanıldıklarından, altyordama giriş yapıldığı kaydedicilerin içerikleri yığına atılır ve altyordamdan çıkış yapıldığında te geriye alınırlar.

D ve E kaydedicilerinin içerikleri yığına atıldıktan sonra, altyordama giriş kaydedicisindeki değer D'ye aktarılır. Bu, altyordamın bir sonraki çağrılışında değerin tekrar kullanılabilmesi için, altyordam çalıştığında C'deki değerin sa olarak kalmasına olanak tanır. D'deki değer, altyordam içinde kullanılan t değerdir.

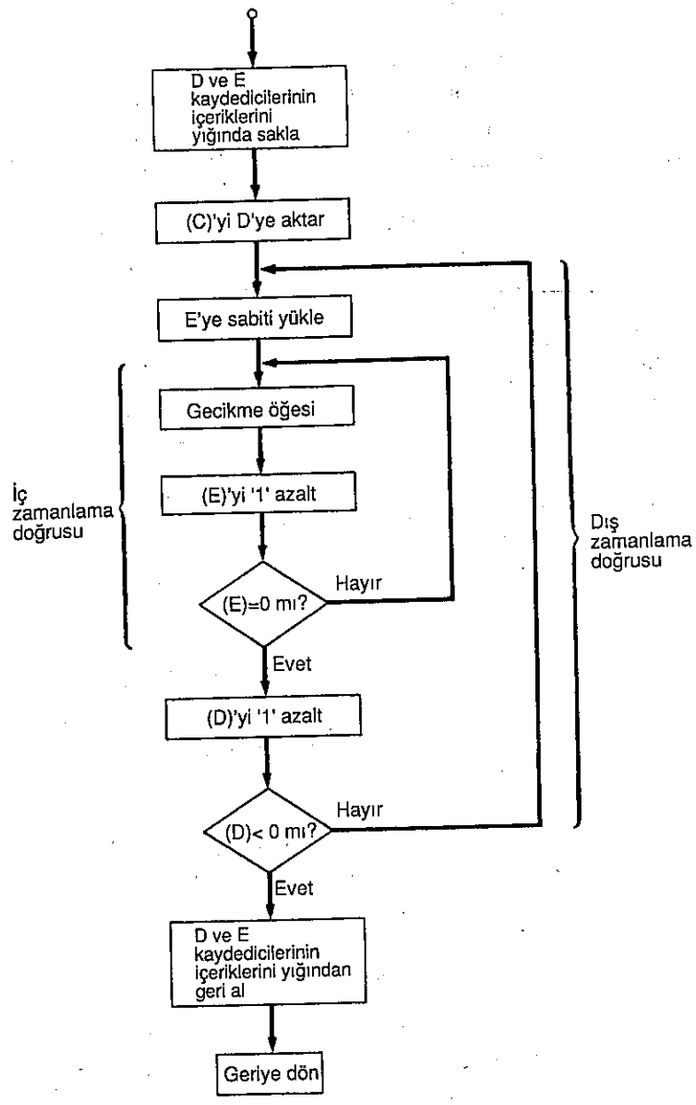
İstenen zaman gecikmesi, altyordam içinde birbiriyle iç içe geçmiş iki zamanla döngüsü ile üretilir. İç döngü (Şekil 7.14) bir gecikme öğesi, yani boşta bekleme neden olan komutlar içerir. Programın yürütülmesi esnasında bu döngünün kaç döneceği, döngü yürütülmeden önce E kaydedicisine yerleştirilen değer tarafında denetlenir.

İç döngü, dış döngünün içerisindedir ve dış döngünün saykılı, D kaydedicisinde değere, yani S_0 ve S_1'den okunan sayıya göre gerçekleştirilir.

D'deki sayı '0' ise, iç döngü bir kez saykıllanır; '1' ise iki kez; '2' ise üç kez ve ise dört kez çevrimlenir. Bunun nedeni, en dıştaki döngünün düzenleniş biçimidir.

Bu nedenle, iç döngü tarafından üretilen gecikme saniye ise, altyordam tarafında üretilen toplam gecikme, τ , 2τ, 3τ veya 4τ saniye olabilir. Bu rakamlar tam doğ değildir, çünkü altyordamda döngülerin dışındaki ek komutlar da küçük bir mikt gecikme yaratır ve periyodu biraz uzatırlar.

Dış döngüden çıkışta, ana programa RETURN (dönüş) komutu yürütülmeden önc altyordam kodu D ve E kaydedicilerinin saklanmış değerlerinin yığından çıkarılmasını ve bu kaydedicilere yerleştirilmesini sağlar.



Şekil 7.14

Bu nedenle, zaman-gecikmesi altyordamı kodu, Tablo 7.1'de gösterildiği gibidir. Önceki örneklerde olduğu gibi, altyordam kodunun bellekte 0301 adresinden başladığı ve yığının başlangıcının da 0404 adresinde olduğu varsayılmıştır.

Tablo 7.1

Adres	Komut: Ondalık	Komut: On altılı	Anlamı
0301	11010101	D5	(D) ve (E)'yi yığına at
0302	01010001	51	(C)'yi (D)'ye taşı
0303	00011110	1E	0F değerini E'ye dolaysız taşı
0304	00001111	0F	(ondalık değer 15)
0305	00000000	00	'İşlem yapma' komutları
0306	00000000	00	(gecikme ögesi)
0307	00011101	1D	(E)'yi 1 azalt
0308	11000010	C2	
0309	00000101	05	Sonuç ≠ 0 ise
030A	00000011	03	0305 adresine atla
030B	00010101	15	(D)'yi 1 azalt
030C	11110010	F2	
030D	00000011	03	Sonuç ≥ 0 ise,
030E	00000011	03	0303 adresine atla
030F	11010001	D1	(D) ve (E)'yi geriye al
0310	11001001	C9	RETURN (Geriye dön)

Tüm adres ve sabitler on altılı sayılar olarak ifade edilmiştir. Bu altyordamda gecikme ögesi, iki İŞLEM YAPMA (No operation) komutundan ibarettir. Bunlar sadece gecikme süresi üretilmesini sağlarlar. E'ye yüklenen sabitin 0F (on altılı) yani 15 (ondalık) olduğu kabul edilmiştir.

Ana program kodu Ana programın program kodu parçası, Tablo 7.2'de gösterildiği gibidir. Sadece, yığın kullanımı ve gecikme altyordamının çağrılmasına ilişkin bölümler eklenmiştir. Anahtarların durumunun okunması, çıkış düzeyinin değiştirilmesi gibi eklenmemiş olan bölümler de kolayca anlaşılabilir. Bu gibi programlar örnekleri ilerki bölümlerde verilecektir.

Ana program kodu, daha önce olduğu gibi 00B0 adresinden başlar. Programdaki ilk komut, yığın işaretçisindeki değeri, 0404'e (on altılı) getirir ve dolayısıyla bellekteki yığının başlangıç noktasının konumunu düzenler.

Daha sonra gelen PUSH, POP, CALL ve RETURN komutları, bu taban konumun-
S 7.8 dan başlayarak işaretçideki değer üzerinde işlem yapar.

Tablo 7.2

Adres	Komut: Ondalık	Komut: On altılı	Anlamı
00B0	00110001	31	Yığın işaretçisini kullanıma hazırla
00B1	00000100	04	(0404 - yığının en üst konumunun
00B2	00000100	04	adresi - ile yükle)
00B3			
.	.	.	.
.	.	.	.
.	.	.	.
00E0	11001101	CD	DELAY'i çağır (yığına 00E3'ü at
00E1	00000001	01	ve program sayıcısına 0301'i
00E2	00000011	03	yerleştir)
00E3			Bir sonraki komut
.	.	.	.
.	.	.	.
.	.	.	.

Bir aritmetik örnek

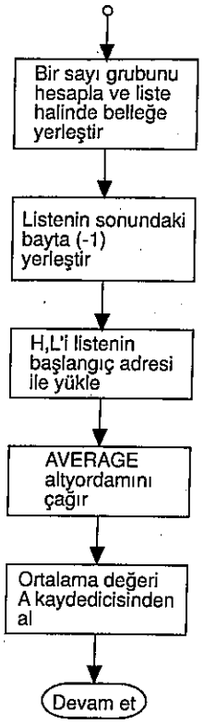
Bir önceki örnekte, ana programdan altyordama sadece tek bir parametre aktarılması gerekiyordu. Bundan başka, alt programın görevi yalnızca bir zaman aralığı üretmek olduğundan, geriye, altyordamdan ana programa, parametre aktarımı yapılmamıştı.

Diğer altyordamlar, ana program ile büyük miktarlarda veri alışverişinde bulunmayı gerektirebilirler. Daha önce gösterildiği gibi (Kısım 5.3) bu, ana belleğe veri yerleştirmek ve altyordamın üzerinde bu veriye işaretçi vermek suretiyle gerçekleştirilir.

Bir pozitif sayı grubunun ortalama değerini, düzenli aralıklarla hesaplayan bir ana programı ele alalım. Bu işlevi gerçekleştirecek bir AVERAGE (ORTALAMA) altyordamı yazılacaktır.

Sayı grubunun, ana program tarafından gerçekleştirilen hesaplamalardan ileri geldiğini ve belleğe yerleştirildiğini varsayacağız. Ortalaması alınacak sayıların sayısı değişken olabilir. Bu nedenle, bir altyordam çağrısında, sözgelimi 20 sayı, diğerinde 55 sayı bulunabilir.

Altyordamın, bellekteki sayı listesinin sonuna ne zaman ulaştığını söyleyebilmesi için, en son sayı negatif yapılır. Bu değer, liste bitiminin bir göstergesi olarak işlev görür ve ortalamaya katılmaması gerekir.



Şekil 7.15

Bu nedenle, ana programın akış diyagramının bir bö
Şekil 7.15'de gösterildiği gibidir. Program bir değerler gr
nu hesaplar ve onları ana belleğe yerleştirir. Listenin so
negatif bir sayı koyar (herhangi bir değer verilebilir, an
genellikle -1 uygundur). Liste, bir başka veri tarafından i
edilmemiş, uygun herhangi bir bellek alanına yerleştirile
Yerini işaret etmek üzere, H,L kaydedici çifti, altyord
girilmeden hemen önce, listenin ilk baytının adresi ile yükle

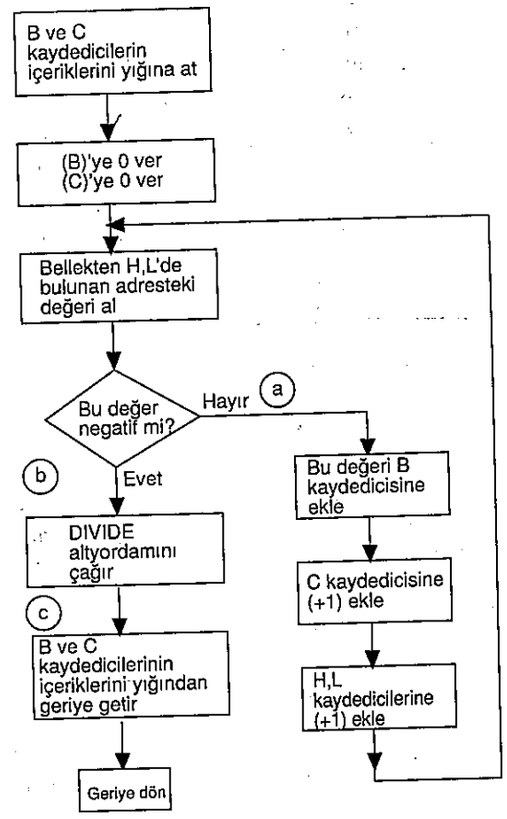
Altyordam tarafından hesaplanan ortalama değer, geriye
işlemci kaydedicisindeki ana programa aktarılır.
Daha sonra ana program, diğer başka ortalama değerle
hesaplanması gerektiğinde, altyordamı çağırarak işlemine d
vam eder.

AVERAGE (ORTALAMA) Altyordamı İstenen ortalamayı h
saplamak için yazılan altyordamın akış diyagramı, Şek
7.16'da gösterilmiştir. Bunun çalışma biçimi, ortalaması al
nacak değerleri bellekten getirmek ve her defasında birini B
kaydedicisine eklemek üzerine kuruludur. Bu nedenle, B
kaydedicisi, bu değerlerin *toplamını* tutmak için kullanılır
B'ye eklenen değerlerin *sayısı* ise C kaydedicisinde tutulur.

Böylece, programın ana döngüsü, sürekli olarak, bellekten bir
değeri getirmek, bu değeri B kaydedicisine eklemek, C
kaydedicisine '1' eklemek (yani C'yi '1' artırmak) ve H,L
kaydedicilerini '1' artırmak saykılını tekrarlar. Bu son işlem
ortalamaları alınacak değerleri birbiri ardı sıra sıralayarak,
programın, her defasında bir bayt olmak üzere, bellek boyunca
ilerlemesini sağlar.

Yukarıda anlatılan saykıl, bellekten alınan değerler pozitif
olduğu sürece devam eder. Ancak, negatif bir değerle karşıla-
şıldığında, altyordam kodu test kutucuğundan (b) noktasına
çıkar (Şekil 7.16).

Bu aşamada, B kaydedicisindeki toplamı, C kaydedicisinde
kaydedilen değerlerin sayısıyla bölmek suretiyle, istenen or-
talamayı hesaplamak gereklidir. Bunun için, diğer bir altyor-
dam, DIVIDE (BÖL), çağrılır.



Şekil 7.16

Burada, DIVIDE altyordamların ayrıntılarına girilmeyecektir. Bu altyordamın, iki giriş parametresini alıp, bunları, B ve C kaydedicilerine aktardığını ve B'nin içeriğini C'ninki ile böldüğünü söylemek yeterlidir. Sonuç, A kaydedicisine yerleştirilir ve çağıran programa dönüş yapılır.

Bu nedenle, Şekil 7.16'nın (c) noktasında, istenen ortalama, AVERAGE altyordamından, ana programa geriye aktarılmaya hazır biçimde A kaydedicisinde bulunmaktadır. Bununla birlikte, bu dönüş yapılmadan önce B ve C kaydedicilerinin içerikleri, tekrar orijinal değerlerine getirilir.

AVERAGE altyordamın kodu, Tablo 7.3'de verildiği gibidir. Uyumlu olması açısından, kodun bellekte başlangıç adresi olarak yine 0301 gösterilmektedir ve yığının başlangıç noktası ise 0404 adresidir.

Tablo 7.3

Adres	Komut: Ondalık	Komut: On altılı	Anlamı
0301	11000101	C5	(B) ve (C)'yi yığına at
0302	00000110	06	(B)'ye 0 değeri ver
0303	00000000	00	
0304	00001110	0E	(C)'yi 0 değeri ver
0305	00000000	00	
0306	01111110	7E	(H,L) adresindeki veriyi bellekten A'ya aktar
0307	11000110	C6	
0308	00000000	00	A'daki değerin negatif olup olmadığını
0309	11111010	FA	Negatif ise, 0313 adresine atla
030A	00010011	13	
030B	00000011	03	
030C	10000000	80	A'ya (B)'yi ekle
030D	01000111	47	(A)'yı B'ye aktar
030E	00001100	0C	(C)'yi 1 arttır
030F	00100011	23	(H, L)'yi 1 arttır
0310	11000011	C3	
0311	00000110	06	0306 adresine atla
0312	00000011	03	
0313	11001101	CD	
0314	XX	XX	DIVIDE'ı çağır (DIVIDE'ın
0315	YY	YY	adresi YYXX'dir)
0316	11000001	C1	(B) ve (C)'yi geriye al
0317	11001001	C9	RETURN (Geriye Dön)

Parametre-aktarma alanının, yani ortalamaları alınacak değerleri saklamak için kullanılan bellek bölümünün, 0501 adresinden başladığı varsayılmıştır.

Birkaç noktanın üzerinde biraz durulması gerekir:

1 (B) ve (C)'yi '0'a kurmak için kullanılan komutlar, 'Dolaysız 0 taşı' komutlarıdır. Bunlar, sadece, uygun kaydedicilere '0' değerini yerleştirirler.
2 Bellekten, 0306 adresindeki komut ile A kaydedicisine taşınan değerin negatif olup olmadığını test etmek için, bir 'Negatifse atla' komutu kullanılır (0309 adresindeki FA çalıştırılır). Bununla birlikte, bu komut 'işaret' durum bayrağını test ederek çalışır ve bu bayrak, sadece bir taşıma komutu gerçekleştirildiğinde etkilenmez. Bu nedenle, 'Dolaysız 0 ekle' komutu, 'atlama'dan önce 0307 ve

Tablo 7.4

Adres	Komut: İkili	Komut: On altılı	Anlamı
00B0	00110001	31	Yığın göstergesini kullanıma hazırla
00B1	00000100	04	
00B2	00000100	04	
00B3			
.	.	.	
.	.	.	
.	.	.	
00F4	00111110	3E	A kaydedicisine (-1) yerleştir
00F5	11111111	FF	
00F6	01110111	77	Belleğe (-1) yaz
00F7	00100001	21	(H,L)'yi 0501 (parametre alanının
00F8	00000001	01	adresi) ile yükle
00F9	00000101	05	
00FA	11001101	CD	AVERAGE alt programını çağır
00FB	00000001	01	
00FC	00000011	03	Bir sonraki komut
00FD			
.	.	.	
.	.	.	
.	.	.	

0308 adreslerine eklenmiştir. Bu komut, A kaydedicisindeki değeri değiştirmez, ancak işlemin sonucu tarafından belirtildiği biçimde, işaret bayrağına uygun değeri verir.

3 A'daki değeri B'ye eklemek için, iki işlem yapılır. İlk olarak, B'deki değer A'ya eklenir ve ardından sonuç A'dan B'ye taşınır. Buna gerek duyulur, çünkü bu aritmetik işlem yalnızca A kaydedicisinde gerçekleştirilebilmektedir.

Ana program kodu Ana program kodunun altyordama girişle ilgili bölümleri Tablo 7.4'de gösterilmektedir. Bir önceki örnekte olduğu gibi, yığın işaretçisi, programın başında kullanıma hazırlanmalıdır.

00F6 adresindeki komut, akümülatörde tutulan (-1) değerini belleğe aktarır. Bellekte erişilen adres, H,L kaydedici çiftindeki adrestir. Yukarıda, bu kaydedici çiftinin, parametre alanını doldurmak için gereken adresleri tutmak üzere ana programda kullanıldığı ve sonuç olarak, son parametreyi tutan baytın ardındaki baytı işaret etmek için düzenlendiği varsayılmıştır.

AVERAGE altyordamının assembly dili versiyonu Bir önceki kısımda sıralı AVERAGE altyordamının versiyonu tabii ki makine dilindedir. Bu yaklaşım CALL ve RETURN altyordam komutlarının işleyişinin tam olarak görülebilmesi ve programın çeşitli parçaları için kullanılan adreslerin açık bir şekilde ifade edilebilmesi için seçilmiştir.

Altyordam bir assembly dili versiyonu aşağıda verilmektedir :

```
Assembly komutları          Anlamı
        PUSH  B       ;(B) VE (C)'yi yığına at
        MVI   B,00    ;(B)'ye 0 değeri ver
        MVI   C,00    ;(C)'ye 0 değeri ver
SONR:   MOV   A,M     ;Veriyi bellekten A'ya taşı
        ADI   00      ;A'ya dolaysız 0 ekle
        JM    HESAP   ;Negatif ise HESAP'a atla
        ADD   B       ;(B)'yi A'ya ekle
        MOV   B,A     ;(A)'yı B'ye taşı
        INR   C       ;C'ye 1 ekle
        INX   H       ;H,L kaydedici
                      ;çiftine 1 ekle
        JMP   SONR    ;SONR'ye atla
HESAP:  CALL  BÖLME   ;BÖLME altyordamını çağır.
        POP   B       ;B ve C'deki değerleri geriye al
        RET           ;Geriye dön
```

Görüldüğü gibi, bu versiyonun okunması ve anlaşılması, makine kodu eşdeğerinden çok daha kolaydır.

S 7.4-7.7

Sorular

7.1 Program sayıcısının içeriklerinin, bir altyordama girişte saklanabileceği çeşitli yolları açıklayın. Her bir yöntemin:

(a) İç içe altyordam ve
(b) Özyineli altyordamlara uygulanabilirliğini açıklayın.

7.2 Bir ana program, sırayla B altyordamını çağıran bir A altyordamını çağırmaktadır. B altyordamının kendisi ise bir başka C altyordamını çağırmaktadır. A alt yordamı, bellekte 03A0, B altyordamı 03D4, C altyordamı ise 0605 adresinden başlamaktadır.

A'yı ana programdan çağıran CALL komutu 00BE adresinde, B'yi A'dan çağıran CALL komutu 03B6 adresinde, C'yi B'den çağıran CALL komutu da 04E1 adresinde ise, programın şu noktalarında, program sayıcısının, yığın işaretçisinin ve yığının içeriklerini gösterin:

(a) A altyordamına giriş yapıldıktan hemen sonra
(b) B altyordamına giriş yapıldıktan hemen sonra
(c) C altyordamına giriş yapıldıktan hemen sonra
(d) B altyordamından, A'ya dönüş yapıldıktan hemen sonra

Ana programın, başlangıç olarak, yığın işaretçisini 1000 değeri ile yüklediğini varsayın.

7.3 İşlemci kaydedicilerinin içeriklerini saklamak için kullanılan komutların, bir altyordam çağıran programa ya da altyordamın kendisine nasıl yerleştirilebileceğini gösterin. Her bir yöntemin özelliklerini tartışın.

7.4 Intel 8085 mikroişlemcisi:
DAD B
DAD D komutlarını içerir.

İlk komut, B ve C kaydedici çiftinin içeriklerini, H, L kaydedici çiftinin içeriklerine ekler. İkincisi ise, D, E kaydedici çiftinin içeriklerini, H, L çiftininkine ekler. Her iki durumda da, sonuç H, L kaydedici çiftine yerleştirilir.

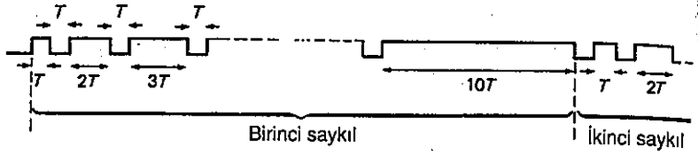
Toplama işlemi yapmak üzere bu komutları kullanarak, 0 ile *N* arasındaki tek sayıların toplamını hesaplayan bir altyordam yazın. 0 ile 100 arasında bir değer olan *N*, bir parametre olarak altyordama aktarılacaktır.

7.5 *N!*'i hesaplayacak bir altyordam yazın. [*N!*'in değeri = *N* x (*N*-1) x (*N*-2) x (*N*-3) x ... x2 x 1. Böylece, örneğin, 4! = 4x 3x 2 x1 = 24'e eşit olmaktadır]. 0 ile 7 arasında değişebilen *N* değeri, altyordama, D kaydedicisinden aktarılacaktır. +32.768'e kadar çarpma işlemi yapabilen bir MULTIPLY altyordamının elinizde mevcut olduğunu varsayın.

7.6 *N* (on altılı) ile *M* (on altılı) arasındaki tüm bellek konumlarına '0' değeri atayacak bir altyordam yazın. Hem *N*, hem de *M*'nin, 0000-FFFF aralığında değiştiğini ve altyordama parametre olarak aktarıldığını varsayın.

7.7 Bir sayı grubu (X_n), 0B00-0CFF adres bloğuna yerleştirilmiştir. Bu
X0, 0B00 adresindeki baytta; X1, 0B01'de; X2, 0B02'de, vb. tutulm
Altyordama girilen herhangi bir n değeri için, $1 + X_n + X_n^2$'nin d
hesaplayan bir altyordam yazın.

7.8 Bir mikrobilgisayar sistemi, Şekil 7.17'de görüldüğü gibi bir darbe
üretmek için kullanılacaktır. Dizideki darbelerin uzunluğu, 10 adımda T



Şekil 7.17

10T'ye artmaktadır. Ardından saykıl, yine T genişliğinde bir darbe ile ba
yarak tekrar eder. Darbelerin arasındaki boşlukların süresi, hepsinde sabit
T uzunluktadır.

Bu darbe dizisini üretmek için, zaman gecikmesi üreten bir ana program
bir altyordamın nasıl kullanılabileceğini gösterin.

Bölüm 8 Kesmeler (Interrupts)

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

1 *Bir mikrobilgisayar ile çevresel birimlerin çalışma hızları arasında oldukça geniş bir farklılık olabileceğini değerlendirebilmeli,*
2 *Bu hız farklılığı nedeniyle, çevresel birim ve işlemcinin senkronizasyonunda (eşzamanlanmasında) bazı sorunların ortaya çıkabileceğini değerlendirebilmelisiniz. Bu problemler:*
   (a) *belli bir işlem zamanının boşa harcanması nedeniyle, işlemcinin verimsiz kullanımı ya da*
   (b) *etkin olmayan bir çevrebirim tepkisi ile sonuçlanır.*
3 *Kesmelerin, işlemcinin dikkatini çekmek istediğinde bir çevresel birim tarafından, gönderilen elektriksel sinyalleri olduğunu anlayabilmeli,*
4 *Bu tip bir sinyalin, işlemcinin yürürlükteki programının yürütümünü kesmesine ve kesme hizmeti programına girmesine neden olduğunu değerlendirebilmeli,*
5 *Kesme hizmet programının, kesmeye neden olan durumu ele aldığını ve ardından denetimi kesilmiş bulunan programa aktardığını değerlendirebilmeli,*
6 *Kesilmiş programın, tam kesildiği yerden, hiç kesintiye uğramamış gibi işlemine devam edeceğini değerlendirebilmeli,*
7 *Bu işlemin, her biri, gerektiğinde kesme ile sağlanan anlık eşzamanlama ile kendi hızında çalışan bir mikrobilgisayar ile çevrebirimi arasında yeterli veri aktarımını mümkün kılmak için kullanılabileceğini değerlendirebilmeli,*
8 *Bir mikrobilgisayar sisteminin pek çok kesme girişine sahip olabileceğini; böyle bir durumda, eğer iki ya da daha fazla kesme isteği olursa, olası karışıklıkları önlemek için, kesme girişlerine önem derecelerine göre farklı öncelikler verilmesi gerektiğini anlayabilmeli,*
9 *İç içe alt programlar ile kesilmiş kesme hizmet programları arasındaki benzerlik de dahil olmak üzere, altyordam ile kesme hizmet programları arasında benzerlikler olduğunu değerlendirebilmeli,*
10 *Kesme hizmeti sırasında, ana program tarafından kullanılan kaydedicilerin içeriklerini saklamak ve tekrar geriye almak için, genellikle bazı özel düzenlemeler yapıldığını anlayabilmeli,*
11 *Bu türde kaydedici içeriklerinin saklanmasının, normalde mikrobilgisayar yığını kullanarak gerçekleştirildiğini değerlendirebilmeli,*
12 *Özellikle, gerçek-zamanlı bir sistemde, kesme ile bağlantılı bazı tehlikeler bulunduğunu, çünkü her bir kesme hizmet programının çalışmak için belli bir zamana gerek duyduğunu değerlendirebilmeli,*

13 Kesmelerin kabul edilebilecekleri hızın, onlara hizmet vermek için ger toplam işlem zamanı ile sınırlı olabileceğini değerlendirebilmeli,
14 Intel 8085 gibi, pratik bir mikrobilgisayar sisteminde kesmeleri denetlemek kesme işlemini gerçekleştirmek için bulunan olanakları kullanabilmeli,
15 Mikrobilgisayar sistemlerinde, kapsamlı kesme olanaklarının geliştirilmesin yardımcı olmak üzere, programlanabilir kesme denetleyicileri gibi başka de relerin mevcut bulunduğunu değerlendirebilmelisiniz.

8.1 Giriş

Bu bölümde daha sonra açıklanacağı gibi, kesmeler, bir mikrobilgisayar sistemini ona bağlı çevresel birimlere senkronize etmek (eşzamanlamak) için kullanılan sinyallerdir. 1. Bölümde, bilgisayar ile çevrebirimleri arasındaki hız farklılıklarından söz edilmişti. Farklılıkların oluşturacağı bu sorunların üstesinden gelmek için, genellikle kesmelere ihtiyaç duyulur.

Bir mikrobilgisayar ile dış dünya arasındaki veri alışverişi (2., 3., ve 4. Bölümlerde anlatıldığı gibi); programa, veri aktarım komutları yerleştirilerek gerçekleştirilir.

Bu tür komutlar, örneğin, Intel 8085'deki IN ya da OUT gibi özel çevresel aktarım komutları, ya da eğer bellek-haritalı G/Ç (Bölüm 4) kullanılacaksa normal veri, MOV komutları olabilir. Her iki durumda da, her bir veri parçasını çevresel birime göndermek ya da çevresel birimden veri almaktan sorumlu özel bir program komutu vardır.

Bu basit sistem, bir çok çevrebiriminin çalışma yavaşlığı nedeniyle, tek başına yeterli değildir.

Çevrebirimlerinin Hızları

Mikrobilgisayarın bir komutu tipik olarak bir ya da iki mikrosaniyede yürütebileceğinden 1. Bölümde söz edilmişti. Çevrebirimlerin çalışma hızları çok daha yavaştır. Örneğin, bir kağıt bant okuyucusunun 8 bitlik bir karakteri okuması yaklaşık 1 milisaniye ve bir teletaypın bir karakteri yazması ise yaklaşık 100 msn sürer. Bu nedenle, bant okuyucusu, bilgisayarın binde biri bir hızda; teletayp ise yüz binde biri bir hızda çalışır.

Bilgisayar ve çevrebiriminin işlemlerini senkronize etmek için bazı araçlara sahip olmak gerektiği açıktır.

8.2 Kesme ihtiyacı

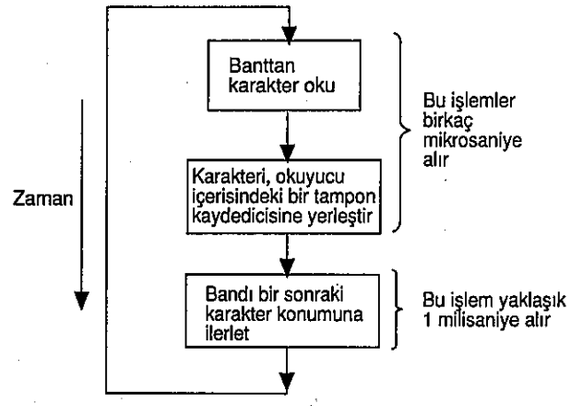
Çevresel birimlerle eşzamanlama

Kesme ihtiyacı, en iyi pratik bir mikrobilgisayar-çevrebirimi iletişimi örneği ele alınarak açıklanabilir. Bir kağıt bant okuyucusunun bir mikrobilgisayara bağlanacağını varsayalım. Yukarıda bahsedildiği gibi bu, bir milisaniyede maksimum 8 bitlik bir karakter hızında bilgi sağlayabilir.

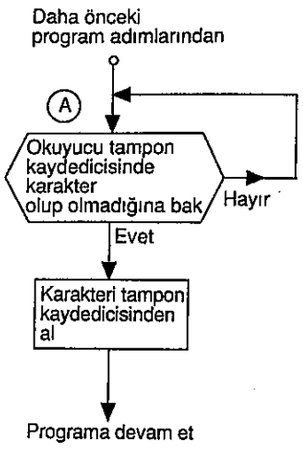
Okuyucunun çalışma şekli şöyledir :

1 Yarım inç genişliğinde bir kağıt bant parçası üzerinde, 8 delikli bir dizi olarak delinmiş bir karakteri algılar.
2 Bu karakterin değerini, 8 bitlik bir ikili sayı olarak, denetim devresindeki bir *tampon kaydedicisine* (bir sabitleştirici) aktarır.
3 Bandı, bir inçin onda biri oranında, bir sonraki 8 bitlik karakterin konumuna taşımak için gerekli mekanik işlemleri başlatır.
4 1. basamağa döner ve 1-3. adımları tekrar eder. Bu dizi Şekil 8.1'de bir akış diyagramı olarak gösterilmiştir.

Görülebileceği gibi, ilk iki adım (karakteri algılama ve onu bir tampon kaydedicisine yerleştirme) sadece birkaç mikrosaniye sürer. Mekanik olarak bandı, okuyucu üzerinden bir sonraki karakter konumuna taşımak yavaş bir işlemdir.



Şekil 8.1 Kağıt bant okuyucusunun çalışması



Şekil 8.2

Banttan okunan karakter, tampon kaydedicisine yükle
yaklaşık 1 msn sonra, bu kaydedicinin, bir sonraki kara
tekrar yüklenişinin başlangıcına kadar, bilgisayara al
hazır durumdadır.

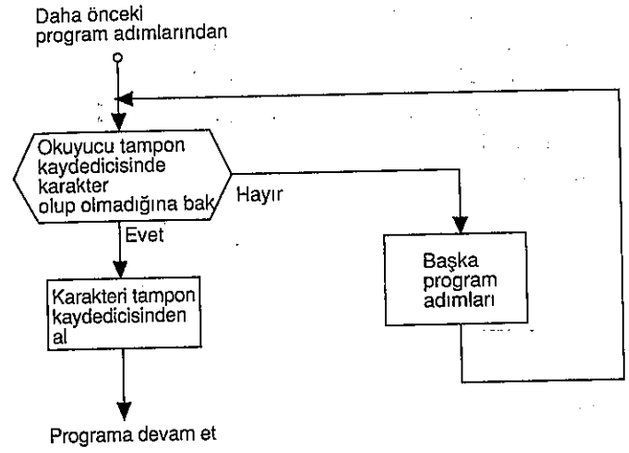
Karakter kullanıma hazır hale gelir gelmez bilgisaya
program komutuyla, karakteri tampondan alabilir. Daha
gösterildiği gibi, bu aşama yalnızca bir kaç mikrosaniye
mektedir. Bu nedenle bilgisayarın, karakterin alınmas
sonra, bir milisaniye süreyle ya da bir sonraki karakter kull
labilir hale gelinceye kadar beklemesi gerekir. Bunu yap
mesi için, bekleme periyodunun bittiğini ve yeni bir kara
tamponda hazır bulunduğunu algılayabilmesi gerekir.

Bu algılama, okuyucudan ya da okuyucuyu mikrobilgisa
anayoluna bağlayan arabirimden alınan bir durum sinyali
gerçekleştirilir. Kısım 2.4'de, programlanabilir çevresel a
birimindeki (PPI) kaydedicilerin, bilgi aktarımlarının durum
nu kaydetmek üzere bu biçimde kullanıldıklarından söz edil
mişti.

Bilgisayarın, bir IN komutu kullanmak suretiyle, okuyucun
durumunu algılayabildiğini varsayalım. PPI'nin C port
kaydedicisindeki bitlerin biri, okuyucu tamponundan bir ka
rakter okunabildiğinde, sözgelimi 1; diğer anlarda ise 0 olsu
O halde, mikrobilgisayarın çalışması, Şekil 8.2'de gösterildi
gibi olacaktır.

Okuyucudan bir karakter alınacağı zaman, program ilk ola
durum bitini inceler. Bu bit 1 ise, bir sonra gelen kom
okuyucu tamponundan bir karakter okur ve program devam
eder. Ne var ki, eğer bitin değeri 0 ise program, okuyucunu
işlem saykılını bitirmesini ve bir karakterin kullanılmaya hazır
olmasını bekleyerek döngüye girer. Program, bekleme döngü
sünün her saykılında okuyucunun durumunu test eder.

Bu işlem, mikrobilgisayarın okuyucudan, yalnızca doğru za
manlarda veri okumasını sağlar. Bununla birlikte ciddi bir
dezavantajı da vardır: mikrobilgisayarın çalışma hızını ban
okuyucusunun hızına düşürür.



Şekil 8.3

Eğer banttan bir kaç karakter alınacaksa, bilgisayar, ilkini okuduktan sonra, hemen (A) noktasına döner. (Şekil 8.2). O zaman, bir sonraki karakteri almadan önce, 'Kullanıma hazır olup olmadığını test et' döngüsü etrafında yaklaşık bir milisaniye dönmesi gerekir. Bunu okuduktan sonra, tekrar (A) 'ya döner.

Görülebileceği gibi, aygıtın çalışması, birkaç mikrosaniye süren, karakteri alan bir komut tarafından takip edilen ve bir milisaniye civarında süre alan uzun bir 'Kullanılabilir bir karakter varsa test et' komut dizisinden ibarettir ve ardından 1 milisaniyelik başka bir döngü gelir ve işlem böylece sürer.

Kuşkusuz bilgisayar programının, okuyucu tampon kaydedicisine yerleştirilecek bir karakteri bekleyerek döngü yapması *zorunlu* değildir. Buna bir alternatif, Şekil 8.3'de gösterilmektedir. Bunda, mikrobilgisayar, bir karakterin henüz kullanıma hazır olmadığını algıladıktan sonra, 'Hazır olup olmadığını test et' komutuna dönmeden önce diğer bazı program komutlarını yürütür. Diğer bir deyişle, okuyucunun bir karakteri getirmesini beklerken, bazı yararlı hesaplamalar yapar.

Bu durum daha önceki olanakların gelişmiş bir halidir, ancak yine de bazı dezavantajlara sahiptir. Örneğin, Şekil 8.3'deki 'Diğer program adımları' kutucuğuna ne kadar yararlı işler konulabileceğini düzenlemek zordur.

Burada, yalnızca küçük boyutlarda bir işlem yapıldığında bile, bilgisayar, yine de, 'Hazır olup olmadığını test et' işlemleri nedeniyle önemli miktarda bir zaman harcar. Diğer taraftan, çok fazla işlem yapıldığında da bilgisayar 'Hazır olup olmadığını test et' komutuna çok sık başvuramayacak ve okuyucuya gönderilen

yanıt zayıf kalacaktır, yani bilgisayar okumaya çalışmadan okuyucu, önemli oranda bir süre için, alınacak olan karakteri tutmuş olacaktır.

Bu yararlı hesaplamalar ve iyi yanıt arasında, program hangi uzunlukta seçilirse seçilsin, kaçınılmaz olarak bir dengeleme yapmak gerekecektir. Kesmelerin kullanımı bu problemi ortadan kaldırır.

Gerçek-zamanlı yanıt

Burada, kesme sisteminin yerine getirdiği bir başka koşuldan da söz edilmesi yararlı olacaktır. Bu, bilgisayarın, dış dünyadan gelen isteklere hızlı bir şekilde yanıt verebilmesinin çoğunlukla önemli olmasıdır. Bazı durumlarda, bilgisayarın dışında meydana gelen bir olay, bilgisayarın bu olayla hemen ilgilenmesini gerektirecek ivedilikte öneme sahiptir.

S 8.2

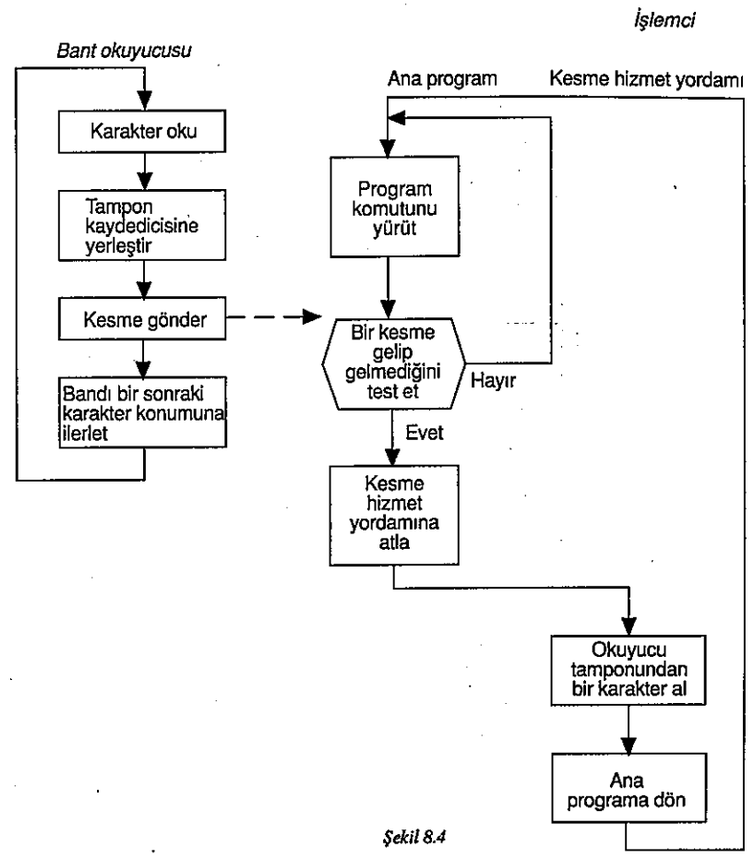
8.3 Kesmeler Nasıl Çalışır

Kesmeler, mikrobilgisayarın işlemcisine doğrudan beslenen elektriksel sinyallerdir. Bu sinyal, işlemcinin yürütmekte olduğu programı durdurmasına ve kesmeye neden olan durumla ilgilenmesine neden olur.

İşlemcinin bir programı yürüttüğünü, yani bellekte saklı bir komut dizisi boyunca ilerlediğini ve bu komutların ifade ettiği işlemleri yürütüyor olduğunu varsayalım. Her bir yürütme saykılının bitiminde (Bölüm 1), işlemcinin devreleri, üzerinde bir sinyal bulunup bulunmadığını görmek amacıyla kesme girişini gözden geçirir.

Eğer bir kesme sinyali varsa, şu işlemler yapılır:

1 İşlemci, program sayıcısının içinde tutulan değeri saklar. Ayrıca, işlemci kaydedicilerinin tümünün ya da bir bölümünün içerdiği değerler de saklanır.
2 Programın yürütülmesi, *kesme hizmet yordamı* ya da *kesme yönetim yordamı* denilen özel bir program kodu kısmına dallanır.
3 Kesme hizmet yordamı, -kesmeye neden olan durum ile ilgili hangi giriş ve çıkış işlemleri gerekiyorsa- bu işlemleri gerçekleştirir.
4 Program sayıcısı ve işlemci kaydedicilerinin içerikleri, kesmeden önceki değerlerine geri alınırlar.
5 Kesilen programın yürütülmesine, kaldığı yerden devam edilir.



Şekil 8.4

Buna bir örnek olarak, daha önce anlatılan bant okuyucusunu ele alalım. Eğer kesme yöntemi, bu birim ile bilgisayar arasında bağlantı kurmak amacıyla kullanılırsa, sistemin çalışması Şekil 8.4'de gösterildiği gibi olacaktır. Şeklin solunda gösterilen okuyucunun çalışması, temelde daha önce olduğu gibidir. Şekil 8.1'deki blok diyagram ile karşılaştırıldığında görülen tek değişiklik, tampon kaydedicisine bir karakter yerleştirdiğinde okuyucu, işlemciye bir kesme göndermesidir. Bu, şekilde kesikli ok ile gösterilmektedir. Aslında, kesme işlemciye bir karakterin kullanıma hazır olduğunu bildirmektedir.

Okuyucu çalışırken, işlemci bir program yürütmektedir. Bu süreç, işlemcinin kesme hizmet yordamına dallandığı noktada, okuyucudan kesme gelene kadar devam eder. Bu yordama okuyucudan gelen kesme nedeniyle girildiğinden, yordam özel olarak okuyucu ile birlikte çalışacak şekilde tasarlanmıştır. Bu nedenle,

bu yordam içinde, okuyucu tamponundan işlemciye karakter alan bir IN ko
bulunur. Karakter okunduğunda, ana programa dönülür. Kesme hizmet yordam
girişte, kaydedicilerde tutulan değerler, eski değerlerine geri alınabilirse, daha
ifade edildiği gibi ana programa, *tam olarak* kesildiği yerden devam edilebilir

Bu aşamada, IN komutunun yürütümünden önce işlemcinin, okuyucunun durum
nu test etmesine ilişkin herhangi bir sorun yoktur. Karakter, alınmaya hazır durum
da olmalıdır, aksi halde, kesme sinyali üretilmeyecek ve kesme hizmet program
girilmeyecektir.

Böylece bu durumda, kesme kullanımı, okuyucunun ve işlemcinin kendi hızların
çalışmasına ve aralarında bilgi alışverişine gerek duyulduğunda, bu kısa s
boyunca, ikisinin senkronize edilmesine izin verir.

Bu tür bir sistemde, işlemci zamanının çok küçük bir bölümü harcanır. Çoğu zam
işlemci, sanki okuyucu hiç yokmuş gibi davranarak sadece programını yürütür
programı kesmek üzere, kesme hizmet yordamını, bu durumda yaklaşık bir mil
saniyelik sık olmayan aralıklarla çalıştırmak için, yalnızca birkaç mikrosaniye
ihtiyaç duyar.

Aynı mekanizma, mikroişlemcinin dış isteklere yanıt vermesini sağlamak amacıyl
da kullanılabilir. Eğer bu istekler, işlemciye kesmeler şeklinde gönderilirse, bu
isteklere ilişkin işlemleri uygulamak üzere birleşik hizmet yordamları kullanılabi
lir.

Çoklu kesmeler

Yukarıdaki anlatımdan, bir mikrobilgisayar sisteminin genelde, birden fazla kesme
girişine ilişkin işlemeleri yönetme gereği duyacağı açıkça görülmektedir. Bu
durumda, işlemcinin, bu girişlerin her birine ilişkin uygun hizmet yordamlarına
dallanabilmesi için, önce, kesmeleri birbirinden ayıracak bir yönteme sahip olması
gerekir.

Ayrıca, kesme girişlerinin önceliklerini belirleyecek bir yönteme de sahip olmalı
dır. Eğer birden fazla kesme hattı varsa, kaçınılmaz bir biçimde, bazen iki ya da
daha fazla sayıda hatta aynı anda kesme sinyali ortaya çıkabilecektir. Bu durumda
işlemci, ilk olarak hangi kesme isteğinin işleme alınacağını seçebilmelidir. Bu
nedenle, kesmelerin bir önem sırasına konulması gerekmektedir.

Son olarak, yazılım olanaklarının, işlemcinin kesme girişlerini denetleyebilmesini sağlaması gerekliliği vardır. Yani, makinenin komut kodunda, yazılımın denetimi altında, herhangi bir kesmeyi kullanıma açma (*yetkilendirme*) ya da kullanıma kapamaya (*yetkisizleme*) imkan verecek komutlar bulunması gerekir. Bu noktalar üzerinde daha sonra durulacaktır.

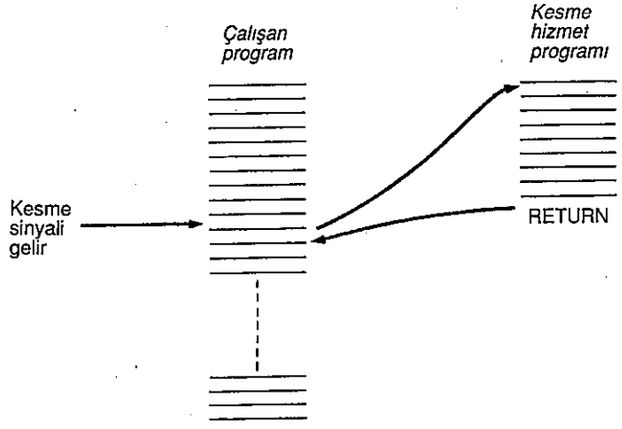
8.4 Altyordamlar ile kesmelerin karşılaştırılması

Kesme sistemleri ile altyordamlar arasında birçok benzerlikler vardır :

1 Bir dış birimden bir kesme sinyali alındığında, program yürütmesi ilgili kesme hizmet yordamına dallanır. Bir programın yürütümü esnasında bir altyordam çağrı komutu ile karşılaşıldığında, yürütme çağrılan altyordama dallanır.
2 Kesme hizmet yordamına girmeden önce işlemci, programdan dönüşün doğru biçimde gerçekleştirilebilmesi için, program sayıcısının içeriklerini saklar. İşlemci ayrıca program sayıcısının içerdiği değerleri , bir altyordama girmeden önce de saklar.
3 Bir kesme hizmet yordamından ve bir altyordamdan dönüşün her ikisi de, Intel 8085 'deki RET komutu gibi, girişte saklanan değeri kullanarak program sayıcısını tekrar yükleyen bir komut yardımıyla gerçekleştirilir.
4 Bir kesme hizmet yordamına veya bir altyordamına girişte program sayıcısında tutulan değeri saklamak için kullanılan yöntemler aynıdır. Burada yöntem tipik olarak, değerin sistem yığınının üzerine atılmasıdır.
5 Bir kesme hizmet yordamını yürütmeden önce, tıpkı bir altyordamın yürütümünden önce olduğu gibi, işlemci kaydedicilerinin içeriklerini saklamak gerekebilir. Eğer kaydediciler her iki tip yordamda da kullanılırsa, kesilen (ya da çağıran) programa devam edilmeden önce, kaydedici içerikleri tekrar geriye alınmalıdırlar.

Bu nedenle, bir kesme hizmet yordamının aslında, bir programda kullanılan CALL komutundan çok, dıştan gelen bir elektrik sinyaline karşılık olarak girilen bir altyordam olduğu görülebilir. Çünkü, kesme sinyali herhangi bir anda ortaya çıkabilir, ya da diğer bir deyişle, bir programın yürütülmesinde herhangi bir noktada, herhangi bir komuttan sonra hizmet yordamına giriş yapılabilir. Buna karşılık, bir altyordama girişte ise, altyordama dallanma yalnızca CALL komutu yürütüldüğünde gerçekleşir. Fakat, kesme nerede meydana gelirse gelsin, program sayıcısının içerdiği değerleri saklama ve geriye alma mekanizması her zaman çalışır.

Bir kesme sinyali alındığında, denetim akışının aldığı biçim Şekil 8.5'de şematik olarak gösterilmiştir. Altyordama girişle olan benzerlikler (Şekil 5.2) açıkça görülmektedir.



Şekil 8.5

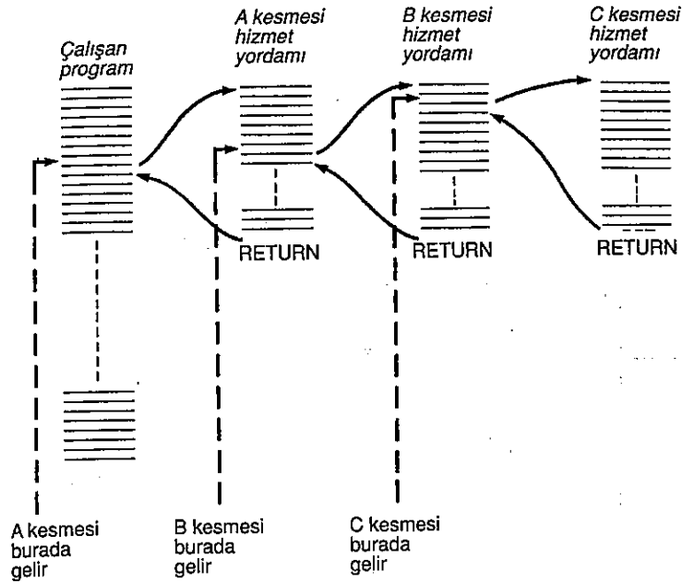
Daha önce açıklandığı gibi, bir kesme sinyalinin gelmesi, programın yürütülmesinin, o kesmeye ilişkin hizmet yordamına aktarılmasına neden olur. Fakat, pratikte bazı küçük karmaşıklıklar ortaya çıkar.

Çoklu kesmeler içeren bir sistemde, birkaç kesmenin hızlı bir şekilde art arda meydana gelmemesi için hiçbir neden yoktur. Ve eğer böyle bir durum ortaya çıkarsa, ikinci kesme, ilk kesmenin kesme yönetim yordamı yürütülüyorken ve üçüncü kesme de, ikinci kesmeye ilişkin kesme yönetim yordamı yürütülüyorken ortaya çıkabilir. § 8.3

Bu kesmelerin her birine ilişkin doğru işlemci yanıtı, her bir kesmeye ilişkin hizmet yordamına, o kesme meydana geldikten hemen sonra girmek olmalıdır (bu ifadeye ilişkin olarak yapılacak bazı değerlendirmeler vardır, ancak bunlar daha sonra alınacaktır). Böylece, durum Şekil 8.6'da gösterildiği gibi olacaktır. Bu işlemin, içe altyordam çağrıları ile benzerliği açıktır.

Ana program (çalışan program, Şekil 8.6) 'A' kesmesi ile kesildiğinde, A hizmet yordamına girilir. Bu arada, ana program çalışmaktayken, 'B' kesmesi gelir. Bu nedenle, A hizmet yordamının yürütülmesi durdurulur ve B hizmet yordamına girilir. Bunun ardından, benzer bir işlem, B tamamlanmadan önce C hizmet yordamına girişe neden olur.

C'den B'ye, B'den A'ya, oradan da çalışan programa geri dönmek için gerekli dönüş adresi, iç içe altyordamlarda olduğu gibi, bir yığına saklanır saklanmaz,



Şekil 8.6

verilen kesme düzeyi sayısı yazılım ile değil, genelde, mevcut işlemciye olan kesme girişi sayısı ile sınırlıdır. Bu konu daha sonra ele alınacaktır.

8.5 Öncelikler

Kesme sinyallerinin öncelikleri Kısım 8.3'de verilmişti. Orada ifade edildiği gibi, kesme sinyalleri, normalde, bir öncelik sırasına konur, yani her birine bir öncelik sırası atanır. Bu öncelikler, işlemcinin, birlikte ortaya çıkan iki ya da daha fazla kesme arasından seçim yapmasına izin verir; dahası, kesmelerin ve kesme hizmet yordamlarının birbirleriyle etkileşim yollarını denetleyebilirler.

Kesme sistemlerinin önceliklerinin tam olarak nasıl işledikleri, işlemciden işlemciye değişir. Bazıları, örneğin bu bölümde ileride tartışılan Intel 8085, bir kesmenin kabul edilmesinin ardından, geriye kalan diğer tüm kesmeleri otomatik olarak yetkisizler (kapatır). Diğerleri, öncelik kullanımını bir süre daha taşımaya devam eder.

Şekil 8.6'yı ele alalım. Burada, B kesmesinin gelmesinin, A hizmet yordamının yürütülmesinin kesilmesine neden olduğu ve -bazı sistemlerde olduğu gibi- B hizmet yordamına yapılacak girişin, ancak B'nin A'dan daha yüksek önceliğe sahip

olması halinde doğru olduğu varsayımı yapılmıştır. Eğer değilse, o zaman işl
B'ye dönmeden önce, A altyordamına ilişkin komutların yürütümünü tamam
caktır.

Bu düzenlemede, bir kesme sinyali, kendisinden daha yüksek önceliğe sahip d
bir kesme sinyalinin hizmet programını kesemez. Bu, kuşkusuz yerinde bir dü
lemedir. Örneğin, mikrobilgisayar sisteminde kritik hata koşuluyla ilgili yük
öncelikli bir kesmeye ilişkin hizmet yordamının, daha az önemdeki bir çevr
istekle kesilmemesi gerekir.

8.6. Veri saklama ve geriye alma ihtiyacı

Daha önce anlatıldığı gibi, kesmeler ve altyordam çağrıları arasında önemli b
zerlikler bulunmasına karşın, kesme bir anlamda bir altyordam çağrısından ç
daha az denetlenebilen bir olaydır. Bu, kesme sinyallerinin geliş zamanlarını
gelişigüzel olması nedeniyledir.

Altyordam çağrıları, program içerisinde, programcı tarafından belirli bir nokt
yapılır. Kesme ise, program içerisinde herhangi bir yerde ortaya çıkabilir. B
nedenle, kesilen programa dönmek için gerekli tüm veri ve parametrelerin, kesm
hizmet yordamı çalışmaya başlamadan önce saklanması çok önemlidir. Kesilen b
program, ancak bu yapıldığı sürece, tam olarak kesildiği noktada çalışmaya yeni
başlayabilir.

Veri ve parametreleri saklamak için, altyordam girişi ile bağlantılı şu iki yönte
ele alınmıştı:

1 Veriler, CALL komutu yürütülmeden önce, ana programa yerleştirilen komu
ları kullanarak, ya da
2 Veriler, altyordamın kendisinin başında bulunan komutlar kullanılarak sakl
nabilir. Bunlar, ana alt yordamın çalışmaya başlamasından önce yürütülür.

Kesme yönetim yordamları ile kullanıma, bu alternatiflerden yalnızca ikinci
uygundur. Kesme sinyallerinin gelişindeki gelişigüzellik, kesme yönetim
yordamına girmeden önce veri ya da parametreleri saklamanın mümkün olmadığı
anlamına gelmektedir. Bu nedenle, kesme yönetim yordamının kendisinin, para
metre ve veriyi saklamak için gerekli komutları içermesi gerekmektedir.

Bu nedenle yaygın düzenleme, kesme sinyalinin gelişi için, program sayıcısının
içeriklerinin saklanmasını ve kesme hizmet yordamına dallanılmasını sağlamaktır.

Bu programdaki komutlar, daha sonra, işlemci kaydedicilerinin içeriklerini, veri değerlerini, vb. hemen saklayacaktır.

Öncelikli kesmelerin esnekliğine ve bir kesme hizmet yordamının Şekil 8.6'da gösterildiği gibi kesilebilmesine olanak tanımak için, veri ve kaydedici değerleri normalde yığın kullanılarak saklanırlar. Bu nedenle, kesme hizmet yordamının başındaki ilk birkaç komut, PUSH işlemlerini gerçekleştiren komutlardır.

Benzer biçimde, hizmet yordamının bitimindeki RETURN komutundan hemen önceki bir iki komut da, POP işlemlerini gerçekleştiren komutlar olacaktır. Bu nedenle, 'RETURN' komutu yürütüldüğünde, kesilen program kaldığı yerden devam edebilir.

Burada bir diğer olanaktan da sözetmek gerekmektedir; yalnızca tek bir kesme düzeyine izin verildiğinde, Z80'de olduğu gibi (Bölüm 2) bir çift işlemci kaydedici takımının kullanımı çok yararlı olabilir. Kesme yordamına girildiğinde, kaydedici değiş-tokuş komutu (Kısım 2.2), kullanımdaki kaydedicileri ana kaydedici grubundan, alternatif kaydedici grubuna aktarmak için kullanılır. O zaman, kesme hizmet yordamı, ana kaydedici grubunun içeriklerini bozmaksızın, hesaplamalarında bu alternatif kaydedici grubunu kullanır. Kesme hizmet yordamından ana programa dönüş yapıldığında da, kullanımdaki kaydedici içerikleri ana kaydedici grubuna geri alır ve program çalışmaya devam eder.

Kaydedici grubunu bu şekilde değiş-tokuş etmenin avantajı, yalnızca tek bir komut çalıştırma gerektirmesidir. Veri değerlerini, bir yerden diğerinin aktarılmasına ya da belleğe erişim yapılmasına gerek yoktur. Bu nedenle işlem çok hızlı gerçekleş-
S 8.4 mektedir.

8.7 Kesmelerin tehlikesi

Kesmeler, mikrobilgisayardan yanıt isteyen birimlere karşılık olarak verilen sinyallerdir. Ancak burada, çoğunlukla zaman açısından bir *kritiklik* söz konusudur, yani işlemci kesme geldikten sonra sınırlı bir süre içerisinde yanıt vermek zorundadır. Eğer yanıt çok yavaş olursa, gereken önlemleri almada çok geç kalınabilir.

Yanıta karşılık vermenin ivedi olması bazı sorunlar doğurabilir. Bu, en kolay biçimde bir örnek üzerinde incelenebilir.

Kesmeler, gerçek-zamanlı bir işletim ortamında çalışan bir mikrobilgisayar siste-

minde özellikle yararlı olabilir. Bunda mikrobilgisayar, kendisine bağlı bazı ç
çek işlemleri kontrol etmekten sorumludur. Böyle pek çok uygulama var
Bunlardan önemli biri, kimyasal işleme tesislerinin denetlenmesinde ortaya çık
Mikrobilgisayar; sıcaklık, akış hızı, vb. çeşitli üretim değişkenlerinin ölçümle
denetleyebilir. Ve ardından, sistem, istenen çalışma sınırları içerisinde faali
gösterecek biçimde, valfleri, ısıtıcıları, pompaları ve diğer denetim birimleri
çalıştırır.

Bu tür bir uygulama, mikrobilgisayar sisteminin çok yoğun giriş/çıkış trafiğin
üstesinden gelebilmesini gerektirir. Bundan başka, mikrobilgisayarın hangi kont
değişkenlerinin değiştirilmesi ve bunlara hangi değerlerin atanması gerektiğ
belirlemek için gerekli hesaplamaları da yapması gerekir.

Bu nedenle, sistemin *kesme ile sürülen* tipte düzenlenmesi uygun olacaktır; yan
mikrobilgisayar, bir kesme gelinceye kadar çalıştırdığı bir ana program içerir. Dah
sonra, mikrobilgisayar, kesmeye neden olan durumla ilgilenir ve ana program
devam eder. İşlemciye bağlı pek çok gereç ve birim bulunduğundan, birçok kesme
türü olabilir. Örneğin, eğer bir sıcaklık algılayıcısının periyodik aralıklarla işlem
ciye okunması gerekiyorsa, sistemde bir zamanlayıcı yongası (Bölüm 2) tarafından
üretilen düzenli kesmelerin, bir kesme yönetim yordamına girmesine neden
olması gerekir. Bu da, algılayıcıdan sıcaklık değerini sırasıyla okuyacak komutlar
içerir.

Bu yöntemin avantajı, algılayıcıdan değer okuma aralıklarının zamanlamasındaki
sorumluluğun, mikroişlemcinin üzerinden alınıyor olmasıdır. Bu nedenle, mikro
işlemci, gerekli kontrol hesaplamalarını gerçekleştirmek üzere serbest kalmaktadır.

Her kesme yanıtı, işlem süresine ek bir *yük* getirir. Bir kesme geldiğinde, gördüğü
müz gibi, işlemcinin kesme programına dallanması gerekir. Bu arada program
sayıcısı ve işlemci kaydedicilerin içeriklerinin de saklanması gerekir. Bu nedenle
kesme programı herhangi yararlı bir iş yapmadan önce bile, işlemci zamanının bir
bölümünü harcayacaktır. Buna ek olarak, işlemci zamanının bir bölümü de, kes
meden ana programa dönüşte harcanır. İşlemci kaydedici içeriklerinin tekrar geriy
alınması ve program kaydedicinin dönüş adresi ile tekrar yüklenmesi gerekir.

İşte bu nedenle, mikrobilgisayar sistemiyle ilgili her bir kesmenin, gerçek işlemler
için kullanılabilen zamanı (en azından potansiyel olarak) azalttığı görülebilir. Ağır
işlem yüküyle çalışan bir sistemde bu, sistemin gelen çeşitli isteklerle uğraşmadaki
hızını azalttığından sistemin çalışmasını ciddi biçimde etkileyebilir.

Bu tür gerçek-zamanlı bir sistemde bu, mikrobilgisayarın, gereken tüm işlemlerle uğraşabilecek biçimde düzenlenmesi önem taşımaktadır. Sistem, giriş okuma değerlerini alabilmeli ve tesisin kararlı bir şekilde çalışmasını sağlamak üzere istenen hızda, çıkış denetim sinyallerini güncelleyebilmelidir.

Bundan başka, ortaya çıkan bir kesmenin durdurulması da mümkün olabilmelidir. Eğer işlemci, kritik bir zaman periyodunda tamamlanması gereken bir hesaplama ile uğraşıyorsa, bunun işlemin ortasında kesintiye uğramamasını sağlamak çok önemlidir. Bunu sağlayabilmenin tek yolu, işlemcinin, hesaplamayı bozabilecek her türlü kesmeyi etkisiz hale getirmesidir (yetkisizlemesidir). Bu, program komutları ile yapılabilir.

Bu nedenle, Intel 8085 işlemcisinde, tek baytlık DI *'kesmeleri yetkisizle'* komutu kullanılabilir. Bu türde bir komut, ardından bir *'kesmeleri yetkilendir'* (Enable Interrupt) (EI) komutu gelene kadar, herhangi bir kesmenin işlemcinin çalışmasını etkilemesini önleyecektir.

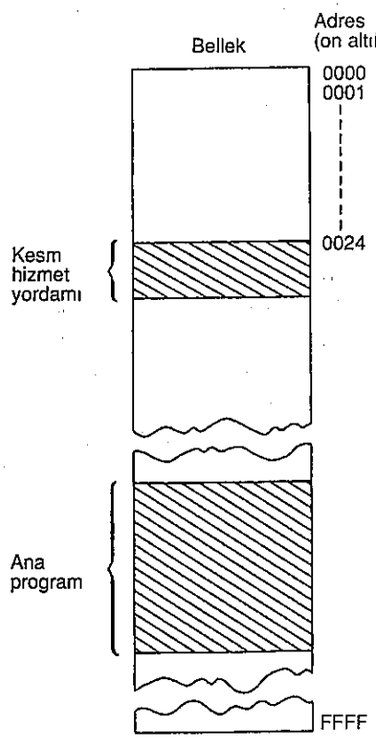
Tüm kesmeleri bu yolla etkisiz hale getirmek riskli olabilir. Sonuçta, daha önce açıklandığı gibi, kesmelerin gördüğü işlev, ilgilenilmesi gereken dışsal bir durumu işlemciye bildirmektir. Kesme sinyallerinin işlemciye erişmesinin önlenmesi, işlemcinin dış dünyaya karşı duyarlılığını yitirmesine neden olur.

Bu nedenlerden dolayı, uygulamada, kesme hizmet yordamının mümkün olduğu kadar kısa tutulması uygun olacaktır. Bu tür bir programda, özellikle kesmelerin yetkisizlendiği bölüm, mümkün olduğu kadar kısa tutulmalıdır.

Böylece, özel olarak, kesmeler içeren bir mikrobilgisayar sisteminin ele alınmasında, göz önünde bulundurulması gereken çeşitli noktalar vardır:

1 Kesmelerin kullanımı bir zaman sorunu doğurur. Bu, bir kesme geldiğinde, yordamlar arasında geçiş yapma ihtiyacından doğmaktadır.
2 Herhangi bir kesmenin geliş zamanı, genellikle önceden kestirilemez.
3 Bazı kesmeler, kritik bir zaman aralığında işlerini tamamlamak zorundadırlar.
4 Gerçek zamanlı bir sistemde, toplam mevcut zamanın, yapılması gereken tüm hesaplamalara izin verecek uzunlukta olmasına dikkat edilmelidir. Kesme yönetiminden kaynaklanan fazladan süreler de bu hesaplamalara katılmalıdır.
5 Kesmelerin yetkisizlendiği zaman aralığı minimumda tutulmalıdır.

S 8.6 6 Kesme hizmet yordamları mümkün olduğu kadar kısa tutulmalıdır.



Şekil 8.7

8.8 Gerçek kesme olanaklarına örnekler

Pratik mikrobilgisayar sistemlerinde sunulan kesme olanakları iki aşamada düşünülebilir. Birincisi, işlemcinin kendisinin sunduğu olanaklardır. İkinci olarak da, işlemciye bağlı çevresel yongalarca sunulan olanaklar vardır. Intel 8255A programlanabilir çevresel arabirimi (Bkz: Bölüm 2) bu türde çevresel yongalara bir örnektir.

İşlemci kesme olanakları

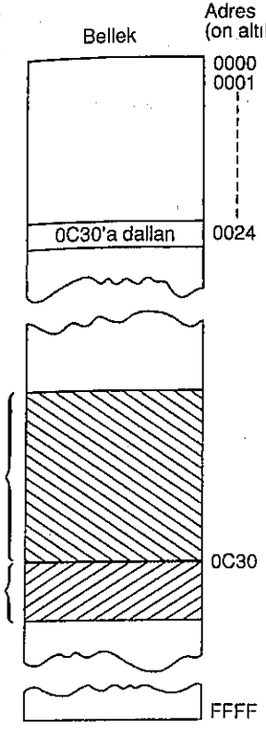
Bu noktada, bir parça geriye gitmek ve bir mikroişlemcide ne tür bir pratik kesme olanağına gerek duyulacağını incelemek uygun olacaktır.

En basit olanak, işlemci yongası üzerinde, çevrebirimlerinden gelecek bir kesme istek sinyalinin giriş yapabileceği bir giriş bacağına yer verilmesidir. Bu sinyal alındığında, işlemci:

1 Program sayıcısının içeriklerini saklar (genellikle sistem yığınında).
2 Kesme hizmet yordamına dallanır. Bu minimum sistemde yalnızca bir kesme mevcut bulunduğundan, bellekte sabit bir konumda olabilecek yalnızca tek bir hizmet yordamına gerek duyulacaktır (ya da en azından, programın ilk komutu sabit bir adreste olabilir).
3 Kesme hizmet yordamını çalıştırır. Daha önce açıklandığı gibi bu, işlemci kaydedici içeriklerini saklama ve tekrar geriye alma komutları içerebilir.
4 Kesme hizmet yordamının sonunda ana programa döner.

Program belleğinin bu düzenlemeye ilişkin organizasyonu Şekil 8.7'deki gibi gösterilebilir. Bu şekilde, kesme hizmet yordamının (gelişigüzel olarak) gösterilen sabit RAM konumlarında olduğu varsayılmıştır. Yani kesme hizmet yordamı, 0024 (on altılı) adresinden başlamaktadır. Ana (kesilen) program, RAM'de daha yüksek adreslere yerleştirilmiştir.

Kesme hizmet yordamının daima (varsaydığımız) 0024 adresinden başladığı için, işlemcinin kesmeye ilişkin olarak yapacağı işlem basittir:



Şekil 8.8

1 Program sayıcısının içeriklerini saklar.
2 0024 adresine dallanır.
3 Yukarıdaki gibi.
4 Yukarıdaki gibi.

Örneğimizde yalnızca tek bir kesme hizmet yordamı bulunduğundan, bir önceki kesme ile ilgili işlem yapılıyorken, bir başka kesmenin kabul edilmeyeceği noktası önem taşır. Bu nedenle, işlemcinin işlem dizisine bir diğer adım daha eklenmelidir. Herhangi bir kesme geldiğinde, kesme girişinin yetkisizlenmesi gerekir. Bu, otomatik olarak işlemci donanımı tarafından gerçekleştirilir.

Böylece, bir kesmenin kabulündeki işlem adımları şu şekilde yeniden yazılabilir:

1 Kesme girişini yetkisizlemek.
2 Program sayıcısı içeriklerini saklamak.
3 0024 adresine dallanmak.
4 Daha önce olduğu gibi.
5 Daha önce olduğu gibi.

Eğer bu sistem kullanılırsa, kesme hizmet yordamı kesilemez. Bununla birlikte, kesme girişi, kesme hizmet yordamı sona erdikten ve kesilen programa dönüş yapıldıktan hemen sonra, mümkün olduğunca ivedilikle tekrar yetkilenmelidir.

Kesme için, her zaman sabit bir adrese, bu örnekte 0024 (on altılı) adresine dallanmak uygunken, kesme hizmet programını daima bu konuma yerleştirmek zorunda kalmak uygun değildir. Sabit bir konuma bir dallanma komutu yerleştirmek suretiyle bu sorunun üstesinden, kolaylıkla gelinebilir. Bu komut, programın, RAM'de herhangi bir konumdaki kesme hizmet yordamına dallanmasına neden olur.

Şekil 8.8'de bu nokta gösterilmiştir. Kesme sinyali, daha önce olduğu gibi, 0024'e dallanmaya neden olur. Bu bayt (ya da

dallanma komutu 3-baytlık bir komut olduğu için, bu bayt ve ardından gel
bayt) kesme hizmet yordamına bir dallanma komutu içerir. Örneğin, bu pro
ana programın ardından gelen, 0C30 vd. adreslerine yerleştirilmiştir; ancak,
önce de ifade edildiği gibi, pratikte, uygun herhangi bir yerde olabilir.

Intel 8085 kesme olanakları Tek bir kesme girişi, yukarıda izah edildiği
kullanışlıdır, ancak birkaç farklı kesmenin mümkün olabilmesi, pratikte çok d
fazla esneklik getirir.

Intel 8085 mikroişlemci, işlemci yongasında beş kesme girişi bacağına sahi
Bunlar RST 5.5, RST 6.5, RST 7.5, TRAP ve INTR olarak adlandırılmı
Çevrebirimlerinden gelen kesme sinyalleri, bu pinlerden herhangi birine gönd
lebilir. Bu tür bir kesme, işlemci tarafından kabul edildiğinde, bir çıkış bac
$\overline{\text{INTA}}$'ye, bir kesmenin alındığını bildirmek üzere bir alçak düzey sinyali gönde

Bu beş kesme girişinden herhangi biri, işlemci tarafından, herhangi bir an
etkinleştirilebileceğinden, işlemci, bunlar için bir öncelik sistemine ihtiyaç duy
caktır (8.3 ve 8.5 Kısımlar). Bu aşağıda gösterilmektedir:

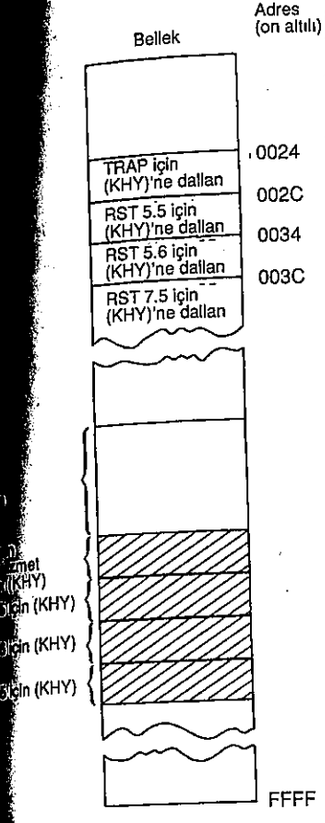
Kesme adı	*Öncelik sırası*
TRAP	1
RST 7.5	2
RST 6.5	3
RST 5.5	4
INTR	5

Böylece, TRAP Kesmesi, RST 7.5'dan, RST 7.5 da RST 6.5'dan daha yüksek
önceliğe sahiptir.

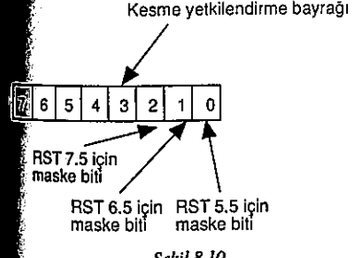
Bu kesmelerin ilk dördü, işlemcinin bellekte ayrı ayrı adreslere dallanmasına neden
olur. Böylece:

TRAP, 0024 (on altılı) adresine dallanmaya,
RST 7.5, 003C (on altılı) adresine dallanmaya,
RST 6.5, 0034 (on altılı) adresine dallanmaya,
RST 5.5, 002C (on altılı) adresine dallanmaya neden olur.

Bu konumlar, bazen *ayrılmış* özel konumlar olarak adlandırılır, bazen de kesmele
rin *vektör adresleri* olarak bilinirler. Benzer biçimde, her bir kesme için ayrı



Şekil 8.9



Şekil 8.10

ayrı pinlerin bulunduğu ve özel bir pindeki bir sinyalin sabit bir adrese dallanmaya neden olduğu bu tip bir sistem, bir *vektörlendirilmiş* kesme sistemi olarak bilinir.

Bu sistemi kullanabilmek için, program yürütmenin uygun kesme hizmet yordamına (ISR) aktarılmasını sağlayacak biçimde, dallanma komutlarının, kesmelerin vektör adreslerine yerleştirmesi gerekmektedir. Bu, Şekil 8.9'da gösterilmiştir.

Beşinci kesme girişi INTR, biraz farklıdır. Eğer bu kesme kullanılırsa, çevrebiriminin, işlemciye bir program komutu da iletmesi gerekecektir. Kesme isteği bildirildiğinde, çevrebiriminin bu komutu veri yoluna çıkarması gerekir. Bu, işlemcinin kesmenin ardından yürüteceği bir sonraki komut olarak kullanılır. Bu nedenle, INTR'nin kullanımı daha karmaşıktır, çünkü çevrebiriminin fazladan bilgi sağlaması gerekmektedir; ancak, INTR işlev açısından çok yönlü bir nitelik gösterir.

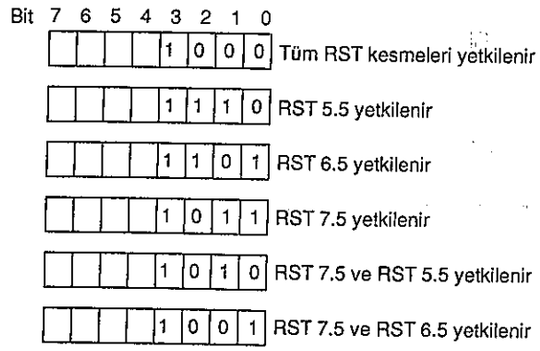
Kesmelerin öncelik sırası gösterildiği gibidir. Bu sıra, birkaç kesme birden alındığında, yalnızca kesme isteklerinin kabul ediliş biçimini etkiler. Hizmet yordamının çalışma biçimini değiştirmez.

TRAP kesmesi, en yüksek önceliğe sahiptir ve yetkisizlenemez. Bu nedenle TRAP kesmesi ortaya çıktığında hemen kabul edilir. Bu özelliği nedeniyle, işlemciyi kritik durumlardan (güç kaynağı arızası, alarmlar vb.) haberdar etmek için en uygun olanıdır.

Farklı önceliklerinin olması dışında, üç RST kesmesi benzer biçimde çalışır:

1 Teker teker *maskelenebilirler.*
2 'Kesmeleri yetkisizle' (DI) komutu ile tümü yetkisizlenir.
3 (TRAP ve INTR dahil olmak üzere) herhangi bir kesme, işlemci tarafından kabul edildiğinde, tümü yetkisizlenmiş olur.

Her bir RST kesmesi, birleştirilmiş bir *maske* biti ile denetlenir. Bu üç bit ve tüm kesme kilitleme biti, işlemcideki *kesme-*



Şekil 8.11

maske kaydedicisinde tutulur (Bkz: Şekil 8.10).

Herhangi bir kesme için maske biti 0 olduğunda, (kesmeyi yetkileme bayrağı I yapıldığı sürece) o kesme yetkilenir, maske biti 1 ise, söz konusu kesme yetkisizlenir. Şekil 8.11'de bazı desenler gösterilmektedir.

Maske bitleri, özel bir komut kullanılarak ayarlanabilir ya da sıfırlanabilir:

```
SIM  (Kesme maskesini ayarla).
```

Bu komut, A kaydedicisinin içeriklerini maske kaydedicisine aktarır. Böylece, tüm RST kesmelerini yetkilendirmek için

```
MVI     A,08
SIM
```

komut dizisi kullanılabilir. Bu, en alttaki dört kesme maske konumunu 1000'e getirir.
RST 5.5 kesmesini yetkilendirmek için komut dizisi:

```
MVI     A,0E
SIM
```

RST 6.5 kesmesini yetkilendirmek için:

```
MVI     A,0D
SIM
```


RST 7.5 kesmesini yetkilendirmek için:

```
MVI     A,0B
SIM
```

ve bu şekilde sürer.

Maske bitleri, özel bir komut kullanılarak incelenebilir:

```
RIM  (kesme maskesini oku)
```

Bu komut, kesme-maske kaydedicisinin içeriklerini A kaydedicisine aktarır.

Bu özellikler kullanılarak, çeşitli kesmelere verilen sistem yanıtı esnek bir biçimde denetlenebilir.

Daha önce gördüğümüz gibi, beş kesme girişinin herhangi birisinden gelen bir kesmenin kabul edilmesinin ardından, tüm RST kesmeleri otomatik olarak yetkisizlenir ve bir EI komutu yürütülene kadar, bu durumlarında kalırlar. Bu yapıldığında, RST kesmeleri, maske bitlerinin aldıkları değerlere göre yetkilenir.

Böylece, bir kesme hizmet yordamındaki uygun maske bitlerini ayarlamak ya da sıfırlamak amacıyla doğru komutları yerleştirmek ve ardından bir EI komutu koymak suretiyle, programcı, hangi kesmenin o hizmet yordamını kesebileceğini denetler. Eğer hizmet yordamının, kesintiye uğramaksızın yürütümünü tamamlamasını isterse, örneğin, bu yürütme sona erene kadar bir EI komutu yerleştirmez. Bununla bağlantılı olarak, TRAP kesmesinin yetkisizlenemeyeceğini hatırlamakta yarar vardır.

S 8.7
8.10

Programcı isterse, herhangi bir kesme hizmet yordamının başka bir kesme ile kesilmesine izin verir. Bu, hizmet yordamına giriş yapılmasına neden olan kesmeden daha alçak ya da yüksek öncelikli bir kesme olabilir.

Zilog Z-80 kesme olanakları Zilog Z-80 işlemcisinin iki tane kesme giriş bacağı vardır. Bu nedenle, iki ayrı kesme olabilir. Bunlardan biri, (TRAP gibi) programcı tarafından maskelenemez ve bu nedenle her ne zaman ortaya çıkarsa çıksın kabul edilir. Diğeri ise bir 'kesmeleri yetkisizle' komutu (daha önce verilen DI) kullanılarak maskelenebilir.

Programın denetimi altında, kurulabilen ya da sıfırlanabilen tek bir kesme mas biti vardır.

Maskelenemez kesme kabul edildiğinde, kullanılan vektör adresi sabit olup 00 (on altılı) değerindedir. Maskelenebilen kesme kabul edildiğinde de, kullanı vektör adresi:

1 0038'de (on altılı) sabit, ya da
2 Kesmeye neden olan çevresel birim tarafından verilmiş olabilir.

S 8.5, 8.8, 8.9 Bu olasılıklardan hangisinin kullanılacağı, kesme yanıtı *moduna* bağlıdır. Bu ikincisi, programın denetimi altında değiştirilebilir.

Çevresel yonga olanakları

Intel 8255A programlanabilir çevresel arabirim olanakları, Kısım 2.4'de özetlenmişti. Orada açıklandığı gibi PPI, mod 0, mod 1 ve mod 2 olarak bilinen üç *moddan* herhangi birinde çalışabilir:

Mod 0, temel G/Ç modudur.
Mod 1, 'onay' G/Ç modudur.
Mod 2, 'onay', iki yol modudur.

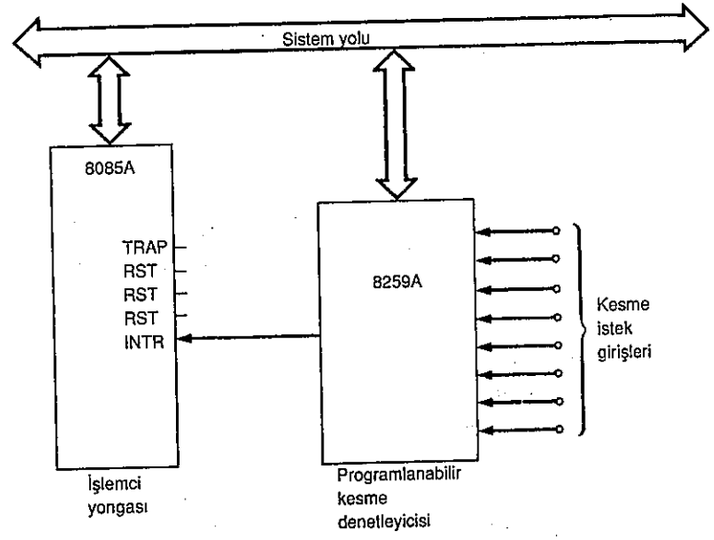
Bu modların herhangi birinde, PPI; A, B ve C portlarını kullanarak, çevresel birimlerle veri değiş-tokuşu yapar. Mod 0'da, tüm portlar veri aktarımı için kullanılır. Mod 1 ve 2'de, C portu bazı denetim işlevlerini yerine getirirken, A ve B portu da veri aktarımı için kullanılır.

1. ve 2. modlarda, C portundaki kaydediciler, çevresel birimlerle eşzamanlamayı sağlamak amacıyla onay sinyalleri üretmek ve kesme isteklerini ve kesme maskeleme bitlerini tutmak için kullanılırlar. Bu nedenle, örneğin mod 1'de, C portu kaydedicisi, iki kesme maske bitini tutar ve iki kesme sinyali üretir. Bu sinyaller, işlemci kesme girişlerine gönderilebilir.

Maske bitleri, C portuna normal yazma işlemleri (OUT komutları) ile kurulabilir veya sıfırlanabilir; kesme ve maske bitlerinin durumu da, C portu kaydedicisini okumak için bir IN komutu kullanılarak okunabilir.

Programlanabilir Kesme Denetleyicileri

Programlanabilir Kesme Denetleyicileri olarak bilinen özel devreler kullanılarak, işlemci yongasındaki kesme olanakları, daha fazla girişe olanak tanımak üzere genişletilebilir. Bu tür birimlere örnek olarak Intel 8259A PIC verilebilir. Bu devre, üzerine çevresel birimlerden gelen kesme isteklerinin konulabileceği, 8 giriş hattı içerir. İşlemcinin kesme girişine, sözgelimi INTR girişine bağlı bir çıkışı vardır. Bu durum, Şekil 8.12'deki gibi gösterilebilir.



Şekil 8.12

Pratikte, birçok 8259A yongası 64 istek giriş hattı oluşturmak için birbirine bağlanabilir.

Birim, yerleşik öncelik-belirleme devreleri kullanarak, istekleri öncelik sırasına göre girişlerine yerleştirir ve en yüksek öncelikli olanı işlemciye iletir.

Devre, ayrıca, kesmeyi tanımlamak için işlemci veri yoluna gönderilmesi gereken bilgiyi de üretmektedir. Bu bilginin, kesme isteğinin ardından gelen ve işlemci tarafından, bir sonraki komut olarak yürütülen bir komuttan ibaret olduğu anımsanacaktır. Yani, işlemci, INTR'deki bir kesme isteğinin ardından yürütülecek komutu, bellek yerine veri yolundan (ve dolayısıyla çevre biriminden) alır.

8259A, bu bilgi için bir CALL komutu üretir. Bu komuttaki adres, hizmet edilen kesmeye uygun hizmet yordamının adresidir. Bu nedenle, bu CALL komutunun yürütülmesi, program sayıcısının içeriklerinin saklanmasına ve uygun yordama girilmesine neden olur.

S 8.11, 8.12 8259A, kesmeyi maskeleme, kesme durumunu saklama vb. diğer olanaklar da içerir. Bunlar bu aşamada ayrıntı niteliğindedir.

Sorular

8.1 Bir mikrobilgisayar sisteminde 'kesme' sözcüğü ile ne anlatılmak istenmektedir? İşlemci tarafından bir kesme alındığında meydana gelen işlemleri anlatın ve her birinin nedenlerini açıklayın.

8.2 Mikrobilgisayar sistemlerindeki kesme özelliklerinin önem taşımasının nedenlerini kısaca özetleyin. Cevabınızı, işlemcinin, dış birimlerden gelen isteklere yanıt verme biçimini tartışarak ve işlemcinin yavaş çalışan çevrebirimleri ile nasıl iletişim kurduğunu açıklayarak gösterin.

8.3 Mikrobilgisayarın, bir kesme isteği ve bir altyordam çağrısına verdiği yanıtların bir karşılaştırmasını yapın. Bunlar arasındaki benzerlik ve farklılıkları belirtin.

8.4 Kesme hizmeti süresince, işlemci kaydedici içeriklerinin niçin saklanması gerektiğini açıklayın. Pratik mikrobilgisayar sistemlerinde bunun nasıl yapıldığını anlatın.

8.5 Bir mikroişlemci ile çevresel birim arasındaki bilgi aktarımı için kesmelerin nasıl kullanılacağını ayrıntılı olarak gösterin. Bunu yaparken, RST 5.5 kesme hattının kullanılacağını, kesme hizmet yordamının 00BO (on altılı) adresinden başladığını, yığın işaretçisinin 0100 içerdiğini ve ana programın 01FF adresinden başladığını varsayın. Özellikle, program yürütmesinde doğru denetim akışını ve kesme girişiminin doğru yönetilebilmesini sağlamak üzere, kesme hizmet yordamında ve ana programda bulunması gereken komutları gösterin.

8.6 Kesmelerin kullanımı nedeniyle, bir mikrobilgisayar sisteminde ne gibi problemler ortaya çıkabilir? Örnekler vererek açıklayın.

8.7 Bir mikrobilgisayar sisteminde, niçin birden fazla kesme giriş hattına gerek duyabileceğini açıklayın. Tek bir kesme girişinin olması durumuyla karşılaştırıldığında, çoklu bir kesme sisteminde bulunması gereken ek olanakların neler olduğunu açıklayın. Bu olanakların gerekli olmasının nedenleri nelerdir?

8.8 Pratik bir mikrobilgisayar sistemi, kesme özelliklerini denetlemesine olanak tanıyan hangi komutlara sahiptir? Kesme maskeleri ve özel komutlar EI ve DI'ın kullanımlarını açıklayın.

8.9 Bir kimyasal işleme tesisi, bir Intel 8085 mikrobilgisayarı tarafından denetlenecektir. Tesis, 8085'e ilişkin dört kesme hattı, yani TRAP, RST 7.5, RST 6.5 ve RST 5.5 kullanacaktır. Kesmelerin:

(a) RST 7.5 kesmesinin, RST 6.5 ve RST 5.5 kesme hizmet yordamını kesebilmesi,

(b) RST 6.5 kesmesinin, RST 7.5'inkini değil, RST 5.5'nin hizmet yordamını kesebilmesi,

(c) RST 5.5 kesmesinin, RST 6.5'inkini değil, RST 7.5'nin hizmet yordamını kesebilmesi,

(d) TRAF'ın diğer herhangi bir kesme hizmet yordamını kesebilmesi, biçiminde düzenlenmesi istenmektedir.

TRAP'a ait hizmet yordamı, diğerleri tarafından kesilemez. Her bir hizmet yordamına ve ana programa yerleştirilmesi gereken kesmeleri denetlemek için gerekli komutları sıralayarak, bunun nasıl yapılabileceğini açıklayın.

8.10 Vektörlenmiş sistem kavramını ve bunun nasıl çalıştığını açıklayın. Soruyu, pratik bir mikrobilgisayar üzerinde örnekler vererek yanıtlayın.

8.11 Bir mikroişlemcinin kesme yeteneğini genişletmek için, özel çevresel kesme-yönetim yongaları nasıl kullanılabilir? Üzerinde veri mevcutsa, Intel 8259A programlanabilir kesme denetleyici yongasını bir örnek olarak kullanın.

8.12 Bir Intel 8085 işlemcisi kullanan bir mikrobilgisayar sistemi, üç kesme sinyaline bağlanacaktır. Bunlar:

(a) Sayaç/zamanlayıcı yongasından bir kesme.

(b) Güç kaynağı arızasını bildiren ve gerilim-algılama devresinden gelen bir kesme.

(c) Bir karakterin, işlemci tarafından alınmak üzere hazır olduğunu bildiren ve bir bant okuyucusundan gelen kesme.

Bu sinyaller için uygun bir öncelik sırası önererek, her birimin kullanım gereken 8085 işlemci girişlerini belirtin. Kesmelerin nasıl denetlenebileceğ gösterin ve bunun denetlenmesinden sorumlu üç kesme hizm programındaki komutları sıralayın.

Bölüm 9 Dış dünya ile arabağlantı kurulması ve eşzamanlama

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

1 *Bir çevresel aktarım işlemini başlatan birimin, normal olarak o aktarım için ana birim, kullanılan diğer birimlerin ise uydu birimler olduğunu değerlendirebilmeli,*
2 *Çevrebirim taramasının, ana birimle çevrebirimleri arasındaki bir iletişim yöntemi olduğunu anlayabilmeli,*
3 *Bir tarama sisteminin çalışma biçimini ve bu tür bir sistemde, taramanın avantaj ve dezavantajları ile birlikte, öncelik verilme biçimini anlayabilmeli,*
4 *Giriş/çıkışlara ilişkin altyordamları kullanabilmeli,*
5 *Bir çevresel arabirim devresi kullanarak işlemcinin:*
   (a) Işık yayan diyotlar (LED) gibi çeşitli iki-durumlu işaretçileri denetleyebildiğini ve,
   (b) İşaretçilerin, anahtarlama birimlerinin durumunu görüntülemesine izin verecek biçimde, iki durumlu işaretçileri çalıştıran bir çıkış altyordamı ile birlikte birçok iki-durumlu anahtarlama birimlerinin ayarlama değerlerini okuyan bir giriş alt programını kullanabilmeli,
6 *Çevresel aktarım işlemlerinde, yazılım ve donanım ihtiyacının birbirleriyle yakın ilişkili olduğunu değerlendirebilmeli,*
7 *Bir mikrobilgisayar sisteminde seri ve paralel veri aktarımı arasındaki farklılıkları ayırt edebilmeli,*
8 *Asenkron seri veri iletiminin ilkelerini anlayabilmeli,*
9 *(a) Dupleks iletim*
   (b) Yarı dupleks iletim
   (c) Simpleks (tek yönlü) iletimi açıklayabilmeli,
10 *Seri veri iletimi yapan bir çevresel birim ile iletişimdeki bir kaydırmalı kaydedicinin kullanımını ve bu uygulamada kaydırmalı kaydedicinin işleyişini anlayabilmeli,*
11 *Mikrobilgisayarlarla seri veri iletim hatları arasında arabağlantı kurulmasında kullanılmak üzere çeşitli LSI yongalarının mevcut olduğunu değerlendirebilmeli,*
12 *(a) Hassas bir aralık ölçümünün gerekli olduğu (çamaşır makinesi gibi) birimlerde ve*
   (b) İşlemlerin kesin zamanlarda gerçekleşmesinin gerektiği (örneğin, daldırma tipi ısıtıcısının günün belli zamanlarında açılıp, kapatılması gerektiği

bir ev gibi) sistemlerdeki uygulamalarda aralık zamanlayıcılarının bulunduğunu değerlendirebilmeli,

13. Mikrobilgisayarın yazılımı ve donanımının ve yukarıda 12. maddede verilen uygulamalarda olduğu gibi aralık zamanlayıcısının çalışmasını ve birbirleriyle olan etkileşimini anlayabilmelisiniz.

9.1 Giriş

Mikrobilgisayarın dış birimler ve sistemlerle nasıl iletişim kurduğu sorusu, bu kitabın bundan önceki bazı bölümlerinde ele alınmıştı. Örneğin, Kısım 1.4'de, işlemci ile ona bağlı çeşitli çevresel birimler arasındaki hız farklılıkları izah edilmiş; Kısım 1.5'de de, işlemci ile çevrebirimi arasındaki tek bir veri parçasının (örneğin bir baytlık verinin) aktarımı sırasında, bu ikisini eşzamanlamak için anlaşmanın nasıl kullanıldığı ana hatlarıyla belirtilmişti. Yine o bölümde, bu tür bir veri aktarımı yapıldığında, bunun için kullanılan iki tekniğin:

1 Çevrebirimlerinin işlemci tarafından taranması ve
2 Kesmeler

olduğundan söz edilmişti.

Kesmeler, 8. Bölümde ayrıntılı olarak anlatıldı. Bunda, çevrebiriminin *ana* birim olarak işlev gördüğü düşünülebilir.

Ana ve uydu birimler

Bir çevrebirimi aktarımında ana durumda bulunan birim, aktarımı başlatan birimdir. Bu nedenle, işlemciye gönderilmeye hazır bir veriye sahip olan bir çevrebirimi, bir kesme gönderir. Daha önce görüldüğü gibi, bu kesme, veri değiş-tokuşuna neden olan bir olaylar dizisini başlatır. Uygulamada değiş-tokuş, onay kullanılarak senkronize edilir.

Değiş-tokuşun ya da en azından değiş-tokuşu gerçekleştirecek olaylar dizisinin başladığı zaman, kesme sinyalinin ortaya çıkışıyla sabitlenir. Kesme ile sürülen böyle bir sistemde işlemci kesme aldığında, yalnızca G/Ç aktarımları yapar; bu nedenle, bu türde aktarımlar açısından *uydu* birim olmaktadır.

İşlemcinin her zaman *ana* olduğu diğer bir veri aktarımı yöntemi ise, tarama tekniği kullanmaktır.

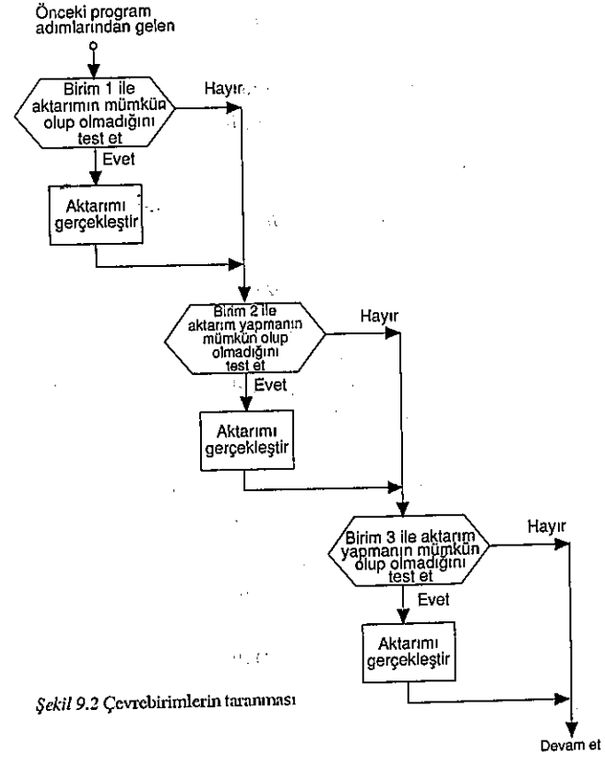
Şekil 9.1

9.2 Çevrebirimlerin taranması

8. Bölümde işlemcinin, bir çevresel birimin bir bilgi aktarımına hazır olup olmadığını, o aktarımı başlatmadan önce test etmesi gerektiği anlatılmıştı. Açıklamalar, özel bir birimin - bir kağıt bant okuyucusunun - çalışma şekline göre verilmişti. Ancak bu örnek doğal olarak, çok çeşitli çevrebirimlerine uygulanabilme özelliğine de sahipti.

Bu nedenle, okuyucu ile mikroişlemciyi senkronize etmenin (pek iyi olmayan) bir yolunu gösteren Şekil 8.2, Şekil 9.1'de gösterildiği gibi genelleştirilebilir.

Bilgisayar, çevresel birimle iletişim kurmaya ihtiyaç duyduğunda, ilk olarak iletişimin mümkün olup olmadığını, yani birimin veri alışverişine hazır olup olmadığını test eder. Eğer hazırsa, istenen değiş-tokuş yapılır ve program devam eder. Değilse, işlemci A noktasına geri döner ve birimi tekrar test eder.



Şekil 9.2 Çevrebirimlerin taranması

Bu yöntemde, görülebileceği gibi, işlemci sürekli olarak "Çevresel birimin çalışmaya hazır olup olmadığını test et" komutunda döngü yapar. Bu döngü zorunlu değildir ve genellikle işlemcinin zamanının boşuna harcamasına yol açar.

İşlemcinin birden fazla çevrebirimini *taradığı* bir sistem, bunun genişletilmiş bir biçimidir. Bir önceki cümlede de belirtildiği gibi, tarama yalnızca, işlemci birden fazla çevrebirimi ile iletişim kurduğunda uygulanabilir. Şu da var ki, böyle bir durum olduğunda, tarama uygun bir çalışma modu oluşturabilir.

Tarama sistemleri, Şekil 9.2'de gösterilen biçimde çalışır. İşlemci ile iletişim kurabilecek herhangi bir çevrebirimi, yani gönderecek bir bilgisi olan, bilgiye ihtiyacı olan ya da başka bir nedenden ötürü dikkat çekmek durumunda olan herhangi bir çevrebirimi, daha önceki bölümlerde anlatıldığı gibi durum bayrağını (bir bitlik bir durum işaretçisini) ayarlar.

Mikrobilgisayar sisteminde çalıştırılan program, çevrebirimi bayraklarının durumunu sıra ile birbiri ardı sıra test etmekten sorumludur. Bu nedenle, akış diyagramının en üstündeki 1. Birim ile aktarımın mümkün olup olmadığını test et komutunu çalıştırır (Şekil 9.2).

1. birimi kullanarak bir aktarım yapmak o an için mümkün değilse, yani test 'Hayır' çıkışına dallanıyorsa, 2. birim aynı yoldan test edilir. Bu testin sonucu da olumsuzsa, 3. birim test edilir ve test bu şekilde sürer. Böylece işlemci, çeşitli birimler etrafında, bu birimlerin istekte bulunup bulunmadığını inceleyerek döngü yapar.

Herhangi bir birim, test edildiğinde veri alışverişine hazır ise, işlemci, akış diyagramındaki test kutucuğunun 'Evet' çıkışından çıkar ve istenen aktarım gerçekleştirilir.

Görülebileceği gibi, bu iletişim yönteminde gerçekleştirilen tüm işlemler, mikroişlemcinin denetimi altında ve mikroişlemci tarafından başlatılır. Bu konuda çevrebirimleri tümüyle edilgen durumdadırlar ve tarama çevrimi kendilerine ulaşana kadar beklemede kalırlar. Bu nedenle, herhangi bir aktarımda işlemci ana ve birimler de uydu durumdadırlar.

Bunun, çevrebirimlerinin işlemciyi, kendileriyle olan veri değiş-tokuşunu denetleyen programa girmeye 'zorladığı' kesme sistemiyle bir zıtlık sergilediğini tekrar vurgulamak gerekir.

S 9.1

Taramanın Avantajları

Bilgisayarın ana görevinin, çevrebirimleri ile iletişim kurmak olduğu ve hesaplamaların yapılabileceği hızın, dışsal sisteme ayak uydurabilecek yeterlilikte olduğu uygulamalarda, tarama çok uygun bir veri değiş-tokuş yöntemi sağlar. Şu avantajlara sahiptir:

1 Tarama, tümüyle işlemcinin denetimi altında olduğundan, kesme ile sürülen bir sistemde olduğu gibi, verileri veya kaydedici içeriklerini saklamaya gerek yoktur. Bu nedenle, bu işlemlerde ortaya çıkan ek yüklerden büyük ölçüde kaçınılmış olur (Kısım 8.7).
2 Kolayca uygulanabilen bir yöntemdir.
3 Herhangi özel bir uygulamada, tarama işlemince harcanan zamanı tam olarak hesaplamak mümkündür. Bu nedenle, sistem, bağlı olduğu dışsal sistem tarafından getirilen sınırlamalara uyabilecek biçimde ayarlanabilir.

Taramanın Dezavantajları

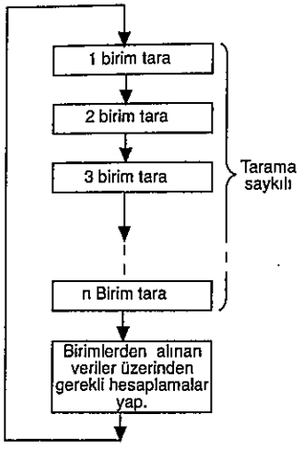
Kuşkusuz, bir veri aktarımı yöntemi olarak kullanılan taramanın beraberinde getirdiği bazı dezavantajlar vardır:

1 Özellikle, birimlerin test edilme hızı, aktarımın gerçekleştirilme hızından çok daha yüksek ise, işlemcinin zamanının bir bölümü boşa harcanmış olur.
2 Birimler, tarama çevrimi onlara ulaşana kadar, bir uyarıyı beklemek zorunda olduklarından, bunu gerçekleştirebilecek bir mantık devresine ihtiyaç duyarlar. Örneğin; bir birim, mikrobilgisayara okunacak veriler üretiyorsa, işlemci verileri almaya karar verene kadar, verileri bu biçimiyle saklamalıdır. Diğer bir deyişle, böyle bir tarama sistemindeki çevrebirimleri, bir kesme sisteminde yapabildikleri gibi, doğrudan işlemciden uyarı isteyemezler.

Tarama sistemlerindeki öncelikler

Bir tarama sisteminde, bir çevresel birimin öncelik sırası, işlemci taramasının düzenlendiği sıra ile belirlenir. Şekil 9.3'de, işlemcinin, çevresel birimleri taradığı, ardından sistemin gerek duyduğu çeşitli hesaplamaları yaptığı, daha sonra tekrar çevresel birimleri taradığı ve işlemleri bu şekilde sürdürdüğü tipik bir sistem yazılımının akış diyagramı gösterilmiştir.

Verilen bu şekilde, '1. birimi tara', '2. birimi tara' gibi ku
cukların, birimlerin test edilmesine ve eğer olabiliyorsa, v
değiş-tokuşunun gerçekleştirilmesine ilişkin işlem adımla
içerdiği varsayılmıştır.

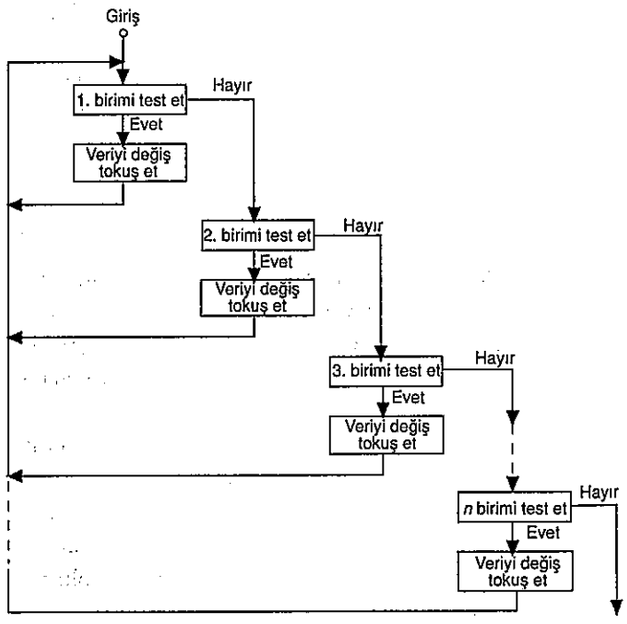


Şekil 9.3

Gösterilen şema, 'sonsuz şerit' tipi bir taramadır. Bir birim
öncelik sırası tarama listesindeki konumudur. Bu nedenle, 1
birim en yüksek önceliğe, 2. birim ikinci önceliğe, vb. sahip
Bu şemayı değişik biçimlerde hazırlamak mümkündür. Örn
ğin, verilerin bir birimle değiş-tokuş edilmesi durumund
tarama saykılının en başına dönmek mümkündür. Bu, Şek
9.4'de gösterilmiştir. 1. birimin testinin sonucu olumsuzsa
tarama saykılı 2. birime geçer. Bununla birlikte, 1. birimin tes
başarılı ise, veri değiş-tokuşu yapılır, tarama saykılı tekra
baştan başlar ve 1. birim tekrar taranır. Bir birimin test sonu-
cunun her olumlu çıkması halinde bu gerçekleşir.

S 9.2, 9.3

Bu yöntem, listenin başındaki birimlerin önceliklerini, alt
sıradakilere göre artırmak sonucunu getirir.



Şekil 9.4

9.3. Giriş/Çıkış altyordamları

Kısım 8.4'de, mikrobilgisayarın IN komutları ya da bellek-haritalı giriş kullanarak, iki durumlu anahtarlar gibi bazı basit giriş birim durumlarını nasıl inceleyebileceği anlatılmıştı.

Bir sistemde G/Ç işlevlerinin altyordamlar tarafından gerçekleştirilmesi çok yaygın bir uygulamadır. Bu nedenle, Kısım 4.8'de ana hatlarıyla belirtildiği gibi, çevresel iletişim ile ilgili program parçası ayrılarak bir altyordam biçiminde yazılır.

Bu kavram, özellikle pek çok G/Ç işlevine sahip büyük sistemlerde yararlı olmaktadır. Bu tür sistemlerde, programcının birleşik çevresel birimi kullanması gerektiğinde çağrılabilen tanımlanmış giriş ya da çıkış parametreli bir standart kütüphane altyordamı varsa, sistemdeki programlama uygulamalarının kolaylığı büyük ölçüde artmış olur.

Programcı, G/Ç işlemlerini gerçekleştirmek için, yalnızca gerekli parametreleri ayarlamaya ve G/Ç alt yordamını çağırmaya gerek duyar. Kullanılan çevresel arabirim devresinin çalışma modlarını ayarlamak gibi ayrıntılı bir işlemle veya sistemin çalışmasıyla ilgili öteki fonksiyonların herhangi birisiyle uğraşmasına gerek yoktur. Bunlar altyordamda yer almaktadır.

G/Ç alt yordamlarının kullanımına basit bir örnek, bir önceki iki-durumlu girişlerden değer okumaya ilişkin örnekten türetilebilir.

Altyordam kullanan bir G/Ç örneği

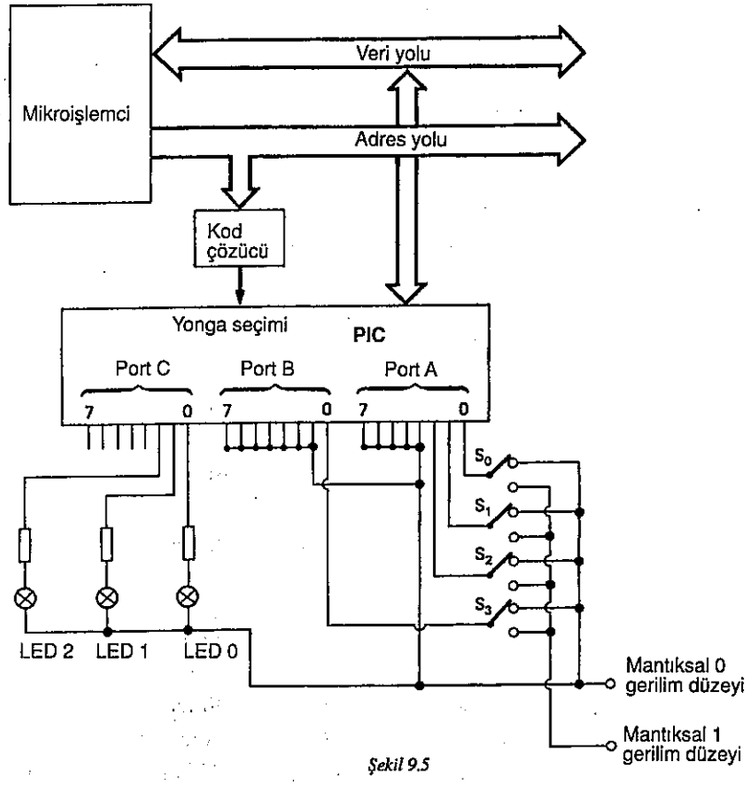
Mikrobilgisayar, anahtarlardan veri aldığı gibi, çeşitli birimlere veri de gönderebilir. Yalnızca iki-durumlu işlev çıkışlarının mantıksal 0 ve mantıksal 1 gibi gösterilmesi ya da çıkışın enerjilenip enerjilenmediğinin gösterilmesi gerektiğinde, basit lambalar ya da *ışık-yayan diyotlar* (LED'ler) kullanılabilir.

Işık-yayan diyot, üzerinden akım geçtiğinde herhangi bir renkte ışık veren bir diyottur. Üzerinden geçen akım, bir direnç yardımıyla uygun bir değere ayarlanabilir. Bu nedenle, bilgisayarın çıkış hatlarından biri, 0 volt (mantıksal 0) ile +5 volt (mantıksal 1) arasında değişirse, bir ışık-yayan diyot ile birlikte bir direnç çok ekonomik ve uygun bir hat durum göstergesi oluştururlar.

Işık-yayan diyot tabii ki bir bellek içermediğinden akım uygulandığında yanar ve

akım kesildiğinde söner. Bir diyotu 'yanan' durumda tutmak için, bir mandala (bir-bitlik bir bellek, bkz: Kısım 2.2 ve 3.6) bağlanması gerekir . Mandal, çıkış hattının son durumunu 'hatırında tutar' ve buna göre diyotun yanık ya da sönük kalmasını sağlar. Bölüm 2'de izah edildiği gibi, çevresel arabirim devrelerinin çıkış portları, çıkış verisini tutmak amacıyla mandal içerirler.

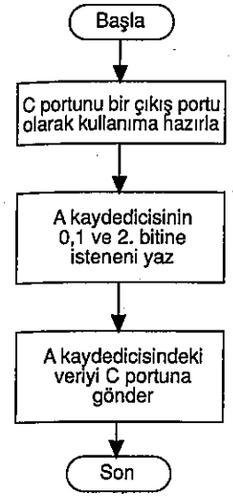
Böylece, bir mikrobilgisayar sistemi, Şekil 9.5'de gösterildiği gibi, belli sayıda anahtar ve ışık yayan diyota bağlanabilir. Bu, Şekil 4.17'nin geliştirilmiş bir biçimidir.



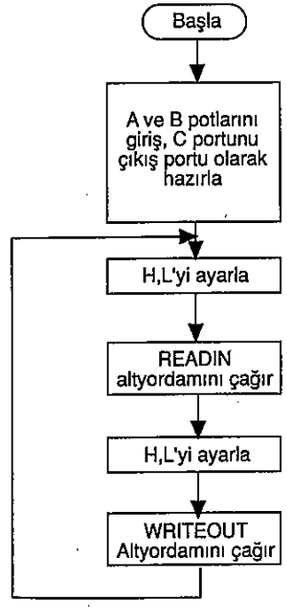
Şekil 9.5

Dört giriş anahtarı S_0, S_1, S_2 ve S_3, çevresel arabirim devresinin A ve B portlarına bağlanır. Üç ışık-yayan diyot, LED 0, LED 1 ve LED 2, C portuna bağlanır. Yukarıda belirtildiği gibi, C portunun çıkışında mandallar bulunduğu varsayılmıştır.

C portunun çıkışına değerler veren ve böylece LED'leri yakıp söndüren bir programın akış diyagramı Şekil 9.6'da gösterilmektedir.



Şekil 9.6



Şekil 9.7

S 9.4

Programa, C portunu çıkış olarak kullanıma hazırlamakla başlanmalıdır. Bu, Kısım 4.8'de anlatıldığı gibi:

```
MVI     A, 92
OUT     0B
```

komutları ile gerçekleştirilir.

Bu komutların ve daha sonra kullanılacak olanların, Bölüm 4'de kullanılmış olanlar gibi, çevresel arabirim devresi ve arabirim devresi portları için aynı adresleri kabul ettiklerine dikkat edin. Referans olmak üzere bunlar:

PIC	08 (on altılı)
Port A	08 (on altılı)
Port B	09 (on altılı)
Port C	0A (on altılı)

adresleridir.

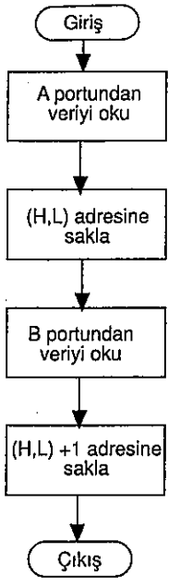
Yukarıda verilen komutlar, 10010010 ikili desenini, adresi 0B (on altılı) olan çevresel arabirim devresi denetim kaydedicisine gönderirler. Şekil 2.13'de gösterildiği gibi, bu kaydedicinin 0. ve 3. bitlerinin etkisi, C portunun işlevini düzenlemektir. Bu bitlerin her ikisi de 0 olduğunda, C portu bir çıkıştır. Aynı denetim sözcüğü, bundan başka, A ve B portlarının da işlevlerini belirler. Daha önce, bunların her ikisinin de giriş moduna ayarlandığı görülmüştü.

Verileri, C portundaki kaydediciye, ve dolayısıyla LED'lere iletmek için bir OUT komutu yürütüldüğünde, C portuna aktarılacak olan değer, A kaydedicisindeki değerdir. Bu nedenle, OUT komutu yürütülmeden önce, çıkış programının A kaydedicisinin 0, 1 ve 2 konumlarındaki bit desenini ayarlaması gerekir. Bu, Şekil 9.6'da gösterilen akış diyagramındaki 2. kutunun fonksiyonudur.

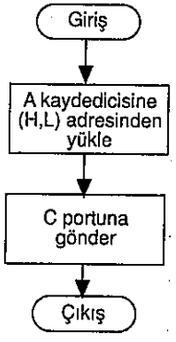
Son olarak:

```
OUT     0A
```

komutu, C portuna bir değer aktarır ve böylece üç LED'i istenilen biçimde yakıp söndürür.



Şekil 9.8 READIN alt yordamı



Şekil 9.9 WRITEOUT alt yordamı

Artık, Şekil 4.18 ve 9.6, sırasıyla giriş yordamı ile çıkış yordamı olarak birleştirilebilir. Bu örneğin amacına uygun olarak bunların her birinin bir alt yordam olarak yazıldığını varsayacağız. Bundan başka, çevresel arabirim devresinin portlarını kullanıma hazırlanma işinin, ilgili alt yordamı çağıran ana programda yapıldığını varsayacağız.

A portuna bağlanmış anahtarlardan veri okuyan ve bu veriyi C portuna bağlı LED'leri yakıp söndürmek için kullanan komple bir G/Ç programının blok diyagramı, Şekil 9.7'de gösterilmektedir.

Program, A ve B portunu giriş, C portunu da çıkış olarak kullanıma hazırlar. 7. Bölümde tanımlandığı gibi, programın birbiri ardı sıra yapılan alt yordam çağrıları için hazır olan bellekteki yığının konumunu ayarlamak amacıyla, yığın işaretçisini kullanıma hazırlar. Ardından, H, L kaydedici çiftine bir değer yerleştirir. Bu değer, A portundan okunan baytı tutmak için kullanılan bellek adresidir. B portundan okunan bayt ise, onu izleyen adrese yerleştirilir.

Kullanıma hazırlama işleminden sonra, program READIN altyordamını çağırır. Sonra, tekrar H,L'ye bir değer yerleştirir ve WRITEOUT altyordamını çağırır.

READIN altyordamının blok diyagramı Şekil 9.8'de gösterilmektedir. A portunda okunan değer, ana program tarafından H, L kaydedici çifti kullanılarak altyordama aktarılan bellek adresine yerleştirilir. B portundan okunan değer ise bellekteki bir sonraki adres konumuna yerleştirilir.

WRITEOUT altyordamının blok diyagramı Şekil 9.9'da gösterilmektedir. Bu alt yordamın işleyişi oldukça basittir. H, L'de tutulan adresteki bellekten bir baytlık veriyi alır ve PIC'in (Çevresel Arabirim Devresinin) C portuna gönderir. Baytın en alttaki üç biti, bu sayede LED'lerde görüntülenir.

Eğer READIN'e giriş yapılmadan önce H,L'e yüklenen adres ile WRITEOUT'a giriş yapılmadan önce doğrudan H,L'ye yerleştirilen adres aynı ise, üç anahtarın, S_0, S_1 ve S_2'nin

durumları LED'lerde görüntülenir.

WRITEOUT'a giriş yapılmadan önce, H,L'ye yerleştirilen adres, READIN'e girişten önce kullanılandan daha büyük ise, S_3 anahtarının durumu LED 0'da görüntülenir. Bu durumda LED 1 ve LED 2 sürekli sönük kalır.

Ana program sürekli çalıştırıldığından (Şekil 9.7), amaç (A portundan okunan verinin görüntüleneceğini varsayarak) LED'lerin anahtarlar tarafından denetlenebilmesine izin vermektir. Bu basit gibi görünen bir sonuçtur, ancak altında biraz karmaşıklık yatar.

Kolay anlaşılırlığı sağlamak açısından, diğer bir noktadan daha söz edilmesi gerekir. Şekil 4.22'de, verilen giriş programının orijinal akış diyagramında, anahtarların konumunun ayrı ayrı belirlenmesinin gerekli olduğu varsayılmıştır. Bu nedenle, akış diyagramının 3. fazında, anahtar değerlerinin *ayrı ayrı* saklanacağı ifade edilmiştir.

Az önce izah edilen, bir baytlık bir verinin giriş altyordamından doğrudan çıkış altyordamına aktarıldığı uygulamada, anahtarların durumlarını birbirinden ayırmaya gerek yoktur.

G/Ç Programı

G/Ç programının bir assembly dili versiyonu şöyledir:

```
BEGIN:  MVI   A,92        ;A,B,C portlarını kullanıma hazırla
        OUT   0B
        LXI   SP,10D4     ;Yığın işaretçisini kullanıma hazırla
START:  LXI   H, 1090     ;(H,L)'i 1090'a (onaltılı) ayarla
        CALL  READIN      ;READIN alt yordamını çağır
        LXI   H, 1090     ;(H,L)'i 1090'a (onaltılı) ayarla
        CALL  WRITEOUT    ;WRITEOUT alt yordamını çağır
        JMP   START       ;START'a geri dön
READIN: IN    08          ;A portunu oku
        MOV   M, A        ;Belleğe sakla
        IN    09          ;B portunu oku
        INX   H           ;(H,L)'e (+1) ekle
        MOV   M, A        ;Belleğe sakla
        RET               ;Geriye dön
```

READIN alt yordamı

```
            WRITEOUT:  MOV   A, M      ;Değeri A'ya yerleştir
WRITEOUT                OUT   0A       ;C portuna çık
altyordamı              RET            ;Geriye dön
```

Burada, yığın işaretçisinin 10D4 (on altılı) adresinde kullanıma hazırlandığı 1090 (on altılı) adresinin, READIN alt yordamı tarafından A portundan okun baytı, çıkış alt yordamı WRITEOUT'a ve dolayısıyla C portuna bağlı işaretçile iletmek için kullanıldığı varsayılmıştır. Bu adreslerin herhangi bir özelliği yokt yalnızca bu amaçlar için kullanmak üzere bellekteki uygun konumlardır.

Donanım ve Yazılım İhtiyacı

Yukarıda verilen örnek oldukça açıktır. Bununla birlikte, burada iki önemli nokta ortaya çıkmıştır. İlki, bu tür basit bir sistemde bile, donanım ve yazılımın birbiriyle yakın ilişkili olduğudur. Sistemde, gösterge ve anahtarlar için gerekli arabirim devresi ile bunlar arasındaki veri iletimi için gereken kullanıma hazırlama işlemine ve G/Ç programlarına gerek vardır. Bu, genel özelliktir; komple bir sistemde, daima yazılım ve donanıma ihtiyaç duyulacaktır.

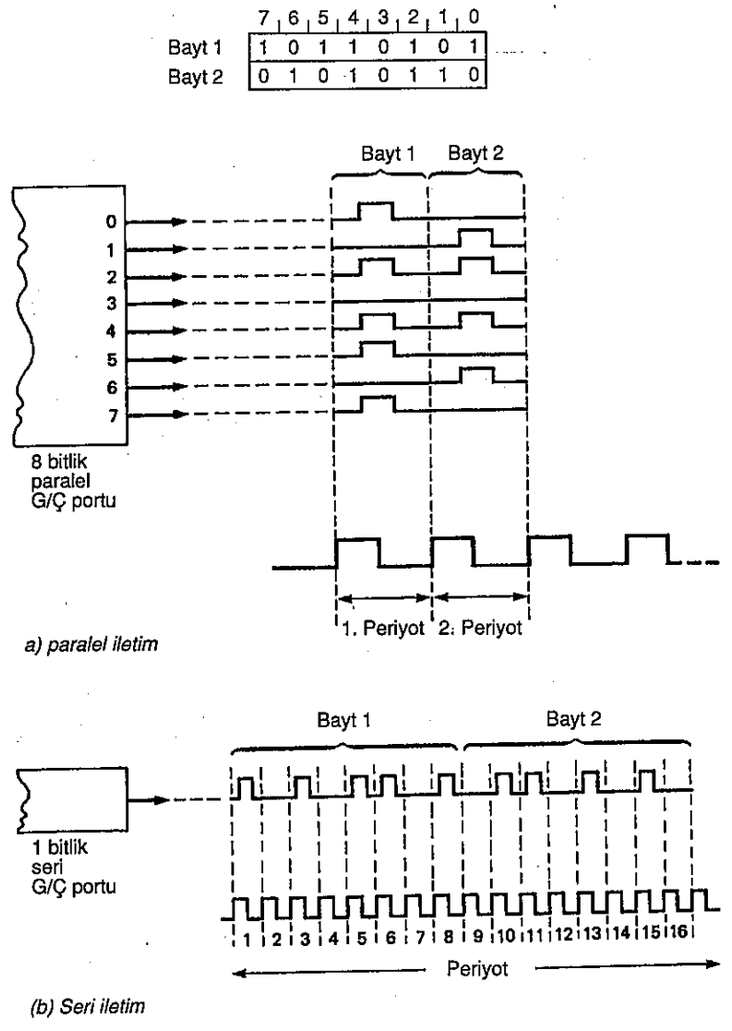
İkinci nokta ise, G/Ç için altyordam kullanımının, doğrudan bir program kullanmaktan daha fazla esneklik sağlamasıdır. Bu örnekte, elde edilen esneklik, S_0, S_1 ve S_2 veya S_3 anahtarlarının, ana programda çok basit bir düzenlenme yapılarak görüntülemeye olanak tanımasıdır. Genelde, alt yordama girmeden önce, istenen baytın adresi H,L'ye yerleştirilmek suretiyle, gösterge alt yordamı WRITEOUT bellekteki herhangi bir baytın en küçük değerlikli üç bitinin durumunu göstermede kullanılabilir.

Yalnızca en küçük değerlikli üç bitin görüntülenebileceği şeklindeki sınırlama, kuşkusuz bir yazılım sınırlaması değildir. LED'ler C portunun tüm bitlerine bağlanırsa, bellekteki herhangi bir baytın tüm bitleri görüntülenebilir. Yine, burada da istenen sonucu elde etmek için yazılım ve donanımın ortak kullanımına ihtiyaç duyulmaktadır.

9.4 Seri giriş/çıkış

1. Bölüm'de, paralel ve seri G/Ç arasındaki farklılıklara ilişkin kısa bir özetleme yapılmıştı. Bu bölümde söz edildiği gibi, belli bir zaman biriminde 1 bitlik bilgi girişi ya da çıkışı yapılan seri G/Ç, mikrobilgisayarın, veri değiş-tokuşu yaptığı birime bağlanmasında gereken bağlantı hat sayısını minimuma indirir.

Seri ve paralel veri aktarımının karşılaştırılması, Şekil 9.10'da verilmiştir. Şeklin (a) bölümü bilgisayardan, daha önce ayrıntılı bir şekilde anlatılan tipteki 8-bitlik paralel bir G/Ç portu aracılığıyla gönderilmekte olan iki baytlık bir veriyi gösterir. Bunların iletimi iki zaman periyodu sürer. 1. periyotta, ilk bayt (bayt 1) ve 2. periyotta da bayt 2 gönderilir. 8 port-çıkış hatları üzerindeki darbeler, gösterildiği gibidir. Darbelerin alt kısmında verilen zamanlama sinyalleri referans içindir. Şeklin (b) bölümü, bu iki baytın bir seri porttan tek bir çıkış hattı boyunca gönderilmesi gösterilmiştir. Bunların iletimi 16 periyot sürer. Bu periyotlar, 16



Şekil 9.10

tanesinin aynı sayfaya sığabilmesi için sıkıştırılarak çizilmiştir.

0 periyodunda, bayt 1'in 0. biti gönderilir.
1 periyodunda, bayt 1'in 1. biti gönderilir
2 periyodunda, bayt 1'in 2. biti gönderilir.
ve bu
16 periyodunda, bayt 2'nin 7. biti
gönderilene kadar böylece devam eder.

Yukarıda gösterildiği gibi, seri iletim, veriyi göndermek için gerekli bağl sayısını 16 kat azaltır; ancak, aynı miktarda veriyi göndermek için gereken zam da 16 katına çıkarır.

Verinin seri olarak gönderilme hızını artırmak için, bitler için gereken periyo kısaltmak, ya da diğer bir deyişle, saat hızını artırmak gereklidir. Yaygın bir şekil kullanılan bazı hız örnekleri daha sonra verilecektir. Bununla birlikte, seri iletim hızının normalde *baud* cinsinden ifade edildiğini belirtmekte yarar vardır. Bir baud saniyede 1 bit demektir. Bu nedenle, bir *megabaud* iletim hızı, saniyede 1 milyon bitlik bir hız anlamına gelmektedir.

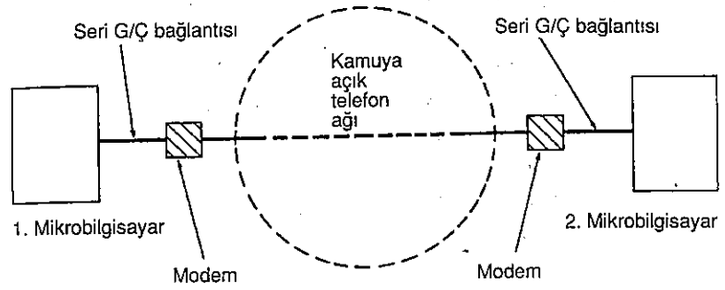
Seri bilgi iletimi, birbirinden uzakta bulunan iki bilgisayar arasında veya bir bilgisayarla uzağında bulunan bir çevresel birim arasında bilgi değiş-tokuşu yapılacağı zaman, özellikle kullanışlı olmaktadır.

Bu durumda bu iki birimi, özel olarak kurulan bir bağlantı ile birbirine bağlamak mümkün olabilir. Bununla birlikte, eğer mesafe çok uzunsa, bağlantıyı gerçekleştirmek için telefon hatlarının kullanılması gerekebilir. Bu durumda, bilgisayar sisteminde kullanılan ikili sinyalleri, telefon hatları boyunca iletime uygun bir şekle dönüştürmek üzere özel cihazlar kullanılır.

Bu cihazlar MODEM (*modülatör-demodülatör*) olarak bilinirler. Böylece, iki mikrobilgisayar arasında telefon şebekesi üzerinden kurulan bağlantı, Şekil 9.11'de gösterildiği gibi yapılır.

Çevresel birimler

Bilgisayarları çevresel birimlere bağlamada, yıllardır seri iletişim bağlantıları kullanılmıştır. Elbette, çevrebirimlerin tümü seri bağlantılar kullanmaz. Manyetik disk saklama birimleri gibi yüksek veri iletim hızına sahip birimlerin, gerekli hızı elde edebilmeleri için paralel yollar kullanmaları gerekir. Fakat, seri bağlantıları



Şekil 9.11

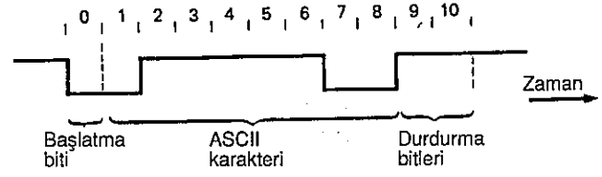
kullanan ve kullanabilen pek çok birim vardır.

Bunların en yaygını teletaypdır. Modern mikrobilgisayar sistemlerinde görsel görüntüleme birimi (VDU) - bir klavye ve televizyon tipi katot-ışınlı ekran- teletaypın yerini almaya başlamıştır. Ancak hâlâ, teletayplar, özellikle basılı kopyanın, yani mikrobilgisayarla yapılan iletişimin basılı kaydının tutulmasının gerekli olduğu durumlarda, görüntüleme ekranından daha fazla tercih edilir.

Teletayplar uzun yıllardan beri kullanıldığı için, teletayplarn işlemciyle nasıl iletişim kuracağını belirten çeşitli standartlar kabul edilmiştir. Bunların en çok kullanılanları, EIA RS 232 standardı ve 20 miliamper akım döngüsüdür (current loop). Bunlar, işlemci ile çevrebirimi arasında değiş-tokuş edilen elektrik sinyallerinin standartlarını tanımlarlar.

Teletaypın (ya da aslında, görsel görüntüleme birimi ve klavyesinin) bir tuşuna basıldığında bilgisayara, çoğunlukla ASCII kodunda bir karakter gönderilir. Standart kod ASCII'dir. ASCII, Bilgi Alışverişi için Amerikan Standart Kodu sözcüklerinin İngilizce karşılıklarının baş harflerini ifade eder. Bu kod:

1 Büyük harflerin,
2 Küçük harflerin,
3 Rakamların,
4 Çeşitli sistem tiplerinde kullanılmak üzere, virgül, nokta ve denetim desenleri



Şekil 9.12 Teletayp iletimi

gibi geniş bir özel karakterler grubunun, standart ikili desenler biçimi gösterilmesine olanak tanımak üzere tanımlanmıştır.

ASCII karakterleri, bilgisayara seri olarak gönderilir ve Şekil 9.12'de göster gibi 8 bitten oluşurlar.

Operatörün her tuşa basışında teletayp, bir karakter gönderir. Karakterlerin gön hızı, operatörün giriş hızına bağlıdır.

Karakterler işlemciye herhangi bir anda gelebileceğinden, karakter iletimi *as rondur*. İşlemcinin bunun üstesinden gelmesine ve verinin gelip gelmediğini de leyebilmesine olanak tanımak için, her bir karakterin başına bir başlatma biti konu

Benzer nedenlerden dolayı, karakterlerin sonuna da iki *durdurma biti* konu (bazen bir durdurma biti, bir de eşlik biti kullanılır, ancak bu bir ayrıntıdır). Han karakterin gönderiliyor olduğunu göstermek için sekiz bit kullanılır, dolayısıy karakter başına gönderilen bit sayısı on birdir. Şekil 9.12'de, 'A' karakteri göndermek için kullanılan 11 bit gösterilmiştir.

Bir teletayp ile mikrobilgisayar arasındaki ya da daha doğrusu, herhangi iki iletişim birimi arasındaki seri iletim *dubleks (ikiyönlü)*, *yarı-dubleks* ya da *simpleks (te yönlü)* olabilir.

Bir *dubleks (iki-yönlü)* sistem, aynı anda iki yönde de bilgi gönderebilir. Örneğin teletayp, mikrobilgisayardan kendisine, yazılmak üzere karakter gönderilirken kendisi de aynı anda mikroişlemciye karakter gönderebilir. Bu tür bir sistemde, her bir iletim yönü için ayrı hat olması gerekir.

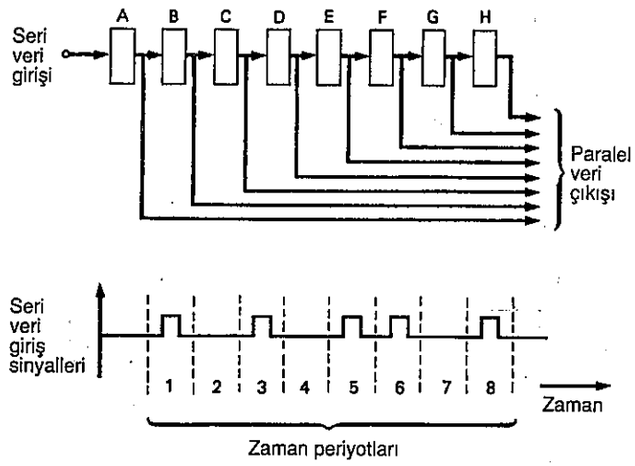
Bir *yarı-dubleks* sistem de her iki yönde veri gönderebilir; ancak, aynı anda iki-yönlü iletim yapamaz.

Simpleks sistem ise tek yönlüdür.

Diğer pek çok çevrebirimleri de, daha önce açıklandığı gibi değişik seri iletim teknikleri kullanırlar. Bu nedenle, bilgisayar sisteminin veriyi bu şekilde, alı gönderebilmesi gerekmektedir.

Seriden paralele ve paralelden seriye dönüşüm

Mikrobilgisayar sistemleri, hemen her zaman paralel biçimli veri üzerinde işle



Şekil 9.13 Seriden paralele dönüşüm

yaparlar. Daha önce görüldüğü gibi, bu sistemler çoğunlukla 8 ya da 16 bitlik veri 'parçaları' kullanırlar. Birimlerin daha sonraki kuşaklarında, bu birimlerin paralelliklerini artırmak yönünde bir eğilim görülmektedir (en yeni işlemciler 32 bitliktir). Bu nedenle, bir mikrobilgisayardan seri olarak sürülen bir çevrebirimine veri göndermek için, paralelden seriye bir dönüştürme gerçekleştirmek gereklidir. Buna karşılık, seri bir hattan alınan veri, mikrobilgisayar belleğine yerleştirileceği zaman, seriden paralele bir dönüşümü yapmak gerekecektir.

Kaydırmalı kaydedici, seriden paralele ya da paralelden seriye dönüşümü gerçekleştirmede çok uygun bir birimdir. Şekil 9.13'de bu şematik biçimde gösterilmiştir.

Kaydırmalı kaydediciler, belli sayıda bir-bitlik bellek ya da *flip-flop*'lardan (iki-durumlu birimlerden) oluşmuştur. Bunlardan daha önce, veri sabitleştiricilerin birimleri olarak söz edilmişti. Şekilde, A, B, C,...,H olarak adlandırılmış ve A'nın çıkışı B'nin girişine, B'nin çıkışı C'nin girişine vb. girecek şekilde bağlanmış 8 flip-flop (iki durumlu birim) gösterilmiştir.

Veriler, "seri veri girişi" hattından, A flip-flopunun girişine gelir. Şeklin altında gösterilen 8-bitlik bir desenin bu girişe uygulandığını varsayalım. Giriş sinyali, 2. periyodunda 0; 3. periyotta 1 vb. değerindedir. 1. periyot süresinde A flip-flopunun girişine (kaydırma kaydedicisinin A *katına*) verilen sinyal A katına alınır ve saklanır. Böylece, 1. periyodun sonunda, A katında 1 değeri bulunur.

Yine, 1. periyot süresinde, B katının girişi (A katının çıkışındaki) sinyal, B katı alınır ve saklanır. Aynı işlem, kaydedici boyunca devam eder. Böylece, C katı B katının çıkışını, D katı C katının çıkışını saklar.

Eğer başlangıçta, 1. periyodun öncesinde, kaydedici boyunca "0" değeri tutuluyorsa, 1. periyodun sonunda kaydedicideki desen:

10000000

biçiminde olacaktır.

2. periyot süresinde, yine aynı işlemler gerçekleşir. Veriler, giriş hattındaki - burada 0 olan- sinyal A katına, A'nın çıkışı (1) B'ye alınır, vb.. Böylece, 2. periyodun sonunda, kaydedicideki desen:

01000000

biçimini alır.

3. periyodun sonunda:

10100000

4. periyodun sonunda:

01010000

5. periyodun sonunda:

10101000

6. periyodun sonunda:

11010100

7. periyodun sonunda:

01101010

Son olarak 8. periyodun sonunda da:

10110101

biçimini alır.

Görülebileceği gibi, giriş hattındaki seri veri, kaydedicide kaydırılmaktadır.

8. periyot sona erdikten sonra, artık kaydedicide saklı olan giriş deseni, şekilde gösterilen sekiz veri çıkış hattı kullanılarak paralel biçimde okunabilir. Bu, tek bir periyot sürer. Bu nedenle, kaydırmalı kaydedici, bir *seriden-paralele dönüştürücü* olarak işlev görmektedir.

Verilerin, kaydırmalı kaydedicinin A katına yerleşebileceği ve bir kattaki verinin bir sonraki konuma kaydırılacağı anların kesin olarak tanımlanabilmesi için, kaydedici normal olarak (şekilde gösterilmeyen ve) üzerine zamanlama darbelerinin yerleştirildiği bir saat girişine sahiptir.

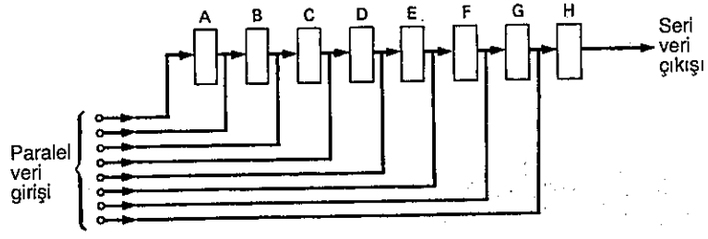
Şekil 9.14'de, bir *paralelden-seriye dönüştürücü* gösterilmektedir. Çalışma ilkesi temelde, az önce anlatılan seriden paralele dönüştürücüyle aynıdır. Ancak, bu durumda veriler 'paralel veri giriş' hatları kullanılarak, tek bir periyot süresince kaydedicinin her katına paralel bir biçimde, ayrı ayrı yerleştirilir. Ardından, sekiz periyot süresince, kaydedicinin çıkışına, 'seri veri çıkışı' hattına kaydırılır.

Bir teletayp ya da bir görsel görüntüleme birimi gibi, seri bir çevresel birim ile iletişim kurabilmek için işlemci, bir kaydırmalı kaydedici kullanan paralelden-seriye dönüştürücüye paralel veri gönderebilir. Ardından, seri biçimdeki veri kaydediciye, verileri kaydedicinin çıkışına kaydıracak şekilde, sekiz saat darbesi uygulayarak seri veri elde edilebilir.

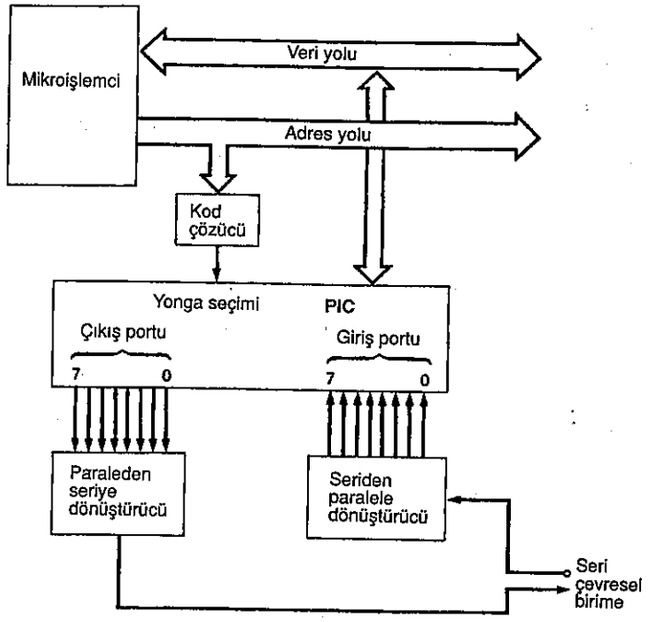
Benzer bir şekilde, bir teletayptan gelen seri veri, bir seriden paralele dönüştürücü içinden kaydediciye kaydırılır. Daha sonra, paralel olarak işlemciye okunur.

İşlemci ile bir seri çevrebirimi arasındaki iletişime ilişkin komple bir sistem Şekil 9.15'de gösterilmektedir. Bir çevresel arabirim devresinin bir bölümü, 8-bitlik paralel bir veriyi, bir paralelden seriye dönüştürücüye ve oradan da seri olarak birime göndermek için kullanılır.

Bir giriş olarak ayarlanan PIC'in diğer bir portu bir seriden-paralele dönüştürücüden veriyi okur.



Şekil 9.14 Paralelden seriye dönüştürme



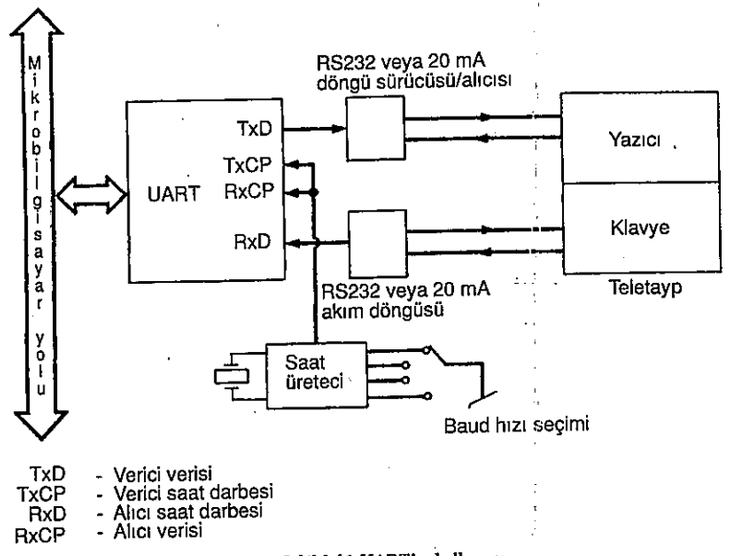
Şekil 9.15

Seri iletim için LSI yongaları

1. Bölümde söz edildiği gibi, mikrobilgisayar sistemleri ile seri veri iletim hatları arasında bir arabağlantı kurmak amacıyla kullanılan, çok çeşitli büyük ölçekli entegre (LSI) devreler bulunmaktadır.

Bu türde yongalar, UART (Genel Asenkron Alıcı ve Verici) sınıfı yongalar olarak bilinir. Şekil 9.16'da, bu tür bir yonganın, bir teletayp ile mikrobilgasayarı bağlamak üzere nasıl kullanıldığı gösterilmektedir. UART doğrudan paralel mikrobilgisayar veri yoluna bağlanır. İçinde, gerekli üç-durumlu sürücüler ve bu yoldan doğru biçimde yararlanmak için gerekli alıcı devreler bulunur.

Asenkron seri iletim, yalnızca göreceli olarak daha düşük (9600 baud'a kadar) hızdaki bağlantılar için uygundur. Bu değerin altında, 110 baud'dan başlayan bir iletim hızı aralığında, teletayp hızı kullanılır. Bu nedenle, UART'ın çalışma frekansı, bir kristal denetimli osilatör tarafından belirlenir. Bazı sistemlerde, UART'a uygulanan saat darbelerinin hızı, şekilde gösterildiği gibi, alma ve gönderme baud



Şekil 9.16 UART'ın kullanımı

hızlarının istendiği gibi ayarlanmasına imkan verecek şekilde, 'anahtarla-seçilebilir' özelliktedir.

UART içinde bulunan alıcı kendini, teletayp tarafından gönderilen başlatma bitiyle senkronize eder ve ardından gelen veri bitlerini örnekler. Bir karakterin tümü alındığında, (ASCII kodundaki eşlik biti kullanılarak) kontrol edilir ve veri anayolu üzerinden işlemciye gönderilir. UART, alınan veri üzerinde gerekli seri-paralel dönüşümü gerçekleştirir.

UART'ın verici bölümü, mikrobilgisayar yolundan gelen paralel verileri seriye dönüştürür, başlatma, durdurma ve eşlik bitlerini ekler ve bağlantı üzerinden veriyi gönderir.

UART, mikrobilgisayar yolu ile kendi arasındaki veri aktarımlarının senkronizasyonunu sağlamak için, denetim giriş ve çıkışlarına sahiptir. Bu nedenle, seri
S 9.11 birimler ile arabağlantı kurmak için gerekli tüm fonksiyonları içerir.

Senkron seri veri iletimi

Asenkron veri iletimi (ve alımı), daha önce belirtildiği gibi, yalnızca göreceli olarak

daha düşük hızdaki bağlantılar için uygundur. Bir iletişim ağında bulunan çok sayıdaki mikrobilgisayarın birbirleriyle bağlantısı örneğinde olduğu gibi, daha yüksek hızdaki bağlantılar için, *senkron* iletim tercih edilecektir.

Asenkron iletim, kesikli çalışma modu nedeniyle bazen, bir *başlat-durdur* iletim şeklinde de tanımlanır. Diğer yandan, eşzamanlı iletimde karakterler, iletim hattı boyunca sürekli olarak kesintisiz bir biçimde gönderilirler.

Bir hat boyunca gönderilecek veri yoksa, alıcı ve verici birimlerin senkron olarak birbiriyle kilitli kalmasını sağlamak üzere bir "senkronizasyon karakterleri" dizisi gönderilir. Bu, başlatma ve durdurma bitlerinin eklenmesi gereğini ortadan kaldırır, böylece hat kullanımı daha verimli olur.

Bir iletimdeki karakter dizisi, bloklara bölünür ve her bir bloğun sonunda hata-denetimi yapılır. Blok uzunluğu, alıcı ve verici birimlere ve gönderilen verinin yapısına bağlı olarak değişir. Örneğin, delikli kartlardan gelen veriler, bir bağlantı üzerinden gönderilirse, her biri 80 karakter içeren bloklar kullanmak uygun olacaktır. Kart, her biri tek bir karakteri gösterebilen 80 sütun içerdiğinden, gönderilen her blok da tek bir kartı gösteriyor olacaktır.

Geçtiğimiz son birkaç yıl içerisinde, çeşitli uluslararası standartlar komiteleri kurulmuştur. Bunlar, veri iletim şebekeleri için, eşzamanlı iletim blok ve mesaj formatlarına ilişkin esasları içeren *'protokoller'* tanımlamışlardır.

Birkaç yıldan beri piyasada, eşzamanlı seri veri iletiminde kullanılabilen büyük ölçekli entegre devreler bulunmaktadır. Bu tür devreler USART (Genel Senkron/Asenkron Alıcı Verici) birimlerdir. Bu birimler mikrobilgisayar adres ve veri anayollarına her zaman olduğu gibi bağlanır ve asenkron ya da senkron modda çalışmak üzere programlanabilirler.

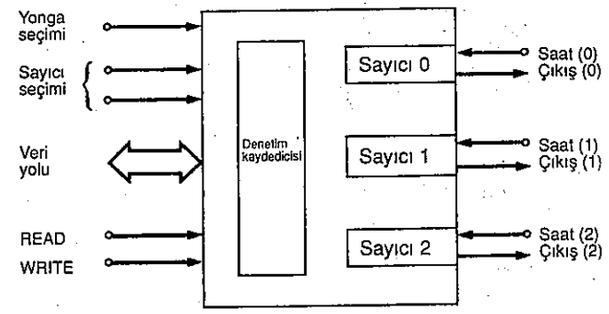
S 9.9 9.12 Eşzamanlı çalışma için iletim hızları 56 Kbaud'a kadar değişir (Intel 8251 programlanabilir iletişim arabirimi).

9.5 Çevresel zamanlama devreleriyle arabağlantı kurulması

Aralık Zamanlayıcıları

Aralık zamanlayıcılarının donanımı ve mikrobilgisayar anayolu ile arabirim üzerinden bağlantı kurulma yolları, Kısım 2.5'de anlatılmıştı. Burada, aralık zamanlayıcısının, her biri normal bir OUT komutu kullanarak, mikrobilgisayardan bir değer ile yüklenebilen birçok sayıcı içeren bir birim olduğu anımsatılacaktır. Sayıcılar, daha sonra "saat" giriş hatlarına dışsal bir kaynaktan darbeler uygulamak suretiyle, bu değerden geriye saydırılabilir.

Örnek olarak, bu türde bir zamanlayıcının blok diyagramı Şekil 9.17'de gösterilmiştir. 'Saat 0' hattına, saat darbelerinin uygulanması, Sayıcı (0)'ın geriye saymasına neden olur. İçeriği sıfır olduğunda, ÇIKIŞ (0) çıkışından bir sinyal üretilir. (1) ve (2) sayıcıları da benzer bir biçimde çalışır.



Şekil 9.17

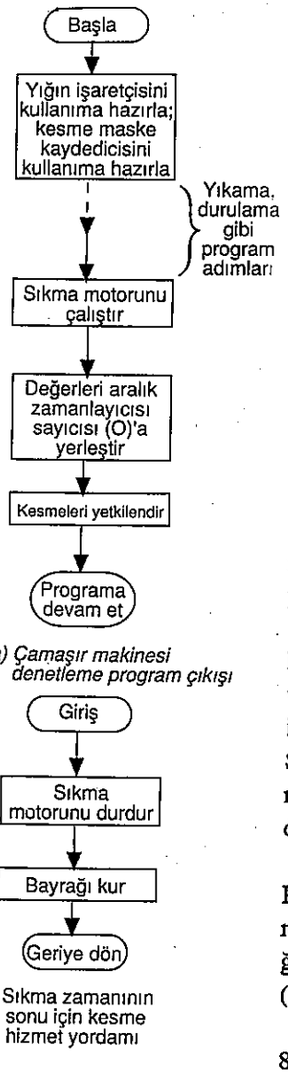
Pek çok mikrobilgisayar uygulaması, zaman aralıklarının hassas biçimde ölçümünü gerektirir. Bu aralıklar, uygulamaya göre birkaç mikrosaniyeden dakikalara kadar değişebilir. Bu nedenle, zamanlayıcı yongasının, mümkün olduğu kadar esnek olması önemlidir.

Tipik olarak, her bir sayıcı 16 bit içerir. Bu nedenle, içinde 65 535'e (2^{16}-1) kadar bir sayı saklanabilir. Giriş saat frekansının 50 Hz olduğu varsayılırsa, ölçülebilecek maksimum aralık:

65.535/50sn
= 1.310,7sn
= 21 dak 50.7sn'dir.

Bu periyot:

1 sayıcıya yerleştirilen değeri değiştirmek ya da
2 saat frekansını değiştirmek, suretiyle değiştirilebilir.



a) Çamaşır makinesi denetleme program çıkışı

b) Sıkma zamanının sonu için kesme hizmet yordamı

Şekil 9.18

Eğer daha uzun bir aralık istenirse, saat giriş frekan azaltılması gerekir. Daha kısa bir aralık gerektiğinde sayıcıya yerleştirilen değer azaltılabilir. Örneğin, 10 Hz'lik saat frekansı, yaklaşık 100 dak. kadar uzunluktaki periyot ölçümüne izin verecektir. Buna karşılık, orijinal 50 Hz'lik frekansı kullanılırsa, sayıcıya yerleştirilen bir 10,000 d 200 sn' lik bir periyot değeri verecektir.

Sayıcıdaki değer sıfıra ulaştığında, ÇIKIŞ hattında üreti sinyal genellikle mikrobilgisayarı kesmek için kullanılır. A lık zamanlayacılarının kullanımına bir örnek, evde kullanıl çamaşır makinalarıdır. Bu tür araçlarda:

1 Yıkama süresi
2 Durulama süresi
3 Sıkma süresi

gibi birkaç zaman aralığının ayarlanması gerekir.

Bu nedenle, bir çamaşır makinesinde denetleyici olarak kulla nılan bir mikrobilgisayarın, bu işlemler için uygun uzunlukta periyotlar üretmesi gerekir.

Mikrobilgisayarın bir aralık zamanlayıcısı yongası içerdiğini ve örneğin Sayıcı (0)'ın makinenin sıkma zamanını denetlemek için kullanılacağını varsayalım. Sayıcı (2), durulama süresi ve Sayıcı (3) de, yıkama süresi için kullanılabilir. Çamaşır maki nasını denetleyen programın akış diyagramının Şekil 9.18 (a)'- daki diyagramda verildiği gibi bölümler içermesi gerekir.

Programın başında, yığın işaretçisi, bellekteki yığının konu munu belirlemek için kullanıma hazırlanır. Bundan sonra yı ğın, artık herhangi bir kesme geldiğinde program sayıcısını ve (gerekirse) kaydedici değerlerini kabul etmeye hazırdır.

8. Bölümde ana hatlarıyla belirtildiği gibi, kesme maske kaydedicisi, seçilen kesmelerin meydana gelmesine izin verecek biçimde ayarlanır. Eğer bu örnek için, ÇIKIŞ (0) çıkış sinyalinin, RST 5.5 girişinde bir kesme gerçekleştirebilmesi için kullanıldığını varsayarsak, maske bitinin, buna izin

vermek veya en azından işlemciye ulaşmak için kurulması gerekir.

Program, yıkama çevriminde, sıkma yapılması istenen noktaya ulaşılana kadar, çamaşır makinesinin diğer çeşitli işlevlerini denetlemeye devam eder. Daha sonra motor, motoru denetleyen çevrebirimi portuna bir OUT komutu göndermek suretiyle, sıkma motorunu çalıştırır, aralık zamanlayıcısı kaydedicisine [Sayıcı (0)'a] sıkma süresini denetleyen bir sayı yükler ve bir EI komutu ile kesmeleri yetkilendirir.

Ardından, gerekli diğer işlemlere devam eder. Sıkma işlemi bitene kadar, yapacak başka bir şey yoksa, program 'Sıkma süresinin sonu' kesmesinin gelmesini bekleyerek döngüde kalır.

Bu kesmenin ne ortaya çıkardığını söyleyebilmesi için, program, her bekleme döngüsü sonunda bir 'bayrağı' (bir-bitlik işaretçi) gözden geçirir. Bu bayrak normalde sıfırlanmıştır (0'dadır) ancak görülebileceği gibi kesme hizmet yordamı tarafından 1'e kurulur.

Aralık zamanlayıcısının Sayıcı (0)'ına bir kez bir sayı yerleştirildiğinde, zamanlayıcı giriş saatinin hızında geriye doğru saymaya başlar. 50 Hz'lik bir sinyal, a.c. (alternatif akım) şebekesinden kolaylıkla üretilebileceğinden, bu uygulama için saat hızının 50 Hz olduğu varsayılmıştır.

Sayıcı (0)'daki değer sıfıra ulaştığında, ÇIKIŞ (0) çıkış hattından çıkan sinyal, RST 5.5 kesme girişine uygulanarak mikrobilgisayarın kesintiye uğramasına ve bu da, Şekil 9.8 (b)'de gösterilen kesme hizmet yordamına girilmesine neden olur.

Kesme hizmet yordamı çok kısadır: Sıkma motorunu kapatır, yukarıda belirtildiği gibi (ana programa bir zamanlayıcı kesmesi geldiğini göstermek için) 'bayrağı' kurar ve kesildiği noktada denetimi ana programa aktarır.

Ana programdaki ve kesme hizmet yordamındaki çevirici dili komutlarından bazıları şöyledir:

```
Ana Program
        LXI     SP, 10D4    ;Yığın işaretçisini kullanıma hazırla
        MVI     A, 0E       ;RST 5.5 için kesme maskesini
        SIM                 ;kullanıma hazırla
        .
        .
        .
        MVI     A, 01       ;C portu aracılığıyla sıkma moturunu çalıştır
        OUT     0A          ;(bir önceki örnekte verilen adres)
        MVI     A, 30       ;Aralık zamanlayıcısındaki Sayıcı (0)'a
        OUT     13          ;16-bitlik bir sayı ata.
        MVI     A, 30
        OUT     10
        MVI     A, 75
        OUT     10
        EI
        devam et
        ----------------------------------------------
Kesme hizmet yordamı
INT:    MVI     A, 00       ;C portu aracılığıyla sıkma moturunu durdur
        OUT     0A
        MVI     B, 01       ;(B) yi 1'e ayarla
        RET                 ;Geriye gön
```

Yukarıda, yığın işaretçisinin bir önceki örnekte olduğu gibi 10D4 (on altılı) olarak kullanıma hazırlandığı ve PIC'in C portunun sıkma motorunu denetlemek için kullanıldığı varsayılmıştır. Bu portun 0. biti motoru çalıştırmak için 1'e, kapamak için de 0'a ayarlanır.

RST 5.5 kesmesi geldiğinde, işlemci program yürütümünün denetimini 002C (on altılı) adresindeki komuta aktarır. Bu nedenle, bu bellek konumunun, yukarıdaki kesme hizmet yordamına girişe neden olacak şekilde, JMP INT komutu içermesi gereklidir.

16 bitlik bir sayıyı Sayıcı (0)'a aktarmak için üç basamak gereklidir. Aralık zamanlayıcısı bir denetim kaydedicisi içerir (Şekil 9.17) ve ilk olarak bu kaydediciye bir denetim sözcüğünün yerleştirilmesi gerekir. İkinci olarak, sayının en küçük değerlikli 8 biti Sayıcı (0)'a ve son olarak, en büyük değerlikli 8 biti de Sayıcı (0)'a gönderilir.

Mikrobilgisayar yolundaki aralık zamanlayıcısının adresi 10 (on altılı) ise, denetim kaydedicileri ile ve 0, 1 ve 2 sayıcılarının adresleri:

Sayıcı 0	10 (on altılı) (=10+00)
Sayıcı 1	11 (on altılı) (=10+01)
Sayıcı 2	12 (on altılı) (=10+02)
Denetim kaydedicisi	13 (on altılı) (=10+03)

Böylece, ilk olarak:

```
MVI A,30
OUT 13
```

komutları ile denetim sözcüğü gönderilir.
İkinci olarak, sayıcı değerinin en küçük değerlikli baytı gönderilir:

```
MVI A,30
OUT 10
```

Son olarak da, en büyük değerlikli bayt gönderilir:

```
MVI A,75
OUT 10
```

Sıkma süresinin 10 dakika olduğunu varsayalım. Böylece, Saat (0)'a uygulanan 50 Hz'lik bir saat hızı ile, Sayıcı (0)'a yerleştirilen sayı, 30.000 (on dalık) yani 7530 (on altılı) olmalıdır. Bu nedenle en küçük değerlikli bayt, 30 (on altılı) ve en büyük değerlikli bayt ise 75'dir (on altılı).

S 9.13

Gerçek-zamanlı saatler

2. Bölümde, aralık zamanlayıcısı yongalarının gerçek-zaman saatleri olarak kullanıldığından söz edilmişti. Bunlar da pek çok uygulama için oldukça önemlidir.

Bir evdeki işlemleri denetlemek üzere mikrobilgisayar kullanıldığını düşünelim. Bu tür bir sistemde, mikrobilgisayar için günlük zamanın kaydının tutulması önemlidir. Zamana bağlı işlemleri denetleyebilmek için buna ihtiyaç vardır. Örneğin, sistem belli zamanlarda merkezi ısıtmayı açabilir ya da her akşam çocukların banyo zamanından hemen önce su ısıtıcısını açabilir. Ev halkını sabahları uyandırmaktan ve buna benzer diğer zamana bağlı işlemlerden sorumlu olabilir.

Bir aralık zamanlayıcısının kullanımı bu tür uygulamalarda mikrobilgisayara yardımcı olabilir. Mikrobilgisayar, bellekte, günün o anki zamanının bir kaydını tutar.

Aralık zamanlayıcısı da, bir dakikalık aralıklarla kesme göndererek mikrobilgisayarı keser.

Her bir kesme gelişinde, beraberindeki hizmet yordamı, dakika cinsinden zamana birer dakika ekleyerek zamanı günceller. Sonuç altmış dakikaya ulaştığında, saat cinsinden zaman bir artırılır ve dakikalar sıfırlanır.

Bu işlemler, işlemcinin, her dakikada yalnızca birkaç komutu yürütmesini gerektirdiğinden, işlemci zamanı açısından küçük bir yük getirirler.

Mikrobilgisayarda, mikrobilgisayarın doğru zamana ayarlanabilmesinde kullanılacak anahtar ya da basmalı düğmeler de bulunması gerekir. Bununla birlikte, bu ayarlama bir kez yapıldığında, saklanan zaman değerlerinin duyarlılığı, artık yalnızca aralık zamanlayıcısına uygulanan saat frekansının duyarlılığı ile sınırlı bulunmaktadır.

Mikrobilgisayar, bu yöntemi kullanarak 'gerçek' zamanın daima doğru bir kaydını tuttuğundan, yukarıda ana hatlarıyla belirtilen uygulamalarda kullanılabilir.

Zamanlayıcı tarafından üretilen kesmelere ilişkin işlemler, bir önceki örnekte verilenlerle hemen hemen aynıdır.

Sorular

9.1 Bir mikrobilgisayar ile çevresel birimleri arasındaki veri iletimi kapsamında sözü edilen ana ve uydu birimler ile ne anlatılmak istendiğini açıklayın. Buna göre, kesme ile sürülen bir sistem ile tarama sistemleri arasında ne gibi farklılıklar olduğunu açıklayın.

9.2 Çevresel birimlerin taranması yöntemini kullanan bir sistem nasıl çalışır?
(a) Tarama sistemlerinde öncelik sırasının verilişini,
(b) Taramanın avantajlarını ve
(c) Dezavantajlarını
tartışın.

9.3 Bir seyahat acentasında kullanılan mikrobilgisayara, işlemcilerin sistem belleğinde saklı bilgileri soruşturmak için kullanabileceği sekiz ekran bağlıdır. Bu birimleri denetlemek için bir tarama sisteminin nasıl kullanılabileceğini tartışın.

9.4 Mikrobilgisayar belleğinde, 10.000 (ondalık) ile 10.255 (ondalık) adresleri arasındaki baytlara bir veri bloğu saklanmıştır. Sekiz adet ışık-yayan diyotun, bir çevresel arabirim devresi kullanılarak mikrobilgisayara nasıl bağlanabileceğini ve veri baytlarını birbiri ardına incelemek için nasıl kullanılabileceğini gösterin. Her bir bayt, LED'lerde yaklaşık 5 saniye süreyle görüntülenmelidir.

9.5 4. soruda tanımlanan sistemde, giriş portuna, düğmenin her basılışında yeni bir bayt görüntülenebilecek şekilde bir basma düğmesi bağlanabilir mi? Bu tür bir sistemin, bir önceki soruda olduğu gibi kısa bir süreyle değil de, düğmenin bir sonraki basılışına kadar görüntülenen baytlarla bellek üzerinde döngü yapabileceğini varsayalım. Buna olanak verecek yazılım ve donanımlar konusunda öneriler getirin.

9.6 (Intel 8255 A gibi) bir PIC'in A ve B portlarını giriş olacak şekilde kullanıma hazırlayacak kısa bir kod parçası yazın. Şekil 2.13'de gösterilen biçimde denetim kaydedicisi kullanın. Programı, iki baytlık verinin bu portlardan okunmasına ve 8085 işlemcisinin B ve C kaydedicilerine yerleştirilmesine olanak verecek biçimde devam ettirin.

9.7 A ve B portlarını giriş, C portunu ise çıkış olarak kullanıma hazırlayacak şekilde kısa bir kod parçası yazın. Önce A portu ve ardından da B portu üzerinden birer veri parçası okuyun, ikisini toplayın ve sonucu C portuna gönderin.

9.8 Işık yayan bir diyotun, bir mikrobilgisayar sisteminin bir çıkış portuna nasıl bağlanabileceğini ve uygun bir program ile birlikte, birer saniyelik aralıklarda 500 msn'lik yanma süresi elde etmek için nasıl kullanılabileceğini gösterin. 7. Bölümde anlatıldığı gibi, gerekli zaman aralıklarını üretmek için bir zaman gecikmesi alt yordamı kullanın ve her bir komutun 2 µsn. sürdüğünü varsayın.

9.9 Bir mikrobilgisayar sisteminde veri giriş çıkışı niçin önemlidir? Seri ve paralel G/Ç arasındaki farklılıkları belirtin ve bunlara yardımcı olarak kullanılabilecek devreleri tartışın.

9.10 Aşağıdaki terimler ne anlama gelmektedir?
(a) Kesme ile sürülen sistem
(b) Tarama sistemi
(c) Tam dubleks
(d) Yarı dubleks
(e) Eşzamanlı

9.11 *(a)* Kaydırmalı bir kaydedicinin, seri veri giriş-çıkışı yapan bir mikro
ciye yardımcı olmak amacıyla nasıl kullanılabileceğini açıklayın.
(b) Bir UART tarafından sunulan olanakları kısaca belirtin ve seri G/Ç
nasıl kullanılabileceğini gösterin.

9.12 *(a)* Asenkron ve
(b) Senkron seri veri iletimi arasındaki
farklılıkları belirtin.
Bunlar arasındaki farklılıkları vurgulayarak uygulama alanlarını tartışın.

9.13 Bir aralık zamanlayıcı yongası, kare dalga darbe dizisi üretiminde, mikro
lemciye yardımcı olmak üzere kullanılacaktır. Kare dalganın periyodu, 50
msn ile 100 msn arasında değişmektedir. Aralık zamanlayıcısına uygulanac
saat girişi için uygun frekansları tartışın ve sayıcıya yerleştirilecek uygu
değerler önerin. Bu uygulamanın programını ana hatlarıyla belirtin.

9.14 Dört anahtar (A, B, C ve D) ve dört ışık-yayan diyot (L, M, N ve O) bir
mikrobilgisayarın çevresel arabirim devrelerine bağlanacaktır. Sistemin, a
nahtar durumlarını algılaması ve aşağıdaki kurallara göre LED'leri yakıp
söndürmesi istenmektedir:
(a) C kapalı olduğunda L yanar, açık olduğunda söner.
(b) C ve B kapalı, D açık olduğunda M yanar, A durum değiştirmez.
(c) A kapalı ve M yanık olduğunda, N yanar.
(d) A veya B kapalı ve C ve D açık olduğunda O yanar.
Bu işlemleri gerçekleştiren bir program yazın.

Bölüm 10 Programlama

Bu bölümün amaçları: *Bu bölümü bitirdiğinizde:*

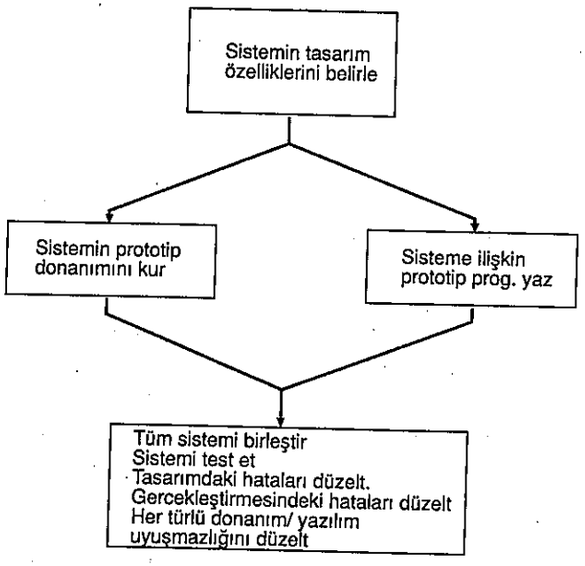
1 *Mikrobilgisayar-tabanlı bir sistem tasarımının, yazılım ve donanımın her ikisinin geliştirilmesini gerektirdiğini değerlendirebilmeli,*
2 *Bu geliştirme çalışmalarındaki önemli bir aşamanın, yazılım ve donanımın birleştirildiği ve bunlardaki hataların algılanıp düzeltildiği aşama olduğunu değerlendirebilmeli,*
3 *Yazılım geliştirmenin, sistem özelliklerinden makine kodu programına kadar birçok aşamayı içerdiğini anlayabilmeli,*
4 *Programcıya yardımcı olmak ve her birine ilişkin genel bir fikir edinmek için, (düzenleyiciler, yükleyiciler, çevirici programları, yüksek düzeyli dil derleyicileri, kütüphaneler ve hata ayıklama programlarını içeren) birçok yazılım geliştirme yardımcıları olduğunu değerlendirebilmeli,*
5 *Assembly dili deyimlerinin formatını anlayabilmeli ve assembly dilindeki sözde komutları kullanabilmeli,*
6 *(a) G/Ç programlarının kullanımı*
(b) Alt programların kullanımı
(c) Kesme hizmet yordamlarının kullanımı dahil olmak üzere, komple bir programın yapısını kavrayabilmelisiniz.

10.1 Giriş

Bir mikrobilgisayar sisteminde yazılım ve donanımın her ikisinin de gerekli olduğu, daha önceki bölümlerde birkaç kez vurgulanmıştı. Mikrobilgisayar sisteminin donanımla ilgili yönleri (işlemci, bellekler, çevresel G/Ç yongaları, aralık zamanlayıcıları ve benzeri) ayrıntılı olarak ele alınmış ve uygun yerlerde, bunların programla olan karşılıklı etkileşimleri ana hatlarıyla belirtilmişti. Bu bölümün ana amacı, mikrobilgisayarın yazılımını biraz daha ayrıntılı olarak ele almak ve bir mikrobilgisayarın komple bir uygulama örneğini bir kronometre uygulaması üzerinde incelemektir.

10.2 Mikrobilgisayar Sistem Tasarımı

Mikrobilgisayarlar gibi programlanabilir bir birimi de içine alan komple bir mühendislik sistemi tasarımı, zorunlu olarak yazılım ve donanımın her ikisinin de



Şekil 10.1

geliştirilmesini gerektirir. Geliştirme çalışmaları, tipik olarak, Şekil 10.1'de gösterilen bir sırada yapılır.

Bünyesinde mikrobilgisayar bulunduran yeni bir sistem (belki yeni bir ürün) tasarlanacaksa, ilk adım, bir sistem belirtiminin (spesifikasyonunun) oluşturulması olmalıdır. Bu, sistemin karşılaması gereken teknik amaçları içine almaktadır.

Bir kez bu aşama tamamlandığında, sistem yazılım ve donanımın geliştirilmesi, birbiriyle paralel olarak büyük bir hızla ilerleyebilir. Bu nedenle, bir mühendis grubu, fiziksel sisteme girişi yapılacak ve çalışmasını denetleyecek programları yazmaya girişirken, diğer bir grup bu fiziksel sistemi oluşturan parçaları bir araya getirebilir.

Son olarak tüm sistem bir araya getirilir ve yazılım ile donanım birlikte test edilir. Bu aşamada, birçok hatanın ortaya çıkması kaçınılmazdır. Yazılım ve donanım geliştirme grupları, ayrı ayrı çalışırlarken yakın ilişki içerisinde olsalar da, küçük yanlışlık ya da kusurların yazılım ve donanım arasında bazı uyuşmazlıklara neden olacağı durumlar da olacaktır.

Yazılım grubu kendi çalışma alanında hata yapabileceği gibi, donanım grubu da kendi alanında hatalar yapacaktır. Başta sistemin tasarım belirtiminde yetersiz

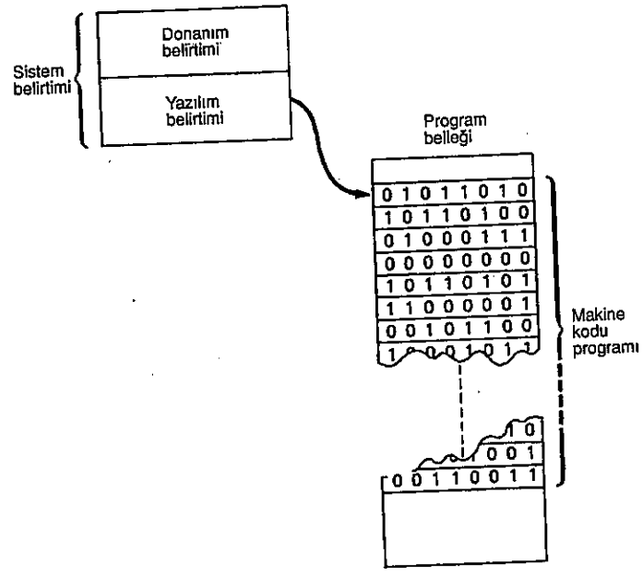
kalınmış ya da hata yapılmış olabilir. Tüm bu faktörlerin neden olduğu problemlerin, sistemin test edilmesi aşamasında algılanması ve düzeltilmesi gereklidir.

Yazılım Geliştirme

Yukarıda belirtildiği gibi, bu bölüm genelde, sistem yazılımının geliştirilmesi ile ilgilidir. Bu, kendi içinde de belli aşamalar içerir.

Tüm sistem belirtimi, yazılım ve donanım amaçlarını kapsayan bölümler içerir. Yazılım tasarımı, bu belirtim esaslarına uygun olarak başlar ve mikrobilgisayarın program belleğindeki makine kodu komutları üretimi ile sona erer. Bu, Şekil 10.2'de gösterilmektedir. Açıkça görülmektedir ki, bu iki nokta arasında, genellikle tek bir işlemde tamamlanması çok zor olan büyük bir mesafe bulunmaktadır. Diğer bir deyişle, belirtim ile makine kodu programı arasında, sırayla birinden diğerine ilerlemeye yardımcı olacak şekilde, ara aşamalar olması gereklidir.

İlk adım, programlanacak problemi çok basit bir işlemler dizisi biçiminde ele almaktır. Altbölümlere ayırma işlemi, bu amaca uygun bir *algoritma* oluşturmak olarak bilinir. Algoritma bir işlem listesi olarak yazılabilir ya da daha önceki örneklerde kullanılan bir akış diyagramı şeklinde çizilebilir.



Şekil 10.2

Bunun ardından, algoritmadaki adımların program komutu olarak yazılmaları ya algoritmanın *kodlanması* gerekir. Bunun yapılabileceği birkaç yol vardır.

Çok basit bir mikrobilgisayar sisteminde, algoritmayı doğrudan makine kodu komutları olarak kodlamak gerekebilir. Eğer böyle yapılacaksa, algoritmanın kendisinin de çok ayrıntılı olması ve hazırlanışında çok özen gösterilmesi gerekir. Bundan başka, önerildiği gibi, ister ikili isterse de on altılı biçimde olsun, programların makine koduyla yazılması ne çabuk ne de kolay bir yoldur.

Kodlama işlemini gerçekleştirmenin çok daha iyi bir yolu, makine kodundan daha *yüksek-düzeyli* bir dil kullanmaktır. 1. Bölümde anlatıldığı ve daha önceki birkaç örnekte de kullanıldığı gibi, assembly dili makine kodundan bir düzey daha yüksektir.

Hatırlanacağı gibi, assembly dili programcının, adres ve veri için isim ve etiketlerle birlikte bir komutun işlem kodu kısmını göstermek için anımsatıcı kullanmasına imkan tanır. Bununla birlikte, assembly dili komutları bir çevirici programından geçirildiğinde, yalnızca bir makine kodu komutu üretilir. Bu nedenle, assembly dilinde programlamak da yine, çok ayrıntılı bir algoritma kullanılmasını gerektirmektedir.

Yüksek düzeydeki diller (HLL'ler), yapılacak işin belirtildiği dil ile programın yazıldığı dil arasındaki boşluğu tamamlamak için tasarlandığından, daha iyi bir çözüm getirirler.

Bir program, FORTRAN, PASCAL, BASIC, ALGOL ya da COBOL gibi yüksek düzeyli bir dilde yazıldığında, ilgili programın mikrobilgisayar belleğine yerleştirilmeden önce makine koduna dönüştürülmesi gerekir. Bu, bir *derleyici* ile gerçekleştirilir. Derleyici, bir anlamda Kısım 1.7'de sözü edilen çevirici gibi bir programdır. Giriş verisini, burada FORTRAN, ALGOL, vb.'deki kaynak programı alır, makine diline dönüştürür ve mikrobilgisayar belleğine yerleştirir.

Derleyicilerle çevirici arasındaki fark, derleyicinin çok daha karmaşık olmasıdır. Yüksek düzeyli diller, uygulama yönelimlidirler, yani programcı tarafından gerçekleştirmek istenen işlemleri ifade edecek deyimlerle program yazımına olanak tanırlar. Bu deyimlerin makine koduna çevirilmesi, her deyime karşılık çok sayıda makine kodu komutu üretilmesine neden olur.

Bu nedenle, derleyicinin görevi çeviricininkinden daha zordur. Bunun bir sonucu olarak derleyiciler, mikrobilgisayar belleğinde büyük boyutlarda saklama alanı

gerektiren büyük programlar üretme eğilimindedirler.

Yüksek düzeyli dillerin avantajı, kullanımlarının kolay olmasıdır. Yüksek düzeyli dil kullanılarak karmaşık ifadeler yazılabileceği için, verilen bir görev için gerekli algoritma büyük oranda kısaltılıp basitleştirilebilir. Kaynak programın okunması çok daha kolaydır.

Son olarak, yüksek düzeyli dil kullanımı, programların bir mikrobilgisayardan diğerine taşınabilir olması anlamına gelmektedir. Bunun nedeni şudur: kaynak programlar makineden-bağımsız olmasına karşın, derleyicilerin herhangi bir makineye uyarlanması sözkonusu değildir. Örneğin:

1 Bir Z80 mikrobilgisayar sisteminin FORTRAN derleyicisi, FORTRAN kaynak programlarını Z80 makine koduna dönüştürür.
2 Bir 8085 mikrobilgisayar sisteminin FORTRAN derleyicisi, FORTRAN kaynak programlarını 8085 makine koduna dönüştürür.
3 Bir Motorola 6809 mikrobilgiyasarının FORTRAN derleyicisi, FORTRAN kaynak programlarını 6809 makine koduna dönüştürür, vb.

Yazılım geliştirmede gerekli adımlar

Bir iş, algoritma olarak formüllendirildikten ve istenen fonksiyonları yerine getirecek bir program yazıldıktan sonra, hâlâ programın tamamlanması için atılması gereken birkaç adım vardır. Bunlar:

1 Program, mikrobilgisayar sistemine girilmeli,
2 Program düzenlenmeli,
3 Program, çevirici ya da derleyici aracılığıyla, makine koduna dönüştürülmeli,
4 Program, gereken herhangi bir kütüphane altyordamına *bağlanmalıdır* (bkz: Kısım 5.2). Bu bağlantı, kütüphaneden altyordamı çekmeyi, onu programa yerleştirmeyi ve değiş-tokuş edilecek veri ve parametreler için uygun düzenlemeleri yapmayı gerektirir.

Programın mikrobilgisayara girilmesi, normal olarak, bir klavye ve görüntüleme birimi (ekran) kullanılarak gerçekleştirilir. Program deyimleri, makinedeki bir program ile birlikte bu birim kullanılarak belleğe yazılır.

Düzenleyici program, program istenen biçimde görev yapmazsa, içerdiği hataları düzeltmek ya da gördüğü işlevi değiştirmek amacıyla, programcının program deyimlerini değiştirmesine olanak tanır.

Yazılım geliştirme yardımcıları

Programcıya, sistem belirtiminin makine dili program koduna dönüştü
işleminde, yardım edecek birçok yardımcı vardır. Bunlar, düzenleyiciler, y
ciler, çeviriciler, HLL derleyicileri, çapraz yazılımlar, izleme programları, y
cı programlar, kütüphaneler ve hata ayıklayıcı programlardır.

Yukarıda da ifade edildiği gibi, *düzenleyiciler*, kullanıcının programını yazm
düzeltmesine ve gerektiğinde değiştirmesine yardımcı olan programlardır.

Yükleyiciler, programları, (kağıt bant, manyetik teyp, disket gibi) uygun
ortamından okumak ve bunları bellekte uygun bir konuma yerleştirmek için
nılır. Genellikle, bir program birçok parçadan oluşmuştur. Ayrı modüllerin
araya getirilmesinde gerekli olduğundan yükleme işlemi, bunları bilgisayara
mak, uygun bir yerde saklama alanı ayırmak ve adresler için bir takım ayarlam
yapmak, vb. işlemlerden ibarettir.

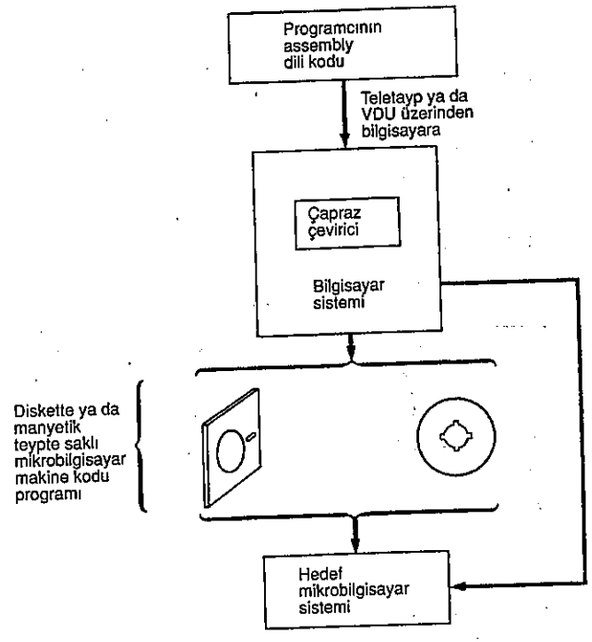
*Çevirici ve Yüksek Düzeyli Dil derleyicileri*nden daha önce söz edilmişti.

Çapraz yazılımlar, bir mikrobilgisayarda çalıştırılabilen ve bir diğeri için program
lar üretebilen programlardır. Örneğin, bir assembly dili programının, diğer bi
çevirici programı tarafından bir giriş verisi olarak okunduğunu görmüştük.
Çevirici, assembly dili komutlarını makine koduna çevirir. Bu işlemde, çevirici
sonuçta üretilecek olan makine kodlu programı çalıştıracak mikroişlemciyle, ayn
tip mikroişlemci üzerinde çalıştırılmasına gerek yoktur. Örneğin, çevirme işlemi,
bir PDP 11 bilgisayar sisteminde gerçekleştirilebilir ve bir Z80 mikrobilgisayar
için makine kodu programları üretebilir. Bunların tümü:

(a) PDP 11 koduyla yazılan çevirici programını
(b) Çevirici tarafından üretilen Z80 kodunun, bunu çalıştıracak olan Z80 mikro-bilgisayarına aktarımı için bir yöntemin bulunmasını gerektirir.

Bir makinede çalıştırılan ve bir diğeri için yazılım kodu üreten bu tip bir programa *çapraz-yazılım* denir. Assembly dilini makine koduna dönüştüren bir program ise *çapraz-çevirici*'dir. Eğer bu program bir Yüksek Düzeyli Dili makine koduna çeviriyorsa, *çapraz-derleyici* adını alır.

Bir çapraz-çevirici tarafından üretilen progam kodunun, bu kodu kullanacak olan mikrobilgisayar sistemine aktarılabileceği çeşitli yollar vardır (sözü edilen bu ikinci



Şekil 10.3

sistem genellikle *hedef* sistem olarak anılır). Örneğin bu, çapraz-çeviriciyi çalıştıran makine tarafından diskete yazılabilir veya manyetik teybe saklanabilir. Bu durumda hedef makine, programı alabilmek için uyumlu bir bant okuyucusuna veya disket sürücüsüne ihtiyaç duyacaktır.

Buna alternatif, *aşağı-uca-yükleme (down-line loading)* yöntemini kullanmaktır. Şekil 10.3'de, bu yöntemle disket ya da manyetik teyp kullanan yöntemin bir karşılaştırması gösterilmektedir.

Aşağı-uca-yükleme yönteminde, çapraz çeviriciyi kullanan bilgisayar sistemi, doğrudan hedef sisteme bağlanır. Bağlantı, genellikle 9. Bölümde tartışılan tipte bir seri asenkron bağlantıdır.

Program kodu, bu bağlantı üzerinden doğruca hedef sistem belleğine gönderilir. Hızlı ve kullanışlı bir yöntemdir.

Monitör (izleme) programlarından 1. Bölümde kısaca söz edilmişti.

Bir izleme programı, temel makine fonksiyonlarını denetlemek için norma mikrobilgisayarda yerleşik halde bulunan programdır. Basit bir sistemde, izle programı oldukça küçük ve basittir ve yalnızca sistem işleyişine temel olan ver programların giriş ve çıkış özellikleri ve bir programın yürütümünün başlatıl biçimi vb. fonksiyonları içerir.

Disk bellekli daha büyük mikrobilgisayarlarda, izleme programı daha büyük bo yutta ve karmaşık olur ve bir işletim sistemi olarak bilinir. Bu tür sistemler disk(ler) üzerindeki saklama alanı ayırma işlemini düzenleyen *işletim siste* içinde bir dosya yönetim bölümüne gerek duyulmaktadır.

Yardımcı programlar : İzleme programları ya da işletim sistemleri, genellikle kend işlemleri için gereken ve sistem kullanıcıları tarafından yazılan programlarla bir likte kullanılabilen program kodu parçaları içerirler. Birkaç örnekle vurgulamak gerekirse:

(a) Karakterleri ASCII kodda okuyan
(b) ASCII kodda çıkışa veren
(c) ASCII kodu ikili biçime dönüştüren
(d) İkili bir sayıyı ASCII karakterlere dönüştüren
(e) Bir metin dizisi basan
(f) Ondalığı on altılı koda çeviren
(g) Onaltılıyı on dalık koda çeviren

program bölümlerine genellikle program içerisinde sık sık ihtiyaç duyulur. Bunlar işletim sisteminde bulunduğundan kullanıcının bunlardan yararlanması oldukça kolaydır ve bu sık sık yapılır. Bunlar *yardımcı program* olarak adlandırılırlar.

Hata ayıklama programları, kullanıcının programındaki hataları bulmasına yar dımcı olur. Programlar ilk kez çalıştırıldıklarında, hata yapmaları kaçınılmazdır. Hem programın kuruluş mantığında, hem bu mantığa dayalı olarak türetilen algo ritmada ve hem de algoritmanın kodlanış biçiminde hatalar bulunabilir. Program istenen biçimde çalıştırılmadan önce, bu hataların belirlenmesi, ayıklanması ve düzeltilmesi gereklidir.

Hataları bulmak için kullanılan teknikler:

1 Programın seçilen kısımlarını çalıştırmak, yani programda durdurma noktaları belirlemek. Bir durdurma noktası, programda yürütümün durdurulduğu bir noktadır. Böylece, program bu noktaya kadar çalıştırılabilir ve işlemin doğru olup olmadığını görmek için, elde edilen ara sonuçlar incelenebilir.

2 Ara program sonuçlarının incelenebilmesi için, hata ayıklama programı, sistemin seçilen kısımlarının içeriklerinin görüntülenmesine izin verir. Bunlar işlemci kaydedicilerinin ya da bellekteki konumların içerikleri olabilir.
3 Bilinen test verilerini kullanarak programın çalıştırılabilmesi için, seçilen bellek konumlarının ya da işlemci kaydedicilerinin içerikleri, hata ayıklama programı kullanılarak yeniden düzenlenebilir.
4 Kullanıcı programının yürütümü belli bir adresten başlatılabilir.

Burada tümünden söz edilmesi mümkün olmayacak kadar çok başka olanaklar da vardır. Bunlar hep birlikte, program hatalarının etkin biçimde düzeltilebilmesi olanağını sağlarlar.

Özet

Önceki bölümlerde verilen çok kısa açıklamalardan da görülebileceği gibi, sonuçta çalışan prototip makinenin belleğindeki makine kodlu program ile sistem yazılım belirtimi arasındaki boşluğu kapatmakta programcıya yardımcı olabilecek pek çok gereç vardır. Bu gereçlerin her biri yalnızca en genel yönleriyle ele alınmıştır. Diğer ayrıntılar, özellikle bu konu başlıklarını işleyen metinlere bırakılmıştır.

Programcıya yardımcı olan yazılım geliştirme gereçleri gibi, mühendislere de sistem devrelerini kurmada yardımcı olan yardımcı donanım birimleri bulunmaktadır. Yine, bunları tartışmak için yerimiz yeterli değildir; ancak, mantık durum analizleyicisi gibi test araçları ile birlikte imalatçının mikrobilgisayar geliştirme sistemini kullanmanın, en kapsamlı ve uygun yaklaşım yöntemini sağladığına dikkat edilmesinde yarar vardır. S 10.2

10.3 Assembly dili

Bu kitaptaki programlama örnekleri makine dilinde ya da assembly dilinde yazılmış olup, bunlardan assembly dili hakkında özet bir bilgi Kısım 1.7'de verilmişti. Mikrobilgisayarın kullanımına ilişkin bir örneğe geçmeden önce, assembly dili tanımına ilişkin biraz daha ayrıntıya girmek uygun olacaktır.

Daha önce anlatılanları toparlarsak, assembly dili:

1 Komutların adlandırılmasına,
2 Komutların komut-kodu kısımlarının bir anımsatıcı biçiminde yazılmasına,
3 Değişkenlerin isimlendirilmesine imkan verir.

Genelde, bir assembly dili deyiminin formatı, yani assembly dilinin s
şöyledir:

Etiket alanı	*Komut (veya işkodu) alanı*	*İşlenen alanı*	*Açıklama alanı*

Örneğin bir komut aşağıda verildiği gibi olabilir:

```
START:   MOV     D,C          ; Veriyi C'den D'ye taşı
```

olabilir. İlk alan, komutu ifade etmek için kullanılan adı içerir. Bu nedenle
örnekte, denetimi bu komuta aktarmak için bir 'JMP START' komutuna ya
benzeri bir komuta ihtiyaç olacaktır.

Komutun 2. ve 3. alanları ise komutun amacını-komutun ne yapacağını (işkodunu
artı kullanılan işlenen ya da işlenenlerin neler yapacaklarını içerir. Verilen örne
C kaydedicisinde tutulan verinin, D kaydedicisine yerleştirilmesine neden olur.

4. alan ise, programcının, genellikle komutun ne yapacağını ya da ne yapmak
istediğini açıklayan bir açıklama eklemesine olanak tanır. Açıklama alanının
kullanılması zorunlu değildir (boş bırakılabilir), ancak programın özellikle başka
kişilerce kolaylıkla anlaşılabilmesi isteniyorsa, kullanılmasında yarar vardır.

Çevirici Yönergeleri

Komutların, yukarıda tanımlanan formatta yazılabilmesinin yanında, assembly dili
ayrıca, bilgileri çevirici programının kendisine aktarma özelliklerini de içerir.
Bunlara *çevirici yönergeleri* ya da bazen *sözde-komutlar* denir.

Çeviricinin gerek duyabileceği bilgiler:

1 Makine koduna dönüştürüldüğünde programın başlayacağı bellek adresi.
2 Çeviri işleminin yürütüldüğü sırada programda yerleştirilen değerlerdir.

Örneğin, Intel 8080 veya 8085 işlemcilerinin assembly dili şu çeviricinin yönerge-
lerini içerir:

ORG - Programın başlangıç adresini ayarlamak için kullanılır.
EQU - Bir değişken adına değer atamak için kullanılır.

DB - Baytı tanımlar. Bir sonraki kullanılabilen bayta bir değer atamak için kullanılır.
DW - Sözcüğü tanımlar. Bir sonraki kullanılabilen iki bayta bir değer atamak için kullanılır.

Bu yönergelerin, çevirici dili programlarında bulunmalarına karşın, sonuçta oluş-turulan makine kodu programında herhangi bir komut üretmediklerine dikkat edilmelidir. Bunlar, çevirici programının kendisine girilen komutlardır. Bu nedenle bazen sözde-komutlar olarak da anılırlar.

10.4 Programlama Örneği: Bir Kronometre

Giriş

Elektronik bir kronometre yapmak için bir mikrobilgisayar sisteminin kullanılaca-ğını varsayalım. Bu tip birimler bir süredir piyasada mevcuttur ve birçok olanak sunarlar.

Temelde kronometreler, belli bir olayın sürdüğü zamanın doğru olarak ölçülebil-mesinde kullanılırlar. Bu olay, bir atletizm karşılaşmasındaki bir koşunun süresi, fabrikada parça üreten bir hidrolik presin çalışması, bir dersin süresi ya da birçok durumdan biri olabilir.

Bunların her birinde temel koşul, operatörün kronometreyi, olayın tam başında başlatabilmesi ve sonunda durdurabilmesidir. Saat (kronometre), bu iki olay ara-sında geçen zamanı ölçer.

Kuşkusuz, temel yapının birçok olası varyasyonu vardır. Örneğin, çoğu zaman bir çok olayı birbiri ardı sıra ölçmek gerekebilir. Başlangıçta, kronometre sıfırdan başlar. İlk olay bittikten sonra, o olayın aldığı süreyi tutar (ve görüntüler). Daha sonra şu iki biçimden birinde çalışır.

1 İkinci olayın zamanını başlatmak için 'başlatma' düğmesine basıldığında, daha önce olduğu gibi sıfırdan başlayabilir.
2 Ya da, 'başlatma' düğmesine ikinci kez basıldığında, ilk olayın bitimindeki değerden başlayabilir.

Bunlardan ilki bir olaylar dizisinin aldığı sürelerin ayrı ayrı ölçülmesine imkan tanırken, ikincisi bu olaylar dizisinin aldığı toplam süreyi verir. Kuşkusuz, her iki durumda da, değerlerde bir aritmetik işlem yapmak suretiyle, bir sonuçtan diğerini

elde etmek mümkündür. Bununla beraber, eğer kronometre her iki modda çalışabiliyorsa daha yararlı olacaktır.

Şimdi, 'tur süresi' kavramı ile konunun biraz daha derinine inelim. Adından anlaşılacağı gibi bu, çok turlu bir koşunun turlarının ayrı ayrı zaman ölçümlerine izin verir. Bununla, kronometrenin ekranı, bir turun sonunda durdurulabilir, ancak kronometre çalışmaya devam eder. Böylece operatör, bir sonraki turun süresini bozmadan, her turun süresini ayrı ayrı kaydedebilir.

Tur süresi özelliği olmayan ancak:

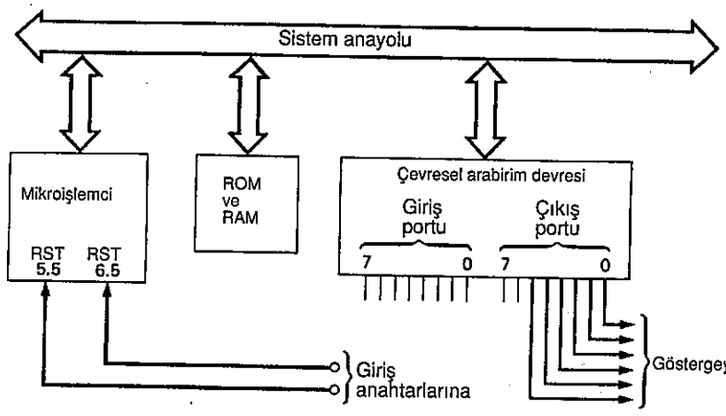
1 Her bir zamanlama aralığının sonunda kronometreyi sıfırlayabilecek ya da
2 Bir önceki değerden çalışmaya devam edebilecek bir kronometreyi ele alalım.

Donanım

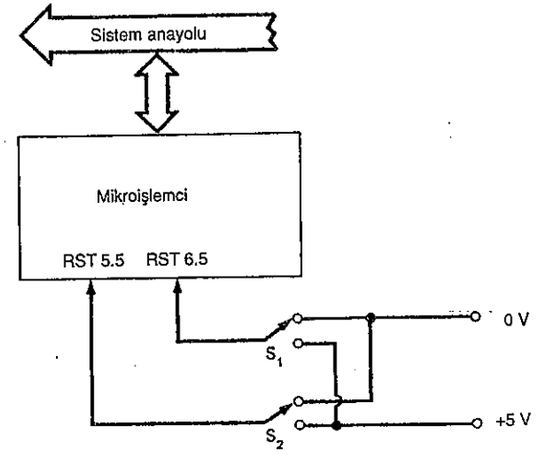
Olası bir mikrobilgisayar sisteminin blok diyagramı Şekil 10.4'de gösterilmektedir. Bunun, Şekil 7.10'da gösterilen, karedalga darbe dizisi üreten sistemle olan benzerliği, birkaç önemli farklılık olmasına rağmen, açıktır.

Bir mikroişlemci ile kronometredeki değeri gösteren bir sıvı-kristalli gösterge arasındaki bağlantıyı sağlamak için, bir çevresel arabirim devresi kullanılır. Kronometreye girişler, iki yay-yüklemeli tuş aracılığıyla yapılır ve doğrudan doğruya işlemcinin RST 5.5 ve RST 65 kesme girişlerine uygulanır.

Bu noktada, eğer pratikte bu uygulama için bir Intel 8085 mikrobilgisayar kullanı-



Şekil 10.4



Şekil 10.5

lacaksa, ayrıca bir çevresel arabirim devresi kullanmaya gerek olmayacağını belirtmekte yarar vardır. Temelde bir 8085 üç yongadan oluşur. Bunlar genelde: işlemci, bir ROM yongası ve bir RAM yongasıdır.

Bununla beraber, her bir bellek yongası, saklama işlevlerinin yanı sıra başka olanaklara da sahiptir. Örneğin, her ikisi de G/Ç portları içerir ve bunlar, gösterilen fazladan çevresel arabirim devresi yerine, görüntü çıkışlarını sürmek için kullanılabilir.

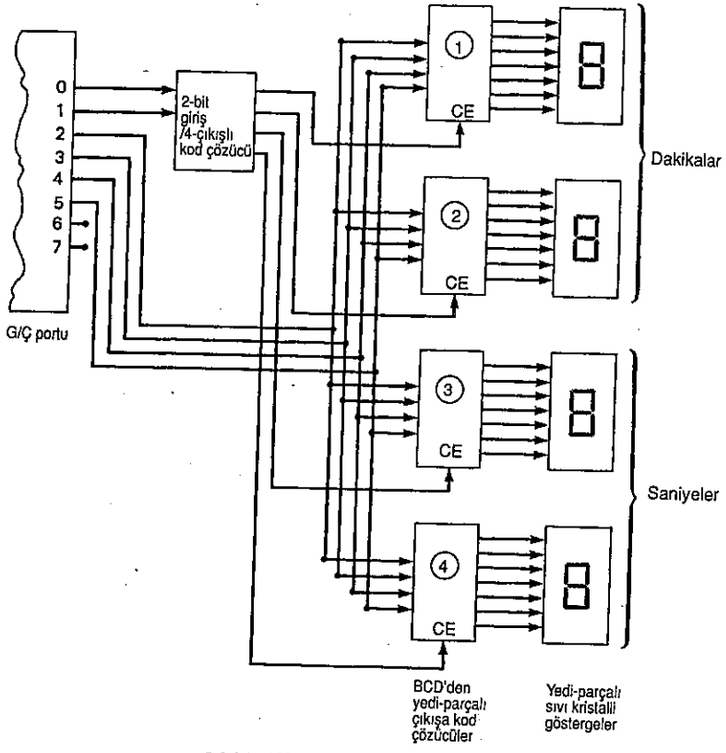
Şekil 10.5'de mikrobilgisayarın giriş devreleri gösterilmiştir. S_1 ve S_2 olarak adlandırılan iki anahtar, RST 5.5 ve RST 6.5'e bağlanır. Bu kesmeler bir yüksek (+5 volt) düzeyi tarafından tetiklenir. Bu nedenle, RST 6.5'i +5 volta bağlayan S1 anahtarına basılırsa, kesme maskeleme düzeyleri doğru olarak ayarlanır ve kesmeler yetkilenirse, program işleyişinin denetimi 0034 (on altılı) adresine aktarılır. Benzer biçimde, S_2 anahtarına basmak, denetimin 002C (on altılı) adresine aktarılmasına neden olarak RST 5.5'dan bir kesme üretir. S_1 anahtarı, kronometreyi çalıştırmak ve durdurmak için kullanılacaktır. Kronometrenin durdurulduğunu ve ölçülen zaman aralığını ekranında gösteriyor olduğunu düşünelim. S_1'e basılması, kronometrenin görüntülenen değerden tekrar saymaya başlamasına neden olur. Tekrar S_1'e basılırsa kronometre durdurulur. Üçüncü bir basışta kronometre tekrar çalışmaya başlar ve bu şekilde zamanlama fonksiyonunu bir çalıştırıp bir durdurarak devam edilir.

Kronometre her durdurulduğunda, daha önce ölçülen aralıkların toplam süresini gösterir. Ancak S_2 anahtarı, kronometreyi sıfırlamak için kullanılabilir.

Bu nedenle önce S_2'ye, ardından da S1'e basmak kronometreyi sıfırdan ba
Ayrı ayrı zaman aralıklarını ölçmek için:

1 S_2'nin ardından süreyi başlatmak için S_1'e basılır.
2 Süreyi durdurmak için S_1'e basılır. Bu durumda, ilk aralığın süresi görünt
ve bir yere not edilebilir. Bunun ardından:
3 Yine S_2'nin ardından tekrar S_1'e basılır. Böylece 2. aralığın süresi başlar
sürenin sonunda tekrar S_1'e basılır ve 2. aralığın değeri not edilir.
4 Bu işlem istenildiği kadar tekrar edilir. Kronometre, geçen zamanı saniye
dakika olarak ölçer. 99 dak. ve 59 sn.'ye kadar herhangi bir zaman aralığı
geçerli olabileceği varsayılmıştır. Bu sınırlama, kullanılan çıkış ekranı sad
dört ondalık karakteri gösterebildiğinden gereklidir. Ekrandaki basamakla
ikisi dakikalar, diğer ikisi de saniyeler için kullanılır.

Ölçülen zaman yukarıda verilen maksimum aralığı aşarsa, kronometrede taşma olu
ve kronometre tekrar sıfırdan başlar, yani 99 dak. 59 sn'den bir saniye sonra
gösterilen zaman 00 dak. 00 sn. olacaktır. Şekil 10.6'da, görüntüleme birimlerinin



Şekil 10.6 Kronometrenin çıkış göstergeleri

Tablo 10.1

Ondalık	*BCD gösterimi*
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

mikrobilgisayara nasıl bağlanabileceği gösterilmiştir. Şekil 10.4'den de görülebileceği gibi, çıkış portunun altı hattı kullanılır.

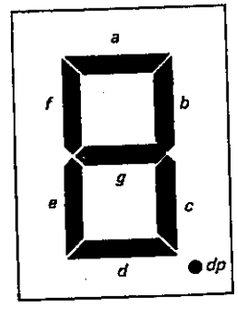
Görüntülenecek sayılar, bilgisayardan ikili kodlanmış ondalık (BCD) karakterler olarak çıkarlar. Bu nedenle, her bir karakter dört bitten oluşmuştur. Referans olarak, Tablo 10.1'de, 0'dan 9'a kadar olan sayıların BCD eşdeğerleri verilmiştir.

Çıkış portunun altı hattından dördü, ekrana çıkılan sayının BCD kodunu göndermek için kullanılır. Diğer iki hat ise, göstergeye hangi basamağının gönderileceğini belirten bir kodu iletmek için kullanılır. Bu dört basamak şunlardır:

1 Dakikaların en büyük değerlikli basamağı
2 Dakikaların en küçük değerlikli basamağı
3 Saniyelerin en büyük değerlikli basamağı
4 Saniyelerin en küçük değerlikli basamağı

Bunlar, portun 2'den 5'e kadar numaralandırılan dört hattında, (1) ardından (2) ardından (3) ve ardından (4) sırasını izleyen çıkışlardır.

Çıkışa verilen BCD kodları, paralel olarak tüm dört BCD yedi-parçalı kod çözücü yongalarına girer. Bununla birlikte, bunlardan yalnızca biri her bir basamağı alır. Porttan (1) gönderildiğinde, ardından 00 kodu da 0 ve 1 port çıkışlarına gönderilir. Bu iki bit, 2-bit giriş/4 hat çıkışlı kod çözücü yongasına uygulanır ve bu, 1 numaralı BCD yedi-parçalı kod çözücünün yonga yetkileme (CE) girişini etkinleştirir. Diğer üç kod çözücünün yonga yetkileme girişleri çalışmaz. Bu nedenle, (her ikisi de dahil olmak üzere) 2'den 5'e kadar olan hatlardaki kod, yalnızca 1 No'lu BCD yedi-parçalı kod çözücüsüne okunur.



Şekil 10.7 Yedi-parçalı sıvı-kristalli gösterge

2-5 hatlarının çıkışında (2) olduğunda, 01 kodu, (2)'nin 2. kod çözücüye saklanmasına neden olarak 0 ve 1 hatlarına gönderilir.

Çıkışta (3) olduğunda, 10 kodu; çıkışta (4) olduğunda 11 kodu,

0 ve 1 hatlarına gönderilir.

Böylece, porttan gönderilen karakterler, sırayla 1., 2., 3. ve 4. kod çözücü yerleştirilir.

Çıkış sayıları sıvı kristalli göstergelerde verilir. Bunların her biri, Şekil 10.7'de gösterildiği gibi, bir ondalık nokta ile birlikte yedi tane parçadan oluşan bir rakamı gösterebilir.

Belli bir rakamı gösterebilmek içinse, yalnızca gereken parçalar etkinleştirilir. Bu nedenle, 'sekiz' rakamı tüm parçaları; 'beş' rakamı *a,f, g, c* ve *d*'yi; 'bir' *b* ve *c*'yi; 'altı' *a, f, g, e, c* ve *d*'yi kullanır. Benzer şekilde diğer sayılar üretilir.

BCD yedi-parçalı kod çözücüleri yalnızca BCD giriş sayısının kodunu çözer ve ekrandaki gerekli parçaları etkinleştirecek sinyalleri üretirler.

Böylece özetlersek, bilgisayarın, ekranı güncellemek için gerek duyduğu şeylerin tümü, hangi basamağın hangisi olduğunu belirten 2-bitlik doğru kodlarla birlikte, dakikaları ve saniyeleri gösteren dört basamağı doğru sırada göndermektedir. Bunun için, dört OUT işlemine gerek vardır.

Kronometrenin çalışma modu ve donanımı tanımlanmış bulunmaktadır. Geriye kalan, bu donanımı denetleyen yazılımı (programları) hazırlamaktır.

Yazılım

Görüleceği gibi yazılım, birkaç ayrı bölümden oluşmaktadır. Bunlar, örneğin:

1 Kronometrenin tüm çalışmalarını belirlemek ve diğer programları koordine etmekten sorumlu bir ana program,
2 Temel zaman tutma işlevini gerçekleştiren bir GECIK alt yordamı,
3 S_1'e basıldığında meydana gelen kesme ile ilgilenmek üzere bir kesme yönetim yordamı,
4 S_2'ye basıldığında ortaya çıkan kesme ile ilgilenmek üzere bir kesme yönetim yordamı,
5 Ekranı yönetmek ve güncellemek için bir çıkış alt yordamı,
6 Mikrobilgisayarda ikili biçimde saklı değerleri ekranda görüntülemek üzere, BCD biçimine dönüştürecek bir yordam gerekmektedir.

Ana program

Şekil 10.8'de, kronometre ana programının akış diyagramı gösterilmektedir. Kronometrenin faaliyetleri, bu diyagramda gösterilen DEVAM adlı bir parametre ile kontrol edilir. DEVAM bire ayarlandığında kronometre çalışır. DEVAM sıfıra ayarlandığında, kronometre çalışmayı durdurur ve ulaşmış olduğu zaman değerini göstererek döngüye girer.

Böylece, baştan itibaren akış diyagramını izleyerek, kronometre önce kullanıma hazırlanır ve DEVAM sıfıra kurulur. Kullanıma hazırlama bölümü, kesme maskeleme kaydedici ve yığın işaretçisini doğru değerlerine kurar, çevresel arabirim devresi portlarının durumunu ayarlar ve daha sonra kullanılacak gecikme parametresinin değerini yerleştirir. Ayrıca, kronometre açıldığında, sürenin sıfırdan başlayabilmesi için, DEVAM'ı sıfırlar.

Kullanıma hazırlanmasının ardından program, dakika ve saniye değerlerinin çıkışından başka herhangi bir işlemin yapılmadığı döngüye girer.

DEVAM 1'e ayarlanır ayarlanmaz (ve bu işlem, S_1'e basıldığında girilen kesme hizmet yordamı tarafından yapılır yapılmaz), kronometre ana zamanlama moduna girer.

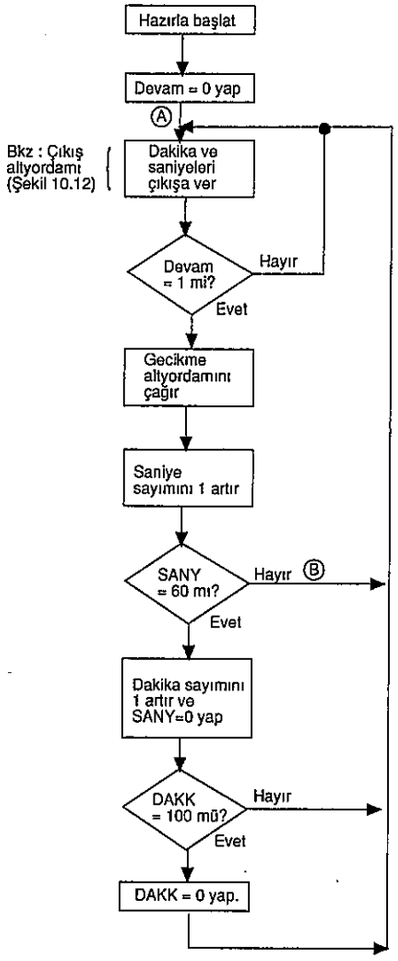
Böylece, Şekil 10.8'de gösterildiği gibi programı, hassas biçimde tanımlanmış bir süreyle bekleten bir gecikme altyordamını çağırır. Bunun ardından, SANY'yi 1 artırır.

Kronometre ile ölçülen aralığın değeri, (saniye değerlerini tutan) SANY ve (dakika değerlerini tutan) DAKK adlı iki değişkende tutulur.

SANY'nin her artımında, 60'a ulaşıp ulaşmadığını görmek için test edilmesi gerekir. Ulaşmışsa 1 dakika geçmiştir ve DAKK sayacı 1 artırılır. Aynı anda SANY sıfırlanır.

Benzer biçimde, DAKK da, her artırıldığında daha önce belirlenen maksimum değeri aşıp aşmadığını görmek için test edilir. Eğer DAKK 100'e ulaşmışsa, DAKK ve SANY'nin her ikisi de sıfırlanır.

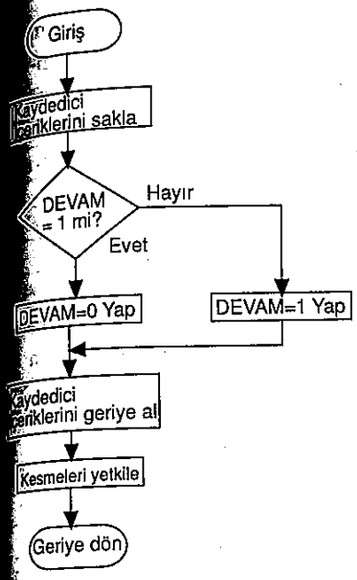
Kronometrenin zaman ölçümünün doğruluğu, hassas bir biçimde, döngünün (A)'dan (B)'ye ve tekrar A'ya dönüşünde aldığı süreye bağlıdır. Bu süre tam olarak 1 saniye olmalıdır ve GECIK alt yordamıyla sağlanan gecikme, bu değeri verinceye kadar ayarlanmalıdır.



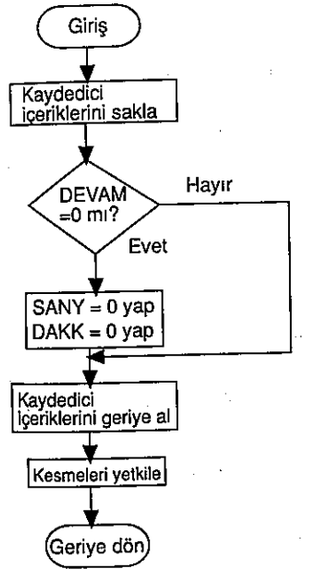
Şekil 10.8 Kronometre ana programı

Gecikme, bu döngünün yürütmesi 1 saniye alacak şekilde ayarlandığında, SANY sayacı her saniyede bir ve buna bağlı olarak, DAKK sayacı da her dakikada bir artırılır. Böylece, kronometre geçen sürenin doğru bir ölçümünü tutmuş olur.

Kronometre, SANY ve DAKK'i birer artırmaya ve buna göre DEVAM sıfırlanana



Şekil 10.9 S_1 anahtarı kesmesi için kesme hizmet yordamı.



Şekil 10.10 S_2 anahtarı kesmesi için kesme hizmet yordamı

kadar göstergeyi güncellemeye devam eder. Bu işlem, aynı zamanda S_1'e basıldığında çağrılan kesme hizmet yordamı tarafından da gerçekleştirilir. Bu nedenle, kronometre durdurulursa, S_1'e basmak, DEVAM'a 1 vererek yeniden çalışmasına neden olur. Kronometre çalışıyorsa, S_1'e basmak, DEVAM'a 0 değeri atayarak durmasına neden olur.

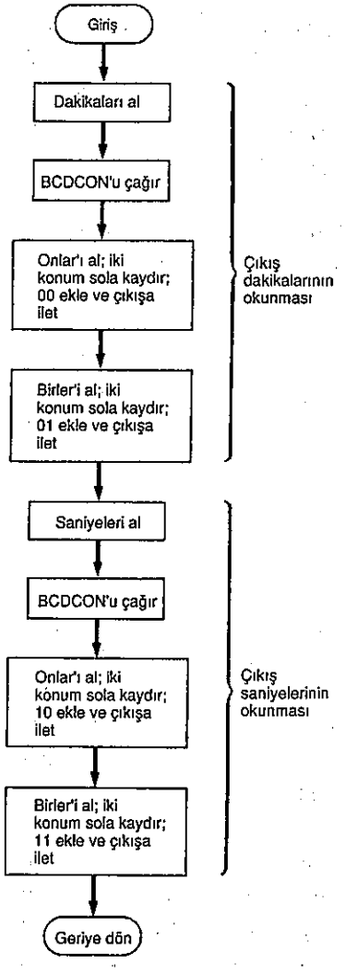
Kesme Hizmet Yordamları

S_1'e ilişkin kesme hizmet yordamı, Şekil 10.9'da bir akış diyagramı olarak gösterilmiştir. Bu yordamın çalışması çok basittir. Eğer DEVAM girişte 1 ise, altyordam DEVAM'ı sıfırlar, girişte 0 ise 1'e kurar.

Bir kesme hizmet yordamına girişteki olağan işlemler uyarınca, Şekil 10.9'da, yordamın başında saklanan ve yordamdan çıkmadan önce tekrar geriye alınan kaydedici gösterilmiştir. İleride göreceğimiz gibi, pratikte, bu kaydedicilerin programda nasıl kullanıldığının dikkatlice seçimi, bu yordama girişte bunlardan herhangi birinin saklanma zorunluluğunu ortadan kaldıracaktır. Ancak yine de akış diyagramına, normalde kaydedici saklanmasına gerek duyulacağının belirtildiği bir işlem adımı da eklenmiştir.

S_2'ye basıldığında meydana gelen kesme hizmet yordamı Şekil 10.10'da gösterilmektedir. Bu yordam ilk olarak DEVAM'ın '0' durumunda olup olmadığına bakar. Eğer değilse, o halde '1' durumundadır ve dolasıyla kronometre çalışıyordur. Bu durumda, yordam başka herhangi bir işlem yapmaksızın doğrudan ana programa döndüğünden S_2'ye basmanın herhangi bir etkisi yoktur.

Ancak, DEVAM '0' ise, kronometre durmuş demektir. Bu durumda, S_2 kesme hizmet yordamı, SANY ve DAKK'nin sıfıra ayarlanmasına neden olur. Kronometre böylece sıfırlanır ve S_1'e tekrar basıldığında, sürenin ölçümü sıfır değerinden başlar.



Şekil 10.11 Çıkış altyordamı

Gecikme altyordamı

Bir önceki kısımda sözü geçen gecikme alt yordamı, Kısım 7.5'de gösterilen olan programa benzer. 7. Bölümde açıklandığı gibi, alt yordam tarafından üretilen gecikme değeri, giriş-işlemcinin C kaydedicisi kullanılmak suretiyle altyordama aktarılan bir parametre ile belirlenir.

Çıkış altyordamı

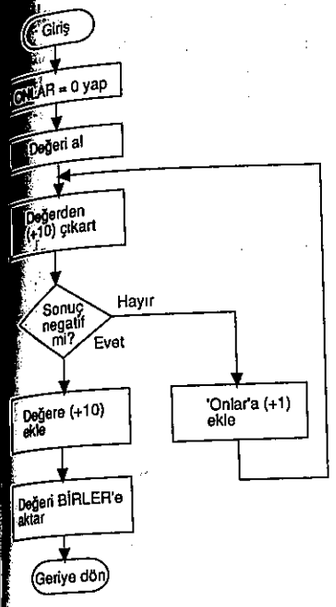
Daha önce açıklandığı gibi, dakika ve saniye değerlerini gösteren dört BCD karakteri göstergelere, aşağıda belirtilen sırada gönderilir:

1 Dakikalar (en büyük değerlikli karakter)
2 Dakikalar (en küçük değerlikli karakter)
3 Saniyeler (en büyük değerlikli karakter)
4 Saniyeler (en küçük değerlikli karakter)

DAKK'de tutulan dakika sayısı ve SANY'de tutulan saniye sayısı ikili biçimdedir. Bu nedenle çıkışa verilmeden önce, bunların BCD koduna dönüştürülmesi gerekir. Bu, biraz sonra anlatılacak olan (BCD DÖNÜŞTÜRME altyordamının kısaltması olarak BCDDON şeklinde bilinen) ikili-BCD dönüşümü alt yordamı ile gerçekleştirilir.

Çıkış alt yordamı Şekil 10.11'de gösterilmektedir. Bu alt yordama giriş büyüklükleri DAKK ve SANY değerleridir. Az önce anlatıldığı gibi, önce DAKK'nin ve ardından da SANY'nin değeri göstergeye çıkılır.

Altyordam önce BCDDON'u çağırır ve DAKK'in ikili değerini DAKK alt yordamına aktarır. BCDDON bunu iki değere; 'onlar' ve 'birler' hanelerine ilişkin BCD karakterlerine dönüştürür. Örneğin, (ondalık 54 değerini gösteren) 00110110 ikili değeri, iki BCD karakterleri olan 0101 ve 0100'a dönüştürülür. 'Onlar' hanesinde görüntülenecek karakter, 0101 (5) ve 'birler' hanesinde görüntülenecek karakter ise 0100 (4)'dür.



Şekil 10.12 İkiliden İkili-kodlanmış-ondalık'a dönüştürme altyordamı (BCDCON)

Çıkış alt yordamı, dakikalar için 'onlar' ve 'birler' değerlerine sahip olduğunda bunları çıkışa verir. Bunu yapmak için, ilk olarak 'onlar' karakterini alır, çıkış portuna yapılan bağlantıların gerektirdiği gibi (Şekil 10.6), bu karakteri 2, 3, 4 ve 5. bit konumlarına almak için sola doğru iki basamak kaydırır ve çıkış kod çözücüsüne tanıtmak için en alttaki iki bit konumunda ona 00 ekler. Ardından çıkışa verir.

Dakikaların 'birler' karakteri de, 0 ve 1 bit konumlarında kendisine 01 kodunun eklenmesinin dışında benzer biçimde işlem görür.

İki DAKK karakterinin çıkışa verilmesinin ardından, aynı şekilde SANY değeri de işleme alınır. SANY'nin 'onlar' değeri 10 kodu ile, ardından 'birler' değeri 11 kodu ile çıkışa verilir.

İkili-BCD dönüşümü alt yordamı

BCDCON adlı ikili-BCD dönüştürme alt yordamı, Şekil 10.12'de, gösterilmiştir. Çalışma yöntemi açıktır. Akış diyagramındaki 'DEGER' adlı parametreden bir ikili giriş sayısını alır ve bir döngü yaparak, kaç tane 'onlar' içerdiğini belirler. Her döngüleme sırasında, DEGER'den 10 (ondalık) çıkartılır ve ONLAR değişkenine (+1) eklenir.

Her döngülemede, çıkartma sonucunun negatif olup olmadığının testi yapılır. Negatif değilse işlem devam eder. Negatif ise, fazladan bir '10' çıkartılmıştır. Bu nedenle, DEGER'e 10 tekrar eklenir. Sonuç, DEGER'in 'birler' basamağıdır ve çağıran programa aktarmak için ONLAR değişkenine yerleştirilir.

Parametre aktarma

Program iki kesme yönetim yordamı ve üç alt yordam içermektedir. Aşağıda açıklandığı gibi çeşitli parametre değerlerinin bu alt yordamlarla değiş-tokuş edilmesi gerekir.

1 DEVAM, ana program tarafından kullanılan genel bir değişkendir ve kesme yönetim yordamları tarafından değiştirilir.
2 Gecikme altyordamlarına, istenen gecikmenin süresini belirten bir parametre değeri verilmelidir.
3 SANY ve DAKK, ana programda, S_2 kesme hizmet yordamı ve çıkış gecikme yordamında kullanılan genel değişkenlerdir.
4 Dönüştürülecek sayının değerinin BCDCON altyordamına aktarılması gerekli dir. Onlar ve birler değerleri, bu altyordamdan geriye, çağıran programa aktarılmalıdır.

Önceki bölümlerde açıklandığı gibi, bu parametrelerin mikrobilgisayarda tutulabileceği çeşitli yollar vardır. Örneğin, bu değerlerin her biri için bellek konumları ayrılabilir. Daha sonra bunlar, H,L kaydedici çiftinde tutulan adresler veya LDA ve STA komutları (bkz. Ek 1) kullanılarak, yeniden geriye getirilebilir ve yeniden yazılabilirler.

Ya da, parametreler işlemci kaydedicilerinde tutulabilir. Tabii ki, bunun olabilmesi için, parametrelerin sınırlı sayıda olması gerekir. Bununla birlikte, bu durumda ve parametre değerlerinin, kaydedicileri saklayıp geriye almaksızın kullanan alt-yordamlar tarafından bozulmamasına özen gösterdikçe, en uygun ve en hızlı yöntem budur.

Bu nedenle, parametrelerin aşağıdaki gibi tutulması önerilir:

1 DEVAM, B kaydedicisinde tutulur.
2 Gecikme parametresi, C kaydedicisindeki gecikme altyordamına aktarılır. Bu, 7. Bölümde kullanılmış olan yöntemin aynısıdır.
3 SANY, E kaydedicisinde, DAKK da, D kaydedicisinde tutulur.
4 DEGER, A kaydedicisini kullanan BCDCON'a aktarılır. 'Onlar' ve 'birler'in değerleri, H ve L kaydedicilerindeki çağıran programa geriye aktarılır.

Program kodu

Kronometre programını oluşturan yordamlar için önerilen bazı program listeleri bir sonraki kısımda verilmiştir. Bu kitaptaki diğer örneklerde olduğu gibi bunlar da, Intel 8085 anımsatıcıları kullanılarak yazılmıştır. Bu örneklerde:

1 Rakam ekranı için kullanılan çıkış portunun, bir PIC'in C portu olduğu ve daha önce olduğu gibi 0A (on altılı) adresinde olduğu varsayılmıştır.

2 Yığın, daha önce olduğu gibi, 10D4 (on altılı) adresinde kullanıma hazırlanır.
3 Komutlarda yazılan tüm sabitler ve adresler on altılı koddadır.
4 0034 adresinin (RST 6.5 kesmesi için kesme vektör adresi) JMP KESME1 komutunu içermesi gerekir.
5 002C adresinin (RST 5.5 kesmesi için kesme vektör adresi) JMP KESME2 komutunu içermesi gerekir.
6 Kullanılan gecikme altyordamlarının D ve E kaydedicilerindeki değerleri koruduğu ve B kaydedicisini kullanmadığı varsayılmıştır. Bu koşullar, 7. Bölümde anlatılan GECIK altyordamı tarafından yerine getirilir.

Ana program

Ana program komutları aşağıda verilmiştir:

```
        MVI A,  92      ;C portunu kullanıma hazırla
        OUT     0B
        LXI     SP, 10D4 ;Yığın işaretçisini kullanıma hazırla
        MVI     A, 0C   ;RST 5.5 ve RST 6.5 kesmelerini
        SIM             ;kullanıma hazırla
        MVI     C, XX   ;Gecikme parametresini XX'e ayarla
        MVI     B, 00   ;(B)'yi-DEVAM parametresini -
                        ;0'a ayarla

DONGU:  CALL    CIKIS   ;CIKIS altyordamını çağır.
        MOV     A, B    ;DEVAM=1 olup olmadığını test et. DEVAM=0 ise
        DCR     A       ;DONGU'ye atla
        JNZ     DONGU
        CALL    GECIK   ;GECIK alt yordamını çağır.
        INR     E       ;SANY'i 1 artır.
        MVI     A, 3C   ;SANY=60 mı? 60'ı not et.
        CMP     E       ;(ondalık) = 3C (on altılı)?
        JNZ     DONGU   ;Hayır. DONGU'ye atla
        INR     D       ;DAKK'yı 1 artır.
        MVI     E, 00   ;SANY = 0'a ayarla
        MVI     A, 64   ;SANY=100 mü? 100'ü not et.
        CMP     D       ;(ondalık) = 64 (on altılı)?
        JNZ     DONGU   ;Hayır. DONGU'ye atla
        MVI     D, 00   ;DAKK= 0'a ayarla
        JMP     DONGU   ;DONGU'ye atla
```

Daha önce açıklandığı gibi, gecikme süresi, döngüye tam 1 saniyelik bir gecikme verecek biçimde ayarlanmalıdır. Bunun tam değeri, listenin başından itibaren 3. komutta yazılı olan XX'in (X herhangi bir on altılı karakteri gösterir) yerini alır. Bu, ayar için kullanılır.

Kesme Hizmet Yordamları

S_1'den gelen kesme için kesme hizmet yordamına KESME1; S_2'den gelen kesme için kesme hizmet yordamına da KESME2 denir. KESME1 ve KESME2 komutları şöyledir:

```
KESME1:  MVI   A,01     ;DEVAM=1 olup olmadığını test et
         CMP   B
         JNZ   DONUST   ;Hayır. DONUST'e git
         MVI   B, 00    ;Evet. DEVAM = 0 yap.
         JMP   CIKIS
DONUST:  MVI   B, 01    ;DEVAM = 1 yap
CIKIS :  EI            ;Kesmeleri ekle
         RET           ;Geriye dön

KESME2:  MVI   A, 00    ;DEVAM=0 olup olmadığını test et
         CMP   B
         JNZ   DEVAMET  ;Hayır. DEVAMET'e git
         MVI   E, 00    ;Evet. SANY = 0 yap
         MVI   D, 00    ;Evet. DAKK = 0 yap
DEVAMET: EI            ;Kesmeleri erkle
         RET           ;Geriye dön
```

Çıkış altyordamı

Bu altyordama CIKIS denir ve komutları şöyledir:

```
CIKIS:  MOV   A, D     ;DAKK'in değerini A'ya koy
        CALL  BCDDON   ;Dönüştürme yordamını çağır
        MOV   A, H     ;ONLAR'ı A'ya koy
        RLC            ;Sola doğru iki kere kaydır
        RLC
        ANI   FC       ;11111100 deseni ile maskele
        ADI   00       ;00 ekle
        OUT   0A       ;C portuna çık
```



```
        MOV   A, L     ;A'ya BİRLER'i koy
        RLC            ;DAKK'in BİRLER'ini çık
        RLC
        ANI   FC
        ADI   01
        OUT   0A
        MOV   A, E     ;SANY'in değerini A'ya koy
        CALL  BCDCON   ;Dönüştürme yordamını çağır
        MOV   A, H     ;SANY'in ONLAR'ını çık
        RLC
        RLC
        ANI   FC
        ADI   02
        OUT   0A
        MOV   A, L     ;SANY'in BİRLER'ini çık
        RLC
        RLC
        ANI   FC
        ADI   03
        OUT   0A
        RET            ;Geriye dön
```

İkili-BCD dönüştürme altyordamı

Buna BCDCON denir ve komutları şöyledir:

```
BCDCON:  MVI   H, 00    ;ONLAR'a 0 ver
CEVRIM:  SUI   0A       ;A'dan (+10) (ondalık) çıkart
         JM    GERIAL   ;Eğer sonuç (-) ise GERIAL'a git
         INR   H        ;H kaydedicisine 1 ekle
         JMP   CEVRİM   ;CEVRİM'e git
GERIAL:  ADI   0A       ;A'ya (+10) (ondalık) ekle
         MOV   L, A     ;BİRLER'i L kaydedicisine yerleştir
         RET            ;Geriye dön
```

10.5 Tartışma

Önceki bölümlerde anlatılan sayısal kronometre, hem bir çıkış alt yordamı, hem kesme yönetim yordamları, hem de bir gecikme altyordamı içermektedir.

Kronometrenin hassaslığı, temel zamanlama döngüsünün (Şekil 10.8'daki A'dan B'ye ve tekrar A'ya olan döngüsünün) hassas biçimde ayarlanmasına bağlıdır. Döngü içinde ayrıca gecikme alt yordamı artı, göstergeye çıkılması, çıkış değerlerinin BCD'ye dönüştürülmesi, saniye sayacını 1 artırmak vb. için diğer komutlar bulunmaktadır. Döngünün bir saykılının aldığı süre, tüm bu komutların yürütüm zamanları toplanarak bulunur. Bu saykıl süresinin en büyük bölümünü büyük bir olasılıkla gecikme altyordamı oluşturmaktadır.

Döngüde, gecikme altyordamının dışında, her biri ortalama 2 µsn ya da o civarda bir yürütme süresine sahip diğer komutlardan sözgelimi 200 tane olsa bile, bunlar sadece 400 µsn sürerler.

Döngü toplam süresinin 1 sn olması gerektiğinden, gecikme alt yordamının:

(1.000.000 - 400) = 999.600 µsn

sürmesi gerekir. Pratikte, daha önce anlatılan GECIK alt yordamı üzerinde küçük bir düzeltme yapmaksızın bu uzunlukta bir gecikme elde etmek güç olabilir.

Bu alt yordamın aldığı süre:

1 İçerideki bir döngünün ne kadar sıklıkla çevrimleneceğini belirleyen E kaydedicisine yüklenen bir sabit ve,
2 İşlemcinin C kaydedicisindeki altyordama geçirilen bir parametre tarafından denetlenir.

Bu sabitlerin her biri maksimum sekiz bitlik, yani 255'dir (ondalık). Bu nedenle, GECIK'deki iç döngünün mümkün olan maksimum çevrim sayısı:

255 x 255 = 65.025'dir.

Altyordam tarafından üretilecek gecikme, yukarıda önerildiği gibi 999.600 µsn olacaksa, iç döngünün,

999.600/65.025 = 15.4 µsn (yaklaşık)

süre alması gerekir. Bu uzunlukta bir gecikme elde edebilmek için, GECIK'in iç döngüsünün yürütme süresini artırmak gerekebilir. Örneğin, bu döngünün içine, birkaç NOP komutu daha eklenebilir.

Kronometre ana programındaki bir saniyelik döngünün hassaslığı da önemlidir. Kronometrenin kullanılabileceği maksimum süre 100 dakikadır. Kronometre zamanlama hatasının bu aralıkta 1 saniyeden az olacağını varsayalım. 100 dakikada, zamanlama döngüsü 100 x 60 = 6000 kez çevrimlenir. Her bir saykılda ± 166 µsn'lik bir hata payının hoşgörülebilir olduğu kolaylıkla gösterilebilir. Örneğin, 6000 saykılda hata payı +166 µsn ise geçecek süre,

6.000 x 1.000.166 µsn
= 6 x 1.000.166
= 6.000.996 sn.

olacaktır. Bu da 100 dakikada 1 saniyeden daha az hata anlamına gelmektedir. Bu nedenle, bu döngünün alacağı süre üzerindeki küçük değişimleri algılama imkanı vardır. Her geçen dakikanın sonunda, örneğin kronometrenin, SANY'yi sıfırlamak ve DAKK'yı 1 artırmak için gerekli fazladan komutları gözden geçirmek için eklenecek süre, herhangi önemli bir hataya neden olacak kadar büyük değildir.

Yukarıda tanımlanan gibi bir kronometre gerçekleştirmenin diğer bir yolu, kuşkusuz 1 sn, hatta 0.1 sn'lik aralıklar üretecek bir aralık zamanlayıcısı ve kesmeler
S 10.4 kullanmak, bunları kullanarak temel zaman aralığını denetlemek olacaktır.

Sorular

10.1 Mikrobilgisayar-tabanlı bir ürünün tasarımında gerçekleştirilen işlemleri açıklayın. Bu işlem ile tel-bağlantılı mantık devresi kullanan alışılagelmiş sistem tasarımı arasındaki farklılıklar nelerdir?

10.2 Yazılım geliştirmede gereken işlem adımlarını tartışın. Bu işlemlere yardımcı olacak birimleri, her birinin uygulamasını kısaca açıklayarak, özetleyin.

10.3 Bu bölümde anlatıldığı gibi, bir mikrobilgisayar sistemini bir kronometre olarak çalıştıracak bir program yazın ve çalıştırın.

10.4 Elinizde bir aralık zamanlayıcısı yongasının mevcut bulunduğunu varsayarak, zamanlayıcıdan her 0.1 sn'de bir gelen kesmelerden yararlanmak için, bir kronometre programının yeniden nasıl değiştirilebileceğini tartışın. Bu kesmelerin, işlemci yongasına RST 7.5 girişinden uygulandığını varsayın.

10.5 Bu bölümde verilen akış diyagramlarını yeniden düzenleyerek, Kısım 10.
anlatılan tur süresi özelliğinin kronometreye nasıl eklenebileceğini göst

10.6 Bir tur süresi özellikli kronometre için program kodu yazın ve prog
çalıştırın.

10.7 Kronometre programına ilişkin akış diyagramını ve çıkış gösterge sistemini
zaman aralıklarını dakika, saniye ve saniyenin onda biri olarak ölçülmesine
imkan verecek şekilde yeniden düzenleyin. Burada, çıkışın görüntülenebil
mesi için sıvı-kristalli göstergede toplam 6 basamağa gerek vardır.

10.8 7. soruda verilen kronometre programı için program kodunu yazın.

10.9 Bir mikrobilgisayar kullanılarak sayısal bir saat yapılacaktır. Saatin aşağıdaki
giriş ve çıkışlara sahip olması istenmektedir:

(a) Saat modunu değiştirmek için giriş denetim düğmesi. Mevcut üç mod;
normal zaman göstergesi, alarm ve zaman ayarından ibarettir. Bu dene-
tim düğmesine basmak, modun sırayla değişmesine neden olmalıdır.
Örneğin, saat, normal zaman göstergesindeyken düğmeye bir kez basıl-
masıyla alarm ayarına geçmeli; saat zaman ayarı modunda ise, düğmeye
basınca normal zaman göstergesi moduna dönmelidir.

(b) Saatleri ayarlamak için denetim düğmesi. Bu düğmeye, alarm ayarı ya
da zaman ayarı modlarında bir kez basılması, sırasıyla alarm zamanını
ya da gerçek zamanın 'saatler' değerini gösterecek biçimde değişmelidir.

(c) Dakikaları ayarlamak için denetim düğmesi. Saat düğmesinin çalıştığı
gibi çalışmalıdır.

(d) Zamanın saat ve dakika olarak görüntülenmesini denetlemek için çıkış
hatları.

(e) Alarmı çalıştırmak için bir çıkış hattı.

Giriş ve çıkışların mikrobilgisayara nasıl bağlanması gerektiğini belirtin, gerekli program modüllerini gösteren akış diyagramlarını çizin ve programı kodlayın.

Intel 8085 Komut Takımı

Bu ek bölümde, Intel 8085 mikroişlemcisinin komut takımı özetlenmiştir. Bu bilgiler, Intel Corporation'un izniyle sunulmuştur. Komutlar:

KOMUT KISALTMA	KAYNAK YA DA VARIŞ YERİ VEYA (eğer uygunsa) HER İKİSİ	ON ALTILI KODLAMA

biçiminde verilmiştir.

Böylece, örneğin

```
MOV   C,H    4C
```

komutu, H kaydedicisindeki veri baytının C kaydedicisine yazılmasına neden olur (bu komut tipinde verinin varış yeri, kaynaktan önce belirtilir). Komutun on altılı gösterimi 4C'dir.

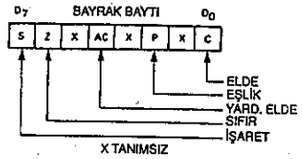
Bu komutlarda:

A, B, C, D, E, H, L	işlemci kaydedicilerini,
M	adresi H, L kaydedici çiftinde tutulan bellek konumunu,
byte (bayt)	8-bitlik bir veri büyüklüğünü ya da bir portun adresini
dble	16-bitlik bir veri büyüklüğünü
adr	16-bitlik bir bellek adresini gösterir.

Komutlar, örneğin LXI komutlarının gösterdiği gibi bir kaydedici çiftini gösterdiğinden, çiftler aşağıdaki gibi gösterilir.

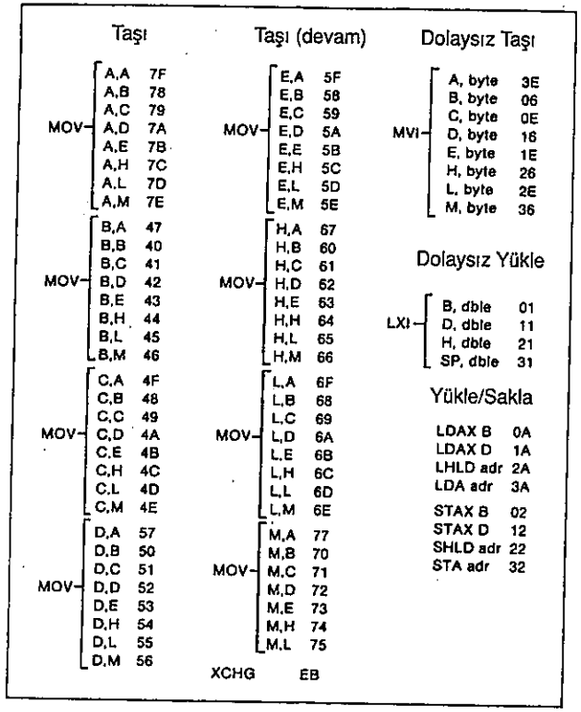
PSW	- A ve F kaydedici çiftini,
B	- B ve C kaydedici çiftini,
D	- D ve E kaydedici çiftini,
H	- H ve L kaydedici çiftini,
SP	- (16-bitlik) yığın işaretçisini
PC	- (16 bitlik) program sayıcısını gösterir.

Durum baytı (F) aşağıdaki gibi bayrak bitleri içerir:



Veri Aktarım Komut Grubu

Bu komutlar (bkz: Şekil A1) işlemci kaydedicileri arasında ya da bellek ile bir kaydedici arasında veriyi taşır. Herhangi bir bayrak bitini değiştirmezler.



Şekil A1 Veri aktarım komutları

Aritmetik ve Mantıksal Komut Grupları

Bunlar Şekil A2'de gösterilmiştir.

* tüm bayrak bitlerini (C, Z, S, P ve AC) etkileyen komutları,

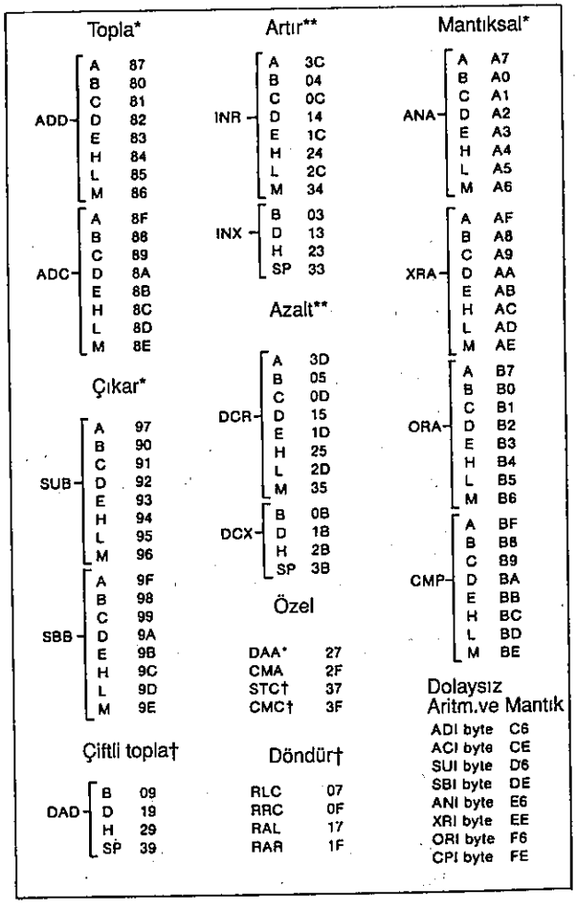
** elde bayrağının dışındaki tüm bayrak bitlerini etkileyen komutları,

† sadece elde bayrağını etkileyen komutları göstermektedir.

Denetim aktarma grubu komutları

Bu komutlar programın bir atlama gerçekleştirmesine olanak tanır. Atlama, çağırma ve dönüş komutlarından oluşmuştur. Koşullu bir atlama, çağırma ya da dönüş belirtildiğinde, kullanılan koşul kodları:

NZ - sıfır değil (bayrak Z=0)
Z - sıfır (bayrak Z=1)



Şekil A2 Aritmetik ve mantıksal komutlar

Atla	
JMP adr	C3
JNZ adr	C2
JZ adr	CA
JNC adr	D2
JC adr	DA
JPO adr	E2
JPE adr	EA
JP adr	F2
JM adr	FA
PCHL	E9
Çağır	
CALL adr	CD
CNZ adr	C4
CZ adr	CC
CNC adr	D4
CC adr	DC
CPO adr	E4
CPE adr	EC
CP adr	F4
CM adr	FC
Geriye Dön	
RET	C9
RNZ	C0
RZ	C8
RNC	D0
RC	D8
RPO	E0
RPE	E8
RP	F0
RM	F8

Şekil A3 Denetim aktarma komutları

NC - elde yok (C bayrağı = 0)
C - elde (C bayrağı=1)
PO - tek eşlik (P bayrağı = 0)
PE - çift eşlik (P bayrağı = 1)
P - pozitif (S bayrağı = 0)
N - negatif (S bayrağı = 1)

Böylece, örneğin,

JNC adr

komutu (bkz: Şekil A3), C bayrağı = 0 ise, program yürütümünün (adr) adresindeki komuta atlamasına neden olur.

Giriş/Çıkış Komutları

OUT bayt D3
IN bayt DB

komutlarındaki 'bayt', üzerinden veri alacak olan G/Ç portunun adresidir.

Yığın İşlemleri

Bu komutlar (bkz: Şekil A4) verinin kaydedici çiftleri ve yığın arasında değiş-tokuş edilmesini sağlarlar.

XTHL komutu, H, L çiftindeki verinin, yığının en üstündeki değerlerle değiş-tokuş edilmesini sağlar.

SPHL komutu, yığın işaretçi kaydedicisinin (SP), H,L çiftinden yüklenmesini sağlar.

Yığın İşle		
PUSH	B	C5
	D	D5
	H	E5
	PSW	F5
POP	B	C1
	D	D1
	H	E1
	PSW*	F1
XTHL		E3
SPHL		F9

Şekil A4 Yığın işlemleri

Makine Denetim Komutları

DI F3 - kesmeleri yetkisizle
EI FB - kesmeleri yetkilendir

Albaştan		
RST	0	C7
	1	CF
	2	D7
	3	DF
	4	E7
	5	EF
	6	F7
	7	FF

Şekil A5 Denetim komutlarını yeniden başlatma

NOP 00 - işlem yapma
HLT 76 - durdur
RIM 20 - kesme maskesini oku
SIM 30 - kesme maskesini kur

Denetim komutlarını yeniden başlatma

Bunlar Şekil A5'de gösterilmektedir.

Bu komutlar, işlemcinin, dışsal bir kesme RST 5.5, RST 6.5 veya RST 7.5 giriş bacaklarının birine geldiğinde oluşan duruma benzer bir işlem görmesine neden olur.

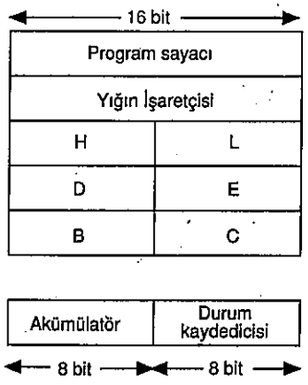
RST komutları için vektör adresleri:

RST 0 - adres 0000 (on altılı)
RST 1 - adres 0008 (on altılı)
RST 2 - adres 0010 (on altılı)
RST 3 - adres 0018 (on altılı)
RST 4 - adres 0020 (on altılı)
RST 5 - adres 0028 (on altılı)
RST 6 - adres 0030 (on altılı)
RST 7 - adres 0038 (on altılı)

Ek 2 Bazı mikroişlemci yongaları

Sekiz-bitlik sistemler

Intel 8085



Şekil A6

Teknoloji: NMOS

Aritmetik: İkinin tümleyeni ikili ve ikili-kodlanmış ondalık sayı toplama ve çıkartma.

Saat frekansı: 3 ya da 5 MHz.

Yol organizasyonu: 8085, veri yolundaki alt sıralı 8 adres bitini çoğullar.

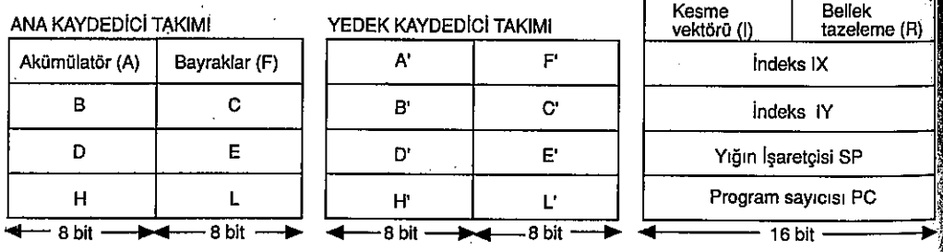
Adres aralığı: 64 Kbaytlık bir adres alanı sağlayan 16-bitlik adresler kullanılır.

İşlemci kaydedicileri: Bkz: Şekil A6.

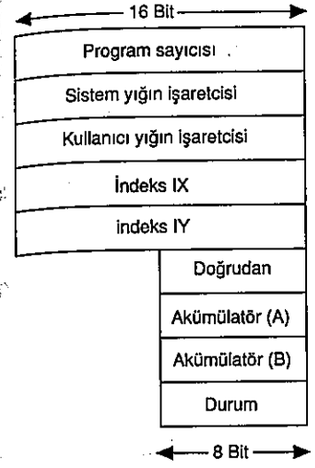
Zilog Z80

Teknoloji: NMOS.

Aritmetik: 8 ya da 16 bit ikiye tümleyeni ikili ve ikili kodlanmış ondalık toplama ve çıkartma.



Şekil A7



Şekil A8

Saat frekansı: 2.5 veya 4 MHz.

Yol organizasyonu: Ayrı veri (8-bit) ve adres (16-bit) yolları

Adresleme alanı: 64 Kbayt.

İşlemci kaydedicileri: Bkz. Şekil A7.

Adres modları: Dolaysız, dolaysız genişletilmiş, bağıl adres, genişletilmiş adres, indeksli adres, kaydedici adresi, örtük adres, kaydedici dolaylı adresi, bit adresi.

Motorola MC 6809

Şekil A9

Teknoloji: NMOS.

Aritmetik: İkili ikinin tümleyeni toplama ve çıkartması artı işaretsiz 8-bitlik çarpma. Bazı 16-bitlik aritmetik işlemler.

Saat frekansı: 4 MHz.

Yol organizasyonu: Ayrı veri (8-bit) ve adres (16 bit) yolları.

Adresleme alanı: 64 Kbayt.

İşlemci kaydedicileri: Bkz. Şekil A8.

Adresleme modları: Doğrudan sayfa (doğrudan sayfa kaydedicisi kullanarak), indeksli adres, otomatik artırma ve azaltma adresleri, bağıl adres, dolaysız.

16-bitlik sistemler:

Intel 8086

Teknoloji: NMOS.

Aritmetik: 16-bitlik ikili ikinin tümleyeni ve BCD aritmetik;

16-bitlik işaretsiz ikili çarpma ve bölme.

Saat frekansı: 5 MHZ.

Yol organizasyonu: 16-bit veri ve alt sıralı 16 adres biti ortak yol üzerinde çoğullanır.

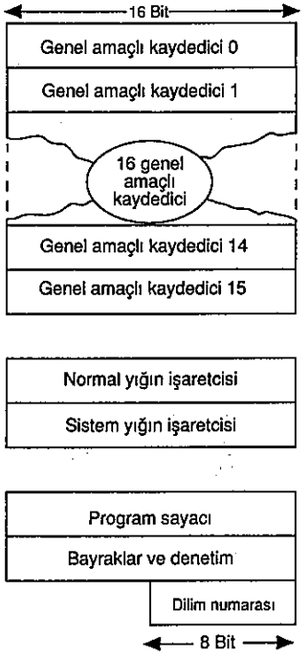
Adresleme alanı: 1 Mbaytlık bir adresleme alanı sağlayan 20 bitlik adresler kullanılır.

İşlemci kaydedicileri: Bkz. Şekil A9.

Adresleme modları: Doğrudan adres; bağıl adres (taban ve indeks kaydedicilerine), kaydedici adresi, indeksli adres, dolaysız.

Veri tipleri: Bitler, baytlar, sözcükler (16-bit), diziler

Zilog Z8000



Şekil A10

Teknoloji: NMOS.

Aritmetik: 16 bit ikili ve BCD aritmetiği. İşaretli çarpma ve bölme.

Saat frekansı: 4 MHZ.

Yol düzenlemesi: 16 bitlik veri ve alttaki 16 adres biti ortak yol üzerinde çoğullanır.

Adresleme alanı: Z8002 64 Kbaytlık, Z8001 8 Mbaytlık bir adresleme alanına sahiptir.

İşlemci kaydedicileri: Bkz. Şekil A10. Genel amaçlı kaydediciler, 16-bitlik 16 kaydedici, veya 32-bitlik 8 kaydedici ya da 64-bitlik 4 kaydedici olarak kullanılabilirler.

Adres modları: Kaydedici, kaydedici dolaylı, doğrudan, indeksli, dolaysız, bağıl, taban adresi, taban indeksli, otomatik artımlı veya azalımlı.

Veri tipleri: Bitler, BCD karakterleri, baytlar, sözcükler (16 bit), uzun sözcükler (32 bit) ve bayt ya da sözcük dizileri.

Motorola MC 68000

Teknoloji: HMOS.

Aritmetik: 8-, 16-, veya 32-bitlik ikili aritmetik işaretli ve işaretsiz çarpma bölme.

Saat frekansı: 8 MHZ.

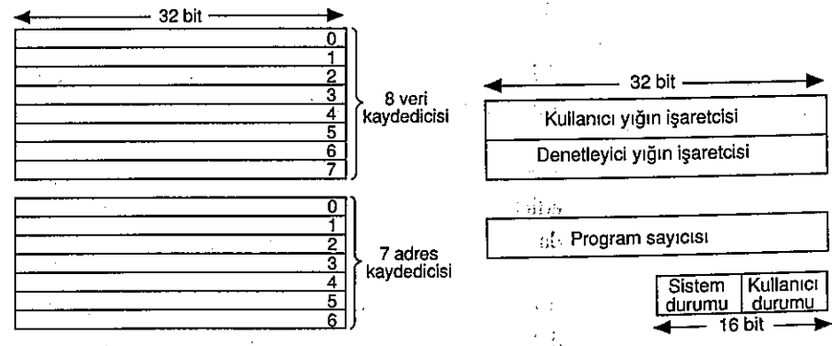
Yol organizasyonu: Ayrı 16-bitlik veri yolu ve 23-bitlik adres yolu. 24. adres biti işlemci yongası dışında üretilir.

Adresleme alanı: 16 Mbaytlık bir adresleme alanı sağlayan 24-bitlik adresler kullanılır.

İşlemci kaydedicileri: Bkz. Şekil A11

Adres modları: kaydedici, mutlak (doğrudan), bağıl, kaydedici dolaylı (artırma sonrası ya da azaltma öncesi), dolaysız, örtük.

Veri tipleri: bitler, BCD karakterleri, baytlar, sözcükler (16 bit), uzun sözcükler (32 bit).



Şekil A11

Soruların Cevapları

1.1 Bkz: Kısım 1.1

1.2 Bkz: Kısım 1.2

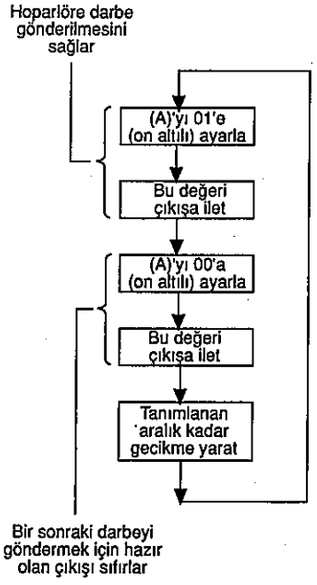
1.3 Bkz: Kısım 1.1, 1.2 ve 1.3. Bu soruyu tam olarak cevaplamak için daha fazla bilgiye ihtiyaç vardır.

1.4 Bkz: Kısım 1.3, 1.6(e) ve 1.7. Diğer bellek ayrıntıları Kısım 2.3'de verilmiştir.

1.5 2^{24} adres, yani 16.384 Kbayt ya da 16 Mbayt.

1.6 Bkz: Kısım 1.2. Bir programda bir JUMP komutu ile karşılaşıldığında, şu işlemler gerçekleşir:

1 İşlemci, (eğer varsa) JUMP'da belirtilen koşulun karşılanıp karşılanmadığını test eder.
2 Koşul yerine getirilmiş ise, komuttaki adres program sayıcısına yüklenir.
3 Koşul yerine getirilmemiş ise, program sayıcısı, bir sonraki komutu gösterecek şekilde artırılır.



Şekil A

1.7 Bkz: Kısım 1.3

1.8

```
(a)  MVI   C, 92
(b)  MVI   D, 64
(c)  MOV   A, C
(d)  ADD   D
(e)  STA   3130
```

1.9 Müzik notası üretmek üzere, bilgisayarın hoparlöre bir darbe dizisi göndermesi gerekmektedir. Bu dizinin frekansı duyulan notanın titreşimini belirlemektedir. Örneğin, her 200 µsn'de bir 10 µsn'lik darbe gönderilirse, hoparlör 5000 Hz frekanslı bir kare dalga kabul edecektir. Her 400 µsn'de bir 100 µsn'lik darbe gönderilirse, hoparlör 2500 Hz temel frekanslı bir darbe dizisi alacaktır. İşlemin bir akış diyagramı Şekil A'da gösterilmektedir. Gecikme periyodunu değiştirerek, notanın frekansı değiştirilebilir.

1.10

```
BASLA:  MVI   A, 01      ;Çıkış hattını 1'e ayarla
        OUT   32
        MVI   A, 00      ;Çıkış hattını 0'a ayarla
        OUT   32
        MVI   A, 14
TEKRAR: NOP              ;20 kere çevrimlenen gecikme döngüsü
        DCR   A
        JNZ   TEKRAR
        JMP   BASLA      ;BASLA'ya geri dön.
```

Gecikme döngüsünün saykıl sayısını değiştirerek, yani MVI A, 14 komutundaki sabiti değiştirerek, farklı frekanslarda notalar elde edilebilir.

1.11 $d_0, d_1, d_2, \ldots d_9$ değerlerinin, 0B00 adresinden başlayarak belleğe saklandığını varsayalım:

```
BASLA:  MVI   A, E7
TEKRAR: NOP              ;Gecikme döngüsü (1.976 msn)
        NOP
        DCR   A
        JNZ   TEKRAR
        LXI   H, 0B00    ;H,L kaydedici çiftini, 0B00 adresini
                         ;işaret edecek şekilde ayarla
        MOV   A, M       ;Bellekten bir (dn) değeri
                         ;getir
        OUT   32         ;Değeri çıkışa ver
        INR   L          ;(L)'yi 1 artır
        MVI   A, 09      ;(L)=9 olup olmadığını test et
        SUB   L
        JZ    SIFIRLA
        JUMP  BASLA      ;Bir sonraki değere git
SIFIRLA: MVI  L,00       ; (L)=0 yap
        MVI   A,00       ;Çıkış değeri=0 yap
        OUT   32         ;Rampayı sıfırla
        JMP   BASLA
```

Gecikme döngüsü 247(ondalık) kere çevrimlenir.

2.1 Bkz: Kısım 2.1, 2.3, 2.4 ve 2.5

2.2 Bkz: Kısım 2.1

2.3 Bkz: Kısım 2.2 ve Ek 2

2.4 Bkz: Kısım 2.2. Dördüncü kuşak makineler kendilerinden önceki makinelerden daha güçlüdür. Veri tiplerini işlemek için daha geniş aralıkta komutlara sahiptirler, daha hızlıdırlar, daha geniş aritmetik özelliklerine sahiptirler ve daha geniş bir bellek alanını doğrudan adresleyebilirler. Bu özellikler dördüncü kuşak makineleri, güçlü, genel-amaçlı bilgisayarların temeli olarak kullanılmak üzere, daha önceki makinelerden daha kullanışlı yapar. Büyük ölçekli entegre devrelerin üretim maliyetlerinin azalmasıyla birlikte, büyük çapta hesaplama gücü, sürekli artan sayıda kullanıcı tarafından tercih edilecekleri anlamına gelmektedir.

Dördüncü kuşak mikroişlemciler, aynı zamanda küçük-ölçekli birimlerin kapasitelerine büyük çapta katkıda bulunmaktadır. Buna örnek olarak, pratisyen doktorların çalışmalarında ve bazı klinik durumların teşhislerinde yardımcı olacak bir sistem tasarımı şu an için mümkün hale gelmiştir.

2.5 Bkz: Kısım 2.2

2.6 Bkz: Kısım 2.2 ve 2.3

2.7 Bkz: Kısım 2.3

2.8 Bkz: Kısım 2.4 ve 4.8

2.9 1 Adres hattı 0'ın PPI'ın en düşük değerlikli 'kaydedici seçim' girişine bağlı olduğunu,

2 Adres hattı 1'in PPI'ın en büyük değerlikli 'kaydedici seçimi' girişine bağlı olduğunu,

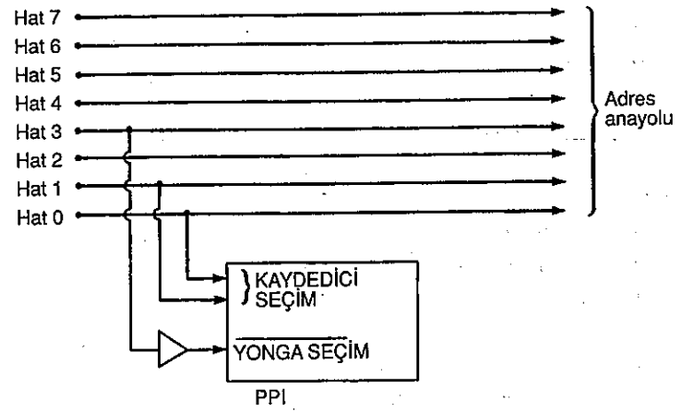
3 Adres hattı 3'ün (bir tersleyici üzerinden) PPI'ın $\overline{\text{CHIP SELECT}}$ girişine bağlı olduğunu varsayın.

Bu durumda PPI'da kullanılan adresler:

A portu	= 08 (on altılı)
B portu	= 09 (on altılı)
C portu	= 0A (on altılı)
Denetim kaydediciyi	= 0B (on altılı)

Sistem Şekil B'de gösterilmektedir. PPI'ı kullanıma hazırlamak için gerekli komutlar:

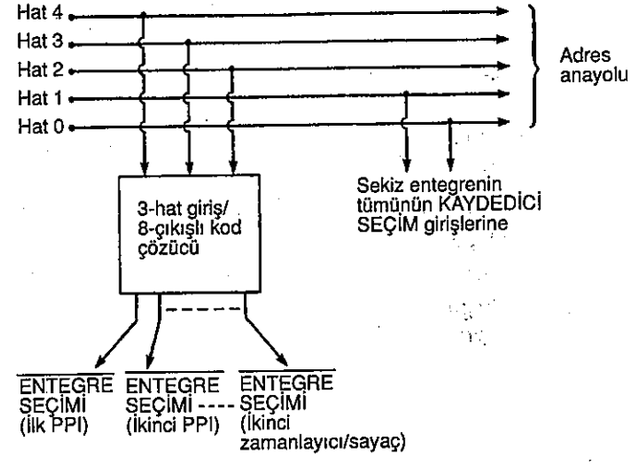
```
MVI     A, A7
OUT     0B
```



Şekil B

2.10 Adres ataması:

00'dan 03'e	birinci PPI
04'den 07'ye	ikinci PPI
08'den 0B'ye	üçüncü PPI



Şekil C

0C'den 0F'ye	dördüncü PPI
10'dan 13'e	beşinci PPI
14'den 17'e	altıncı PPI
18'den 1B'ye	birinci zamanlayıcı /sayıcı
1B'den 1F'ye	ikinci zamanlayıcı/sayıcı

Bu, her bir birim için dört adrese izin verir. Sistem, Şekil C'de gösterilmiştir.

2.11 Bkz: Kısım 2.4

2.12 Bkz: Kısım 2.5 ve 1.3

2.13 Sayım kaydedicisi 0'ın, 'olaylar'ı saymak için ve sayım kaydedicisi 1'in, dış saat darbesi üreteci ile birlikte, bir 5 sn aralığı ölçmek için kullanıldığını varsayın. En küçük değerlikli adres yolu hattı (hat 0), istenen sayım kaydedicisini seçmek için kullanılır. Adres yolu hattı 1, zamanlayıcı-sayacı seçmek için kullanılır. Böylece:

Kaydedici 0'ın adresi = 02 (on altılı)
Kaydedici 1'in adresi = 03 (on altılı)

Her iki sayacın, başlangıçta içlerine yerleştirilen bir değerden aşağıya doğru saydığını varsayın.
5 sn'lik periyot süresinde, saat üreteci darbe sayısı 5x25 = 125'dir.
Program şöyledir:

```
        MVI   A, 7D     ;A'ya +125 yükle
        OUT   03        ;Sayım kaydedicisi 1'e çık
        MVI   A, 78     ;A'ya +120 yükle
        OUT   02        ;Sayım kaydedicisi 0'a çık
OKU:    IN    03        ;Kaydedici 1'i oku
        CPI   00        ;Değeri 0 mı?
        JZ    SON       ;Süre tamamlanmışsa, SON'a git
        JMP   OKU       ;Süre tamamlanmamışsa, OKU'ya git
SON:    İN    02        ;Kaydedici 0'daki sayıyı oku
        MOV   B, A      ;Olayları saymak için değeri,
        MVI   A, 78     ;120'den çıkart
        SUB   B
        STA   adrs      ;Sonucu 'adrs' adresinde
                        ;sakla
        HLT             ;Dur
```


2.14 Delgi birimine giren veri giriş hatları, PPI'nın, çıkış olarak kullanılan, sözgelimi B portuna bağlanır. PPI, B grubu mod 1'de (onay modunda) çalışacak şekilde ayarlanır. Delgiden çıkan 'ready (hazır)' hattı, PPI'da $\overline{\text{ACK}}$ (C portunun 2. bitine)'ye bağlanır. Delgideki 'veri kullanılabilir' hattı ise, $\overline{\text{OBF}}$'den (C1 biti) bir tersleyici aracılığıyla sürülür. PPI adreslerinin, soru 2.9'daki gibi olduğunu varsayın. PPI'yı konfigüre etmek için:

```
MVI     A, 04
OUT     0B
```

Veriyi çıkışa vermek için:

```
MVI     A, (veri)
OUT     09
```

ya da, eğer çıkışa verilecek veri bir kaydedicide, sözgelimi işlemci B kaydedicisinde tutulursa:

```
MOV     A,B
OUT     09
```

3.1 Bkz: Kısım 3.1

3.2 Bkz: Kısım 1.3, 2.1, 2.2, 2.3 ve 3.1

3.3 Bkz: Kısım 3.2 ve 3.6

3.4 Bkz: Kısım 3.1 ('Durumlar') ve 3.4

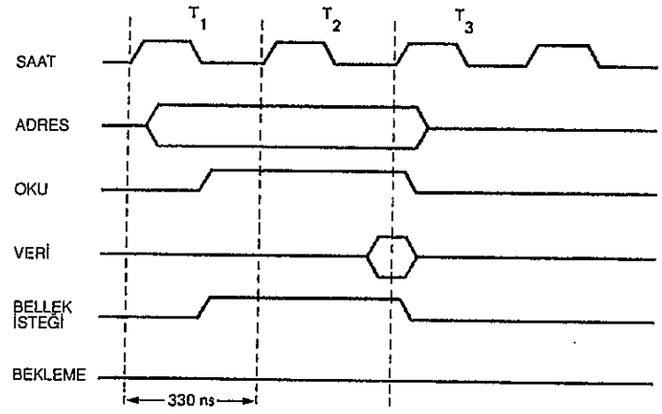
3.5 Bkz: Kısım 1.3 ve 3.3

3.6 Bkz: Kısım 3.7

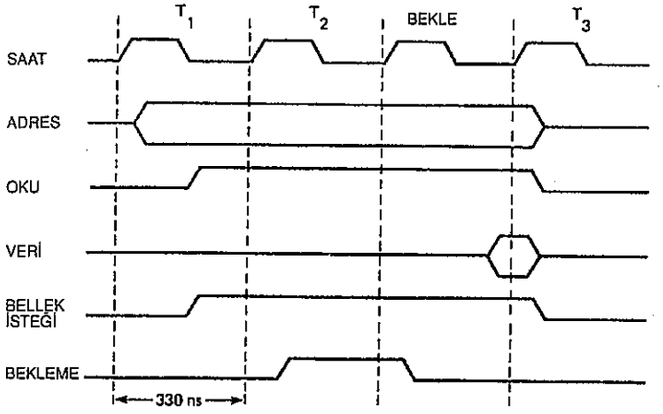
3.8 Bkz: Kısım 3.4 ve 3.5 ve Şekil D, E ve F

3.9 Bkz: Kısım 3.4 ve 3.5

4.1 Bkz: Kısım 4.1 ve 4.2



Şekil D 400 nsn'lik ROM'dan okuma işlemine ilişkin diyagram



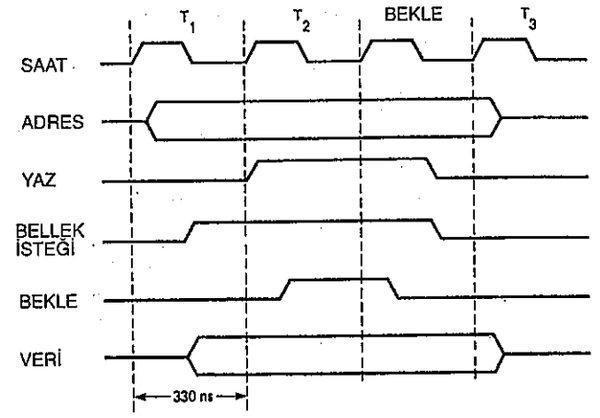
Şekil E 550 nsn'lik RAM'dan okuma işlemine ilişkin diyagram

4.2 Bkz: Kısım 4.1

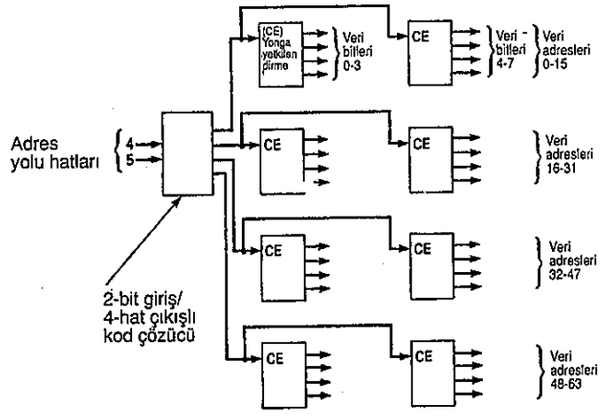
ROM Başlangıç adresi 0000 Bitiş adresi 2FFF
RAM Başlangıç adresi 6000 Bitiş adresi 7FFF

4.3 Bkz: Kısım 4.7

(a) RAM, sekiz Intel 3101A RAM yongasından oluşabilir. Konfigürasyon, Şekil G'de gösterilmektedir. Adres yolunun, (her ikisi de dahil olacak biçimde) 0-3 hatları tüm yongalarla bağlantılıdır. Gösterildiği gibi, iki-bit giriş/dört-hat çıkışlı bir adres kod çözücü gereklidir.



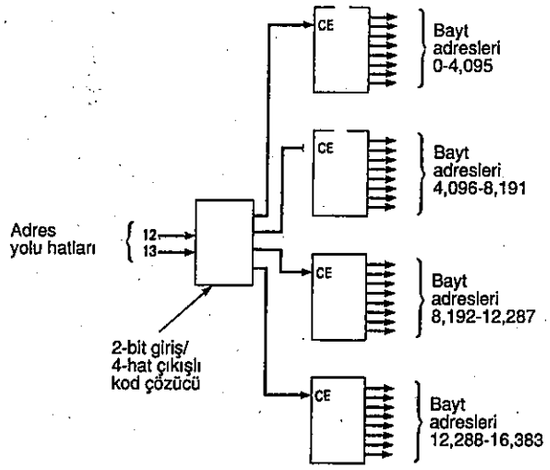
Şekil F 550 nsn'lik RAM'a yazma işlemine ilişkin diyagram



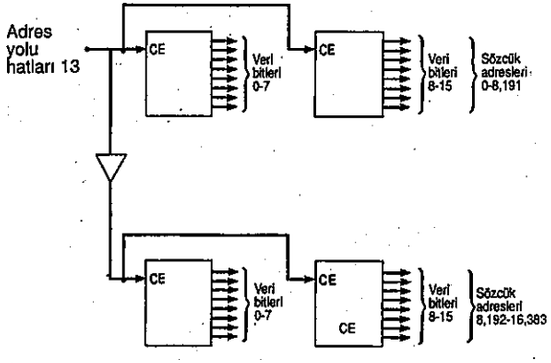
Şekil G

(b) EPROM, Şekil 1-1'de gösterildiği gibi, dört Intel 2732 EPROM yongasından oluşturulabilir. Her bir yonga 4.096 baytlık bir saklama alanına sahiptir. Adres yolunun, (her ikisi de dahil olacak biçimde) 0-11 hatları tüm yongalarla bağlantılıdır. Yine, bir iki-bit giriş/dört-hat çıkışlı kod çözücü kullanılır.

(c) ROM, INTEL 2364A yongalarından oluşturulabilir. Bunların her biri 8.192 baytlık saklama alanına sahiptir. Konfigürasyon, Şekil I'da



Şekil H



Şekil I

gösterilmektedir. Adres yolu hattı, gösterildiği gibi, 0-8.191 ya da 8.192-16.383 adreslerini seçmek için kullanılır. Adres kod çözücüsüne gerek yoktur.

4.4 Intel 2114 RAM yongası, her biri 4 bitlik 1024 adrese sahiptir. Bu nedenle, 8.192 baytlık bir bellek oluşturmak için, aşağıdaki gibi, bayt adresleri meydana getirmek için 2114 yonga çiftleri kullanılır:

Birinci çift	0'dan 1.023'e
İkinci çift	1024'den 2.047'ye
Üçüncü çift	2048'den 3.071'e
Dördüncü çift	3072'den 4.095'e
Beşinci çift	4096'dan 5.119'a
Altıncı çift	5.120'den 6.143'e
Yedinci çift	6.144'den 7.167'ye
Sekizinci çift	7.168'den 8.191'e

Adres yolunun A_0-A_9 hatları tüm yongalara paralel olarak girerler. A_{10}, A_{11} ve A_{12} adres yolu hatlarının, üç-bit giriş/sekiz-hat çıkışlı bir kod çözücüde kodları çözülür. Yukarıda gösterildiği gibi bellek yongası çiftlerini seçmek için, 8 çıkış kullanılır.

4.5 Bkz: Kısım 4.7. Adres yolu hatları, bellek girişlerini sürmek için tamponlanmalıdır. Tamponlar A_0-A_{12} hatlarına yerleştirilir ve dörtlü tamponlar kullanılırsa, bunun için fazladan dört yongaya ihtiyaç vardır. Veri yolunun D_0-D_7 hatları da bellek veri girişlerinden tamponlanmalıdır. Bunun için, iki yongaya ihtiyaç duyulacaktır. Toplam yonga sayısı (16 bellek yongası dahil olmak üzere) 22'dir.

Şekil J

4.6 Bkz: Kısım 4.2

4.7 En üstteki altı adres hattının (A_{10}-A_{15} dahil) kodları çözülmelidir. Bu hatlardaki bit deseni 111110 ve 111111 olduğunda, bir çevresel birim isteği gönderiliyor demektir. Bundan farklı herhangi bir bit deseni belleğe karşılık gelir. Bu uygun bir çevresel adres seçimidir, ancak daha büyük (6-bitlik) bir kod çözücüyü gerekli kılar.

4.8 Bkz: Kısım 4.8.

4.9 Bkz: Kısım 4.5.

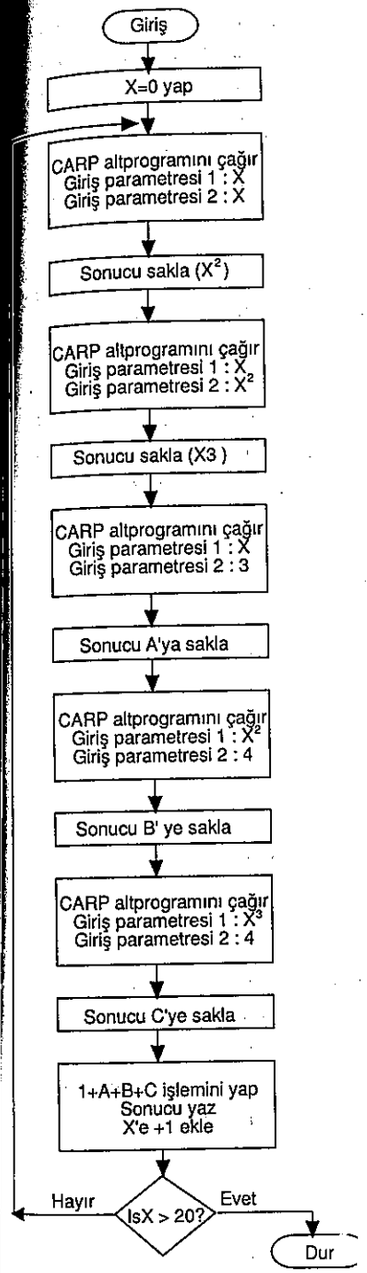
(a) Her biri 6 girişe sahip 64 VE kapısı gereklidir. Her bir kapı, 6 bitin olası 64 kombinasyonundan birini algılar.

(b) 9 tane üç-bit giriş/8-hat çıkışlı kod çözücü gerek-lidir. Bunların sekizi, doğrudan girişlerine besle-nen 6 girişin en küçük değerlikli üçüne sahiptir. Bu sekiz yongadan 64 çıkış elde edilir. 9. yonga, giriş olarak 6 bitin en ağırlıklı üçüne sahiptir. Sekiz çıkış, diğer yongaların genişletme girişleri-ni yetkilendirmek için kullanılır. Ayrıca bkz: Şe-kil 4.14.

4.10 Mikrobilgisayarın, Şekil 4.21'de gösterildiği gibi, bir çevresel arabirim devresi içerdiğini varsayın. Çamaşır makinasından gelen sinyaller, bu birimin A portunun en alttaki dört bitine (0, 1, 2 ve 3) bağlanabilir - Şekil J'nin akış diyagramına bakın. A portunun adresinin 08 (on altılı) ve PIC denetim kaydedicisinin adresinin 0B (on altılı) olduğunu varsayın. Program kodu aşağıda verildiği gibi olacaktır:

Şekil K

```
MVI   A, 92      ;A portu (ve bu arada
                  B portu)
OUT   08         ;giriş olacak biçimde
                  ;PIC'i kullanıma hazırlayın.
IN    08         ;A portundan veriyi oku
MOV   B, A       ;Giriş verisini B kaydedicisine yerleştir
ANI   01         ;Bit 0'ı ayırmak için B kaydedicisini
                  ;01 (on altılı) VE'le
STA   (adres 1)  ;Sonucu 'adres 1'e sakla
MOV   A, B       ;Giriş bit desenini A'dan geriye al
ANI   02         ;Bit 1'i ayırmak için 02 (on altılı)
                  ;ile VE'le
RRC              ;Sonucu en küçük değerlikli
                  ;konuma kaydır
STA   (adres 2)  ;Sonucu 'adres 2'ye sakla
MOV   A, B       ;Giriş bit desenini A'dan geriye al
ANI   04         ;Bit 2'yi ayırmak için 04 (on altılı)
                  ;ile VE'le
RRC              ;Sonucu en küçük değerlikli
RRC              ;ile VE'le
STA   (adres 3)  ;Sonucu 'adres 3'e sakla
MOV   A, B       ;Giriş bit desenini A'dan geriye al
```



Şekil L


```
ANI   08         ;Bit 3'ü ayırmak için 08 (onaltılı)
                  ;ile VE'le
RRC              ;Sonucu en küçük değerlikli
RRC              ;konuma kadar kaydır
RRC
STA   (adres 4)  ;Sonucu 'adres 4'e sakla
```

Bu program kodu, 'yıkama kazanı dolu' bitini Adres 1'e, 'Su ısıtıcısı açık' bitini Adres 2'ye, 'yıkama kazanı boş' bitini Adres 3'e ve 'suyun sıcaklığı yeterli' bitini Adres 4'e yerleştirir.

4.11 Bir çevresel arabirim devresinin kullanıldığını, 'açık-kapalı' anahtarının, A portunun en az ağırlıklı biti (bit 0)'ne bağlandığını ve 'süre' anahtarlarının A portunun 1., 2. ve 3. bitlerine bağlandığını varsayın. B portu, bit 0 konumunda ortaya çıkan kare dalga ile, bir çıkış portu olarak kullanılır. Şekil K'nın akış diyagramına bakın.

5.1 Bkz: Kısım 5.1 ve 5.2

5.2 Bkz: Kısım 5.2, 5.3 ve 5.4

5.3 Bkz: Kısım 5.2

5.4 Bkz: Kısım 5.2, 5.4, 5.5 ve 6. Kısım

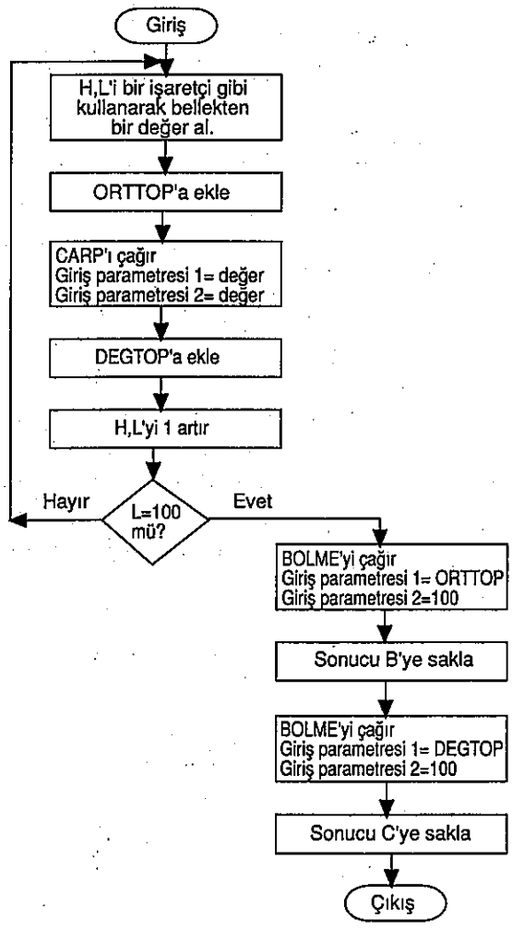
5.5 Bkz: Kısım 5.5

5.6 Bkz: Kısım 5.4 ve 7.8

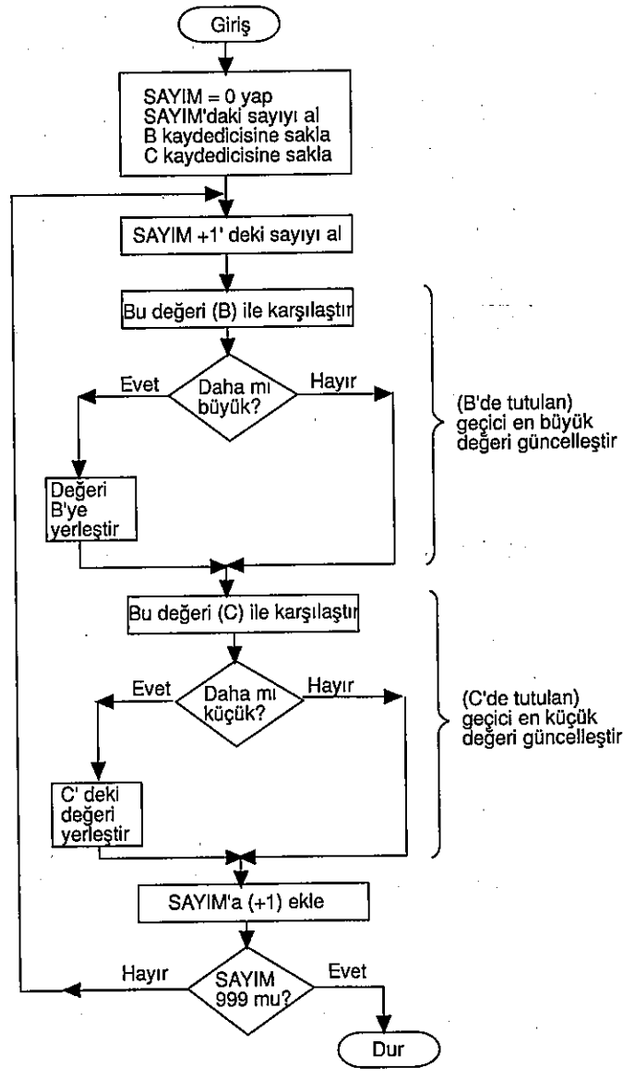
5.7 İki giriş parametreli (birbirleriyle çarpılacak işlenenler) ve bir çıkış parametreli (çarpımın sonucu) bir çarpma alt programı, CARP'ın kullanılacağını varsayın - Şekil L'nin akış diyagramına bakın.

5.8 Alt program:

(a) Sayılar bellekten birer birer alınmalı ve 100. ra-



Şekil M



Şekil N

kama ne zaman ulaşıldığı algılanarak teker teker toplanmalıdır.

(b) Her bir sayının karesi alınmalı ve bunları teker teker toplamalıdır.

(c) Her iki sonucu 100'e bölmeli (ya da 0.01 ile çarpmalı).

(d) Her iki sonucu çağıran programa geriye aktarmalı. 100 giriş parametresi ve iki çıkış parametresi (ortalama ve değişkenlik) vardır. Böylece giriş parametreleri, en iyi şekilde, sözü edildiği gibi RAM'de, 0B00-0B63 adreslerinde saklanıp ve bunlara H ve L'de bir işaretçi vermek suretiyle alt programa aktarılır.

Çıkış değerleri, işlemci kaydedicileri kullanılarak geriye, çağıran programa aktarılabilirler.

5.9 Giriş parametrelerini alt yordama aktarmak için, bu değerler 0B00 ve 0B63 adreslerine saklanır. Ardından çağıran program, işaretçiyi H,L kaydedici çiftindeki veriyle yüklemek için,

```
LXI H, 0B00
```

komutunu çalıştırır. Alt yordam, giriş değerlerini taramak için:

```
MOV A,M
INX H
```

komutlarını tekrarlayarak bu değerlere ulaşır. Eğer hesaplanan sonuçlar A kaydedicisinde işleme sokulduysa, çağıran programa dönüş için:

```
MOV B, A
```

tipi komutla, bunları diğer uygun (B, C, D veya E) kaydedicilerine yerleştirecektir.

5.10 Şekil M'den de görülebileceği gibi, 8085 mikrobilgisayarındaki çarpma ve bölme komutlarının eksikliğinin giderebilmek için, iki alt yordama, CARPMA ve BOLME altyordamları gerekecektir.

5.11 Programın 'güncelleme' parçaları (bkz: Şekil N) altyordam olarak yazılabilir. Sayılar, bu program parçalarına, yürürlükteki en büyük veya en küçük değer ile karşılaştırmak üzere, bir kaydedici aracılığıyla aktarılabilir.

6.1 Bkz: Kısım 6.1 ve 6.4.

6.2 Bkz: Kısım 6.1 ve 6.3.

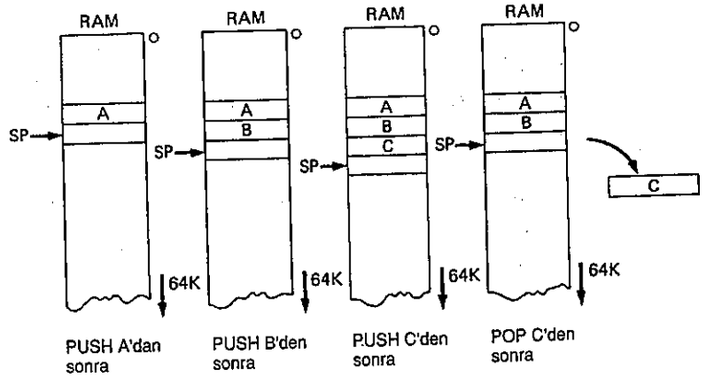
6.3 Yapı aşağıdaki gibidir. Alt yordam çağrıları oklarla gösterilmektedir.

```
M → A → C → G → H
M → B → F
M → C → G → H
M → D → A → C → G → H
M → E ↗ D → A → C → G → H
      ↘ A → C → G → H
M → F
M → G → H
M → H
M → I
```

M ana programı gösterir. Yığının boyutunu, altyordamların iç içe geçirilebileceği maksimum derinlik belirler. E çağrısı için bu değer 6'dır. Gereken yığın boyutu 18 konumdur.

6.4 Bkz: Kısım 6.3.

6.5 Çarpma alt yordamı çağrıldığında, işlemci kaydedicilerinin (3) ve program sayıcısının (2) içeriklerini saklamak için 5 baytlık yığın gereklidir. Böylece geriye 3 bayt kalır. İki tane 8-bitlik işlenenin bir çarpma yordamına aktarılması ve 16-bitlik sonucun bu altyordamdan geriye alınması için, bu yeterli bir alandır. Argümanları aktarmak için en uygun seçenek, onları tutmak için işlemci kaydedicilerini kullanmak olacaktır.



Şekil O

6.6 Bu, eğer PUSH ve POP komutları aşağıdaki gibi kullanılırsa mümkündür.

PUSH = Veriyi sakla, sonra yığın işaretçisini 1 artır.
POP = Yığın işaretçisini 1 azalt sonra veriyi geriye al.

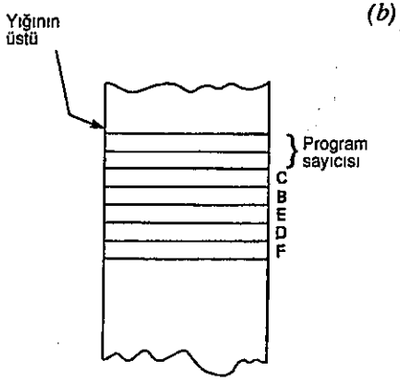
Bkz: Şekil O.

6.7 *(a)* Argümanların işlemci kaydedicilerini kullanılarak değiş-tokuş edildiğini varsayın.

Giriş argümanları B - B kaydedicisinde
C - C kaydedicisinde
D - D kaydedicisinde
E - E kaydedicisinde
F - H kaydedicisinde tutulur.

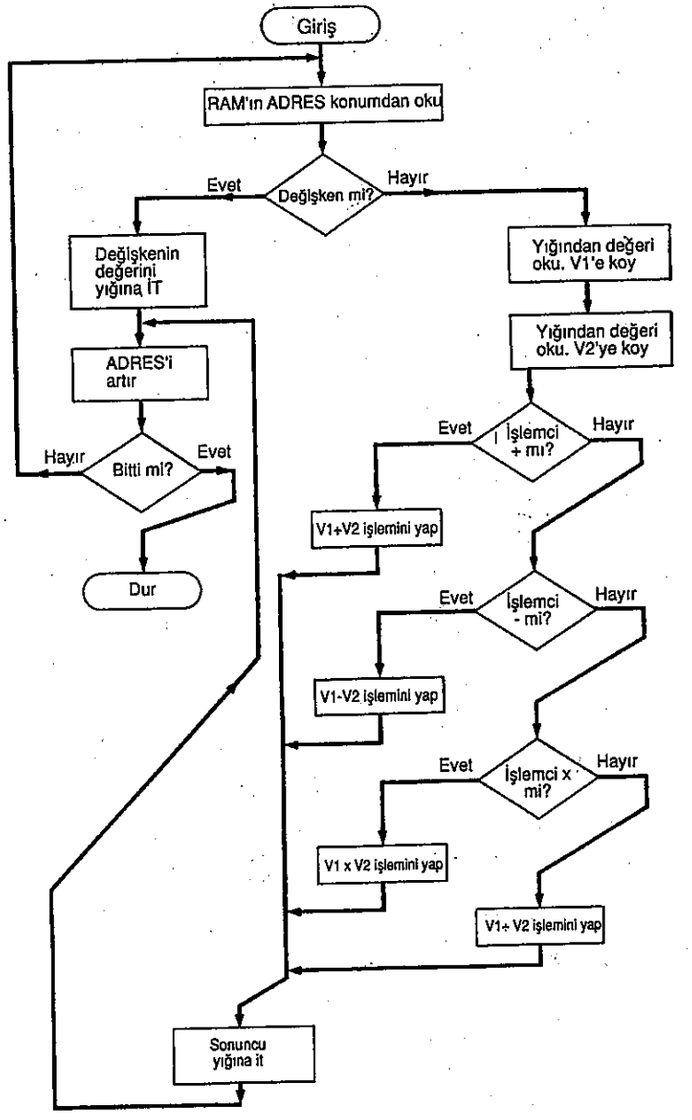
```
MOV   A, B    ;B'yi A kaydedicisine yerleştir
ADD   C       ;(B + C) işlemini yap
MOV   B, A    ;(B + C)'yi B kaydedicisine sakla
MOV   A, D    ;D'yi A kaydedicisine yerleştir
SUB   E       ;(D - E) işlemini yap
MOV   D, A    ;(D- E)'yi D kaydedicisine yerleştir
MOV   A, B    ;(B + C) - (D - E) + F işlemini yap
SUB   D
ADD   H
RET           ;Geriye dön
```

(b) Argümanların yığın kullanılarak değiş-tokuş edileceğini varsayın. Alt yordama girişteki yığın içerikleri Şekil P'de gösterildiği gibidir.



Şekil P

```
POP   H       ;PC'yi H,L'e sakla
POP   B       ;B ve C'yi yığından al ve topla
MOV   A, B    ;ve sonucu B kaydedicisine yerleştir
ADD   C
MOV   B, A
POP   D       ;D ve E'yi yığından al, (D - E) işlemini
MOV   A, D    ;yap ve sonucu D kaydedicisine yerleştir
SUB   E
```

Şekil Q

```
MOV     D, A
MOV     A, B    ;(B + C) - (D - E) + F işlemini yap
SUB     D
POP     B
ADD     C
PUSH    H       ;PC'yi yığına geri al
RET
```

Her iki alt yordamda da sonuç, A kaydedicisindeki çağıran programa geri döndürülür.

6.8 İfadelerin 'ADRES' adresinden başlayarak RAM'e saklandığını varsayın. Akış diyagramı Şekil Q'da gösterildiği gibidir.

7.1 Bkz: Kısım 7.2 ve 7.3

7.2 *(a)* Program sayıcısı 03A0
Yığın işaretçisi 0FFE

Yığın	ADRES	
	0FFE	- C0
	0FFD	- 00
	1000	- XX

(b) Program sayıcısı 03D4
Yığın işaretçisi 0FFC

Yığın	ADRES	
	0FFC	- B8
	0FFD	- 03
	0FFE	- C0
	0FFF	- 00
	1000	- XX

(c) Program sayıcısı 0605
Yığın işaretçisi 0FFA

Yığın	ADRES	
	0FFA	- E3
	0FFB	- 04
	0FFC	- B8
	0FFD	- 03
	0FFE	- C0
	0FFF	- 00
	1000	- XX

(d) Program sayıcısı 03A0
Yığın işaretçisi 0FFE

Yığın	ADRES	
	0FFE	- C0
	0FFD	- 00
	1000	- XX

7.3 Bkz: Kısım 7.4.

7.4 N'nin değerinin A kaydedicisindeki altyordama ve sonucun da alt yordamdan H ve L kaydedicisine geriye aktarıldığını varsayın:

```
        LXI     H, 0000     ;(H,L) = 0 yap
        LXI     D, 0001     ;(D,E) = 1 yap
TEKRAR: CMP     E           ;(A'daki) N değeri [E]'den küçük mü?
        RC                  ;Evet ise, geriye dön
        DAD     D           ;(D,E)'yi H,L'e ekle
        INX     D           ;D,E'yi (+2) artır.
        INX     D
        JMP     TEKRAR
```

Bu altyordam, makine durumunu saklamaz. Ancak, eğer gerekirse, bunu yapmak için fazladan komutlar kolaylıkla eklenebilir.

7.5 Çarpma alt yordamının, aşağıdaki gibi, kendisine aktarılan iki işleneni kabul ettiğini varsayın:
1 B,C kaydedici çiftindeki işlenen 1 (B'deki en büyük değerlikli bayt)
2 D,E kaydedici çiftindeki işlenen 2 (D'deki en büyük değerlikli bayt)
Ayrıca çarpma işleminin sonucunun B, C kaydedici çiftinde geri döndürüleceğini varsayın.

```
        MOV     E, D        ;N'yi E kaydedicisine yerleştir
        DCR     D           ;(N - 1)'i C kaydedicisine yerleştir
        MOV     C, D
        MVI     D, 00       ;(D) = 0 yap
        MVI     B, 00       ;(B) = 0 yap
SONR:   CALL    CARPMA      ;Çarpma altyordamını çağır
        DCR     E           ;D,E'ye (N-2) değerini yerleştir
        RZ                  ;(E) = 0 ise geriye dön
        JMP     SONR
```

Bu alt yordamın sonucu B ve C kaydedicilerinde tutulur.

7.6 N'nin değerinin, H, L kaydedici çiftinde ve M'nin değerinin ise B,C kaydedici çiftinde altyordama aktarıldığını varsayın:

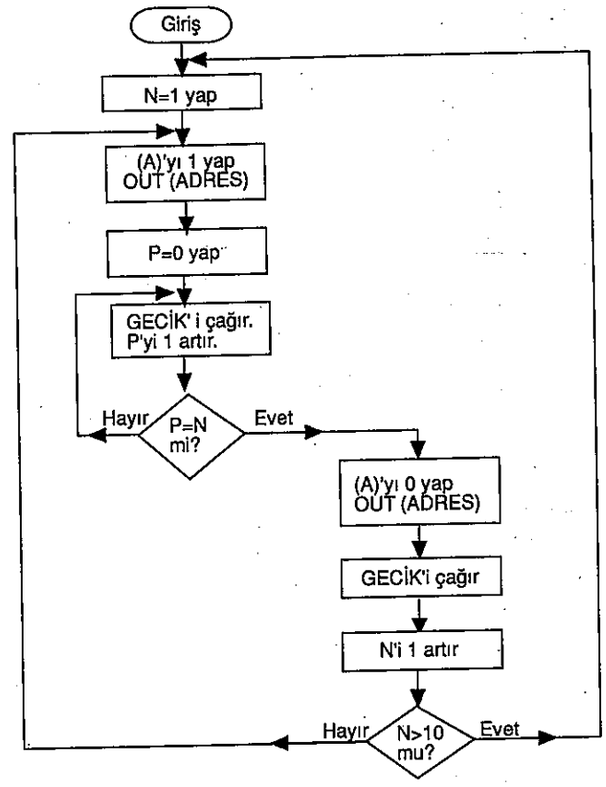
```
        MVI   A, 00    ;(A) = 0 yap
TEKR:   MOV   M, A     ;Bellek adresini (H,L)'yi 0'a kur
        INX   H        ;(H,L)'yi 1 artır
        MOV   A, H     ;(H)'yi A kaydedicisine yerleştir
        CMP   B        ;B ile karşılaştır
        JNZ   TEKR     ;H = B ise TEKR'ye atla
        MOV   A, L     ;(L)'yi A'ya yerleştir
        CMP   C        ;C ile karşılaştır
        JNZ   TEKR
        MOV   M, A     ;Son adresi sil
        RET            ;Geriye dön
```

7.7 *n* değerinin, H,L kaydedici çiftindeki altyordama aktarıldığını ve elinizde bir CARP altyordamının olduğunu varsayın. İşlenenler, CARP'a B ve C kaydedicilerine, sonuç ise B ve C kaydedici çiftine aktarılır:

```
MVI     A, 0B   ;Xn'in adresini bul
ADD     H
MOV     H, A
MOV     A, M    ;Xn'yi A'ya getir
MOV     B, A    ;Xn'yi B ve C'ye yerleştir
MOV     C, A
CALL    CARP    ;CARP altyordamını çağır
MOV     L, A    ;Xn'yi L kaydedicisine yerleştir
MVI     H, 00   ;(H) = 0 yap
INX     H       ;H,L'deki 1+Xn'i hesapla
DAD     B       ;H,L'e Xn^2'yi ekle
RET             ;Geriye dön
```

Bu altyordamın sonucu, H,L kaydedici çiftinde tutulur.

7.8 Gecikme altyordamı *Ts*'lik bir zaman gecikmesi üretsin. 'ADRES' adresli portun en küçük değerlikli bitinden darbe dizisi geldiğini varsayın. Programın akış diyagramı Şekil R'de gösterildiği gibidir.



Şekil R

8.1 Bkz: Kısım 8.1, 8.2 ve 8.3.

8.2 Bkz: Kısım 8.1 ve 8.2.

8.3 Bkz: Kısım 8.4.

8.4 Bkz: Kısım 8.6.

8.5 Bkz: Kısım 8.3 ve 8.8. RST 5.5 kesmesi için, yani 002C adresindeki konuma bir komut, JMP 00B0, yerleştirilmeli ve kesme hizmet yordamında, işlemci kaydedicilerinin içeriklerini saklamak için komutlar bulunmalıdır. Bu nedenle program, bu verilenlere benzer komutlarla başlamalıdır:

```
PUSH B          [(B) ve (C)'yi yığına at]
PUSH D          [(D) ve (E)'yi yığına at]
PUSH PSW        [(A) ve (durum)'u yığına at]
```

Hizmet yordamının sonunda bu değerler, şu komutlarla geriye alınırlar:

```
POP PSW         (A ve durum kaydedici değerlerini geriye al)
POP D           (D ve E kaydedici değerlerini geriye al)
POP B           (B ve C kaydedici değerlerini geriye al)
```

Kesme sistemini denetleyebilmek için birkaç alternatif vardır. Herhangi bir kesme kabul edildiğinde, tüm kesmeler yetkisizlenir. Hizmet yordamının yürütülmesi boyunca da bu şekilde kalırlar ve sonra:

```
MVI     A, 08
SIM
EI
```

komutları ile tekrar yetkilenirler.

Alternatif olarak, RST 5.5 yalnızca:

```
MVI     A,0E
SIM
EI
```

komutları ile yeniden yetkilenebilir. Diğer bir olasılık ise, RST 5.5 hizmet yordamının başında özel kesmeleri tekrar yetkilendirmektedir.

8.6 Bkz: Kısım 8.7.

8.7 Bkz: Kısım 8.3, 8.5 ve 8.7.

8.8 Bkz: Kısım 8.8.

8.9 RST 5.5 hizmet yordamı, RST 6.5 ve RST 7.5 kesmelerinin meydana gelmesine izin vermelidir.
RST 6.5 hizmet yordamı, RST 7.5 kesmelerinin meydana gelmesine izin vermelidir.
RST 7.5 hizmet yordamı, RST 5.5 kesmelerinin meydana gelmesine izin vermelidir.
Kesme denetim komutları:

Ana program-tüm maskeleme bitlerini sıfırlar ve tüm kesmeleri yetkilendirir:

```
MVI     A,08
SIM
EI
```

RST 5.5 hizmet yordamı-RST 6.5 ve RST 7.5 kesmelerini yetkilendirir ve RST 5.5 kesmesini yetkisizler:

```
MVI     A,09
SIM
EI
```

RST 6.5 hizmet yordamı-RST 7.5'i yetkilendirir ve RST 5.5 ve RST 6.5'u yetkisizler:

```
MVI     A,0B
SIM
EI
```

RST 7.5 hizmet yordamı-RST 5.5'u yetkilendirir, RST 6.5 ve RST 7.5'u yetkisizler:

```
MVI     A,0E
SIM
EI
```

TRAP kesmesi, herhangi bir hizmet yordamını otomatik olarak kesebilir. Kesmeler, TRAP kesme hizmet yordamının başında yetkilenirler. TRAP dahil olmak üzere tüm kesmeler her bir yordamın bitiminde yetkilenmez.

8.10 Bkz: Kısım 8.8.

8.11 Bkz: Kısım 8.8.

8.12 Güç kaynağı arızası en kritik kesmedir, zamanlayıcı kesmesi ikinci en kritik ve bant okuyucusundan gelen kesme de en az kritik kesmelerdir. Bu nedenle, güç kaynağı arızası için TRAP kesmesini, zamanlayıcı için RST 7.5 ve okuyucu için de RST 6.5 kesmesi kullanılacak şekilde düzenleyin. O zaman, TRAP, RST 7.5 ve RST 6.5 kesmelerini yetkilendiren ana program:

```
MVI     A,09
SIM
EI
```

Her ne zaman bir kesme yordamı çalıştırılırsa, daha yüksek öncelikli kesmeler tekrar yetkilendirilirler.

Bu nedenle, RST 6.5 hizmet yordamı:

```
MVI     A,0B
SIM
EI
```

RST 7.5 hizmet yordamı:

```
MVI     A,0F
SIM
EI
```

ve TRAP hizmet yordamı:

```
MVI     A,0F
SIM
EI
```

9.1 Bkz: Kısım 9.1, 9.2 ve bir önceki bölüm.

9.2 Bkz: Kısım 9.2.

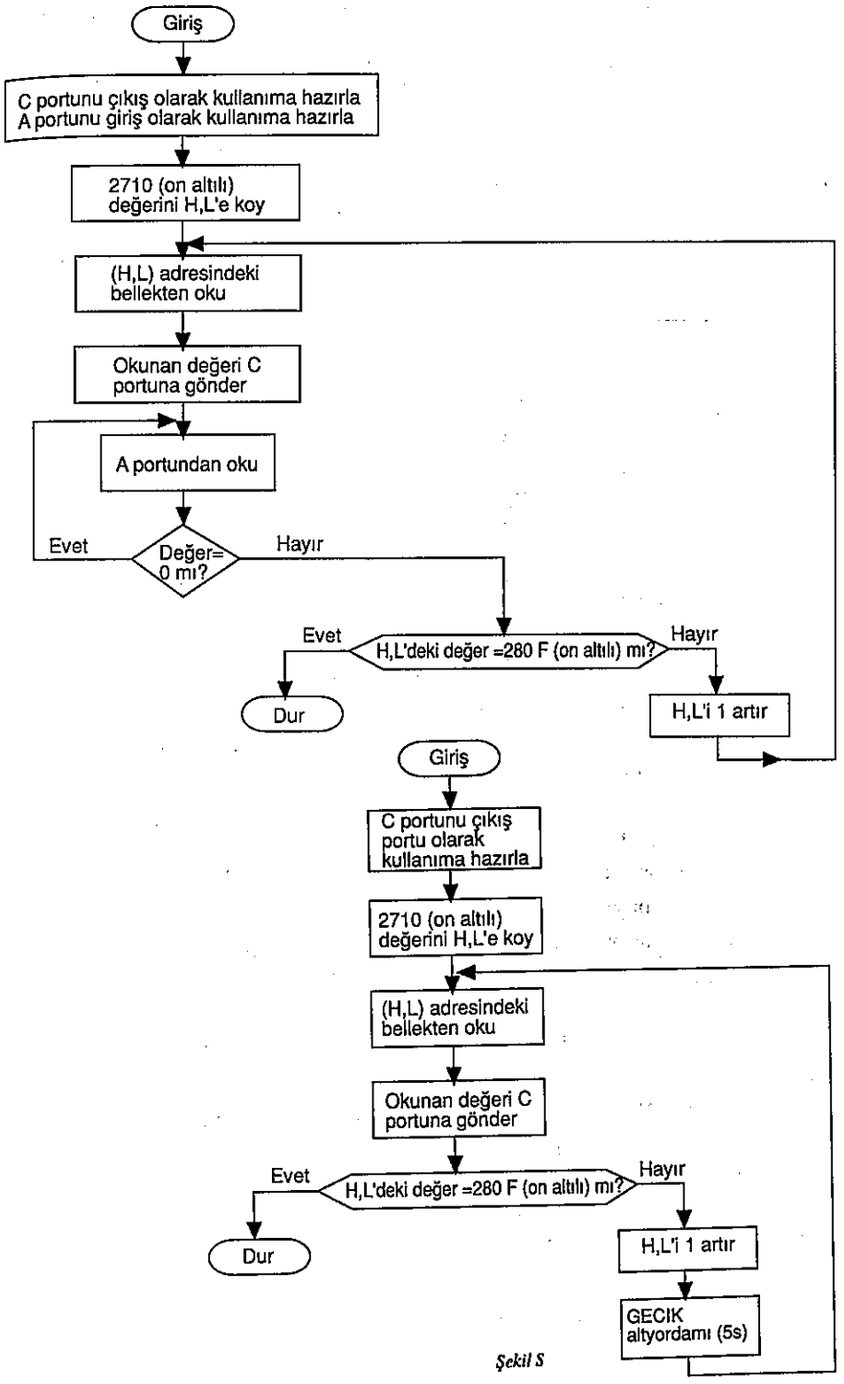
9.3 Bkz: Kısım 9.2.

9.4 Işık yayan diyotlar, bir PIC çıkış portunun, örneğin C portunun sekiz bitine bağlanabilir, bkz. Şekil 9.5. 10.000-10.255 [2710 (onaltılı)-280F(onaltılı)] bellek konumlarını inceleyen programın akış diyagramı Şekil S'de gösterilmektedir. Burada, C portu çıkışının bir sabitleyici içerdiği varsayılmıştır.

9.5 Mümkündür. Basma-düğmesi, giriş olarak hazırlanmış olan A portunun en küçük değerlikli hattına bağlanabilir (bkz: Şekil 9.5). PIC'in 1. modu kullanılırsa, düğmeye basıldığında A portundaki bir bit 1 olacak ve işlemci, porttan okuma yaptığında sıfırlanacak biçimde düzenlenebilir. Programın akış diyagramı Şekil T'de gösterildiği gibidir.

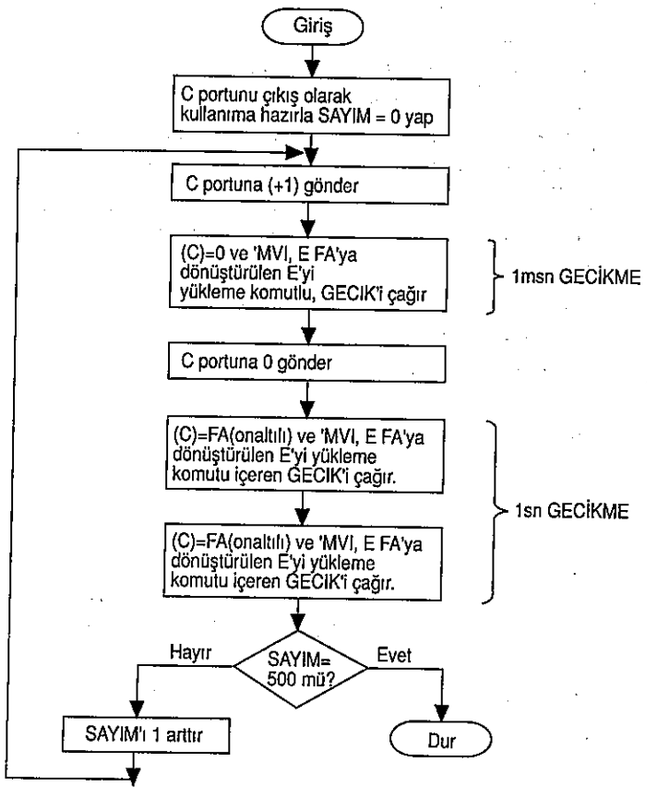
9.6 Tüm portların 0 modunda olduğunu varsayın. PIC denetim kaydedicisine yerleştirilen bit deseni, 10010010 olabilir. Bu örnekte, C portunu denetleyen bitler ilgisizdir. Kod:

```
MVI     A,92
OUT     (adres, denetim kaydedicisi)
IN      (adres, A portu)
MOV     B,A
IN      (adres, B portu)
MOV     C,A
```

Şekil T

Şekil S



Şekil U

Bu örnekte, 'adres, denetim kaydedicisi':

(a) PIC'in kendisini ve

(b) PIC'in içindeki denetim kaydedicisini seçmek için yola çıkarılan adrestir.

'Adres, A portu' ve 'Adres, B portu' da benzer biçimdedir.

9.7 A ve B portlarını giriş ve C portunu çıkış olarak kulanıma hazırlamak için, denetim kaydedicisine yerleştirilen kod 10010010 olmalıdır. Bu yüzden:

```
MVI     A,92
OUT     (adres, denetim kaydedicisi)
IN      (adres, A portu)
MOV     B,A
IN      (adres, B portu)
```



```
ADD     B
OUT     (adres, C portu)
```

9.8 Işık yayan diyot, çıkış olarak kullanıma hazırlanmış olan C portunun en küçük değerlikli bitine bağlanmış olsun. 7. Bölümün zaman-gecikme alt-yordamını düşünün. İç döngünün bir saykılı ile üretilen gecikme, bu döngü 4 komut içerdiği için, 8 µsn'dir. Bu nedenle, bu döngü 125 kere çevrimlenirse, gecikme süresi (125x8) µsn= 1 msn olur. Bu döngü 250 kere ve dış döngü de 250 kere saykıllanırsa, toplam gecikme (8x250x250) µsn = 500 msn olacaktır.

Dış döngüdeki komutlar tarafından gelen fazladan süre çok küçük olduğu için ihmal edilebilir.

Böylece programın akış diyagramı Şekil U'da gösterildiği gibi olabilir.

9.9 Bkz: Kısım 1 ve 9.4.

9.10 Bkz: Kısım 9.2 ve 9.4.

9.11 Bkz: Kısım 9.4.

9.12 Bkz: Kısım 9.4.

9.13 Kullanılan saat frekansı, kare dalgada gerekli seçme gücüne, yani frekansın değiştirilebileceği adımların büyüklüğüne bağlıdır. Bu adımlar 2sn'lik ise:

Saat frekansı = 500 kHz
500 µsn için sayıcı değeri = 250
100 msn için sayıcı değeri = 50.000

1 sn'lik bir çözünürlük değeri, 100 msn'lik bir gecikme için 64K'dan daha büyük bir sayıcı değerine ihtiyaç duyacağından, mümkün değildir.

Sayıcıyı ayarlamak için:

```
MVI     A, 30   ;Denetim sözcüğünü sayıcıya gönder- Bkz:
OUT     13      ;Kısım 9.5
MVI     A, FA   ;Sayıcıyı 00FA (250)'ye ayarla
OUT     10
MVI     A, 00
OUT     10
        ;Diğer program basamakları
```

```
MVI     A, 50   ;Sayıcı değerini C350 (50.000)'e ayarla
OUT     10
MVI     A, C3
OUT     10
.
.
```

9.14 Anahtarların bir PIC'in A portunun en küçük değerlikli dört bitine ve LED'lerin de C portunun en küçük değerlikli dört bitine bağlandığını varsayın. Anahtar konumları ve LED durumları arasında Tablo A'da gösterildiği gibi bir ilişki vardır (bir anahtar kapalı olduğunda, uygun sütuna bir '1' ve bir LED yanıyor olduğunda bir '1' girilir).

Tablo A

Anahtarlar				*LED'ler*				Adres
A	B	C	D	L	M	N	O	(onaltılı)
0	0	0	0	0	0	0	0	0B00
0	0	0	1	0	0	0	0	0B01
0	0	1	0	1	0	0	0	0B02
0	0	1	1	1	0	0	0	0B03
0	1	0	0	0	0	0	1	0B04
0	1	0	1	0	0	0	0	0B05
0	1	1	0	1	1	0	0	0B06
0	1	1	1	1	0	0	0	0B07
1	0	0	0	0	0	0	1	0B08
1	0	0	1	0	0	0	0	0B09
1	0	1	0	1	0	0	0	0B0A
1	0	1	1	1	0	0	0	0B0B
1	1	0	0	0	0	0	1	0B0C
1	1	0	1	0	0	0	0	0B0D
1	1	1	0	1	1	1	0	0B0E
1	1	1	1	1	0	0	0	0B0F

L,M,N ve O değerlerine ilişkin tablo, bellekte uygun bir yerde, örneğin gösterildiği gibi 0B00-0B0F adreslerinde saklanır. Program:

```
MVI     H, 0B       ;H,L kaydedici çiftine 0B00 adresini ver
MVI     L, 00       ;C çıkış portu, A giriş portu
MVI     A, 92       ;olacak şekilde PIC'i
OUT     (Adres      ;kullanıma
        denetim     ;hazırla
        kaydedicisi)
```



```
TEKR:   IN      (adres,     ;Anahtarları oku
                A portu)
        ADD     L           ;Anahtarlardaki değere
                            ;L kaydedicisine ekle
        MOV     L, A
        MOV     A, M        ;Tablodan değer al
        OUT     (adres,     ;Işıklara çık
                C portu)
        MVI     L, 00       ;(L)'yi bir sonraki adım için sıfırla
        JMP     TEKR        ;Tekrar et
```

10.1 Bkz: Kısım 10.2.

10.2 Bkz: Kısım 10.2 ve 10.3.

10.3-10.8 Bu bölümdeki bu altı soru, mikrobilgisayarın bir kronometre olarak kullanımını göstermek için programların geliştirilmesi ve çalıştırılması ile ilgilidir. Bu konu bu bölümün metin kısmında etraflıca anlatılmış (Kısım 10.4) ve Soru 10.3'deki işlem için Intel 8085 çeviri dilinde örnek program kodu verilmiştir.

Bu sorular için üretilen gerçek kod, üretilen programları çalıştırmak için kullanılan mikrobilgisayar tipi ile belirlenecektir. Bu nedenle, Kısım 10.4'deki açıklamaları, deneyleriniz için bir temel olarak kullanmak, ancak program kodunu hangi mikrobilgisayarda çalıştırmanız gerekiyorsa, ona uyarlamak suretiyle soruları çözmeye çalışmanız önerilir.

Büyük bir olasılıkla, geliştirilen programların doğruluğu (en azından ana çalışması itibarıyla), hemen belli olacaktır. Diğer bir deyişle, hatalı programlar ya hiç çalışmayacaklar ya da hatalı olduğunu gösteren ilgisiz sonuçlar üreteceklerdir. Bununla birlikte, bu tür hatalarınızı bulmanızı ve düzeltmenize yardımcı olması için bir uzmanın yardımına ihtiyaç duyabilirsiniz. Bu nedenle, programlarınızın denetim altında çalıştığından emin olmanız önerilir. Ayrıca sonunda programınız çalıştığında, belirlenen zamanlama işlemlerinin gerçekleştirileceği doğruluğu ölçmeye ve mümkün olduğu kadar duyarlı ve kesin olana kadar iç parametrelerini ayarlamaya dikkat etmek gerekir. İpuçları:

10.4 Aralık zamanlayıcısı her 0.1 sn'de bir kesme üretir. Bunlar zamanlama için kullanılır; bu nedenle gecikme alt yordamı kullanılmayabilir. Program, her

bir zamanlayıcı kesmesi meydana geldiğinde sayımı 1 artırır. Bu sayım ona vardığında, SANY 1 artırılır. Bu çalışma yönteminde, program komutlarının alacağı toplam süre, mevcut aralıkta komutlar yürütülebildiği sürece, önemli değildir.

10.5 Kronometreye yeni bir 'tur süresi' düğmesi eklenmesi önerilir. Bu düğmeye basmak bir kesme yordamına, KESME, girişe neden olmalıdır. DEVAM '1'de iken, KESME'ye girilirse, kronometre çalışmaya devam etmeli (SANY ve DAKK olarak sayarak), fakat görüntülenen zaman değeri dondurulmalıdır. KESME tarafından ayarlanan ve sıfırlanan yeni bir değişken, 'DONDUR', bunu denetlemek için kullanılabilir.

10.9 Giriş denetim düğmeleri, basıldığında mikrobilgisayara kesme gönderecek şekilde bağlanmalıdır. 'Mod' düğmesi kesme hizmet yordamı, her etkinleştirmede 'MOD' değişkenini 1 artırır. MOD 2'yi geçerse sıfırlanır. Böylece:

MOD=0 'normal zaman göstergesi'
MOD=1 'alarm ayarı'
MOD=2 'zaman ayarı' demektir.

'Saat' düğmesine her basılışında, karşılık gelen kesme hizmet yordamı MOD'un değerini test eder. Bu değer 2 ise, normal zaman sayımının saat ayar değeri 1 artırılır. 1 ise, alarm zamanı sayımının saat ayar değeri 1 artırılır. 0 ise, hiçbir işlem yapılmaz.

'Dakikalar' düğmesi de benzer biçimde çalışır. Böylece üç kesme hizmet yordamı olmaktadır. Bunları çalıştırmak için, RST 5.5, RST 6.5 ve RST 7.5 kullanılabilir. Çıkış göstergesi, daha önce anlatılmış olan kronometreninkine benzer. 4-basamak gerektiren 24-saat göstergesi kullanılabilir. Alarm tek bir çıkış hattı ile çalıştırılabilir. Alarmın belirli bir sayıda (sözgelimi 10) ve belli periyotlarla (sözgelimi 1 sn açık 1 sn kapalı) açılıp kapanmasına neden olacak, kısa bir program kodu bölümü geliştirilmesi önerilir.

İndeks

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 bit 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 (Teş 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 alışverişini 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'ların sayısının tek ya da çift olmasına göre 0 veya 1 değeri 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 elemanını 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 bellekten 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.

İşkodu (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 bellekten 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 oskit 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, işlenenler 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 sinyalleri, 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 program 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 anlm. 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 yörüngelerin- den 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.

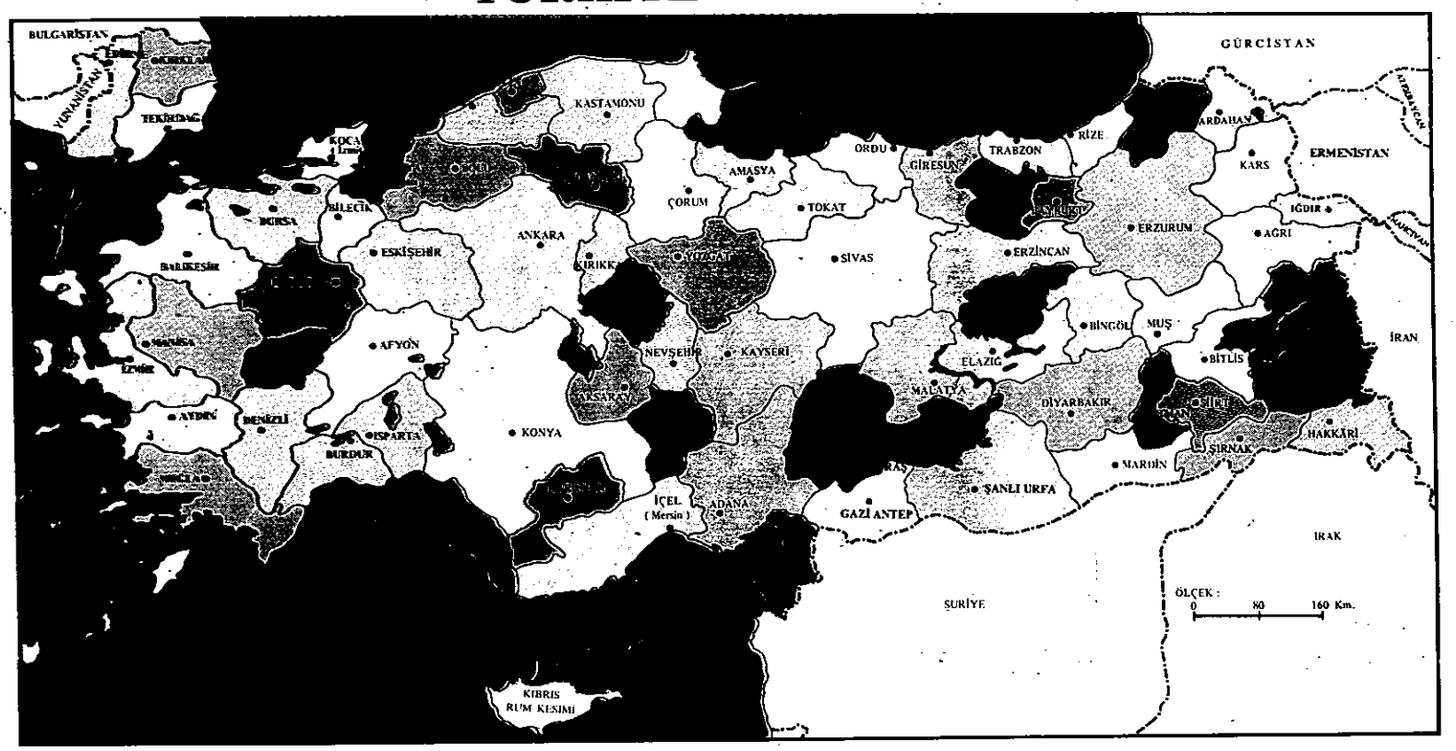
Yığın (Stack): Verileri geçici bir süreyle -örneğin, çıkışta geriye alınak üzere bir altyordama girerken- saklandığı ve bilgiler, belirli bir sırada giriş-çıkış yapacak şekilde düzenlenmiş bellek alanı.

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