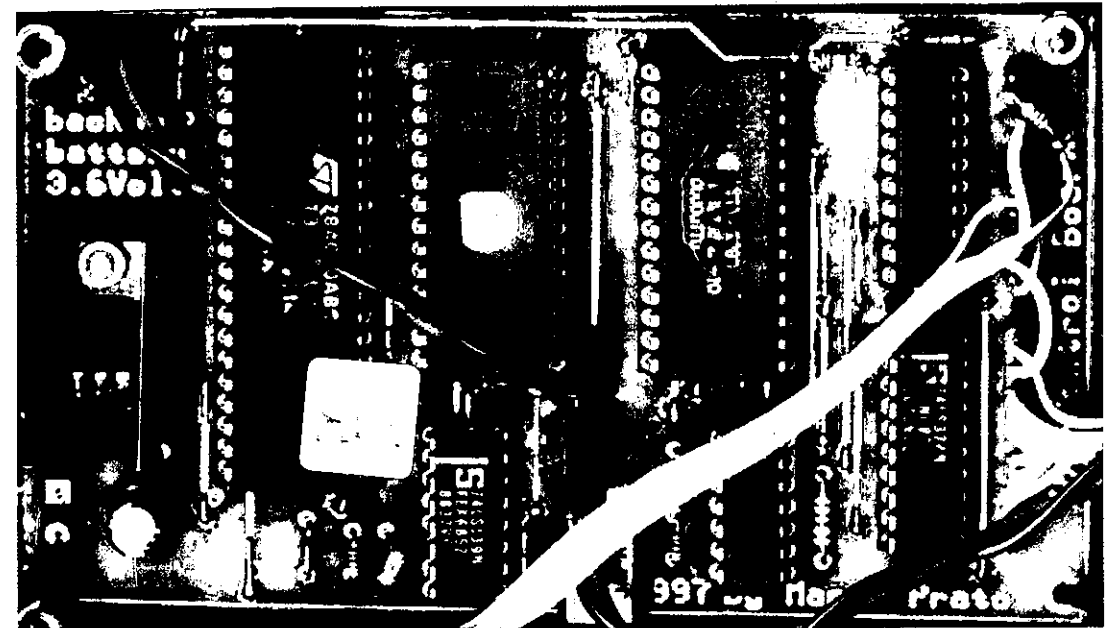
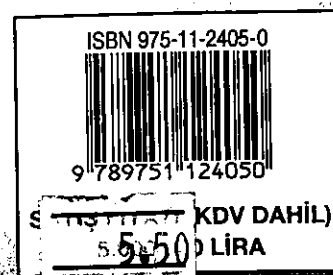


MİKRO İŞLEMCİLER

TEMEL DERS KİTABI



MİKRO İŞLEMÇİLER



3821

867

**MESLEKİ VE TEKNİK ÖĞRETİM
OKULLARI İÇİN**

Serif Balta
11-Nisan 06

**MİKROİŞLEMCİLER
TEMEL DERS KİTABI**

YAZAN
Mehmet HOŞGÖREN



DEVLET KİTAPLARI

BİRİNCİ BASKI

SERLER-İstanbul.2003

Her hakkı saklıdır ve Millî Eğitim Bakanlığı'na aittir. Kitabın metin, soru ve şekilleri kısmen de olsa hiçbir surette alınıp yayımlanamaz.

ISBN: 975-11-2405-0

Millî Eğitim Bakanlığı Talim ve Terbiye Kurulunun 23.05.2003 gün ve 157 sayılı kararı ile ders kitabı olarak kabul edilmiş, Yayınlar Dairesi Başkanlığı'nın 18.07.2003 gün ve 5811 sayılı yazıları ile birinci defa 20.000 adet basılmıştır.



İSTİKLÂL MARŞI

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

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

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

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

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

Basın yerleri "toprak" diyerek geçme, tanı!
Düşün altındaki binlerce kefensiz yatanı.
Seni selim toğlusun, 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,
Etnesin tek vatanımdan beni dünyada cüdâ.

Ruhumun senden, İlahi, 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 cerihamdan, İlahi, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerred gibi yerden nâ'sım;
O zaman yükselerek arşa değer belki başım.

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

Mehmet Âkif ERSOY

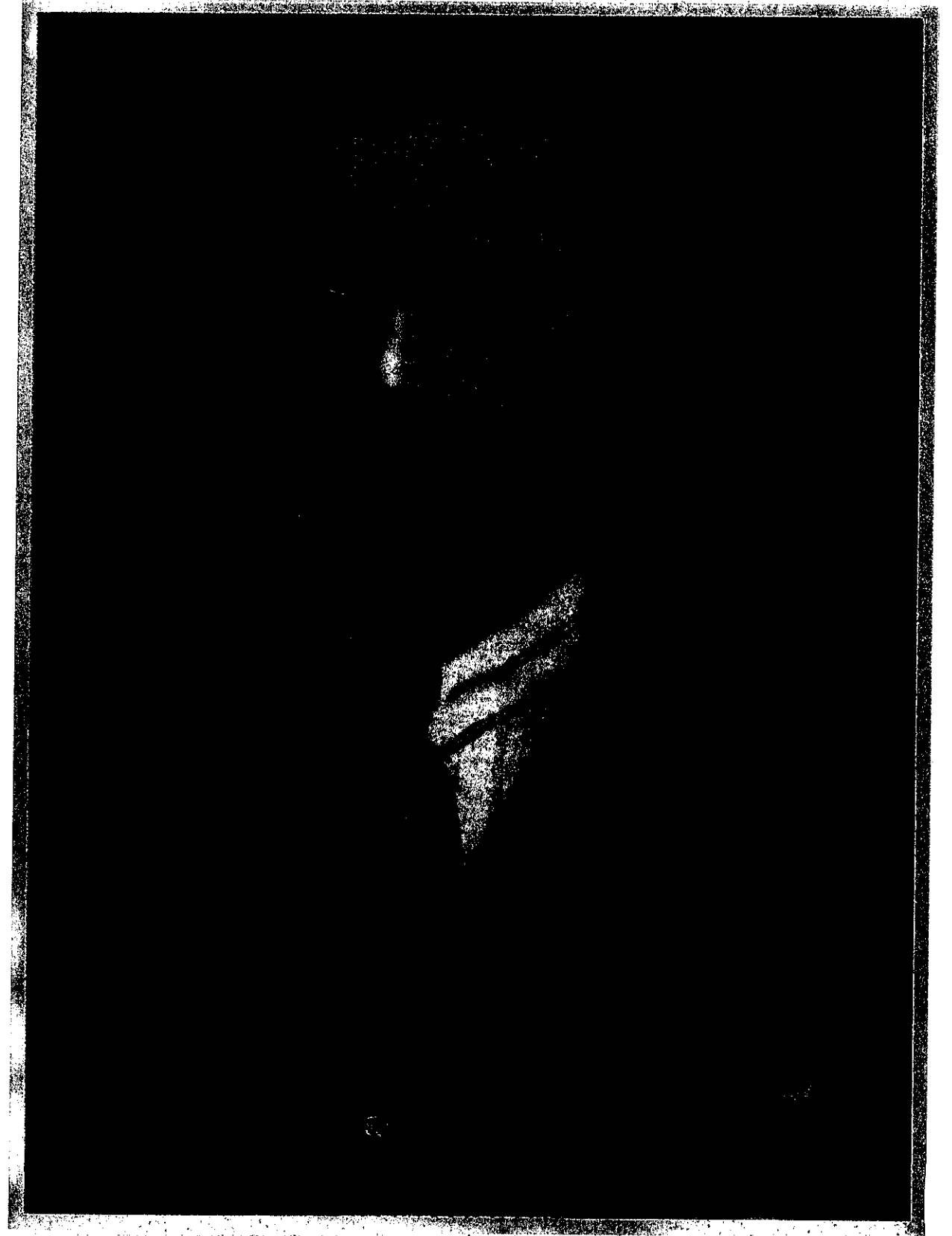
ATATÜRK'ÜN GENÇLİĞE HITABESİ

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 hâzineden, 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 nâmûsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedece düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın, bütün kaleleri zaptedilmiş, 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 fakru 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 asil kanda mevcuttur!

K. Atatürk



Mustafa Kemal ATATÜRK (1881-1938)

ÖNSÖZ

“Bilgi”nin stratejik öneme sahip olduğu bir çağda yaşıyoruz. “Bilgi”yi üreten ve onu yaşamın her alanında kullanan toplumların diğerlerinden bir adım önde olduğu yadsınamaz bir gerçek...

Dünya devletleri, “En iyi yatırım, insana yapılan yatırımdır.” ilkesinden hareketle “bilimsel ve teknik eğitim”i millî hedef olarak kabul edip çalışmalarını bu doğrultuda sürdürmektedirler. Bu sebeple mesleki ve teknik eğitim öğrencilerine bilimsel ve teknolojik alandaki gelişmelere paralel olarak çağın gerektirdiği bilgi ve becerileri kazandırmak, “yaparak ve yaşayarak öğrenme” sürecini destekleyecek materyalleri seviyelerine uygun bir şekilde sunmak günümüzün vazgeçilmez şartları arasında yer almaktadır.

Günlük hayatta kullanılan ve klasik elektronik devreler içeren birçok cihaz, yerini daha fonksiyonel, programlanabilir, başka cihaz ve bilgisayarlarla iletişim kurabilen, arıza ve periyodik bakım zamanlarını kullanıcıya bildirecek şekilde tasarlanan cihazlara bırakmıştır. Bilgisayarlar, cep telefonları, yazıcılar, fotokopi makineleri, telefon santralleri, uydu alıcısı ve VCD player gibi günlük hayatımızın her alanına girmiş cihazlarda mikroişlemciler kullanılmaktadır. Bunların yanında mikroişlemciler; otomasyon sistemleri, robotlar, PLC’ler gibi bir çok endüstriyel uygulama alanlarında kullanılmaktadır. Bu sebeple mikroişlemcileri programlamayı bilen, mikroişlemci ile devre tasarlayan, mikroşlemcili sistemlerin arıza ve bakımlarını yapabilen elemanların yetiştirilmesi önem kazanmıştır.

“Mikroişlemciler Temel Ders Kitabı” bu amaçlara uygun olarak Milli Eğitim Bakanlığımızın Mesleki ve Teknik Öğretim okullarına bağlı Anadolu Teknik, Teknik, Anadolu Meslek ve Meslek liselerinin; Bilgisayar, Elektronik, Endüstriyel Elektronik, Hidrolik Pnömatik, Tıp Elektroniği, Telekomünikasyon ve CNC bölümlerinde okutulan “Mikroişlemciler” dersinin müfredatını kapsayacak şekilde hazırlanmıştır.

Kitapta ele alınan konular açıklayıcı şekil ve resimlerle desteklenmiş, teorik bilgilerin ve program örneklerinin okullarımızdaki mikroşlemci deney setleriyle uygulanmasına imkan sağlayacak örneklerle yer verilmiştir. Konuların sonundaki değerlendirme çalışmaları anlatılan bilgilerin pekişmesine yardımcı olacaktır. Bu kitaptan ve mikroşlemci dersinden verim almanın; temel bilgisayar programlama, dijital elektronik ve elektronik bilgisine sahip olmaya bağlı olduğu dikkate alınmalıdır.

Meslektaşlarımıza ve öğrencilerimize yararlı olması dileğiyle...

Ocak 2003
Nallıhan / ANKARA

Mehmet HOŞGÖREN
Elektronik Öğretmeni
mehmet_hosgoren@hotmail.com

İÇİNDEKİLER

ÜNİTE 1 SAYI SİSTEMLERİ	1
1.1 İKİLİK (BINARY) SAYI SİSTEMİ	1
1.2 ONALTILIK (HEXADECIMAL) SAYI SİSTEMİ	1
1.3 SAYI SİSTEMLERİNİN BİRBİRİNE DÖNÜŞTÜRÜLMESİ	1
1.3.1 Onluk Sayıların İkilik ve Onaltılık Sayılara Çevrilmesi	1
1.3.2 İkilik ve Onaltılık Sayıların Onluk Sayılara Çevrilmesi	2
1.3.3 İkilik Sayıların Onaltılık Sayılara Çevrilmesi	3
1.3.4 Onaltılık Sayıların İkilik Sayılara Çevrilmesi	3
1.4 NEGATİF VE KESİRLİ SAYILARIN İKİLİK SAYI SİSTEMLERİNDE İFADE EDİLMESİ	4
1.4.1 Negatif Sayıların İkilik Tabandaki Gösterimi	4
1.4.2 Ondalık Sayıların İkilik Tabandaki Gösterimi	6
1.5 BİLGİSAYAR SİSTEMLERİNDE ONDALIKLI SAYILARIN TAM SAYI OLARAK İFADESİ	6
ÜNİTE 2 İNFORMASYONLARIN BİTLERLE İFADE EDİLMESİ	9
2.1 BİT, NIBBLE, BYTE VE WORD KAVRAMLARI	9
2.2 BİLGİSAYARDA BİLGİ BİRİMİ	11
2.3 VERİ İLETİM HIZI	12
2.4 ASCII KODU YARDIMLARIYLA ALFASAYISAL İŞARETLERİN İFADE EDİLMESİ	13
2.5 KOMUTLARIN KODLARLA İFADE EDİLMESİ	15
ÜNİTE 3 MİKROİŞLEMCİLERE GİRİŞ	19
3.1 MİKROİŞLEMCİ TANIMI	19
3.2 MİKROİŞLEMCİ BLOK DİYAGRAMI	20
3.3 MİKROİŞLEMCİ ÇEŞİTLERİ	21
3.3.1 32 Bit Mikroşlemciler	23
ÜNİTE 4 8 BİT VE 16 BİT MİKROİŞLEMCİLERİN İÇ YAPISI	24
4.1 MİKROİŞLEMCİLERİN İÇ YAPISI	24
4.2 KAYDEDİCİLER	26
4.2.1 Kaydediciler	26
4.2.2 Genel Amaçlı Kaydediciler (General Purpose Registers)	28
4.2.3 Özel Amaçlı Kaydediciler (Special Purpose Registers)	28
4.2.3.1 Program sayıcı (PC, Program Counter)	28
4.2.3.2 İndeks kaydedicisi (IX, Index Registers)	29
4.2.3.3 Yığın göstericisi (SP, Stack Pointer)	29
4.2.3.4 Durum kodu kaydedicisi (CCR, Condition Code Register)	30
4.2.3.5 Kesme vektörü (I, Interrupt register)	33
4.2.3.6 Hafıza tazeleme kaydedicisi (R, Refresh Memory)	33
4.2.4 Segment Kaydediciler	33
4.2.4.1 Offset ve segment nedir?	34
4.3 ARİTMETİK MANTIK BİRİMİ (ALU)	36
4.3.1 Aritmetik Mantık Biriminin İşlevi	36
4.3.2 Aritmetiksel İşlemler Kavramı	36
4.3.3 Mantıksal İşlemler Kavramı	37
4.4 BUFFER VE MULTIPLEXER	38
4.5 KONTROL VE DENETİM BİRİMİ	40
4.6 İLETİŞİM HATLARI (BUSSES)	40
4.6.1 Adres Yolu (Address Busses)	41
4.6.2 Veri Yolu (Data Busses)	42
4.6.3 Kontrol Yolu (Control Busses)	42
ÜNİTE 5 KOMUT ÇALIŞMA EVRELERİ	44
5.1 KOMUTUN TANIMI VE FORMATI	44
5.2 KOMUT İŞLEME AŞAMALARI	46
5.2.1 Komutun Bulunup Getirilmesi (Fetching)	46
5.2.2 Komutun çözülmesi (Decoding)	47

5.2.3	Komutun uygulanması (Executing)	48
5.3	KOMUT VE ZAMAN İLİŞKİSİ	48
5.3.1	Komutun bulunup getirilmesi ve zaman ilişkisi (grafiği)	50
5.3.2	Okuma komutu ve zaman ilişkisi (grafiği)	51
5.3.3	Yazma komutu ve zaman ilişkisi (grafiği)	52
5.3.4	I/O (Okuma/Yazma) zaman ilişkisi (grafiği)	53
ÜNİTE 6	MİKROİŞLEMCİLERİN DİŞ YAPISI	55
6.1	HAFİZALAR	55
6.1.1	Manyetik Hafızalar	55
6.1.1.1	Manyetik hafızalar	55
6.1.1.2	Teyp bandı	56
6.1.1.3	Hard disk	56
6.1.1.4	Floppy disk	58
6.1.2	Statik Hafızalar	59
6.1.2.1	ROM'lar (Read Only Memory)	59
6.1.2.2	RAM'lar (Random Access Memory)	60
6.1.3	Dinamik Hafızalar	60
6.1.4	CD Hafızalar	61
6.2	GİRİŞ/ÇIKIŞ (INPUT/OUTPUT) ÜNİTELERİ	62
6.2.1	Paralel Giriş/Çıkış Üniteleri	62
6.2.2	Seri Giriş/Çıkış Üniteleri	62
6.3	MİKROBİLGİSAYAR SİSTEMİNDE KULLANILAN ENTEGRELERİN KATALOG BİLGİLERİNİN İNCELENMESİ	65
6.3.1	Motorola MC6802 Mikroişlemcisinin Özellikleri	66
6.3.2	Zilog Z80 Mikroişlemcisinin Özellikleri	67
6.3.3	Mos Tec. R6502 Mikroişlemcisinin Özellikleri	69
6.3.4	Motorola MC6821 Paralel Giriş/Çıkış Entegresinin Özellikleri	70
6.3.5	Intel 8255 Paralel İletişim Entegresinin Özellikleri	71
6.3.6	Intel 8251A Seri İletişim Entegresinin Özellikleri	72
6.3.7	Motorola 6850 Seri Giriş/Çıkış Entegresinin Özellikleri	73
6.3.8	2764 EPROM'un Özellikleri	74
6.3.9	6264 RAM'm Özellikleri	75
6.3.10	RS232 uygunlaştırıcı Max232 Entegresinin Özellikleri	75
ÜNİTE 7	8080/8085, 8086/8088 VE 80286/80386 MİKROİŞLEMCİLERİ	77
7.1	INTEL 8080/8085 MİKROİŞLEMCİLERİ	77
7.2	INTEL 8086/8088 MİKROİŞLEMCİLERİ	79
7.3	INTEL 80286 MİKROİŞLEMCİSİ	84
7.3.1	80286 Mikroişlemcisinin Teknik Özellikleri	84
7.3.2	80286 Mikroişlemcisinin Sistem Konfügrasyonu	87
7.4	INTEL 80386 MİKROİŞLEMCİSİ	88
7.4.1	80386 Mikroişlemcisinin Teknik Özellikleri	88
ÜNİTE 8	MİKROİŞLEMCİ KOMUTLARI	91
8.1	MİKROİŞLEMCİ KOMUTLARI	91
8.2	YÜKLEME KOMUTLARI	91
8.2.1	8 Bitlik Yükleme Komutları	93
8.2.2	16 Bitlik Yükleme Komutları	95
8.2.3	Takas Komutları	98
8.2.4	Blok Transfer Komutları	99
8.2.5	Blok arama komutları	102
8.3	ARİTMETİK VE MANTIK İŞLEM KOMUTLARI	105
8.3.1	8 Bitlik Aritmetik Ve Mantık İşlem Komutları	105
8.3.2	16 Bitlik Aritmetik Ve Mantık İşlem Komutları	107
8.3.3	Genel Amaçlı Aritmetik Komutları	108
8.4	TEST VE DALLANMA KOMUTLARI	110
8.4.1	Şartlı Dallanma Komutları	110

8.4.2	Şartsız Dallanma Komutları	113
8.5	KAYDIRMA VE DÖNDÜRME KOMUTLARI	114
8.5.1	Sola Kaydırma Komutları	114
8.5.2	Sağa Kaydırma Komutları	114
8.5.3	Sola Döndürme Komutları	115
8.5.4	Sağa Döndürme Komutları	116
8.6	BİT İŞLEME KOMUTLARI	116
8.7	GİRİŞ/ÇIKIŞ KOMUTLARI	118
8.7.1	Giriş/Çıkış Komutları	118
8.8	ALT PROGRAMLARA VE HİZMET PROGRAMLARINA İLİŞKİN KOMUTLAR	119
8.8.1	Bilgisayar Programlamada Alt Program Yöntemi	119
8.8.2	Alt Program Komutları	121
8.9	MİKROİŞLEMCİLERDE YIĞIN İŞLEMLERİ VE KOMUTLARI	122
8.9.1	Yığın Çalışma İlkeleri	122
8.9.2	Yığın İşleme Komutları	123
8.10	KONTROL KOMUTLARI	125
ÜNİTE 9	ADRESLEME MODLARI	127
9.1	ADRESLEME MODU KAVRAMI	127
9.2	İVEDİ ADRESLEME (IMMEDIATE ADDRESSING)	130
9.3	KAYDEDİCİ VE İMALİ ADRESLEME (ACCUMULATOR AND IMPLIED ADDRESSING)	131
9.4	DOĞRUDAN ADRESLEME (DIRECT ADDRESSING)	132
9.5	DOĞAL ADRESLEME (ABSOLUTE ADDRESSING)	134
9.6	SIRALI ADRESLEME (INDEX ADDRESSING)	135
9.7	BAĞIL ADRESLEME (RELATIVE ADDRESSING)	137
9.8	DOLAYLI ADRESLEME (INDIRECT ADDRESSING)	139
ÜNİTE 10	INTERRUPT (KESME) VE RESTART (YENİDEN BAŞLATMA)	141
10.1	INTERRUPT (KESME)	141
10.1.1	Maskelenebilir Kesmeler	142
10.1.2	Maskelenemez Kesmeler	144
10.1.3	Kesme Vektörleri	145
10.2	RESTART (YENİDEN BAŞLATMA)	148
ÜNİTE 11	MİKROBİLGİSAYARLARDA HAFIZA ORGANİZASYONU	149
11.1	MİKROİŞLEMCİNİN HAFIZA ALANI	149
11.2	HAFIZA PAYLAŞIMI	151
11.2.1	Tek RAM ve ROM'dan Oluşan Sistemler	151
11.2.2	Birden Fazla RAM ve ROM'dan Oluşan Sistemler	154
ÜNİTE 12	MİKROİŞLEMCİLERDE TEMEL GİRİŞ/ÇIKIŞ KAVRAMLARI	158
12.1	GİRİŞ/ÇIKIŞ ARABİRİMLERİ	158
12.2	MİKROİŞLEMCİLERDE GİRİŞ/ÇIKIŞ ARABİRİMLERİNİN ADRESLENMESİ	159
12.3	PARALEL GİRİŞ/ÇIKIŞ ARABİRİMİ	160
12.3.1	MC6821 Paralel Giriş/Çıkış Entegresi	161
12.4	SERİ GİRİŞ/ÇIKIŞ ARABİRİMLERİ	163
12.4.1	MC6850 Seri Giriş/Çıkış Entegresi	163
12.5	GİRİŞ/ÇIKIŞ ARABİRİMLERİ İLE İLETİŞİM YÖNTEMLERİ	167
12.5.1	Giriş/Çıkış Arabirimlerinin Adreslenmesi	167
12.5.2	MC6821 Paralel Giriş/Çıkış Entegresinin Programlanması	169
12.5.3	MC6850 Seri Giriş/Çıkış Entegresinin Programlanması	171
12.6	RS232 STANDARDI	173
ÜNİTE 13	ASSEMBLY PROGRAM YAZIMI	176
13.1	PROGRAMLAMA TEMEL KAVRAMLARI	176
13.1.1	Assembly Programı	177
13.1.2	Komutların Hafıza Gereksinimleri ve İşlem Süreleri	178

13.1.3 Assembler Yönlendiricileri (Pseudo; Sahte)	179
13.2 ASSEMBLY PROGRAM YAZMANIN TEMELLERİ	180
13.2.1 Problem Analizi	181
13.2.2 Akış Diyagramı	181
13.2.3 Assembly Kodlama	183
13.2.4 Assembly Komutlarının El ile Makine Diline Çevrilmesi	185
13.2.5 Assembly Komutlarının Assembler Programı ile Makine Diline Çevrilmesi	186
13.2.6 Hataların Düzeltilmesi (Debugging)	189
ÜNİTE 14 ASSEMBLY PROGRAM ÖRNEKLERİ	193
14.1 BASİT PROGRAM ÖRNEKLERİ	193
14.1.1 İki Sayının Toplanarak Sonucun Başka Bir Hafızaya Yükleme	193
14.1.2 Bir Hafıza Hücresinin veya Kaydedicinin Silinmesi	193
14.1.3 Hafızadaki Verilerin Yer Değiştirilmesi	193
14.1.4 Kaydedicideki Bir Verinin İlk Dört Bitinin Maskelenmesi	194
14.1.5 Hafızadaki İki Sayının Karşılaştırılması	194
14.1.6 Eğer Z Biti "0" ise Dallen	195
14.1.7 Alt Programa Dallenma ve Alt Programdan Geri Dönme (JSR ve RTS Komutları)	195
14.1.8 Herhangi Bir Adrese Dallenma (JMP Komutu)	196
14.1.9 Hafızadaki Veriyi Sola Kaydırma	196
14.2 ARİTMETİK VERİLİ PROGRAM YAZIMI	196
14.2.1 İki Sayıdan Büyüğün Bulunması	197
14.2.2 8 Bitlik İki Sayının Çarpımı	198
14.2.3 8 Bitlik Mikro işlemcilerde 16 Bitlik Sayıların Toplanması	200
14.2.4 Dört Haneli İki BCD Sayının Toplamı	201
14.2.5 Bir Sayı Dizisinde En Büyük Sayının Bulunması	202
14.3 KARAKTER VERİLİ PROGRAM YAZIMI	203
14.3.1 Belirli Bir Hafıza Bölgesinde Bir Byte Veri Arama Programı	203
14.3.2 Belirli Bir Hafıza Bölgesini Bir Sabit ile Doldurma Programı	204
14.3.3 Dizi İçerisindeki Belirli Bir Veriyi Başka Bir Veri ile Değiştirme	205
14.3.4 Belirli Bir Hafıza Aralığını Başka Bir Hafıza Bölgesine Taşıma	206
14.4 KOD DÖNÜŞTÜRÜCÜ PROGRAM YAZIMI	207
14.4.1 BCD Kodlu Sayının İkili Karşılığına Dönüşümü	207
14.4.2 BCD Kodlu Sayının 7 Segment Display İçin İkili Koda Dönüştürülmesi	209
14.5 ZAMAN GECİKTİRME PROGRAM YAZIMI	211
14.5.1 Zaman Geciktirme Alt Programlarına Duyulan İhtiyaç	211
14.5.2 30 Saniyelik Uzun Zaman Geciktirme Alt Programı	216
14.6 GİRİŞ/ÇIKIŞ İLİŞKİLİ PROGRAM YAZIMI	218
14.6.1 PIA'nın Temel Giriş/Çıkış İşlemleri için Programlanması	218
14.6.2 Anahtar ile Led Yakıp Söndürme Programı	219
ÜNİTE 15 ANALOG/DİJİTAL VE DİJİTAL/ANALOG SİSTEMLER	222
15.1 ANALOG VE DİJİTAL NİCELİKLER	222
15.2 DİJİTAL/ANALOG ÇEVİRİCİLER	224
15.2.1 Paralel Girişli D/A Çevirici	225
15.2.2 Merdiven tipi (R/2R) D/A Çevirici	226
15.2.3 Entegre D/A Çeviriciler	228
15.3 ANALOG/DİJİTAL ÇEVİRİCİLER	229
15.3.1 Kodlayıcı (Encoder) ile A/D Çevirici	229
15.3.2 Dirençli Merdiven Dalgalı A/D Çevirici	230
15.3.3 Tek Meyil Tamamlaması ile Yapılan A/D Çevirici	231
15.3.4 Çift Meyil Tamamlaması ile Yapılan A/D Çevirici	232
ÜNİTE 16 KARAKTER GENARATÖRLERİ	237
16.1 KARAKTER GENARATÖRLERİ	237
16.2 NOKTA MATRİSİ	237
16.3 CRT GÖSTERGELİ KARAKTER GENARATÖRLERİ	238

16.4 KARAKTER GENERATÖRLERİNİN ÇALIŞMA PRENSİPLERİ	239
16.4.1 CRT Tipindeki Görüntü Arabirimlerinde Karakter Görüntülenmesi	239
16.4.2 7 Segment ve Nokta Matris Displaylerde Karakter Görüntülenmesi	240
ÜNİTE 17 MİKRODENETLEYİCİLERLE KONTROL	245
17.1 MİKRODENETLEYİCİLERİN YAPISI	245
17.1.1 İç Yapısının Blok Diyagram Üzerinde İncelenmesi	245
17.1.2 Hafıza Bağlantıları (RAM ve EPROM)	250
17.1.3 Aritmetik Birim	250
17.2 MİKRODENETLEYİCİ KOMUTLARI	251
17.2.1 Aritmetik İşlem Komutları	251
17.2.2 Lojik işlem komutları	251
17.2.3 Veri Transfer Komutları	252
17.2.3.1 Yükleme komutları	252
17.2.3.2 Giriş/çıkış komutları	252
17.2.3.3 Basit bir giriş/çıkış uygulaması	252
17.2.4 İkili işlem Komutları	253
17.2.5 Program Dallenma Komutları	254
17.3 KESME KONTROLLÜ İŞLEMLER	254
17.4 MİKRODENETLEYİCİLERİN SİMÜLATÖR PROGRAMI	256
17.5 EPROM YAZMA YÖNTEMİ İLE PROGRAM AKTARIMI	256
17.6 SERİ PORT'TAN PROGRAMIN AKTARILMASI	258
ÜNİTE 18 ROBOTLAR	261
18.1 ROBOTUN TANIMI	261
18.2 ROBOTUN ENDÜSTRİDEKİ YERİ VE İNSANLA KARŞILAŞTIRILMASI	262
18.3 ROBOT SENSÖRLERİ	263
18.3.1 Duyular	263
18.3.2 Ölçüm ve Ölçerler	263
18.3.3 Sensörler/Ölçerler/Dedektörler	264
18.4 ROBOTUN PROGRAMLANMASI	266
18.5 ROBOTLARIN HAREKET BİÇİMLERİ	267
18.5.1 Eksenler ve Hareket Biçimleri	267
18.5.2 Kartezyen Koordinat	268
18.5.3 Silindirik Koordinat	268
18.5.4 Diğer Koordinatlar	269
18.5.4.1 Küresel koordinat konfigürasyonu	269
18.5.4.2 Mafsallı kol konfigürasyonu	269
18.6 ROBOT TİPLERİ	270
18.6.1 Basit Robotlar	271
18.6.2 Orta Teknoloji Robotları	271
18.6.3 Yüksek Teknoloji Robotları	271
18.7 ROBOT KONTROL VE ÇALIŞMA SİSTEMLERİ	271
ÜNİTE 19 PROGRAMLANABİLEN MANTIKSAL DENETLEYİCİLER (PLC)	274
19.1 PLC SİSTEMİ	274
19.1.1 PLC Nedir ?	274
19.1.2 PLC'nin Yapısı	274
19.2 MERDİVEN (LADDER) DİYAGRAMI	276
19.3 PROGRAMLAMA DİYAGRAM DİLİ	277
19.3.1 PLC sembolik (mnemonic) komutları	277
19.3.2 Ladder diyagramının PLC komutlarına dönüştürülmesi yöntemi	279
19.4 GİRDİ / ÇIKTI CİHAZLARI	281
19.5 PLC İLE ÜÇ FAZLI ASENKRON MOTORUN DEVİR YÖNÜNÜ DEĞİŞTİRME	282
EK VERİ SAYFALARI	284
TEKNİK TERİMLER SÖZLÜĞÜ	294
DİZİN	297
KAYNAKÇA	300

ÜNİTE 1 SAYI SİSTEMLERİ

1.1 İKİLİK (BINARY) SAYI SİSTEMİ

Niçin ikilik sayı sistemi? Bilgisayarların alt yapısını oluşturan dijital elektronik, mantıksal ifadelerin elektronik devrelere uyarlanması sonucu ortaya çıkmıştır. İnsanlar günlük hayatta onluk sayı sistemini kullanmasına karşılık, mantıksal önermeler için “0” ve “1” sayıları kullanılır. Örneğin, “kapı açık” “1” ile ifade ediliyorsa “kapı kapalı” durumunun “0” ile ifade edilmesi gerekir. Mantıksal ifadelerde “kapı yarım açık”, “kapı yarım kapalı” veya “kapı az açık” gibi ifadeler yer almaz. Dijital elektronik, “elektrik var” veya “elektrik yok” ilkesi ile çalışır. Dolayısıyla “elektrik var” ifadesi için “1”, “elektrik yok” ifadesi için “0” rakamları kullanılmıştır. Onluk sayı sisteminde “0,1,2,3,4,5,6,7,8,9” on adet rakam bulunmaktadır. İkilik sayı sisteminde iki adet rakam “0,1” kullanılmaktadır. Tüm ağırlıklı sayı sistemlerinde kullanılan formülü ikilik sayı sisteminde de aynen kullanabiliriz.

$$Q_n R^n + \dots + Q_3 R^3 + Q_2 R^2 + Q_1 R^1 + Q_0 R^0 \text{ formülü ile } (1011011)_2 \text{ sayısını;} \\ 1.2^6 + 0.2^5 + 1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 \text{ şeklinde yazılır.}$$

Burada Q rakamı “1” veya “0”, R ise tabanı göstermektedir. İkilik sayı sisteminin tabanı “2” ile gösterilir. Formülde görüldüğü gibi sağdan sola gidildikçe her terimin, ikinin bir üst derecedeki kuvveti olduğu görülür.

Özellikle değişik tabanda sayılarla çalışılıyorsa bir karışıklığa meydan vermemek için işlem yapılan sayının tabanını belirtmek gerekmektedir. Örneğin “100” sayısı onluk, ikilik ve onaltılık tabanlarda farklı farklı sayısal değerleri temsil etmektedir. “100” sayısının, onluk tabanda “yüz”, ikilik tabanda “dört”, onaltılık tabanda “iki yüz elli altı” olduğu görülür. Böyle bir karışıklığı gidermek için sayının tabanı, sağ alta yazılır. Örneğin (yyyy)_x şeklindeki bir sayıda, “yyyy” sayının rakamlarını, “x” sayının tabanını gösterir. Eğer x=2 ise sayı ikilik tabanda, x=10 ise sayı onluk tabanda, x=16 ise sayı onaltılık tabanda olduğu anlaşılmaktadır.

1.2 ONALTILIK (HEXADECIMAL) SAYI SİSTEMİ

Bilgisayar sistemlerinde, hesaplamalarda kolaylık sağladığı için “hexadecimal” sayı sistemi tercih edilmektedir. Bu sayı sisteminde kullanılan rakamlar 0-15 arasındadır. Kullanılan rakamlar 0 ile 9'a kadar onluk sayı sistemi ile aynı, 10,11,12,13,14,15 sayıları ise birer harf ile gösterilmiştir. Onaltılık sayı sisteminde kullanılan rakamlar şöyledir:

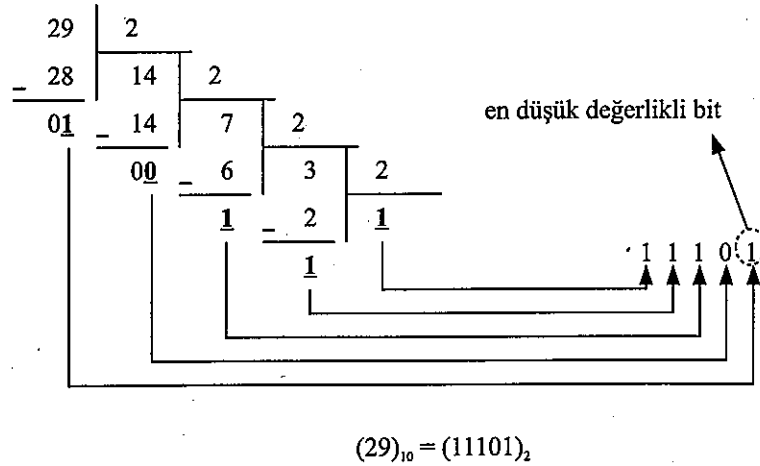
Onluk tabandaki sayılar	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Onaltılık tabandaki sayılar	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

1.3 SAYI SİSTEMLERİNİN BİRBİRİNE DÖNÜŞTÜRÜLMESİ

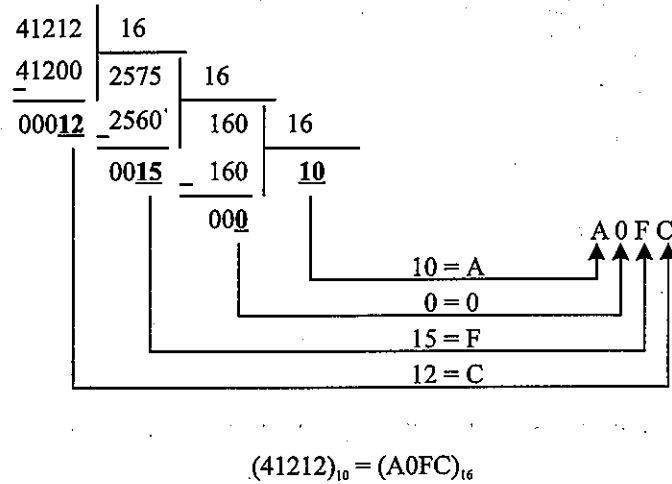
1.3.1 Onluk Sayıların İkilik ve Onaltılık Sayılara Çevrilmesi

Onluk tabandaki bir sayıyı ikili veya onaltılık tabanlı bir sayıya çevirmek için sayıyı virgülsüz bölününceye kadar tabana böleriz. İlk bölme işleminden elde edilen kalanı en düşük değerlikli bit olacak şekilde, diğer kalanları da sondan başa doğru sıralayarak sonucu buluruz.

Örnek 1 : $(29)_{10}$ onluk tabandaki sayıyı ikilik tabana çevirelim.



Örnek 2 : $(41212)_{10}$ onluk tabandaki sayıyı onaltılık tabana çevirelim.



1.3.2 İkilik ve Onaltılık Sayıların Onluk Sayılara Çevrilmesi

İkilik veya onaltılık tabandaki bir sayının onluk tabandaki bir sayıya dönüştürülmesi için aşağıdaki formül kullanılır. Formüldeki “R” taban sayısı, “Q” ilgili basamaktaki sayının rakamı, “n” ise basamağın derecesini gösterir.

$$Q_n R^n + \dots + Q_3 R^3 + Q_2 R^2 + Q_1 R^1 + Q_0 R^0$$

Örnek 1 : $(10011110)_2$ ikilik tabandaki sayıyı onluk tabana çevirelim.

$$\begin{aligned} (10011110)_2 &= 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ &= 1 \cdot 128 + 0 \cdot 64 + 0 \cdot 32 + 1 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 \\ &= 128 + 0 + 0 + 16 + 8 + 4 + 2 + 0 \\ &= (158)_{10} \end{aligned}$$

Örnek 2 : $(A0FC)_{16}$ onaltılık tabandaki bu sayıyı onluk tabana çevirelim.

$$\begin{aligned} (A0FC)_{16} &= A \cdot 16^3 + 0 \cdot 16^2 + F \cdot 16^1 + C \cdot 16^0 \\ &= 10 \cdot 16^3 + 0 \cdot 16^2 + 15 \cdot 16^1 + 12 \cdot 16^0 \\ &= 10 \cdot 4096 + 0 \cdot 256 + 15 \cdot 16 + 12 \cdot 1 \\ &= 40960 + 0 + 240 + 12 \\ &= (41212)_{10} \end{aligned}$$

1.3.3 İkilik Sayıların Onaltılık Sayılara Çevrilmesi

İkilik tabandaki bir sayıyı onaltılık tabandaki bir sayıya çevirmek için önce ikilik tabandaki sayıyı sağdan başlayarak dörderli gruplara ayırırız. Örneğin sayımız $(11110110001110)_2$ şeklinde bir sayı olsun. Sayı önce en sağdan başlanarak dört rakamlı gruplara ayrılır. Sonra en soldaki grubun rakamlarına, sıfır ilâve ederek dörde tamamlarız.

$$(11 \ 1101 \ 1000 \ 1110)_2 \rightarrow (0011 \ 1101 \ 1000 \ 1110)_2$$

Şimdi bu sayıda her dörderli haneyi ayrı ayrı ele alarak onaltılık sayı sistemine tablo 1.1.'den faydalanarak dönüştürebiliriz.

$(0011)_2$	$(1101)_2$	$(1000)_2$	$(1110)_2$
$(3)_{16}$	$(D)_{16}$	$(8)_{16}$	$(E)_{16}$

İkilik tabandaki sayımızın karşılığı $(0011 \ 1101 \ 1000 \ 1110)_2 = (3D8E)_{16}$ olmaktadır.

1.3.4 Onaltılık Sayıların İkilik Sayılara Çevrilmesi

Onaltılık sayı sisteminde gösterilen bir sayıyı, ikilik tabanda gösterebilmek için sayının her rakamı ayrı ayrı ikilik karşılığında yazılır.

Örnek : Sayımız $(F03A)_{16}$ şeklinde bir sayı olsun.

F	0	3	A
$(1111)_2$	$(0000)_2$	$(0011)_2$	$(1010)_2$

Onaltılık tabandaki sayımızın karşılığı $(F03A)_{16} = (1111000000111010)_2$ olmaktadır.

Gerçek hayatta onluk sayı sisteminin kullanılmasına karşılık dijital elektronikte ve bilgisayar sistemlerinde çoğunlukla ikilik ve onaltılık sayı sistemleri kullanılmaktadır. Bu sayılar arasındaki dönüşümün kuralları vardır. Yukarıda bu dönüşüm kurallarından bahsedilmiştir. Tablo 1.1'de onluk ve onaltılık sayıların ikilik karşılıkları verilmiştir.

Tablo 1.1 : Onluk ve onaltılık tabandaki sayıların ikilik tabandaki karşılıkları

Onluk sayı sistemi	Onaltılık sayı sistemi	İkilik sayı sistemi
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

1.4 NEGATİF VE KESİRLİ SAYILARIN İKİLİK SAYI SİSTEMLERİNDE İFADE EDİLMESİ

1.4.1 Negatif Sayıların İkilik Tabandaki Gösterimi

Onluk sayı sisteminde işaretli sayılar, sayının önüne “-” ve “+” işaretleri konarak tanımlanır. Onluk sayı sistemindeki pozitif sayıları göstermede çoğu zaman “+” işareti kullanmaya gerek duyulmaz. İkilik sayı sisteminde, işaretli sayılar bu şekilde gösterilmez. Fakat ikilik sayı sisteminde, işaretli sayılarla aritmetiksel işlemler yapmak için de bir yöntem gereklidir. İkilik sayı sisteminde işaretli sayılar, “işaret biti” kullanılarak ifade edilir. Bunun için işaretli bir ikilik sayının en yüksek değerlikli biti “işaret biti” olarak ayrılmıştır. Bu bite bakarak işaretli bir sayının pozitif veya negatif olduğunu anlarız. İşaretli bir sayının en değerlikli bit “0” ise sayı “pozitif”, “1” ise sayı “negatif”tir.



Tümleyen sayılar:

Bilgisayarlarda tümleyenler, çıkarma işlemlerini sadeleştirmek için kullanılmaktadır. Tümleme yöntemi ile çıkarma işlemleri için toplama devreleri kullanılabilir. Bu da ek çıkarıcı devre gereksinimini ortadan kaldırmaktadır. Sayı hangi tabanda olursa olsun iki tip tümleyeni vardır. Sayı tabanı r ise tümleyenleri (r) ve $(r-1)$ olur. Örneğin sayı tabanı 10 ise sayının “10” ve “9” tümleyenleri, sayı tabanı 2 ise sayının “2” ve “1” tümleyenleri olarak adlandırılır. İkilik tabandaki bir sayının 1'e tümleyenini bulmak için sayıdaki tüm “1”ler “0”, “0”lar “1” yapılır. Örneğin $(10011101)_2$ sayısının 1'e tümleyeni $(01100010)_2$ 'dir.

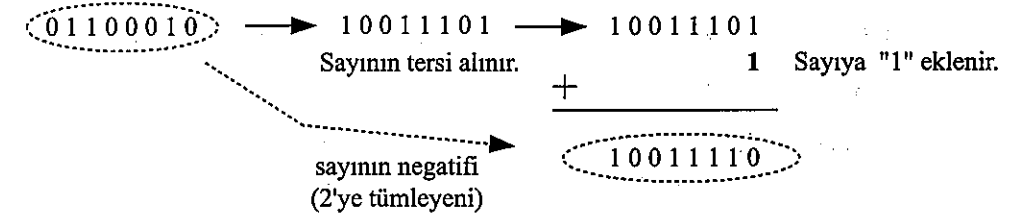
İkilik tabandaki bir sayının 2'ye tümleyenini bulmak için ise 1'e tümleyende olduğu gibi sayıdaki tüm “0”lar “1”, “1”ler “0” yapılarak bulunan sayıya “1” eklenir. Örneğin $(10011101)_2$ sayısının 2'ye tümleyeni $(01100011)_2$ 'dir.

$$\begin{array}{r} (11000101)_2 \\ + (00111011)_2 \\ \hline (00000000)_2 \end{array} \quad \begin{array}{r} (-59)_{10} \\ (+59)_{10} \\ \hline 0 \end{array}$$

İşaretli sayıların önündeki işaret biti o sayının pozitif veya negatif olduğunu göstermesine rağmen, pozitif ikilik bir sayı “0” işaret bitiyle gerçek değerinde gösterilir, negatif ikilik bir sayı ise “1” işaret bitiyle 2'ye tümleyenini şeklinde gösterilmektedir. Yukarıdaki örnekte, onluk tabandaki 59 sayısının pozitifinin ve negatifinin ikilik karşılıkları görülmektedir. “+59” sayısının ikilik karşılığı bu sayının normal olarak ikilik tabana çevrilmesiyle elde edilmiştir. “59” sayısının ikilik karşılığı ise ancak “59” sayısının normal ikilik karşılığının 2'ye tümleyenini alınarak bulunabilir.

Bir byte'lık işaretli sayıların 8. biti işaret biti olarak kullanıldığından, kalan 7 bit ile en fazla -128 ile +127 arasındaki sayılar gösterilebilmektedir. Burada dikkat edilecek nokta eğer 1 byte'lık işaretli sayılarla çalışılacaksa sayıların -128 ve +127 aralığında kalmasıdır. Bu aralığın dışına çıkıldığında sayının işaretli formu bozulur.

2'ye tümleme ile ikilik tabandaki sayıların negatif gösterilmesi: Pozitif ikilik bir sayının negatifini bulurken kullanılan yöntem, ikilik tabandaki n bitlik sayının tüm bitlerinin tersi alınır ve bulunan sonuca “1” eklenir. Örneğin $(+98)_{10}$ 'in ikilik tabandaki karşılığı olan $(01100010)_2$ sayısının negatifini bulmak için (dolayısıyla 2'ye tümleyenini bulmak için) bütün bitlerin tersi alınır. Bulunan bu sayı $(10011110)_2$ olacaktır. Buna “1” ilâve edildiğinde sonuç $(10011111)_2$ olacaktır.



Yukarıdaki yapılan işleme göre +98'in negatifi bulunduğu göre bu sayının negatifi ile kendisi toplandığında sonucun sıfır olması gerekecektir. Yani $(01100010)_2 = (+98)_{10}$ ile $(10011110)_2 = (98)_{10}$ topladığında sonuç sıfır olmalıdır. Aşağıdaki işlemde de görülmektedir.

$$\begin{array}{r} 01100010 \\ + 10011110 \\ \hline 10000000 \end{array}$$

Taşan bit ihmal edilir. →

Şekil 1.1'de işaretli sayıların 8 bitlik RAM hafızadaki yerleşimi gösterilmektedir. Hafızadaki negatif ikilik sayılar, 2'ye tümleyenini şeklinde gösterildiğine dikkat ediniz. +12 ile -12 sayılarının işaretli ikilik tabandaki gösterimi aynı değildir.

onluk sayılar	işaretili ikilik sayılar	işaret bitleri
-12	11110100	sayının 2'ye tümleyeni
+56	00111000	normal ikilik düzende
+27	00011011	normal ikilik düzende
-49	11001111	sayının 2'ye tümleyeni
-102	10011010	sayının 2'ye tümleyeni
+12	00001100	normal ikilik düzende
-117	10001011	sayının 2'ye tümleyeni

Şekil 1.1 : İşaret bitiyle gösterilen pozitif ve negatif sayılar

1.4.2 Ondalık Sayıların İkilik Tabandaki Gösterimi

Onluk tabandaki ondalıklı bir sayıyı ikilik tabana çevirmek üzere; tam kısım normal olarak ikilik tabana çevrilir. Ondalık sayı daima iki ile çarpılır ve çıkan sonucun "tam kısmı" yazılır. Çıkan sonucun tam kısmı ya "1" ya da "0" olacaktır. Tam kısım yazıldıktan sonra sayının ondalıklı kısmı tekrar iki ile çarpılır. Bu işlem sonuç sıfır oluncaya kadar devam eder. Şimdi onluk tabandaki $(34,40625)_{10}$ ondalıklı sayıyı ikilik tabanda gösterelim:

Sayı tam ve ondalık olarak iki kısımdan oluşmaktadır. Tam kısım olduğu gibi ikilik taban çevrilir. Yani sayımızın tam kısmı olan 34 sayısı ikilik tabanda $(100010)_2$ şeklinde çevrilir. Ondalık kısmı ise aşağıdaki gibi "0" oluncaya kadar 2 ile çarpılır.

$0,40625 \times 2$	$= [0],8125$
$0,81250 \times 2$	$= [1],6250$
$0,62500 \times 2$	$= [1],2500$
$0,25000 \times 2$	$= [0],5000$
$0,50000 \times 2$	$= [1],0000$

Yapılan işlemlerde elde edilen sonuçların tam kısımları yukarıdan aşağıya doğru sıra ile yazılır $(0,40625)_{10} = (0,01101)_2$. Daha sonra tam ve ondalıklı kısım birleştirilir. $(100010,01101)_2$ sayısı $(34,40625)_{10}$ sayısının ikilik tabanındaki karşılığı olacaktır.

1.5 BİLGİSAYAR SİSTEMLERİNDE ONDALIKLI SAYILARIN TAM SAYI OLARAK İFADESİ

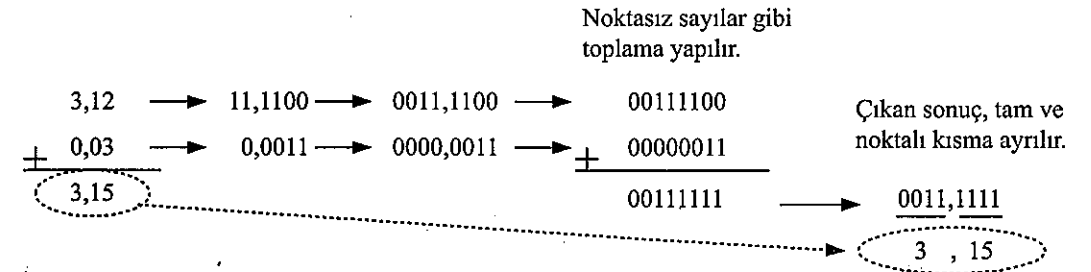
Kâğıt üzerinde ondalıklı sayılar üzerinde işlemler yaparken sayıların tam ve ondalık kısımlarını gösteren "," işaretini gereken yere koyabiliriz. Bilgisayar sistemlerinde sayıları tutan hafıza hücrelerinde ondalıklı sayıların noktalama işaretlerinin konulabileceği herhangi bir alan mevcut değildir. Bilgisayar sistemlerinde ondalıklı sayılar üzerinde de işlemler yapabilmek için iki tür yöntem kullanılmaktadır:

- 1- Sabit noktalı sayılar
- 2- Kayan noktalı sayılar

Sabit noktalı sayılarla işlemlerin nasıl yapıldığını anlamak için aşağıdaki örnekleri inceleyelim:

$$\begin{array}{r} 54596232 \\ + 43269857 \\ \hline 97866089 \end{array} \quad \begin{array}{r} 54,596232 \\ + 43,269857 \\ \hline 97,866089 \end{array}$$

Tamsayılarla yapılan toplama işlemi ile ondalıklı sayılarla yapılan toplama işlemi, noktalama işaretini görmezsek aynıdır. Bu ilke bilgisayarlarda da noktalı sayılarla aritmetiksel işlemler yapmada kullanılmıştır fakat bir farkla bilgisayarın hafızasında tutması gereken tek ayrıntı, nokta işaretinin yeri olacaktır. Bilgisayara yapılan tüm girişlerde ve çıkışlarda bu nokta göz önüne alınarak işlemler yapılmaktadır. Yapılan tüm aritmetiksel işlemlerde, ondalıklı sayılar (noktanın yeri başka bir hafıza hücresinde saklı tutulmak üzere) tam sayıymış gibi işlem görür.

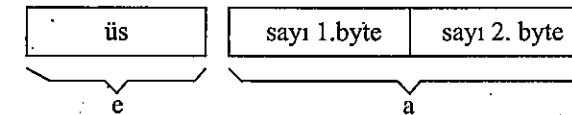


3,12 ve 0,03 gibi iki sayıyı toplayıp sonucu yazdırmak istediğimizde ilk önce sayıların tam ve ondalıklı kısımları ayrı ayrı ikilik tabana dönüştürülür. Bu aşamada sayılar $(00111100)_2$ ve $(00000011)_2$ olarak işleme sokulacaktır. Daha sonra normal toplama işlemi yapılır ve çıkan sonucu ikiye bölerek tam kısmı $(0011)_2$ ve ondalık $(1111)_2$ kısmı noktalama işareti de dikkate alınarak ekrana 0011,1111 olacak şekilde bastırılır. Çok büyük sayılarla işlemler yapabilmek için birkaç hafıza hücresi beraber kullanılarak sayı aralığı genişletilebilir.

Daha çok bilimsel hesaplamalarda karşımıza çıkan çok yüksek sayı aralıkları ve bu sayılarla yapılan işlemler için "sabit noktalı sayılar" yöntemini kullanmak hafızada gereğinden çok yer kaplayacaktır. Görülüyor ki sayıların sabit noktalı biçimde gösterilmesinin birtakım yetersizlikleri vardır fakat donanım ve işlemci mimarisi açısından en basit yöntemdir. Bilgisayarlarda çok büyük veya çok küçük sayılarla işlemler yapabilmek için "kayan noktalı sayılar" yöntemi geliştirilmiştir. Kayan noktalı sayılarla işlemler yapan mikroişlemciler bu yönleri geliştirilmiş mikroişlemcilerdir. Günümüzdeki çoğu bilgisayarlarda kullanılan mikroişlemciler kayan-noktalı sayıları destekleyecek mimari yapıdadır.

125,625 sayısını $0,125625 \times 10^3$ şeklinde de gösterebiliriz. Kayan noktalı sayılar üzerinde işlemler yapan mikroişlemciler de benzer biçimde bir yaklaşımla çalışmaktadır. $1111101,1001110001$ sayısı sistem içerisinde $0,11111011001110001 \times 10^7$ olarak gösterilmektedir ve 10^7 deki 7 de ikilik tabanda saklanmaktadır. Böyle olunca sayımız $(11111011001110001)_2$ ve $(0111)_2$ şeklinde iki kısım olarak ayrı ayrı saklanmaktadır. Kayan noktalı sayılar veya üstel sayılar hafızada art arda şekil 1.2'deki gibi yer alırlar.

Hafızada bu şekilde bulunan sayı $a \cdot 2^e$ şeklinde formüle edilir.



Şekil 1.2 : Üstel sayıların hafızadaki yerleşimi

DEĞERLENDİRME ÇALIŞMALARI

- 1) Bilgisayar sistemlerinde neden en çok ikilik ve onaltılık sayı sistemleri kullanılmaktadır?
- 2) Kendinizin belirlediği 10 tane onluk tabandaki sayıyı ikilik ve onaltılık sayı tabanlarına çeviriniz.
- 3) $(1011010)_2$, $(11101111)_2$, $(01011011)_2$, $(11010)_2$, sayılarını onluk sayı tabanına çeviriniz.
- 4) $(45)_{16}$, $(EF)_{16}$, $(2D)_{16}$, $(C2)_{16}$, sayılarını onluk sayı tabanına çeviriniz.
- 5) $(11101)_2$, $(10011110)_2$, $(01011101)_2$, $(10000001)_2$, sayılarını onaltılık sayı tabanlarına çeviriniz.
- 6) $(27)_{16}$, $(DF)_{16}$, $(3F)_{16}$, $(80)_{16}$, sayılarını ikilik sayı tabanlarına çeviriniz.
- 7) $(01101101)_2$, $(00111011)_2$, $(00111010)_2$, $(01111110)_2$, pozitif sayıların negatiflerini yazınız.
- 8) Bilgisayarlarda ondalıklı sayıların ikilik tabanda gösterilmesi nasıl yapılmaktadır?
- 9) 12.0625, 8.128, 3.425, 32.826 ondalıklı sayılarını ikilik sayı tabanında gösteriniz.

ÜNİTE 2 İNFORMASYONLARIN BİTLERLE İFADE EDİLMESİ

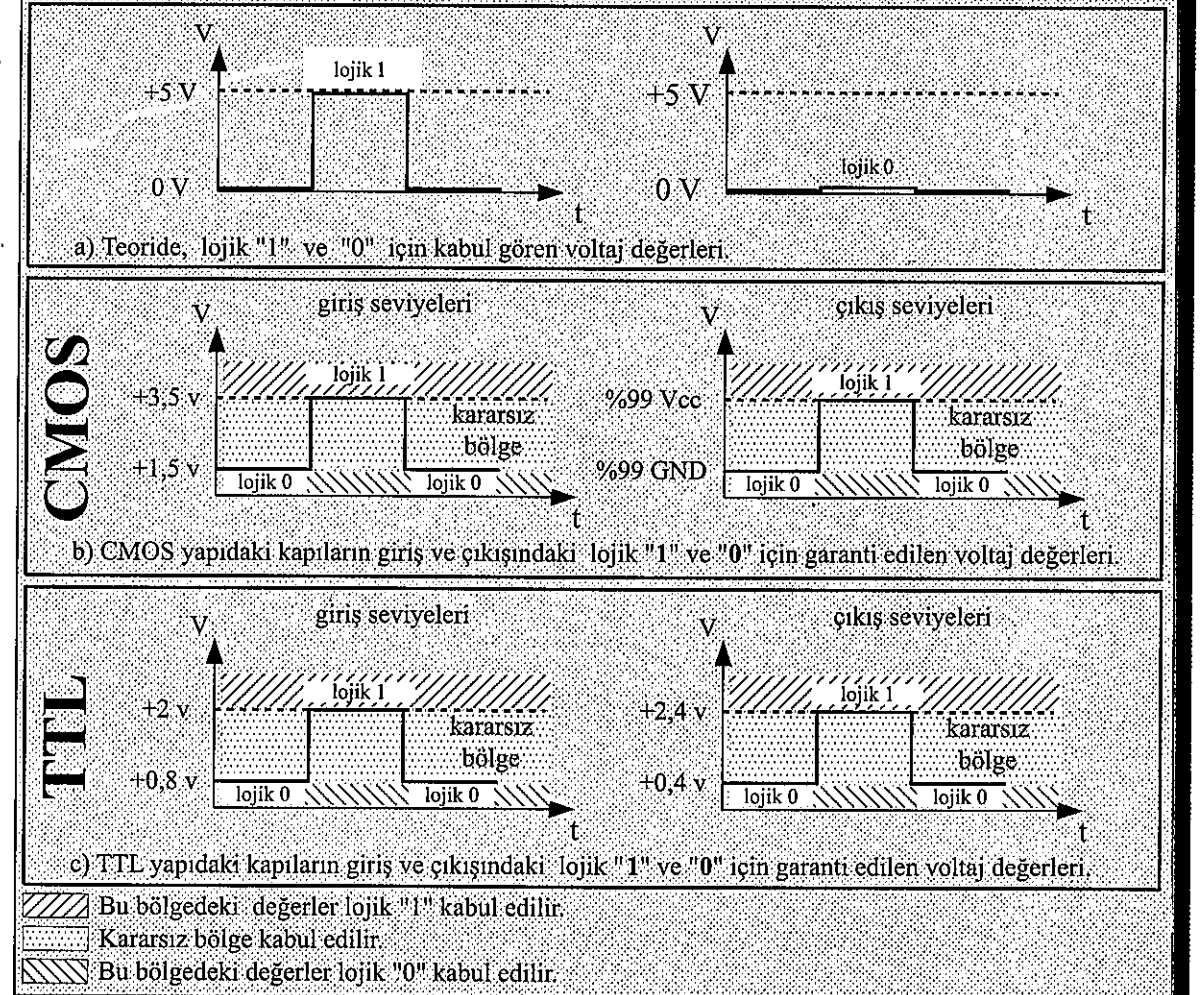
2.1 BİT, NIBBLE, BYTE VE WORD KAVRAMLARI

Dijital elektronik, matematiksel ve mantıksal ifadelerin elektronik ortamlarda değerlendirilmesidir. Dijital elektronik devrelerinde kullanılan sayısal kapılar, içerisinde bulunan transistorların iletimde veya kesimde olması ile çalışır. Sayısal kapıların çıkışında +5 voltluk bir değer "1" ile 0 voltluk bir değer "0" ile temsil edilmiştir. Dijital sistemlerde giriş/çıkışların "1" ve "0" ile temsil edilmesi bu sistemler üzerinde yapılan çalışmalarda ikilik tabandaki sayı sisteminin kullanılmasını gerektirir.



Lojik Elemanların Mantıksal Durumlara Karşılık Gelen Gerilim Seviyeleri:

Yukarıda ifade edilen +5 volt ve 0 volt değerleri tanım gereği kullanılmıştır. Gerçekte CMOS veya TTL yapıdaki sayısal kapıların çıkış ve girişlerindeki lojik "1" ve "0" değerlerine karşılık gelen gerilim seviyeleri aşağıda görülmektedir. (5 v. besleme gerilimi altında).



Bit; bilgisayarlarda üzerinde işlem yapılabilecek en küçük bilgi birimi “bit”tir. Bit, bize yalnızca bir bilginin (information) var veya yok olduğunu gösterebilir. Genellikle bir bitin “1” veya “0” durumlarına karşılık gelen bilgiyi biz tanımlarız. Örneğin “1” durumuna “pencere açık”, “0” durumu da “pencere kapalı” şeklinde olabilir.

Nibble; tek bit ile en fazla iki önermeyi ifade edebiliriz daha fazlası mümkün değildir. Eğer birden fazla bit kullanırsak bu bitlerle oluşturabileceğimiz bilgi ifadelerini artırabiliriz. Nibble 4 bitlik veri kümesidir. Tablo 2.1’de 4 bitlik veri ile tanımlanan ifadeler yer almıştır.

Tablo 2.1’de görüldüğü gibi 1 bitle yalnız iki önermeye yer verebilirken 2 bit kullandığımızda dört adet önermeye yer verebiliyoruz. Kullandığımız bit sayısı 4’e çıkardığımızda ise kullanabileceğimiz önerme sayısı on altıya çıkmıştır. Demek ki kullanılan bit sayısı daha da artırılırsa ifade edilebilecek önermeler de artacaktır. Çoğu zaman mikrobilgisayarlarda¹ kullanılan ikilik sayının sayısal değerinden çok bitlerin ifade ettiği anlamlar daha önemli olabilmektedir.

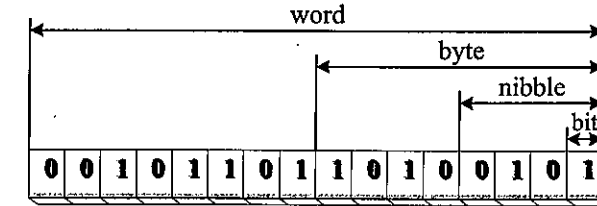
Tablo 2.1 : Bitlerle ifade edilebilecek önermeler

Kullanılan bit sayısı	Bit deseni	Bitlere yüklenen anlamlar (Sayısal veya mantıksal ifadeler verilebilir.)
1 Bit	0 0	Kapı kapalı
	1 1	Kapı açık
2 Bit	00 0	Kapı kapalı, pencere kapalı
	01 1	Kapı kapalı, pencere açık
	10 2	Kapı açık, pencere kapalı
	11 3	Kapı açık, pencere açık
4 Bit (nibble)	0000 0	Kapı kapalı, pencere kapalı, lâmba kapalı, klima kapalı
	0001 1	Kapı kapalı, pencere kapalı, lâmba kapalı, klima açık
	0010 2	Kapı kapalı, pencere kapalı, lâmba açık, klima kapalı
	0011 3	Kapı kapalı, pencere kapalı, lâmba açık, klima açık
	0100 4	Kapı kapalı, pencere açık, lâmba kapalı, klima kapalı
	0101 5	Kapı kapalı, pencere açık, lâmba kapalı, klima açık
	0110 6	Kapı kapalı, pencere açık, lâmba açık, klima kapalı
	0111 7	Kapı kapalı, pencere açık, lâmba açık, klima açık
	1000 8	Kapı açık, pencere kapalı, lâmba kapalı, klima kapalı
	1001 9	Kapı açık, pencere kapalı, lâmba kapalı, klima açık
	1010 10	Kapı açık, pencere kapalı, lâmba açık, klima kapalı
	1011 11	Kapı açık, pencere kapalı, lâmba açık, klima açık
	1100 12	Kapı açık, pencere açık, lâmba kapalı, klima kapalı
	1101 13	Kapı açık, pencere açık, lâmba kapalı, klima açık
	1110 14	Kapı açık, pencere açık, lâmba açık, klima kapalı
	1111 15	Kapı açık, pencere açık, lâmba açık, klima açık

Byte; bilgisayarlarda, bilgileri saklamak ve üzerinde işlem yapabilmek için birden fazla bitten oluşan bir yapı kullanır. Birden fazla bitten oluşan bu yapıya “byte” (okunuşu; bayt) denilir. Byte olarak belirlenen veri kümesindeki bit sayısı 8’dir. Tüm bilgisayarlarda, temel veri boyutu olan ve 8 bitten oluşan veri kümelerini yani byte’ları kullanabilirler.

¹ Mikrobilgisayar : İçerisinde mikroişlemci, hafıza, giriş/çıkış birimleri ve klavye/display ünitesi bulunan olan en küçük bilgisayar.

Word (Kelime); iki byte’ın birlikte kullanılması durumunda oluşan birleşik yapıya “word” denir. Mikroişlemcinin 1 byte’tan daha büyük veriler üzerinde işlem yapabilmesi için veri yolu ve kaydediciler gibi mimari altyapısında 16 biti destekleyecek şekilde tasarlanması gerekmektedir. Zilog Z80, Motorola 68021 byte’lık veriler üzerinde işlem yaparken Intel 8086 ve 80286, Motorola 68000 ve Zilog Z8000 mikroişlemcileri 2 byte’lık veriler üzerinden işlem yaparlar.



Şekil 2.1 : Örnek bir sayı üzerinde word, byte, nibble ve bit

2.2 BİLGİSAYARDA BİLGİ BİRİMİ

Bilgi¹, bilgisayar sistemlerinde saklanabilen, üzerinde her türlü aritmetiksel ve mantıksal işlemlerin yapılabildiği ve hafıza birimlerinde fiziksel olarak yer işgal eden bir değerdir. Nasıl ki okuduğumuz bir kitapta bulunan bilgiler kitabın sayfalarını doldurup belli bir sayfa sayısı ve kalınlık oluşturuyorsa bilgisayarlarda da bilgi, fiziksel bir yer kaplar. Bilgisayarlarda bilgi kapasiteleri belli bir birimle ifade edilmektedir. Nasıl ki uzunluk birimi cm, ağırlık birimi gr ve bunların ast/üst katları kullanılıyorsa bilgisayarlardaki bilgiler de byte ve bunun üst katları ile ifade edilmektedir.

Bilgisayarlarda ve dijital elektronikte bilgileri temsil eden en küçük bilgi birimi “bit” tir. Bir bit “1” veya “0” rakamlarından birisi ile ifade edilir. Bu iki değerle ancak sınırlı sayıda ki en fazla iki değişik bilgiyi ifade edebiliriz. Birden fazla bitlerin kullanılmasıyla daha fazla bilgiler kodlanabilir. Böylece “sekiz bitten” oluşan kodlama tekniği ile “byte” kavramı oluşmuştur. Yani sekiz adet “bit” bir araya getirilerek oluşturulan grup ile “byte” düzeni oluşturulmuştur. Bilgisayarlarda anlamlı (ifade edebileceği bilgi bakımından) en küçük bilgi birimine “byte” denir. Byte’ın üst katları Tablo 2.2’de gösterilmiştir.

BİRİMİ	AÇIKLAMA
1 Bit	“1” ve “0” rakamlarından birisi ile gösterilir.
1 Byte = 8 Bit	8 bit 1 byte eder. Örneğin (10011101), gibi.
1 Kilobyte = 1024 Byte	2^{10}
1 Megabyte = 1024 Kilobyte	2^{20}
1 Gigabyte = 1024 Megabyte	2^{30}
1 Terabyte = 1024 Gigabyte	2^{40}

¹ Bilgi (veri, data, information): Bilgisayar terminolojisinde, bilgi terimi işlenebilir veri anlamında kullanılır. Bilgi sayısal bir değer olabileceği gibi alfasayısal bir değer de olabilir. Bu bilgiler ses, video, grafik, metin vb. formlarda bulunabilirler.

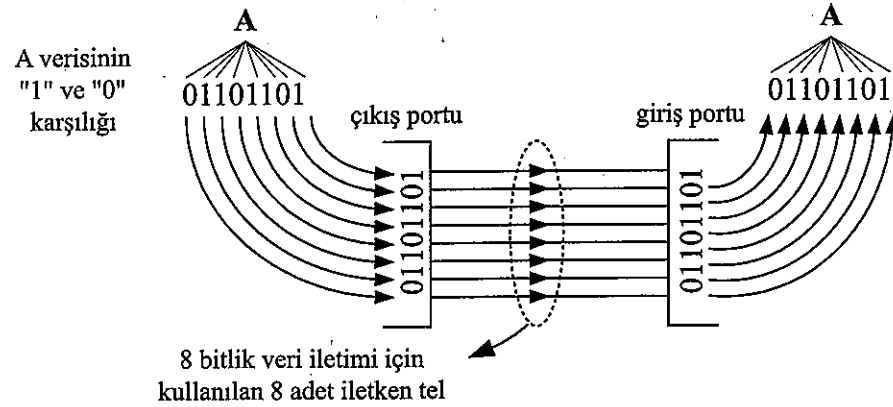


Niçin 1 Kilobyte 1000 Byte Değilde 1024 Byte Eder?

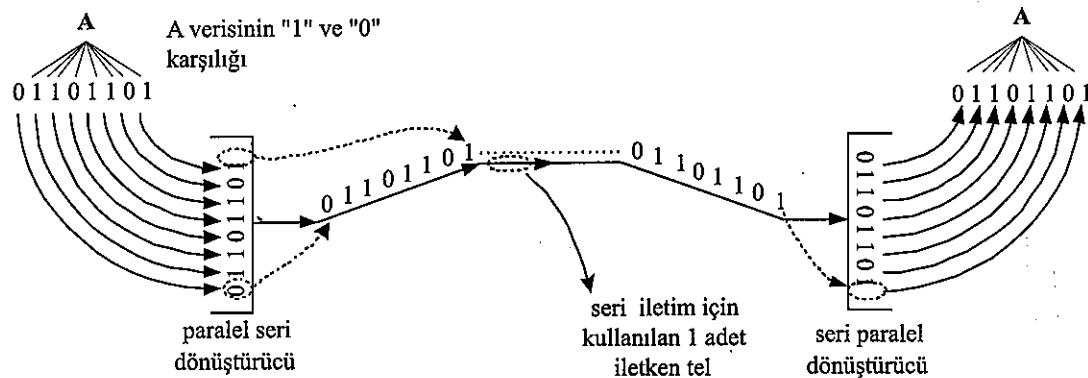
İkili sayı sisteminde 2^{10} "iki" nin 10. kuvveti 1024'e eşittir. Onluk sayı sisteminde ise 10^3 "10"un her kuvveti 10'un katlarına (1000, 100000 vb...) eşit olduğundan böyle bir farklılık vardır.

2.3 VERİ İLETİM HIZI

Bilgisayarda verilerin "1" ve "0" rakamları ile temsil edildiğini, "1" ve "0" rakamlarının da aslında elektriksel değerlere (+5 V, 0 V) karşılık geldiğini biliyoruz. Verilerin iletimi; veriyi tanımlayan elektriksel sinyallerin, iletken tellerle diğer ortamlara aktarılması demektir. Bilgisayar ortamında veriler, sürekli olarak bir yerden başka bir yere taşınarak, birtakım işlemlerden (matematiksel veya mantıksal) geçirilir. Verilerin taşınmasında iki yöntem kullanılır; "seri iletim" ve "paralel iletim". Bu iletim yöntemleri ile verilerin nasıl iletildiği şekil 2.2 ve şekil 2.3'de şematik olarak gösterilmiştir.



Şekil 2.2 : "A" verisinin paralel olarak iletilmesi



Şekil 2.3 "A" verisinin seri olarak iletilmesi

Bölüm 1'de bahsedildiği gibi "1" ve "0"ları kullanarak anlamlı bilgiler oluşturmak için 8 bitlik kümeler (1 byte) kullanılmış ve bu da bir standart (ASCII standardı) olarak kabul edilmiştir. ASCII standardında karakter, rakam ve semboller 8 bit ile kodlanmıştır.

Şekil 2.2'de bir "A" verisinin paralel şekilde nasıl iletildiği görülmektedir. Farklı bir tasarım düşünülmeyişi sürece, "paralel iletim tekniğinde" iletim hattının sayısı 8 iletken telden oluşmuştur. Sistemde kullanılan mikroişlemci 16 bitlik bir mikroişlemciyse bilgiler 8'er bitlik paketler hâlinde gönderilir.

Şekil 2.3'de ise "A" verisinin seri olarak nasıl iletildiği gösterilmiştir. Seri iletimde, veriler paralel yapıdan kurtarılıp tek tek bitlere ayrılır. Bu bitler sırası ile tek bir yol üzerinden alıcı tarafa gönderilir. Alıcı taraf gelen bitleri tekrar eski paralel yapısına kavuşturmak için birleştirir. Seri iletimde, verinin iletildiği hattın yanında ayrıca güvenli ve doğru veri iletilişi için "el sıkışma hatları" da (hand shaking) bulundurulur.

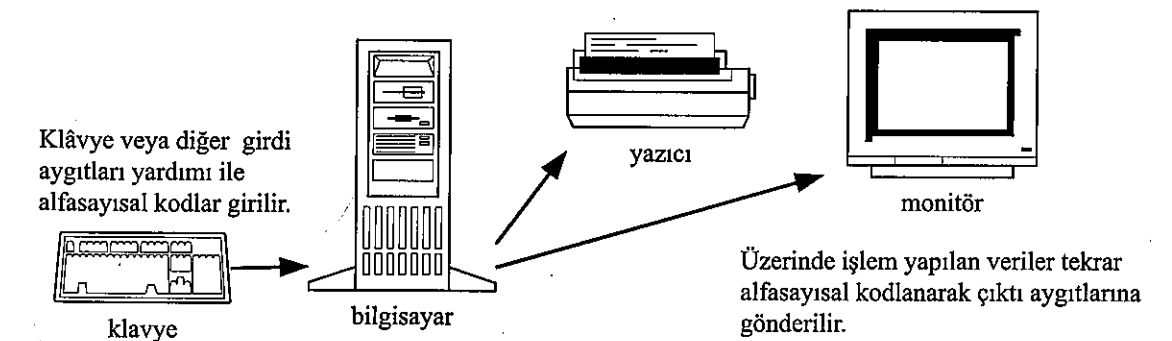
Veri iletiminde hangi tekniği kullanmamızın daha avantajlı olacağına sistemin durumuna göre biz karar veririz. Paralel iletim yöntemi, mesafenin az olduğu (yaklaşık 10 m) ve yüksek hızda iletimin gerektirdiği durumlarda kullanılması daha uygun olur. Seri iletim ise mesafenin uzun olduğu (10 m'den fazla 100 m'ye kadar. Daha uzun mesafeler için hat yükselteçleri kullanılmaktadır.) ve iletim hızının çok önemli olmadığı durumlarda kullanmak gereklidir. Ayrıca paralel iletim tekniklerinin daha basit ve kolay gerçekleştirildiğini fakat seri iletimin biraz karmaşık ve dikkat gerektirdiğini söylememiz gerekir. Seri iletim tekniğinde "seri" kelimesinden, paralelden daha hızlı olduğu anlamı çıkartmamak gerekir. "seri" kelimesi burada art arda anlamında kullanılmıştır.

2.4 ASCII KODU YARDIMLARIYLA ALFASAYISAL İŞARETLERİN İFADE EDİLMESİ

Bilgisayarlar yalnızca sayısal verilerle işlem yapan makineler değildir. Bilgisayarların hızından ve veri saklama yeteneğinden yeterince yararlanmak için sembollerle (alfasayısal; sayısal olmayan) de çalışması sağlanmıştır. Bilgisayar yapı itibarı ile sayısal veriler üzerinde işlem yapabilmektedir. Bunun için bilgisayarda harflerden ve sembollerden oluşan birtakım verileri kodlama yoluna gidilmiştir. Kodlama yolu ile her harf veya sembolü bilgisayarlarda kullanabilmek amacıyla ikili sistemde, 8 bitten oluşmuş karşılıklar atanmıştır. 1'ler ve 0'larda oluşan bu bit desenlerinin bir anlamı yoktur. Bu bit deseni, bizim ona verdiğimiz anlamla bir nitelik kazanır.

Bilgisayar sonuçta yine sayısal verilerle işlem yapmaya devam edecektir fakat bilgisayar, üzerinde işlem yaptığı verinin alfasayısal bir veri olduğunu tanımlayamaz. Bilgisayar için "veri"; üzerinde mantıksal veya matematiksel işlem yapılacak bir değerdir. Bilgisayar işlediği veya ürettiği verileri, kodlama ve kod çözme işlemlerinden sonra alfasayısal veya sayısal verilere çevirerek insanların günlük hayatta kullandığı harflere, rakamlara ve sembollere dönüştürür.

Alfasayısal kodlar sayısal verilere çevrilerek üzerinde aritmetiksel veya mantıksal işlemler yapılır.



Şekil 2.4 : Bilgisayarlarda alfasayısal karakterlerin kullanılması

ASCII (American Standards Code for Information Interchange) kodu uluslararası kabul görmüş bir kodlama sistemidir. Tablo 2.3'de ASCII standardına göre belirlenmiş 128 adet harf, sembol ve fonksiyon kodları bulunmaktadır. Tabloda kiremit rengi ile gösterilen hücrelerdeki ASCII kodları, bilgisayar sistemini direkt olarak etkileyen fonksiyon kodları olarak kullanılmaktadır.

Boş
Başlığın başlangıcı
Metnin başlangıcı
Metnin sonu
İletimin sonu
Sorgulama
Onay
Zil sesi
Arka boşluk
Yatay tablolama
Besleme hattı
Düşey tablolama
Besleme şekli
Şaryo dönüşü
Dışa kayma
İçe kayma
Veri yolu kontrol
Devre kontrolü 1
Devre kontrolü 2
Devre kontrolü 3
Devre kontrolü 4
Olumsuz onay
Senkronizasyon
İletim sonu
İptal
Ortamın sonu
Yer değiştirme
Kaçış
Dosya ayrıcı
Grup ayrıcı
Kayıt ayrıcı
Birim ayrıcı
Boşluk karakteri

ASCII kod	Onluk	Onaltılık	İkilik	ASCII kod	Onluk	Onaltılık	İkilik		
NUL	0	0	0	0000 0000	@	64	4	0	0100 0000
SOH	1	0	1	0000 0001	A	65	4	1	0100 0001
STX	2	0	2	0000 0010	B	66	4	2	0100 0010
ETX	3	0	3	0000 0011	C	67	4	3	0100 0011
EOT	4	0	4	0000 0100	D	68	4	4	0100 0100
ENQ	5	0	5	0000 0101	E	69	4	5	0100 0101
ACK	6	0	6	0000 0110	F	70	4	6	0100 0110
BEL	7	0	7	0000 0111	G	71	4	7	0100 0111
BS	8	0	8	0000 1000	H	72	4	8	0100 1000
HT	9	0	9	0000 1001	I	73	4	9	0100 1001
LF	10	0	A	0000 1010	J	74	4	A	0100 1010
VT	11	0	B	0000 1011	K	75	4	B	0100 1011
FF	12	0	C	0000 1100	L	76	4	C	0100 1100
CR	13	0	D	0000 1101	M	77	4	D	0100 1101
SO	14	0	E	0000 1110	N	78	4	E	0100 1110
SI	15	0	F	0000 1111	O	79	4	F	0100 1111
DLE	16	1	0	0001 0000	P	80	5	0	0101 0000
DC1	17	1	1	0001 0001	Q	81	5	1	0101 0001
DC2	18	1	2	0001 0010	R	82	5	2	0101 0010
DC3	19	1	3	0001 0011	S	83	5	3	0101 0011
DC4	20	1	4	0001 0100	T	84	5	4	0101 0100
NAK	21	1	5	0001 0101	U	85	5	5	0101 0101
SYN	22	1	6	0001 0110	V	86	5	6	0101 0110
ETB	23	1	7	0001 0111	W	87	5	7	0101 0111
CAN	24	1	8	0001 1000	X	88	5	8	0101 1000
EM	25	1	9	0001 1001	Y	89	5	9	0101 1001
SUB	26	1	A	0001 1010	Z	90	5	A	0101 1010
ESC	27	1	B	0001 1011	[	91	5	B	0101 1011
FS	28	1	C	0001 1100	\	92	5	C	0101 1100
GS	29	1	D	0001 1101	]	93	5	D	0101 1101
RS	30	1	E	0001 1110	^	94	5	E	0101 1110
US	31	1	F	0001 1111	-	95	5	F	0101 1111
SP	32	2	0	0010 0000	`	96	6	0	0110 0000
!	33	2	1	0010 0001	a	97	6	1	0110 0001
"	34	2	2	0010 0010	b	98	6	2	0110 0010
#	35	2	3	0010 0011	c	99	6	3	0110 0011
\$	36	2	4	0010 0100	d	100	6	4	0110 0100
%	37	2	5	0010 0101	e	101	6	5	0110 0101
&	38	2	6	0010 0110	f	102	6	6	0110 0110
'	39	2	7	0010 0111	g	103	6	7	0110 0111
(	40	2	8	0010 1000	h	104	6	8	0110 1000
)	41	2	9	0010 1001	i	105	6	9	0110 1001
*	42	2	A	0010 1010	j	106	6	A	0110 1010
+	43	2	B	0010 1011	k	107	6	B	0110 1011
,	44	2	C	0010 1100	l	108	6	C	0110 1100
-	45	2	D	0010 1101	m	109	6	D	0110 1101
.	46	2	E	0010 1110	n	110	6	E	0110 1110
/	47	2	F	0010 1111	o	111	6	F	0110 1111
0	48	3	0	0011 0000	p	112	7	0	0111 0000
1	49	3	1	0011 0001	q	113	7	1	0111 0001
2	50	3	2	0011 0010	r	114	7	2	0111 0010
3	51	3	3	0011 0011	s	115	7	3	0111 0011
4	52	3	4	0011 0100	t	116	7	4	0111 0100
5	53	3	5	0011 0101	u	117	7	5	0111 0101
6	54	3	6	0011 0110	v	118	7	6	0111 0110
7	55	3	7	0011 0111	w	119	7	7	0111 0111
8	56	3	8	0011 1000	x	120	7	8	0111 1000
9	57	3	9	0011 1001	y	121	7	9	0111 1001
:	58	3	A	0011 1010	z	122	7	A	0111 1010
;	59	3	B	0011 1011	{	123	7	B	0111 1011
<	60	3	C	0011 1100		124	7	C	0111 1100
=	61	3	D	0011 1101	}	125	7	D	0111 1101
>	62	3	E	0011 1110	~	126	7	E	0111 1110
?	63	3	F	0011 1111	DEL	127	7	F	0111 1111

Del : Silme

2.5 KOMUTLARIN KODLARLA İFADE EDİLMESİ

Mikroişlemciler çalışmasını, programcının hazırladığı program doğrultusunda yürütür. Programcı da ancak hazırladığı programla arzu edilen işlemi mikroişlemciye yaptırabilir. Programcı, program hazırlarken kullandığı mikroişlemcinin komut setindeki komutları kullanarak program hazırlamalıdır. Üretici firmalar, ürettikleri mikroişlemcinin kullandığı komutları ve diğer tüm teknik özelliklerini ayrıntılı bir el kitabı ile piyasaya sunarlar. Programcı bu el kitabında belirtilen komutlardan faydalanarak programını hazırlar. Kullanılan mikroişlemcinin komut setinde bulunmayan bir komutun kullanılması durumunda, mikroişlemci bu kodu ya tanımlayamaz ya da yanlış tanımlayarak yürütülen programın hatalı çalışmasına sebep olur. Bir mikroişlemcinin komut setindeki komutları kullanarak program yazmaya “**makine dilinde programlama**” denilir. Programların hafızada saklanan son biçimi ikilik düzendedir. Fakat programcı değişik programlama dilleri kullanarak program hazırlayabilir. Tablo 2.4’te değişik programlama dilleri ile yazılmış komutlar görülmektedir.

Tablo 2.4 : Diller arası kodlama farklılıkları

Makine dili ile yazılmış program kodları	Assembly dilinde yazılmış program kodları	Üst düzey bir programlama dili ile yazılmış program kodları
110001110	MVI A,4FH	T=0
111110101	CMA	FOR K=1 TO 10
110111011	INR A	T=T+1
101101011	MOV E,A	NEXT K
101110111	NOP	

(Not: Tablodaki programlama dillerinin komutları, birbirlerinin karşılığı değildir.)

Mikroişlemci kullanılarak gerçekleştirilecek bir sistem için öncelikle bazı tanımların kavranması gerekmektedir. Aşağıdaki bilgi kutusunda makine dili, mikroişlemci ve diğer assembly programlama kavramlarını inceleyiniz.

Kullanılan bir mikroişlemci komutunun hemen ardından o komutun üzerinde işlem yapacağı işlenen bulunmalıdır (komutların tek başlarına kullanıldığı durumların haricinde, çoğunlukla işlenen ile birlikte kullanılırlar). Bir komutla birlikte iki tip işlenen olabilir. Komutun işleyeceği bir işlenen “**adres**” veya “**sayısal/alfasayısal**” bir değer olabilir. Aşağıda aynı komutun, makine ve assembly programlama dillerindeki kodlanmış şekilleri görülmektedir.

Komut	İşlenen	Kullanılan programlama dili
(10010110) ₂	(01011111) ₂	Makine dili komutu (ikilik kodu)
(96) ₁₆	(5F) ₁₆	Makine dili komutu (hex kodu)
LDAA	#\$5F	Assembly dili komutu (sembolik komut)



Bazı Programlama Terimlerinin Anlamları:

Program: Program, bilgisayara istenilen işlemleri yaptırabilmek için hazırlanan komutlar bütünüdür.

Komut: Komut, mikroişlemcinin yorumlayabildiği ve bunu eyleme dönüştürebildiği sözcüklere denir. Komut, mikroişlemcinin yapacağı işlemleri tanımlar.

Program algoritması ve akış diyagramı: İyi bir programcı herhangi bir programı hazırlamadan önce ön hazırlık yapmalıdır. Programın genel görünüşü bu aşamada belli olmalıdır. Bu çalışmaların şematik olarak gösterilmesine "Akış diyagramı" denir.

Makine dili: Mikroişlemcilerin tanımlayıp eyleme dönüştürebildiği (Programın, kullanılan mikroişlemcinin anlayabileceği komutlara göre hazırlanması şartı ile.) "1" ve "0" lardan oluşan belirli bir kod düzenindeki komutlara denir. Mikroişlemciler yalnızca makine dilinden anlarlar. Makine dilinin ikilik tabandaki komutları daha anlaşılır olsun diye onaltılık tabanda gösterilebilir.

$(10000110)_2$
 $(10000110)_2 \rightarrow (86)_{16}$

Assembly dili: Makine dili ile yazılmış komutların, anlamlı kısaltmalarla ifade edilmesine "Assembly dili" denir. Assembly dili makine dilinin kullanım zorluğunda ortaya çıkmıştır. "1" ve "0" lara göre daha kullanışlı olacağı açıktır. Assembly dili, makine dilindeki "1" ve "0" şeklinde komutların anlaşılır bir şekilde gösterilmesinden başka bir şey değildir.

$(10000110)_2 (86)_{16} \rightarrow \text{LDA A}$

Komut kodlarının kısaltmalar şeklinde kullanımına Assembly Mnemonic'leri (Nemonik okunur) denir. "LDA A" aslında İngilizce de "Load Accumulator A" (A akümülatörünü yükle) kelimelerinden kısaltılmıştır. Biz de istersek LDA A yerine YUK A diyebiliriz. Fakat sonuçta böyle bir program kodlamasına gitmek yalnızca program yazma ve geliştirme aşamasında bir kolaylık sağlar. Yine mikroişlemciye programı assembly komutların karşılığı olan ikilik veya onaltılık (HEX) kodlara çevirerek girmek durumundayız. Bu çevirme işlemini biz elle yapabileceğimiz gibi derleyici programlar aracılığı ile de yapabiliriz.

Komut kodu: Assembly dilinde yazılmış komutların, orijinal makine dilinde kodlanmış hâline (ikilik tabandaki veya onaltılık tabandaki gösterimi) verilmiş isimdir. Örneğin assembly dilindeki "LDA A" komutunun, makine dilindeki karşılığı $(10000110)_2$ veya $(86)_{16}$ dır. Bunlar komutun "komut kodu" olarak ifade edilir.

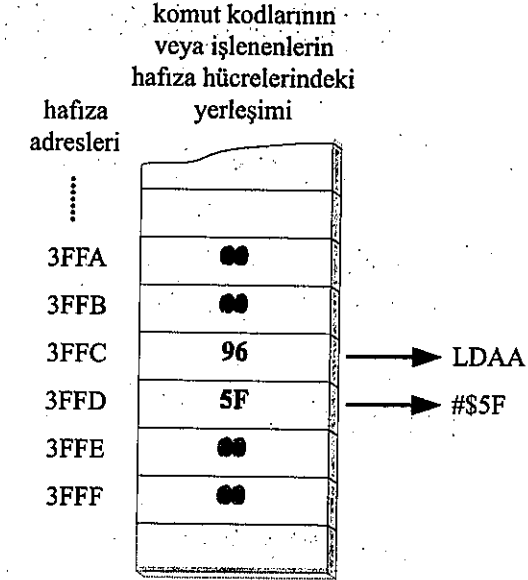
Adres: Mikroişlemci komutunun veya işlenecek verinin bulunduğu hafıza hücresinin fiziksel adresidir.

Veri (data): Mikroişlemci komutlarının üzerinde işlem yapacağı sayısal veya alfasayısal bilgidir.

İşlenen: Mikroişlemci komutları veri veya adres üzerinde aritmetiksel veya mantıksal işlemler yaparlar. Komutların işlem yaptığı "işlenen" bir "adres" olacağı gibi sayısal veya alfasayısal bir "veri" de olabilir. İşlemlerinin yapıldığı adres veya veri genel olarak "işlenen" diye ifade edilir.

Adresleme modu: Bir komutunun üzerinde işlem yapacağı veriler birkaç değişik tiptedir. Komutla kullanılan her değişik veri için komutun işlem kodu farklılık gösterir. Aynı komutun değişik işlenenlerinden dolayı farklı kodlarla kullanılmasına adresleme modu denilmektedir.

Mikroişlemci komutlarının nasıl kodlandığı yukarıda gösterilmiştir. Bu örnekte komut ve işlenen 1'er byte'tır. Bu komut işlenen veri ile birlikte hafızada 2 byte yer kaplar. Mikroişlemci komutları hafızada 1 byte yer tutmasına karşılık verinin 1 veya 2 byte olması muhtemeldir. Eğer işlenen 2 byte'lık ise komutla birlikte düşünüldüğünde, tüm komut hafızada toplam 3 byte yer işgal eder. Şekil 2.5'te 1 byte'lık komutla birlikte kullanılan 1 byte'lık işlenenin, hafızadaki yerleşimi gösterilmiştir.



Şekil 2.5 : Komutlarla birlikte işlenenlerin hafızadaki durumu

Assembly komutunun genel yapısı şöyledir:

Komut [işlenen]

İşlenen, komutun işlevine göre üç şekilde olabilir ;

1. İşlenen 1 veya 2 byte'lık veri olabilir. (\$5F, \$7F05)
2. İşlenen 1 veya 2 byte'lık adres olabilir. (\$5F, \$E005)
3. İşlenen hiç olmayabilir. Komutun kendisi işlem için yeterlidir.

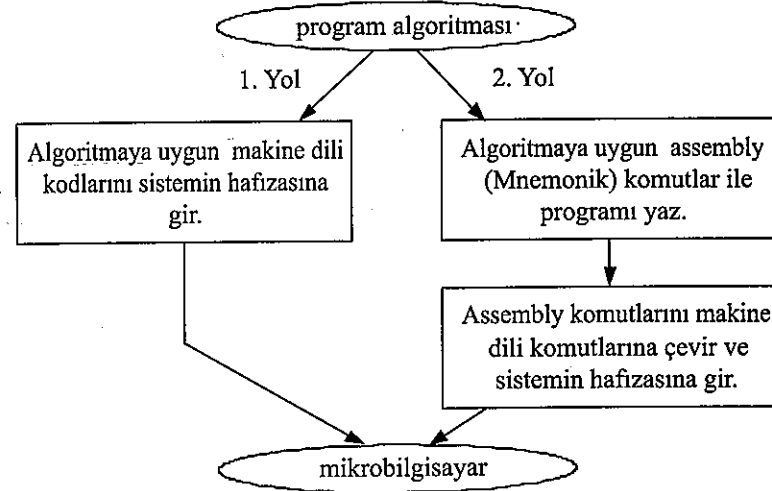


ve \$ Operatörleri:

Diyez sembolü "#" (veri operatörü): İşlenenin bir "veri" olduğunu gösterir. Diyez sembolü olmazsa işlenen bir "adres" tir. Örneğin "\$0100" onaltı tabanında bir veri iken "\$0100" onaltı tabanındaki bir adresi gösterir.

Dolar sembolü "\$" (onaltılık sayı tabanı operatörü): İşlenenin onaltılı tabanındaki bir sayı olduğu anlaşılır.

Mikroişlemcilerde komutlar "1" ve "0" lardan oluşturulan ikilik sayı tabanındaki kodlardan meydana gelir. Bu kodların anlaşılması ve yazılması zor olduğundan, ikilik tabandaki kodlar onaltılık (hex) tabanda gösterilir. Fakat komutlar ve işlenenler ister ikilik tabanda gösterilsin, ister onaltılık tabanda gösterilsin yine de yeterince anlaşılır değildir. Programcının, programın mantıksal işleyişinden çok birtakım sayılarla gösterilen komut kodlarını ve verileri hatasız yazması için aşırı dikkat ve yoğunluk harcaması gerekir. Bunu önlemek için komutları anlamlı sözcüklerle kısaltma yoluna gidilmiştir. Bu tür anlamlı kısaltmalarla yazılan assembly program komutlarına "**mnemonik**" denilmektedir. Assembly programlarının direk makine kodları (1. yol) veya mnemonik komutlarla (2. yol) mikrobilgisayar hafızasına girilmesi şekil 2.6'da gösterilmiştir.



Şekil 2.6 : Komut kodlarının assembly dili ile tanımlanarak kullanılması

Birinci yolun tek işlem basamağı ile yapıldığının görülmesine karşılık tecrübeler bize bu yolla hazırlanan programda yapılacak muhtemel dikkatsizliklerden ötürü bu sürenin daha uzun olduğunu göstermiştir. İkinci yol programcıya sadelik, hız ve daha az hatalı program yazma olanağı sağladığı için tercih edilen yoldur. Mnemonik komutların makine dili karşılıklarını bizim yerimize oluşturan derleme programları mevcuttur. Yani bizim assembly komutları ile hazırladığımız programı "**makine kodlarına**" dönüştüren ve yaptığımız yazım hatalarını (syntax error) bildiren programlardır.

DEĞERLENDİRME ÇALIŞMALARI

- 1) Bit, Nibble, Byte ve Word kavramlarını birer örnek vererek açıklayınız.
- 2) Bilgisayarda depo edilen verilerin büyüklükleri hangi birimlerle ifade edilir? Bu büyüklüklerin üst katlarını yazınız.
- 3) İki sistem arasında gerçekleştirilen veri transferi kaç şekilde gerçekleştirilir? Açıklayınız.
- 4) Veri iletim şekillerinin birbirlerine göre avantaj ve dezavantajlarını belirleyiniz.
- 5) ASCII kodu nedir? Bu kodlama yöntemi nerelerde kullanılır açıklayınız.
- 6) Assembly programlama dilinde bir komut satırında olması gerekli alanları yazınız.
- 7) Assembly programlama dilinde kullanılan, sembolik komut (mnemonic) ve işlem kodu (komut kodu) nedir? Açıklayınız.
- 8) Makine dili veya assembly dili ile program hazırlama arasında ne fark vardır?
- 9) Program nedir? Programı meydana getiren ifadelere ne denir? Açıklayınız.

ÜNİTE 3 MİKRO İŞLEMCİLERE GİRİŞ

3.1 MİKRO İŞLEMCİ TANIMI

1960'lı yıllardan sonra yarı iletken teknolojisindeki gelişmelere paralel olarak elektronikteki hızlı değişim ve uygulama alanlarının çeşitliliği bu bilim dalının değişik kollara ayrılmasına sebep olmuştur. Elektronik, temelde iki ana koldan "analog" ve "dijital" olarak gelişimini sürdürmektedir. Analog ve dijital elektroninin ayrı ayrı kendine göre mantık ve kuralları oluşmuştur. Dijital elektronikteki gelişme sürecinde, Intel Firmasının 1971 yılında "4004" koduyla ürettiği 4 bitlik bir işlemci, ilk mikroişlemcilerin başlangıcı kabul edilebilir. Bu, bir mikrobilgisayar için üretilmemiş olmasına rağmen, çalışması bir işlemcinin temel özelliklerini yansıtmaktadır.

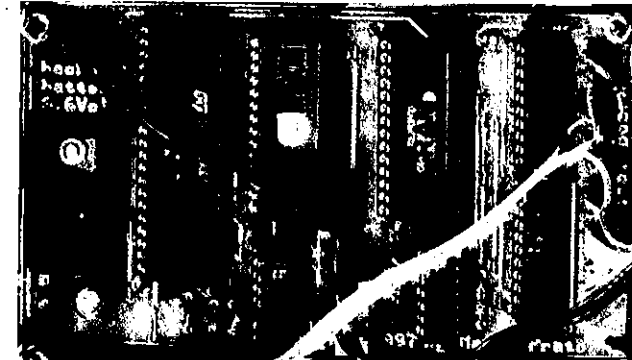
Mikroişlemciler ikilik sayılar üzerinde aritmetiksel ve mantıksal işlemler yapabilen, bu işlemlerin sonuçlarına göre çalışmasını yönlendirebilen veya bu sonuçları belirli hafıza bölgelerine yazıp okuyabilen tümleşik bir devre elemanıdır.

Dijital elektronikteki gelişmeyle ortaya çıkan mikroişlemcilerin asıl popüler olmasındaki gerçek, bu elemanların programlanabilmesidir. Mikroişlemci, çalışmasını kendisi için hazırlanmış olan programın komutlarını tek tek işleyerek yürütür. Mikroişlemci kullanılarak geliştirilen bir sistemin işleyişi üzerinde bir değişiklik gerektiğinde, yapılması gereken tek şey, o sistemdeki mikroişlemci için hazırlanan programın değişen şartlara uygun olarak yeniden programlanmasıdır. Sistemin mevcut durumdaki değişiklikler ve gelişen şartlara uygunluk gösterebilmesi için sistemde çok önemli fiziksel değişiklikler yapmadan, yalnızca aynı işlemciyi yeniden programlayarak zamandan ve maliyetten önemli ölçüde tasarruf sağlanır.

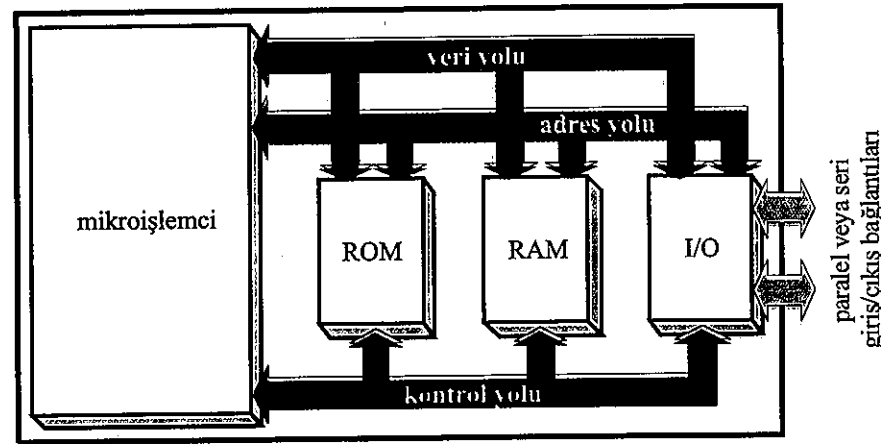
Mikroişlemci kullanılarak yapılan bir cihaz, ayrıntı elemanlarla (lojik kapı ve entegreler) yapılan aynı cihazdan çok daha sade, maliyeti düşük, daha küçük olması ve programlanabilmesi nedeni ile daha modüler ve esnek bir yapıya sahiptir. Tasarım sadeliği, maliyet düşüklüğü ve performans yüksekliği mikroişlemcileri, araştırma geliştirme kuruluşlarının ve ileri teknoloji üreten firmaların, ürünlerinde mutlaka kullanmak istediği bir eleman hâline getirmiştir. Çevremizde gördüğümüz yazar kasalar, fotokopi makineleri, dâhilî ve haricî ve telefon santralleri, elektronik teraziler, elektronik yazı panoları, kat kaloriferi vb. cihazların yanında robotlarda, otomobil endüstrisinde, nükleer santrallerde, uzay çalışmalarında ve tıbbî cihazlarda mikroişlemcili sistemler kullanılmaktadır.

Mikroişlemciler, 1970'li yıllarda (Intel 8085 1974, Motorola MC6800 1975, Mos. Tec. 6502 1976) gelişmeye başlamıştır. Günümüzde çok değişik mimarî, hız ve kapasitede mikroişlemciler bulunmaktadır. Bir sistemde mikroişlemci kullanılarak bir çalışma yapılacaksa kullanılacak olan MİB'nin hızı, mimarî yapısı, adresleme modları, adresleyebileceği maksimum hafıza kapasitesi ve komut seti vb. birçok teknik verinin çok iyi bilinmesi gereklidir.

Mikroişlemci kullanılarak gerçekleştirilen elektronik bir devreye "Mikrobilgisayar" denir. Mikrobilgisayar olarak tanımlanan bir sistemde mutlaka mikroişlemcinin bulunması gerekir. Bunun yanında değişik hafıza elemanları (RAM, EPROM, vb.) giriş/çıkış elemanları da (PIA, ACIA, vb.) bulunur. Resim 3.1'de Zilog Z80 mikroişlemcisi kullanılarak yapılmış bir mikrobilgisayar görülmektedir. Sistemde Z80 mikroişlemcisinin yanında EPROM, RAM ve giriş/çıkış işlemleri (klavye, display...) yapabilmek için uygun giriş/çıkış arabirimleri de bulunmaktadır.



Resim 3.1 : Basit yapıda bir "mikrobilgisayar" örneği

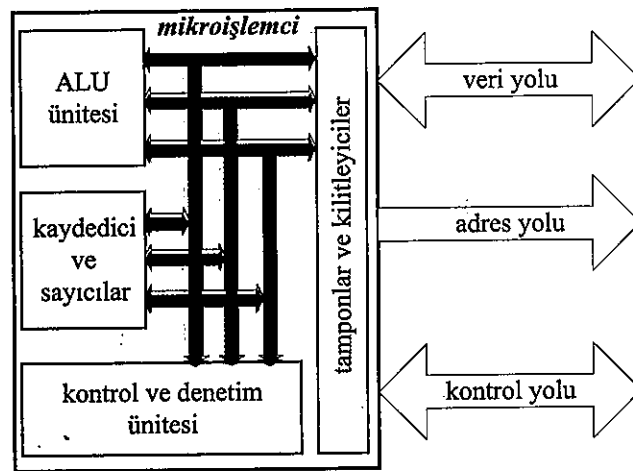


Şekil 3.1 : Bir mikrobilgisayarın blok diyagramı

Mikroişlemciler, endüstrinin çok değişik kollarında kullanıldığı gibi, kişisel bilgisayarlarda da (PC; personel computer) farklı bir gelişim sürdürerek ayrı bir bilim ve teknoloji dalının gelişmesini sağlamıştır. Günümüzde mikroişlemcilerin kullanım alanı ve gelişim seyri iki kola ayrılmıştır. Mikroişlemciler bir taraftan endüstride kontrol ve denetim mekanizmalarında kullanılırken diğer taraftan PC ve diğer büyük kapasiteli bilgisayarlarda kullanılmaktadır.

3.2 MİKROİŞLEMCİ BLOK DİYAGRAMI

Basit yapıdaki bir mikroişlemcinin içyapısı incelendiğinde, temelde dört ana bloktan oluştuğu görülür.



Şekil 3.2: Bir mikroişlemcinin blok diyagramı

- ❖ **ALU (Aritmetik Lojik Unit):** MİB içerisinde aritmetiksel ve mantıksal işlemlerin yapıldığı kısımdır.
- ❖ **Kontrol ve Denetim Merkezi:** Komutların çözülüp işletilmesini sağlayan ve MİB içerisindeki ve dışarıdaki kontrol sinyallerini oluşturan veya değerlendiren ünedir.
- ❖ **Kaydediciler (Registers) ve Sayaçlar (Counters):** Programların gerektirdiği durumlarda, MİB'in kullandığı dâhilî geçici hafızalara kaydedici denir. Sayaçlar, program adresi ve yığın adresi gibi bilgileri saklayan hafıza hücreleridir.

- ❖ **Tamponlar (Buffers) ve Kilitleyiciler (Latches):** Tamponlar, MİB'i çevresel birimlerin olumsuz etkilerinden korur. Tutucuların görevi ise MİB'in oluşturduğu adres, veri ve kontrol sinyallerinin bir sonraki değişikliğe kadar saklanmasını sağlamaktır.

3.3 MİKROİŞLEMCİ ÇEŞİTLERİ

Günümüzde çok çeşitli mikroişlemciler üretilmiştir. Ancak bu mikroişlemcileri birtakım teknik özelliklerine göre sınıflara ayrılarak karşılaştırma yapmak mümkündür. Mikroişlemcileri sınıflandırmada kullanılan temel kriterler şunlardır:

- ❖ **Hız:** MİB'lerin çalışması için kalp atışı gibi saat sinyallerine gereksinim vardır. MİB her saat sinyalinde bir sonraki işlem basamağına geçer. MİB'in hızını incelerken, "haricî saat frekansına" ve "komut çevrim sürelerine" bakmak gerekir. Burada sözü edilen "haricî saat frekansı", mikroişlemciye dışarıdan uygulanan osilatörün frekansıdır. "komut çevrim süresi" ise herhangi bir komutun görevini tamamlayabilmesi için geçen süredir. Her mikroişlemcinin "komutun çevrim süresi" farklı sayıda saat periyotları ile tamamlanabilmektedir. Tablo 3.1'de değişik mikroişlemcilerin "komut çevrim süreleri" birbirine denk komutlara bakarak karşılaştırılmıştır. "haricî saat frekansı" ve "saat periyotlarının" mikroişlemcilerin komut çevrim sürelerini nasıl etkilediği gösterilmiştir.

Tablo 3.1 : Karşılaştırmalı hız tablosu

Haricî saat girişi	Mikroişlemci	Komut	Saat periyodu	Süre
2 MHz	R6502A	ADC	2	1 ns
2 MHz	68B00	ADCA	2	1 ns
5 MHz	8085A	ACI	7	1,4 ns
4 MHz	Z80A	ADC	7	1,75 ns

Görüldüğü gibi bir işlemcinin hızını sadece haricî saat frekansına bakarak değerlendirmek yanıltıcı olabilmektedir. Eş değer bir komutu, 2 MHz'lik 68B00 ve R6502A işlemcileri, 4 MHz'lik Z80A ve 5 MHz'lik 8085A işlemcilerinden daha çabuk yürütebilmektedir.

- ❖ **Fiziksel Olarak Adresleyebileceği Maksimum Hafıza Kapasitesi:** Bir işlemcinin adresleyebileceği veya direkt olarak erişebileceği hafıza alanıdır. Bu miktar MİB'in adres hattı sayısına bağlıdır. Bu hattın sayısı tasarlanacak sistemde kullanılacak hafıza miktarını da belirlemektedir. Eğer çok büyük hafıza miktarı gerektiren bir sistem tasarlanacaksa bu noktada adres hattı sayısı önem kazanmaktadır. Örneğin Motorola 6802, Zilog Z80 ve Intel 8085, 16 adres hattına sahiptir ve adresleyebileceği maksimum hafıza $2^{16} = 65,536$ byte'tır (64 Kb). 23 adres hattına sahip Motorola 68000 işlemcisi maksimum $2^{23} = 8,388,608$ byte'lık (8192 Kb) hafıza bölgesini adresleyebilmektedir. İhtiyacımız olan hafıza miktarı 64 Kb'tan büyükse Motorola 68000 işlemcisi gibi adres hattı sayısı çok olan bir işlemciyi tercih etmeliyiz.
- ❖ **İşlem Yapabildiği Veri Boyutu:** Bir MİB'in bir seferde işleyebildiği bit sayısıdır. MİB'lerin tanınmasında bu değer önemi büyüktür ve işlemci seçiminde dikkat edilmesi gereken bir noktadır. Veri yolunun genişliği, toplam işlem süresini dolaylı yoldan etkiler. İşlenen veri sayısının artması aynı sürede yapılan işin miktarını artırmakta, başka bir deyişle aynı işin süresini azaltmaktadır. Özellikle 8 bitlik bir işlemci ile 16 bitlik ve daha büyük sayılar üzerinde işlemler yaparken işlemcinin veri büyüklüğünün önemi daha çok ortaya çıkar. Örneğin 16 bitlik bir işlemci ile 16 bitlik iki sayının toplanması veya çarpılması gibi aritmetik işlemler tek komutla yapılırken 8 bitlik bir işlemcide 16 bitlik sayılar üzerinde aritmetik işlemler için küçük bir program yazmak gerekir. İşlemciler 4 Bitlik, 8 bitlik, 16 bitlik, 32 bitlik olarak sınıflandırılır. Endüstride daha çok 8 ve 16 bitlik mikroişlemcili mikrobilgisayarlar kullanılmaya karşın daha büyük veri kapasitesine ihtiyaç duyan PC'lerde en az 32 bitlik mikroişlemciler kullanılmaktadır.

- ❖ **Komut Sayısı :** Komut sayısının çokluğu, kullanılan işlemci için hazırlanacak programda rahatlık ve esneklik sağlar. Çünkü her komut, işlemci içinde bir işe karşılık gelmektedir. Dolayısıyla programcı bazen birkaç komutla yapmak istediği bir işi bir tek komutla yapabilmektedir.
- ❖ **Adresleme Modları :** Mikroişlemciler adresleme modlarına bağlı kalarak programlanır. Yazılan assembly programlarının makine koduna el ile yapılan çevirileri, yeni başlayanlar için işin en karmaşık noktasıdır. Bu noktada derleyici programlar bizi adresleme modlarının karmaşıklığından kurtarır ve program yazma süresini de önemli ölçüde kısaltır. Her komutun bir veya birden fazla adresleme modu vardır. Adresleme modu; işletilmekte olan komutun işleneni (veri yada adres) hakkında mikroişlemciye bilgi verir. Adresleme modlarının çeşitliliği program hazırlanırken programcıya zaman kazandırır.
- ❖ **Kesme Girişleri:** Mikroişlemciler kullanılarak yapılan çoğu sistemlerde mikroişlemcilerin kesme girişlerinden faydalanılır. Bu girişlerin sayısı, öncelik sırası ve programlama yöntemleri mikroişlemcinin seçiminde önemli bir kısıttır.

Mikroişlemcilerin endüstride çok sık kullanılmasıyla tek amaca yönelik üretimler için tek yonga işlemciler üretilmeye başlanmıştır. Piyasada çok değişik özellikler içeren tek yonga işlemcilerin içinde, genel olarak küçük bir hafıza (bu bazen RAM bazen EPROM bazen de ikisi birlikte bulunmaktadır.) zamanlayıcı (timer), paralel ve seri giriş/çıkış üniteleri, A/D dönüştürücüler yer almaktadır. Daha çok endüstriyel uygulamalarda tercih edilen tek yonga işlemciler, maliyet, tasarım sadeliği, düşük güç sarfiyatı ve az yer kaplaması nedeni ile yoğun bir talep görmüştür. Bu şekilde gerekli olan minimum bileşenleri (mikroişlemci, RAM, EPROM, PIA, ...) içinde barındıran mikroişlemcilere “mikrodenetleyici” (microcontroller) adı verilmiştir.

Birçok firma değişik özelliklerde mikroişlemciler ve mikrodenetleyiciler üretmektedir. Tablo 3.2’de bunlardan bazılarının teknik özellikleri tablo hâlinde verilmiştir. Bunlardan 6802 işlemcisi 6800’de olmayan dâhilî bir osilatöre ve 128 byte’lık bir karalama belleğine (chace) sahiptir. 6805 işlemcisi, EPROM, ROM ve giriş/çıkış birimlerini içinde bulunduran bir modeldir.

Tablo 3.2 : Değişik mikroişlemcilerin karşılaştırmalı teknik özellikleri

Üretici firma	Üretim yılı	Mikroişlemci	Harcı saat frekansı	Veri yolu genişliği	Adres yolu genişliği	Adresleyebileceği maksimum hafıza miktarı	Temel komut sayısı	Genel amaçlı kaydedici sayısı
Intel	1971	4004	0.74 MHz	4 bit	9 bit	640 byte	46	16 adet 4 bit
Intel	1972	8008	0.8 MHz	8 bit	14 bit	16 Kb	48	6 adet 8 bit
Intel	1976	8085	3 MHz	8 bit	16 bit	64 Kb	80	8 adet 8 bit
Intel	1978	8086	5 MHz	16 bit	20 bit	1 Mb	97	16 adet 16 bit
Intel	1982	80286	25 MHz	16 bit	24 bit	16 Mb	141	16 adet 16 bit
Intel	1985	80386 DX	20-40 MHz	32 bit	30 bit	1 Gb	146	8 adet 32 bit
Intel	1989	80486	66 MHz	32 bit	32 bit	4 Gb		
Intel	1993	Pentium	100 MHz	64 bit	32 bit	4 Gb		
Motorola	1974	6800	2 MHz	8 bit	16 bit	64 Kb	72	2 adet 8 bit
Motorola	1979	68000	8 MHz	16 bit	24 bit	16 Mb	56	15 adet 32 bit
Motorola	1983	68010		16 bit	24 bit	16 Mb		32 bit
Motorola	1993	PowerPC	66 Mhz	64 bit	32 bit	4 Gb		32 adet 32bit
Zilog	1975	Z80	4 MHz	8 bit	16 bit	64 Kb	158	16 adet 8 bit
MOS Tec.	1976	6502	2 MHz	8 bit	16 bit	64 Kb	55	2 adet 8 bit

3.3.1 32 Bit Mikroişlemciler

1970’li yıllarda başlayıp 1984 yıllarına kadar üretilen mikroişlemciler için hazırlanan bilgisayar programlarının gelişmesi beraberinde hız ve hafıza gereksinimleri doğurmuştur. 1984 yıllarına kadar üretilen 8 ve 16 bitlik işlemciler yerini artık 32 bitlik mikroişlemciler bırakmaya başlamıştır. 32 bitlik mikroişlemcilerden Motorola 68020 1984, Intel 80386 1985 yıllarında üretilmiştir. 80386 mikroişlemcisi 33 MHz’lik saat hızı ile çalışmakta ve ayrıca içerisinde 275000 transistör bulunmaktadır. Bu yıllarda mikroişlemciler 1 mikron ölçeğine ve hassasiyetine sahip bir teknoloji ile üretilmektedir. 32 bitlik veri işleme kapasitesine sahip bu mikroişlemcilerin adres hattı sayısının da 32 bit olmasına rağmen ofset ve segment adresleme tekniği kullanılarak 2⁴⁶ tera byte’lık bir hafıza bölgesini kullanabilmektedir. 1987 ve 1989 yıllarında üretilen mikroişlemcilerde hafıza yönetim birimi, kayar noktalı hesaplama birimi (FPU) ve önbellek (chace) gibi birimleri mikroişlemci içerisinde alınarak verimliliğin artması sağlanmış oldu. 1989 yılında üretilen 80846X mikroişlemcisinde 1.2 milyon transistör kullanılmış ve 0.8 mikron teknoloji ile üretilmiştir.

1993 yılına üretilen ilk Pentium mikroişlemcinin veri yolu 64 bit olarak üretilmiştir. FPU birimi yeniden tasarlanmış ve bazı aritmetiksel işlemleri 80486’ya göre 10 kat daha hızlı sonuçlandırmıştır. Bu mikroişlemcinin içerisine 3.3 milyon transistör bulunmaktadır ve 0.65 mikron teknolojisi ile üretilmiştir. Pentium mikroişlemcilerin değişik saat frekanslarında (100 ile 200 Mhz) çalışan tipleri üretilmiştir. 1997 yıllarında üretilen Pentium MMX serisi mikroişlemciler multimedia komutlarını içerecek şekilde üretilmiştir. Böylelikle multimedya uygulamalarının performans ihtiyaçları karşılanmıştır. Bu mikroişlemciler ile 233 Mhz’lik saat frekanslarına ulaşılmıştır. Daha sonraları 300...450 Mhz’lik saat hızlarında çalışan Pentium II’ler üretimeye başlandı. PII mikroişlemciler 64 bit veri ve 36 bit adres yoluna sahip olarak üretilmiştir. PII’ler 7.5 milyon transistör sayısına 0.25 mikron üretim teknolojisi kullanılarak imal edilmiştir. Mikroişlemcilerin içerdiği transistör sayısının artırılması ile birlikte ısı problemini ortaya çıkarmıştır. Bunun için mikroişlemcilerin soğutulmasında kullanılan soğutucu metaller ve fanlar mikroişlemcilerle birlikte tüketiciye sunulmuştur.

İlk mikroişlemcilerin üretilmesinden kısa bir süre sonra mikroişlemciler için geliştirilen programların hız ve hafıza gereksinimleri, daha gelişmiş mikroişlemcilerin tasarlanması ihtiyacını doğurması ile başlayan bu serüven günümüzde yeni bir çağın “bilgi çağını” doğmasına sebep olacak kadar dünyayı etkileyen bir teknolojidir. Bilgisayarlar sayesinde internet ortaya çıkmış dünya üzerindeki bilgilerin paylaşıldığı ve tartışıldığı bir zemin olmuştur.

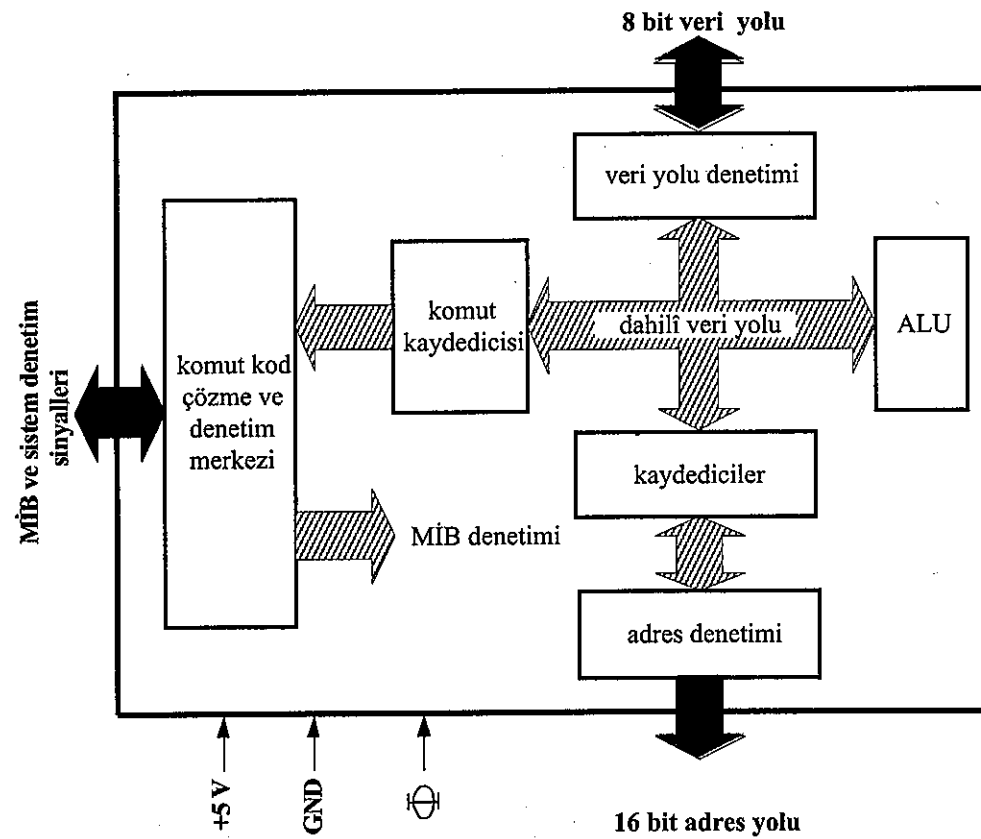
DEĞERLENDİRME ÇALIŞMALARI

- 1) Mikrobilgisayar, mikroişlemci, mikrodenetleyiciler nedir? Arasındaki farkları açıklayınız.
- 2) Temel bir mikrobilgisayarın blok diyagramını çizin. Blok diyagramda tüm birimlerin isimlerini belirtiniz.
- 3) Bir mikrobilgisayarda bulunan temel birimlerin görevlerini yazınız.
- 4) Mikroişlemci ile mikrodenetleyici arasında ne fark vardır? Açıklayınız.
- 5) Mikroişlemciler karşılaştırılmaları hangi kriterlere göre yapılmalıdır?
- 6) Mikroişlemcilerin saat hızları o mikroişlemcinin işlem performansını belirler mi?
- 7) Benzer komutları işleyen, farklı mikroişlemcilerin (değişik saat hızlarına sahip), bu komutları tamamlama sürelerini karşılaştırarak arasındaki farklılıkları veya benzerlikleri tartışınız.
- 8) Piyasada bulunan 8 bitlik mikroişlemcileri araştırınız. Bu mikroişlemciler kullanılarak yapılan sistemleri araştırınız.
- 9) 6802, Z80, 6502, 8085 mikroişlemcilerinin genel teknik özelliklerini karşılaştırınız.
- 10) Okullarımızda bulunan mikroişlemci deney setlerine mikrobilgisayar denebilir mi? Neden.

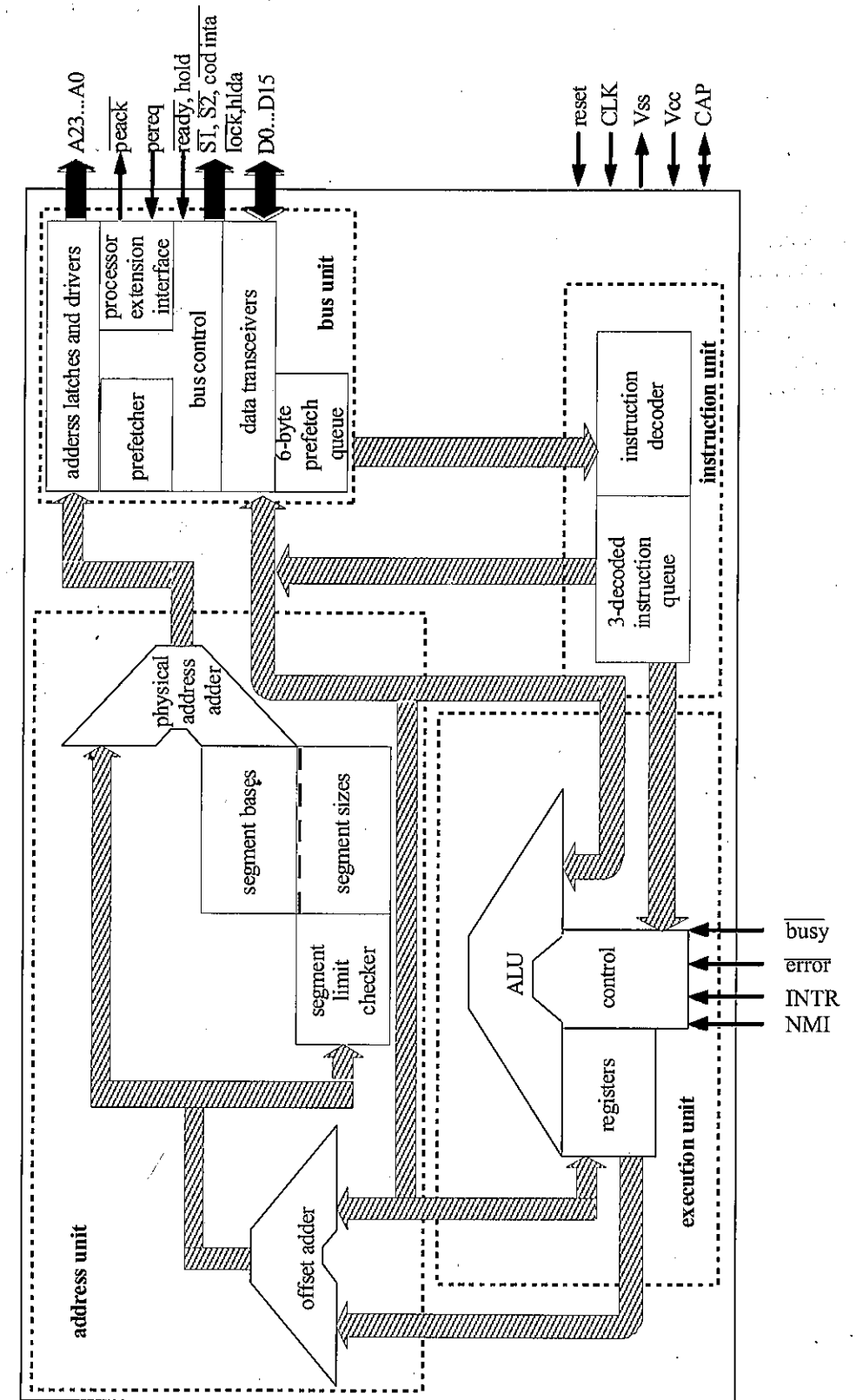
ÜNİTE 4 8 BİT ve 16 BİT MİKROİŞLEMCİLERİN İÇ YAPISI

4.1 MİKROİŞLEMCİLERİN İÇ YAPISI

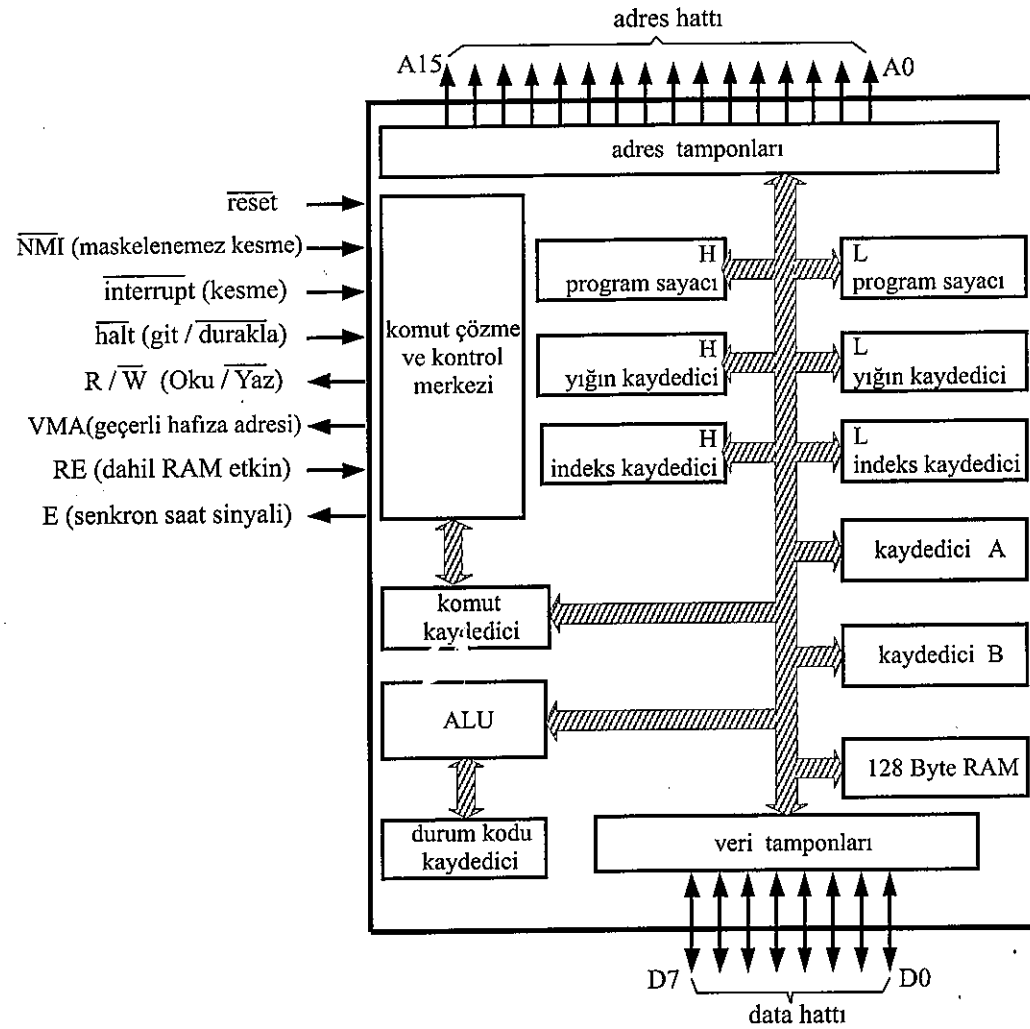
Mikroişlemciler ile herhangi bir sistem veya mikrobilgisayar tasarlanmadan önce kullanılacak mikroişlemcinin hızının, maksimum hafıza kapasitesinin kaç bitlik olduğunun (8 bit, 16 bit), komut setinin ve kaydedicilerinin isteklerimize cevap verip veremeyeceği araştırılmalıdır. Yapılacak işe uygun şartları taşıyan mikroişlemciye karar verdikten sonra o mikroişlemcinin detaylı teknik dokümanlarını elde etmek yararlı olacaktır. Tasarım öncesi mikroişlemci hakkında ne kadar çok bilgi edinirsek tasarım o kadar mükemmel ve kolay olacaktır. Kullanılacak mikroişlemcinin iç yapısını ve çalışmasını çok iyi kavramak gerekir. Aşağıdaki şekillerde (şekil 4.1, 4.2, 4.3) en popüler 8 bit ve 16 bitlik mikroişlemcilerin iç şemaları verilmiştir. Bu şemalar bize diğer mikroişlemcilerin de iç yapısı hakkında genel bilgi verecektir.



Şekil 4.1 : 8 bit Zilog Z80 mikroişlemcinin iç yapısı



Şekil 4.2 : 16 bit Intel 80286 mikroişlemcinin blok diyagramı



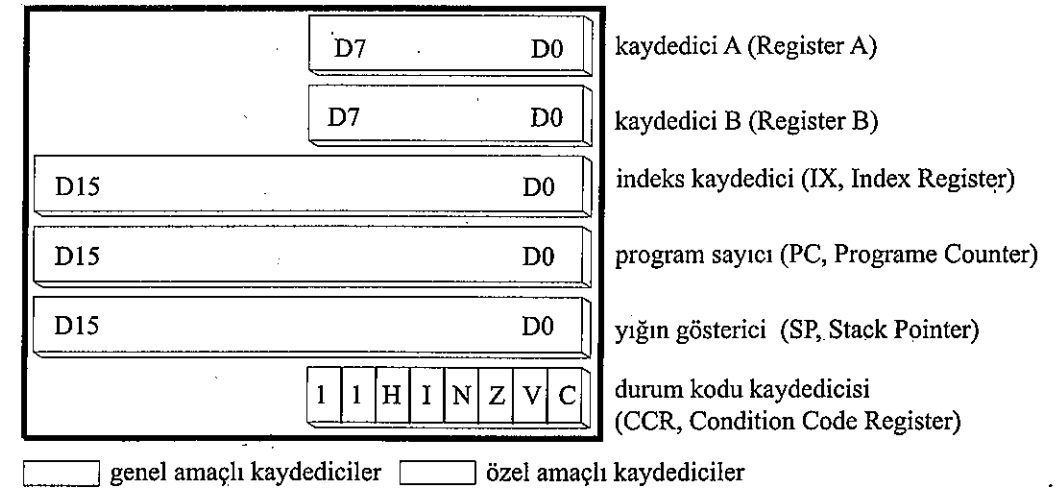
Şekil 4.3 : 8 bit Motorola 6802 mikroişlemcisinin blok diyagramı

4.2 KAYDEDİCİLER

4.2.1 Kaydediciler

Bir programcının “assembly” dili ile program yazımı sırasında en çok ihtiyaç duyduğu geçici hafıza hücreleri, kaydedicilerdir. Mikroişlemcilerde kaydediciler, “genel amaçlı kaydediciler”, “özel amaçlı kaydediciler” ve “segment kaydediciler” olarak üç grupta incelenir. Tüm mikroişlemcilerde bu gruplara dâhil edebileceğimiz değişik özellikte ve sayıda kaydediciler bulunur. Kaydedicilerin sayısının programcının işini kolaylaştırmasının yanında programın daha sade ve anlaşılır olmasını da sağlar. Bazı kaynaklarda kaydedicilere “register”, “yazmaç” veya “yazaç” gibi isimler verilmiştir.

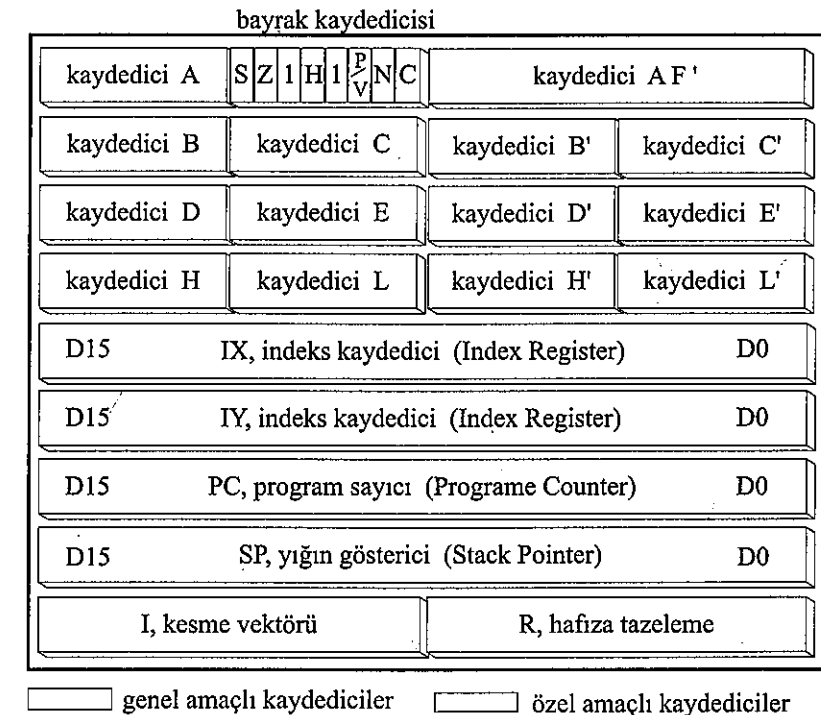
Aşağıda (şekil 4.4, şekil 4.5, şekil 4.11) bazı mikroişlemcisinin kaydedicileri örnek olarak gösterilmiştir.



Şekil 4.4 : 8 Bit mikroişlemcide, kaydediciler (MC6802)

Şekil 4.4. ve şekil 4.5. incelendiğinde, MC6802 mikroişlemcisinde “kaydedici A” ve “kaydedici B” gibi iki tane 8'er bitlik genel amaçlı kaydedici ile toplam 128 byte'lık dâhilî RAM (chace; karalama belleği) bulunur. Zilog Z80 mikroişlemcisinde ise 12 adet 8'er bitlik (B, B', C, C', D, D', E, E', H, H', L, L') genel amaçlı kaydedici çiftleri, 4 adet 8'er bitlik (A, A', F, F') ALU durum kodu kaydedicisi, 2 adet 8'er bitlik hafıza tazeleme ve kesme vektörü kaydedicisi ile 6 adet 16'şar bitlik özel amaçlı kaydediciye sahiptir. Z80 mikroişlemcisi, gerektiğinde 16 bitlik verilerle işlem yapabilmek için 8'er bitlik genel amaçlı kaydedici çiftlerinin, 16 bit olarak düzenlenme imkânı vardır. Bu kaydedicileri çiftlerinin tek parça olarak kullanılması imkânı, hafıza adreslerini tutmada programcıya büyük kolaylık sağlar.

Her mikroişlemcinin kendine has kaydedici yapısı ve kaydedici isimleri vardır. Herhangi bir mikroişlemciyi programlamaya başlamadan önce mutlaka bu kaydedicilerinin isimlerini ve ne tür işlemlere sahip olduklarının iyi bilinmesi gerekir.



Şekil 4.5 : 8 Bitlik mikroişlemcide kaydediciler (Z80).

4.2.2 Genel Amaçlı Kaydediciler (General Purpose Registers)

Programcının üzerinde işlem yapacağı verileri tuttuğu geçici hafıza hücrelerine “genel amaçlı kaydediciler” denir. Bu kaydedicilerin veri boyutu mikroişlemcinin veri işleme kapasitesi ile sınırlıdır. Eğer bir mikroişlemci 8 bitlik bir tasarıma sahipse içindeki kaydedicilerin de veri boyu 8 bit olacaktır. Motorola MC6802 mikroişlemcisinde 2 adet genel amaçlı kaydedici, Zilog Z80 mikroşlemcisinde ise 12 adet genel amaçlı kaydedici bulunmaktadır.

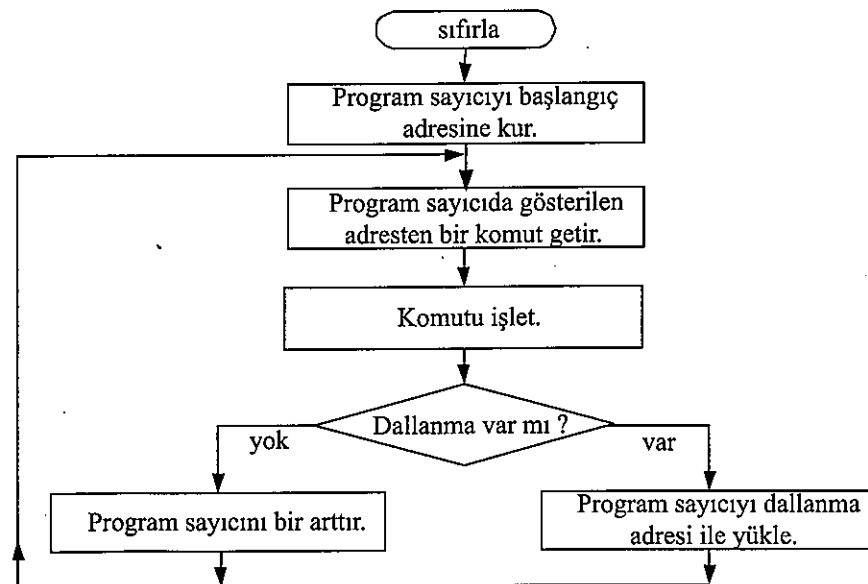
Tüm mikroşlemcilerde programlama sırasında kaydedicilerin kullanılabilmesi için her kaydediciye ayrı ayrı isimler verilmiştir. Bunu, yükleme yapmak istediğimiz kaydedicinin adını belirterek kullanabiliriz. Örneğin 6802 mikroşlemcisinde A kaydedicisine $(3F)_{16}$ sayısını yüklemek istiyorsak yazılacak komut satırı şu şekilde olmalıdır; “LDAA #3F”. B kaydedicisine $(3F)_{16}$ sayısını girmek istiyorsak yazılacak komut satırı ise “LDAB #3F” şeklinde olur. Alt çizgili olarak belirtilen harfler, üzerinde işlem yapılan kaydedicileri işaret etmektedir.

4.2.3 Özel Amaçlı Kaydediciler (Special Purpose Registers)

Özel amaçlı kaydediciler; program adreslerini, ALU’da yapılan bir işlemin sonucunu veya yığın olarak kullanılan hafıza bölgesinin adresini, indisli işlemlerde, indis sayıcı bilgilerinin tutulmasını ve benzeri bilgilerin saklanması için kullanılan kaydedicilerdir. Bu kaydedicilerin de özel adları vardır: IX, PC, SP... Mikroşlemci bu kaydedicilerden bazılarını kendisi düzenlerken bazılarının kullanımı da programcıya bırakılmıştır. Programcı, mikroşlemcideki bütün kaydedicilerin içeriğini okuyabilir, değer girebilir, değişiklik yapabilir veya başlangıç değerleri girebilir. Mikroşlemcilerde bulunan bazı özel amaçlı kaydedicilerden birkaçını inceleyelim.

4.2.3.1 Program sayıcı (PC, Program Counter)

Program sayıcı; mikroşlemcinin yürütmekte olduğu programın adres bilgisini tuttuğu özel amaçlı bir kaydedicidir. Motorola MC6802 ve Zilog Z80 mikroşlemcilerde bu kaydediciler 16 bit uzunluğundadır. Mikroşlemci, hafızadan aldığı her komuttan sonra program sayıcı otomatik olarak bir sonraki işletilecek olan komutun adresini gösterecek şekilde değiştirir. Bir sonraki komutun hangi adreste olacağını o an işletilen program komutu tayin eder, dolayısı ile program sayıcı ona göre düzenlenir. Mikroşlemci bir sonraki komutun adresini bu kaydediciye bakarak belirlemiş olur. Şekil 4.6’da birçok mikroşlemcinin izlediği temel algoritma görülmektedir.



Şekil 4.6 : Bilgisayarların izlediği temel algoritma

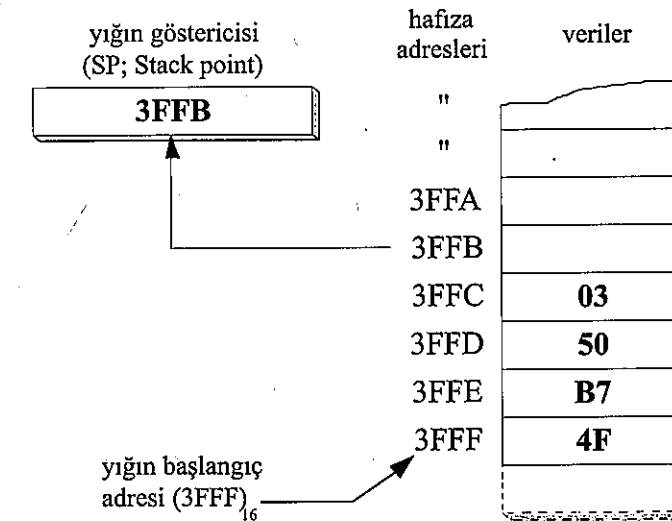
4.2.3.2 İndeks kaydedicisi (IX, Index Registers)

Özellikle belirli bir hafıza bölgesine sıralı bir şekilde ulaşmak istendiğinde bu kaydedici kullanılır. Verilerin belli bir adresten alınıp art arda işlenmesi istendiğinde bu kaydedici çok kullanışlıdır. Programcı verilerin başlangıç adresini yükledikten sonra bu kaydedicinin içeriği yine programdaki direktifler doğrultusunda otomatik olarak değişir. Bu kaydedici de 16 bit uzunluğunda özel amaçlı bir kaydedicidir. Gerekliğinde 16 bitlik uzunluğundaki bilgileri tutmak için genel amaçlı bir kaydedici gibi de kullanılabilir. Z80 mikroşlemcisinde aynı işi yapan iki adet kaydedici bulunmaktadır. Kaydedicilerden biri “IX” diğeri ise “IY” olarak adlandırılmıştır. Bu kaydedicilerden biri indeks adresleme modunda kullanıldığında hafızada saklanacak veya geri alınacak verinin bulunduğu hafıza bölgesinin taban adresini tutmaya yarar (Zilog Z80 ve Motorola MC6802).

4.2.3.3 Yığın göstericisi (SP, Stack Pointer)

Bir mikrobilgisayar oku/yaz belleğin herhangi bir bölümünü yığın olarak kullanabilir. Yığın mikroşlemcinin kullandığı geçici hafıza bölgesi olarak tanımlanabilir. Yığın göstericisi ise yığının adresini tutan özel amaçlı bir kaydedicidir. Yığın kaydedicisi, adres bilgisi göstereceği için 16 bit uzunluğunda bir kaydedicidir. Bu kaydediciye daha programın başında yığının başlangıç adresinin yüklenmesinde fayda vardır. Programcı tarafından, yığının başlangıç adresi, yığın göstericisine yüklendikten sonra artık belleğin bu bölgesi, depo benzeri bir görev yürütür. Yığına veri girişi yapıldıkça yığın göstericisinin değeri de değişir. Yığına her veri girişinde yığın göstericisinin değeri bir azalmakta, yığından her veri çekildiğinde ise yığın göstericisinin değeri otomatik olarak bir artmaktadır. Yığına gönderilen veri, yığın göstericisinin işaret ettiği adresteki hafıza hücresine yazılır. Yığından veri çekilirse yığın göstericisi bir önceki verinin adresini işaret edecek şekilde bir azalacaktır.

Ayrıca yığın göstericisinin çok önemli bir görevi daha vardır. Mikroşlemci yürürlükteki ana programdan alt programa dallandığında veya bir kesme sinyali ile kesme hizmet programına dallandığında, mevcut kaydedicilerin içeriklerini ve dönüş adresini saklayabilmek için otomatik olarak yığına atar. Böylece alt programdan ve kesme hizmet programından ana programa geri dönüşümleri durumunda, ana programda kaldığı yerin adresini ve kaydedicilerdeki verileri kaybetmemiş olur. Eğer programcı alt program veya kesme hizmet programı kullanıyorsa mutlak surette yığın için oku/yaz hafızada bir bölgeyi bu iş için tahsis etmek zorundadır. Bazı mikroşlemcilerin içerisinde, programcının yığın olarak kullanabileceği oku/yaz hafıza bulunabilmektedir. Kullanılan mikroşlemcide böyle bir özelliğin olup olmadığı araştırılmalıdır. Mikroşlemcide böyle bir özellik sunuluyorsa programcının bunu kullanması yararlı olacaktır. Yığın göstericisinin çalışmasını şekil 4.7’de şematik olarak gösterilmiştir (Zilog Z80 ve Motorola MC6802).



Şekil 4.7 : Yığının hafızadaki durumu

Burada yığın olarak belirlenecek hafıza bölgesi adresi $(3FFF)_{16}$ SP kaydedicisine ilk değer olarak yüklenmiştir. Yığma gönderilen 4F, B7, 50, 03 verilerinden sonra SP'nin değeri otomatik olarak $(3FFB)_{16}$ 'yi gösterecek şekilde azaltılmıştır. Her mikroişlemcinin yığma veri yüklemesi ve yığından veri çekmesi için farklı komutları vardır.

Tablo 4.1'de yığının işleyişi sırasında bulundurduğu değerler her adımda gösterilmiştir. Tabloda görüleceği gibi yığma her veri girildiğinde yığın göstercisi bir azalmakta ve yığın, yukarı doğru yükselmektedir. Yığından veri çekildiğinde ise yığın göstercisi artmakta ve yığın aşağı doğru inmektedir.

Tablo 4.1 : Yığın işlemi

İşlem	Yığın göstercisinin (SP) içeriği	Açıklama	Yığındaki son durum
İşlem öncesi durum	SP 010F	Yığın göstercisi, önce herhangi bir adres değeri içerebilir.	010E 00 010F 33 0110 56
Yığın göstercisine $(3FFF)_{16}$ adres bilgisini yükle	SP 3FFF	SP artık $(3FFF)_{16}$ adresini işaret etmektedir. Yığının başlangıç adresi artık $(3FFF)_{16}$ dir.	3FFD 00 3FFE 00 3FFF 00
Yığma $(4F)_{16}$ verisini yükle	SP 3FFE	SP'nin içeriği bir azalmıştır.	3FFD 00 3FFE 00 3FFF 4F
Yığma $(B7)_{16}$ verisini yükle	SP 3FFD	SP'nin içeriği bir azalmıştır.	3FFD 00 3FFD 00 3FFE B7 3FFF 4F
Yığma $(50)_{16}$ verisini yükle	SP 3FFC	SP'nin içeriği bir azalmıştır.	3FFC 00 3FFD 50 3FFE B7 3FFF 4F
Yığından veri çek. En üstteki veri yığından alınır.	SP 3FFD	SP'nin içeriği bir artmıştır.	3FFC 00 3FFD 00 3FFE B7 3FFF 4F

4.2.3.4 Durum kodu kaydedicisi (CCR, Condition Code Register)

Durum kodu kaydedicisi 8 bitlik bir kaydedicidir. Bu kaydedicinin her bir biti ayrı ayrı anlam ifade eder. Mikroişlemci içinde veya dışarıdan yapılan herhangi aritmetiksel, mantıksal veya kesmelerle ilgili işlemlerin sonucuna göre bu bitler ya "set" edilir ya da "reset" edilir. Yani yapılan son işlemin sonucuna göre mikroişlemci ilgili biti ya "1" ya da "0" yapar. Sonraki işlemin sonucunda bu bitler yeni değerler ile yüklenebilir bir önceki değerleri tutamaz. Programcı bu bitlerde oluşacak değerlere göre programa yön verebilir. Durum kodu kaydedicisinde bulunan her bitin hangi işlem sonucunda 1 veya 0 olduğunu mikroişlemcinin kataloğuna bakarak tespit etmek gerekir. Her mikroişlemciye bu bitlerin anlamı farklıdır. 80286 (şekil 4.8), 6802 (şekil 4.9), Z80 (şekil 4.10) mikroişlemcilerinin durum kodu kaydedicilerinin şeması ve her bitin anlamı (tablo 4.2 ve tablo 4.3) verilmiştir. Bu bitlere bayrak adı da verilir. (Zilog Z80 ve Motorola MC6802)



Şekil 4.8 : Durum kodu kaydedicisi bit düzeni "Intel 80286"

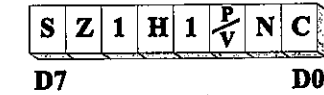
Tablo 4.2 : Durum kodu kaydedicisindeki bitler (Intel 80286 mikroişlemcisi)

Bayrak adı	Açıklama
C	Carry flag
P	Parity flag
A	Auxiliary carry flag
Z	Zero flag
S	Signal flag
T	Trap flag
I	Interrupt enable flag
D	Direction flag
O	Over flow flag
I	I/O privilege level flag
NT	Nested task flag

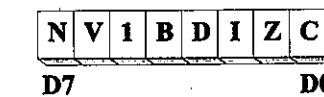
8 bitlik diğer mikroişlemcilerin "durum kodu kaydedicileri"



Şekil 4.9 : Motorola MC6802 "durum kodu kaydedicisi" bit düzeni



Şekil 4.10 : Zilog Z80 "durum kodu kaydedicisi" bit düzeni



Şekil 4.11 : Mos. Tec. 6502 "durum kodu kaydedicisi" bit düzeni

Tablo 4.3 : MC6802 ve Z80 "durum kodu kaydedici" bitlerinin işlevleri

Durum biti	Etkilendiği durum	Örnek
H (Z80 ve 6802)	Yarım elde bayrağı (half carry): Son yapılan toplama veya çıkarma işleminde ilk dört bitin toplamından elde veya çıkarılmasından borç oluşabilir. Bu elde "yarım elde" bayrağı ile gösterilir. BCD tabanlı aritmetiksel işlemlerde kullanışlı bir bayraktır. BCD sayı düzeninde bir basamağın değeri 1001'i aşamaz, aşarsa görüldüğü gibi elde oluşur.	İlk dört bitin toplamından elde oluştuğu için yarım elde bayrağı "H" bitini "1" yapar. $\begin{array}{r} 1000\ 1000 \\ + 0000\ 1000 \\ \hline 1001\ 0000 \end{array}$
I (6802 ve 6502)	Kesme bayrağı (interrupt): Eğer programcı, mikroişlemcinin gelen kesmelere, cevap vermesini istiyorsa bu bayrağı "1" veya "0" yapması gerekir. "kesme denetim bayrağı" programcı tarafından yazılımla "1" veya "0" yapılabilir.	
N (6802, 6502)	Negatif sonuç bayrağı (negative): Son yapılan aritmetiksel veya mantıksal bir işlemin sonucunda negatif bir değer üretilirse bu "1" yapılır. Aksi hâlde "0"dır. İşaretli sayılarda 7. bit "0" ise sayı pozitif, "1" ise sayı negatif anlamındadır.	İşaret bitleri $\begin{array}{r} 1110\ 1010 \\ + 0000\ 1010 \\ \hline 1111\ 0100 \end{array}$ -22 +10 -12
Z (Z80, 6802, 6502)	Sıfır sonuç bayrağı (zero): Son yapılan işlemin sonucunda oluşan değer sıfıra eşitse bu bayrak "1" yapılır. Aksi hâlde 0'dır.	$\begin{array}{r} 1110\ 0001 \\ - 0001\ 1111 \\ \hline 0000\ 0000 \end{array}$
V (6802, 6502)	Taşma bayrağı (over flow): Yapılan aritmetiksel veya mantıksal işlemin sonucu, işaretli sayılardaki 128, +127 aralığını aşarsa bu bayrak "1" yapılır. İşaretli sayılar 7 bittir. 8. bit işaret bitidir.	İşaret bitleri $\begin{array}{r} 0100\ 1110 \\ + 0100\ 0010 \\ \hline 1001\ 0000 \end{array}$ +78 +66 +144
C (Z80, 6802, 6502)	Elde bayrağı (carry): Son yapılan işlemin sonucu eğer 8 bitten fazla ise bu bayrak "1" yapılır. Bir toplama işleminin sonucunda "elde" oluşursa bu bayrak kullanılır. Ancak bu bayrak mantıksal işlemlerde de kullanılır. Genel amaçlı kaydediciler üzerinde döndürme ve öteleme işlemleri yapıldığında taşan bit, 1 veya 0 ne olursa olsun, "elde biti, C" hücresine kayar.	$\begin{array}{r} 1111\ 1111 \\ + 0000\ 0001 \\ \hline 1\ 0000\ 0000 \end{array}$ Bu örnekte sonuç sıfır olduğundan elde bitinin yanı sıra "Z" bitide "1" yapılır.
S (Zilog Z80)	İşaret bayrağı (sign): Son yapılan aritmetiksel işlemin sonucu negatif ise "S" biti "1", pozitifse "0" olur. 6802 mikroişlemcisinde bu iş "N" biti ile görülmektedir.	
P/V (Zilog Z80)	Eşlik/taşma bayrağı (parity/overflow): Mantıksal işlemler "eşlik" ile aritmetiksel işlemler "taşma" ile sonuçlanırsa bu bayrak etkilenir. Mantıksal işlemlerin sonucunda 1'lerin sayısı çift olduğunda "1", tek olduğunda "0" olur. Aritmetiksel işlemlerin sonucu -128, 127 aralığını geçerse bu bayrak yine "1" yapılır.	
N (Zilog Z80)	Toplama/çıkarma (add/subtract): Z80 için bu bitin anlamı BCD sayılarda yapılan son işlem toplama ise "0" çıkarma ise "1" vardır.	
B (6502)	Dur (break): BRK komutunun işlenişini MIB2e bilirmek için kullanılır. Bu bayrak bir komutla "1" veya "0" yapılamaz.	
N (6502)	Onalık mod bayrağı (decimal mod flag): mikroişlemci bu bayrağı "1" gördüğünde aritmetiksel işlemleri BCD moduna göre yapar.	

4.2.3.5 Kesme vektörü (I, Interrupt register)

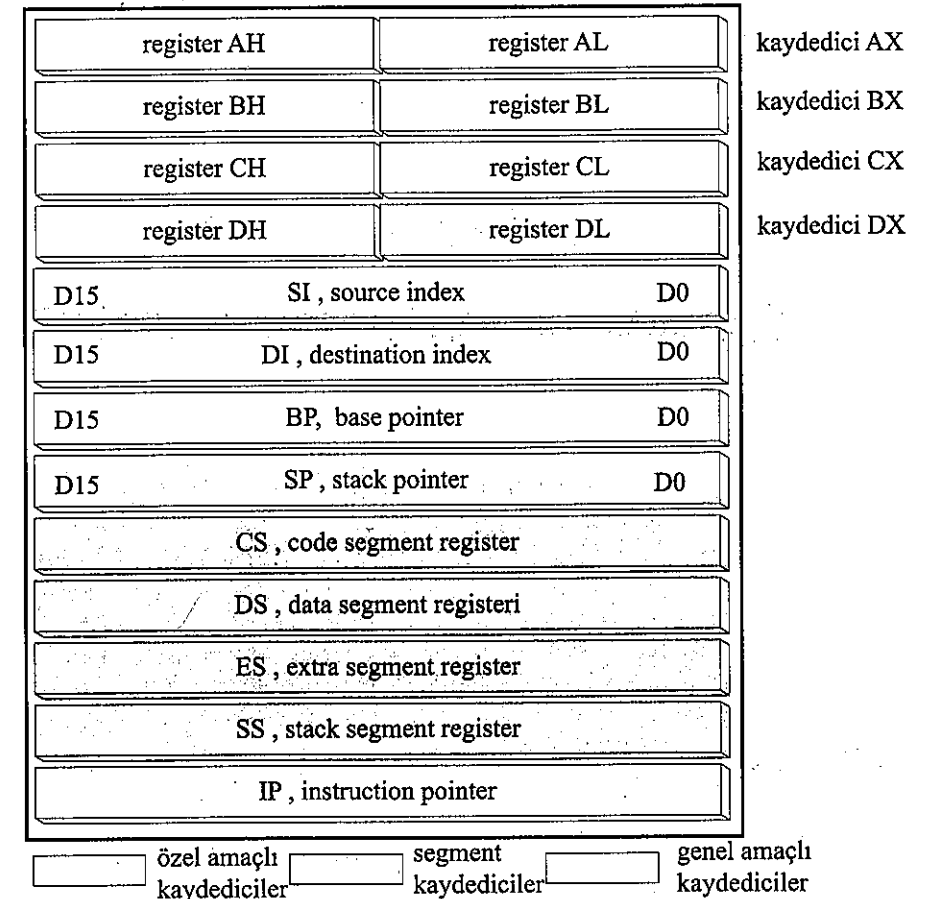
Sadece Z80 mikroişlemcisinde bulunan bir kaydedicidir. Gelen bir kesme sinyaline cevap olarak hafızanın herhangi bir konumuna, dolaylı bir çağrının gerçekleştirilebileceği bir modda çalıştırılabilir. Bu amaçla "I" kaydedicisi dolaylı adresin yüksek değerlikli kısmını saklamak için kullanılır. Dolaylı adresin düşük değerlikli kısmı, kesmeyi yapan birim tarafından veri yoluna bırakılarak sağlanmaktadır. Mikroişlemci kesme hizmet programına dallanacağı zaman, 16 bitlik kesme alt programının, başlangıç adresinin yüksek değerlikli kısmını bu kaydediciden, adresin düşük değerlikli kısmı da kesme sinyalini gönderen çevre birimin veri yoluna bıraktığı değerden alır (Zilog Z80).

4.2.3.6 Hafıza tazeleme kaydedicisi (R, Refresh Memory)

Sistemde "dinamik hafıza" elemanları kullanılmak istendiğinde bu hafızaların belirli periyotlarla tazelenmeleri gerekmektedir. Bu kaydedicideki veri bir tazeleme denetim sinyali ile birlikte adres yolunun ilk 8 biti üzerinden dışarı gönderilerek tazeleme işlemi gerçekleştirilir. Tüm bu işlemler, mikroişlemci bir komutu çözerken ve işletirken yapıldığı için mikroişlemcinin hızını yavaşlatmaz (Zilog Z80).

4.2.4 Segment Kaydediciler

Intel 80286 ve daha sonra çıkan diğer Intel mikroişlemcilerinde bulunan "AX, BX, CX, DX" kaydedicileri, programcının herhangi bir amaç için kullanacağı genel amaçlı kaydedicileridir. Fakat bu mikroişlemcilerde, bazı komutlar bu kaydedicileri belirli görevler için kullanırlar. SI, DI, BP, SP özel amaçlı kaydediciler grubundadır. CS, DS, ES, SS Segment bilgilerini tutan kaydedicilerdir.



Şekil 4.12 : 16 bitlik Intel 80286 mikroişlemcisinde kaydediciler.

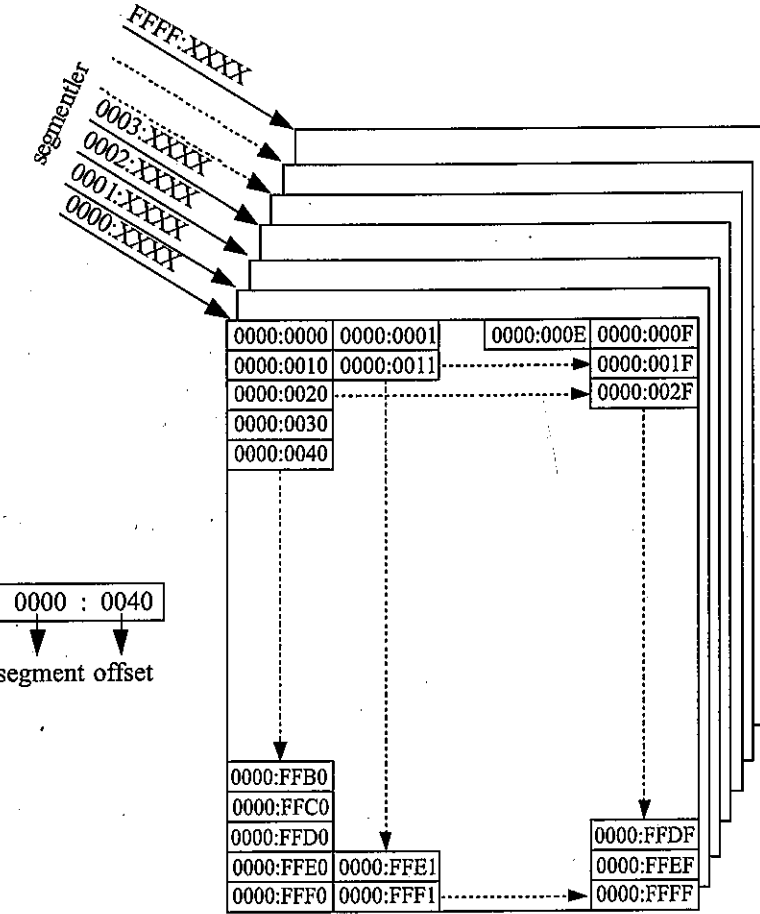
Tablo 4.4 : Intel 80286 kaydedicileri ve görevleri

	Kaydediciler	Açıklama
Genel amaçlı metni	AX	Çarpma, bölme, giriş, çıkış ve bazı string operasyonlarında kullanılır.
	BX	Hafıza içindeki verilerin adreslenmesinde kullanılır.
	CX	Sayaç kaydedicisidir. LOOP komutu veya tekrarlı string işlerinde kullanılır.
	DX	Çarpma ve bölmelerde, giriş/çıkış işlemlerinde port numarasını tutmada kullanılır.
Özel amaçlı kaydediciler	SP (stack pointer)	Çeşitli hafıza bölgelerindeki verilere ulaşmak için kullanılır.
	BP (base pointer)	SP ile birlikte yığın işleri için kullanılır. Ve yığındaki OFFSET adresi taşır.
	SI (source pointer)	Kaynak sıra kaydedicisi.
	DI (destination index)	Amaç sıra kaydedicisi. SI ve DI kaydedicileri veri adresindeki artma ve azalma miktarlarının belirlenmesinde kullanılır.
	IP (instruction pointer)	Bir sonraki işlem görece komutun adresini gösterir. Komut göstericidir.
Segment kaydedicileri	CS (code segment)	Mikroişlemci, yürütmekte olduğu program kodlarının 64 Kb'lık segment adresini bu kaydedici tutar. Burada makine dilindeki program kodları yer alır.
	DS (data segment)	64 Kb'lık veri segmentinin adresini tutar. Genel olarak program değişkenleri tarafından kullanılır.
	SS (stack segment)	Yığın için rezerve edilen adres bölgesinin 64 Kb'lık yığın segment adresini tutar.
	ES (extra segment)	Özel kullanımlar için ayrılmış segment adres kaydedicisidir. Genel olarak string operasyonları için kullanılır.

4.2.4.1 Offset ve segment nedir?

Örneğin 8086/8088 mikroişlemcilerinin adres bacağı 20 tanedir yani adres yolu 20 bittir ($A_{19}...A_0$). Adreslenebilecek maksimum hafıza alanı $2^{20} = 1,048,576$ byte bu da yaklaşık 1Mb'lık bir hafıza alanının fiziksel (devre bağlantısı ile) olarak direkt adreslenebileceğini gösterir. Fakat bu mikroişlemcilerin içerisindeki adres kaydedicileri, 16 bitlik kaydedicilerdir. 16 bitlik bir kaydedici ile en fazla $2^{16} = 65,536$ byte yaklaşık olarak 64 Kb'lık bir hafıza alanı adreslenebilir. Bu mikroişlemcilerin fiziksel olarak adresleyebileceği toplam hafıza alanı 1 Mb olabilirken 16 bitlik adres kaydedicileri ile 1 MB'lık hafıza alanı nasıl adreslenebilecektir? 1Mb'lık hafıza alanlarını gösterebilmek ve direkt olarak adresleyebilmek için en az 20 bitlik adres kaydedicilerin bulunması gerekmektedir. Konuyu kavrayabilmek için öncelikle sorunun ne olduğunu sergiledikten sonra bunun ne şekilde çözümlendiğini inceleyelim.

Bir mikroişlemcinin içerisinde bulunan 16 bitlik kaydediciler ile en fazla 64 Kb'lık hafıza adresleri ile işlem yapabilirken 1 Mb'tan daha büyük bir hafıza alanını adreslemesi durumunda iç kaydedicilerinde 20 bitlik adres değerlerini tutmak için iki ayrı kaydedici kullanılmaktadır. Hafıza adresleri de offset ve segment olarak iki kısma ayrılmıştır. Tüm hafıza alanı her biri 64 Kb'lık olan segmentlere bölünmüştür. Segment içerisindeki her hafıza hücresi de offset denilen numaralama ile adreslenmiştir. Bu tip adreslemede hafızada bulunan herhangi bir hücrenin adresi F100:0100 şeklinde gösterilir. Buradaki F100 hafızanın segment değerini, 0100 de adresin offset değerini göstermektedir.



Şekil 4.13 : Hafızada offset ve segment bloklarının gösterimi

Segmentler 64 Kb'lık bloklardan oluşur. Yani segmentler 64 Kb'lık bir hafıza bölgesidir. Her segment kendi içerisinde adreslenmiştir ve bu adreslemeye "offset" denilir. 000C:7B40 şeklinde adreslenen veri hücresinin gerçek adresi (fiziksel adresi) ise:

$$\text{gerçek adres} = (\text{segment adresi} \cdot 16) + \text{offset adresi}$$

Formülü ile hesaplanabilir. Segment adresini 16 ile çarpmak yerine, sağdan "0" girerek bir basamak sola kaydırıp offset adresi ile toplayarak da gerçek adresi kolayca bulabiliriz. Aşağıda bu işlemin pratik olarak nasıl yapıldığı görülmektedir.

$$\begin{array}{r} 000C:7B40 \\ + \quad 000C0 \leftarrow 0 \\ \hline 07C00 \end{array}$$

000C:7B40 şeklinde adreslenmiş olan bir hafıza hücresinin hafıza haritasındaki gerçek yeri 07C00 olarak bulunur.

4.3 ARİTMETİK MANTIK BİRİMİ (ALU)

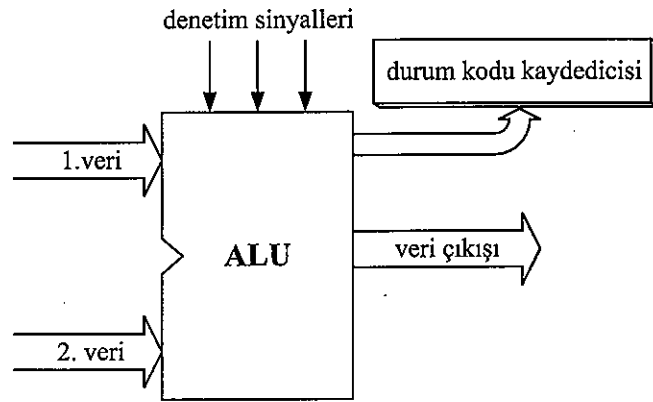
4.3.1 Aritmetik Mantık Biriminin İşlevi

Mikroişlemci biriminin en önemli kısmını ALU (Aritmetik Lojik Unit) ünitesi oluşturur. Bu ünite, kaydediciler üzerinde; toplama, çıkarma, karşılaştırma, kaydırma ve döndürme işlemleri yapar. Yapılan işlemin sonucu kaydediciler üzerinde saklanır bazen de yalnızca “durum kodu kaydedicisini” etkiler. Bazı gelişmiş mikroişlemcilerin içindeki ALU'larda çarpma ve bölme işlemleri yapılabilmektedir. ALU'nun işlem yapabildiği en büyük veri, mikroişlemciye kaydedicilerin veri büyüklüğü ile sınırlıdır. Dolayısıyla 8 bitlik mimariye sahip bir mikroişlemciye ALU en fazla 8 bitlik sayılar üzerinde işlem yapar.

Durum kodu kaydedicisi (CCR) ile ALU direkt olarak bağlantılı ve bir bakıma ALU'ya ait bir kaydedicidir. Zaten isminin “durum kodu kaydedicisi” olmasının sebebi ALU'daki yapılan işlemin sonucunun (pozitif, negatif, sıfır, eşit vb.) burada saklanmasıdır. ALU'da yapılan her işlem direkt olarak durum kodu kaydedicisindeki ilgili bayrağı etkiler. Programcı için çoğu zaman ALU'da yapılan işlemin sonucunda etkilenen bayrakların durumu daha önemlidir. ALU'da yapılan hangi işlemlerin sonucunun, hangi bayrakları etkilediğini kullanan mikroişlemcinin kataloğuna bakarak anlayabiliriz. ALU'daki bir işlem sonucunda durum kodu kaydedicisindeki bayrakların birkaçı birden etkilenebilir veya hiçbiri etkilenebilir. Burada özellikle şu nokta önemlidir. ALU'da yapılan en son işlemin durum bilgisini gösteren durum kodu kaydedicisindeki ilgili bayraklar her işlemten sonra yeniden düzenlenir. Aşağıda ALU'da yapılan aritmetiksel ve mantıksal işlemleri ayrı ayrı inceleyelim.

4.3.2 Aritmetiksel İşlemler Kavramı

ALU'da yapılan aritmetiksel işlemler mikroişlemcinin yapısına göre çeşitlilik gösterebilir. 8 Bitlik mimariye sahip çoğu mikroişlemcilerde yalnızca toplama ve çıkarma işlemleri yapılabilirken gelişmiş mikroişlemcilerde çarpma, bölme işlemleri ve noktalı sayılar üzerinde matematiksel işlemler yapan ALU'lar bulunmaktadır. Bu tip mikroişlemcilere hazırlanan assembler programı, sadeleştiği gibi karmaşık ve uzun sayıların hesaplanma süresi de kısadır.



Şekil 4.14 : Aritmetik-Mantık biriminin blok diyagramı

Toplama işlemi, iki binary sayının toplanması ve sonucun bir kaydedicide saklanmasıyla son bulur. Programcı, toplanacak sayıların kaynağını ve toplamın saklanacağı kaydediciyi veya hafıza hücreğini program kodlarında belirtir. Böylece mikroişlemci bu komutu işletirken sayıların alınacağı yeri ve sonucun nereye kaydedileceğini bilir. Çıkarma işleminde de bu durum değişmez. Mikroişlemci tarafından aritmetik işlemlerin yapılması için kullanılan komutlar vardır. Bu komutlar her mikroişlemciye farklı olabilir ama yaptığı iş aynıdır.

Tablo 4.5'de sembolik komutlar ile yapılan toplama işlemi gösterilmiştir. Aritmetiksel işlemlerin, mikroişlemcilerin komut setlerinde bulunan gerçek komutlarla nasıl yapıldığı “mikroişlemci komutları” bölümünde ayrıntılı olarak ele alınmıştır.

Tablo 4.5 : TAB (Topla A ile B'yi) komutu ile toplama işlemi

Komut	Etkilediği durum	
YKLA 4F	A kaydedicisi 4F	YKLA 4F; mikroişlemci bu komutu işledikten sonra A kaydedicisi 4F sayısı ile yüklenir.
YKLB 10	B kaydedicisi 10	YKLB 10; mikroişlemci bu komutu işledikten sonra B kaydedicisi 10 sayısı ile yüklenir.
TAB	A kaydedicisi 5F	TAB; bu komut ile A kaydedicisindeki sayı ile B kaydedicisindeki sayı ALU da toplanır ve sonuç A kaydedicisinde saklanır.

YKLA 4F; YüKLe A kaydedicisini 4F sayısı ile yükle.

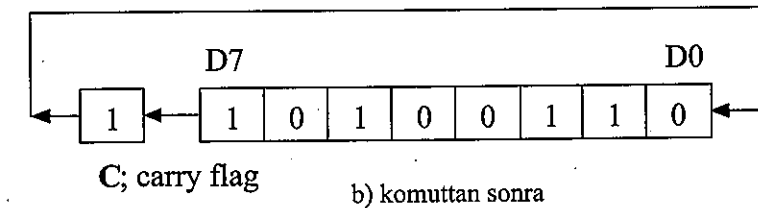
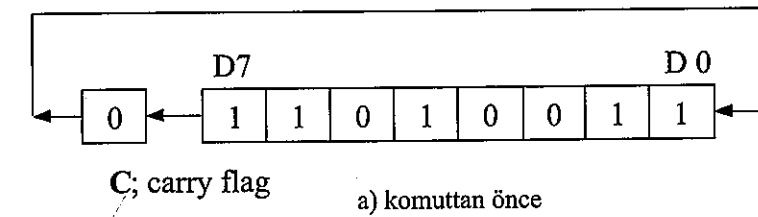
YKLB 10; YüKLe B kaydedicisini 10 sayısı ile yükle.

TAB ; Topla A kaydedicisi ile B kaydedicisini.

4.3.3 Mantıksal İşlemler Kavramı

ALU'larda yalnızca aritmetiksel işlemler yapılmaz. Zaten mikroişlemci ile yapılan birçok uygulamada, “işlenen sayının” değerinden çok “bit düzeni” ile ilgilenilir. Örneğin bir porttan okunan değer, 8 bitlik bir sayıyı değil, harici cihazların durumunu göstermektedir. VE (AND), VEYA (OR), ÖZEL-VEYA (XOR), kaydırma (shift) ve karşılaştırma (compare) gibi birçok mantıksal işlemlerde bu üniteye yapılır. Mikroişlemci, programda mantıksal işlem gerektiren bir komut ile karşılaşıldığında bu sayıyı ALU'ya yönlendirir. ALU da bu sayı üzerinde mantıksal işlemleri gerçekleştirir. ALU'da yapılan örnek bir mantıksal işlem şekil 4.14'de gösterilmiştir. SLDA komutu mantıksal işlemlere örnek vermek için temsili bir komuttur.

SLDA (SoLa Döndür A kaydedicisini) komutu ile mikroişlemciye A kaydedicisindeki veriyi bir bit sola kaydırma emri verilmiştir. A kaydedicisindeki ikilik sayı, 1 bit kaydırılır. Taşan D, biti, durum kodu kaydedicisindeki C (carry) bayrağına kayar. C bayrağındaki bit ise D₀ bitine kayar.

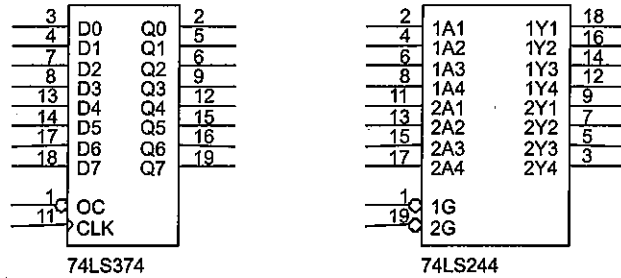


Şekil 4.15 : SLDA komutu ile A kaydedicisini sola döndürme işlemi

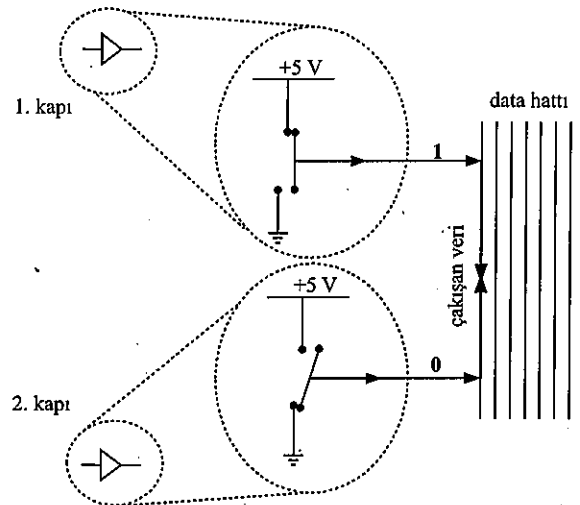
4.4 BUFFER ve MULTIPLEXER

Buffer'ın Türkçe'deki karşılığı tampondur. Mikroişlemci çevre birimleri ile iletişim kurarken adres, veri ve kontrol yolundaki bilgileri bu tamponlar üzerinden gönderir. Adres yolunda bulunan tamponlara “adres tamponu”, veri yolu çıkışındaki tamponlara ise “veri tamponu” adı verilir. Adres tamponu, içeriden dışarıya bilgi akışına izin verecek şekilde tek yönlüdür. Veri tamponu ise çift yönlüdür, dışarıdan içeriye veya içeriden dışarıya bilgi akışı mümkündür. Adres tamponunun tek yönlü olması, hafıza ve çevresel elemanları yalnız kendisinin adresleyebilmesindendir. Aynı zamanda bu tamponlar, çıkıştaki bilgileri bir sonraki değişikliğe kadar tutan “kilitleyici” (latch) devre elemanları içerir.

Mikroişlemcilerin bazıları, sistemdeki adres ve veri yollarını kendisi kullanmadığı zaman diğer çevresel elemanların kullanımına açık tutacak şekilde tasarlanmıştır. Mikroişlemci, bunu bir kontrol sinyali ile adres ve veri tamponlarını kontrol ederek sağlar. Böylece adres ve veri yollarında akan verilerden, mikroişlemci içindeki veriler etkilenmemiş olur. Mikroişlemci diğer çevresel elemanlara kendisini sistemden yalıtıldığını bildiren haricî bir sinyal daha üretir. Diğer çevresel elemanlar bu sinyali kontrol ederek adres ve veri yollarını kullanabilmektedirler. Mikroişlemci adres tamponları tek yönlü ve üç durumlu¹, veri tamponları çift yönlü ve üç durumludur. Dijital elektronikte çok kullanılan bu elemanlardan 74LS374 ve 74LS244 entegreleri benzer bir işlev görür. 74LS374 entegresi kilitleyici ve tampon görevi görürken 74LS244 sadece tampon görevi görür. 74LS244'ün girişindeki veriler ancak 1G ve 2G seçme uçlarına “0” uygulandığında çıkışa taşır.



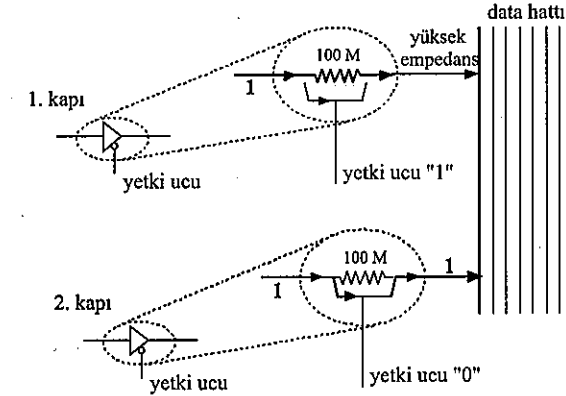
Şekil 4.16 : Tek yönlü kontrollü çıkış veren tampon entegreler



Şekil 4.17 : Yetkisiz (kontROLSÜZ) elemanlarda veri çıkışması

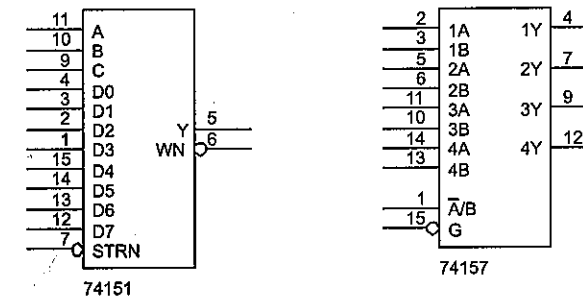
¹ Üç durumlu elemanlar: İletişim hatlarına fiziksel olarak bağlı olan birimlerin port çıkışları üç durumlu kapılar aracılığıyla yüksek empedans durumuna geçerek yolun başka birimlerce kullanılmasına olanak sağlar.

Aynı hattı kullanan elemanların bu hattı kullanmadıkları zamanlarda, yüksek empedans konumuna geçemedikleri takdirde veri yolunda çakışmalara sebep olacaklardır (şekil 4.17). Bu durum elemanların arızalanmasına veya sistemin yanlış çalışmasına sebep olur. Şekil 4.18'de ise veri çıkışları kontrollü elemanların bacak bağlantısı gösterilmiştir. Yetki uçları ile veri yolundan kendilerini yalıtabilirler.

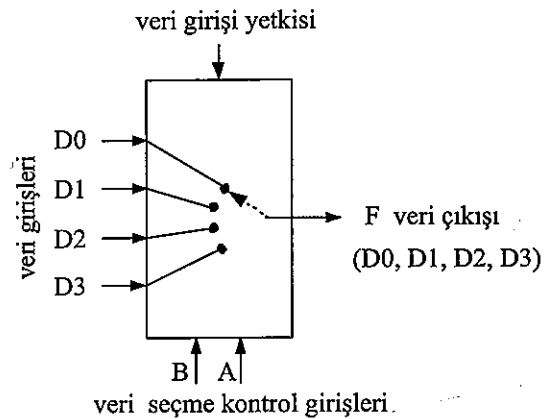


Şekil 4.18 : Yetki ucu ile üç durumlu çıkış verebilen tamponların

Multiplexer (veri seçici), mikroişlemci içindeki ünitelerden o anda ilgili olanını aktifleştirmek için kullanılır. Komut yorumlayıcı tarafından çözülen komutun ardından, ilgili işlemi başlatabilmek için multiplexer aracılığı ile kaydedicilerden, dâhilî yollardan, tamponlardan, ALU ve diğer elemanlardan ilgili olanlar aktifleştirilir veya pasifleştirilir. Multiplexer'in çok sayıda girişi, veri seçme uçları ve bir de çıkışı vardır. Veri seçme uçlarına “adres seçici” de denir. Mikroişlemci içerisinde de şekil 4.18'de gösterilen entegrelerin yaptığı işin aynısını yapan multiplexer elemanlar bulunmaktadır. 74LS151 (içinde 1 tane 8x1) ve 74157 (içinde 4 tane 2x1) entegreleri birer multiplexerdir.



Şekil 4.19 : Multiplexer görevi yapan entegreler



Şekil 4.20 : Multiplexerin çalışma şekli

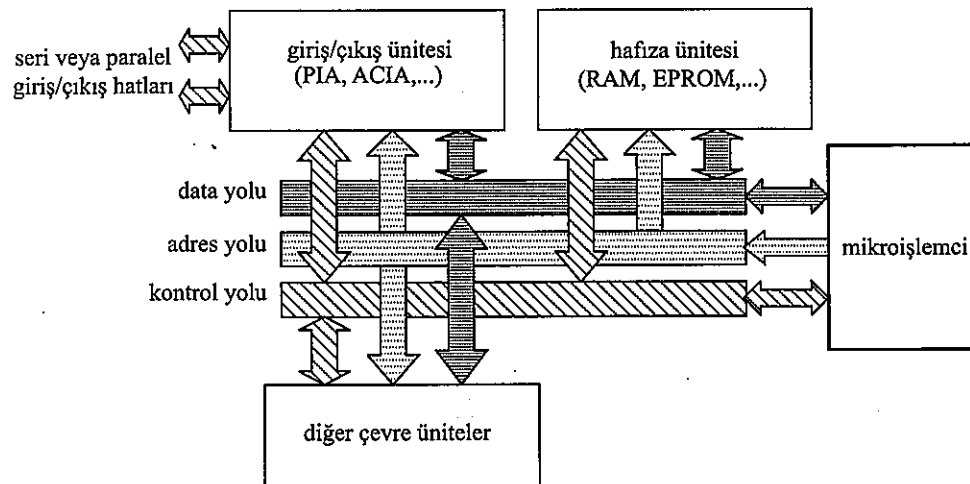
Veri seçme girişleri		Veri girişi yetkisi	Veri çıkışı
B	A		F
0	0	Var	D0
0	1	Var	D1
1	0	Var	D2
1	1	Var	D3

4.5 KONTROL VE DENETİM BİRİMİ

Mikroişlemci işleyeceği komutların işlevini çözüp ona göre eylemi gerçekleştirir. Bunun için de komutları ve verileri hafızadan okuyup getirmesi gerekmektedir. Hafızadan okunan ilk veri mutlaka komuttur. Getirilen bu veri önce mikroişlemci içinde bulunan komut kaydedicisine yüklenir. Komut kaydedicisine yüklendikten sonra bu kod çözülür. Komutun işlevine göre daha sonra işlem devam eder. Kısaca komut kaydedicisi hafızadan getirilen komut kodunun ilk önce yüklendiği ve buradan komut kodu çözümüne çözülme için aktarıldığı mikroişlemci içindeki geçici bir hafıza hücresidir. Kontrol ve denetim biriminin görevi; işletilecek bir komutun, işlevini yerine getirebilmesi için gerekli olan denetim ve kontrol sinyallerini oluşturmaktır. Aslına bakacak olursanız “kontrol ve denetim birimi” karar ve denetim merkezi olmakla bir mikroişlemcinin gerçek fonksiyonelliğini kazanmasında en büyük pay sahibidir.

4.6 İLETİŞİM HATLARI (BUSSES)

Şekil 4.20'de bir mikrobilgisayar sistemindeki, mikroişlemci ile diğer elemanlar arasında iletişim sağlayan adres, veri ve kontrol yollarının ortak kullanım şekli görülmektedir.



Şekil 4.21 : Genel bir mikrobilgisayarda iletişim yolları

4.6.1 Adres Yolu (Address Busses)

Bir mikrobilgisayar sisteminde olması gereken üç anayoldan birisi de adres yollarıdır. Diğer önemli yollar “veri yolu” ve “kontrol yolu”dır. Bu üç yol da mikrobilgisayarda bulunan elemanlar arasında iletişim için gerekli olan sinyalleri taşıyan iletken tellerdir.

Mikroişlemci, yazacağı ve okuyacağı her hafıza hücresine¹ veya çevre birimlerindeki kaydedicilere² erişebilmek için ikilik bir adresleme mantığı kullanır. Adres yolu tek yönlüdür ve bu yolların tüm kontrolü mikroişlemci tarafından sağlanır.

Mikroişlemci okunacak veya yazılacak yerin adresini adres yolu üzerine bırakarak istenilen hafıza hücresine veya kaydediciye erişim sağlar. Buna göre herhangi bir adres değeri, yalnız bir hafıza hücresine veya herhangi bir çevresel birimin kaydedicisine erişecek şekilde belirlenmelidir. Böylece mikroişlemci, programda belirlenen bir hafıza hücresine veya çevresel birim kaydedicisine adres çıkışması olmadan erişir.

Adres yolu aynı zamanda her türden çevre elemanlarının (giriş/çıkış portları, timer'lar, EPROM, RAM, vb.) seçiminde de kullanılır. Örneğin belirli bir adres aralığı EPROM'a atanmasına karşılık başka bir adres aralığı da PIA'nın (paralel input/output birimi) kaydedicilerine erişim sağlar. Bu durum tablo 4.7'de gösterilmiştir.

Tablo 4.7 : (0000)...(FFFF)₁₆ arasında hafıza elemanlarının yerleştirilmesi

(0000) ₁₆ ...(FFFF) ₁₆ adres aralığı	Erişim sağlanan birim
(0000) ₁₆ ... (0FFE) ₁₆	RAM
(0FFF) ₁₆ ... (1FFE) ₁₆	EPROM
(1FFF) ₁₆ ... (2002) ₁₆	PIA
(2003) ₁₆ ... (EFFF) ₁₆	Kullanılmamakta (boş bölge)
(F000) ₁₆ ... (FFFF) ₁₆	EEPROM

Mikroişlemcilerin adresleyebileceği maksimum hafıza bölgesi veya çevresel birimin sayısı adres yolundaki adres hattının sayısına bağlıdır. Adres hattının sayısı 8,12,16,20 veya daha fazla olabilir. 12 bitlik adres hattına sahip bir mikroişlemci en fazla $2^{12} = 4096$ tane hafıza hücresini adresleyebilir. Bir mikroişlemcinin en fazla ne kadar hafıza bölgesini veya çevresel birimi adresleyebileceğini kısaca şöyle hesaplarız:

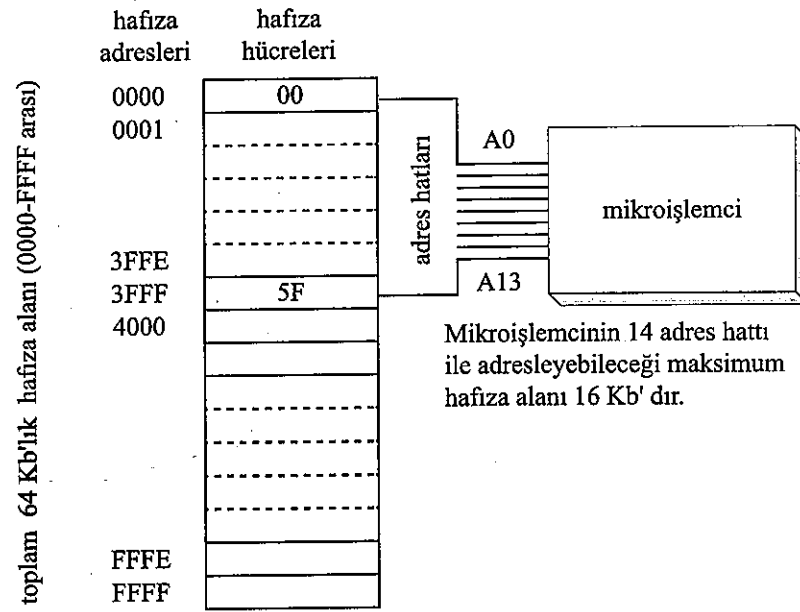
$$2^{(\text{adres hattı sayısı})} = \text{Adreslenebilecek maksimum hafıza aralığı}$$

Örneğin adres hattı sayısı 16 olan bir mikroişlemci $2^{16} = 65,536$ byte'lık (64 Kb) bir alanı adresleyebilir. Eğer maksimum 16Kb'lık bir hafıza bölgesini adresleyebilecek bir mikroişlemciye 64 Kb'lık bir hafıza elemanı bağlanırsa mikroişlemci bu hafızanın yalnızca 16 Kb'lık bölümüne erişebilir, geri kalan 48 Kb'lık hafıza bölgesine erişmesi mümkün değildir.

Her mikroişlemci değişik sayıda adres hattına sahip olabilir. Örneğin Z80, 6802, 8085 ve 8051 gibi mikroişlemciler 16 tane adres hattına sahiptirler ve maksimum 64 Kb'lık bir hafıza bölgesinin tamamını adresleyebilirler. Fakat Intel'in 8086 mikroişlemcisi 20 adet adres hattına, Motorola'nın 68000 mikroişlemcisi 23 adet adres hattına sahiptir.

¹ Hafıza hücresi: RAM, EPROM vb. hafıza elemanlarında 1 byte'lık verinin tutulduğu yer.

² Kaydedici: Gelen anlamı ile MIB, PIA, ACIA, vb. elemanlarda 1 veya 2 byte'lık veri hücresi.



Şekil 4.22 : 14 adres hattına sahip bir mikroişlemcinin adresleyebileceği maksimum hafıza alanı

4.6.2 Veri Yolu (Data Busses)

Bir mikrobilgisayar sisteminde bulunması gereken en temel yollardan biri de “veri yolu”dur. Mikroişlemci veri yolunu, diğer çevresel birimler ve hafıza elemanları arasında bilgi transfer yolu olarak kullanır. Bu yol üzerinde program kodları veya programın kullandığı verilerin çift yönlü akışı gerçekleşir. Veri yolunun sayısı bir seferde ikilik bir bilgidan ne kadarının taşınacağını gösterir. Genellikle 8, 16, 32 adet veya daha fazla paralel hattan oluşur. Çift yönlü bir yoldur. Yani mikroişlemci bu yola hem veri bırakabilir hem de bu yol aracılığı ile diğer çevresel birimlerden ve hafıza elemanlarından veri okuyabilir. Mikroişlemci, bu yolla diğer çevresel elemanlardan veri okumak istediğinde aynı zamanda R/W sinyali R (oku) oluşturarak kontrollü bir okuma gerçekleştirir. Çevresel birimlere veri yazmak istediği zaman R/W sinyalinin ucunu etkinleştirerek diğer çevresel elemanlara veri yazma işlemini gerçekleştireceğini bildirir. Üç durumlu bir yoldur.

Mikroişlemci, haricî elemanlarda okuma ve yazma işlemi gerçekleştirmede zamanlarda bu yol kayar durumdadır yani veri yolu üzerindeki bilgi değişken olabilir. Özellikle gelişmiş sistemlerde mikroişlemci, bu yolu kullanmadığı zaman dilimlerinde, diğer çevresel elemanların bu yolun kullanmasına imkân tanır. Mikroişlemciler, yolun kullanılabileceğini belirten kontrol sinyali (BA; Bus Available) üretir.

4.6.3 Kontrol Yolu (Control Busses)

Kontrol yolu bir bilgisayar sisteminde bulunan tüm elemanların eş zamanlı (senkron) ve kontrollü bir şekilde çalışması için gerekli olan tüm sinyalleri içerir. Kontrol yolunda, senkron çalışmayı sağlayan saat sinyali, okuma/yazma sinyali, kesme (interrupt) ve yeniden başlat (reset) sinyalleri, iletişim hatlarının kullanımını düzenleyen sinyaller gibi kontrol ve denetim sinyallerini içeren hatlar bulunur. Her mikroişlemcinin değişik kontrol sinyalleri bulunur. Bir mikrobilgisayarda, sistemin sağlıklı çalışması kontrol sinyallerinin kullanılmasına bağlıdır.

Örneğin mikroişlemci hafızadan bir veri okumak için önce erişmek istediği hafıza hücresinin adresini adres yoluna koyar. Oku/yaz kontrol yoluna da bu belleği okumak istediğini belirten “oku” sinyali gönderir. Artık adres yolundaki adres ile erişilen hafıza hücresinin içeriği, oku sinyalinden sonra veri yoluna bırakılır. Sonra mikroişlemci veri yolundaki veriyi veri tamponları üzerinden içeri alır.

Mikroişlemci, dışarıda kullandığı kontrol yollarının bir benzerini de içinde bulundurur. Ayrıca her mikroişlemci içinde bulunan üniteler arasında iletişim kurabilmek için “dâhilî adres yolları” ve “dâhilî veri yolları” da kullanılmaktadır. Haricî iletişim yolları bir mikrobilgisayar sisteminin düzenli çalışmasını sağlarken, işleyiş şekli pek farklı olmayan dâhilî yollar ise kendi iç çalışma düzenini sağlar.

DEĞERLENDİRME ÇALIŞMALARI

- 1) Mikroişlemcilerin iç yapısında bulunan temel bileşenlerin görevlerini açıklayınız.
- 2) 8 bitlik bir mikroişlemcinin iç yapısını blok diyagram olarak çizin.
- 3) 16 bitlik bir mikroişlemcinin iç yapısını blok diyagram olarak çizin.
- 4) Kaydedici (register) ne demektir? Mikroişlemci içerisinde görevi nedir?
- 5) Kaydedici çeşitlerini ve kullanım amaçlarını yazınız.
- 6) Aritmetik mantık birimi (ALU) nerede bulunur? Görevi nedir?
- 7) ALU'da yapılan işlemler nelerdir. Bu işlemleri neler olduğunu açıklayarak yazınız.
- 8) Tampon ve multiplexer'lerin çalışmasını anlatınız.
- 9) Tampon ve multiplexer'lerin mikrobilgisayar içerisindeki görevlerini yazınız.
- 10) Bazı lojik kapılar için kullanılan “çift yönlü üç durumlu” ifadesi ne demektir?
- 11) Mikroişlemci içerisindeki kontrol ve denetim birimlerinin görevlerini açıklayınız.
- 12) Adres, veri ve kontrol yollarının görevi nedir? Açıklayınız.
- 13) En fazla 64 Kb'lık bir hafızayı adresleyebilen bir mikroişlemcinin kaç adres hattı vardır?
- 14) 12 adres yoluna sahip 8 bitlik hafıza biriminin kapasitesi ne kadardır?
- 15) Veri yolunun 16 bit olmasının sisteme sağlayacağı avantajları söyleyiniz.
- 16) 6802, Z80, 6502, 8085 mikroişlemcilerinin adres, veri ve kontrol yollarını karşılaştırınız.

ÜNİTE 5 KOMUT ÇALIŞMA EVRELERİ

5.1 KOMUTUN TANIMI VE FORMATI

Mikroişlemciler programlanırken kullanılan programlama dili “assembly”dir. Aslında bütün programlama dilleri (C, Pascal, Basic...) gerçekte mikroişlemcilerin programlanmasını sağlar, fakat hiçbir programlama dili bir mikroişlemcinin tüm özelliklerini doğrudan kullanmaya imkân tanımaz. Üretici firmaların ürettiği her işlemci ailesinin¹ kendi komut seti vardır (Intel, Motorola, Zilog ailesi...) Bu komut setinde yalnız ailesindeki işlemcilerin çözüm eyleme dönüştüreceği komutlar vardır. Programcı, assembly programlama dilini kullanarak bir program yazmaya kalkıştığında, programın çalışacağı sistemdeki işlemcinin komut setini kullanmak zorundadır.

Assembly programlama dili ile bir mikrobilgisayarın ve mikroişlemcinin tüm donanımına doğrudan erişilebilir, istenen düzenleme ve kontrol gerçekleştirilebilir. Assembly dilinin kullanılmasının nedeni donanıma rahat erişilebilmesi ve istenen birimi düzenlemeye ve kontrol etmeye daha etkin bir kullanım sağlamasındandır. Assembly programlama diliyle program yazmak için ayrıca kullanılan mikroişlemcinin ve sistemin mimarî yapısını da çok iyi tanınması gerekirken diğer programlama dillerinde böyle bir zorunluluk yoktur. Assembly dili kullanılarak program hazırlamanın zorluğu, sistemin ve mikroişlemcinin çok iyi bilinmesi gerekliliğindendir. Yoksa assembly programlama dilinde her biri çok kısa ve yalnız bir işlemi gerçekleştiren kısa ve anlaşılması kolay komutlar vardır.

Her mikroişlemci programlanırken kullanılan assembly dili komutları işlemciye göre farklılık gösterir fakat her işlemcide benzer işi yapan bir komut çoğunlukla vardır. Assembly dilindeki komut sayısı, kullanılan mikroişlemcinin komut setindeki komut sayısı kadardır. 8 bitlik bütün mikroişlemcilerin komut setinde bulunan komutların sayısı teorik olarak en fazla 255 olabilir. Gerçekte bu sayı 150....220 civarındadır. Örneğin MC6802 işlemcisinin komut setinde 207 adet komut kodu vardır (Komutların, tüm adresleme modlarındaki kodları da sayılmıştır.).

Assembly programlama dilinin belirli bir yazım düzeni vardır. Hangi mikroişlemci kullanılırsa kullanılsın bu düzen içerisinde hazırlanan program hem anlaşılır hem de olağan hataların bulunup düzeltilmesinde kolaylık sağlar. Tablo 5.1'de genel bir assembly komut satırının düzeni verilmiştir.

Tablo 5.1 : Assembly komut satırı alanları

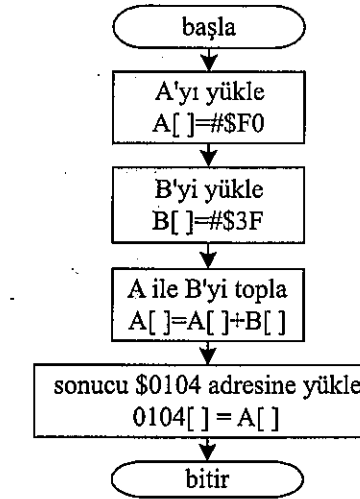
Sıralı sıra numarası	Komutun bulunduğu hafıza adresi	Assembly komutun işlem kodu	Etiket alanı	Assembly komutu alanı	İşlenen değer (adres veya veri)	Komutun yaptığı işin açıklaması
Satır no	Hafıza adresi	İş kodu	Etiket	Assembly komutu	İşlenen	Açıklama
1	0100	86 FF	TEKRAR:	LDA #FF		A Kaydedicisi FF ile yüklenir.

Herhangi bir assembly programı incelendiğinde 7 sütunlu bir düzenle karşılaşılır. İlk sütun komut satırlarının sırasını gösterir. 2. Sütun komutun bulunduğu hafıza adresini gösterir. Bu hafıza adresi komutun bulunduğu hafızadaki gerçek adresidir. İşlem kodu (op-code) assembly komutun makine koduna çevrilmiş hâlidir. Etiket alanı bazı dallanma ve alt programlarda kullanılır ve yer belirtme görevini yapar. 5. sütunda Assembly komutun yazılmasından hemen sonra ise komutun işleyeceği “işlenen”² bulunur. Bu alanda işlenen bir adres olacağı gibi bir veri de olabilir. 7. sütunda komutun yaptığı işin açıklaması bulunur.

¹ İşlemci Ailesi: Mikroişlemci üreten firmaların aynı modelin değişik türlerini ve yan birimlerini (PIA, ACIA...) üretmesi ile oluşan seriye verilen addır.

² İşlenen: Komutun üzerinde işlem yapacağı değer (bir adres bilgisi veya veri).

Aşağıda iki sayının toplanması işlemi için hazırlanmış örnek akış diyagramı (şekil 5.1) görülmektedir. Bu akış diyagramına göre hazırlanmış MC6802 ve Z80 mikroişlemcilerinin assembly programları tablo 5.2 ve 5.3'te ayrı ayrı hazırlanmıştır.



Şekil 5.1 : İki sayının toplanması algoritması

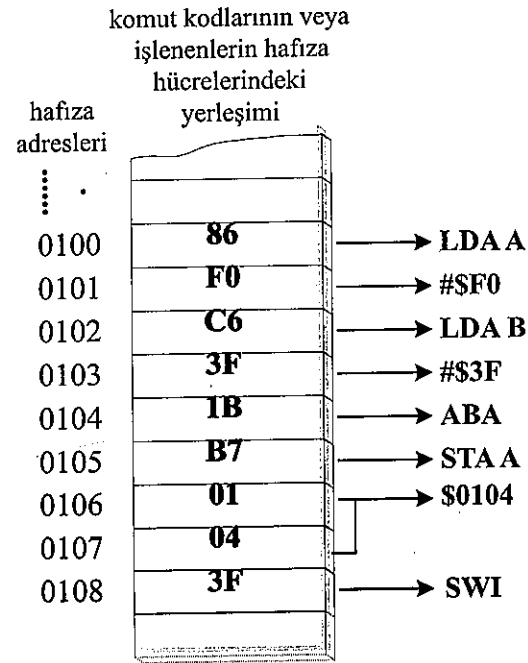
Tablo 5.2 : MC6802 mikroişlemcisi için hazırlanan program örneği

Satır no	Hafıza adresi	İş kodu	Etiket	Assembly komut	İşlenen	Açıklama
1	0100	86 F0		LDA #F0		A kaydedicisine (F0) ₁₆ yüklenir.
2	0102	C6 3F		LDAB #3F		B kaydedicisine (3F) ₁₆ sayısını yükler.
3	0104	1B		ABA		A ile B toplanır A'ya saklanır.
4	0105	B7 01 04		STAA \$0104		A'daki sayı (0104) ₁₆ adresindeki hafıza hücresine yüklenir.
5	0108	3F		SWI		Program sonu komutu

Tablo 5.3 : Zilog Z80 mikroişlemcisi için hazırlanan program örneği

Satır no	Hafıza adresi	İş kodu	Etiket	Assembly komut	İşlenen	Açıklama
1	0100	3E F0		LD A, #F0		A kaydedicisine (F0) ₁₆ yüklenir.
2	0102	06 3F		LD B, #3F		B kaydedicisine (3F) ₁₆ sayısını yükler
3	0104	80		ADD A,B		A ile B toplanır A'ya saklanır
4	0105	30 04 01		LD A, \$0104		A'daki sayı (0104) ₁₆ adresindeki hafıza hücresine yüklenir.

Yukarıdaki her iki programda mikroişlemci içerisindeki kaydedicilere yüklenen sayıların toplanıp sonucun bir adrese yüklenmesi işini yapar. Dikkat edilirse her iki işlemci için yazılan programların aynı işi yapmasına karşılık kullanılan assembly komutları ve bu komutların kodları farklıdır. Şekil 5.2'de, tablo 5.2'deki programın hafızadaki yerleşimi gösterilmiştir.



Şekil 5.2 : Programın hafızadaki görünümü

5.2 KOMUT İŞLEME AŞAMALARI

Mikrobilgisayarlar, hafızalarına yüklenecek program kodlarını adım adım işleterek çalışmalarını yürütürler. Programdaki komut kodları veya veriler hafızanın belli bir bölgesine yerleştirilmiştir (şekil 5.2). Bir mikrobilgisayar çalışmaya başladığı anda, hafızasına yüklenmiş olan programın başladığı adrese giderek buradaki komut kodlarını sırayla işletmeye başlayacaktır. Mikroişlemci bir programı işletirken programda bulunan her komut kodunu önce okuyup içerisindeki komut kaydedicisine yükler. Daha sonra bu kodları çözümler ve kodun emrettiği eylemi gerçekleştirir. Tüm bunlar belirli kontrol ve işlemler zinciri şeklinde belirli zaman dilimlerinde gerçekleşir. Komutların işletilirken üç temel işlem gerçekleşir.

- 1-Komutun bulunup getirilmesi (fetching)
- 2-Komutun çözülmesi (decoding)
- 3-Komutun uygulanması (executing)

5.2.1 Komutun Bulunup Getirilmesi (Fetching)

Bir mikrobilgisayar için hazırlanmış program kodları ve veriler, sistemdeki belli bir hafıza bölgesinde bulunur. Mikroişlemci bir programı yürütmek ve gerekli verilere ulaşmak için öncelikle hafızaya yüklenen program kodlarına erişir ve programdan bir komut getirerek kendi içine yükler. Bu aşamaya “komutun bulunup getirilmesi” denilir.



Mikroişlemciler Çalışmaya Başladığı İlk Anda Ne Yapar ?

Bütün mikroişlemciler ilk çalışmaya başladıkları anda şartsız olarak önceden tanımlanmış bir hafıza adresine daller. Üretici firmalar ürettikleri mikroişlemcilerin bu başlangıç adreslerini önceden kullanıcılarına duyururlar. Çünkü mikroişlemcinin bu davranışı tasarım sırasında fiziksel olarak belirlenmiştir ve sonradan değiştirilemezler. Bu adresin bulunduğu yere “reset vektörü” de denilir. Sistem resetlendiğinde veya ilk defa enerji uygulandığında her zaman aynı adresten çalışmaya başlar. Bazı mikroişlemcilerde (Intel 8051) programı bu başlangıç adresinden itibaren yükleyebileceğimiz gibi bazılarında (Motorola 6802) ise bu adres “sıçrama tahtası” görevi görür. Sıçrama hücresine, programın hafızadaki başlangıç adresi yazılarak mikroişlemcinin bu adrese dalleri sağlanır.

Hafızaya yüklenen tüm programlar istisnasız olarak bir komut ile başlar. İlk program komutunun okunması ile birlikte artık tüm emir ve komuta zinciri programa geçmiştir. Mikroişlemci bundan sonraki tüm çalışmasını program akışındaki komutlara göre yürütür (eğer sistem resetlenmez, kilitlenmez veya fiziksel bir arıza meydana gelmezse).

Bu işlemler sırası ile:

- ❖ Mikroişlemci program sayıcısına bir sonra yürütülecek olan komutun adresini yüklemiştir. PC'de bulunan bu adres, adres tamponları üzerinden adres yoluna bırakılır. Adres yoluna bırakılan bu değer, komutun bulunduğu hafıza hücresine erişilmesini sağlar.
- ❖ Adres yolundaki değerinin gösterdiği hafıza hücresinden okuma yapmak için MİB okuma sinyali gönderir. Okuma sinyalini alan hafıza elemanı, adresi belli olan bu hücredeki değeri veri yoluna bırakır. Veri yolunda artık, belirtilen adresdeki hafıza hücresinin değeri bulunmaktadır.
- ❖ Mikroişlemci veri yolundaki bu değeri direkt olarak veri yolu tamponu üzerinden komut kaydedicisine yükler.
- ❖ Artık programdaki komutun kodu, çözülüp işlenmek için mikroişlemcinin içerisinde.

5.2.2 Komutun çözülmesi (Decoding)

Komut kaydedicisine yüklenen komutun şifresi, “komut kodu çözücüsü” (instruction decoder) tarafından çözülmek için komut kaydedicisinden içeri alınır. Komut kodu çözücüsü tarafından, komutun kodunu çözer ve komutun işlevini yerine getirmek için uygun sinyalleri ve mikro komutları¹ üretir. Komutun çözüldüğünden sonra komutun işlevine göre bazen dışarıdan içeriye veri getirilmesi gereklidir, bazen içeriden dışarı veri çıkarılır bazen de işlemler mikroişlemci içerisinde yürütülür. Komutun çözülmesi için gerekli adımlar şunlardır:

- ❖ Komut kaydedicisindeki komut çözülmek için komut kodu kaydedicisine alınır.
- ❖ Komutun kodu çözüldüğünde, komutun işlevi için gerekli başka verilerin olup olmadığının tespiti bu adımda belirlenir. Hafızada komutun kodunu takiben yer alan ve komutun işlenmesi gereken 1 veya 2 byte daha olabilir.
- ❖ Eğer böyle bir kod çözülmüşse hafızadaki diğer veri byte'ları da birinci aşamadaki gibi getirilip gerekli kaydedicilere yüklenir (Bu işlem komutun fonksiyonuna göre yapılmayabilir.).
- ❖ Tüm veriler getirildikten sonra komutun işlevini eyleme dönüştürmek için gerekli sinyaller ve mikro komutlar üretilerek üçüncü aşamaya geçilir.

¹ Mikro komut: Mikroişlemci içerisinde ve yalnız mikroişlemcinin kendi iç üniteleri arasında eş güdümü ve çalışmayı sağlamak amacıyla kullandığı kodlar veya emir sinyalleri..

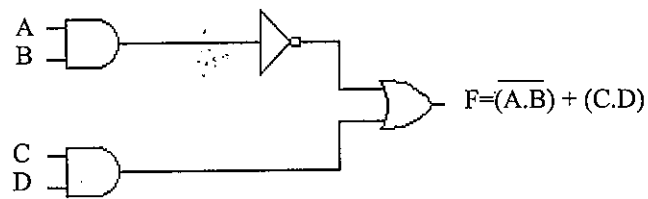
5.2.3 Komutun uygulanması (Executing)

Komutun çözülmesi ile mikroişlemci içerisinde komutun eylemine başlanır. Mikroişlemci mikro komutları da kullanarak kendi içerisinde bulunan alt ünitelerin yönetilip işletilmesini sağlar. Komutun icra edebileceği eylemler; aritmetiksel veya mantıksal işlemler, kaydediciler arasında yüklemeler, alt programlara dallanma, program sayıcısında veya yığındaki değişiklikler ve bazı komutların icrası sonucunda “durum kodu kaydedicisi” bayraklarındaki değişiklikler gibi eylemlerden herhangi birisi olabilir. Komut icrası sırasında bu işlemlerden hangisinin olacağını tabii ki komutun kendisi tayin eder. Üçüncü ve son aşama olarak komutun icrası sırasında şu olaylar oluşur:

- ❖ Kodu çözülen komutun işlevini gerçekleştirmek için gerekli üniteler devreye girer (ALU, kaydediciler, sayıcılar...).
- ❖ Komutun işlevini gerçekleştirdikten sonra gerekirse üzerinde işlediği verileri yine komutun direktifinde belirlenen kaydedicilerde veya harici hafıza hücrelerinde saklar.
- ❖ Komutun işlevine göre durum kodu kaydedicisinde işlemten etkilenen bayraklar “1” veya “0” yapılır.
- ❖ MİB bir sonraki komut kodunu okumak için program sayıcıya bir sonraki komutun bulunduğu adresi yükler. Bunu o an yürütülmekte olan komut bir sonraki komutun da hangi adresten çalıştırılacağını PC’ye (program counter; program sayıcı) yükleyerek belirler.

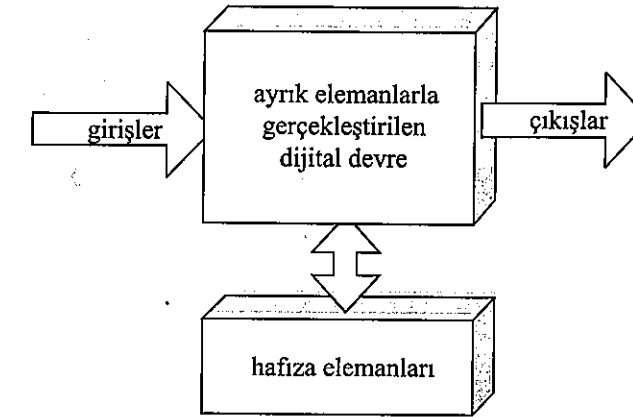
5.3 KOMUT ve ZAMAN İLİŞKİSİ

Komut-zaman ilişkisini incelemekten önce klâsik bir dijital sistemle, mikroişlemcili bir sistem arasındaki işleyiş farkını inceleyelim. Dijital elektronik devreleri çalışması açısından iki şekilde incelenebilir. Birinci gruptaki dijital devreler sabit bir fonksiyonu gerçekleştirmek için tasarlanmıştır. Bu tip devreler, giriş sinyallerindeki değişime bağlı olarak çıkış üretir. Örnek olarak şekil 5.3’deki devre verilmiştir. Bu devrelerin işlem süresi, kullanılan elemanların dâhil çalışma hızları kadardır. Bu devreler, uygulanan giriş bilgilerine çok hızlı cevap verirler. Girişteki verilerin devre tarafından değerlendirilip sonuçlandırılması çok karışık sistemlerde bile 100-800 ns’yi geçmez.



Şekil 5.3 : Ayırık elemanlarla yapılan dijital devre şeması

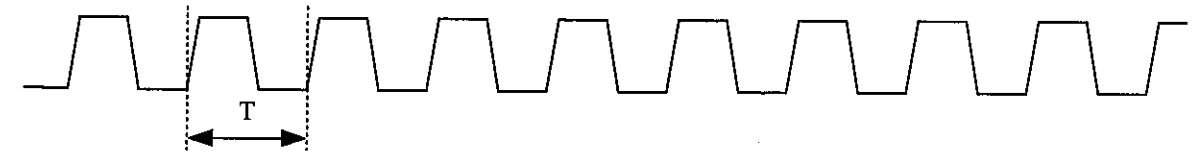
İkinci gruptaki devreler, içerisinde hafıza elemanları bulunan dijital devrelerdir. En küçük hafıza birimi olan “flip-flop” ların çalışmasını hatırlarsak, bu elemanlar bir giriş ve saat sinyali ile durumu değiştirilmediği sürece çıkışındaki bilgiyi sonsuza kadar saklayan elemanlardır. İçerisinde hafıza elemanı bulunan dijital devrelerin çıkışlarındaki değer, girişlerindeki bilgi ve sinyallerin değişiminden başka hafızada bekletilen önceki değerlerin de devre fonksiyonu içerisinde kullanılması sonucu elde edilir.



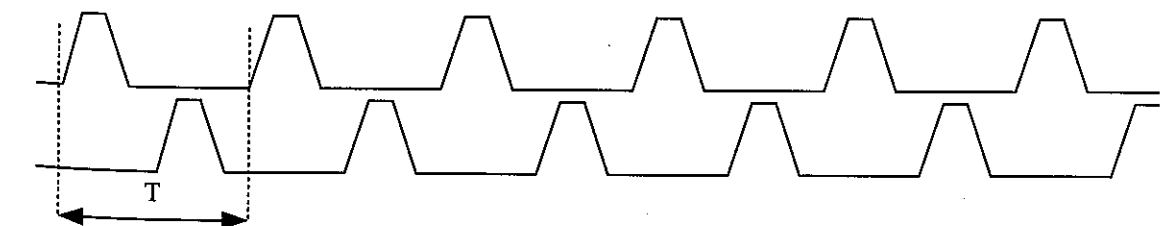
Şekil 5.4 : İçerisinde hafıza elemanları bulunduran dijital bir devrenin blok şeması

Komut-zaman ilişkisi incelenecek olursa; bilindiği gibi hafıza elemanları çalışmasını bir saat sinyali (clock pals) eşliğinde sürdürürler. Bundan dolayı içerisinde bir hafıza elemanı bulunduran dijital devreler çalışmalarını, periyodik bir saat darbesi üreten ve ana saat üreteçleri denilen zamanlama elemanları kullanarak sürdürürler. Mikrobilgisayarlara da genel yapı bakımından içerisinde hafıza elemanları, kilitleyiciler, tamponlar ve örnekleyiciler bulunduran dijital devreler denilebilir. Bir mikrobilgisayarda gerçekleştirilecek tüm işlemler bir “ana saat osilatörüne” senkronize edilir ve zamanlanır. Bu sinyali üreten ünite, işlemlerin gerçekleşmesi için gerekli temel periyodik sinyalleri, mikroişlemcinin kullanımına sunar. Genellikle mikroişlemciler bu sinyalin üretilmesi için dışarıdan bir kristal veya RC osilatör devresine ihtiyaç duyar. Ayrıca farklı mikroişlemciler farklı saat darbelerine ihtiyaç duyabilirler. Aşağıda (şekil 5.5, şekil 5.6, şekil 5.7) değişik MİB’lerinin kullandığı saat darbeleri görülmektedir.

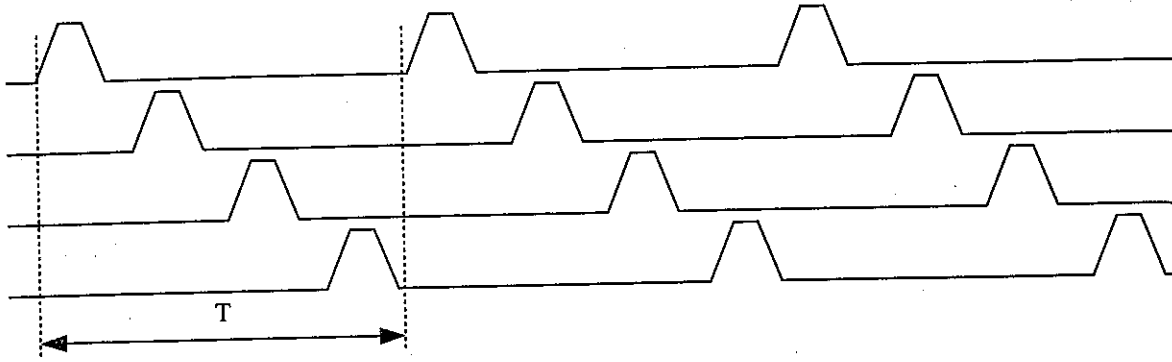
TSM9900 mikroişlemcisi dört fazlı bir saat sinyaline gerek duyar (şekil 5.7) Bu dört ayrı sinyal, işlemcinin dört ayrı bacağına verilir. Bu sinyaller şekillerde (şekil 5.7) görüldüğü gibi zaman ekseninde çakışmayacak şekilde üretilmiştir. Intel 8080A, Motorola 6802 işlemcileri iki fazlı saat sinyaline (şekil 5.6.) gerek duyarken Zilog Z80,Z8000, Intel 8086 ve Motorola 68000 işlemcileri tek fazlı saat sinyaline (şekil 5.5) ihtiyaç duyarlar. Daha önce değinildiği gibi bu sinyaller harici sinyal üretici devrelerden sağlanabilir.



Şekil 5.5 : Tek fazlı saat sinyali



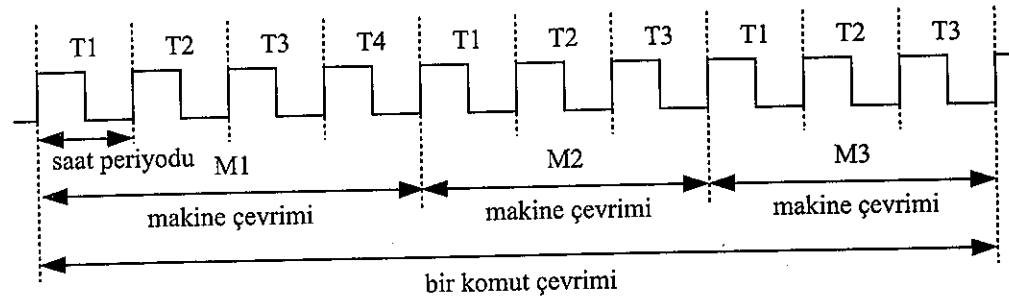
Şekil 5.6 : İki fazlı saat sinyali



Şekil 5.7 : Dört fazlı saat sinyali

Şekil 5.8.'de kesik çizgilerle belirtilen aralıklara (T1,T2,T3,...) "temel saat periyodu" adı verilir ve sembolü "T" birimi saniyedir. Mikroişlemciler herhangi bir komutun işlevini yerine getirmek için birden fazla saat periyoduna ihtiyaç duyarlar. Mikroişlemci üreten firmalar ürettikleri işlemcilerde kullanılan komutların işlevini yerine getirmesi için kaç periyot gerektiğini ürün kataloglarında belirtirler. Bazı zamanlama programlarında, programcı için herhangi bir komutun kaç saat periyodunda işlevini tamamladığı önemlidir.

- T: Mikroişlemcinin temel saat periyodu
M1, M2, M3: Bir komutun işletilmesinde, temel işlemlerin yapıldığı makine çevrimleri
M1; bu zaman diliminde işlem kodunun getirilmesi ve çözülmesi işlemi yapılır (fetching ve decoding).
M2; belleği okuma periyotları (fetching).
M3; belleğe yazma periyotları (executing).



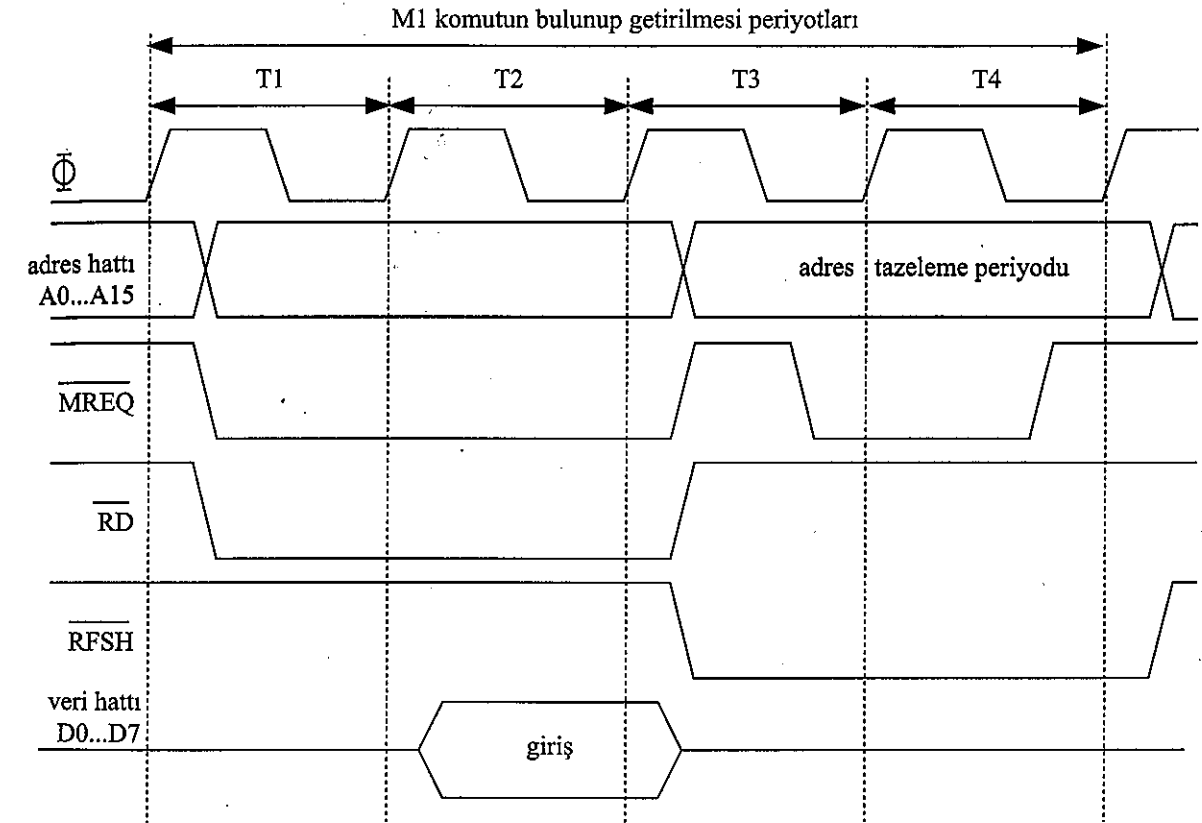
Şekil 5.8 : Bir komutun işlem periyotları

Zamanlama diyagramları, çeşitli kontrol sinyallerinin, veri ve adres yollarının durumlarını ayrıntılı olarak gösteren şekillerdir. Bu diyagramlar temel saat periyotlarına bölünerek incelenmelidir. Her saat periyodunun sonunda kontrol hatlarının durumları, veri ve adres yolları rahatlıkla incelenir.

5.3.1 Komutun bulunup getirilmesi ve zaman ilişkisi (grafığı)

Mikroişlemciler, dışarıdan bir komutun veya verinin getirilmesi için adres, veri ve kontrol yollarında birtakım düzenlemeler yaparlar. Bu düzenlemeleri gösteren diyagram her işlemcide farklılık gösterebilir. Örnek olması bakımından Z80 işlemcisinin bir komutu getirme periyodunu gösteren zaman diyagramı incelenmiştir. Her hattın durumunu tek tek inceleyelim:

- ❖ $\overline{MREQ}$ sinyali bir periyodun yarısına eşit bir zamanda aktif duruma geçer. Bu süre hafızaya uygulanan adresin kararlı olması için yeterli bir süredir.
- ❖ $\overline{RD}$ hafıza okuma sinyali, hafıza elemanından verinin veri yoluna bırakılması için aktif hâle gelir.
- ❖ $\overline{MIB}$, hafızadan veri yoluna bırakılan veriyi T3 yükselen kenarıyla birlikte içeri alır. T3 yükselen kenarı aynı zamanda $\overline{MREQ}$ ve $\overline{RD}$ sinyallerini etkisiz hâle getirir.
- ❖ $\overline{RFSH}$ hattı dinamik hafızaların tazelenmesi için gerekli kontrol sinyalini oluşturur. Bu sinyal T3 ve T4 süresi zarfında oluşur. Burada dikkat edilmesi gereken nokta tazeleme adresi yalnızca $\overline{MREQ}$ süresi boyunca garanti edildiğinden tazeleme sinyali $\overline{MREQ}$ sinyali ile birlikte kullanılmalıdır.

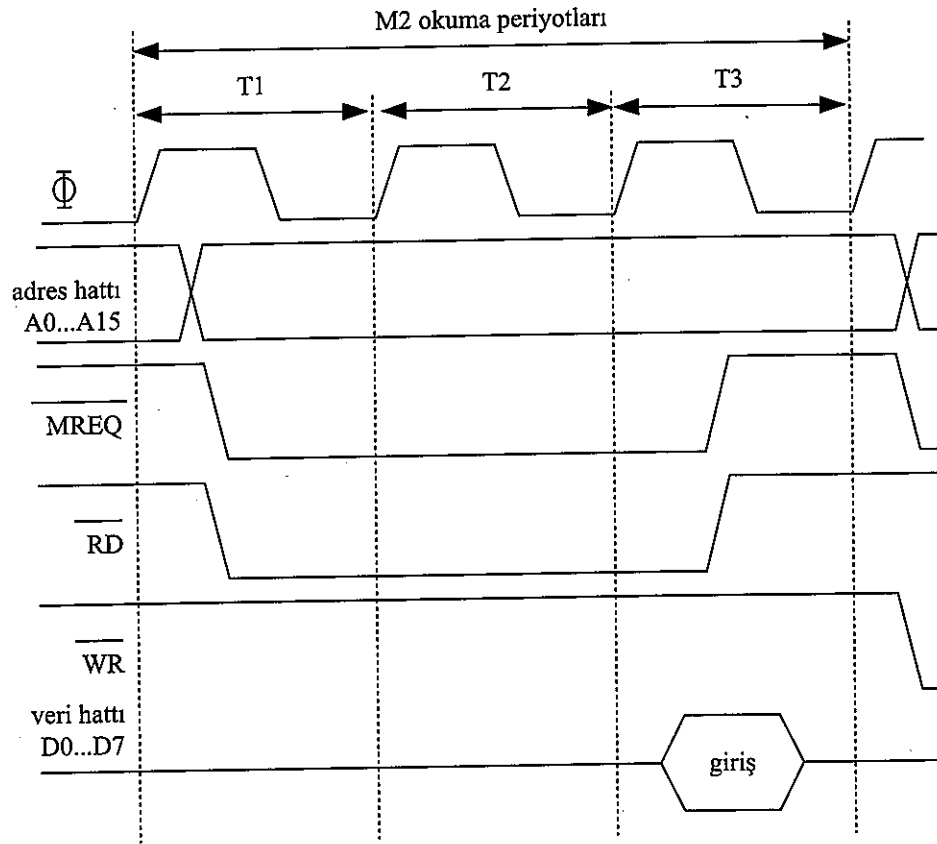


Şekil 5.9 : Komutun bulunup getirilmesi sırasında görülen kontrol sinyalleri

5.3.2 Okuma komutu ve zaman ilişkisi (grafığı)

Şekil 5.10'da hafıza okuma periyotları görülmektedir. Üç periyotta hafıza okuma gerçekleşir. Her periyotta sinyallerin ve yolların durumu görülmektedir.

- ❖ $\overline{MREQ}$ sinyali, iş kodu getirme periyodundaki gibi bir periyodun yarısına eşit bir zamanda aktif duruma geçer. Bu süre belleğe uygulanan adresin kararlı olması için yeterli bir süredir.
- ❖ $\overline{RD}$ hafıza okuma sinyali, hafıza elemanından verinin veri yoluna bırakılması için aktif hâle gelir. $\overline{RD}$ çıkışı, geçerli hafıza adresi sinyali $\overline{MREQ}$ ile birlikte hafızaların yetkilendirme ucu olarak da kullanılabilir.
- ❖ $\overline{WR}$ Belleğe veri yazma çıkışı okuma süresince etkisizdir.
- ❖ Veri yoluna veri, son periyotta bırakılır. $\overline{MREQ}$ ve $\overline{RD}$ sinyallerinin pasifleşmesi ile veri, veri yolunda kaybolur. Adres yolundaki adres bilgisi ise 3. periyodun sonunda kaybolur.

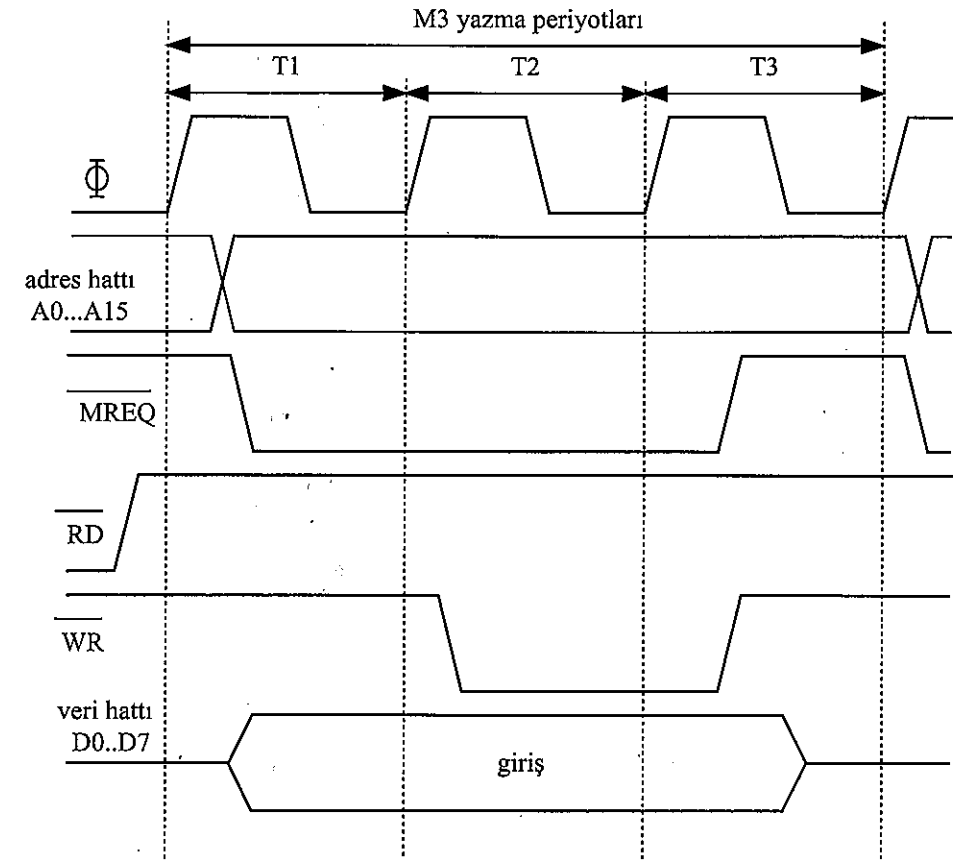


Şekil 5.10 : Hafızadan veri okunması sırasındaki kontrol sinyalleri

5.3.3 Yazma komutu ve zaman ilişkisi (grafığı)

M3 Yazma periyodu da tıpkı okuma periyodu gibi üç periyotta gerçekleşir.

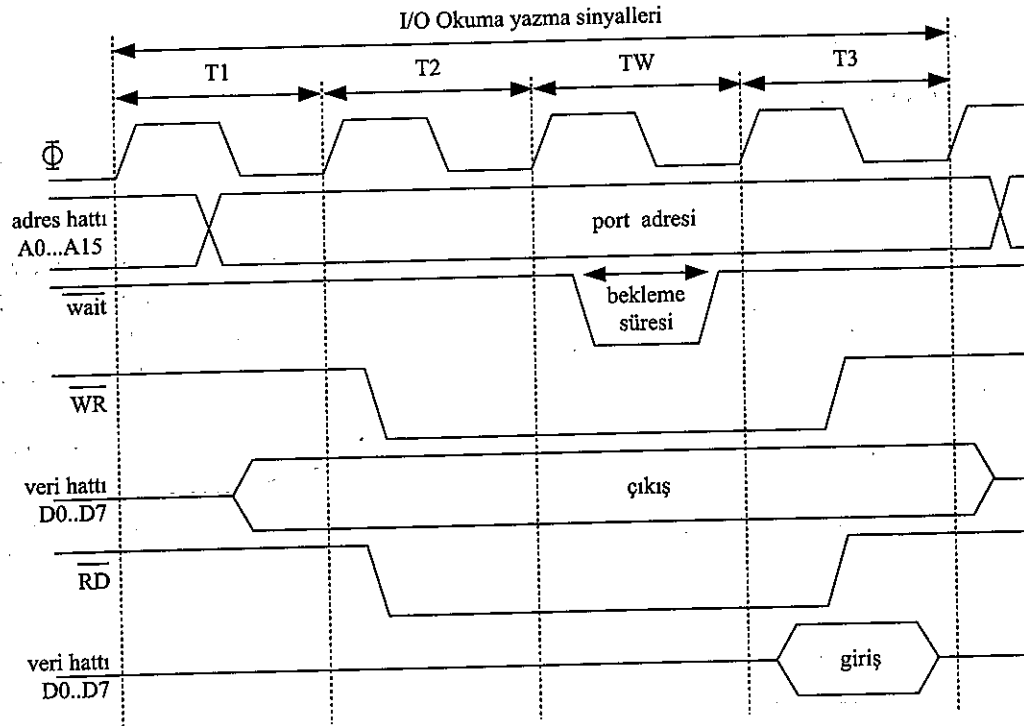
- ❖ MREQ sinyali, iş kodu getirme periyodundaki gibi bir periyodun yarısına eşit bir zamanda aktif duruma geçer. Bu süre belleğe uygulanan adresin kararlı olması için yeterli bir süredir.
- ❖ WR hafıza yazma sinyali, veri yoluna bırakılan verinin, adreslenen hafıza hücresine yazılmasını sağlar. WR çıkışı, geçerli hafıza adresi sinyali (MREQ) ile birlikte hafıza yetkilendirme ucu olarak da kullanılabilir. Dikkat edilirse WR sinyali adres ve veri yolundaki değerler değişmeden yarım periyot önce etkisiz duruma geçer. Böylece yarı iletken hafızalarda, verilerin üst üste binmesini önlemek için gereken gecikme otomatik olarak gerçekleşmiş olur.
- ❖ RD Belleği okuma sinyali de yazma süresince etkisizdir.
- ❖ Veri yolundaki veri, geçerli hafıza adresi sinyali ile hatta bırakılır ve son periyotta hattan çekilir. Veri yolunda verinin yaklaşık üç periyot boyunca bulunması değişik hafıza elemanlarına yazmak için gerekli olan farklı süre ihtiyaçlarını karşılamaktadır. Gerçi hafızaya yazma WR sinyali ile gerçekleşse de geçerli verinin, hafıza elemanın önünde WR sinyalinden önce beklemesi yazma işinin daha güvenilir olmasını sağlar.



Şekil 5.11 : Hafızaya veri yazma sırasındaki kontrol sinyalleri

5.3.4 I/O (Okuma/Yazma) zaman ilişkisi (grafığı)

Şekil 5.12'de bir giriş/çıkış işlemi sırasında oluşan sinyallerinin zaman diyagramları görülmektedir. Giriş/çıkış işlemi yapılacak port'un adresi adres yoluna bırakıldıktan sonra bir okuma veya yazma sinyali ile işlem başlar. Fakat dikkat edilirse TW periyodu bir gecikme süresini gösterir. Bu gecikme işlemci tarafında oluşturulur yani dışarıdan gerçek bir harici gecikme sinyali örneği değildir. Tamamen işlemcinin kendi oluşturduğu bir sinyaldir. Bunun sebebi gerçekte hiçbir giriş/çıkış elemanı mikroişlemci hızında çalışmamaktadır. Bunun için kısa bir gecikme sağlanarak harici giriş/çıkış elemanına süre tanınmaktadır. Giriş/çıkış işlemlerinde gerçekleşen okuma/yazma işlemleri ve oku/yaz sinyalleri tamamen hafıza okuma/yazma işlemlerine benzer.



Şekil 5.12 : Bir giriş/çıkış işlemi sırasındaki kontrol sinyalleri

DEĞERLENDİRME ÇALIŞMALARI

- 1) Bir assembly komut satırında olması gereken alanları yazınız.
- 2) Mikroişlemciler ilk çalışmaya başladığında (resetlendiğinde veya ilk enerji verildiğinde) izlediği adımlar nelerdir?
- 3) Farklı mikroişlemcilerin ilk çalışmada izledikleri yolları karşılaştırınız.
- 4) Mikroişlemcilerin resetlenme süreleri hangi kriterlere göre nasıl hesaplanır?
- 5) Mikroişlemciler neden saat sinyallerine ihtiyaç duyarlar?
- 6) Harici saat devreleri gerektiren mikroişlemciler hangileridir?
- 7) Komutların işlem basamakları nelerdir? Bu basamakları anlatınız.
- 8) Bir komutun işlem periyotlarını çiziniz.
- 9) Komutun bulunup getirilmesi ve zaman ilişkisini gösteren grafiği çizerek her saat sinyalinde değişen durumları inceleyiniz.
- 10) Komutun okunması ve zaman ilişkisini gösteren grafiği çizerek her saat sinyalinde değişen durumları inceleyiniz.
- 11) Hafızaya veri yazma ve zaman ilişkisini gösteren grafiği çizerek her saat sinyalinde değişen durumları inceleyiniz.
- 12) Giriş/çıkış işlemi sırasındaki okuma/yazma zaman ilişkisini gösteren grafiği çizerek her saat sinyalinde değişen durumları inceleyiniz.

ÜNİTE 6 MİKROİŞLEMCİLERİN DİŞ YAPISI

6.1 HAFİZALAR

Bir mikrobilgisayarın kurulabilmesi için sistemde bulunması gereken en temel bileşenler şunlardır: Mikroişlemci, hafıza elemanları ve giriş/çıkış elemanları. Tıpkı insan beyninde olduğu gibi mikroişlemcilerin de bazı bilgileri ilerde kullanabilmesi için saklamaları gerekir. Mikroişlemciler işleyişleri bakımından sistemdeki hafıza elemanlarından veya dış ortamlardan sürekli veri okuyup yazarlar. Bunun için de hafıza elemanlarına gerek duyarlar. Bundan dolayı mikrobilgisayarlarda da hafızalara gereksinim vardır. Hafıza elemanları kaydedilen bilgileri saklayan ortamlardır. Bir mikrobilgisayarda, hafızaların da en az mikroişlemciler kadar önemi vardır. Mikrobilgisayarlarda yer alan giriş/çıkış elemanları ise mikrobilgisayarın çevre birimleri ile iletişimini sağlar. Tasarımcı bir mikrobilgisayar tasarlarırken mikroişlemcinin yanı sıra hafızaların ve diğer çevre elemanlarında teknik özelliklerini, yapısını ve çalışmasını iyi bilmesi gerekir. Çünkü sistemin en iyi performansta çalışması, diğer çevresel elemanların uyumuna bağlıdır. Mikroişlemci üreten firmalar, genellikle ürettikleri mikroişlemci ile hem donanım hem de yazılım bakımından uyumlu diğer çevre elemanlarını da üretirler. Buna üretilen "mikroişlemcinin ailesi" denir. Seçilen mikroişlemci ailesine ait diğer çevresel elemanları araştırmak ve incelemek tasarımı kolaylaştırabilir.

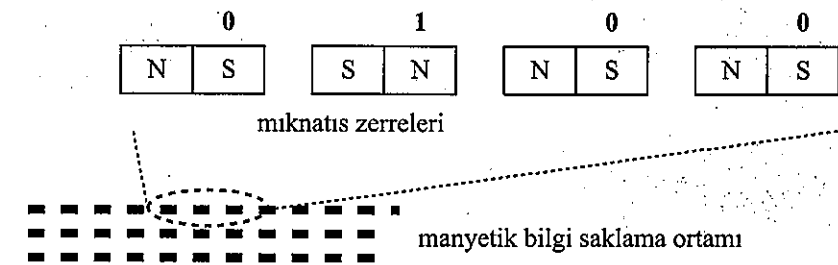
6.1.1 Manyetik Hafızalar

6.1.1.1 Manyetik hafızalar

Mikrobilgisayarlarda, verileri çok uzun süre saklamak için kullanılan bir hafıza çeşididir. Manyetik hafızalı ortamlarda bilgilerin saklanması, bu hafıza elemanlarının üzerindeki çok küçük mıknatis parçalarının kutup yönüne göre kodlanması ile gerçekleştirilir. Mikrobilgisayar sistemindeki bilgiler ikilik düzende işlem gördüğünden, bu bilgileri manyetik ortamda bulunan mıknatis parçalarının N ve S kutuplarına göre kodlamak hiç de zor değildir. Örneğin N kutbu lojik "1" i temsil ediyorsa S kutbu da lojik "0" ı temsil eder. Manyetik hafıza ortamlarına kaydedilen bilgiler, elektromanyetik bir ortama (elektrik transformatörü, monitör veya TV yanı, hoparlör mıknatısı...) girmediği sürece bozulmadan uzun yıllar saklı kalabilir. Mikrobilgisayarların manyetik hafıza elemanlarına bilgileri yazıp okuyabilmesi için özel aygıtlara ihtiyacı vardır. Bilgileri saklamak için en çok kullanılan manyetik hafıza elemanları disketler, teyp bandı ve hard disklerdir. Manyetik hafızaların avantajları ve dezavantajları şunlardır:

Tablo 6.1 : Manyetik hafızaların avantaj/dezavantaj tablosu

☺	Manyetik yüzey deforme olmadığı sürece, tekrar tekrar okuma/yazma olanakı vardır.
☺	Kapasiteleri manyetik yüzey ile doğru orantılıdır.
☹	Fiziksel ve manyetik etkilerden kolay etkilenir.
☹	Manyetik ortamlara bilgileri kaydetmek için özel aygıtlara ihtiyaç vardır.



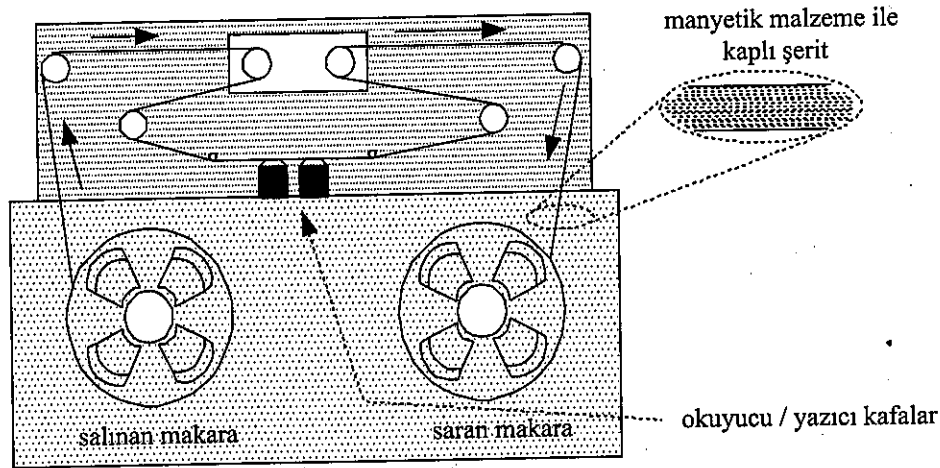
Şekil 6.1 : Manyetik malzemeli ortamlarda dijital bilgilerin "1" ve "0" şekline benzer kodlanması

6.1.1.2 Teyp bandı

Teyp bandı bir makara üzerine plâstik bir şeridin sarılmasından oluşmuştur. Bu şeridin üzerinde, toz hâlinde mıknatıs zerrelere sahiptir. İkili bilgiler, şeridin üzerindeki mıknatıs parçalarının kutup yönünü değiştirerek kodlanır. Teyp bandının kapasitesi manyetik şeridin uzunluğu ile sınırlıdır. Teyp bantları en fazla kapasitesi olan manyetik bilgi saklama ortamlarıdır. Daha çok arşivleme veya yedekleme işlemleri gibi kaydedilen bilgilere çok sık erişim gerektirmeyen durumlarda bilgilerin kalıcı olarak saklanması için kullanılır. Teyp bandının avantajları ve dezavantajları şunlardır:

Tablo 6.2 : Teyp bandı avantaj/dezavantaj tablosu

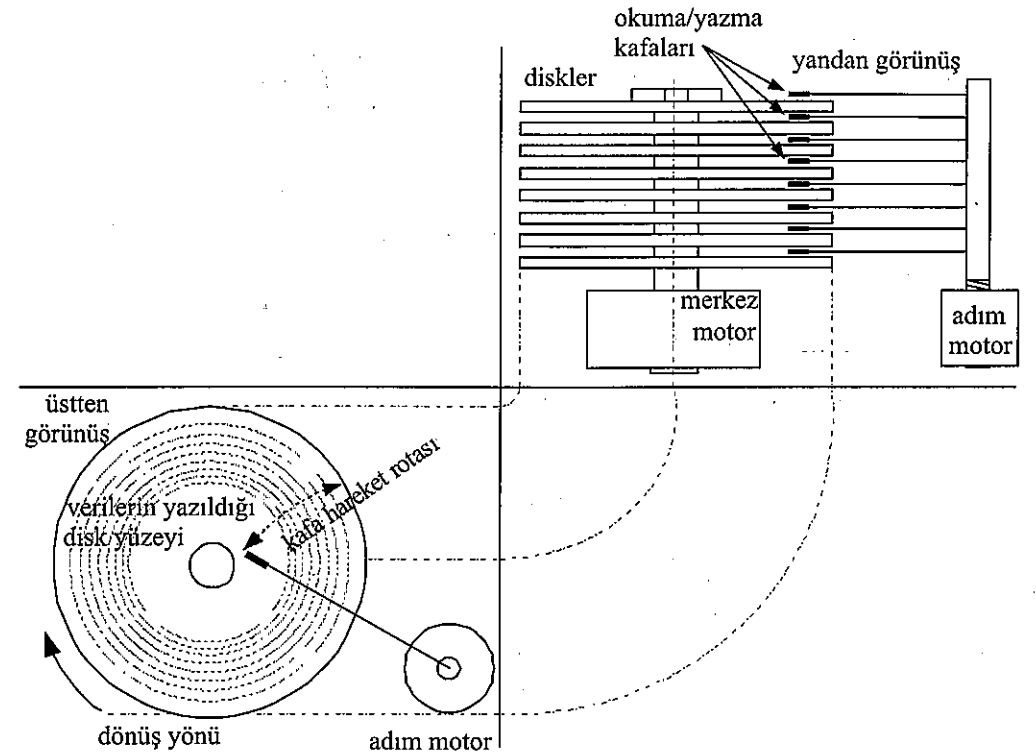
☺	Özellikle yedekleme sistemlerinde işletme maliyeti düşüktür.
☺	Tekrar tekrar kayıt yapabilme olanağı vardır (pratikte yine de bir sınırı vardır; örneğin 500 defa).
☺	Kapasiteleri büyüktür. Kapasite/maliyet oranından dolayı arşivlemelerde idealdir (120 Mb, 400 Mb...).
☺	Bilgiler taşınabilir rulo bant şeklindeki ortamlarda saklanır.
☹	Fiziksel ve manyetik etkilerden kolay etkilenir.
☹	Okuma/yazma ve rastgele bilgiye erişim hızları çok düşüktür. Bilgilere sıralı şekilde erişilir.



Şekil 6.2 : Teyp bandı ve okuma/yazma sistemi

6.1.1.3 Hard disk

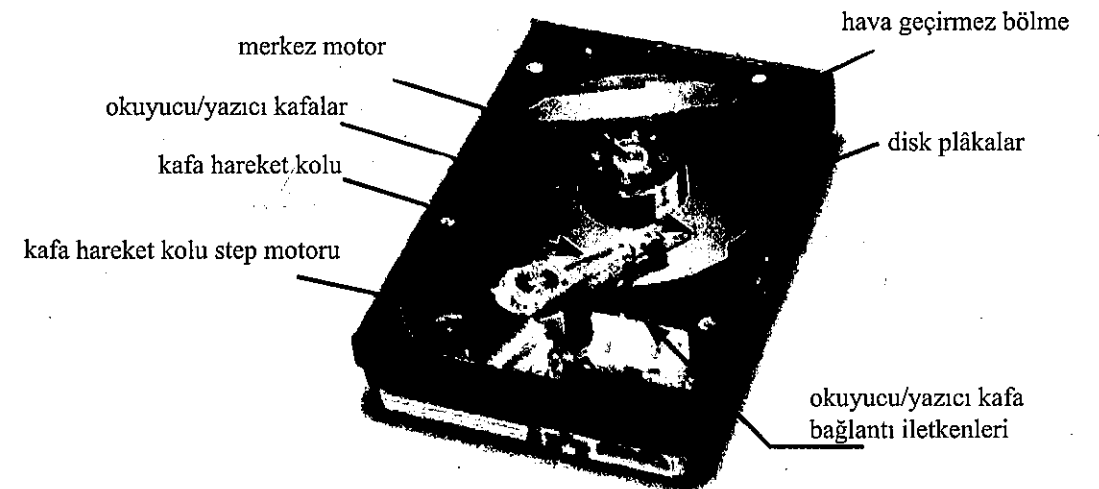
Hard diskler daha çok, gelişmiş mikrobilgisayarlarda (kişisel bilgisayarlar) ve büyük sistemlerde kullanılmaya uygun bilgi saklama ortamlarıdır. Kapasiteleri ve hızları çok yüksektir. Hard disklerin içerisinde üst üste birkaç tane manyetik yüzeyli diskler ve okuyucu/yazıcı kafalar bulunur. Manyetik yüzeyli bu diskler metal alaşımlı maddelerden yapılırlar. Bu disklerin yüzeyinde, her biri büyük bir mıknatıs gibi kuzey (N) ve güney (S) kutuplarına sahip çok küçük mıknatıs parçacıkları bulunur. Okuyucu/yazıcı kafalar bu mıknatısların yönlerini değiştirerek veya yönlerini tespit ederek okuma/yazma işlemini gerçekleştirir. Hard disklerin, gelişmiş bir işletim sistemine ve donanımına ihtiyaç duymasından dolayı, endüstriyel kontrol uygulamaları gibi küçük sistemlerde kullanımı ve kontrolü zorlaşmaktadır. Zaten endüstriyel kontrol sistemlerinde bu kadar büyük kapasiteli bilgi saklama ortamlarına genellikle ihtiyaç duyulmaz (şekil 6.3).



Şekil 6.3 : Hard diskte okuma/yazma kafaları ve manyetik diskler

Tablo 6.3 : Hard diskin avantaj/dezavantaj tablosu

☺	Bilgiler daha güvenilir ortamda saklanır (teyp bantlarına ve disketlere göre).
☺	Kapasitesi büyüktür (100 Mb, 1,2 Gb, 30 Gb.... Kıyaslama için yaklaşık değerler).
☺	Okuma/yazma hızları çok yüksektir.
☺	Tekrar tekrar kayıt yapabilme imkânı vardır.
☹	Çalıştığı sistemde sabittir (taşınması pratik değildir).
☹	İlk maliyeti pahalıdır. Disket veya teyp bandı gibi saklama ortamları kullanmadığından işletme maliyeti yoktur.
☹	Kullanılması için gelişmiş bir işletim sistemine ve donanıma gerek duyar.

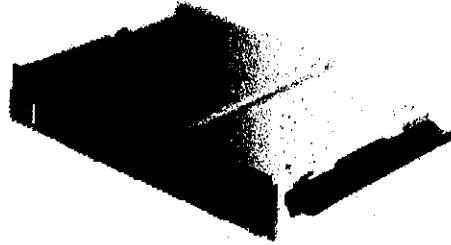


Şekil 6.4 : Hard diskin iç yapısını gösteren kesit

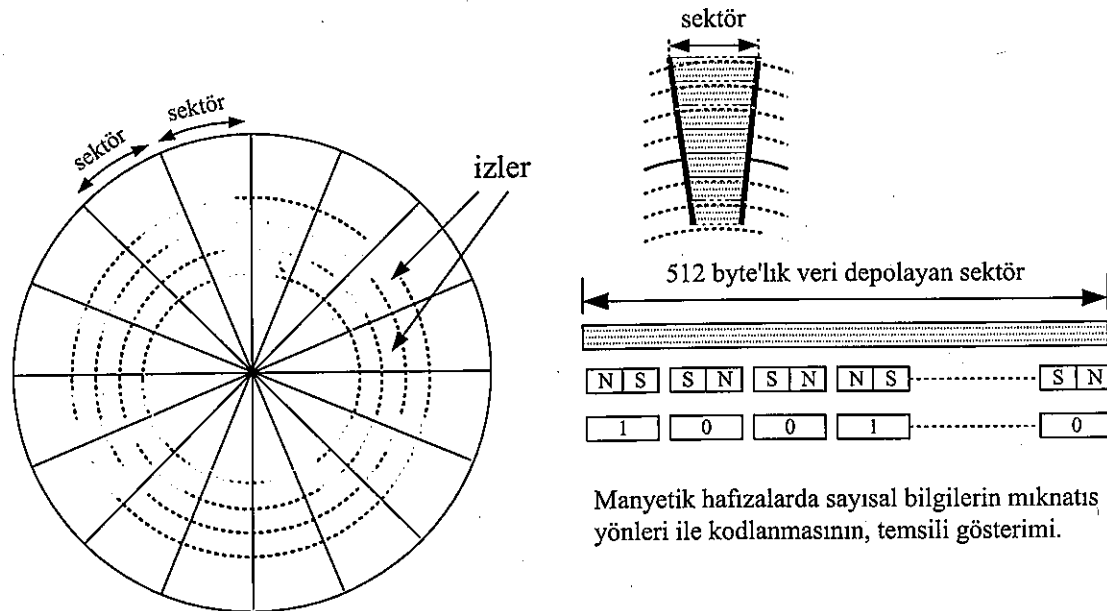
6.1.1.4 Floppy disk

Manyetik bir saklama ortamıdır. Floppy diskler, daire şeklindeki ince plâstik levha üzerine püskürtülmüş manyetik bir malzeme (demir oksit) ile kaplı olarak üretilir. Floppy ismi plâstik malzemeden ve esnek bir yapıda olduğundan verilmiştir. Floppy diskler dış etkilere (manyetik ve fiziksel darbelerden) çok çabuk etkilenir ve hasar görür. Manyetik diskleri okuyup yazmak için kullanılan özel aygıtlar vardır, bunlara disket sürücü adı verilir. Disket sürücüleri, floppy diskin yüzeyindeki bilgileri, okuyucu/yazıcı kafasında bulunan elektromıknatıs aracılığı ile yazar veya okur. Disketler, en çok PC tipi bilgisayarlarda kullanılır. Taşınabilir ve maliyetinin çok düşük olmasından dolayı disketler hâlâ çok kullanılan birer saklama ortamlarıdır.

Disketler üzerindeki bilgileri belli bir format yapısında tutarlar. Bu format yapısına göre iç içe dairesel halkalar yani izler (track) ve onun parçaları olan sektörler bulunur. Bilgiler izler ve sektörlerin kesiştiği alanlara yazılır. Bu alanların her biri yaklaşık 512 byte'lık veri depolamaktadır. Disketin kapasitesi sektör ve iz sayısına göre değişir. 1.44 Mb. kapasitesindeki bir diskette 80 adet iz, bunları dik kesen 18 adet sektör bulunur ve çift yüzlü olarak kullanılmıştır. Örneğin 720 Kb. kapasitedeki bir disket; 80 iz, 9 sektör ve çift yüzlü olarak hazırlandığında her sektörde 512 byte'lık bilgi depoladığına göre kapasitesi; $80 \times 9 \times 512 = 737280$ byte eder. 737280 byte yaklaşık olarak 720 Kb'dır.



Resim 6.1 : Disket sürücüsü ve disket



Şekil 6.5 : Manyetik diskin bölümlerini gösteren sembolik şema

Tablo 6.4 : Floppy disk'in avantaj/dezavantaj tablosu

😊	İşletme ve ilk maliyeti çok ucuzdur.
😊	Tekrar tekrar kayıt yapabilme olanağı (Yine de bir sınırı vardır, örneğin 500 defa).
😊	Bilgiler taşınabilir disketlerde saklanır. Bu diskler için "disket sürücüler" kullanılır.
😊	Rastgele bilgiye erişim hızı teyp bantlarına göre yüksektir.
😞	Kapasitesi çok düşüktür (360 Kb, 1,44 Mb...).
😞	Fiziksel ve manyetik etkilere kolay etkilenir.
😞	Okuma/yazma hızları düşüktür.

6.1.2 Statik Hafızalar

Statik hafızalar, yarı iletken malzemeler kullanılarak yapılan hafıza elemanlarıdır. Bilgiler yarı iletken ortamlarda elektriksel olarak saklanırlar. Bu hafıza elemanları entegre şeklinde imal edilmiştir. Yapı ve kullanım bakımından birçok çeşitleri vardır. Bunlar ROM ve RAM diye iki ana gruba ayrılır. ROM; yalnız okunabilir hafıza (Read Only Memory) diye bilinen ve besleme enerjisi kesilse de sakladığı bilgileri tutan hafıza elemanlarıdır. RAM'lar ise üzerindeki bilgileri besleme enerjisi kesildiğinde tutamayan hafıza elemanlarıdır.

6.1.2.1 ROM'lar (Read Only Memory)

ROM, yalnız okunabilir hafızaların genel adıdır. Bu tip hafızalara, bilgiler özel cihazlarla önceden yazılır. Bu hafıza elemanlarının en büyük özelliği kendisini besleyen enerji kesilse dahi üzerindeki bilgileri saklı tutabilmesidir. Bu elemanlar sürekli kalması istenen bilgilerin saklanması için kullanılır. Örneğin özel kodların tablo olarak saklanması (ASCII kodu gibi) veya karakter genaratörlerinde karakterlerin bit düzeni için kullanılır. Ayrıca kişisel bilgisayarların ilk açılışında sistemi test eden ve işletim sistemine hazır hâle getiren BIOS programları da bu tür sabit hafızalara yazılır. Entegre şeklinde olan bu hafıza elemanlarının kapasiteleri genel olarak 128 byte ile 512 Kb arasında değişir. Sabit hafızaların ROM, EPROM, PROM, EEPROM gibi çeşitleri vardır.

- ❖ **ROM'lara** daha üretim aşamasında iken içerisinde program veya veri yazılır. ROM'ların saklı tuttuğu bilgiler sonradan hiçbir şekilde değiştirilemezler. Bu hafızaların üretimi öncesi birtakım hazırlık aşamaları vardır. Bilgilerin kaydedilmesi imalat aşamasında gerçekleştiğinden uzun bir çalışma gerektirir. Tıpkı matbaalarda olduğu gibi çok sayıda basımlar için yapılan dizgi çalışmalarına benzer. Hazırlık çalışmaları zor ve tasarım maliyetleri yüksektir. Fakat çok sayıda üretimlerde birim başına düşen ortalama maliyet azaldığı için aynı işi yapan diğer ROM'lardan çok daha ucuza gelmektedir. Kısaca az sayıda kullanım için ekonomik ve pratik değildir.
- ❖ **PROM'lar** ise üretildikten sonra kullanıcıya bir tek kez bilgi kaydetme imkânı vermektedir. Bu hafıza elemanı yine entegre şeklindedir. Kaydedilen bilgiler enerji kesilse dahi silinmezler. Üzerine program kodlarını veya verileri yazmak için "PROM programlayıcı" cihazlara gereksinim vardır. Bu hafıza elemanın yapısında küçük sigorta telleri bulunur. Hafıza hücrelerinde, normalde hepsi sağlam durumda bulunan sigortalar lojik olarak "1" i temsil eder. Yazılacak olan bilginin bit düzeninde "0"lara karşılık gelen hücredeki sigorta, küçük bir elektrik akımı ile atılır. Böylece programlanmış olur. Fiyatı EPROM'lara göre daha uygundur fakat bir kez kullanılabilmesi bir dezavantajdır. PROM'lar araştırma geliştirme çalışmalarına uygun değildir fakat hazırlık çalışması bitmiş ve az sayıda üretilecek sistemler için ROM'lara ve EPROM'lara göre ekonomiktir.
- ❖ **EPROM'lara** bilgi, elektriksel olarak yazılır ve mor ötesi ışıkla (ültraviyole) silinir. Yani defalarca yazılıp silinebilen ROM'lardır denebilir. Üzerindeki bilgiler elektrikler kesilse dahi silinmezler. EPROM'daki bilgiler, üzerinde bulunan küçük bir pencereden mor ötesi ışık altında bir süre bekletilerek silinir ve tüm hücreler Lojik olarak 1 konumuna getirilir. Yani boş bir EPROM'un hücreleri lojik "1" ile doludur. EPROM üzerine bilginin yazılması için programlayıcıya, silinmesi için silme cihazlarına gereksinim vardır. EPROM'lara bilgiler elektriksel olarak yazılır. Yazılacak olan değer veri yoluna bırakıldıktan sonra kısa bir süre 25 volt gibi bir gerilim tatbik edilerek veri yazma işlemi gerçekleşir. ROM ve PROM'lardan pahalı olmasına ve üzerine bilgi yazmak ve silmek için özel aygıtlar gerektirmesine rağmen, defalarca silinip yazılabilmesinden dolayı özellikle araştırma geliştirme aşamalarında çok kullanışlıdır. Seri üretim için fiyatı uygun değildir.



Epromlar Nasıl Programlanır?

EPROM'un programlanması için kullanılan "EPROM programlama cihazları" piyasada mevcuttur. Bu cihazlar ya bilgisayar bağlantılıdır ya da deney seti şeklinde olabilir. EPROM'a yazılacak veya okunacak veriler, bilgisayar bağlantılı cihazlarda, bilgisayar ekranından girilir ve veriler "yükle" komutuyla EPROM'a yüklenmeye başlar. Bilgisayarlı cihazlarda veriler istenirse bilgisayara dosya olarak kaydedilebilir. Bazı cihazlarda ise veriler, hexadesimal bir klavye ve gösterge paneli yardımıyla okunur veya yazılır. EPROM'u programlayabilmek için dijital kapılarla yapılmış basit devreler de vardır. EPROM'ların silinmesi için kullanılan cihazlarda ise ultraviyole ışık kullanılmaktadır. EPROM, ultraviyole ışık altında yaklaşık 10-15 dk. bekletilir. Bu zaman sonunda EPROM silinmiş olur.

- ❖ **EEPROM** üzerine bilgiyi, elektriksel olarak yazabildiğimiz ve elektriksel olarak silbildiğimiz hafıza elemanlarıdır. Üzerine bilginin elektriksel olarak yazılıp silinmesine karşın EEPROM'u besleyen enerji kesilse dahi üzerindeki bilgiler kaybolmaz. EPROM'lardan farkı, silme işlemi için mor ötesi ışık kullanılmaz. Böylece daha kullanışlı EPROM'lar elde edilmiştir. EEPROM'ların en büyük avantajı, silinmesi ve yazılması için özel silme ve yazma cihazlarına ihtiyaç duymamasıdır. Gerekliğinde sistem üzerinde iken (sökülüp takılmadan) üzerinde istenen değişiklikler yapılabilir. Fiyatının giderek ucuzlaması ile EPROM'ların yerini alması muhtemeldir.

6.1.2.2 RAM'lar (Random Access Memory)

RAM hafızanın herhangi bir yerindeki veriye erişmek için sıralı bir okuma/yazma gerekmez. Bu elemanlara "rastgele erişilebilir hafıza" denmesi, verinin bulunduğu herhangi bir hafıza hücresine istenildiği zamanda direkt olarak erişilebilmesine imkân sağlamasındandır. ROM tipi hafıza elemanlarından en büyük farkı bu hafıza elemanları saklı tuttuğu bilgilerin enerjisi kesilince silinmesidir. Mikroişlemciler bu hafızalara çalışma anında sürekli bilgi yazıp okuyabilir. Tüm bilgisayarlarda RAM'lar kullanılmaktadır. Çünkü sistem çalışırken üzerinde işlem yaptığı bilgileri daha sonrada kullanabilmesi için bir yerlere yazması gerekmektedir. Bu bilgiler çoğu zaman sistem çalıştığı sürece ihtiyaç duyulan bilgilerdir. Bunun için en iyi çözüm RAM'lardır. RAM'lar yapı bakımından ikiye ayrılırlar; "statik RAM" ve "dinamik RAM". Piyasada EDO, SDRAM, DDRAM gibi değişik teknolojik yapıda üretilen RAM hafızalar kullanılır.

Statik RAM'larda bilgiyi depo edebilmek için flip-flop devreleri kullanılır. Flip-flop'lar, statik RAM'ların bilgi depo eden en küçük yapı taşıdır. Bilindiği gibi flip-flop'lar girişinde bulunan "1" veya "0" bilgisini bir tetikleme sinyali ile birlikte çıkışına yüklerler. Yeni bir tetikleme sinyali gelinceye kadar da çıkışındaki bilgi saklı tutulur. Statik RAM'ların kapasitesi ve hızı dinamik RAM'lara göre daha düşüktür ayrıca güç sarfiyatı yüksektir. Fakat kullanımı dinamik RAM'lardan daha basittir.

Dinamik RAM'larda bilgiler elektriksel yük olarak depo edilir. Saklanan bilgiler sürekli güncelleştirilmezse, elektriksel yük boşalacaktır. Bunu önlemek için hafızada bulunan tüm bilgiler sürekli kontrol edilerek, eğer hücrede "1" varsa tekrar "1"e yükleme yapılır. "0" varsa bir sonraki hücrenin kontrolüne geçilir. Tüm bu işleri yapan yardımcı elektronik devre elemanları vardır. Bazı mikroişlemcilerde (Z80) dinamik hafıza elemanlarının tazelenmesi için kontrol çıkışları bulunmaktadır.

Bu tip hafıza elemanlarına dinamik denmesi sürekli olarak tazeleme gerektirmesinden dolayıdır. Statik RAM'larda böyle bir durum yoktur.

6.1.3 Dinamik Hafızalar

Statik hafızalar bilgiyi flip-floplarda elektrik potansiyeli olarak saklarken dinamik hafızalar bilgiyi elektrik yükü olarak saklamaktadır, bu da dinamik hafızaların güç sarfiyatını önemli ölçüde düşürmektedir. Dinamik hafızaların içerisinde bilgiyi saklayabilmek için her bir bite karşılık bir kapasite ve MOS transistörle kuvvetlendirici vardır. Dinamik hafıza hücrelerindeki bu kapasite elemanlarına lojik "1" bilgisi için elektrik yükü doldurulur, lojik "0" için bu yük boşaltılır. Bilindiği gibi kapasite elemanları üzerindeki yükü belli bir süre saklarlar fakat sızıntı akımlarından dolayı bu yük zamanla sıfırlanır. Eğer

periyodik olarak bu hücrelerdeki lojik seviyeler okunup ve eski değerine yeniden yüklenirse bu hücrelere yazılan bilgi hafıza elemanının enerjisi kesilinceye kadar saklı kalır. Bu işleme tazeleme işlemi adı verilir. Kapasite belli bir seviyeye kadar boşalmadan içerisindeki bilginin birkaç milisaniyede okunup yeniden yazılması gerekmektedir. Geciken tazeleme kaydedilen bilgilerin bozulmasına sebep olur. Dinamik RAM'ların dezavantajı burada başlamaktadır. Dinamik RAM'ları kullanmak istediğimizde tazeleme işini yapmak için ek devre elemanlarına ve mikroişlemcilerin "refresh" kontrol çıkışlarına gereksinim duyarız. Bu da toplam maliyeti artırır ve tasarımı zorlaştırır. Dinamik hafızaların avantajları ise aynı kapasitedeki statik RAM'lara göre daha ucuz olması, bilgi depolama kapasitesinin yüksek olması ve düşük güç sarfiyatıdır. Özellikle yüksek kapasite (512 Kb ve yukarısı için) gerektiren sistemlerde daha uygundur. PC tipi bilgisayarlarda bilinen RAM hafızaları, dinamik RAM'lar oluşturur.

6.1.4 CD Hafızalar

Lâzer teknolojisi birçok alanda kullanılmasıyla birlikte, bilgisayar endüstrisinde de yerini almıştır. Bilgisayarlarda, şimdilik bilgi saklama ortamlarında lazer teknolojisinden faydalanılmıştır. CD (compact disk) plâstik türevi bir maddeden disk şeklinde üretilmektedir. Bu diskin üzerine ancak lazer gibi hassas ve keskin bir ışığın kullanılabileceği şekilde çukurluklar veya tümsekler basılmıştır. CD üzerinde "lojik 1" bilgisine karşılık tümsek, "lojik 0" bilgisine karşılık çukur bulunmaktadır. CD sürücü aygıtı, CD üzerindeki bu tümseklerin ve çukurlukların varlığını CD hızla dönerken üzerine gönderdiği lazer ışığının yansıması veya yansımaması durumuna göre anlar. Yansıyan ışık "lojik 1" yansımayan ışık "lojik 0"ı temsil eder. CD'ler üzerinde yaklaşık olarak 600 Mb. Bilgi tutabilirler. CD'lerin üzerine bilgiler üretim aşamasında kaydedilir ve sonradan değiştirilemezler. CD'lerin taşınabilir ve ekonomik olması çok hızlı bir şekilde popüler olmasını sağlamıştır. Daha sonraları, verinin üzerine yalnız bir kez yazılabildiği CD'ler yani CD-R (CD Recordable) ve verinin defalarca yazılabildiği CD'ler yani CD-RW (CD-ReWrite; Yeniden yazılabilir CD) piyasaya çıkmıştır. CD'ler günümüzde müzik (track, mp3, wav), video (VCD, SVCD, DVD) ve veri CD'leri olarak birçok değişik verileri saklamada kullanılmaktadır.

☺	Bazı türlerinde tekrar kayıt yapabilmek olanağı vardır.
☺	Yüksek kapasiteli ve düşük maliyetlidir (ortalama..650 Mb'tır. Boş bir CD 0,2-0,5 \$'dır.).
☺	Bilgiler taşınabilir disklerde saklanır.
☺	Manyetik alanlardan ve fiziksel ortamlardan etkilenmezler (rutubet, soğuk...).
☹	Yazılabilir türde olanların veri yazma hızları hard diske göre düşüktür.
☹	Bilgiye erişim hızı hard diske kıyasla düşüktür.
☹	CD'leri kullanmak için CD sürücülere ihtiyaç vardır.
☹	CD sürücüler fiziksel olarak hassastır. Özellikle toz ve çizilme bu sürücülerin CD'leri okumamasına sebep olur.



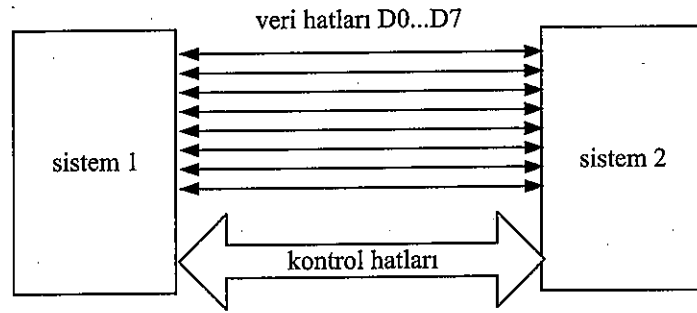
Yazılabilir CD'ler üzerine ısıya dayanıklı bir boya sürülür. Sürülen bu boya CD yazıcının lazeri tarafından işlenir. Lazer, çukuru boya maddesini yakarak açar. Boyanın yakılan yerleri çukurlukları, yakılmayan yerleri tümsekleri oluşturur. Bu çukurluklar ve tümseklere "pits" denir. İşlenen bir veri okunmak istendiğinde lazerin gücü, boya maddesini yakmayacak şekilde düşürülür. Ayrıca boya maddesinin üzerine altın buharı püskürtülür. Bu işlem diskin üzerinde yansıma tabakasını oluşturur.

6.2 GİRİŞ/ÇIKIŞ (Input/Output) ÜNİTELERİ

6.2.1 Paralel Giriş/Çıkış Üniteleri

Mikroişlemciler, hafıza ve diğer çevre birimleri arasında veri akışını sağlamak için paralel iletişim yöntemini kullanır. Paralel iletişim ayrıca bir mikroişlemcinin kendi içerisinde kaydedicileri ve diğer iç üniteleri arasında kullandığı bir iletişim biçimidir. Paralel formdaki verilerin bit genişliği sistemin yapısına göre 8 bit, 16 bit veya daha fazla olabilir. Mikrobilgisayarlar çevre birimleriyle iletişim sağlamak için de paralel giriş/çıkış çevre birimlerine ihtiyaç duyar. Paralel giriş/çıkış üniteleri programlanarak iletişimin yönü ve iletişim şekli belirlenir. Paralel iletişimde veri yolundaki her bit için ayrı bir hat kullanılır. 8 bitlik mikrobilgisayarda paralel iletişim için en az 8 ayrı hat ve diğer kontrol hatları gereklidir. Kontrol hatları, iletişim içinde olduğu cihazla “el sıkışma” diye tanımlanan kontrol sinyallerini içerir. Mikrobilgisayar gönderdiği veriyi alıcı tarafa bu kontrol sinyallerini kullanarak bildirir. Yine dışarıdan bilgi girişi yapılmak istendiğinde ise mikrobilgisayar bu kontrol sinyalleri ile uyarılır.

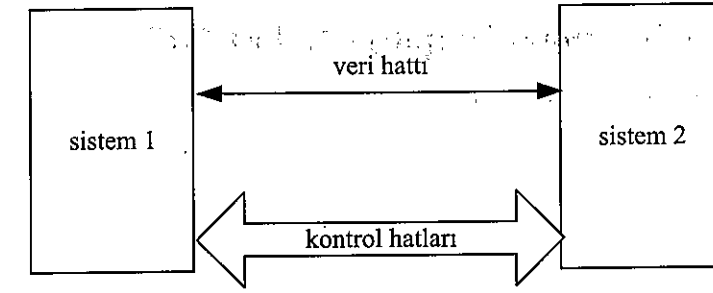
Bilgisayarlar paralel iletişim tekniğini; A/D çeviricilerle, disket sürücülerle, hard disk ve CD sürücülerle, ses ve görüntü kartlarıyla, sistem dışında ise; yazıcı, tarayıcı ve benzeri diğer cihazlarla iletişim kurmada kullanır. Paralel iletişimde, verilerin iki yönlü kullanılabilmesinin yanında kısa mesafeli uygulamalarda çok kolay ve hızlı bir iletişim sağladıklarından tercih sebebidir. Fakat uzun mesafelerde (10 metreden fazla) hat sayısının fazlalığı ve sinyal zayıflamasından dolayı tercih edilmezler. Böyle durumlarda seri iletişim şekli düşünülmelidir.



Şekil 6.6 : Paralel iletişim blok diyagramı

6.2.2 Seri Giriş/Çıkış Üniteleri

Mikrobilgisayarların harici çevre birimleri (yazıcı, harici modem, tarayıcı, vb.) ile iletişim kurmak için kullandığı diğer bir yöntem ise seri iletişimdir. Seri iletişimde veriler bir hat üzerinde art arda gönderilir. Mikroişlemciler, yapı olarak bilgileri paralel şekilde kullandığından, verileri seri forma dönüştürmek için programlanabilir seri iletişim üniteleri (entegreleri) kullanır. Bu üniteler sistemden aldığı paralel bilgileri belirli aralıklarla art arda bitler hâlinde gönderirler. Paralel yöntemle göre kullanımı daha karmaşık ve hata oranı daha yüksektir, fakat uzun mesafeli iletişimlerde en ekonomik yoldur. Seri iletişimde karşılıklı iki tarafında iletişim kurabilmesi için iletişime başlanmadan önce anlaşıldığı bir düzen olmalıdır. Bu düzene “iletişim protokolü” denir. Bu protokolde, iletişimde kullanılan bilginin boyu, iletişim hızı, kontrol bitlerinin “tek eşlik” veya “çift eşlik” mi olacağı, başlama ve durma bitlerinin ne olacağı gibi kurallar belirlenir. Veriler protokolde belirlenen şartlara uygun olarak paketler hâlinde gönderilir. Mikrobilgisayarlarda seri iletişimi gerçekleştiren özel amaçlı üretilmiş entegreler vardır. Bu entegreler mikrobilgisayarlardan aldığı seri bilgiyi belirlenen protokole uygun olarak paketleyip iletirler. Mikroişlemciler bu veri paketlerinin hazırlanması için ayrıca meşgul olmak zorunda değildir.



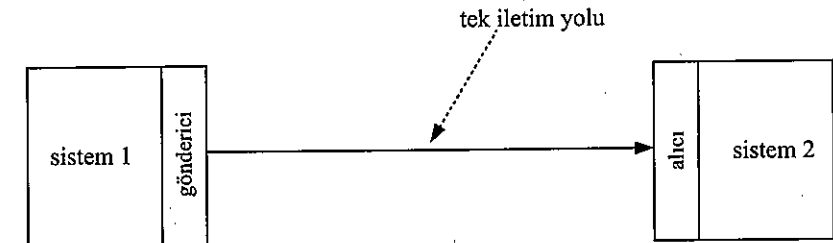
Şekil 6.7 : Seri İletişim blok diyagramı

Seri iletişimde veriler üç şekilde iletilebilmektedir.

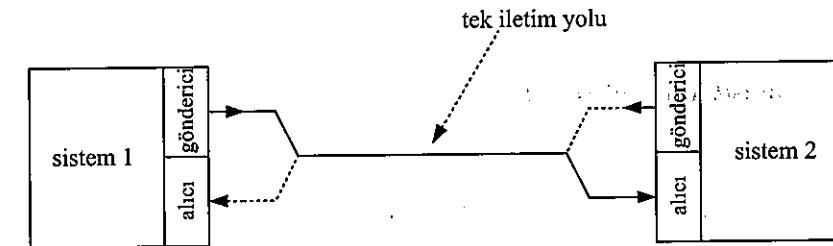
1-Tek yönlü (Simplex) veri iletimi: Bu tür iletişimde veriler tek hat üzerinden ve tek yönlü olarak iletilebilir. Buna televizyon iyi bir örnektir. Antenden gelen yayını televizyonda izlerken aynı yoldan bir bilgi göndermemiz mümkün değildir.

2-Yarı dubleks (Half-duplex) veri iletişimi: Bu tür veri iletişimde her iki tarafta birbirlerine veri gönderebilir yada alabilir. Buna telsiz konuşmaları iyi bir örnektir. Telsizlerde bir taraf konuşacağı zaman diğer taraf dinlemek zorundadır.

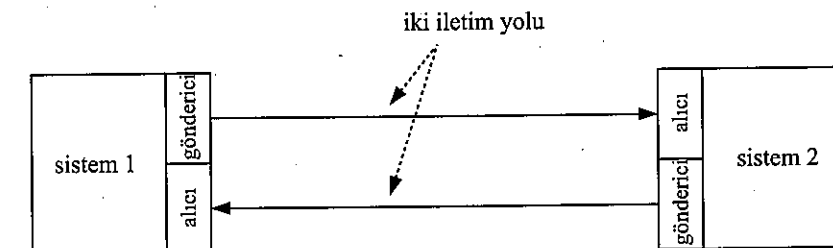
3-Tam dubleks (Full-Duplex) veri iletişimi: Tam dublekste karşılıklı iki tarafta aynı anda hem veri gönderebilir hem de veri alabilir. Tam dubleks iletişimde veri iletim hızı diğerlerine oranla daha yüksektir. Tam dubleks iletişim tekniğini karşılıklı telefon konuşmasına benzetebiliriz.



a) tek yönlü (simplex) iletişim tekniği



b) yarı-dubleks (half-duplex) iletişim tekniği

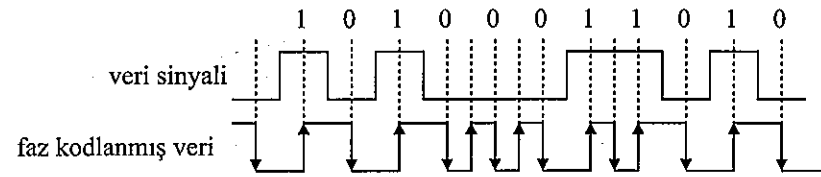


c) tam dubleks (full-duplex) iletişim tekniği

Şekil 6.8 : Seri bilgi iletim teknikleri

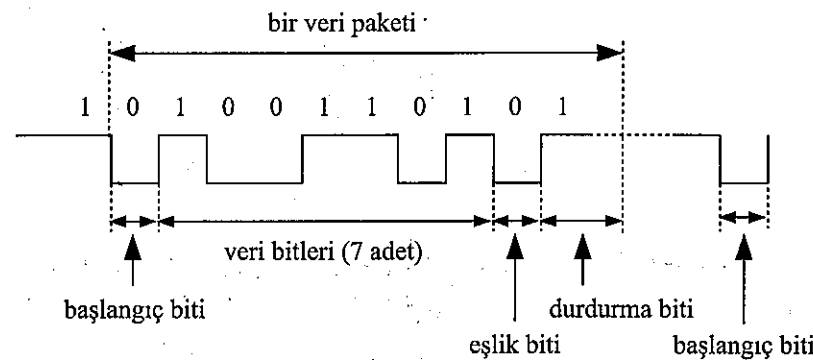
İletişim simplex, yarı dubleks veya tam dubleks hangi teknikle gerçekleştirilirse gerçekleştirilsin gönderilen ve alınan verilerin, taraflarda tekrar anlamlı verilere nasıl dönüştürüleceğini belirleyen iki yöntem vardır. Seri iletişim tekniğinde kullanılan bu yöntemler “Senkron” ve “Asenkron” iletişim yöntemleridir.

Senkron iletişim yöntemindeki temel sorun, gelen veri akışındaki bitlerin nasıl ayırt edileceği ve bu bitlerden anlamlı veri gruplarının nasıl elde edileceğidir. Örneğin taraflar art arda gönderilen iki tane “1” verisini nasıl ayıracak? Eğer karşılıklı sistemlerde eş zamanlı saat kullanılsaydı böyle bir sorun kalmayacaktı. Fakat aynı saat sinyali eş zamanlı olarak kullanmak için veri yoluna ek olarak bir de saat sinyali hattı kullanmak gerektirdiği için pek kullanışlı değildir. Bunun için gerekli senkronizasyon sinyali elde etmek ancak gönderilen ve alınan veri sinyalinin içinde olacak şekilde kodlanmakla mümkün olmuştur. Verinin “faz kodlanarak” nasıl gönderildiği şekil 6.9.’de gösterilmiştir.



Şekil 6.9 : Faz kodlanmış senkron veri sinyali

Asenkron seri iletişim tekniğinde gönderilecek ve alınacak taraflarda verinin; “iletim hızı”, “başlangıç biti”, “veri uzunluğu”, “eşlik biti (kontrol biti)” ve “durdurma bitinin” ne şekilde olacağını belirlemek çok önemlidir. Veri aktarılmadan önce hat sürekli olarak “1” konumundadır. Veri göndermek isteyen taraf bir bit periyodu süre kadar hattı “0” a çeker. Böylece karşı taraf veriyi alma durumuna geçer. Ardından, daha önceden belirlendiği gibi örneğin 7 periyot (8 bit de olabilir) boyunca gelen bitleri “veri” olarak kaydedicide saklar. Gönderici taraf, “veri bitlerini” göndermeyi tamamladıktan hemen sonra alıcı tarafın aldığı verilerin doğru gelip gelmediğini kontrol etmesi için eşlik bitini gönderir. Alıcı taraf eşlik bitine göre, gönderilen veriyi kontrol eder. Alıcı, alınan verinin eşlik bitini hesaplar ve gönderilen eşlik biti ile karşılaştırır. Eğer bir hata ile karşılaşır hata bayrağını kaldırır ve iletişimi keser. Eşlik bitinin ardından durdurma biti (lojik “1”) gönderilir. Durdurma biti bazen iki periyot süresince de olabilir. Alıcı taraf durdurma bitinden sonra yeni gelecek veri için beklemeye başlar.



Şekil 6.10 : Asenkron seri iletişimin veri yapısı

Başlama biti + Veri biti (7 veya 8) + Eşlik biti + Durdurma biti (1 veya 2) = 10 veya 12

Görüldüğü gibi en iyi şartlarda bile asıl verinin 7 bit olmasına karşılık kontrol bitleri ile birlikte bu sayı 10'a çıkabilmektedir. Yani iletişim süresince akan bitlerin %70'i veri, %30'unu kontrol bitleri teşkil eder. Veri bitinin 8, durdurma bitinin 2 olduğu durumlarda ise bu oran %66 veri, %34 kontrol bitleri şeklinde olur. Senkron seri iletişimde bu tip kayıplar yoktur.



Seri iletişimlerde iletişim hızları iki şekilde ifade edilmektedir.

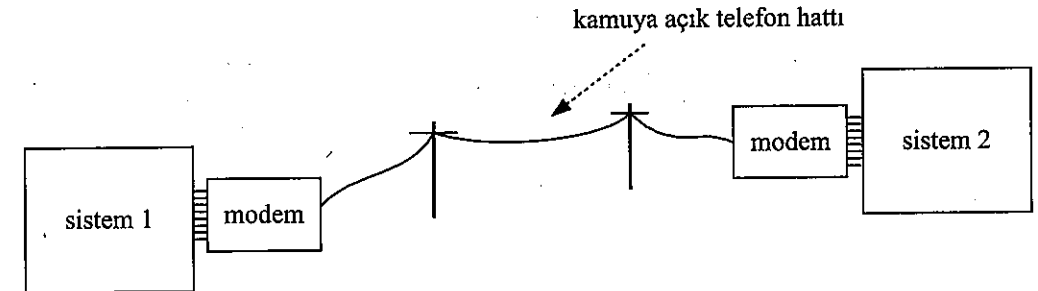
1- **Bps (Bit Per Second):** Bir saniyede iletilen bit sayısı olarak ifade edilir.

2- **Baud-Rate:** Bir bitin aktarımı için geçen zamandır. Bir baud saniyede 1 bit demektir. Bilgi hızı da diyebileceğimiz bu oran standart olarak 110, 300, 1200, 2400, 4800, 9600, 19600 şeklinde belirlenmiştir.

$$2400 \text{ baud'da bir bitin geçiş zamanı; Baud-Rate} = \frac{1}{1 \text{ bitin geçiş zamanı (sn)}}$$

$$1 \text{ bitin geçiş zamanı (sn)} = \frac{1}{\text{Baud-Rate}} = \frac{1}{2400} = 0,000416 \text{ sn} \quad \text{olacaktır.}$$

Seri iletişimde mesafe çok uzunsa iki taraf arasında veri iletişimini sağlamak için standart telefon hatları kullanılabilir. Bu durumda, sistemler arasında kullanılan dijital sinyalleri, telefon hattında iletme uygun bir şekle dönüştüren özel cihazlar kullanılır. Bu cihazlara “modem” (modülatör-demodülatör) denir. Böylece karşılıklı iki sistem arasındaki veri iletişimini telefon hattı üzerinden gerçekleştirebiliriz.



Şekil 6.11 : İki sistem arasındaki verilerin telefon hattı üzerinden iletilmesi

6.3 MİKROBİLGİSAYAR SİSTEMİNDE KULLANILAN ENTEGRELERİN KATALOG BİLGİLERİNİN İNCELENMESİ

Mikrobilgisayar sistemlerinde kullanılması gerekli en az üç eleman bulunur. Bunlar mikroişlemci, giriş/çıkış ve hafıza elemanlarıdır. Bir mikrobilgisayar piyasada bulunan değişik mikroişlemcilerle yapılabilir. Tablo 6.5’de dünyada ve ülkemizde çok bilinen, okullarımızdaki eğitim setlerinde kullanılan mikroişlemciler ve diğer elemanlar verilmiştir. Bu elemanların teknik özellikleri, bacak bağlantıları ve görevleri ayrı ayrı verilmiştir.

Tablo 6.5 : Katalog bilgisi verilen elemanlar

Ürünün cinsi	Üretim kodu
Mikroişlemci	Motorola MC6802
Mikroişlemci	Zilog Z80
Mikroişlemci	Mos Tec. R6502
Paralel giriş/çıkış arabirimi	Motorola 6821
Paralel giriş/çıkış arabirimi	Intel 8255
Seri giriş/çıkış arabirimi	Intel 8251A
Seri giriş/çıkış arabirimi	Motorola 6851
EPROM	Intel 2764 (8 Kb)
RAM	6264 (8 Kb)

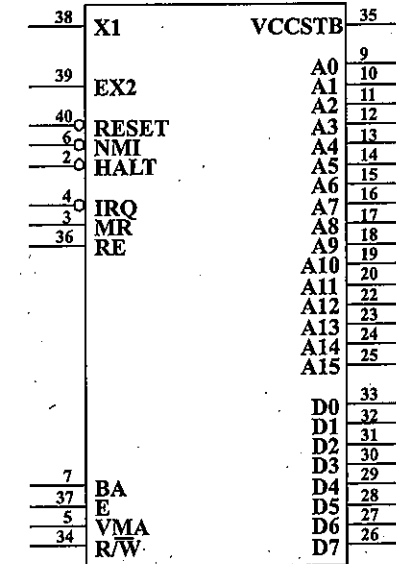
6.3.1 Motorola MC6802 Mikroişlemcisinin Özellikleri

- ❖ Motorola ailesinin 8 bitlik işlemcisidir.
- ❖ 5 voltluk besleme gerilimine ihtiyaç duyar.
- ❖ 16 bitlik adres yolu çıkışı 64 Kb'lık hafıza bölgesini adresleyebilir.
- ❖ Komut setinde 72 adet temel komut bulunur.
- ❖ Değişebilir uzunlukta yığın (stack) kullanabilir.
- ❖ 1 MHz dâhilî işlem hızı.
- ❖ 7 çeşit adresleme tekniğini kullanır (Doğrudan adresleme, dolaysız adresleme, genişletilmiş adresleme, kaydedici adresleme, bağıl adresleme, indeksli adresleme, mutlak adresleme).
- ❖ Altı adet iç kaydedicisi (Genel amaçlı kaydediciler "A ve B", indeks kaydedicisi "IX", program sayacı "PC", yığın işaretleyicisi "SP", durum kodu kaydedicisi "CCR") vardır.
- ❖ Vektörlenmiş yeniden başlatma.
- ❖ Maskelenebilir kesme vektörü (NMI).
- ❖ Doğrudan hafıza adresleyebilme ve çoklu işlem yeteneği.
- ❖ TTL'siz basit yol arabirimleri kullanabilme.
- ❖ Durdurma ve tek komut yürütme yeteneği (Step by step, adım adım).
- ❖ Adres yolu (A0..A15); standart TTL yük sürebilme yeteneğine sahip üç durumlu yol. Üç durumlu olması DMA uygulamalarına uygundur.
- ❖ 128 byte RAM hafıza içerir.
- ❖ Veri ve adres yolları birbirinden bağımsız.
- ❖ Kesme girişleri (IRQ, NMI) ve bekletme (HALT) girişleri bulunmaktadır.



Resim 6.2 : Motorola MC6802 mikroişlemcisi fotoğrafı

MC6802 BACAK BAĞLANTILARI	
X1, EX2	4 MHz kristal osc. bağlantı girişi.
RESET	Lojik "0" da etkinleşen resetleme girişi.
NMI	Maskelenemez interrupt girişi.
HALT	Durdurma girişi.
IRQ	Kesme isteği girişi.
MR	Yavaş hafızalar için gecikmeli hafıza oku girişi.
RE	128 byte'lık RAM belleğin etkinleştirilmesi girişi.
BA	Veri ve adres yollarının kullanılabilirlik durumu çıkışı.
E	Diğer çevre birimleri için gerekli saat sinyali çıkışı.
VMA	Adres yolundaki adresin geçerli olup olmadığını gösterir.
R/W	Oku/yaz sinyal çıkışı. "1" okuma "0" yazma sinyali.
Vcc Stb	128 byte'lık dâhilî belleğin stand-by beslemesi.
A0...A15	Adres yolu (16 bit).
D0...D7	Veri yolu (8 bit).

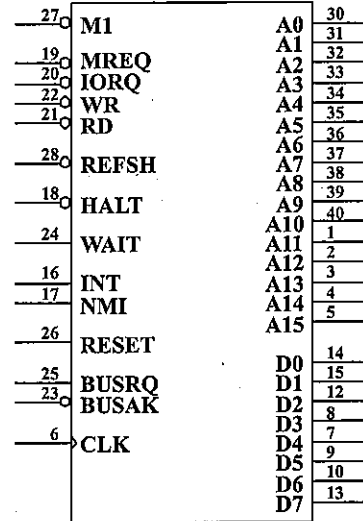


Şekil 6.12 : MC6802 mikroişlemcisinin ayak bağlantıları

6.3.2 Zilog Z80 Mikroişlemcisinin Özellikleri

- ❖ İç veri yolu ve harici veri yolu çıkışı 8 bitlik.
- ❖ Adres yolu çıkışı 16 bitlik.
- ❖ TTL uyumlu.
- ❖ 40 pinli bacak bağlantıları.
- ❖ 5 V'luk besleme girişi.
- ❖ Çalışma hızı 1-2,5 MHz.
- ❖ 24 adet kaydedici bulundurmaktadır.
- ❖ 8 bitlik kaydedicilerin, kaydedici çifti olarak 16 bit olarak kullanma imkânı vardır.

Z80 BACAK BAĞLANTILARI	
M1	Makine safhasının gidip alma (fetch cycle) adımı "0" olur.
MREQ	Hafıza okuma/yazma sırasında adres yolundaki bilgini n geçerli adres bilgisi olduğunu belirten çıkışı verir.
IORQ	Giriş/çıkış işlemi sırasında, adres yolunun düşük değerlikli kısmını kullanarak gerekli adrese ulaşıldığını belirten "0" çıkışı verir.
WR	Hafızaya veya giriş/çıkış portlarına yazma işlemi sırasında veri yoluna yazdırılacak verinin çıkmış olduğunu belirtir.
RD	Hafıza veya giriş/çıkış portlarından okuma yapacağı sırada veri yolunun okunacağını belirten çıkış sinyalini verir.
REFSH	Dinamik hafızaları tazelemek için adres yolunun düşük değerlikli kısmının bu iş için kullanılacağını belirtir. Bu sırada MREQ işaretinden yararlanılmaktadır.
HALT	Bir durma komutu sırasında "0" çı kışını verir. Bu sırada REFRESH aktivitesini devam ettirebilir.
WAIT	Adreslenmiş hafıza veya giriş/çıkış biriminin veri transferi için hazır olup olmadığını belirten giriştir. Bu giriş "0" de kaldığı sürece mikroişlemci çevre birimlerini bekler.
INT	Maskelenebilir kesme girişi.
NMI	Maskelenemez kesme girişi.
RESET	Mikroişlemciyi "0" sinyali ile resetleyerek başlangıçtaki konumuna getirir.
BUSRQ	Mikrobilgisayardaki veri ve adres yollarının o anda başka bir birim tarafından kullanıldığını belirten giriştir.
BUSAK	Mikrobilgisayardaki veri ve adres yolunun çevre birimlerince kullanılabileceğini işaret eden çıkıştır. DMA uygulamaları kullanılır.
CLK	Harici saat sinyali girişi.
A0...A15	Tek yönlü üç durumlu 16 bit adres yolu.
D0...D7	Çift yönlü üç durumlu 8 bit veri yolu.

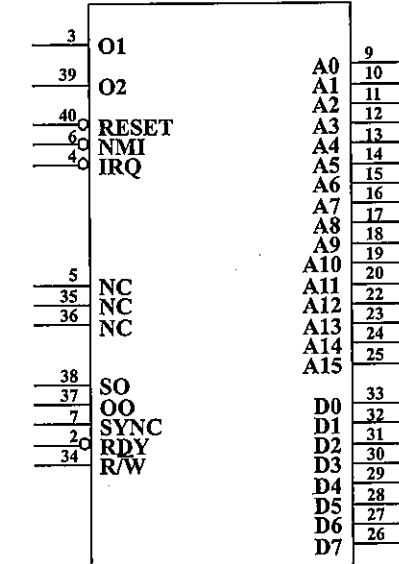


Şekil 6.13 : Z80 mikroişlemcinin bacak bağlantıları

6.3.3 Mos Tec. R6502 Mikroşlemcisinin Özellikleri

- ❖ Veri yolu 8 bitlik.
- ❖ Adres yolu 16 bitlik.
- ❖ 40 pinli bacak bağlantıları.
- ❖ 5 V'luk besleme girişi.
- ❖ Adres ve veri yolları buffer üzerinden TTL yüke bağlanmıştır.
- ❖ Çalışma hızı 1-2,5 MHz.
- ❖ Saati, içindedir.
- ❖ 6 adet 8 bit kaydedici bulundurmaktadır.

6502 BACAK BAĞLANTILARI	
RDY	Okuma safhasında "0" yapılırsa mikroşlemci bekleme hâline geçer. Bu uç düşük hızlı hafıza ve giriş/çıkış arabirimlerinde fayda sağlar.
O1, O2	İşlemcide üretilen iki fazlı saat sinyallerinin çıkışı.
IRQA	Maskelenebilir kesme sinyali.
NMI	Maskelenemez kesme girişi.
SYNC	Mikroşlemcide bilgi alma (fetch) işlemi sırasında SYNC, çıkışı sinyal üretir. İşlemcideki hataların tespitinde kullanılır.
R/W	Oku/yaz sinyal çıkışı.
XTALL	Harici saat sinyali girişi.
SO	SO girişi "0" olduğunda taşıma bayrağını kurmaktadır.
RESET	Yeniden başlatma girişi. Bu giriş en az 8 saat periyodu boyunca "0" da tutulduğunda mikroşlemci resetlenmiş olacaktır.
A0...A15	Tek yönlü üç durumlu 16 bit adres yolu.
D0...D7	Çift yönlü üç durumlu 8 bit veri yolu.

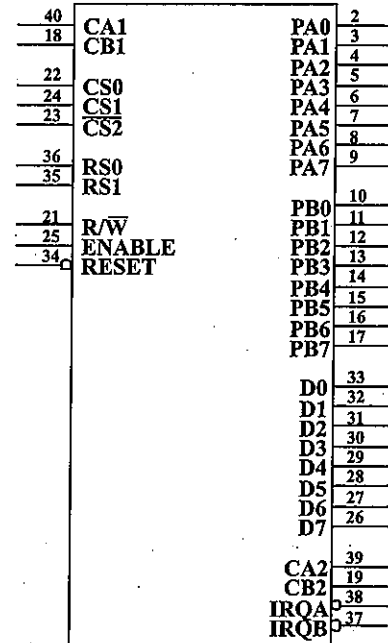


Şekil 6.14 : Mos Tec. 6502 mikroşlemcinin bacak bağlantıları

6.3.4 Motorola MC6821 Paralel Giriş/Çıkış Entegresinin Özellikleri

- ❖ 8 Bitlik iki yönlü ve üç durumlu veri yolu.
- ❖ Çevre birimlerle bağlantı kurmak için A ve B olmak üzere iki adet 8 bitlik giriş/çıkış portu.
- ❖ Ayır ayrı denetlenebilen el sıkışma ve kesme uçları.
- ❖ 3 durumlu giriş/çıkış portları.
- ❖ 650 mW güç harcaması.
- ❖ Resetlendikten sonra 1 sn sonra adreslenebilir hâle gelmektedir.
- ❖ 5 volt besleme gereksinimi.

MC6821 BACAK BAĞLANTILARI	
CA1, CA2	Çevre birimi kesme girişleri.
CS0, CS1, CS2	Entegrè seçme uçları.
RS0, RS1	İç kaydedici seçme uçları.
R/ W	Oku/yaz sinyal girişi.
ENABLE	Yetkilendirme darbesi, sistem saati ile birlikte kullanılır.
RESET	Lojik "0" da etkinleşen resetleme girişi.
PA0...PA7 PB0...PB7	8'er bitlik giriş/çıkış portları.
D0...D7	Veri yolu giriş/çıkış hatları.
CA2,CB2	Çevre birimi denetim hatları.
IRQA , IRQB	Kesme istek girişleri.

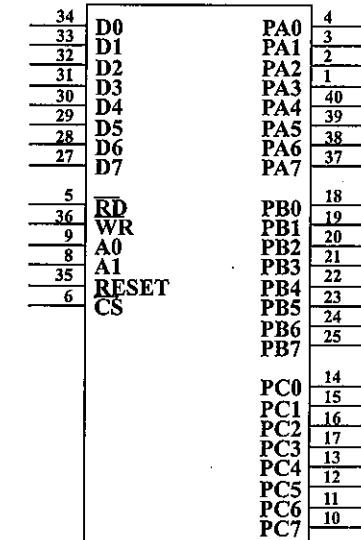


Şekil 6.15 : MC6821 PIA bacak bağlantıları

6.3.5 Intel 8255 Paralel İletişim Entegresinin Özellikleri

- ❖ Intel ve diğer mikroişlemcilerle tam uyumlu.
- ❖ Yüksek hızlı, sıfır bekleme durumu (zero wait state).
- ❖ 8 MHz işlem hızı.
- ❖ 24 adet programlanabilen giriş/çıkış uçları.
- ❖ Düşük güç sarfiyatı.
- ❖ TTL uyumlu.
- ❖ Tüm çıkış uçları 2,5 mA yük sürebilir.
- ❖ Üç ayrı moda çalışma imkânı:
MOD1; standart giriş/çıkış.
MOD2; el sıkışmalı standart giriş/çıkış.
MOD3; çift yönlü giriş/çıkış (A ve C portunun yüksek değerlikli kısmı).

8255 BACAK BAĞLANTILARI	
D0...D7	8 bitlik veri giriş uçları.
RD	Okuma sinyali girişi lojik "0" da aktif.
WR	Yazma sinyali girişi lojik "0" da aktif.
A0, A1	Kaydedicilerin seçimi (Base; taban adresi) Base+0 = Port A (oku veya yaz) Base+1 = Port B (oku veya yaz) Base+2 = Port C (oku veya yaz) Base+3 = Kontrol Register (yaz)
RESET	Tüm kaydediciler resetlenerek ilk konuma gelir.
CS	Seçme ucu girişi.
PA0...PA7	Çift yönlü (giriş/çıkış) A portu 8 bit.
PB0...PB7	Çift yönlü (giriş/çıkış) B portu 8 bit.
PC0...PC7	Çift yönlü (giriş/çıkış) C portu 8 bit.

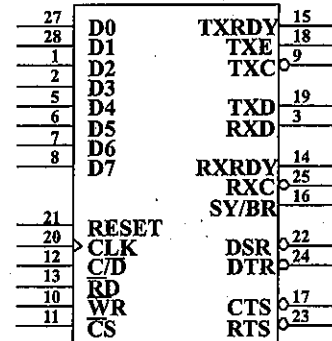


Şekil 6.16 : Intel 8255 PIA'nın bacak bağlantıları

6.3.6 Intel 8251A Seri İletişim Entegresinin Özellikleri

- ❖ Asenkron ve senkron çalışabilme özelliği.
- ❖ Senkron ve asenkron 5-8 bitlik karakterler gönderebilme.
- ❖ Saat hızının Baud hızının 1, 16 veya 64 katı.
- ❖ Çift yönlü, çift tamponlu gönderici ve alıcı.
- ❖ Hatalı başlatma bitinin algılanması.
- ❖ Eşlik ve çığneme hatası kontrolü.
- ❖ 8080/8085 mikroişlemciler ile tam uyumlu.
- ❖ 28 pinli bacak bağlantıları.
- ❖ Bütün giriş ve çıkışlar TTL uyumlu.
- ❖ Tek bir 5 V'luk besleme kaynağı.
- ❖ Tek saat girişi (TTL uyumlu).

8251 BACAĞ BAĞLANTILARI	
D0...D7	Veri yolu giriş/çıkış.
RESET	Sıfırlama girişi.
CLK	Saat girişi TTL.
C/D	Denetim/ bildirme girişi.
RD	Veri yada kaydediciyi oku girişi.
WR	Veri yada denetim kaydedicisine yaz girişi.
CS	Seçme girişi.
TxRDY	Gönderici hazır (MIB'den karakter almak için).
TxE	Gönderici boş.
TxC	Gönderici saati.
TxD	Gönderici verisi.
RxD	Alıcı verisi.
RxRDY	Alıcı hazır (MIB için karaktere sahip).
RxC	Alıcı saati.
SY/BR	Senkronizasyon algılama/durma algılama.
DSR	Veri grubu hazır.
DTR	Veri terminali hazır.
CTS	Veri gönderme izni.
RTS	Veri gönderme isteği.

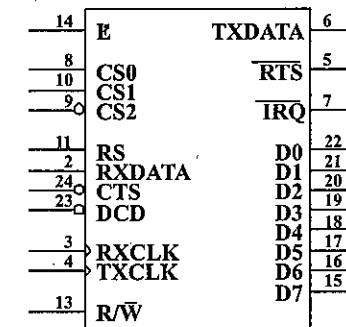


Şekil 6.17 : 8251'in bacak bağlantıları

6.3.7 Motorola 6850 Seri Giriş/Çıkış Entegresinin Özellikleri

- ❖ 5 V besleme gerilimi.
- ❖ Kristal osilatör girişi.
- ❖ TTL uyumlu.
- ❖ Maksimum 19600 baud iletişim hızı.
- ❖ Sistem ile 8 bitlik veri iletişim hattı.

MC 6850 BACAĞ BAĞLANTILARI			
E	Senkronizasyon saat girişi (sistem saatinden alır).		
CS0, CS1, CS2	Entegre seçme girişleri.		
RS	Dâhilî kaydedici seçme girişi ;		
	RS	R/W	Seçilen kaydedici
	0	0	Denetim kaydedicisi
	0	1	Durum kaydedicisi
	1	0	Verici kaydedicisi
	1	1	Alıcı kaydedicisi
RxDATA	Gelen veri girişi.		
CTS	Göndermek için sil.		
DCD	Taşıyıcı sinyal görüldü.		
RxCLK	Alıcının iletişim hızını belirleyen saat işareti girişi.		
TxCLK	Vericinin iletişim hızını belirleyen saat işareti girişi.		
R/W	Oku/yaz sinyal girişi.		
TxDATA	Giden veri çıkışı.		
RTS	Göndermek için istek .		
IRQ	Kesme oluşturan bir durumla karşılaşıldığında mikroişlemciye kesme sinyali üretir.		
D0...D7	8 bitlik veri girişi.		

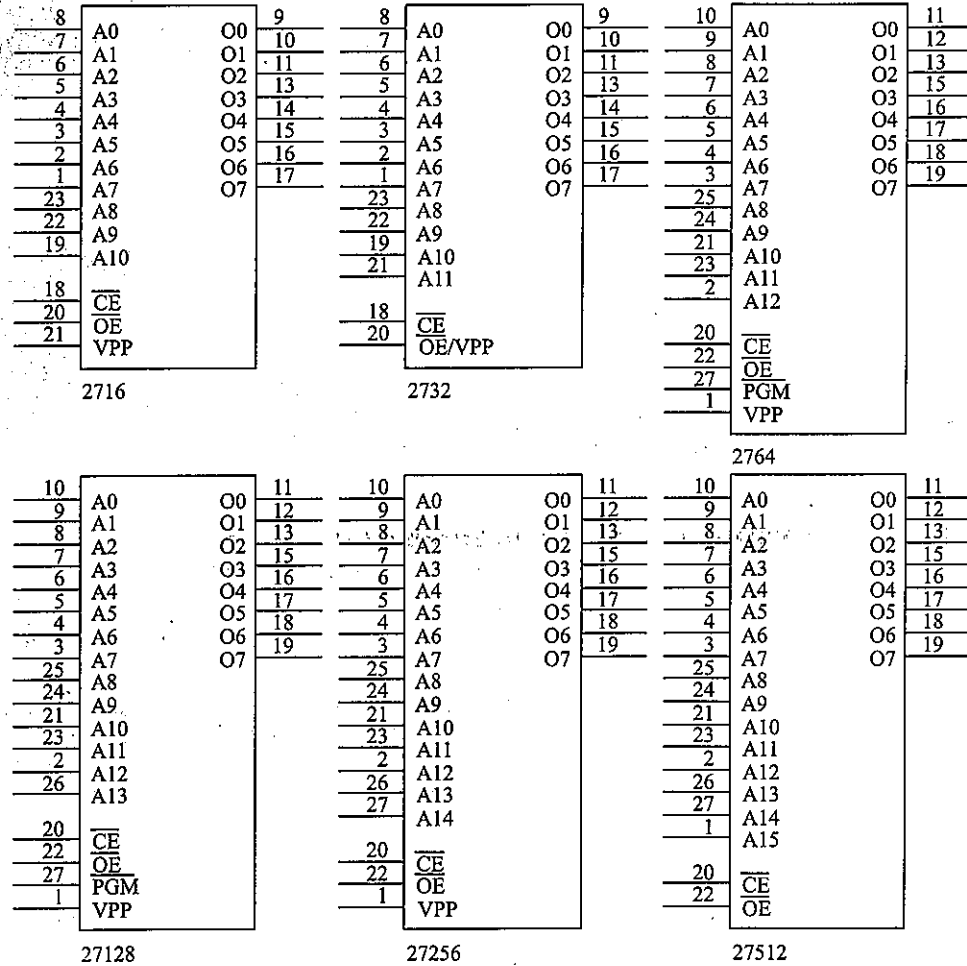


Şekil 6.18 : MC6850 ACIA ayak bağlantıları

6.3.8 2764 EPROM'un Özellikleri

- ❖ 8 Kb veri depolama kapasitesi.
- ❖ 13 bitlik adres yolu genişliği.
- ❖ Ultraviyole ışık ile silinebilme (10-15 dk).
- ❖ 5 volt besleme 25 V programlama gerilimi.
- ❖ Hızlı erişim süresi ortalama 450 ns.
- ❖ Her hücre için 50 msn programlama darbe süresi . Tamamını programlama süresi ortalama 400 sn.
- ❖ Düşük güç harcaması. yaklaşık 500 mW. stand-by bekleme modu ile %75 enerji tasarrufu.
- ❖ Tüm giriş ve çıkışlar TTL uyumlu statik elektrik korumalı.

2716...27512 BACAK BAĞLANTILARI			
A0...A12	Adres yolu 13 bit.		
$\overline{CE}$	Entegre seçme girişi.		
$\overline{OE}$	Veri çıkışı yetkilendirme girişi.		
$\overline{PGM}$	Programlama yetki girişi.		
Vpp	Programlama voltaj girişi.		
O0..O7	8 bit veri çıkışı.		

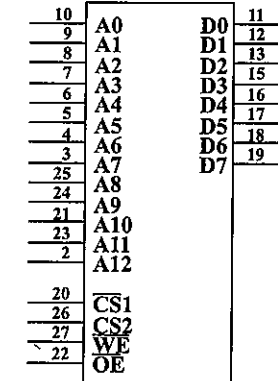


Şekil 6.19 : Değişik kapasitedeki EPROM'ların bacak bağlantıları

6.3.9 6264 RAM'in Özellikleri

- ❖ 8 Kb veri depolama kapasitesi.
- ❖ 5 voltluk besleme gerilimi.
- ❖ Erişim süresi ortalama 350 ns.
- ❖ 13 adres hattı.
- ❖ 8 bit veri yolu genişliği (1 byte).
- ❖ TTL uyumlu tümüyle statik.
- ❖ Stand-by modu (güç harcaması olmadan uyuma modu).

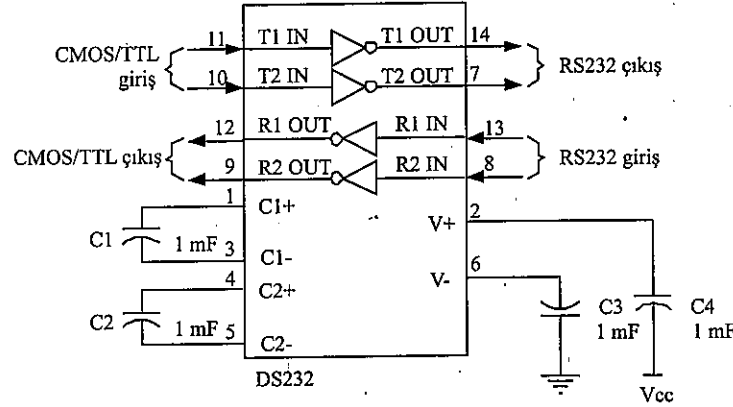
6264 BACAK BAĞLANTILARI	
A0...A12	Adres yolu 13'bit
$\overline{CS1}, \overline{CS2}$	Entegre seçme girişi.
$\overline{WE}$	Veri yazma yetki ucu.
$\overline{OE}$	Veri okuma yetki ucu.
D0..D7	8 bit veri çıkışı.



Şekil 6.20 : 6264 (8 Kb) RAM'in ayak bağlantıları

6.3.10 RS232 uygunlaştırıcı Max232 Entegresinin Özellikleri

Max 232 BACAK BAĞLANTILARI	
T1, T2 IN	TTL, CMOS sinyal girişi.
R1, R2 IN	TTL, CMOS sinyal çıkışı.
T1, T2 OUT	RS232 çıkış.
R1, R2 OUT	RS232 girişi.
C1+, C1-	Kondansatör bağlantısı.
C2+, C2-	Kondansatör bağlantısı.
V+, V-	Kondansatör bağlantısı.



Şekil 6.21 RS232 uygunlaştırıcı Max 232'nin devre bağlantısı

DEĞERLENDİRME ÇALIŞMALARI

- 1) Hafızalar genel olarak kaç'a ayrılır? Bunları özelliklerine göre açıklayınız.
- 2) Manyetik hafıza elemanlarının avantaj ve dezavantajlarını yazınız.
- 3) Hard diskin yapısını ve çalışmasını anlatınız.
- 4) Floppy diskin yapısını ve özelliğini anlatınız.
- 5) Floppy diskin bölümlerini şekil çizerek anlatınız.
- 6) Statik hafızaların genel özelliklerini yazınız.
- 7) Statik hafıza elemanlarını yazınız.
- 8) Dinamik hafıza ve statik hafıza elemanları arasındaki farklılıkları belirtiniz.
- 9) CD-ROM'ların yapısı ve çeşitleri hakkında bilgi veriniz.
- 10) Mikrobilgisayarların çevre birimleri ile iletişim sağlaması için kullanılan arabirimlerinin çeşitlerini ve görevlerini açıklayınız.
- 11) Paralel iletişimin özelliklerini ve kullanım alanlarını yazınız.
- 12) Seri iletişimin özelliklerini ve kullanım alanlarını yazınız.

ÜNİTE 7 8080/8085, 8086/8088 ve 80286/80386 MİKROİŞLEMCİLERİ

7.1 INTEL 8080/8085 MİKROİŞLEMCİLERİ

Intel 8080 mikroişlemcisi kendisi ile birlikte 8224 saat üretici, 8228 çift yönlü veri iletim sürücüsü ve kontrol entegresi ile birlikte kullanılarak çalışmasını gerçekleştirir. Aşağıda 8080 ve 8085 mikroişlemcilerinin teknik özellikleri ve bacak bağlantıları verilmiştir.

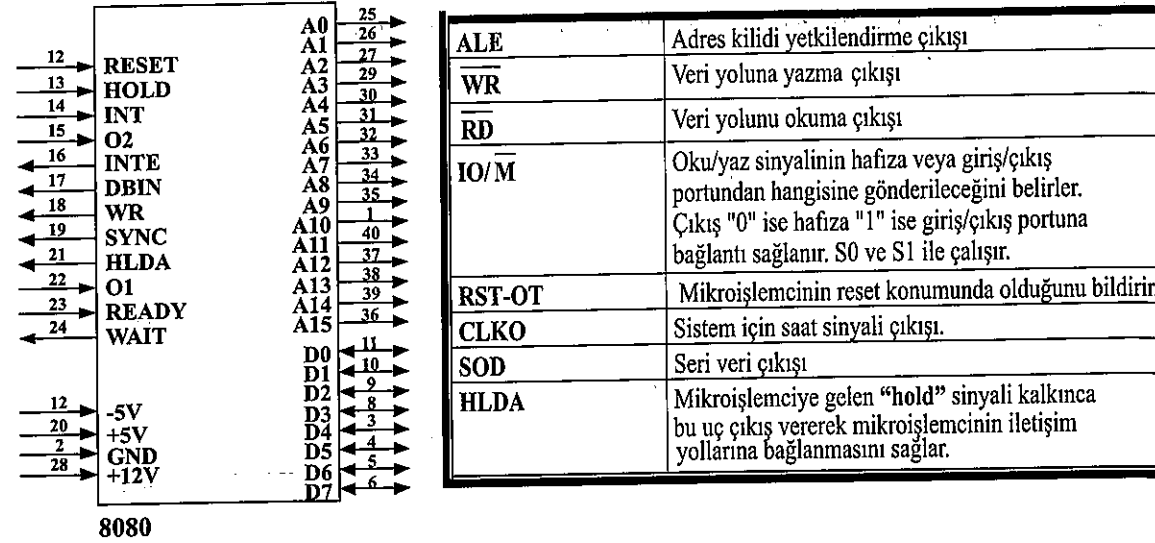
8080/8085 mikroişlemcilerinin teknik özellikleri:

- ❖ Yapım teknolojisi NMOS (8085'de sinyal hatları TTL uyumlu)
- ❖ Besleme gerilimi +5V'tur. (8080 de +12V, +5V, -5V)
- ❖ 64 Kb'lık hafıza bölgesini doğrudan adresleyebilir.
- ❖ Seri giriş/çıkış bağlantı uçları SID, SOD (8085'de)
- ❖ Veri uzunluğu 8 bittir.
- ❖ 8 bitlik çift yönlü veri yolu çıkışı
- ❖ 16 bitlik adres yolu çıkışı (doğrudan adresleme aralığı 65536 byte)
- ❖ Saat frekansları 0.5 MHz-3 MHz.
- ❖ Komut hızı tipik olarak 2 µsn (8085'de 0,8 µsn ... 1,3 µsn).
- ❖ 8085 saat sürücüsü içindedir.
- ❖ Temel komut sayısı 78 adet (8085'de 80 adet)
- ❖ Bacak bağlantı sayısı 40 pin
- ❖ Güç gereksinimi 110 mA (8085 170 mA)

36	RST-IN	AD0	12
1	X1	AD1	13
		AD2	14
		AD3	15
2	X2	AD4	16
		AD5	17
5	SID	AD6	18
6	TRAP	AD7	19
		A8	20
9	RST 5.5	A9	21
8	RST 6.5	A10	22
7	RST 7.5	A11	23
		A12	24
10	INTR	A13	25
		A14	26
11	INTA	A15	27
29	S0	ALE	30
33	S1	WR	31
		RD	32
39	HOLD	IO/M	33
35	READY	RST-OT	34
		CLKO	35
		SOD	36
		HLDA	37

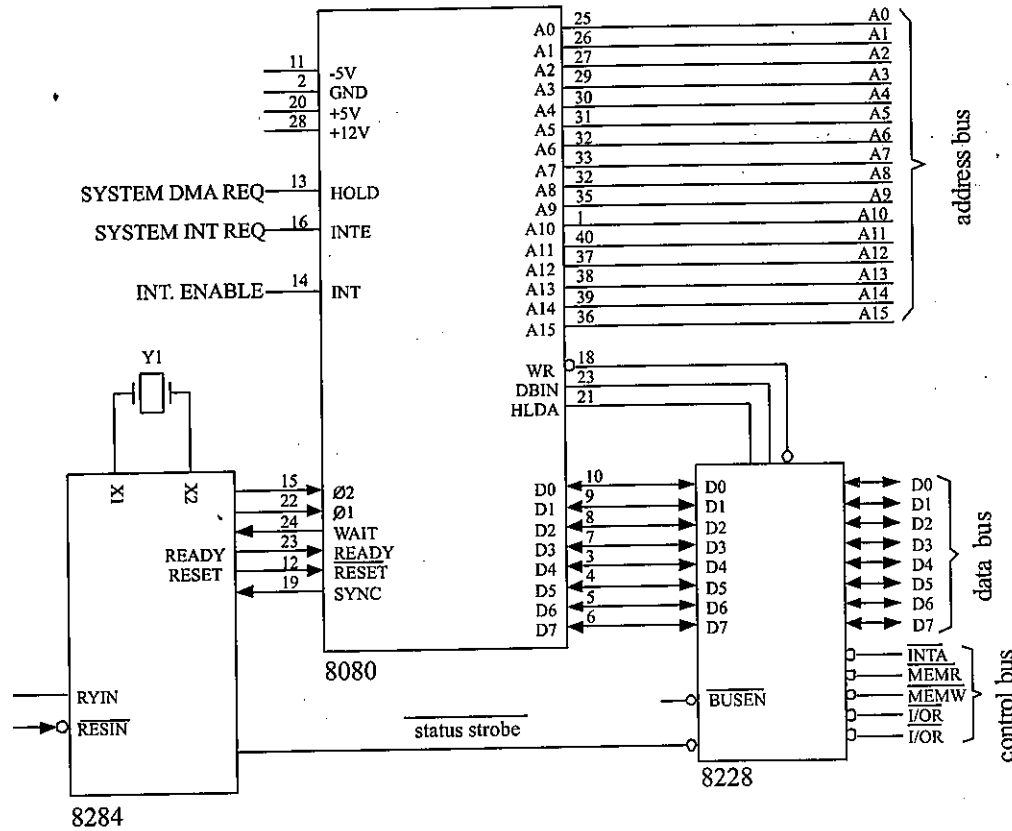
8085

8080 ve 8085 Bacak Bağlantıları			
RST - IN	Mikroişlemciyi "0" sinyali ile resetler.		
X1, X2	Harici kristal girişi		
SID	Seri veri girişi		
TRAP	Maskelenemez kesme girişi. (vektör adresi: (0024) ₁₆ , önceliği -1-)		
RST 7,5	Maskelenebilir kesme girişleri (vektör adresi: (003C) ₁₆ , önceliği -2-) (vektör adresi: (0034) ₁₆ , önceliği -3-). (vektör adresi: (002C) ₁₆ , önceliği -4-)		
RST 6,5			
RST 5,5			
INTR	Genel amaçlı kesme konumuna geçişi sağlar.		
INTA	Kesme bilgisi çıkışı. Kesme alındığında oku (RD) komutu yerine geçer. Bu komut kesme kapısını harekete geçirir. INTR'ye göre daha önceliklidir.		
S0, S1	Veri yolu komut kod çözücüsü		
	S1	S0	Durum
	0	0	Tutma hâli (HALT)
	0	1	Yazma hâli (write)
	1	0	Okuma hâli (read)
	1	1	Veri getirme (fetch)
HOLD	Hold giriş adres veya veri yollarının diğer çevre birimleri tarafından kullanıldığını bildirir.		
READY	Çevre birimlerinden okuma/yazma gerçekleştirileceğini bildiren giriş ucu.		
AD0...AD7	Giriş ve çıkış olarak kullanılır. Veri ve adres yolunun düşük değerlikli kısmını üretir.		
A8...A15	Adres yolunun yüksek değerlikli kısmı olarak hafıza ve giriş/çıkış adreslerinde kullanılır.		



8080

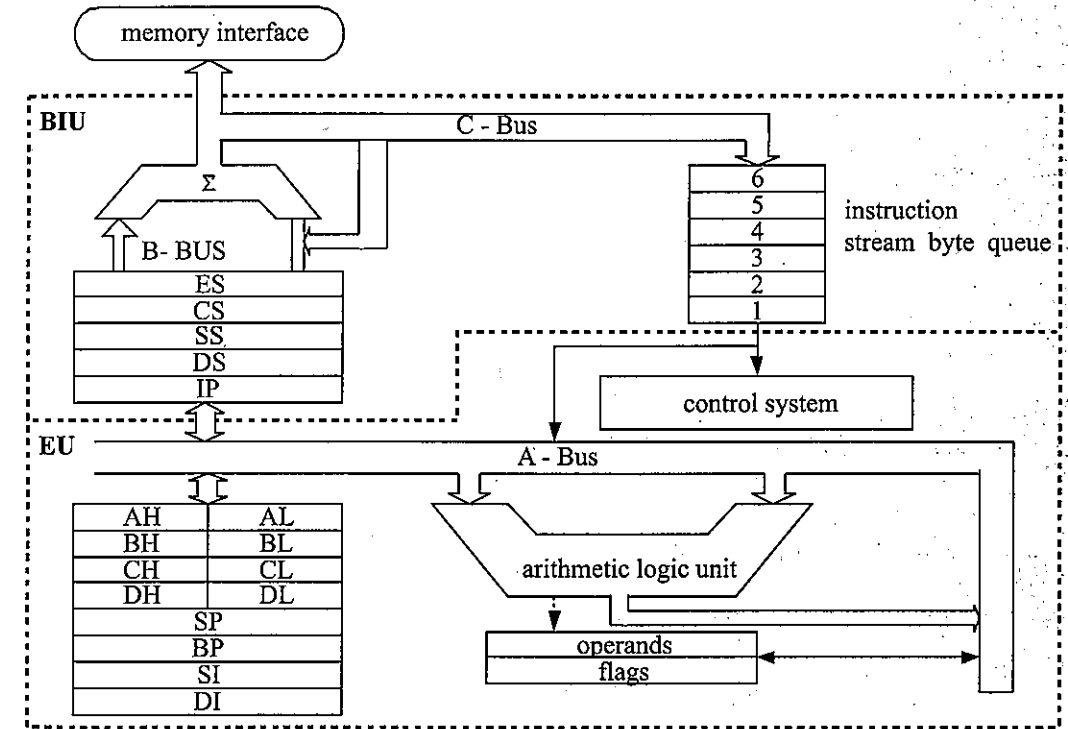
Şekil 7.1'de 8080 entegresi ile birlikte kullanılan yardımcı devre elemanlarının bağlantı şeması görülmektedir.



Şekil 7.1 : 8080 mikroşlemcisi, saat ve kontrol entegrelerinin bağlantısı

7.2 INTEL 8086/8088 MİKROİŞLEMCİLERİ

Intel'in üretmiş olduğu 16 bitlik 8086 (üretim yılı 1978) ve 8088 (üretim yılı 1979) mikroşlemcileri benzer mimari özellikler taşıyan mikroşlemcilerdir. Bu mikroşlemcilerde 8080'de kullanılan yardımcı devre elemanları mikroşlemci içerisinde bulunmaktadır. Aynı komut setini kullanırlar. Bu mikroşlemcilerin arasındaki en büyük fark dış veri yoludur. 8088 mikroşlemcisi 8 bitlik, 8086 mikroşlemcisi 16 bitlik dış veri yolu kullanır. Adres ve veri yollarını multiplexer şeklinde ortak kullanırlar. Bu farkla, dış veri akışı 8086'da daha hızlı gerçekleşmektedir. Ayrıca bu mikroşlemciler 8087 matematik işlemcisi ile takviye edildiklerinde matematiksel işlemlere (çarpma, bölme, trigonometri...) destek sağlayabilmektedirler. Bu mikroşlemciler harici saat devresine ihtiyaç duyduklarından bunun için "8084 clock generatör" entegresi kullanılmaktadır. 8 adet 16 bitlik genel kaydedici, 4 adet 16 bitlik segment kaydedici ve bir adet komut kaydedici bulunmaktadır.



Şekil 7.2 : 8086/8088 mikroşlemcilerinin iç yapısı

Şekil 7.2'deki 8086/8088 mikroşlemcilerinin blok diyagramındaki bölümlerin görevleri şöyledir:

Adres yolu arabirimi (BIU; bus interface unit):

Instruction stream byte queue: Hızlı çalışmayı sağlamak amacı ile hafızadan gelen komutların kontrol devresini yönlendirebilmesi için kullanılan 1'er byte'lık altı kaydedici dizisi.

Address adder (W): Adres düzenleyici.

Extra segment (ES): Ekstra ek segment. Veri için kullanılmaktadır.

Code segment (CS): Kodlama segmenti. Komut adresleri için kullanılır.

Stack segment (SS): Yığın segmenti. İndisli veri işlemlerde sıralama kaydedicisi olarak kullanılır.

Instruction pointer register (IP): Komut adresi kaydedicisi.

Yürütme (komut işletme) birimi (EU, execution unit):*Arithmetic logic unit:* Aritmetiksel ve mantıksal işlem birimi*Control system:* Kontrol sistemi*Flags (bayraklar):* 9 bitlik bir kaydedici olup, aritmetik-mantık işlemleri sonucunda oluşabilecek bazı durumları (sıfır, negatif, elde, yardımcı elde vb.) ve yetkilendirmeleri (trap, interrupt vb.) gösterir.*A, B, C, D registers:* Genel amaçlı kaydediciler. 8 veya 16 bitlik işlemlerde kullanılır (AH ve AL 8085'deki kaydedici görevini yerine getirir.).*Stack pointer (SP):* Yığın kaydedicisi. Veri yığınlarının adreslerini tutar.*Base pointer (BS):* Veri yığınlarının adreslerini tutar. Yığın yolu ile giden verilerin "stack segment"e ulaşmasında kullanılır.*Source indeks register (SI):* Kaynak kaydedici.*Destination indeks register (DI):* Hedef kaydedicisi. Verilerin, veri hafızasına ulaşmasını sağlamak için adres kaydedici olarak kullanılır.

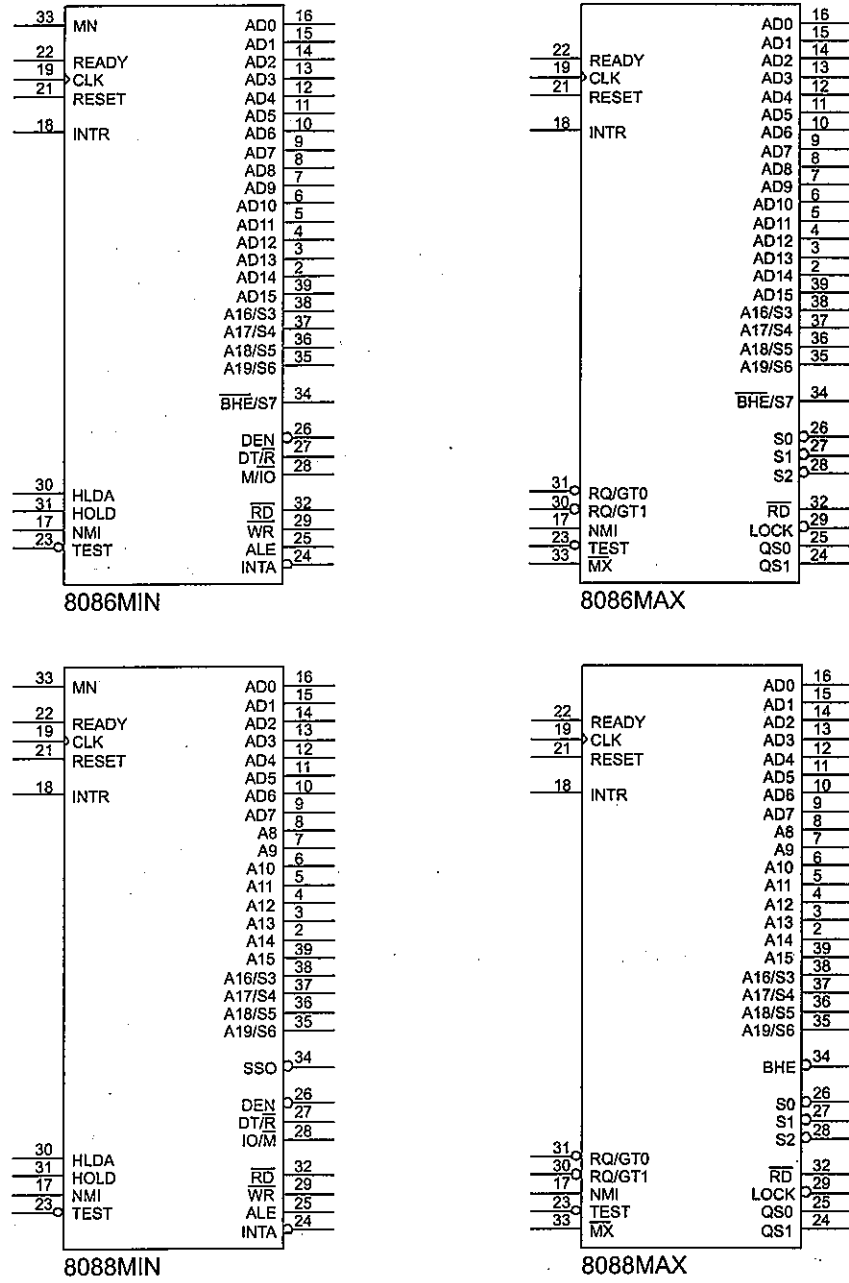
8086 ve 8088 mikroişlemcilerinde, "minimum mod" ve "maksimum mod" diye iki tür çalışma şekli vardır. Her çalışma moduna göre bazı bacak bağlantıları değişmektedir. Mikroişlemci hangi mod'da kullanılacaksa sistem ona göre tasarlanmalıdır. Mikroişlemcinin çalışma modunu entegrenin 33 nolu bacağı belirler. Bu bacak "0" yapıldığında mikroişlemci maksimum mod'da, "1" iken minimum moda çalışacaktır. Tablo 7.1'de bu entegrelerin maksimum veya minimum modlardaki bacak bağlantıları ve görevleri verilmiştir.

Tablo 7.1 : 8086 ve 8088 bacak bağlantıları ve görevleri (maksimum ve minimum mod)

Min	Max	8086 ve 8088 bacak bağlantıları
(33) MN	(33) MX	Mikroişlemcinin "maksimum mod" veya "minimum mod" çalışmasını belirleyen girişler.
(22) READY		Harcı birimlerin hazır olduğunu bildiren giriş.
(19) CLK		Mikroişlemcinin saat girişi. 8284 saat üretici entegresi tarafından sağlanır.
(21) RESET		"0" yapıldığında o anda yapılan bütün işlemler durur, kaydedici içerikleri sıfırlanır. Mikroişlemci (FFFF) ₁₆ adresinden tekrar başlamak üzere hazır hâle gelir. En az 10 makine periyodu "0" da tutulmalıdır.
(18) INTR		Maskelenebilir kesme girişi. Bu giriş interrupt bayrağının "0" yapılması ile maskelenebilir.
(30) HLDA	(30) RQ/GT0, (31) RQ/GT1	Mikroişlemciye gelen "hold" sinyali kalkınca bu uç çıkış vererek mikroşlemcinin iletişim yollarına bağlanmasını sağlar
(31) HOLD		Hold giriş adres veya veri yollarının diğer çevre birimleri tarafından kullanıldığını bildirir.
(17) NMI		Bu girişin "0" dan "1" geçişinde kesme işlemi gerçekleşir. Maskelenemez
(23) TEST		Bir "wait" komutu ile mikroşlemci, test yapılması için çalışmasını durdurur. Bu giriş "0" yapıldığında tekrar çalışmasını sürdürür.

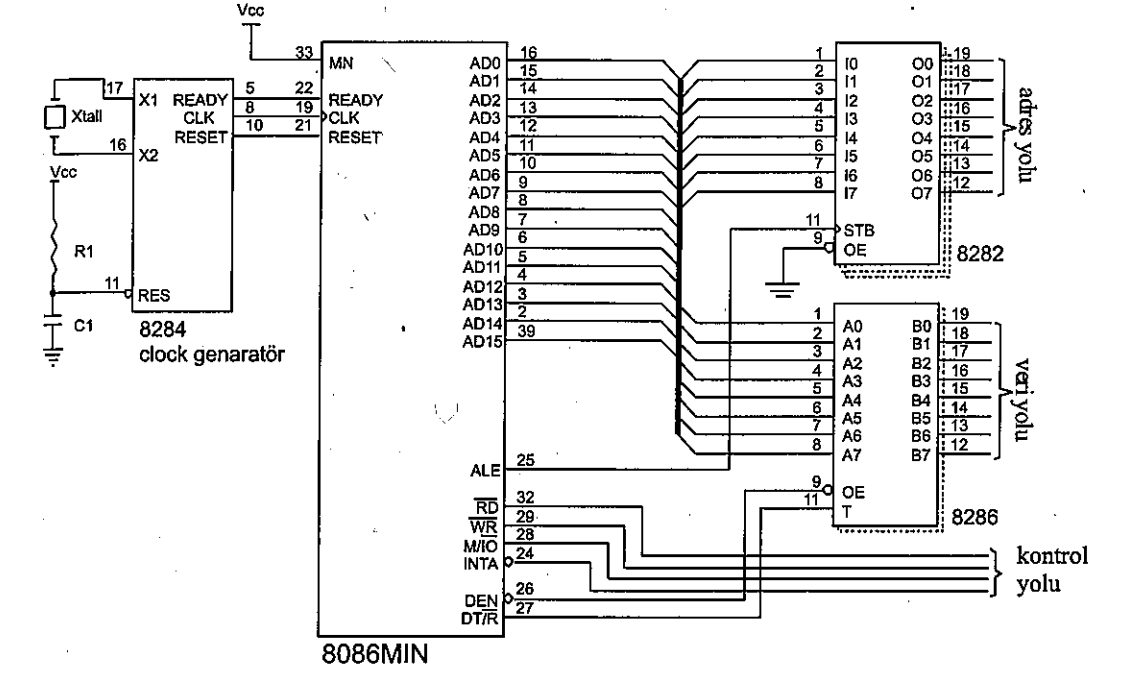
Tablo 7.1'in devamı

AD0...AD15 AD0...AD7(8088)	Ortak veri ve adres yolu çıkışları. 8086'da 16 bitlik, 8088'lerde 8 bitlik veri giriş/çıkışı. Her ikisinde de 16 bitlik adres yolu çıkışı.																																				
(38) A16/S3 (37) A17/S4	Birinci saat periyodunda adres çıkışı olarak görev yapar. Diğer saat periyotlarında çıkış kombinasyonlarına göre hafızanın segment ayrımını belirleyen etkinlik göstermektedir. <table border="1"> <tr> <th>S3</th><th>S4</th><th>Segment kaydedicileri</th></tr> <tr> <td>0</td><td>0</td><td>Extra segment</td></tr> <tr> <td>0</td><td>1</td><td>Stack segment</td></tr> <tr> <td>1</td><td>0</td><td>Code segment</td></tr> <tr> <td>1</td><td>1</td><td>Data segment</td></tr> </table>	S3	S4	Segment kaydedicileri	0	0	Extra segment	0	1	Stack segment	1	0	Code segment	1	1	Data segment																					
S3	S4	Segment kaydedicileri																																			
0	0	Extra segment																																			
0	1	Stack segment																																			
1	0	Code segment																																			
1	1	Data segment																																			
(36) A18/S5	Birinci saat periyodunda adres çıkışı olarak görev yapar. Diğer periyotlarda kesme bayrağının durumunu gösterir.																																				
(35) A19/S6	1. saat periyodundan sonra "kontrol yolunu" kontrol etmek için "0" olur.																																				
(34) BHE /S7 (8086)	Okuma/yazma ve kesme işlemlerinde "0" olurken veri yolunun D8...D15 hattı iletimi gerçekleşir. AD0 hattı ile birlikte kullanılarak hafıza alanlarının (banklarının) seçiminde kullanılır.																																				
(34) SSO (8088Min)	Min mod'da durum hattı olarak kullanılır. Max mod'da S0 ile aynı işleve sahiptir.																																				
(26) DEN : Veri etkin çıkışı	(26) S0, (27) S1, (28) S2 : Bu çıkışlar 8288 kontrol entegresine uygulanarak kontrol yolunda aşağıdaki işlevler elde edilir. <table border="1"> <tr> <th>S2</th><th>S1</th><th>S0</th><th>Yapılan işlem</th></tr> <tr> <td>0</td><td>0</td><td>0</td><td>Kesme bilgisi</td></tr> <tr> <td>0</td><td>0</td><td>1</td><td>Çıkış portunu okuma</td></tr> <tr> <td>0</td><td>1</td><td>0</td><td>Çıkış portunu yazma</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>Durdurma</td></tr> <tr> <td>1</td><td>0</td><td>0</td><td>Kodlama</td></tr> <tr> <td>1</td><td>0</td><td>1</td><td>Hafızayı okuma</td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>Hafızaya yazma</td></tr> <tr> <td>1</td><td>1</td><td>1</td><td>Pasif durum</td></tr> </table>	S2	S1	S0	Yapılan işlem	0	0	0	Kesme bilgisi	0	0	1	Çıkış portunu okuma	0	1	0	Çıkış portunu yazma	0	1	1	Durdurma	1	0	0	Kodlama	1	0	1	Hafızayı okuma	1	1	0	Hafızaya yazma	1	1	1	Pasif durum
S2		S1	S0	Yapılan işlem																																	
0		0	0	Kesme bilgisi																																	
0	0	1	Çıkış portunu okuma																																		
0	1	0	Çıkış portunu yazma																																		
0	1	1	Durdurma																																		
1	0	0	Kodlama																																		
1	0	1	Hafızayı okuma																																		
1	1	0	Hafızaya yazma																																		
1	1	1	Pasif durum																																		
(27) DT/R : Veri gönderme/alma sinyali																																					
(28) M/IO : Okunacak veya yazılacak verinin hafıza veya giriş/çıkış birimlerinden hangisinin olacağını belirleyen çıkış																																					
(32) RD	Hafıza veya giriş/çıkış birimlerinden "veri oku" sinyal çıkışı. Okumanın hafıza veya IO dan mı olacağını M/IO çıkışı belirler.																																				
(29) WR : Hafıza veya giriş/çıkış birimlerine "veri yaz" sinyal çıkışı.	(29) LOCK : Paylaşılan kaynakların geçici olarak kapatılmasında kullanılır. Sistem yolları başka devrelerce kullanılmaz.																																				
(25) ALE : Adres yolunu etkinleştirir.	(25) QS0, (24) QS1: Birden fazla mikroişlemci ile çalışıldığında getirilen komutların ana mikroişlemciden mi yoksa diğer devrelerden mi alınacağını belirler.																																				
(24) INTA : Kesme bilgisi çıkışı																																					

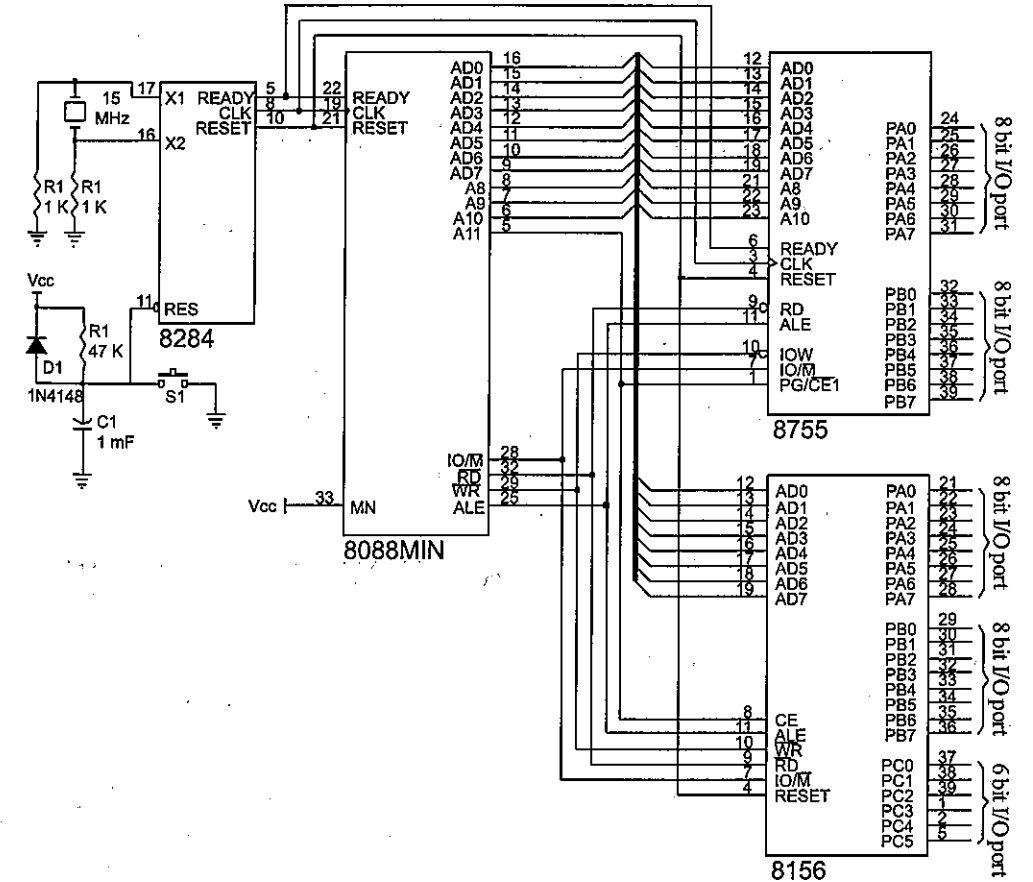


Şekil 7.3 : 8086 ve 8088 mikroişlemcilerinin maksimum ve minimum çalışma modlarındaki bacak bağlantıları.

Minimum mod'da çalışma: Öncelikle mikroişlemcinin bu moda kullanılması için MN/MX girişinin Vcc'ye bağlanması yani "1" yapılması gerekir. Minimum mod tek mikroişlemcili uygulamalarda kullanılır. Minimum mod'da M/I $\overline{O}$ (8088'de IO/M), $\overline{INTA}$, $\overline{RD}$, $\overline{WR}$ çıkışları doğrudan kontrol hatları olarak kullanılır. 8086 ve 8088 mikroişlemcileri adres ve veri yolunu ortak kullandıkları için bu yollar minimum mod da 8755 ve 8156 entegreleri kullanılarak elde edilir. Şekil 7.4 ve şekil 7.5'deki şemalarda minimum modda'da kullanılan 8086 ve 8088 mikroişlemcilerinin bağlantıları görülmektedir.

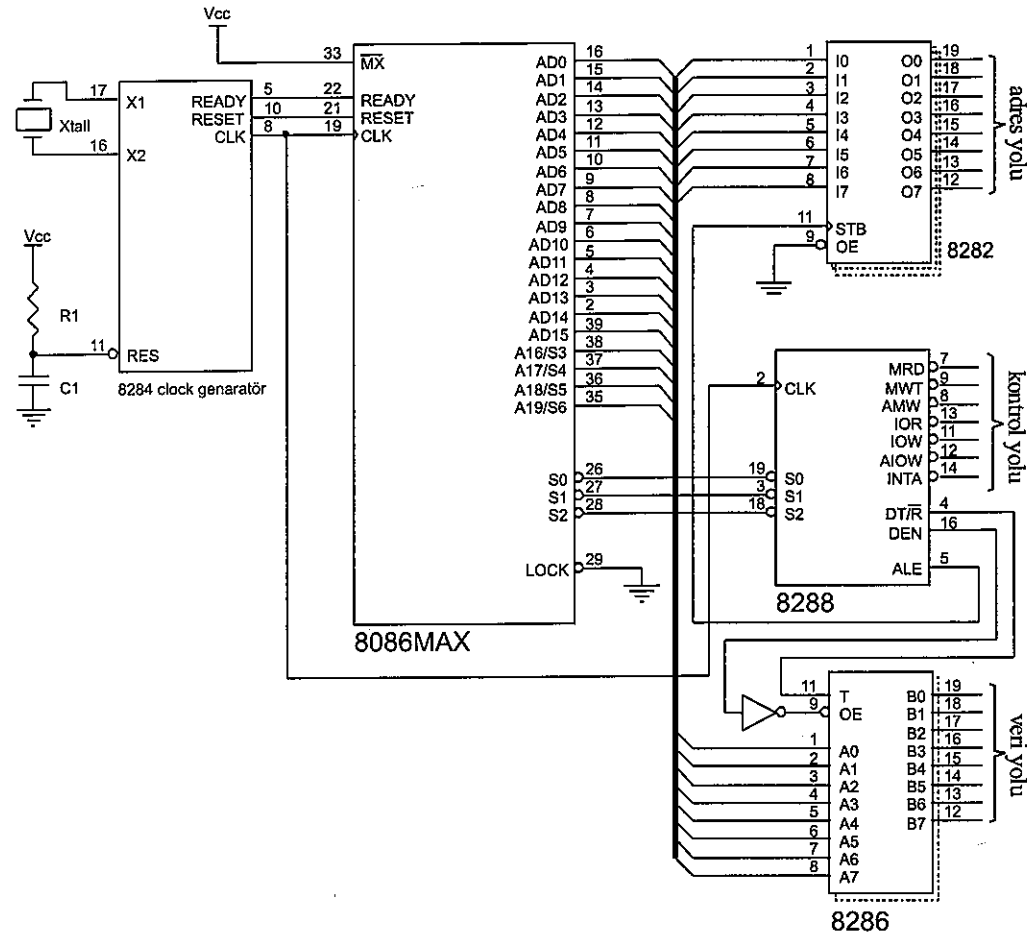


Şekil 7.4 : 8086'nın minimum çalışma modundaki temel devre bağlantısı



Şekil 7.5 : 8088'in minimum çalışma modundaki temel devre bağlantısı

Maksimum mod'da çalışma: Öncelikle mikroişlemcinin bu moda kullanılması için $MN/\overline{MX}$ girişinin V_{ss} 'ye bağlanması yani "0" yapılması gerekir. Maksimum mod'da 8088 ve 8086 mikroişlemcilerinin bağlantıları hemen hemen aynıdır. Maksimum mod çoklu mikroişlemci uygulamalarında kullanılır. Maksimum mod'da 8288 kontrol entegresi tarafından $\overline{S0}$, $\overline{S1}$ ve $\overline{S2}$ kontrol çıkışlarından aldığı sinyallere kodlayarak yeni kontrol sinyalleri üretilir.



Şekil 7.6 : 8086 mikroişlemcisinin maksimum mod bağlantısı

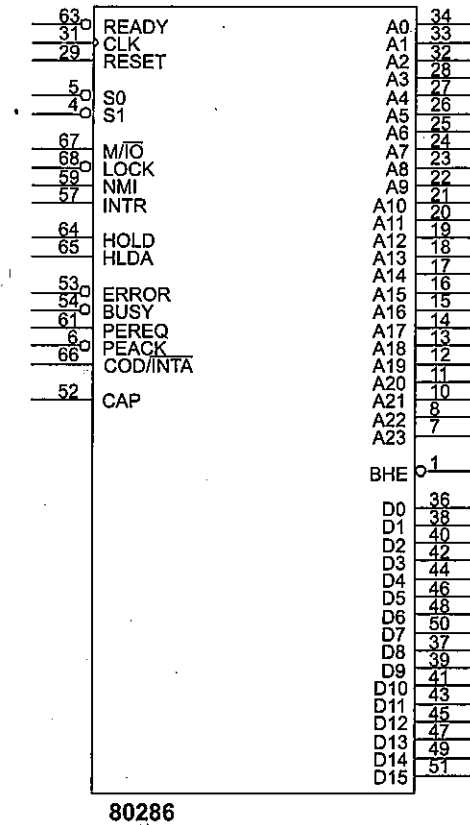
7.3 INTEL 80286 MİKROİŞLEMCİSİ

7.3.1 80286 Mikroişlemcisinin Teknik Özellikleri

- ❖ Değişik saat hızlarında çalışan türleri vardır.
25 MHz (80C286-25), 20 MHz (80C286-20), 16 MHz (80C286-16), 12 MHz (80C286-12)
- ❖ Durağan (stand-by) akım sarfiyatı $I_{cc}=5\text{ mA}$, çalışma hâlindeki akım sarfiyatları;
12 MHz'de max. 220 mA, 16 MHz'de max. 260 mA 20 MHz'de max 310 mA 25 MHz'de max 360 mA

Saat darbeleri yokken diğer bir deyimle devrede yalnızca DC besleme varken mikroişlemci **"stand-by"** durumundadır. Yani bütün kaydediciler içeriğini korumaktadırlar ve her an işlem yapmaya hazırdırlar. Bu hâldeki akım sarfiyatı yalnızca 5 mA gibi çok düşük bir değerdedir. Ayrıca 80C286, maksimum çalışma frekansından minimum çalışma frekansına kadar, bütün fonksiyonlarını korumaktadır. Bu hâldeki güç sarfiyatı, yukarıda listesi verildiği gibi, frekans ile orantılıdır.

- ❖ Veri iletim yolu (D0 ... D15) 16 bit.
- ❖ Adres iletim yolu (A0 ... A23) 24 bit.
- ❖ Güç sarfıyatı 3.3 W
- ❖ Herhangi bir pin ile toprak arasındaki gerilim -1V ile +7V
- ❖ Çalışma sıcaklığı 0 °C ile 70 °C arasında.



Şekil 7.7 : Intel 80286'nın bacak bağlantıları

Tablo 7.2 : Intel 80286'nın bacak bağlantılarının açıklamaları

Bacak	Açıklama		
CLK	Sistem saati (system clock)		
D15, D10	Veri yolu (data bus): Bilgi girişi; hafıza ve I/O portlarını okuma periyodunda. Bilgi çıkışı: Hafıza ve I/O portlarını yazma periyodunda.		
A23... A0	Adres Yolu (address bus): Hafıza ve I/O portlarının seçimi sırasında çıkış yapar.		
BHE	İletim yolu üst yarısı işletimi (Bus High Enable): Veri yolunun D ₁₅ ...D ₈ üst yarısının kullanılmasını sağlar.		
	BHE	A ₀	Fonksiyon
	0	0	Kelime transferi
	0	1	Veri yolunun üst yarısından (D ₁₅ ...D ₈)
	1	0	Veri yolunun alt yarısından (D ₇ ...D ₀)
1	1	Byte transferi hiç bir zaman gerçekleşmez.	

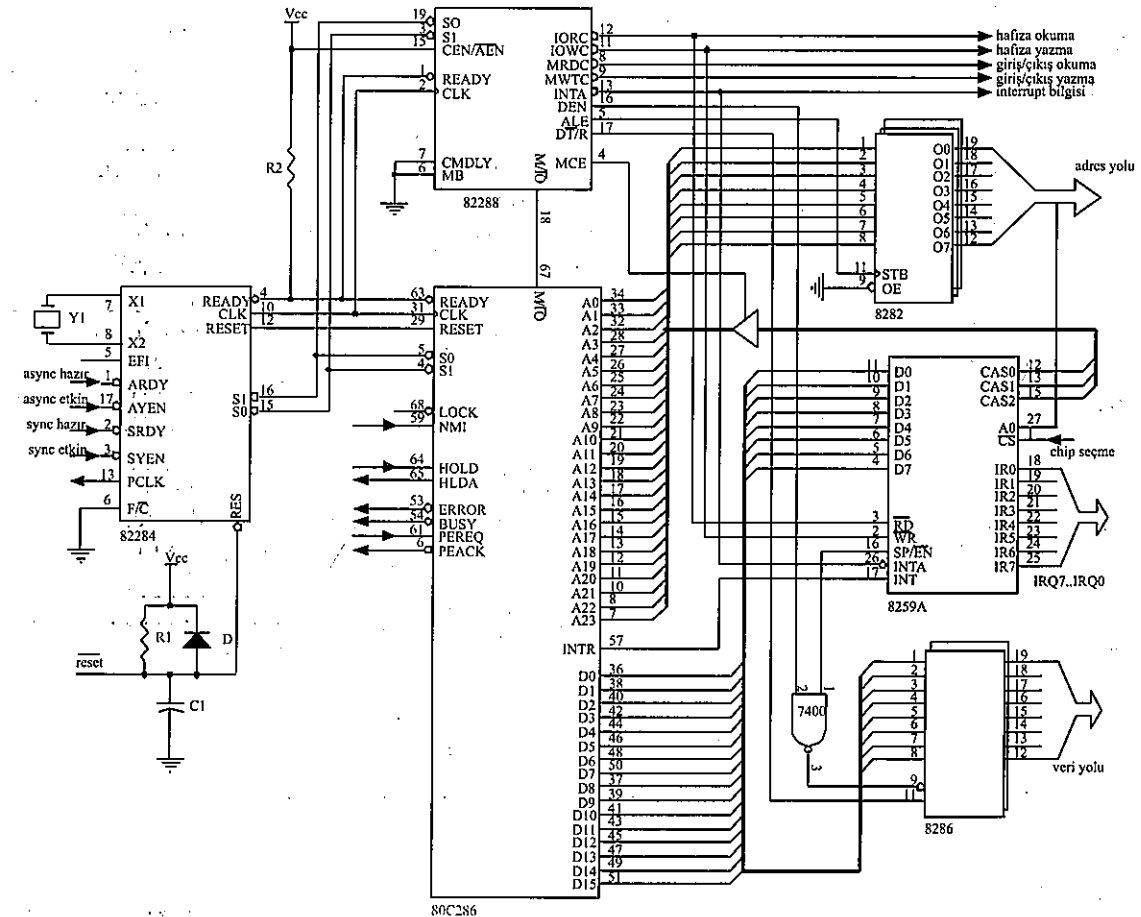
Tablo 7.2'nin devamı

Bacak	Açıklama																																																																																					
	Veri yolu durumu (bus cycle status): İletim yolu işleminin başlangıcını gösterir. İletim yolu statüsünü belirleyen bağlantı değerleri arasındaki ilişkiler bir tablo oluşturur:																																																																																					
	<table border="1"> <tr> <th>COP/ INTA</th><th>M/I/O</th><th>S1</th><th>S0</th><th>İletim yolu döngü başlangıcı (bus cycle initiated).</th></tr> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>Bilgi kesimi (interrupt acknowledge).</td></tr> <tr> <td>0</td><td>0</td><td>0</td><td>1</td><td>Gerçekleşmez (will not occur).</td></tr> <tr> <td>0</td><td>0</td><td>1</td><td>0</td><td>Gerçekleşmez (will not occur).</td></tr> <tr> <td>0</td><td>0</td><td>1</td><td>1</td><td>Durum belirtmez (none ; not a status cycle).</td></tr> <tr> <td>0</td><td>1</td><td>0</td><td>0</td><td>Eğer A1=1 ise duraklama (if A1=1 then halt;else shut down).</td></tr> <tr> <td>0</td><td>1</td><td>0</td><td>1</td><td>Bellekten veri okuma (memory data read).</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>0</td><td>Belleğe veri yazma (memory data write).</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>1</td><td>Durum belirtmez (none ; not a status cycle).</td></tr> <tr> <td>1</td><td>0</td><td>0</td><td>0</td><td>Gerçekleşmez (will not occur).</td></tr> <tr> <td>1</td><td>0</td><td>0</td><td>1</td><td>Giriş/çıkış okuma (I/O read).</td></tr> <tr> <td>1</td><td>0</td><td>1</td><td>0</td><td>Giriş/çıkış yazma (I/O write)</td></tr> <tr> <td>1</td><td>0</td><td>1</td><td>1</td><td>Durum belirtmez (none ; not a status cycle).</td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>0</td><td>Gerçekleşmez (will not occur).</td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>1</td><td>Bellek komut okuma (memory instruction read).</td></tr> <tr> <td>1</td><td>1</td><td>1</td><td>0</td><td>Gerçekleşmez (will not occur).</td></tr> <tr> <td>1</td><td>1</td><td>1</td><td>1</td><td>Durum belirtmez (none ; not a status cycle).</td></tr> </table>	COP/ INTA	M/I/O	S1	S0	İletim yolu döngü başlangıcı (bus cycle initiated).	0	0	0	0	Bilgi kesimi (interrupt acknowledge).	0	0	0	1	Gerçekleşmez (will not occur).	0	0	1	0	Gerçekleşmez (will not occur).	0	0	1	1	Durum belirtmez (none ; not a status cycle).	0	1	0	0	Eğer A1=1 ise duraklama (if A1=1 then halt;else shut down).	0	1	0	1	Bellekten veri okuma (memory data read).	0	1	1	0	Belleğe veri yazma (memory data write).	0	1	1	1	Durum belirtmez (none ; not a status cycle).	1	0	0	0	Gerçekleşmez (will not occur).	1	0	0	1	Giriş/çıkış okuma (I/O read).	1	0	1	0	Giriş/çıkış yazma (I/O write)	1	0	1	1	Durum belirtmez (none ; not a status cycle).	1	1	0	0	Gerçekleşmez (will not occur).	1	1	0	1	Bellek komut okuma (memory instruction read).	1	1	1	0	Gerçekleşmez (will not occur).	1	1	1	1	Durum belirtmez (none ; not a status cycle).
COP/ INTA	M/I/O	S1	S0	İletim yolu döngü başlangıcı (bus cycle initiated).																																																																																		
0	0	0	0	Bilgi kesimi (interrupt acknowledge).																																																																																		
0	0	0	1	Gerçekleşmez (will not occur).																																																																																		
0	0	1	0	Gerçekleşmez (will not occur).																																																																																		
0	0	1	1	Durum belirtmez (none ; not a status cycle).																																																																																		
0	1	0	0	Eğer A1=1 ise duraklama (if A1=1 then halt;else shut down).																																																																																		
0	1	0	1	Bellekten veri okuma (memory data read).																																																																																		
0	1	1	0	Belleğe veri yazma (memory data write).																																																																																		
0	1	1	1	Durum belirtmez (none ; not a status cycle).																																																																																		
1	0	0	0	Gerçekleşmez (will not occur).																																																																																		
1	0	0	1	Giriş/çıkış okuma (I/O read).																																																																																		
1	0	1	0	Giriş/çıkış yazma (I/O write)																																																																																		
1	0	1	1	Durum belirtmez (none ; not a status cycle).																																																																																		
1	1	0	0	Gerçekleşmez (will not occur).																																																																																		
1	1	0	1	Bellek komut okuma (memory instruction read).																																																																																		
1	1	1	0	Gerçekleşmez (will not occur).																																																																																		
1	1	1	1	Durum belirtmez (none ; not a status cycle).																																																																																		
S1, S0																																																																																						
M/I/O	Hafıza veya giriş/çıkış seçimi (memory I/O select): T5 periyodunca M/I/O çıkış "1" olursa hafıza döngüsü, "0" olursa I/O döngüsü vardır.																																																																																					
COD/ INTA	Kod/kesme bilgisi (code/interrupt acknowledge): Hafızadan komut alma döngüsünü, veri okuma döngüsünden ayırır. Etkin çıkışı "0" dır.																																																																																					
LOCK	İletim yolu kilitleme (bus lock): İletim yolu meşguliyetini kontrol eder. Her bir "Interrupt-acknowledge cycle" için "lock" aktivite gösterir. Etkin çıkışı "0"dır.																																																																																					
READY	İletim yolu hazırlama (bus ready): İletim yolu döngüsünü sonuçlandırmaktır, iletim yolu döngüsü, ready "0" tarafından sonuçlandırılıncaya kadar devam eder. Ready sinyali sistem saati ile senkronizedir.																																																																																					
HOLD HLDA	İletim yolunu tutma isteği (bus hold request)/Acknowledge (hold bilgisinin iletimi): Mikroişlemcinin yerel iletim yolunun sahipliğini kontrol ederler. Hold girişi, yerel iletim yolunun diğer bir kullanıcı tarafından kontrol edilmesine müsaade eder. Bu sırada hold girişi "0" dır. Hold "0" girişi HLDA çıkışını aktive eder ve durum saptaması yapılır. Bu durum Hold aktivitesine kadar devam eder.																																																																																					
INTR	Kesme istemi (Interrupt Request): Mikroişlemcinin INTR girişine gelen işaret, mikroişlemciye mevcut programın askıya alınmasını ve işlemcinin dışarıdan gelecek yeni bir isteğe hizmet vermesini sağlar. Bu durum, ilk kesme bilgisi (INTRA) döngüsü tamamlanıncaya kadar devam eder.																																																																																					
NMI	Maskelenemeyen kesme isteği (Non-Maskable Interrupt Request): Mikroişlemcinin, 2. INTRA döngüsü tamamlandığında, iç devrede gelişen vektör değeri ile kesime geçmesini sağlar. Bu arada NMI girişi "0" olur.																																																																																					
PERE PEACK	Ek işlemci için veri transferi isteği (Processor Extension Operand Request and Acknowledge): PEREQ girişi, mikroişlemciye ek işlemci (processor extension) için "data operand" transferi istemi yapar. PEACK çıkış işareti, ek işlemciye transfer işlemini belirler. Peack aktif "0" dır.																																																																																					
BUSY ERROR	Ek işlemci meşgul sinyali ve hata durumu (Processor Extension Busy and Error): Ek işlemcinin işletim şartlarını (operating condition) belirler (indicate).																																																																																					

Tablo 7.2'nin devamı

Bacak	Açıklama
RESET	Durdurma (system reset): Reset girişi aktif "1" yapıldığında, mikroişlemci aktivitesi sonuçlanır ve lojik sistemi temizlenir. Reset işlemi herhangi bir zamanda başlatılabilir ve 16 saat (clock) döngüsü (cycle) boyunca aktif kalır. Reset'in aktif hâli müddetince diğer PIN girişleri şöyle olacaktır: S0, S1, PEACK, A23...A0, BRE, LOCK: "1" M/I/O, COD/INTA, HLDA (hold "0" iken): "0" D15...D0: Üç durumlu portlar veri yolundan yalıtılmıştır. Reset girişi "1" den "0" a geçtiğinde çalışma tekrar başlar. Bu geçiş sistem saati ile senkronize olmalıdır. Reset'in "0" dan "1" e geçişi saat ile senkron da asenkron da olabilir. Senkron olmasının istendiği durum, saatin başka bir saat ile fark senkronizasyonu yapmasının gerekli bulunduğu hâllerdir.
Vss	Sistem toprağı (System Ground): 0V
Vcc	Sistem beslemesi (System Power): +5V
CAP	Entegre gövdenin filtre kondansatörü (substrate filter capacitor): 80286 mikroişlemcilerinde entegre gövde, CAP girişi üzerinden, 0.047µF ±20%, 12V'luk bir kondansatör ile toprağa bağlanmalıdır. Bu kondansatör 1µA'e kadar olan kaçak (leakage) akımları için filtre (süzme) görevi yapmaktadır.

7.3.2 80286 Mikroişlemcisinin Sistem Konfigürasyonu



Şekil 7.8 : Intel 80C286 mikroişlemcisinin sistem konfigürasyonu

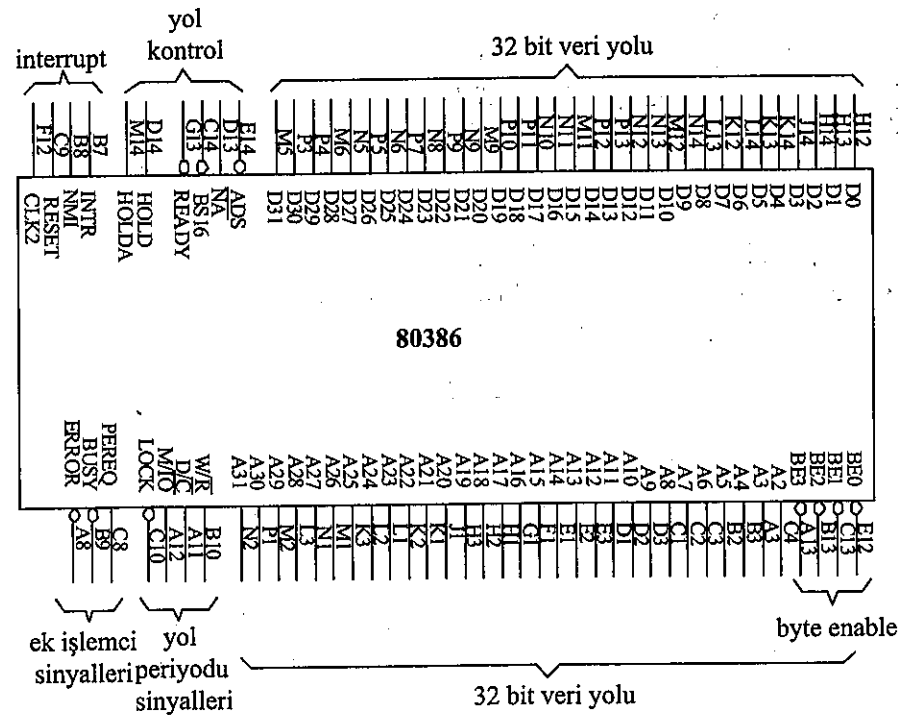
80286 mikroişlemcisinin tablo 7.2'de belirtilen bacak bağlantıları ile belirli işlemleri gerçekleştirmesi için de belirli devrelerden giriş almakta ve belirli devrelere de çıkış yapmaktadır. Bazı devrelerle de hem giriş hem çıkış bağlantısı vardır. Şekil 7.8' 80286'nın dış devre bağlantıları gösterilmiştir.

7.4 INTEL 80386 MİKROİŞLEMCİSİ

7.4.1 80386 Mikroişlemcisinin Teknik Özellikleri

- ❖ Saat frekansı: 20, 25, 33, 40 MHz.
- ❖ 132 adet bacak (pin) bağlantısı vardır.
- ❖ 387 DX matematik işlemciyle uyumludur.
- ❖ 0.8 mikron gelişmiş CMOS teknolojisi ile üretilmektedir.
- ❖ Adres ve veri iletim yolları: 32 bit.
- ❖ 4 Gb'lık fiziki belleği ve 64 Tb (tera byte) göreceli (virtual) belleği adresleyebilir.
- ❖ Depolama sıcaklığı: -65 °C ile +150 °C arası.
- ❖ Çalışır hâldeki gövde sıcaklığı: -65 °C ile +125 °C arası.
- ❖ Toprağa (Vss) göre besleme gerilimi: -0,5 V ile +7 V arası.
- ❖ Besleme gerilimi $V_{cc} = 5V \pm 5\%$
- ❖ Gövde sıcaklığı 0° C ile +85° C arası.

80386 DX mikroişlemcisinin, şekil 7.9'da bacak bağlantısı, şekil 7.10'da devre bağlantısının blok diyagramı verilmiştir.

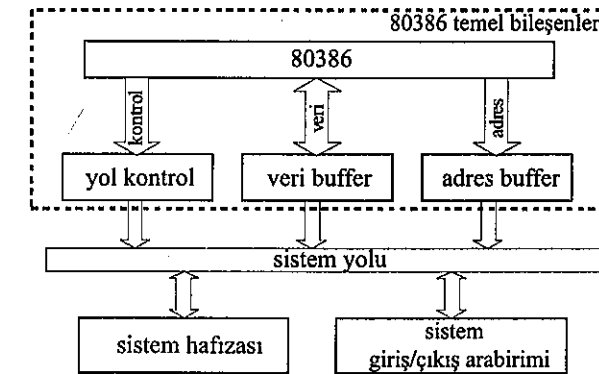


Şekil 7.9 : 80386'nın bacak bağlantıları

Tablo 7.3'de 80386 DX mikroişlemcisinin bacak bağlantılarının isimleri ve kısaca görevleri verilmiştir. 80386 bacak bağlantıları çoğunlukla 80286'larınkine benzerdir.

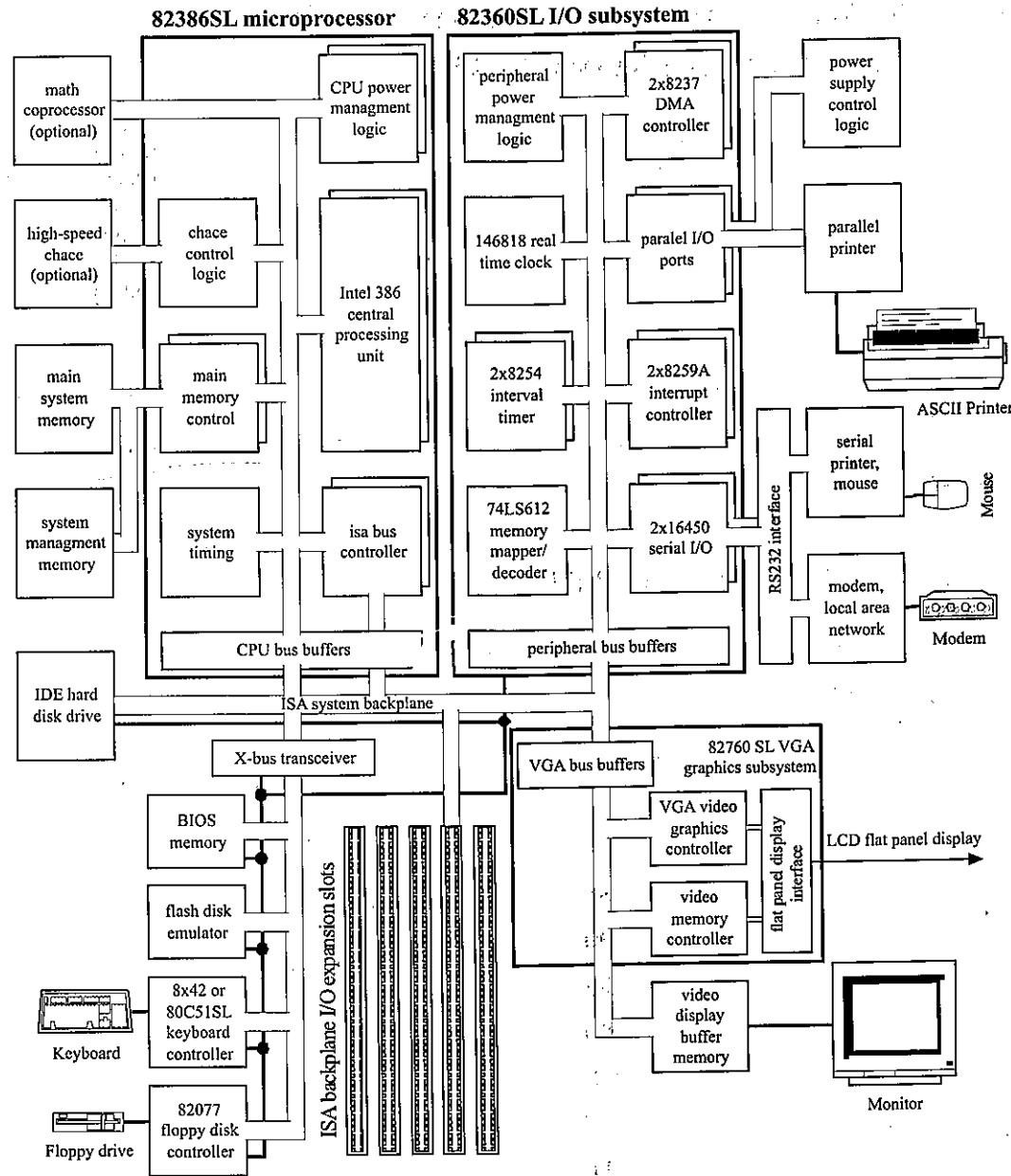
Tablo 7.3 : 80386'nın bacak bağlantıları ve açıklamaları

Bacak	Açıklama
A31...A2	Adres yolu (address bus) fiziki bellek veya giriş/çıkış portu adres çıkışları.
ADS	Adres durum tespiti (address status) Geçerli iletim yolu döngüsünü ve (W/R, D/C, M/I/O, BED/BE1 , BEF , BE3 ve A31...A2) etkinlikleriyle işlenecek adresi belirler.
BE3 ... BE0	Byte etkin (byte enable) "0" de etkindir. Veri yolundaki hangi veri byte'ının hangi döngü bölümünde bulunduğunu belirler.
BS16	İletim yolu genişliği: 16 veya 32 bit veri yolunun direkt bağlantısını sağlar.
BUSY	Mikroişlemci bağlantı devrelerinden herhangi birinin meşgul olduğunu bildirir.
CLK2	Saat zamanlayıcıdır.
D31...D0	Hafıza veya I/O okuma, döngüsü sırasında giriş, yazma döngüsünde çıkış yapar.
D/C	Veri işaretleri ile kontrol işaretlerini ayırır.
ERROR	İsteğe bağlı olarak verilen çıkıştır. İşlem hatalarını bildirir.
HLDA	İletişim yolunun meşgul olduğunu gösterir.
HOLD	Mikroişlemci çalışmasını sonuçlandırarak iletim yolunun diğer devreler tarafından kullanılmasını sağlar.
INTR	Kesme isteği
LOCK	İletişim yolunun diğer devreler karşı kilitli olup olmadığını belirler.
M/MIO	İletişim yolundaki verinin hafıza mı, giriş/çıkışa mı ait olduğunu belirler.
NA	Sıralı adresleme işlemek için kullanılır.
NC	Boş
NMI	Maskelenemez kesme isteği girişi
PEREQ	Ek işlemcide, mikroişlemci kanalıyla transfer yapılacak verilerin bulunduğunu belirler.
READY	İletişim yolu döngüsünü sonuçlandırır.
RESET	Mikroişlemciyi resetler.
Vcc	+5 V mikroişlemci beslemesidir.
Vss	Şase ucu
W/R	Yazma ve okuma sinyali çıkışı



Şekil 7.10 : 80386 mikroişlemcisinin blok diyagramı

80386 mikroişlemcili bir mikrobilgisayarın basitleştirilmiş blok diyagramı şekil 7.11'de görülmektedir.



Şekil 7.11 : I80386 mikroişlemcili bir bilgisayarın sistem şeması

DEĞERLENDİRME ÇALIŞMALARI

- 1) 8080 ve 8085 mikroişlemcilerinin temel çevre birimleri nelerdir?
- 2) 8080/8085 ile 8086/8088 mikroişlemcileri arasındaki farklar nelerdir?
- 3) 8086 ve 8088 mikroişlemcileri kaç bitlik veri yollarına sahiptirler?
- 4) 8086 ve 8088 mikroişlemcilerinin kaç türlü çalışma modları vardır?
- 5) 80286 ve 80386 mikroişlemcileri kaç bitliktir?
- 6) 80286 ve 80386 mikroişlemcilerinin adresleme kapasiteleri ne kadardır?

ÜNİTE 8 MİKROİŞLEMCİ KOMUTLARI

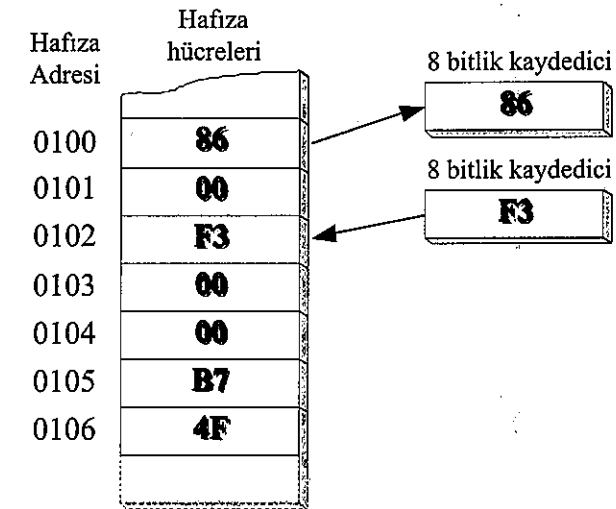
8.1 MİKROİŞLEMCİ KOMUTLARI

Diğer elektronik devre elemanlarından farklı olarak mikroişlemciler, yapılmak istenen iş için programlanması gerekir. Mikroişlemciyi programlamak demek o işlemcinin tanıyıp işleyebildiği bazı kodları belli bir algoritma çerçevesinde mikrobilgisayarın hafızasına yüklemek demektir. Mikroişlemci üreten bir çok firma vardır (Intel, AMD, Mos Tec, Motorola, Zilog, Cyrix...). Her firma kendi ürettikleri mikroişlemcilerin tanıyabileceği komut kodlarını ve diğer teknik kapasitelerini bir dokümanla programcılara sunarlar. Programcı da bu dokümanlardan faydalanarak sistemde kullanılan mikroişlemcinin, tanıyıp işleyebileceği kodlardan oluşan programı mikrobilgisayarın hafızasına kaydeder. Özellikle assembly seviyesinde program yazan programcıların kullandıkları mikroişlemcinin iç yapısını ve komut setini çok iyi bilmeleri gerekir. Komut seti mikroişlemcinin tanıyıp işlediği kodların tümüne denir.

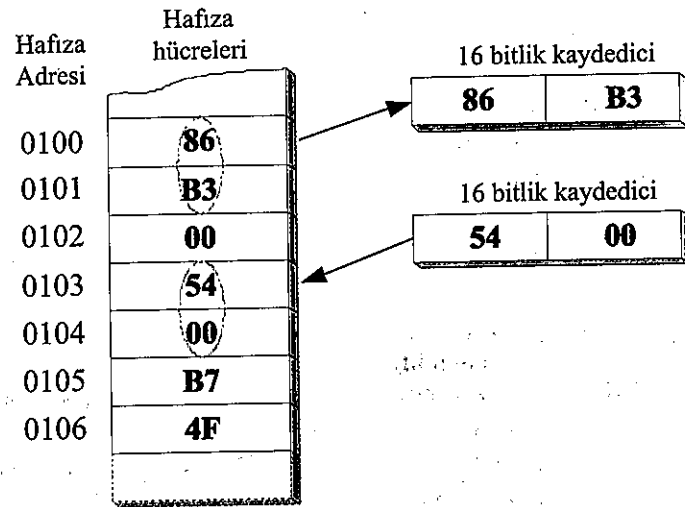
Mikroişlemcilerin markası ve kullandığı komutlar ne olursa olsun yaptığı işler bakımından komutları belli gruplara ayırmak mümkündür. Örneğin bir mikroişlemci komutunun yaptığı işin aynısını başka bir mikroişlemcide farklı bir komutla yaptırmak mümkündür. Bir programcı için önemli olan mikrobilgisayarla yapılması düşünülen bir işin algoritmasını doğru kurmaktır. Yapılması düşünülen iş için uygun mikroişlemciyi ve o mikroişlemciye ait komutları teknik dokümanlardan araştırılabilir.

8.2 YÜKLEME KOMUTLARI

Yükleme komutları, kaydedicilere veya hafıza bölgelerine veri aktarmada kullanılan komutların tümüdür. Yüklem komutları ile bir verinin kaydediciden kaydediciye, kaydediciden hafıza hücresine, hafıza hücresinden kaydediciye doğru yüklemesi yapılabilir. Bu işlemlerin genel şekli şöyledir:

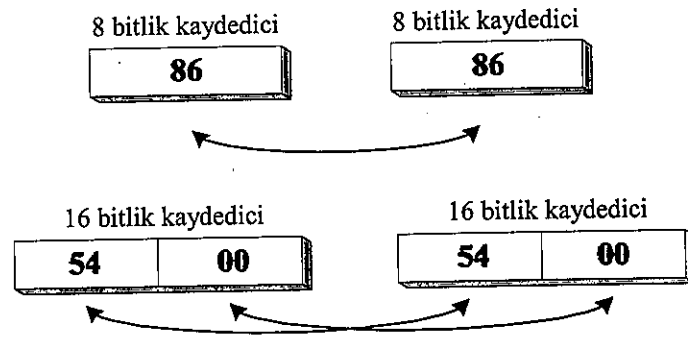


Şekil 8.1 : 8 bitlik kaydedici ve hafıza hücreleri arasındaki yüklemeler



Şekil 8.2 : 16 bitlik kaydedicilerle 8 bitlik hafıza hücreleri arasındaki yüklemeler

Yükleme komutlarıyla 16 bitlik kaydedicilere yapılan yüklemeler için hafızadan iki byte art arda alınır. Aynı şekilde 16 bitlik kaydedicilerden hafızaya yapılan bir yükleme sırasında da veriler hafıza hücrelerine art arda yazılır (şekil 8.2).



Şekil 8.3 : 8 bitlik ve 16 bitlik kaydediciler arasındaki yüklemeler

16 bitlik kaydediciler arasındaki yüklemelerde (takas işlemleri sırasında) düşük byte düşük byte'a, yüksek byte yüksek byte'a karşılık gelecek şekilde yükleme yapılır.



Mikroişlemci Komutları:

Mikroişlemcilerin komutlarında bir standartlaşmaya gidilmediği için her mikroişlemcinin komutları farklıdır. Fakat çoğu mikroişlemcide aynı işi yapan komutlar (komut isimleri ve kodları farklı olsa da) ortak kategoride toplanabilirler. Aşağıda değişik mikroişlemcilerin komutları yaptıkları iş bakımından ortak kategorilerde toplanarak anlatılma yoluna gidilmiştir. Aşağıdaki yükleme komutlarına değişik mikroişlemcilerden örnekler verilmiştir.

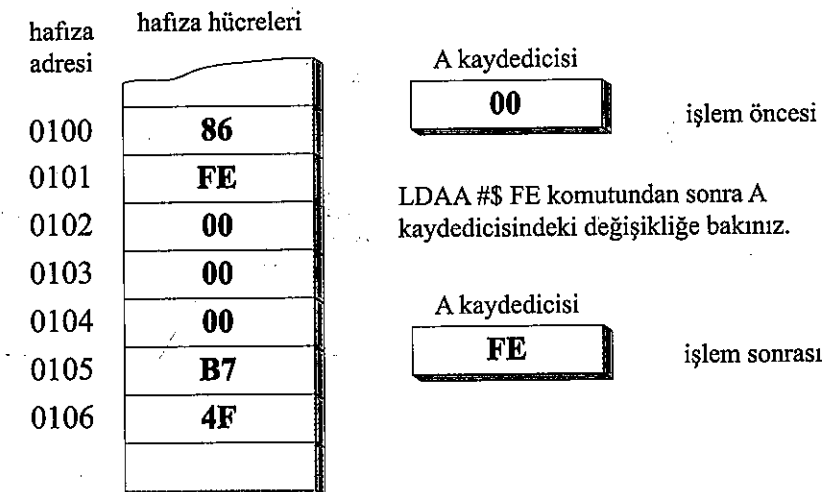
8.2.1 8 Bitlik Yükleme Komutları

8 bitlik verileri yine 8 bit büyüklüğündeki kaydedicilere veya hafıza hücrelerine yüklemek için bu tür yükleme komutları kullanılır. Bu nedenle yüklenecek verinin de yükleme yapılacak yerin de 8 bit (1 byte) büyüklüğünde olması gerekir. Tablo 8.1'de MC6800 mikroişlemci ailesinin komut setinde bulunan 8 bitlik yükleme komutlardan örnekler verilmiştir.

Tablo 8.1 : MC6802 mikroişlemcisinin komut setinde bulunan 8 bitlik yükleme komutlarından bazıları

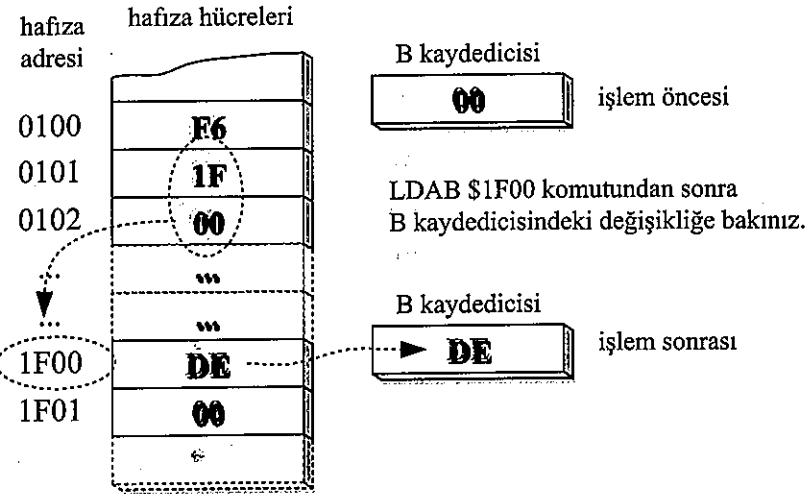
Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	86 FE		LDAA #\$FE	A kaydedicisine (FE) ₁₆ sayısını yükler
0102	C6 3F		LDAB #\$3F	B kaydedicisine (3F) ₁₆ sayısını yükler.
0104	F6 1F 00		LDAB \$1F00	B'ye (1F00) ₁₆ adresindeki veriyi yükler.
0107	A6 F0		LDAA \$F0,X	X[]+F0 adresindeki veriyi A'ya yükler
0109	97 00		STAA \$00	A'yı (00) ₁₆ adresine yükler.
010B	E7 00		STAB \$00,X	B'yi X[]+00 adresine yükle
010D	F7 0F 05		STAB \$0F05	B'yi (0F05) ₁₆ adresine yükle
0110	36		PSHA	A'daki verinin aynısını yığma da yükler
0111	33		PULB	Yığındaki son veriyi B ye yükler.

LDAA #\$FE komutunun işleyişi: A kaydedicisine (FE)₁₆ sayısını direkt yüklemek için LDA komutunun işlem kodu olan "86" kullanılmalıdır. (0100)₁₆ adresinde bulunan "86" komut kodu, kendisinden sonraki veriyi A kaydedicisine yükler. Şekil 8.4'de komutunun işleyişi görülmektedir.



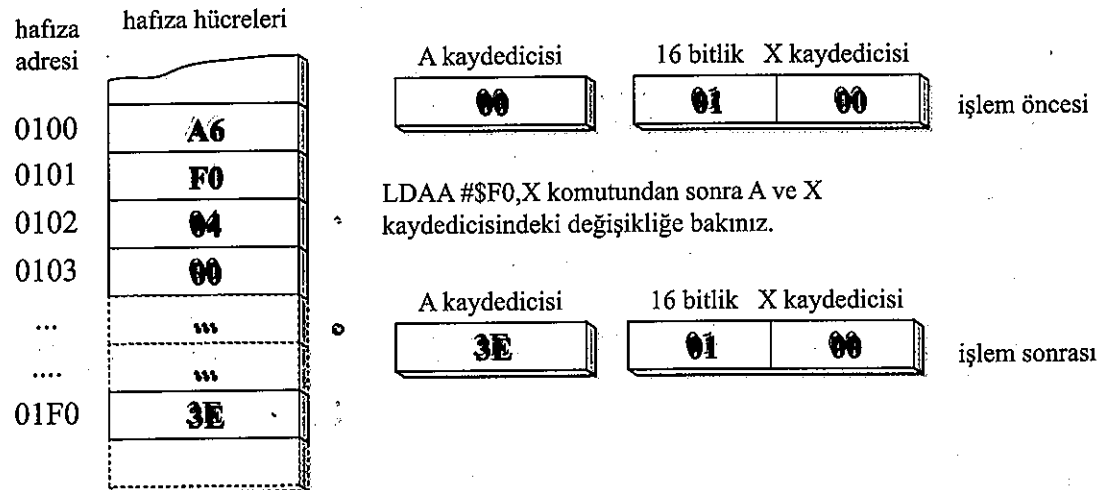
Şekil 8.4 : LDAA #\$FE komutunun işleyişi

LDAB \$1F00 komutunun işleyişi: LDAB \$1F00 komutu ile (1F00)₁₆ adresinde bulunan (DE)₁₆ verisi B kaydedicisine yüklenir. LDAB \$1F00 komutun işlem kodu "F6 1F 00" tür. Komutun işlem kodundan (FE)₁₆ sonra art arda gelen iki byte veri (1F 00)₁₆ B kaydedicisine yüklenecek verinin bulunduğu adresi gösterir.



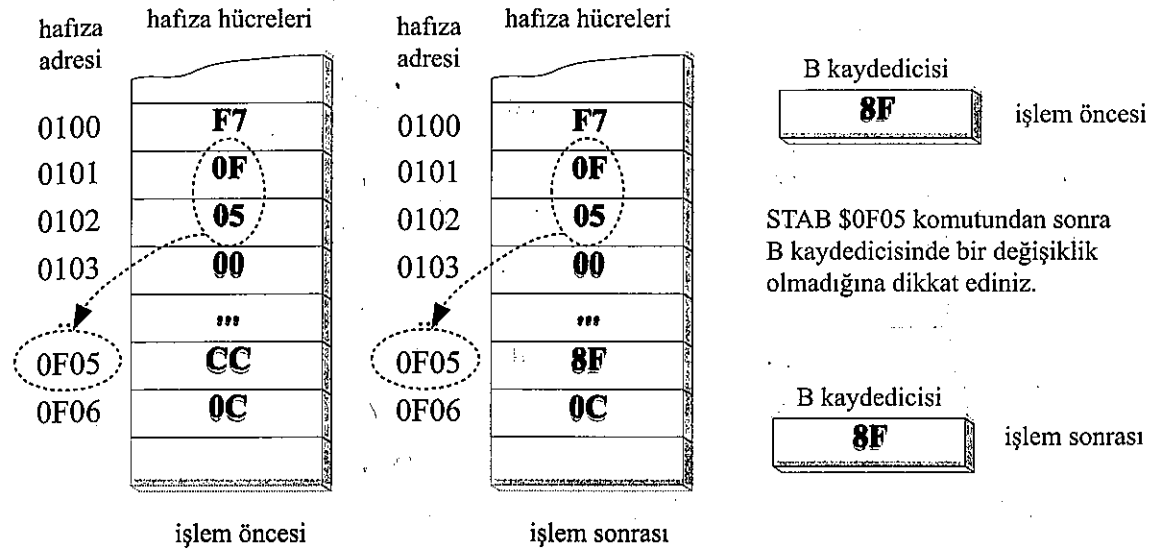
Şekil 8.5 : LDAB \$1F00 komutunun işleyişi

LDAA \$F0,X komutunun işleyişi: LDAA \$F0,X komutunun işletilmeden önce indeks kaydedicisindeki değere (0100)₁₆, komuttaki (F0)₁₆ sayısı eklenerek geçerli adres elde edilir. Elde edilen bu geçici adresteki (01F0)₁₆ veri A kaydedicisine yüklenir. İndeks kaydedicisi bu işlemten etkilenmez, içeriğini korur yani buradaki işlemden sonra indeks kaydedicisindeki (0100)₁₆ olarak kalır. Program komutu (0100)₁₆ adresinden itibaren 2 byte'tır. "LDAA \$F0,X"ın iş kodu "A6 F0"dır. "A6" makine kodu, "F0" ise işlenendir. Şekil 8.6'da işlemlerin hafıza ve kaydedicilerdeki durumu görülmektedir.



Şekil 8.6 : LDAA #\$F0,X komutunun işleyişi

STAB \$0F05 komutunun işleyişi: Bu komut ile B kaydedicisindeki veriyi, istediğimiz bir hafıza hücresine yükleyebiliriz. STAB \$0F05 komutu, B kaydedicisinde bulunan (8F)₁₆ verisini (0F05)₁₆ adresinde saklar. Bu yüklemekten sonra B kaydedicisinin içeriğinin değişmediğine dikkat ediniz. (0100)₁₆ adresinden itibaren üç byte yer kaplayan STAB \$0F05 komutunun işlem kodu "F7 0F 05" şeklindedir. Komutun hafıza haritasındaki yerleşimini ve işlemin sonucunu şekil 8.7'de inceleyiniz.



Şekil 8.7 : STAB \$0F05 komutunun işleyişi

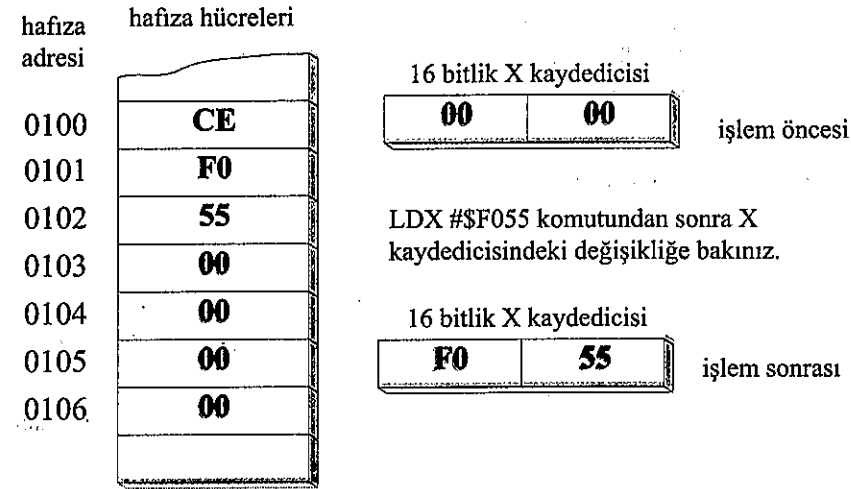
8.2.2 16 Bitlik Yükleme Komutları

16 bitlik sayıların kaydedicilere veya hafıza hücrelerine yüklenmesi için kullanılan assembler komutları farklı olmasına karşın işlevleri 8 bitlik yükleme komutları ile aynıdır. MC6802'de bulunan, yığın kaydedicisi (SP;stack pointer), indeks kaydedicisi (IX;index register) ve program sayıcı (PC;program counter) gibi 16 bitlik kaydedicilerde yükleme işlemi için kullanılan bazı komutları inceleyelim.

Tablo 8.2 : MC6802 işlemcisinin komut setinde bulunan 16 bitlik yükleme konutlarından bazıları

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	CE F0 55		LDX #\$F055	X kaydedicisine (F055) ₁₆ sayısını yükler.
0103	FE F0 00		LDX \$F000	X'e (F000) ₁₆ adresinden ardışık iki byte'ı yükler
0106	FF E0 7F		STX \$E07F	X'i (E07F) ₁₆ adresine art arda yazar.
0109	EF 10		STX \$10,X	Kendi içerdiği değerden 10 byte ileriye kendisini yazar. X[]+10 adresine X'i yaz
010B	8E 00 7F		LDS #\$007F	SP'ye (007F) ₁₆ 'yi yükleyerek yığın başlangıcını (007F) ₁₆ 'den itibaren kullanıma hazırlar.

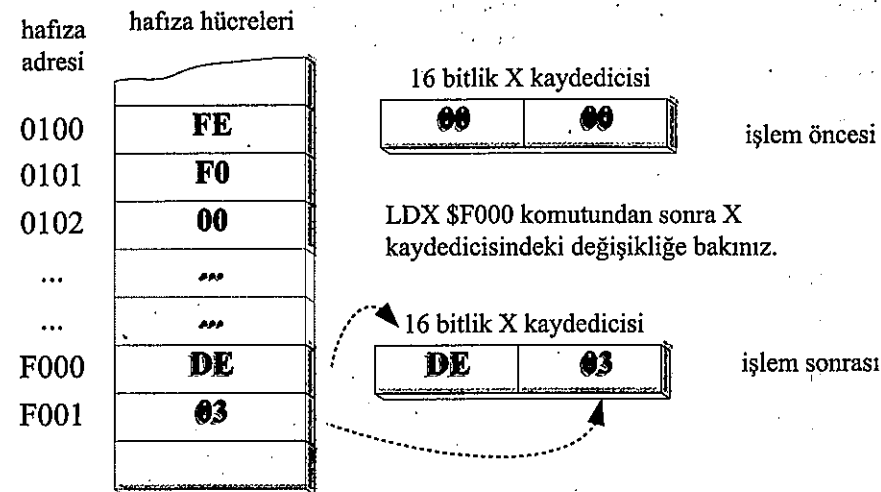
LDX #\$F055 komutunun işleyişi: LDX komutu ile X kaydedicisine 16 bitlik veri yüklemesi yapılabilir. Aynı zamanda ardışık iki hafıza hücresinde bulunan 8 bitlik veriler birleştirilerek 16 bitlik bir verinin yüklemesi de yapılır. İlk adreste bulunan 8 bitlik veri X kaydedicisinin düşük byte'ına (D0..D7), hemen sonraki adreste bulunan veri ise X kaydedicisinin yüksek byte'ına (D8..D15) yüklenir.



Şekil 8.8 : LDX #\$F055 komutunun işleyişi

LDX #\$F055 komutu ile indeks kaydedicisine 16 bitlik (F055)₁₆ sayısı yüklenmiş olur. LDX #\$F055 komutunun işlem kodu "CE F0 55" tir. (CE)₁₆ komutun kodu, (F055)₁₆ ise X'e yüklenecek veridir.

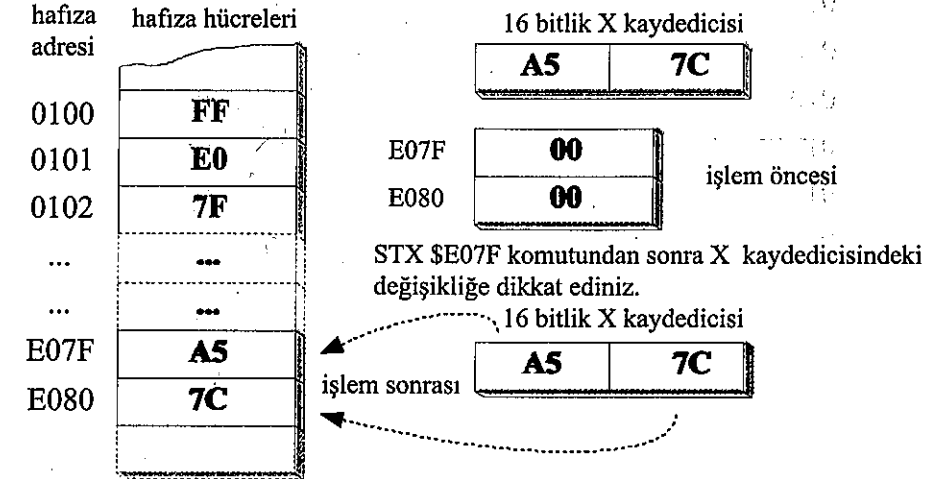
LDX \$F000 komutunun işleyişi: LDX \$F000 komutu ile (F000)₁₆ adresinden itibaren iki byte veri indeks kaydedicisine yüklenir. Yani bu komutla indeks kaydedicisine yüklenecek veri, komuttan sonra belirtilen adresten alınır. Belirtilen adresteki ilk veri indeks kaydedicisinin yüksek byte'ı, ikinci veri ise düşük byte'ı olarak yüklenir.



Şekil 8.9 : LDX \$F000 komutunun işleyişi

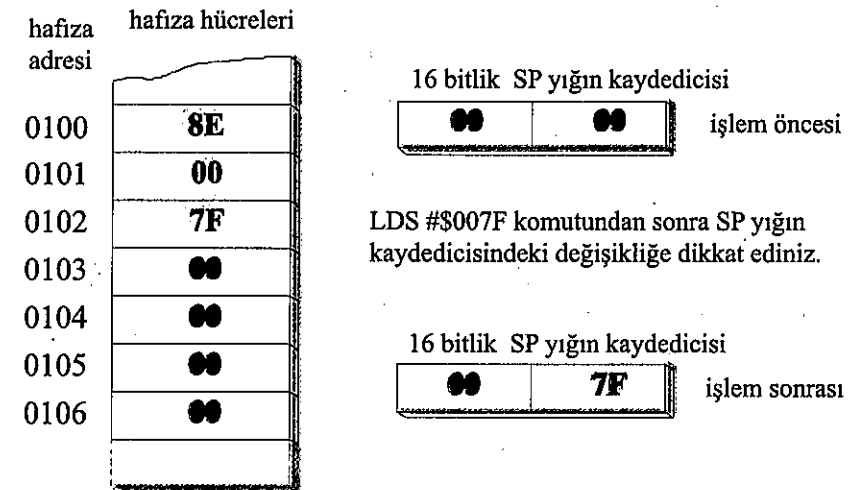
LDX \$F000 komutu ile indeks kaydedicisine (F000)₁₆ ve (F001)₁₆ adreslerindeki veriler indeks kaydedicisine yüklenmiş olur. LDX \$F000 komutunun işlem kodu "FE F0 00" tir.

STX \$E07F komutunun işlevi: STX komutu, indeks kaydedicisinde bulunan 16 bitlik veriyi ardışık iki hafıza hücresine yükler. İndeks kaydedicisinin düşük byte'ı (D0...D7) ilk adrese, yüksek byte'ı (D8...D15) bir sonraki adrese yüklenir. STX \$E07F komutu ile yapılan işlemde, X kaydedicisindeki "A57C" verisi (E07F)₁₆ ve (E080)₁₆ adresli hücelere art arda yerleşir.



Şekil 8.10 : STX \$E07F komutunun işleyişi

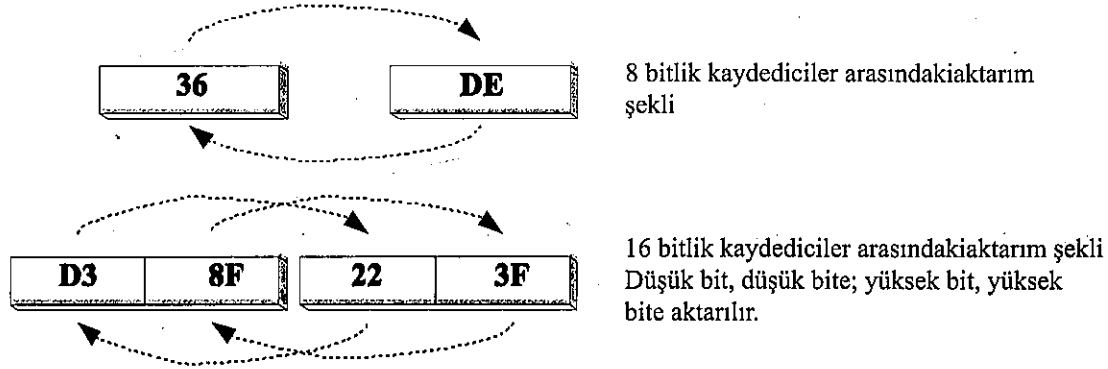
LDS #\$007F komutunun işleyişi: LDS komutu yığın kaydedicisi SP'ye, yığın mekanizmasının başlangıç değerinin yüklenmesinde kullanılır. LDS #\$007F komutu ile yığın kaydedicisine, yığının başlangıç konumu olarak (007F)₁₆ adresi yüklenir. Bu komutun işlem kodu "8E 00 7F" dir. Bu komut işlendikten sonra artık yığına veri aktardığımızda, aktardığımız veri bu adresin bulunduğu hafıza hücresinden itibaren yüklenecektir. Ayrıca yığına aktardığımız (PSH komutu ile) her bir byte'lık veriye karşılık yığın kaydedicisi "1" eksilir, yığından her veri çektiğimizde (PUL komutu ile) ise yığın kaydedicisi "1" artar. Yığının bu tip çalışması LIFO (Last In First Out; son giren ilk çıkar) kuralına göre çalışan yığın düzenlerinde görülür. Mikroişlemcilerde yığın konusu ve yığın kaydedicisinin nasıl çalıştığı ünite 4'te anlatılmıştır.



Şekil 8.11 : STX #\$007F komutunun işleyişi

8.2.3 Takas Komutları

Takas veya yer değiştirme komutları, kaydedicilerde bulunan verilerin karşılıklı olarak yer değiştirilmesinde kullanılır. Bazı komutlarda ise veri tek yönlü değişir. Bu işlem birçok mikroişlemcide bir byte'lık komutlarla yapılmaktadır. Takas için kullanılan komutta, takas edilecek kaydedicilerin hangileri olacağı komut kodunun kendisinde saklıdır, ayrıca iş kodundan sonra takas yapılacak kaydedicileri belirtmeye gerek yoktur. Aşağıda bu işlemlerin şematik gösterimi (şekil 8.12) ve bu işlem için kullanılan komutlara (tablo 8.3) örnekler verilmiştir.

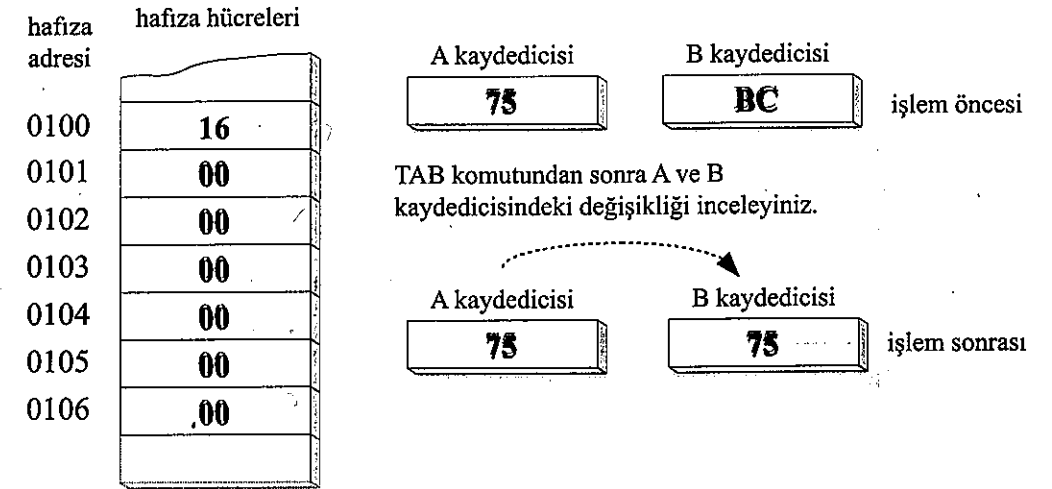


Şekil 8.12 : Takas komutunun kaydedicilerde yaptığı değişiklik

Tablo 8.3 : Takas komutlarından bazıları

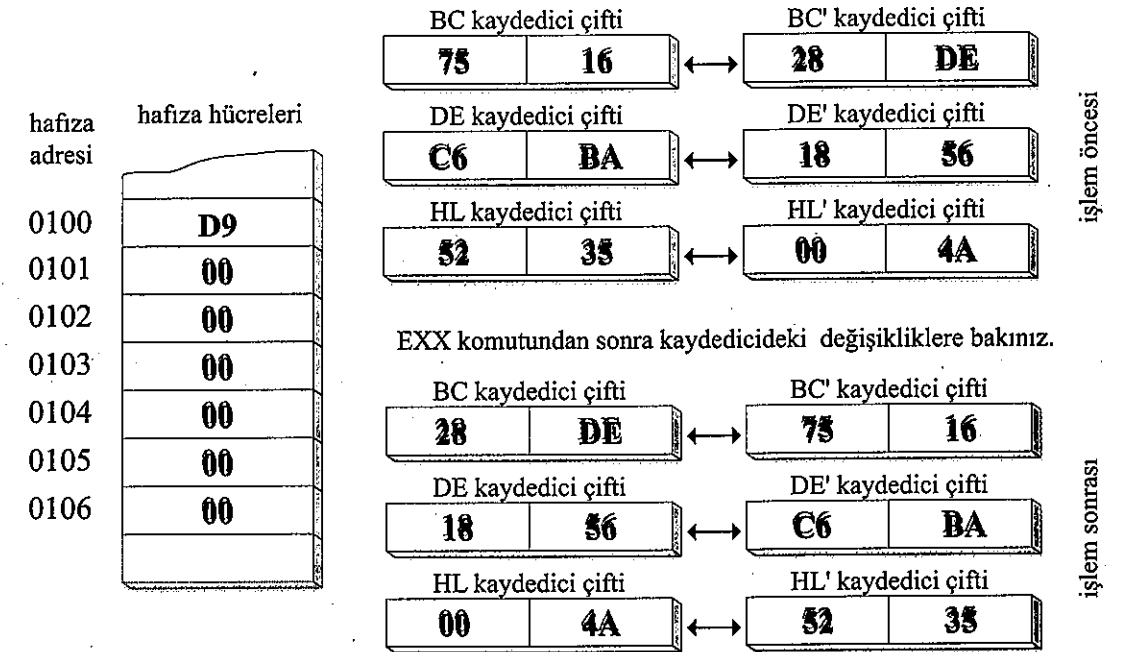
Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	EB		EX DE,HL	(Z80) HL ile DE kaydedicileri içerikleri yer değiştirir.
0101	D9		EXX	(Z80) Asıl kaydedicilerle yedek kaydediciler arasında yer değiştirme.
0102	EB		XCHG	(Intel I8080) HL ve DE kaydedici içeriklerini karşılıklı değiştirir. DE ↔ HL
0103	16		TAB	(MC6802) A kaydedicisindeki veriyi B kaydedicisine aktarır.

TAB komutunun işleyişi: MC6800 mikroişlemci ailesinin komut setinde bulunan bu komut ile A kaydedicisinin içeriği B kaydedicisine transfer edilir. İşlemden sonra B kaydedicisinde bulunan önceki değer kaybolur. Takas işlemi için yani A, B kaydedicilerinin içeriklerini yer değiştirmek için B kaydedicisinin içerdiği önce yığına atılır. Daha sonra TAB komutu işletilerek A'nın içeriği B'ye kopyalanır. B kaydedicisinin içeriği yığından çekilerek A'ya yüklenir. Böylece A, B kaydedicileri arasındaki yer değişikliği gerçekleşir. MC6800 mikroişlemci ailesinde bu işi tek komutla gerçekleştirmek mümkün değildir çünkü bu işlemcinin komut setinde böyle bir komut yoktur.



Şekil 8.13 : TAB komutunun A ve B işleyişi

EXX komutunun işleyişi: Z80 işlemcilerinin komut setinde bulunur. Bu komut ile Z80 ana kaydedicileri ile yedek kaydedicileri arasında bilgi değiş tokuşu gerçekleşir.



Şekil 8.14 : Z80'in EXX komutundan sonra ana ve yedek kaydedicilerindeki veriler yer değiştirir

8.2.4 Blok Transfer Komutları

Blok transfer komutları, hafızada sıralı şekilde bulunan bir grup verinin başka bir hafıza bölgesine transferini gerçekleştirmek için kullanılır. Blok transfer komutları bulunan mikroişlemcilerde, bu komutlar genellikle en az üç adet kaydedici ile birlikte kullanılır. Bu kaydedicilerin ikisi adres işaretleyicisi olarak biri de sayıcı olarak kullanılır. Bu kaydedicilerden biri taşınacak sıralı verinin başlangıç adresini tutar, diğer adres kaydedicisi sıralı verinin taşınacağı hafıza bölgesinin başlangıç adres bilgisini tutar. Sayıcı olarak kullanılan diğer kaydedici ise transfer edilecek verinin uzunluk bilgisini tutar.

Blok transfer komutlarını kullanmadan önce doğal olarak adres işaretleyicilerini, önceden istenen adres değerleri ile yüklemek gerekmektedir. Blok transfer komutlarından önce adres işaretleyici kaydedicilerin doğru değerlerle yüklenmesi programcının sorumluluğundadır. Ancak adres işaretleyicilerini ve sayıcıyı gerekli değerleri yükleyecek komutları kullandıktan sonra blok transfer komutlarını kullanmak, programda bir mantık ve işlem hatasına düşülmesine engel olur.

Tablo 8.4 : Z80 mikroişlemcisi komut setinde bulunan blok transfer komutlarından bazıları

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	ED A0		LDI	¹ (Zilog Z80) HL'de gösterilen adreste bulunan veri DE'nin gösterdiği adrese kopyalandıktan sonra DE ve HL adres işaretçileri bir artar. BC sayıcı bir eksilir.
0102	ED B0		LDIR	² BC=0 olana kadar HL'de gösterilen adreste bulunan veri DE'nin gösterdiği adrese kopyalandıktan sonra DE ve HL adres işaretçileri bir artar BC bir eksilir.

Blok transfer komutları, her mikroişlemci komut setinde bulunmayabilir. Bu tip bir işlevi gerçekleştirmek için kullanılan mikroişlemcinin komut setinde mutlaka blok transfer işlevini gerçekleştirecek komutların bulunması gerekmektedir. Örnek vermek gerekirse Zilog Z80 mikroişlemci ailesinin komut setinde blok transfer komutları bulunmakla birlikte, Intel 8080 ve Motorola MC6800 işlemci ailelerinin komut setinde bu tipten bir işlemi gerçekleştirmeye yarayacak komutlar bulunmamaktadır. Kullanılan mikroişlemcinin komut setinde bu tipte işlevleri gerçekleştirecek komutların bulunup bulunmadığını komut setine bakarak anlayabilirsiniz.

LDIR blok transfer komutunun işleyişi: LDIR komutu ile yapılmak istenen (0100)₁₆ adresinden başlayan 5 byte'lık sıralı bir veriyi, hafızanın (0200)₁₆ adresinden başlayarak taşıma (transfer etmek) işlemidir. Bunun için önce LDIR komutunun otomatik olarak kullandığı ve içerisinde sıralı verinin başlangıç adresinin bulunması gerekliliğinden dolayı önce HL kaydedicisini sıralı verinin başlangıç adresi olan (0100)₁₆ adres bilgisi ile yüklenmiştir. Bu adresi HL'ye yüklemek için "LD HL, 0100H" komutu kullanılabilir. Yine aynı şekilde transferin yapılacağı hafıza bölgesinin başlangıç adresi de DE kaydedicisine "LD DE, 0200H" komutu ile yüklenir. Toplam 5 byte transfer edileceği için BC kaydedicisine "LD BC, 0004H" komutu ile yükleme yapılır. Ancak başlangıç değerleri yüklendikten sonra LDIR komutu kullanılarak doğru transfer gerçekleştirilebilir. Tablo 8.5'de Z80 komut setinden faydalanarak hazırlanmış örnek bir program bulunmaktadır.

Tablo 8.5 : LDIR komutunun örnek bir programda kullanılması

Hafıza adresi	İş kodu	Assembly komutu	Açıklama
0000	21 00 01	LD HL,0100H	HL'de kaynak verinin başlangıç adresi yüklenir.
0003	11 00 02	LD DE,0200H	DE'ye kopyalanacak yerin başlangıç adresi yüklenir.
0006	01 04 00	LD BC,0004H	BC'ye transfer edilecek verilerin sayısı yüklenir.
0009	ED B0	LDIR	Komut HL'nin gösterilen yerin içeriğini, DE'de gösterilen adrese yükleme yaptıktan sonra HL ve DE kaydedicilerini birer arttırır ve BC bir eksiltir. BC sıfır oluncaya kadar kopyalama devam eder.

^{1, 2} Bu komutların doğru işletilmesi için komuttan önce hedeflenen hafıza yeri DE kaydedicisine, kaynak olan hafıza bölgesinin başlangıç adresi de HL kaydedicisine doğru değerlerle yüklemek gerekir. Ayrıca kullanılacaksa BC sayıcı da gereken değerle yüklenmelidir.

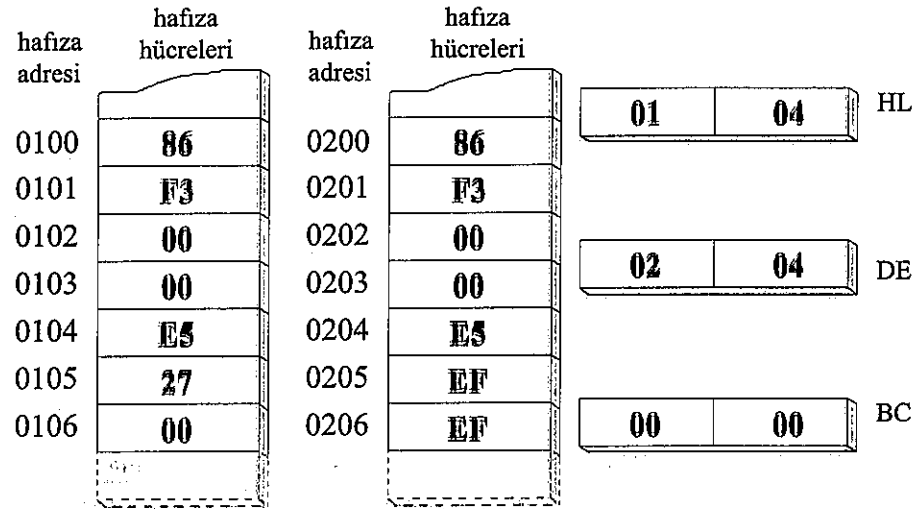
hafıza adresi	hafıza hücreleri
0000	21
0001	00
0002	01
0003	11
0004	00
0005	02
0006	01
0007	04
0008	00
0009	ED
000A	B0
000B	...
000C	...
000D	...

Şekil 8.15 : Tablo 8.5'deki programın hafızadaki yerleşimi

Şekil 8.16'da LDIR komutu yürütülmeden önce istenen değerlerle yüklenmiş kaydedicilerin ve hafıza bölgelerinin içerikleri görülmektedir. LDIR komutu işletildikten sonra (şekil 8.17) BC kaydedicisinin içeriği sıfırlanmıştır. HL ve DE kaydedicileri en son yükleme yapılan adres değerinde kalmıştır. Bu işlem sonucunda hafıza haritasında görüldüğü gibi (0100)₁₆ hafıza hücresinden başlayarak ilk beş byte hafızanın (0200)₁₆ adresinden itibaren yüklenmiştir.

hafıza adresi	hafıza hücreleri	hafıza adresi	hafıza hücreleri	HL kaydedicisine sıralı verinin başlangıç adresi yüklenir. (kaynak adresi)
0100	86	0200	EF	01 00 HL
0101	F3	0201	EF	
0102	00	0202	EF	
0103	00	0203	EF	02 00 DE
0104	E5	0204	EF	
0105	27	0205	EF	
0106	00	0206	EF	00 04 BC

Şekil 8.16 : LDIR komutu işlenmeden önce hafıza ve kaydedicilerin durumu



Şekil 8.17 : LDIR komutu işledikten sonra kaydedicilerde ve hafıza hücrelerindeki durum

8.2.5 Blok arama komutları

Blok arama komutları ile belli bir hafıza bölgesi içerisinde belirlenen byte'ın varlığı araştırılır. Blok arama komutları kullanılmadan önce tıpkı blok transfer komutlarında olduğu gibi bazı kaydedicilere ön yüklemeler yapılmasını zorunlu kılar. Benzeri aranacak veri bir kaydediciye yüklendikten sonra aramanın yapılacağı hafıza bölgesinin başlangıç ve bitiş adresleri ilgili kaydedicilere yüklenir. Daha sonra blok arama komutları çalıştırılmalıdır.

Blok arama komutlarını kullanabilmek için yine üzerinde çalışılmakta olan mikroişlemcinin komut setinde, bu işlevi yerine getirebilecek komutların bulunması gerekmektedir. Eğer kullanılan işlemcide arama işlevini gerçekleştirecek komut yoksa aynı işlevi yerine getirmek için küçük bir program parçası yazmak gerekmektedir. Tablo 8.6'da Zilog Z80 mikroişlemcinin komut setinde bulunan blok arama komutlarından örnekler verilmiştir.

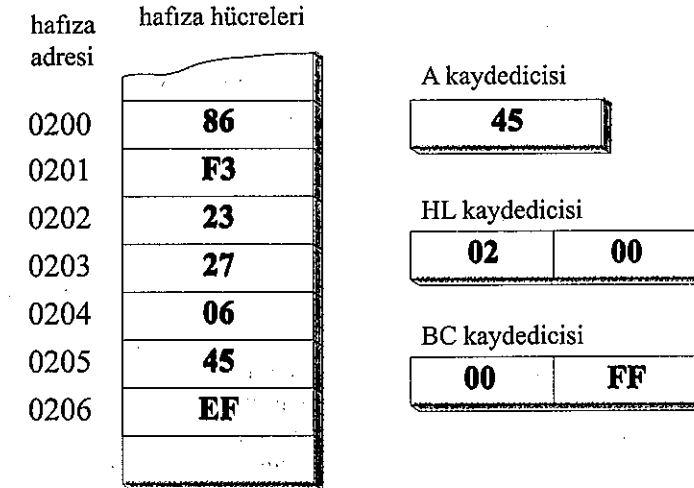
Tablo 8.6 : Bazı blok arama komutları

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	ED A1		CPI	(Zilog Z80) Kaydedicinin içeriğinden HL'nin gösterdiği adreste bulunan veri çıkarılarak karşılaştırma yapılır. Her karşılaştırma sonunda durum kodu bitleri etkilenir karşılaştırma sonrası HL bir artırılır, BC bir eksiltir. Durum kodu kaydedicisindeki etkilenen bitler. A<HL ise; S=1 Z=0 A=HL ise; S=0 Z=1 A>HL ise; S=0 Z=0
0102	ED B1		CPIR	(Z80) Kaydedici ile hafızanın içeriği karşılaştırılır. Her karşılaştırma sonrası adres sayıcı HL bir artırılır BC sayıcı bir eksiltir. Bu işlem içerikler eşit oluncaya veya BC=0 oluncaya kadar devam eder.

CPIR blok arama komutunun işleyişi: CPIR komutu, A kaydedicisinde bulunan değer ile HL adres kaydedicisinin işaret ettiği belleğin içeriğini karşılaştırarak arama işlemini yapar. Aramaya başlamadan önce arama yapılacak olan değer A kaydedicisine yüklenir, sonra aramanın başlanacağı hafıza bölgesinin başlangıç adresi HL kaydedicisine yüklenir. Ayrıca taranacak hafıza bölgesinin toplam miktarı (HL'nin işaret ettiği hafıza adresinden itibaren) BC kaydedicisine yüklenmelidir. Yani tarama yapılacak hafıza bölgesinin başlangıç adresi ve ne kadarlık bir bölgede tarama yapılacağı kaydedicilere yüklendikten sonra CPIR komutu kullanılmalıdır. Tablo 8.7'de Z80 mikroişlemci komutları kullanılarak (0200)₁₆ ile (02FF)₁₆ adresleri arasındaki hafıza bölgesinde (toplam 256 byte'lık bir bölge) (45)₁₆ verisinin nasıl aranacağını gösteren program bulunmaktadır. Şekil 8.18'de ise hafıza ve kaydedicilerin durumu görülmektedir.

Tablo 8.7: CPIR komutu kullanılarak bir verinin (45)₁₆ aranması

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	21 00 02		LD HL,0200H	HL'de aranacak hafıza bölgesinin başlangıç adresi yüklenir.
0003	11 FF 00		LD DE,00FFH	BC'ye aranacak toplam adres bölgesinin değeri yüklenir.
0006	3E 45		LD A,45H	Aranacak veri A kaydedicisine yüklenir.
0009	ED B1		CPIR	Komut HL'nin gösterdiği yerden itibaren DE'de gösterilen adrese kadar A kaydedicisindeki değeri arar. Bulduğu anda aramadan çıkar. Her aramadan sonra DE kaydedicisi 0 oluncaya kadar 1 azaltılır.



Şekil 8.18 : CPIR komutu aramaya başlamadan önce kaydedicilerin ve hafızanın durumu

CPIR komutu HL kaydedicisinin gösterdiği adresten itibaren arama yapmaya başlar. Aramaya HL'deki (0200)₁₆ adresinde bulunan veriden başlar. Bu adresten itibaren hafızada bulunan her veri A kaydedicisindeki veri ile karşılaştırılır. Her karşılaştırmadan sonra HL kaydedicisi bir artırılır, BC kaydedicisindeki adres değeri 1 azalır. Bu işlem BC sıfır oluncaya kadar veya aranan değer ile karşılaşılmadığı sürece devam eder. Aradığı değerle ilk karşılaşıldığında arama durdurulur. Aranan değer hangi hafıza hücresinde bulunduğu HL kaydedicisindeki değere bakılarak anlaşılabilir. Burada aranan değerin (45)₁₆ aynı (0205)₁₆ hafıza hücresinde bulunduğu için arama bu noktada bitmiştir. Şekil 8.19'da arama sonuçlandırıldığında kaydedicilerin durumu görülmektedir.

hafıza adresi hafıza hücreleri

0200	86
0201	F3
0202	23
0203	27
0204	06
0205	45
0206	EF

A kaydedicisi

45

HL kaydedicisi

02

05

BC kaydedicisi

00

FA

Şekil 8.19 : CPIR komutunun aramayı bitirmesinden sonraki kaydedicilerin durumu

CPIR komutunun yaptığı işin aynısını yapan komut her mikroişlemcide bulunmayabilir. Bu tip mikroişlemcilerde aynı işi yapmak için bazen küçük bir program parçası yazmak gerekir. Aşağıda komut setinde arama komutu bulunan bir mikroişlemci ile komut setinde arama komutu bulunmayan bir mikroişlemcideki arama programları görülmektedir. Tablo 8.9'daki program, komut setinde blok arama komutu bulunan bir mikroişlemci için yazılmıştır. Tablo 8.8'deki ise komut setinde blok arama komutu bulunmayan bir mikroişlemci için yazılmıştır. Bu programlar hız, program büyüklüğü ve sadelik bakımından karşılaştırıldığında bir mikroişlemcinin komut setinde arama komutunun bulunmamasının çok büyük bir dezavantaj olmadığı görülür.

Tablo 8.8 : MC6802 mikroişlemcisindeki komutlar kullanılarak yazılan arama programı

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	86 45		LDAA #45	Aranacak veri A kaydedicisine yüklenir.
0102	CE 00 FF		LDX #00FF	Aramanın başlanacağı hafıza adresi indeks kaydedicisine yüklenir.
0105	08	ARA :	INX	İndeks kaydedici 1 artırılır. (İlk seferde indeks kaydedici (0100) ₁₆ adresini işaret edecektir.)
0106	8C 02 00		CPX #0200	Eğer arama yapılacak bölge bitmiş ise SON'a gidilir. Bunun için indeks kaydedici ile (0200) ₁₆ karşılaştırılır. Eşit ise Z biti =1 olur.
0109	27 04		BEQ SON	BEQ komutu ile Z=1 ise sona dallanılır. Z=0 ise program bir sonraki komutla devam eder.
010B	A1 00		CMPA \$00,X	İndeks kaydedicinin işaret ettiği hafıza adresindeki değer ile A kaydedicisindeki değer karşılaştırılır. Karşılaştırma CCR kaydedicisindeki bitleri etkiler.
010D	26 F6		BNE ARA	Karşılaştırma sonucu eşit değilse yani CCR'nin Z biti Z=0 ise ARA etiketine dallanır ve aramaya devam edilir. Bu döngü karşılaştırma eşit oluncaya kadar yani Z=1 oluncaya kadar devam eder. Z=1 olduğunda döngüden çıkılır.
010F	3F	SON :	SWI	Programı bitir komutu.

Tablo 8.9 : Z-80 mikroişlemcisi komut setindeki "CPIR" arama komutu ile arama yaptırmak

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	21 00 02		LD HL,0200H	HL'de aranacak hafıza bölgesinin başlangıç adresi yüklenir.
0003	01 FF 00		LD BC,00FFH	BC'ye aranacak toplam adres bölgesinin değeri yüklenir.
0006	3E 45		LD A,45H	Aranacak veri A kaydedicisine yüklenir.
0009	ED B1		CPIR	Komut HL'nin gösterdiği yerden itibaren DE'de gösterilen adrese-kadar A kaydedicisindeki değeri arar. Bulduğu anda aramadan çıkar. Her aramadan sonra DE kaydedicisi 0 oluncaya kadar 1 azaltılır.

8.3 ARİTMETİK VE MANTIK İŞLEM KOMUTLARI

Aritmetik mantık işlem komutları assembly programlarında en çok kullanılan komutlardandır. Program akışları genellikle bu komutlar işlendikten sonra elde edilen sonuçlara göre yön alır. Aritmetik ve mantıksal işlemlerin tümü mikroişlemcideki ALU ünitesinde gerçekleştirilir. Aritmetiksel işlemler; "toplama, çıkarma, artırma ve eksiltme" işlemleridir. Mantıksal işlemler ise "Karşılaştırma, öteleme, döndürme, AND, OR, EXOR" işlemleridir. Aritmetik ve mantıksal işlemleri kaydediciler veya herhangi bir okunup/yazılabilir hafıza hücresi üzerinde gerçekleştirilebilir ve yine yapılan işlemin sonucu kaydedicilerde veya hafıza hücresinde saklanır. Bu komutların sonuçları DDR (durum kodu kaydedicisi) bitlerini de etkiler. Assembly programlamada bu bitlerin durumu çok sık olarak değerlendirilir.

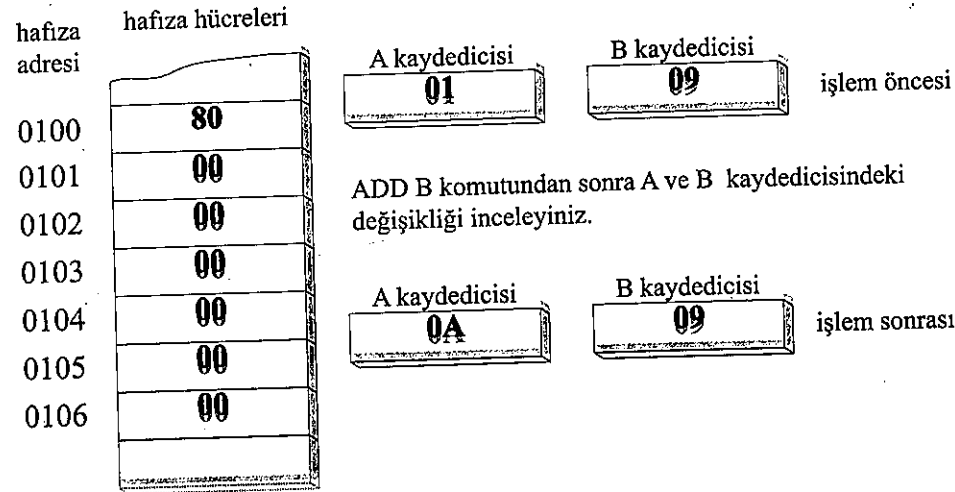
8.3.1 8 Bitlik Aritmetik Ve Mantık İşlem Komutları

8 bitlik aritmetik mantık komutları ile üzerinde işlem yapılacak veri, adres veya kaydedicilerin 8 bit büyüklüğünde olması gerekmektedir. Bu komutlarla yapılan işlemin sonucu yine 8 bit büyüklüğünde olacaktır, dolayısıyla işlemin sonucunda oluşan değer 8 biti aşan kısmı değerlendirilmeye konulamaz. 8 bitlik aritmetik işlemlerde kullanılabilecek en büyük sayı (FF)₁₆ olacaktır. Tablo 8.10'da değişik mikroişlemcilerin komut setinde bulunan 8 bitlik aritmetiksel ve mantıksal komutlar bulunmaktadır.

Tablo 8.10 : Bazı 8 bitlik aritmetik-mantık komutları

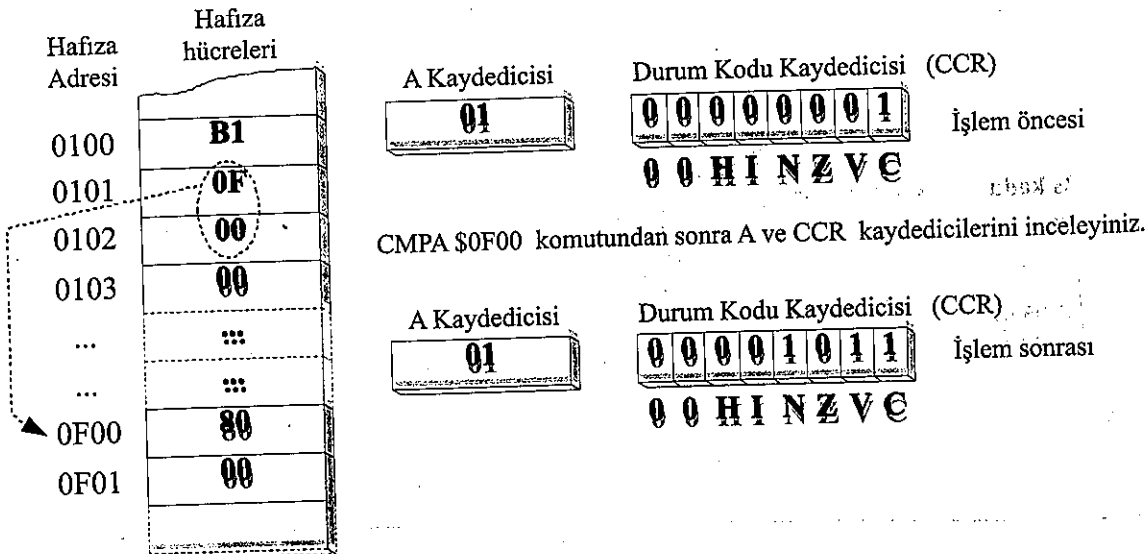
İş kodu	Assembly komutu	Açıklama
80	ADD B	(I8080) A=A+B; A ile B toplanıp sonuç A'ya yüklenir.
81 3F	ADI #3F	(I8080) A=A[]+(3F) ₁₆ ; A kaydedicisindeki değer ile (3F) ₁₆ toplanıp sonuç A'ya yüklenir.
86	ADD HL	(I8080) HL kaydedicisinin gösterdiği adresteki veri ile A kaydedicisindeki veri toplanıp tekrar A'ya yüklenir.
98	SUB B	(I8080) A=A[] -B[]; A'dan B çıkarılır, sonuç A'ya yüklenir.
84 0F	ANDA #0F	(MC6802) A 'daki veri (0F) ₁₆ ile AND işlemine tâbi tutulur.
88 0F	EORA #0F	(MC6802) A 'daki veri (0F) ₁₆ ile EXOR işlemine tâbi tutulur.
81 0F 00	CMPA \$0F00	(MC6802) (0F00) ₁₆ adresindeki veri ile A'daki veri karşılaştırılır. İşlemin sonucu CCR'deki bayrakları etkiler.
5C	INCB	(MC6802) B kaydedicisindeki veri 1 artırılır.

ADD B toplama komutunun işleyişi: Intel'in 8080 kodu ile ürettiği mikroişlemcisinde ve diğer işlemcilerinde kullanılan bir komuttur. Bu komut ile A kaydedicisindeki sayı ile B kaydedicisindeki sayı toplanır. Sonucu yine A kaydedicisine yüklenir. Komutun işlem kodu hafızada yalnızca 1 byte yer kaplar. Şekil 8.20'de komutun hafızadaki durumunu ve kaydedicilerdeki değişiklikleri göstermektedir.



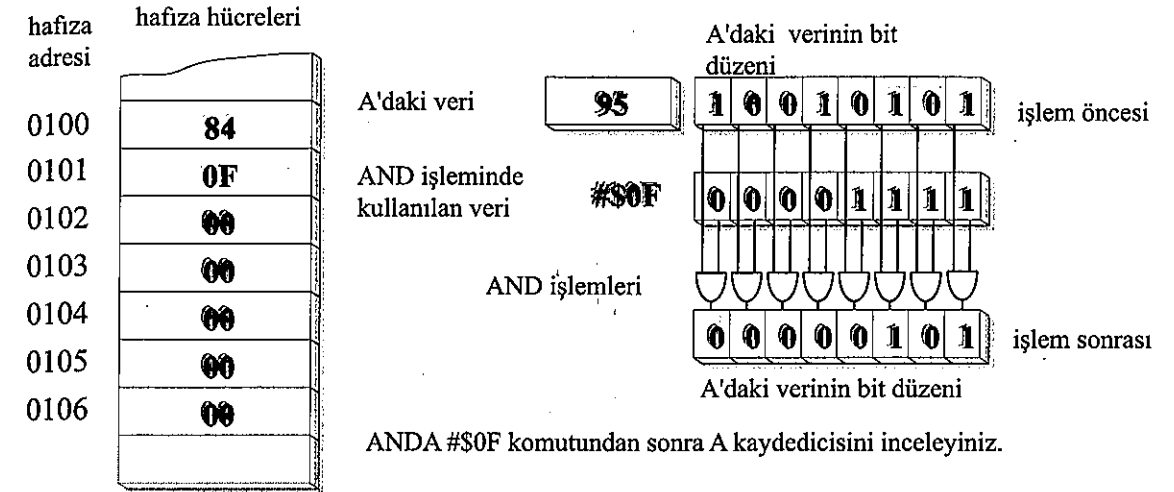
Şekil 8.20 : ADD B komutunun işleyişi

CMPA \$0F00 komutunun işleyişi: MC6800 mikroişlemci ailesinin komut setinde bulunan bir komuttur. CMPA \$0F00 komutu hafızanın (0F00)₁₆ adresinde bulunan veri ile A kaydedicisinde bulunan veriyi karşılaştırır. Karşılaştırma sonucunda A kaydedicisindeki veri ile belirtilen adresindeki veri arasında şu üç durumdan birisi olabilir; A[] > (0F00)₁₆, A[] = (0F00)₁₆ ve A[] < (0F00)₁₆. Bu sonuçlara göre yalnızca "durum kodu kaydedicisi" ndeki negatif (N), sıfır (Z), taşma (V) ve elde (C) bayrakları etkilenir. CMPA \$0F00 komutunun işlem kodu "B1 0F 00" dir. Komut (0100)₁₆ adresinden itibaren yazılmıştır ve hafızada toplam 3 byte yer tutar. Komutun işleyişi ve kaydedicilerin durumu şekil 8.21'de görülmektedir.



Şekil 8.21 : CMPA \$0F00 komutunun işleyişi

ANDA #\$0F komutunun işleyişi: Motorola 6800 işlemci ailesinin komut setindeki mantıksal (lojik) işlevi olan bir komuttur. Bu komut A kaydedicisindeki veri ile (0F)₁₆ verisini VE (AND) işlemine tâbi tutar. ANDA #\$0F komutunun işlem kodu "84 0F" tir. (0F)₁₆ değeri ile AND işlemine tâbi tutulan A kaydedicisindeki veri yine işlemten sonra A kaydedicisinde kalır. Ayrıca CCR kaydedicisindeki negatif (N), sıfır (Z), taşma (V=0) bitlerini de bu işlemin sonucuna göre bir değer alır.



Şekil 8.22 : ANDA #\$0F komutunun işleyişi

8.3.2 16 Bitlik Aritmetik Ve Mantık İşlem Komutları

16 bitlik aritmetik ve mantık komutları ile üzerinde işlem yapılan veri, adres veya kaydedicinin 16 bitlik büyüklüğünde olması gerekmektedir. Tablo 8.11'de değişik mikroişlemcilerin komut setinde bulunan 16 bitlik aritmetiksel ve mantıksal komutlar bulunmaktadır.

Tablo 8.11 : 16 bitlik aritmetik mantık komutları

İş kodu	Assembly komutu	Açıklama
19	ADD HL,DE	(Z80) DE ile HL to planır ve sonuç HL'ye yüklenir. HL=DE+HL
09	DEX	(MC6802) İndeks kaydedici bir azaltılır. X=X-1
32 05 01	STA \$0105	(I8080) A kaydedicisindeki veri belirtilen adrese yüklenir.
8C 00 F0	CPX #\$00F0	(MC6802) İndeks kaydedicisindeki veri ile 00F0 verisi karşılaştırılır.
FD	INC IY	(Z80) Y kaydedicisi 1 arttırılır. Y=Y+1

ADD HL,DE komutunun işleyişi: Bu komut, HL kaydedicisinde bulunan değer ile BC kaydedicisindeki değeri toplayıp, sonucu HL kaydedicisinde saklar. Komutun işleyişi ve kaydedicilerin durumu şekil 8.23'de gösterilmiştir.

hafıza adresi	hafıza hücreleri	HL kaydedicisi	DE kaydedicisi	
0100	19	02	00	işlem öncesi
0101	00			
0102	00			
0103	00			
0104	00			
0105	00			
0106	00			

ADD HL,BC komutundan sonra HL ve DE kaydedicilerini inceleyiniz.

HL kaydedicisi	DE kaydedicisi	
02	F3	işlem sonrası

8.3.3 Genel Amaçlı Aritmetik Komutları

Genel amaçlı aritmetik komutlar, kaydediciler veya hafızalardaki sayılar üzerinde, 1'e tümlleme, 2'ye tümlleme, BCD koduna çevirme ve durum kodu kaydedicilerindeki bitleri "1" ve "0" yapmada kullanılan komutların tümünü oluşturur. Bu komutlardan bazıları tablo 8.12'de gösterilmiştir.

Tablo 8.12 : Genel amaçlı komutlardan örnekler

İş kodu	Assembly komutu	Açıklama
27	DAA	(Z80) kaydedicisindeki sayıyı BCD forma dönüştürür.
0D	SEC	(MC6802) Elde bayrağını "1" yapar. C=1 olur.
ED 44	NEG	(Z80) kaydedicisindeki sayının ikiye tümlenmesini alır.

NEG komutunun işleyişi: Z80 mikroişlemcisindeki NEG komutu A kaydedicisindeki verinin 2'ye tümlenmesini alır. 2 byte'lık bir komuttur. Komutun işlem kodu "ED 44" tür. Şekil 8.24'de A kaydedicisindeki verinin ikiye tümlenmesinin nasıl alındığı ve kaydedicilerin durumu gösterilmiştir. Komut tüm bitlerin tersini alıp sonuca "1" eklemiştir.

hafıza adresi	hafıza hücreleri	A kaydedicisi	
0100	ED	1 0 1 1 1 0 1 0	işlem öncesi
0101	44		
0102	00		
0103	00		
0104	00		
0105	00		
0106	00		

NEG komutu A kaydedicisindeki verinin, 2'ye tümlenmesini alır.

A kaydedicisi	
0 1 0 0 0 1 1 0	işlem sonrası

Şekil 8.24 : NEG komutu ile A kaydedicisindeki verinin 2'ye tümlenmesinin alınması

DAA komutunun işleyişi: Kaydedicideki 8 bitlik sayıyı BCD koda dönüştürür.

hafıza adresi	hafıza hücreleri	A kaydedicisi	A kaydedicisi	
0100	27	0 1 0 1 1 1 0 1	5D	işlem
0101	00			
0102	00			
0103	00			
0104	00			
0105	00			
0106	00			

DAA komutundan sonra A kaydedicisindeki veri BCD for dönüşür.

A kaydedicisi	A kaydedicisi	
0 1 1 0 0 0 1 1	63	işlem

0101 1101 → ilk dört bit 9'dan büyük ise ilk dört bite 0110 eklenir
 + 0000 0110 → taşan bit yüksek değerlikli son dört bite eklenir.
 + 0001 0011 → 63
 0110 → 63

Şekil 8.25 : A kaydedicisindeki verinin BCD forma dönüştürülmesi

Komut bunu yaparken yarım elde (H) bayrağını kullanır. Düşük değerlikli ilk dört bit 9'dan büyükse ilk dört bite (0110)₂ ekler. Yine yüksek değerlikli son dört bit 9'dan büyükse veya C bayrağı 1 ise yine bu sayıya (0110)₂ ekleyerek BCD forma dönüştürür. Z80 komut setindeki 1 byte'lık DAA komutunun işlem kodu "27" dir.

8.4 TEST VE DALLANMA KOMUTLARI

Bu komut grubundaki komutların en büyük özelliği program akışını değiştirebilmeleridir. Şimdiye kadar anlatılan komutlar işletildikten sonra mikroişlemci program sayıcısını bir sonraki komutu yürütmek için "1" arttırırdı. Dolayısıyla programdaki komutlar art arda işletilirdi. Fakat test ve dallanma komutlarının program akışını değiştirebilme özellikleri vardır. Bu komutlar durum kodu kaydedicisindeki (CCR) bitlerin durumunu değerlendirerek programın akışına yön verirler.

Test ve dallanma grubundaki komutlar iki temel gruba ayrılırlar:

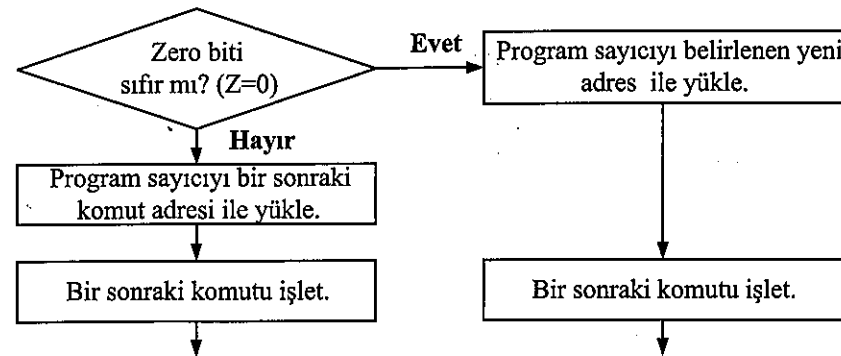
- 1- Şartlı dallanma komutları
- 2- Şartsız dallanma komutları

8.4.1 Şartlı Dallanma Komutları

Bu komutlar bilgisayara, karşılaştığı özel durumlarda ("A=0" veya "A0" gibi) karar verme becerisi kazandıran komutlardır. Şartlı dallanma komutları, şart testini CCR'deki bitlerin son durumuna göre yapar. Programlama sırasında, dallanma komutlarından hemen önce CCR'yi etkileyecek komutlar kullanılarak ancak programın çalışmasına (CCR'deki bitlerin durumuna göre) yön verilebilir. Mikroişlemci bu komutlar ile karşılaştığında komutun önceden belirlediği şartlar oluşmuşsa bir yola, şart oluşmamışsa başka bir yola saparak programın karar verme sürecini işletir. Şartlı dallanma komutları "bağlı adresleme yöntemini" kullanır. Mikroişlemci, şartlı dallanma komutlarından herhangi biri ile karşılaştığında yapacağı işlemler aşağıdaki gibidir.

- ❖ Mikroişlemci dallanma komutunu "komut kod çözücüsü"nde çözerek komutun eylemini gerçekleştirmeye geçer.
- ❖ Dallanma komutuna bağlı olarak CCR kaydedicisindeki ilgili bayrağa bakılır. (N=0, Z=1 mi? gibi)
- ❖ Eğer komut şartı sağlanmışsa program sayıcı yeni adresle yüklenir ve program çalışmasına önceden belirlenen yeni adresten devam eder.
- ❖ Eğer şart sağlanmamışsa program çalışmasını normal olarak sürdürür.

Mikroişlemciler, dallanma komutları ile karşılaştıklarında (örneğin "Z=0 ise dallan" gibi) karar verme süreçlerinin nasıl işlediğini gösteren diyagram şekil 8.26'da gösterilmiştir.



Şekil 8.26 : Dallanma komutunun program akışını değiştirmesi



Bağlı Adresleme Yönteminde Kullanılacak Sapma Miktarının Hesaplanması:

Bağlı adresleme yöntemine göre kullanılan dallanma komutlarında belirlenmesi gereken sapma miktarının bulunmasında aşağıda anlatılan yöntem kullanılır. Dallanma komutu ile program sayıcı ancak 127 byte ileri arttırılabilir veya 128 byte geriye azaltılabilir.

Dallanma komutunda belirtilen sapma değeri eğer negatif bir sayı ise dallanma geriye doğru olacaktır. Geriye doğru dallanmalarda komutta verilen sapma miktarı program sayıcıdan çıkarılır, sonuçta elde edilen adres ikiye tımlenir ve programın çalışmasına devam edeceği yeni adres böylece belli olur.

Eğer dallanma komutunda belirtilen adres pozitif ise komutta belirtilen sapma miktarı program sayıcıya eklenerek programın işleyeceği yeni adres belirlenmiş olur. Anlaşıldığı gibi bağlı adresleme yöntemini kullanan dallanma komutları ile ancak 127 byte ileri, 128 byte geri dallanılabilir. Aşağıdaki programda, bağlı adresleme yöntemini kullanan dallanma komutunun, işleyişi MC 6802 komutları ile gösterilmiştir.

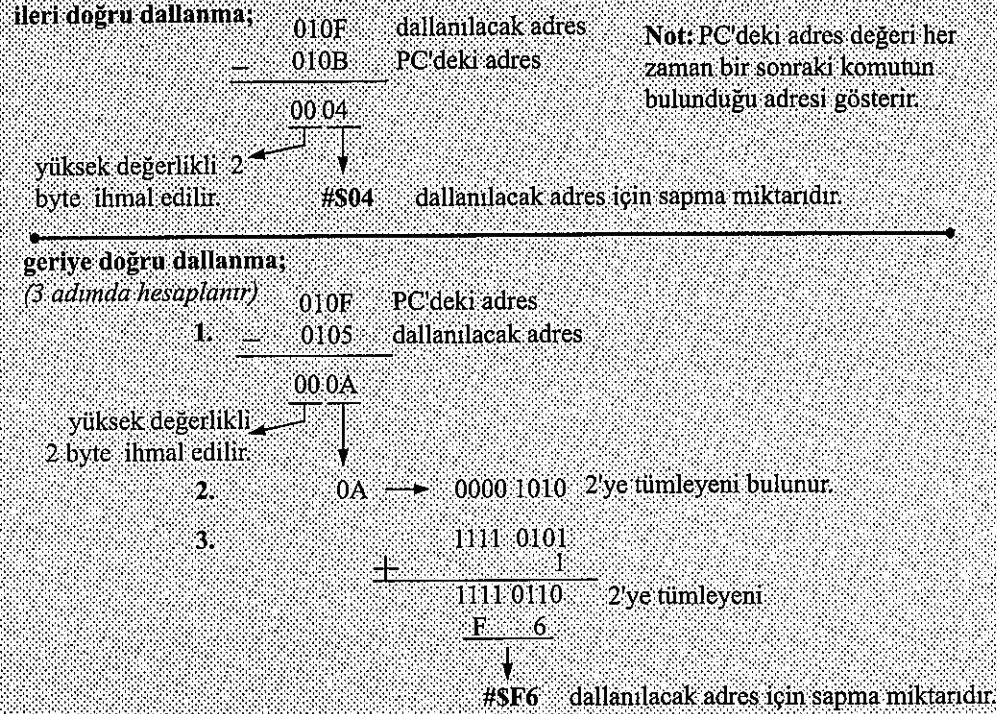
Adres	İş kodu	Etiket	Assembly komutu	Açıklama
0100	86 45		LDAA #45	Aranacak veri A kaydedicisine yüklenir.
0102	CE 00 FF		LDX #00FF	Aramanın başlan acağı hafıza adresi indeks kaydedicisine yüklenir.
0105	08	ARA :	INX	İndeks kaydedici 1 arttırılır. (Komutun ilk işleyişinde indeks kaydedici (0100) 16 adresini işaret edecektir.)
0106	8C 02 00		CPX #0200	Eğer arama yapılacak bölge bitmiş ise SON'a gidilir. Bunun için indeks kaydedici ile (0200) 16 karşılaştırılır. Eşit ise Z biti =1 olur.
0109	27 04		BEQ SON	BEQ komutu ile Z=1 ise sona dallanılır. Z=0 ise program bir sonraki komutla devam eder.
010B	A1 00		CMPA \$00,X	İndeks kaydedicinin işaret ettiği hafıza adresindeki değer ile A kaydedicisindeki değer karşılaştırılır. Karşılaştırma CCR kaydedicisindeki bitleri etkiler.
010D	26 F6		BNE ARA	Karşılaştırma sonucu eşit değilse yani CCR'nin Z biti Z=0 ise ARA etiketine dallanır ve aramaya devam edilir. B u döngü karşılaştırma eşit oluncaya kadar yani Z=1 oluncaya kadar devam eder. Z=1 olduğunda döngüden çıkılır.
010F	3F	SON :	SWI	Programı bitir komutu.

BEQ SON komutu ile ileri dallanılmak istenmektedir (eğer istenilen şart sağlanırsa). BEQ SON komutunun işlem kodunda SON yerine yazılacak değer hesaplanmalıdır. Komut kodunda SON yerine yazılacak değer şöyle hesaplanır. Bu komut yürütülürken program sayıcı (010B)₁₆ adresini göstermektedir. Dallanılmak istenen adres değeri ise (010F)₁₆'dır. Dallanılmak istenen adresten PC'deki adres çıkarılır, sonuç bağlı adres değerini verir, bu sonuç SON yerine yazılacak sapma miktarıdır.

BNE ARA komutu ile şart sağlanmışsa ARA etiketinin olduğu adrese dallanılır. Komuttan sonra ARA şeklinde belirtilen alana komutun işlem koduna çevrilirken yazılacak değer hesaplanması gerekmektedir. Komut yürütülürken PC (010F)₁₆ adresini gösterir. Dallanılmak istenen adres ise (0105)₁₆ dir. Geriye doğru dallanmalarda PC deki değerden dallanılmak istenen adres çıkarılır. Çıkan sonucun yüksek değerlikli bitleri atılır. Geriye kalan sayının ikiye tımleneni bulunarak sapma değeri hesaplanmış olur.



Yukarıdaki programda kullanılan ileri ve geri dallanmanın nasıl hesaplanacağı aşağıda gösterilmiştir.



BNE #SF6 komutunun işleyişi: BNE #SF6 komutu CCR'deki Z bitine göre dallanmayı gerçekleştirecektir. Eğer Z biti sıfır değilse ($Z \neq 0$) dallanma gerçekleşmez ve bir sonraki komutla program çalışmasını sürdürür. Eğer Z biti sıfıra eşit ise ($Z=0$) dallanma gerçekleşecektir. Komuttan sonra program, dallanılan adresten itibaren çalışmasını sürdürecektir. Dallanılacak yer ise BNE komutundan sonra belirtilen sapma kadar olacaktır. BNE #F6 komutundaki sapma miktarı (işaretili sayı olarak değerlendirilecek) negatif olduğundan dolayı, dallanma geriye doğru gerçekleşecektir. Dallanılacak adresin sapma miktarının nasıl hesaplanacağı yukarıdaki bilgi kutusunda anlatılmıştır.



İki Tabanındaki İşaretili Sayıların Sayı Aralıkları:

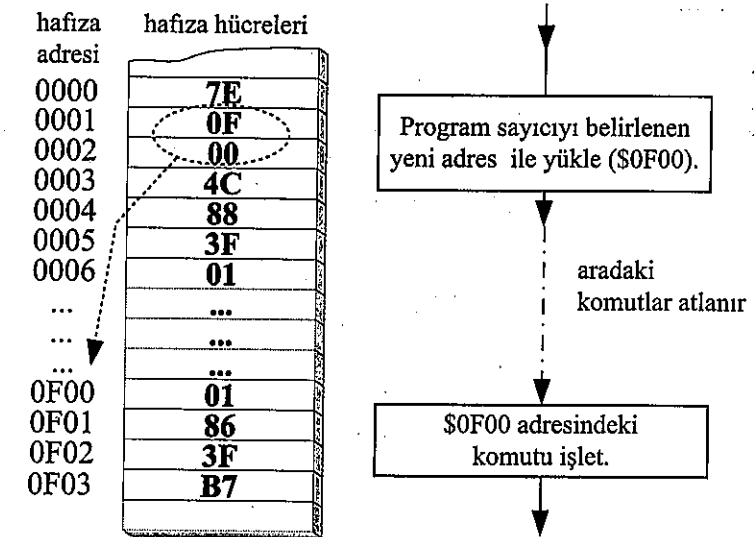
İşaretili sayılarda 8. bit sayının pozitif veya negatif olduğunu gösterir. 8. bite işaret biti de denilir. Eğer işaret biti "0" ise sayı pozitif, "1" ise sayı negatiftir. 8 bitlik işaretili sayılar 128 ve +127 arasında bulunabilir.

Pozitif sayıların bulunduğu sayı aralığı: $(00000000)_2, (01111111)_2 \rightarrow (00)_{16}, (7F)_{16} \rightarrow (0, +127)$

Negatif sayıların bulunduğu sayı aralığı: $(10000000)_2, (11111111)_2 \rightarrow (80)_{16}, (FF)_{16} \rightarrow (-128, -1)$

8.4.2 Şartsız Dallanma Komutları

Mikroişlemci şartsız dallanma komutları ile karşılaştığında, program sayıcının içeriğini programda belirtilen değer ile yükler ve çalışmasına bu noktadan sonra devam eder. Şartsız dallanmak için hiç bir durum göz önüne alınmaz. MC6802 mikroişlemcisinde kullanılan şartsız dallanma komutu JMP'dir. İngilizce'deki JUMP'ın (atla) kısaltılmasıdır. Komutun iş kodu "7E"dir. Eğer şartsız olarak $(0F00)_{16}$ adresine dallanmak istendiğinde yazılacak komut JMP \$0F00 şeklinde olmalıdır. Komutun işlem kodu ise "7E 0F 00" şeklinde olacaktır. Şekil 8.27'de görüleceği üzere $(0000)_{16}$ adresinde bulunan şartsız dallanma komutu mikroişlemci tarafından işletildikten sonra program, akışını $(0F00)_{16}$ adresinden itibaren devam ettirecektir.



Şekil 8.27 : Şartsız dallanma komutunun işleyişi

Tablo 8.13 : Bazı mikroişlemcilerin komut setinde bulunan şartlı ve şartsız dallanma komutları

İş kodu	Assembly Komutu	Açıklama
DA 00 0F	JC \$0F00	(I8085) Elde varsa dallan. CY=1
CA 00 0F	JZ \$0F00	(I8085) Sıfır ise dallan. Z=1
F2 00 0F	JP \$0F00	(I8085) Pozitif ise dallan. S=0
FA 00 0F	JM \$0F00	(I8085) Negatif ise dallan. S=1
C3 00 0F	JMP \$0F00	(I8085) Şartsız dallan.
29 5F	BVS #5F	(MC6802) Taşma bir ise dallan. V=1
24 5F	BCC #5F	(MC6802) Elde sıfır ise dallan. C=0
27 5F	BEQ #5F	(MC6802) Eşit ise dallan. Z=1
7E 0F 00	JMP \$0F00	(MC6802) Şartsız dallan.
C3 00 0F	JP \$0F00	(Z80) Şartsız dallan.
DA 30 40	JPC \$4030	(Z80) C=1 ise dallan.
C2 60 50	JP NZ \$5060	(Z80) NZ=1 ise dallan.

8.5 KAYDIRMA VE DÖNDÜRME KOMUTLARI

Bu gruptaki komutlar ile kaydediciler ve hafıza hücrelerindeki veriler üzerinde; sola kaydırma, sağa kaydırma, sola döndürme, sağa döndürme işlemlerini gerçekleştirir. Bu işlemlerin kaydediciler üzerinde nasıl gerçekleştiği aşağıda anlatılmıştır.

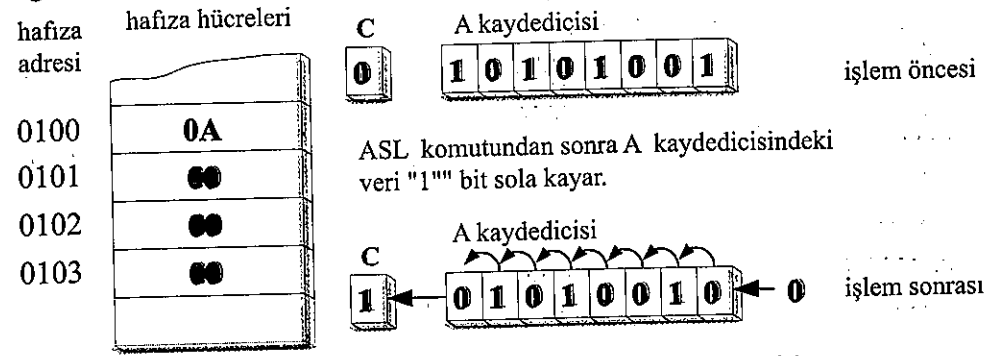
8.5.1 Sola Kaydırma Komutları

Mikroişlemci sola kaydırma komutu ile karşılaştığında kaydedicideki veriyi 1 bit sola kaydırır. Tablo 8.14'te mikroişlemcilerde kullanılan sola kaydırma komutları verilmiştir.

Tablo 8.14 : Farklı mikroişlemcilerin sola kaydırma komutları ve işlem kodları

İş kodu	Assembly komutu	Açıklama
CB 27	SLAA	(Z80) A kaydedicisindeki veriyi "1" bit sola kaydırır.
0A	ASL	(Mos Tec. 6502) A kaydedicisini sola kaydırır.
78 0F 00	ASL #0F00	(MC6802) (0F00) ₁₆ adresindeki veriyi sola kaydırır.

ASL komutunun işleyişi: MOS Tec. firmasının ürettiği 6502 mikroişlemcisinin komut setinde bulunan ASL komutu 1 byte'lık bir komuttur. Komutun işlem kodu "A0" dır. Komut, A kaydedicisindeki veriyi birer bit sola kaydırır. En soldaki D7 biti elde bayrağına kayar. Verinin en sağından ise "0" girer. Komutun işleyişi şekil 8.28'de gösterilmiştir.



Şekil 8.28 : Sola kaydırma komutunun işleyişi

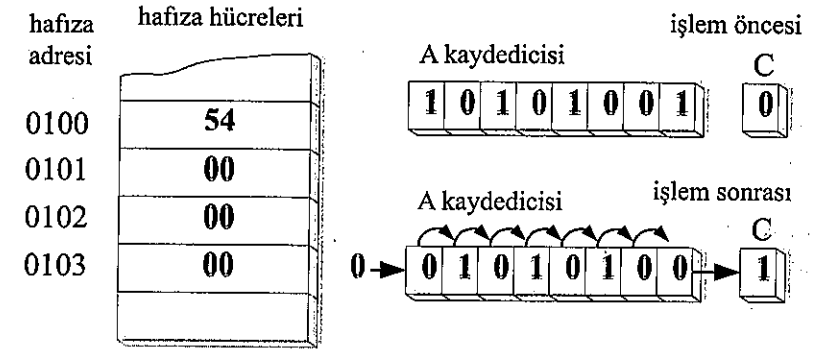
8.5.2 Sağa Kaydırma Komutları

Mikroişlemci sağa kaydırma komutu ile karşılaştığında kaydedicideki veriyi 1 bit sağa kaydırır. Verinin soluna "0" girer. Verinin D0 biti ise elde bitine girer. Tablo 8.15'de değişik mikroişlemcilerde kullanılan sağa kaydırma komutları verilmiştir.

Tablo 8.15 : Değişik mikroişlemcilerdeki sağa kaydırma komutları ve işlem kodları

İş kodu	Assembly komutu	Açıklama
CB 3F	SRLA	(Z80) A kaydedicisindeki veriyi "1" bit sağa kaydırır.
CB 2A	SRAD	(Z80) D Kaydedicini aritmetik olarak sağa kaydırır.
54	LSRB	(MC6802) B kaydedicisindeki veriyi "1" bit sola kaydırır.

LSRB komutunun işleyişi: MC6802'nin komut setinde bulunur. LSRB komutu ile sağa "1" bitlik kayma gerçekleştirilir. 1 byte'lık bir komuttur. Komutun işlem kodu "54" tür. Komut, B kaydedicisindeki veriyi birer bit sağa kaydırır. En sağdaki D0 biti elde bayrağına kayar. Verinin en soluna ise "0" girer. Komutun işleyişi şekil 8.29'da gösterilmiştir.



Şekil 8.29 : Sağa kaydırma komutunun işleyişi

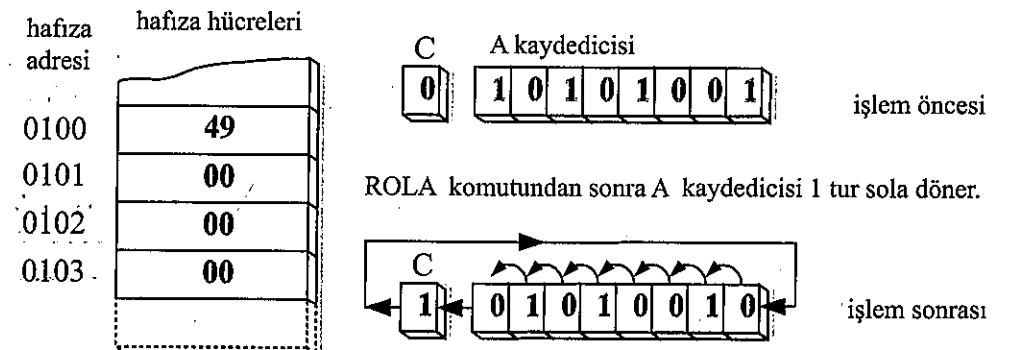
8.5.3 Sola Döndürme Komutları

Döndürme komutları, kaydedicideki veya hafızadaki bir veriyi, dairesel bir kaydırma işlemine tâbi tutar. Dairesel döngüyü tamamlamak için elde biti kullanılır. Her komutun işleyişinde bir bit dönme işlemi gerçekleşir. Eğer 8 kez art arda sola döndürme komutu kullanılırsa veri eski hâlini alır. Sola doğru dönme işleminde, verinin soluna elde biti hücreindeki değer girer. Verinin D7 biti ise elde bitine kayar. Tablo 8.16'da değişik mikroişlemcilerin komut setlerinde bulunan sola döndürme komutları verilmiştir.

Tablo 8.16 : Değişik mikroişlemcilerdeki sola kaydırma komutları ve işlem kodları

İş kodu	Assembly komutu	Açıklama
49	ROLA	(MC6802) A kaydedicisindeki veri sola döndürülür.
17	RAL	(I8080) A'yı elde üzerinden sola döndür.
CB 11	RL C	(Z80) C kaydedicisini elde biti üzerinden sola döndürür.

ROLA komutunun işleyişi: MC6802'nin komut setinde bulunur. 1 byte'lık bir komuttur. Komutun işlem kodu "49" dur. ROLA komutu ile A kaydedicisindeki veri "1" bitlik dönme yapar. Şekil 8.30'da komutun A kaydedicisi üzerinde nasıl bir işlem yaptığı gösterilmiştir.



Şekil 8.30 : Sola döndürme komutunun işleyişi

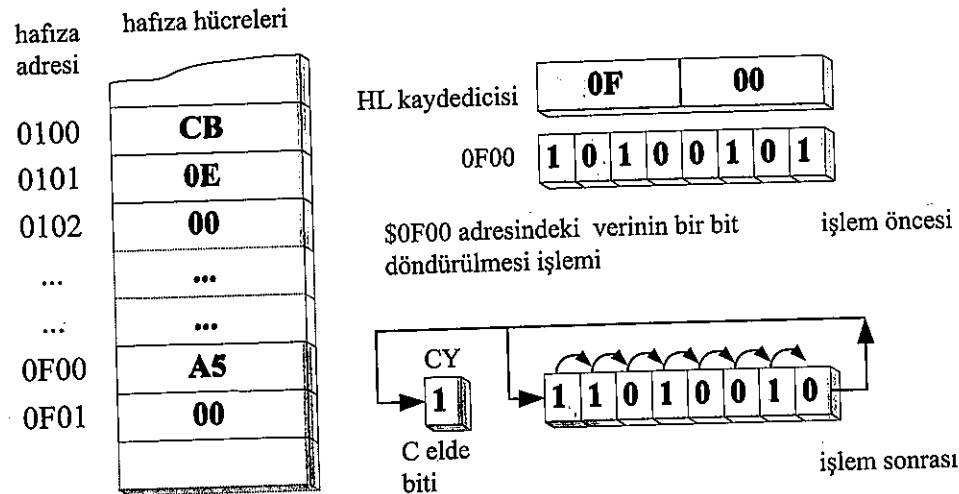
8.5.4 Sağa Döndürme Komutları

Sağa döndürme grubundaki komutlar da kaydedici veya hafızadaki verileri sağa doğru bir bit döndürür. Döndürme işleminde yine elde biti kullanılır. Tablo 8.17'de değişik mikroişlemcilerin komut setlerinde bulunan sağa döndürme komutları verilmiştir.

Tablo 8.17 : Sağa döndürme komutları ve işlem kodları

İş kodu	Assembly komutu	Açıklama
1F	RR A	(Z80) A kaydedicisindeki veri 1 kez sağa döndürülür.
CB 0E	RRC (HL)	(Z80) HL kaydedicisinin gösterdiği adresteki veriyi 1 bit sağa döndürür.
56	RORB	(MC6802) B kaydedicisini sağa döndürür.

RRC (HL) komutunun işleyişi: Z80 mikroişlemcisi komutudur. 2 byte'lık bir komuttur. Komutun işlem kodu "CB 0E" dir. Bu örnekte komut, hafızanın (0100)₁₆ adresinden başlamaktadır. Komut üzerinde işlem yapacağı hafıza hücrelerinin adresini HL kaydedicisinden alır. Dolayısı ile HL kaydedicisinde bulunan adresi döndürme işlemini yapacağı verinin bulunduğu adres olarak kabul eder ve bu adresteki veriyi 1 bit sağa döndürür. Bu komutta döndürülecek veri, HL kaydedicisinin gösterdiği (0F00)₁₆ adresindeki veri olacaktır. Onun için RRC (HL) komutunu kullanmadan önce hangi adresteki veriyi sağa doğru döndürmek istiyorsak o adresi HL kaydedicisine yüklememiz gerekmektedir. Bu tamamen programcının sorumluluğundadır. Bu komut döndürme işleminde hem elde bitini hem de D7 bitini aynı anda değiştirir. Şekil 7.31 'de komutun işleyişi gösterilmiştir. Şekilde dikkat edilecek olursa komut hem CY elde bitini hem de D7 bitini etkilemektedir.



Şekil 8.31 : RRC (HL) komutunun işleyişi

8.6 BİT İŞLEME KOMUTLARI

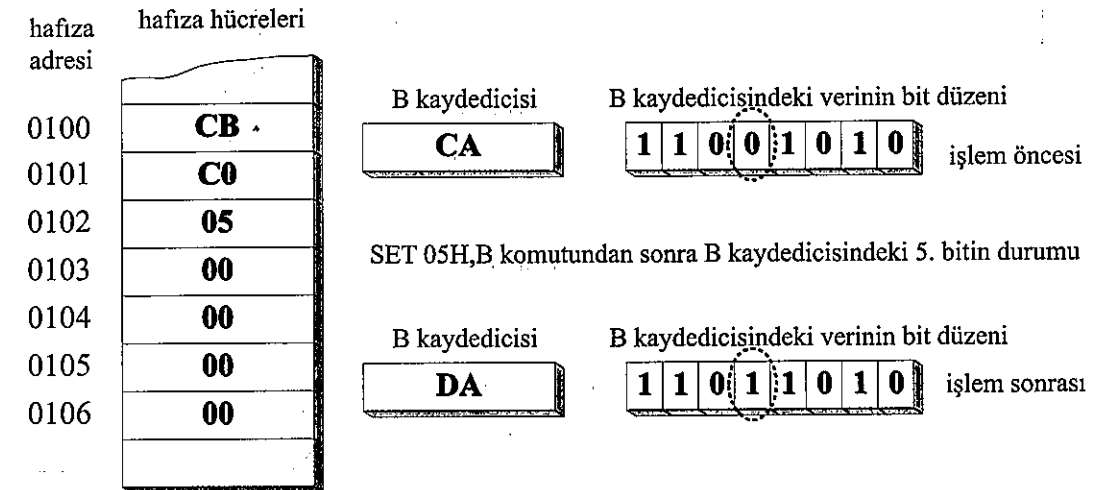
Mikroişlemcilerin bit işleme komutları kaydedicilerdeki veya hafıza hücrelerindeki veriler üzerinde yalnızca "1" bit üzerinde değişiklik yapan komutlardır. Bu komutlar özellikle bit seviyesinde karşılaştırma ve test etme işlemlerinde çok işe yaramaktadır. Mikroişlemci komutları genellikle verilerin tamamı üzerinde bir işlem yaparlar. Bit seviyesinde işlem yapmazlar. Fakat bazı özel durumlarda ihtiyaç duyulduğundan üretici firmalar mikroişlemcilerin komut setlerinde bit seviyesinde işlem yapabilen

komutlar da bulundurmaktadırlar. Bu komutlar genel amaçlı kaydediciler veya hafızada bulunan veriler üzerinde istenen bir bitin (D7...D0) "1" mi "0" mı olduğunun testinde veya istenen bir bitin (D7...D0) "1" e veya "0" a kurmanın yanında özel amaçlı kaydediciler üzerinde de bit seviyesinde işlemler gerçekleştirirler. Özel amaçlı kaydedicilerden olan "Durum Kodu Kaydedicisi"ndeki (CCR; Condition Code Register) son yapılan aritmetiksel veya mantıksal işlemlerin durumunu gösteren bitler üzerinde işlemler gerçekleştirebilir. Örneğin bu bitlerden elde bitini "1" veya "0" etmek, kesme denetim bitini kurmak ("1" veya "0" a kurmak) gibi işlemler olabilir. Tablo 8.18'de bazı mikroişlemcilerin komut setlerindeki, bit seviyesinde işlemler yapan komutlar bulunmaktadır. Bu komutlardan bazılarının işleyişi de aşağıda anlatılmıştır.

Tablo 8.18 : Bit seviyesinde işlemler yapan komutlardan bazıları

İş kodu	Assembly komutu	Açıklama
0C	CLC	(MC6802) Elde bitini "0" yapar. (C=0 olur.)
0A	CLV	(MC6802) Taşma bitini temizler (V=0 olur.)
0E	CLI	(MC6802) Interrupt bayrağını "0" yapar.
0F	SEI	(MC6802) Interrupt bayrağını "1" yapar.
CB 5F	BIT 3,A	(Z80) A kaydedicisindeki D3 bitini Test eder. Eğer bu bit "0" ise CCR'deki Z biti "1" olur değilse "0" olur.
CB C0 05	SET 05H,B	(Z80) B kaydedicisindeki D5 bitini "1" yapar.

SET 05H,B komutunun işleyişi: Z80 mikroişlemcisinin komut setinde bulunan bu komut 2 byte'lık bir komuttur. Komutun işlem kodu "CB C0" dir. Komut hafızada (0100)₁₆ adresinden başlayarak 3 byte yer işgal eder. Bu komutla B kaydedicisindeki verinin 5. bitini "1" yaparız. Bu komuttan sonra CCR kaydedicisindeki bayraklar etkilenmez. Şekil 8.32'de komutun işleyişi ve kaydedicilerin durumu gösterilmiştir.

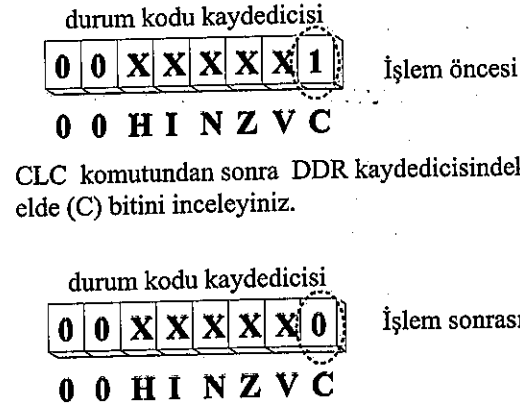


Şekil 8.32 : SET 05H,B komutu ile B kaydedicisindeki 5.bitin değiştirilmesi

CLC komutunun işleyişi: MC6802 mikroişlemcisinin komut setinde bulunur. CLC komutu ismini "CLear Carry" (elde'yi temizle) kelimelerinin baş harflerinden almıştır. "1" byte'lık bir komuttur ve işlem kodu "0C" dir. Komut hafızada (0100)₁₆ adresinden başlayarak 1 byte yer işgal eder. Bu komutla durum kodu kaydedicisindeki elde biti temizlenir yani (C; carry) "0" yapılır. Komutun işleyişi ve kaydedicilerin durumu şekil 8.33'de gösterilmiştir.

hafıza
adresi

0100	0C
0101	00
0102	00
0103	00
0104	00
0105	00
0106	00



Şekil 8.33 : CLC komutunun elde bitini temizlemesi

8.7 GİRİŞ/ÇIKIŞ KOMUTLARI

8.7.1 Giriş/Çıkış Komutları

Giriş/çıkış komutları ancak port yaklaşımlı mikro işlemcilerde bulunur. Bu tip mikro işlemciler adres yolunun alt veya üst sekiz bitini giriş/çıkış cihazlarına iletişimde bağlanmak için kullanır. Teorik olarak 8 bitlik bir adres yolu ile 256 tane giriş/çıkış cihazı ile iletişim sağlanabilirken pratikte özellikle 8 bitlik küçük sistemlerde bu durumla pek karşılaşılmaz. Buna karşılık giriş/çıkış için kullanılacak cihazların sayısı kullanılacak ek devre elemanları ile artırılabilir. Tablo 8.19'da port yöntemli giriş/çıkış tekniğini kullanan mikro işlemcilerin komutlarından örnekler verilmiştir. Bu yapıya uygun Z80 ve I8085 mikro işlemcileri örnek olarak verilebilir. Hafıza haritalı giriş/çıkış tekniğini kullanan mikro işlemciler ise MC6802 ve 6502 mikro işlemcileri verilebilir. Daha önce anlatıldığı gibi hafıza haritalı giriş/çıkış tekniği kullanan mikro işlemcilerde özel olarak kullanılan giriş/çıkış komutları bulunmamaktadır.

Tablo 8.19 : Port yaklaşımlı giriş/çıkış tekniğini kullanan mikro işlemci komutları

İş kodu	Assembly komutu	Açıklama
DB 84	IN A,(84)H	(Z80) A kaydedicisine 84 port adresindeki veriyi yükle.
ED 40	IN B,(C)	(Z80) mikro işlemciye bir kesme geldiğinde C kaydedicisinde belirtilen adresten B kaydedicisine veri yükler.
D3 1F	OUT A,(1F)H	(Z80) A kaydedicisindeki veriyi 1F portundaki aygıta gönderir.
DB 3F	IN A,(3F)H	(I8085) Belirtilen portdaki veriyi A kaydedicisine yükler
D3 44	OUT A,(44)H	(I8085) A kaydedicisindeki veriyi belirtilen porta yükler.

8.8 ALT PROGRAMLARA VE HİZMET PROGRAMLARINA İLİŞKİN KOMUTLAR

8.8.1 Bilgisayar Programlamada Alt Program Yöntemi

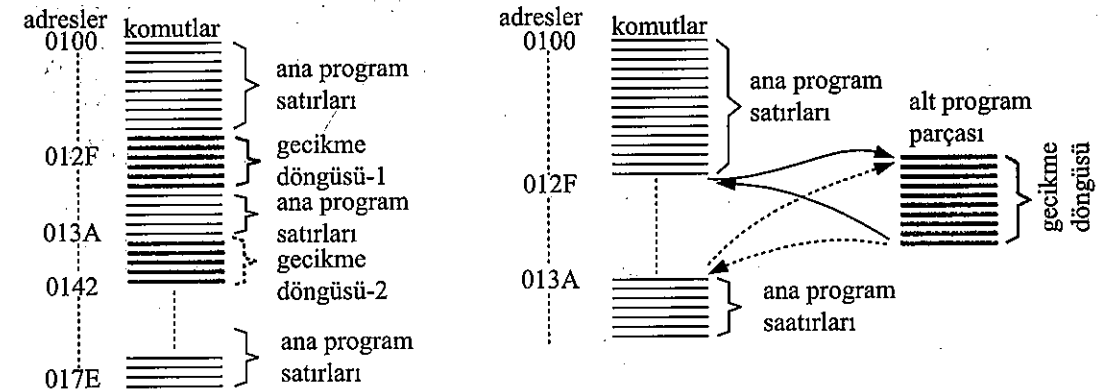
Programcı bir program hazırlamak istediğinde bazı işlemleri sıklıkla tekrar etmesi gerekebilir. Programcı hazırladığı programda çok sık kullanılan program kodlarını alt program şeklinde hazırlayabilir. Büyük bir program içerisinde tekrar tekrar kullanılabilen bu program parçasına "alt program" denir. Bilgisayarlar programlanırken alt programlar kullanmak, programcıya şu faydaları sağlar:

- ❖ Programın bütünü düşünüp dikkatini ve bilgisini bütün programa yoğunlaştırmaya kadar, dikkatini hazırlayacağı modül parçalara vererek daha verimli ve etkin programların hazırlanmasında etkin bir çalışma verimliliği sağlamış olur.
- ❖ Programın bütünü, parçalar halinde (modüler) düşünüp hazırlamak, programlama süresini kısaltır.
- ❖ Hata yapma oranını azaltır,
- ❖ Yapılan hataların tespitini kolaylaştırır,
- ❖ Hataların bulunma süresini kısaltır,
- ❖ Programın komut sayısını azaltarak hafızada daha az yer kaplaması sağlanmış olur,
- ❖ Programın iş yapma süresini kısaltır,

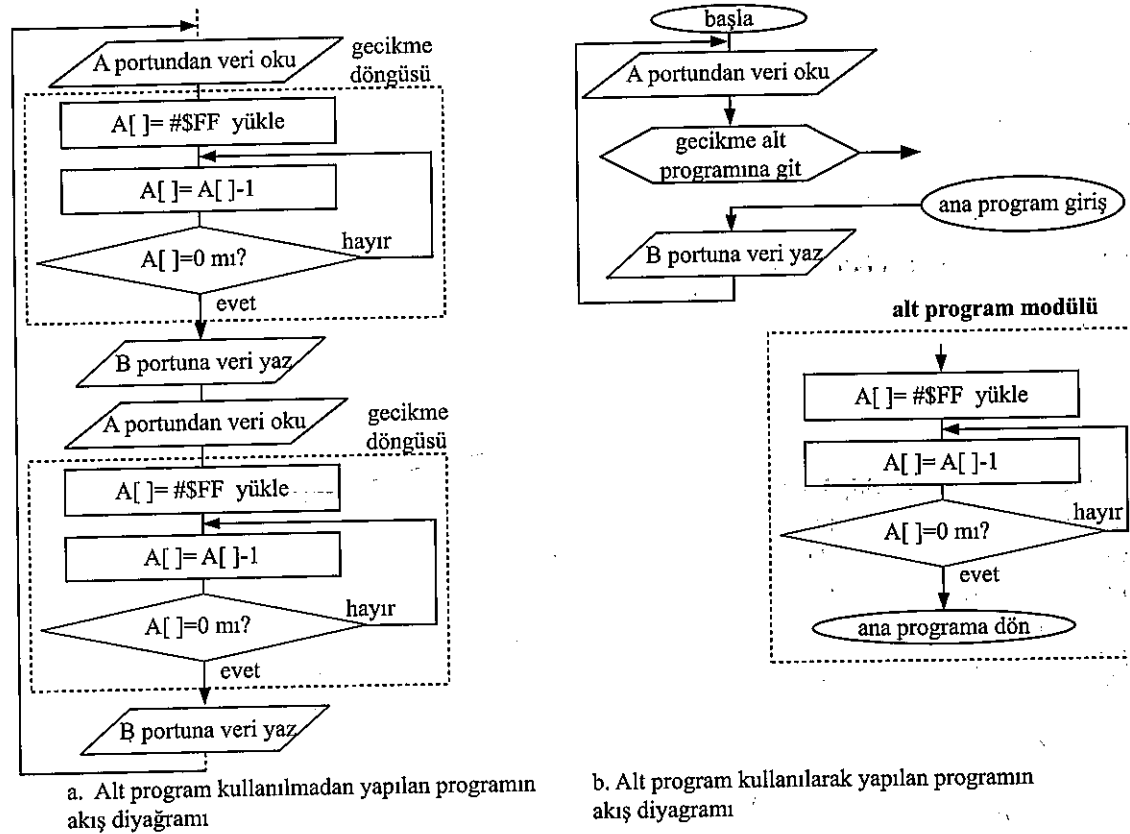
Programcı ve bilgisayar açısından birçok faydaları bulunan bu yöntemin bunların yanında birtakım zorlukları da vardır. Öncelikle alt program olarak hazırlanması düşünülen kısmın yapacağı iş tam olarak tanımlanmalıdır. Ana program içerisinde çağrılan alt programların geri dönüş bilgileri, yaptığı görevin sonuçları, alt programa dallanma ve geri gelme sırasında kaybolmamalıdır. Alt program kullanmak daha fazla dikkat ve modüler düşünme yeteneği ister.

Alt programlara niçin ihtiyaç olabileceğini bir örnekle açıklayalım. Programcılar birçok nedenlerden dolayı hazırladıkları program içerisinde kısa süreli gecikmeler elde etmek için boş döngüler kullanırlar. Gecikme döngüleri program içerisinde birçok kez kullanılacak ise bunu bir alt program modülü olarak hazırlayıp kullanmak ile alt program hazırlamadan kullanmak arasındaki farkı programların akış diyagramlarında da (şekil 8.34) açıkça görülmektedir.

Aşağıdaki program algoritmalarında A portundan veri okunduktan sonra bir süre beklenip daha sonra okunan bu verinin B portuna yüklendiği görülmektedir. Alt program kullanılmadan hazırlanacak programda, her seferinde A portu okunduktan sonra kısa bir gecikme elde etmek için aynı komutları tekrar tekrar yazmak program satırlarını uzatacağından dolayı programın hafıza kapladığı yer artacaktır. Ayrıca bu kadar çok komutun bulunduğu uzun bir programda programcının hata yapma oranı da artacaktır. Gecikme döngülerini alt program kullanarak elde etmek programımızı daha kısa ve kullanışlı yapacaktır. Ayrıca hazırlanan alt program gerektiğinde ana programın her yerinde kullanılabilecektir. Her iki programında aynı işi yapmasına rağmen yukarıda belirtilen yetersizliklerden dolayı alt program kullanmak mantıklı ve profesyonel bir çözüm olacaktır.



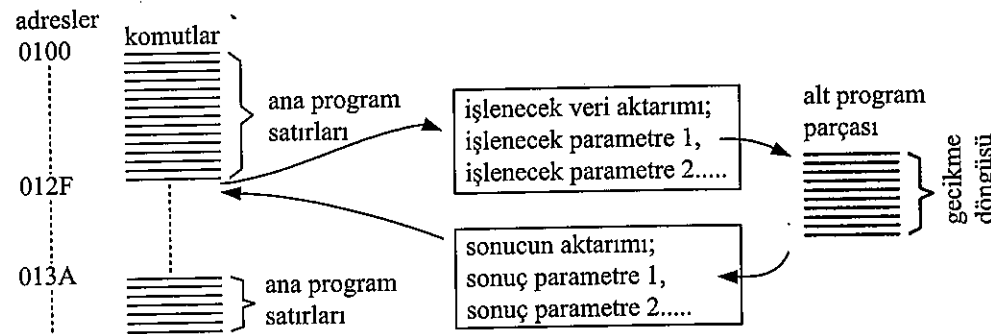
Şekil 8.34 : Ana ve alt program komut satırlarının hafızadaki yerleşimi



Şekil 8.35 : Metin şifreleme programının akış diyagramı

Alt program kullanılarak hem daha az komut kullanılır hem de program hafızada daha az yer kaplar. Verilen örnekte eğer A portundan 30 veri okunur ve her gecikme için yaklaşık 15 komut kullanılırsa $30 \times 15 = 450$ adet komut kullanılacak demektir. Her komutun hafızada "1" byte yer işgal ettiği düşünülürse bu da toplam 450 byte'lık bir hafıza demektir. Fakat gecikme döngüsü alt program olarak hazırlandığında bir kez kullanılacağı için hafızadan ortalama $450 - 15 = 435$ byte yer kazanılacaktır.

Alt programlar hazırlanırken alt programın görevi tam olarak tanımlanmalıdır. Örneğin yukarıdaki alt programın görevi "sabit süreli kısa bir gecikme" oluşturmaktır. Alt programlar görevlerini yerine getirmek için ana programdan birtakım veriler alıp, bunları işleyip ana programa sonuçlarını gönderecek şekilde düzenlenebilirler. Bu işlemlere parametre aktarımı denilir. Yukarıdaki gecikme alt programının süresini ana programda belirleme isteseydik bu sefer alt programa gecikme süresini bildiren bir değer göndermemiz gerekecekti.



Şekil 8.36 : Ana programda alt programa parametre aktarımı

Mikroişlemcilerde ana program içerisinde alt programları çağırmak ve alt programdan ana programa dönmek için kullanılan özel komutlar bulunmaktadır. Her alt program mutlaka geri dönüş komutu ile bitirilmelidir. Örneğin alt program çağrılmadan önce A kaydedicisine istenen değer konulur, daha sonra alt program çağırma komutu kullanılır (CALL). Alt programda A kaydedicisindeki veri üzerinde istenilen işlemler yapıldıktan sonra işlemin sonucunu bir kaydediciye yüklenir. Alt programın son komutuna ise ana programa geri dönmek ve programın çalışmasını sürdürebilmesini sağlamak için geri dönme komutu (RETURN) kullanılır.

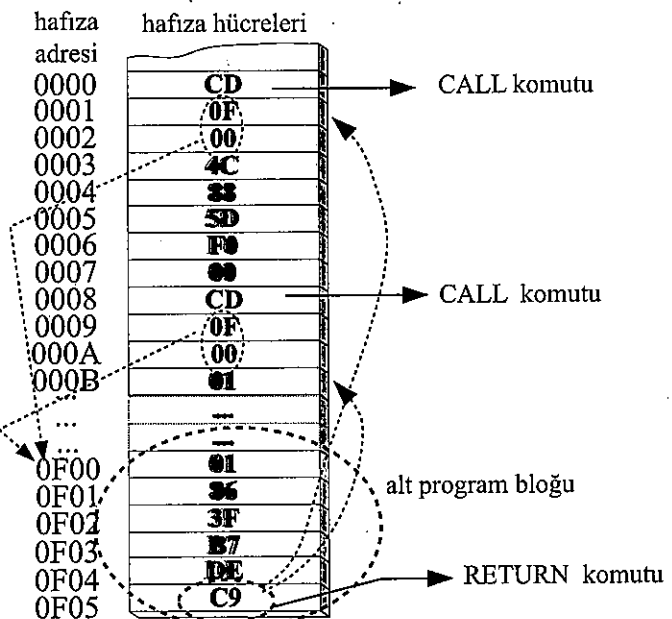
8.8.2 Alt Program Komutları

Programcılar alt programlama tekniğini kullanabilmeleri için bütün mikroişlemcilerde bununla ilgili komutlar bulunmaktadır.

Tablo 8.20 : Farklı mikroişlemcilerin komut setinde bulunan alt program komutları

İş kodu	Assembly komutu	Açıklama
CD 0F 00	CALL 0F00H	(Z80) 0F00 Adresindeki alt programı çağır.
CC 0F 00	CALL Z,0F00H	(Z80) Eğer Z biti 1 ise 0F00 adresindeki alt programı çağır.
E0	RET C	(Z80) Eğer C biti 1 ise ana programa geri dön.
CD 0F 00	CALL 0F00H	(I8085) 0F00 Adresindeki alt programı çağır.
C9	RET	(Z80) Alt programdan ana programa şartsız geri dön

CALL-RET komutunun işleyişi: Z80 mikroişlemcisinin komut setinde bulunan CALL ve RET komutları, şartsız olarak alt programı çağırma ve alt programdan geri dönme komutlarıdır. CALL komutu "3" byte'lık bir komuttur. İlk byte'ı "CALL" komutunun işlem kodudur "CD". CALL komutu ile birlikte alt programın bulunduğu adres belirtilmelidir. Diğer iki byte ise bu adresi için belirlenmiştir. Örnekte (0F00)₁₆ adresinde bulunan alt programı çağırmak için CALL 0F00H komutu kullanılmıştır. Komut (0100)₁₆ adresinde "3" byte yer tutmaktadır "CD 0F 00". Buradan alt programa dallanan mikroişlemci alt programdan RET komutu ile ana programa döner. RET komutunun işlem kodu "C9" dur ve hafızada "1" byte yer tutar. Şekil 8.37'deki örnekte, alt program iki kez çağırılmış ve her defasında alt programdan "RET" komutu ile ana programa geri dönlmüştür. Alt programdan ana programa geri döndüğünde, ana program eski kaldığı yerden çalışmasını sürdürür.



Şekil 8.37 : CALL-RET komutlarının işleyişi

8.9 MİKROİŞLEMCİLERDE YIĞIN İŞLEMLERİ VE KOMUTLARI

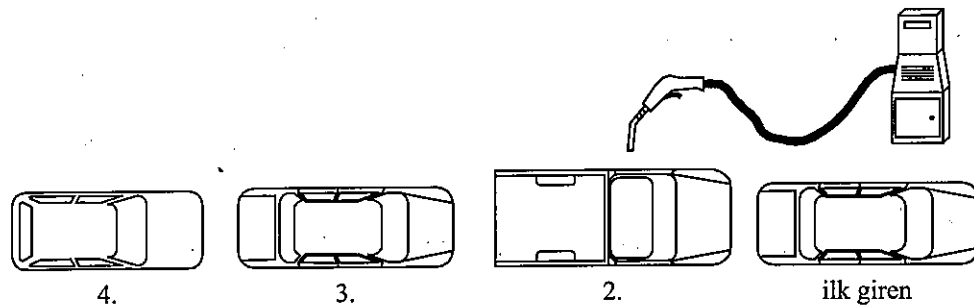
8.9.1 Yığın Çalışma İlkeleri

Mikroişlemciler herhangi bir programı yürütürken veya programcının ihtiyaç duyduğu durumlarda, bazı bilgilerin geçici olarak (hafızanın belli bir bölgesini depo olarak kullanarak) yükleme ihtiyacı duyarlar. Mikroişlemciler özellikle kesme ve alt programlara dallandığı zamanlarda hâlen kaydedicilerde bulunan bilgilerini daha sonra kullanmak üzere hafızanın belirtilen bir bölgesine yüklerler. Mikroişlemciler alt programlara veya kesme ihtiyaçlarına cevap vermek için o an kaydedicilerde bulunan bu bilgileri hafızanın belli bir bölgesine otomatik olarak yüklerler. Programcı ise programın işlevine göre bazı bilgileri saklamak ve daha sonra kullanmak üzere hafızanın belli bir bölgesini kullanır. Fakat programcı, saklama ihtiyacı duyduğu bilgileri hafızaya yüklemek için birtakım komutlar kullanarak mikroişlemciye bildirmek zorundadır. Bu gibi durumlar için mikroişlemcilerin komut setinde bulunan yığın işleme komutları kullanılabilir. Yığın kavramının nasıl kullanıldığı aşağıda incelenmiştir.

Yığın, hafızanın belli bir bölgesini depo olarak kullanma ihtiyacından doğmuştur. Mikrobilgisayarda oku/yaz belleğin (bunun için genellikle RAM hafıza seçilir.) bir kısmını bu iş için ayırmamız gerekmektedir. Bazı mikroişlemciler ise yığın kullanımı için kendi içerisinde bir miktar oku/yaz hafızası bulundurabilmektedirler. Eğer mikroişlemci içerisinde böyle bir yığın alanı yok ise bizim bu işlem için hafızanın hangi bölgesini yığın işlemlerinde kullanacağımızı mikroişlemciye bildirmek zorundayız. Yığın için kullanılacak hafıza bölgesinin başlangıç adresini mikroişlemciye bildirmek için mikroişlemcinin yığın kaydedicisini istenen adres ile yükleme yapmamız gerekir. Yığını kullanabilmek için yapılacak ilk iş, yığın olarak kullanılacak hafıza bölgesinin başlangıç adresini mikroişlemcinin yığın kaydedicisine yüklemek olmalıdır. Her mikroişlemcide yığın kaydedicisine ilk değeri yükleyebilmek için kullanılabilecek komutlar bulunur. Mikroişlemcilerde kullanılan yığın olayı iki şekilde işleyiş gösterebilir. Bunlar:

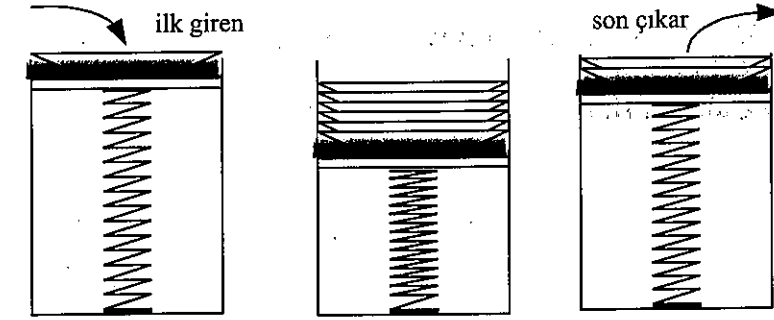
- 1-FIFO (First In First Out; İlk giren ilk çıkar) kuralı ile
- 2-LIFO (Last In First Out; Son giren ilk çıkar) kuralı ile

FIFO kuralı ile işleyen bir yığında ilk giren veri ilk olarak çıkacaktır demektir. Yığın olayında, verilerin yığına yüklenmesi yerine “yığına veri itilmesi” ifadesi, yığından verilerin okunması yerine ise “yığından veri çekilmesi” ifadesinin kullanılması daha yaygındır. FIFO kuralını anlatan benzer bir işleyişin şekil 8.38’de gösterilmiştir.

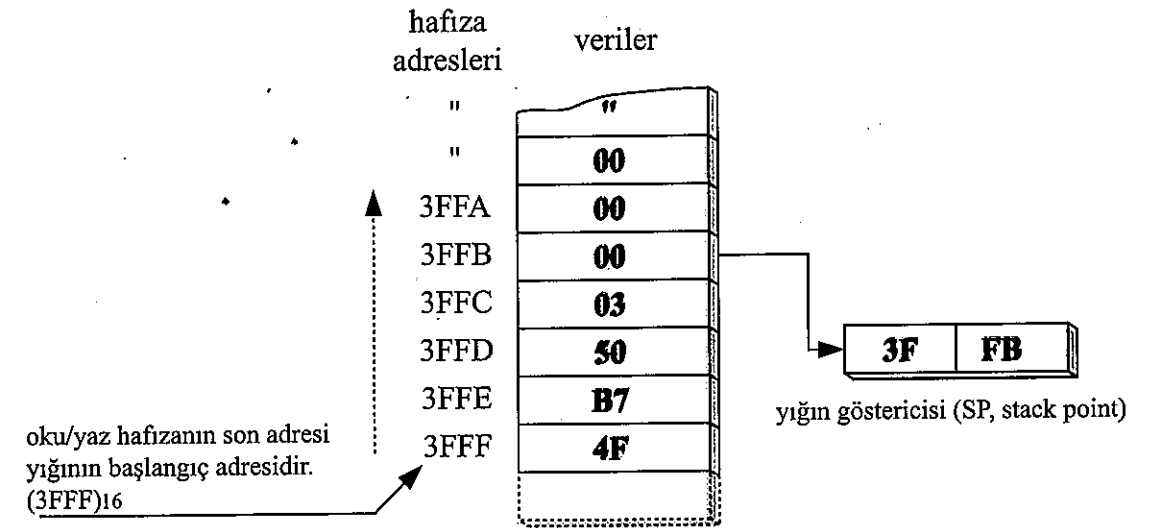


Şekil 8.38 : FIFO kuralını anlatan çalışma şekli

LIFO yöntemi ile çalışan bir mikroişlemci yığını, şekil 8.39’dakine benzer bir teknik kullanır. Üst üste konulan tabaklardan önce konulan tabak en altta kalır. Bu tabağın çıkması için üstündeki tabakların bir bir alınması gerekmektedir. Mikrobilgisayarlarda yığına itilen verilerinde üst üste gelmesinde aynı durum gerçekleşir. Çoğu mikroişlemcilerde bu tipte bir yığın sistemi kullanılmıştır. Onun için programcıların yığına ittikleri verilerin sırasını karıştırmaması ve yığından verileri ona göre çekmeleri gerekmektedir.



Yığın kullanmak için önce yığın olarak kullanılacak hafıza bölgesinin adresinin mikroişlemcinin yığın kaydedicisine yüklenmesi gerekmektedir. Belirtilecek bu adresin sistemdeki oku/yaz belleğin son adresi olmasında fayda vardır. Çünkü yığınların işleyişine göre yığına her itildiğinde, bu yığın sayısının bir azalmasına ve hafızanın daha içerisine doğru kayılmasına neden olur. Böylece ne kadar veri yüklenirse yüklensin yığının alabileceği maksimum veri sistemdeki oku/yaz belleğin miktarı kadar olacaktır. Şekil 8.40’da yığın için kullanılan hafızanın ve yığın kaydedicisinin durumu görülmektedir. Yığın kaydedicisi her zaman yüklenecek verinin yerini gösterecek şekilde bekler. Yığına her veri itildiğinde yığın kaydedicisi otomatik olarak bir azalır, yığından her veri çekildiğinde ise yine otomatik olarak bir artar.



Şekil 8.40 : Hafızanın yığın işlemi için kullanılması ve yığın kaydedicisinin durumu

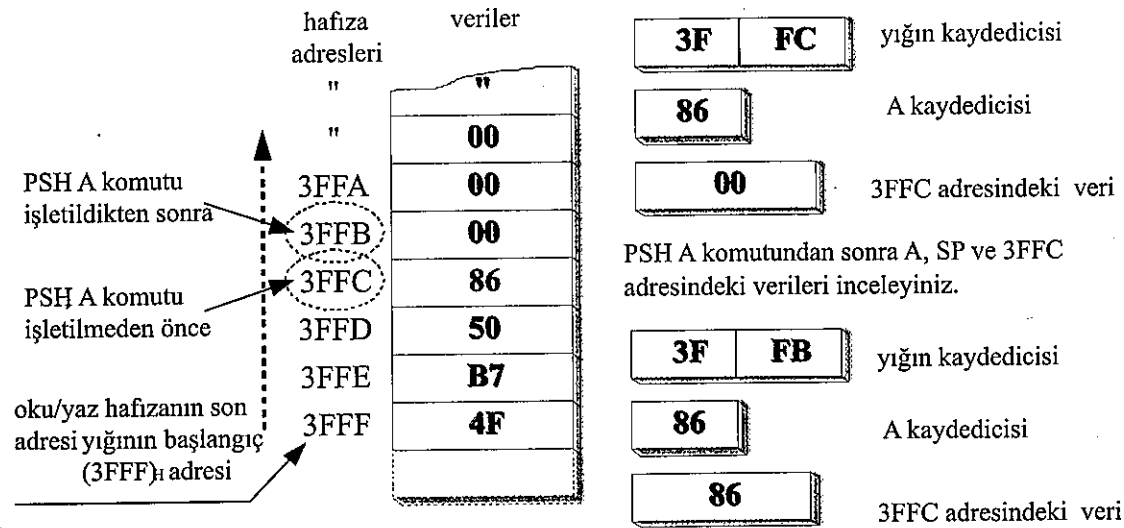
8.9.2 Yığın İşleme Komutları

Değişik mikroişlemcilerin komut setlerinde yığın kullanımını sağlamak için komutlar bulunur. Tablo 8.21’de bazı mikroişlemcilerin komut setlerinde bulunan ve yığın işlemlerinde kullanılan komutlardan örnekler verilmiştir.

Tablo 8.21 : Yığın işlemlerinde kullanılan değişik mikroişlemcilerin komutları

İş kodu	Assembly komutu	Açıklama
F9	SPHL	(I8085) Yığın kaydedicisine HL kaydedicisindeki değeri yükler.
E3	XTHL	(I8085) HL'nin gösterdiği adresteki veri ile yığının en üstündeki veriyi yer değiştirir.
C5	PUSH B	(I8085) B kaydedicisindeki veriyi yığına iter.
C1	POP B	(I8085) Yığının en üstündeki veriyi B kaydedicisine yükler.
8E 3F FF	LDS #3FFF	(MC6802) Yığın kaydedicisini (3FFF) ₁₆ verisi ile yükler ve yığın (3FFF) ₁₆ adresine kurulmuş olur.
36	PSH A	(MC6802) Yığına A kaydedicisindeki veriyi iter.
37	PUL A	(MC6802) Yığının en üstündeki veriyi A kaydedicisine çek.

PSH A komutunun işleyişi: MC6802 mikroişlemcisinin komut setinde bulunur. PSH A komutu A kaydedicisindeki veriyi yığına iter. Örneğin MC6802 mikroişlemcisinin komut setine göre hazırlanan programı (tablo 8.22) inceleyecek olursak yığın kaydedicisinin daha programın başında RAM hafızanın (3FFF)₁₆ adresini gösterecek şekilde yüklendiği görülecektir. Daha sonra ihtiyaç duyulduğunda PSH A komut ile A kaydedicisinin içeriği yığına yüklenmiştir. Mikroişlemci PSH A komutu ile karşılaştığında şu işlemleri gerçekleştirir. İlk olarak yığın kaydedicisinin gösterdiği adrese A kaydedicisindeki veriyi yükler, daha sonra yığın kaydedicisinin içeriğini "1" azaltır. Bu komutla ilgili program parçası ve komutun işleyişi şekil 8.41'de gösterilmiştir.



Şekil 8.41 : Yığına veri yüklenmesi için kullanılan PSH A komutunun işleyişi

Programda yığın kaydedicisi ilk olarak (3FFF)₁₆ adresini gösterecek şekilde yüklenmiştir. Daha sonra programda kullanılan PSA komutları ilk yığına birtakım bilgiler itilmiş veya çekilmiştir. Son olarak yığın kaydedicisi (3FFC)₁₆ yi göstermektedir. En son kullanılan PSH A komutu ile yığına A kaydedicisindeki (86)₁₆ verisi itilmiştir. Bu komuttan sonra artık yığın kaydedicisi (3FFB)₁₆ adresini gösterecek şekilde bir azalmıştır. Eğer yığına 2 byte'lık bir veri itilmiş olsa idi yığın kaydedicisi iki kez azalacaktı. Yani (3FFB)₁₆ 'yi değil de (3FFA)₁₆ 'yi gösterecekti.

Tablo 8.22 : Yığın komutların bulunduğu program parçası

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	8E 3F FF		LDS #3FFF F	Yığın kaydedicisi daha programın başlarında RAM hafızanın (3FFF) ₁₆ adresine kurulmuştur.
.....				Bu arada program komut kodları vardır. Bu kodların yığın kaydedicisine başlangıçta (3FFF) ₁₆ olarak yüklenen değeri, (3FFC) ₁₆ olacak şekilde etkilediğini varsayalım. Yığın kaydedicisi yığına yükleme yapıldıkça azalır.
010A			PSH A	A kaydedicisindeki değer yığına itilir.

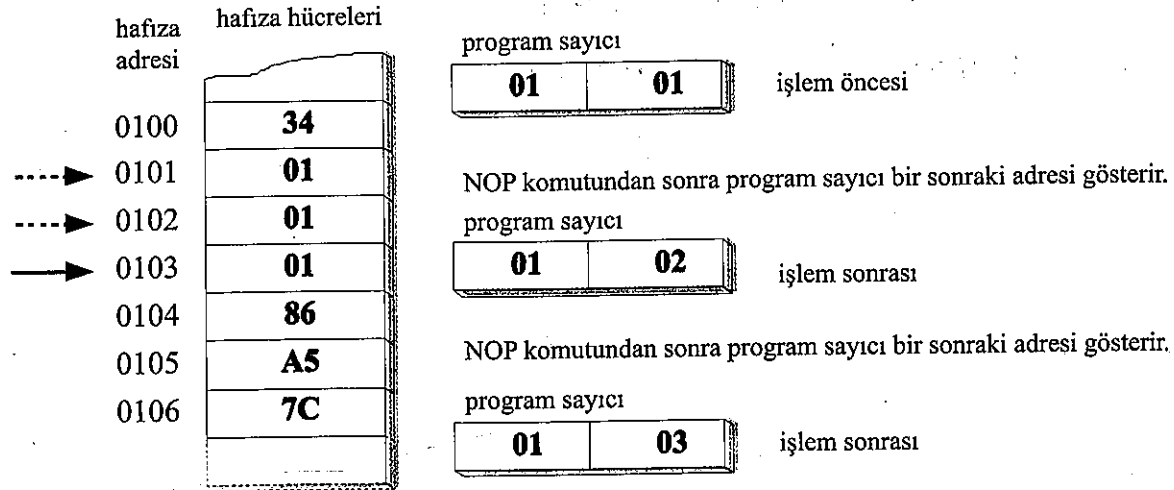
8.10 KONTROL KOMUTLARI

Her mikroişlemcinin komut setinde, kendisinin normal çalışma işleyişini doğrudan etkileyen bazı özel komutlar bulunur. Bu komutlardan bazıları mesela kullandığımız kişisel bilgisayarların üzerinde bulunan "reset" düğmesine bastığımız zaman, nasıl ki bilgisayar kapanıp tekrar açılıyorsa yani "resetleniyorsa" aynı işlemi yapan komutlar da vardır. Şimdi bu gibi komutları inceleyelim.

Tablo 8.23 : Mikroişlemcilerin komut setinde bulunan, bazı özel işlem komutları

İş kodu	Assembly komutu	Açıklama
00	NOP	(I8085) İşlem yapma bir sonraki komuta geç
76	HLT	(I8085) Mikroişlemci çalışmasını durdurur. Tekrar çalışması için resetlenmesi gerekmektedir.
DF	RST 3	(I8085) Donanım kesmesine benzer kesme oluşmasını yazılım komutları ile gerçekleştirir. RST 3 komutu ile 3' nolu kesme vektörüne dallanılır.
20	RIM	(I8085) Kesme maskesini oku.
F3	DI	(I8085) Kesmeleri yetkisiz hâle getir. (Göz ardı et).
FB	EI	(I8085) Kesmeleri yetkilendir. (Gelen kesmelere cevap ver.)
3E	WAI	(MC6802) Kesme sinyali bekle komutu.
3F	SWI	(MC6802) Yazılım kesmesi. Mikroişlemci yazılım kesme komutu ile karşılaştığında çalışmasını yazılım kesme vektöründen devam eder.
01	NOP	(MC6802) Hiçbir işlem yapma bir sonraki komuta geç
76	HALT	(Z80) Mikroişlemci resetleyinceye kadar hiç işlem yapma.
C7	RST 0	(Z80) Mikroişlemciyi "0" nolu kesme vektörü konumundan yeniden başlat.
00	NOP	(Z80) Hiç işlem yapma ve bir sonraki komutla çalışmaya devam et.

NOP komutunun işleyişi: MC6802 mikroişlemcisinin komut setinde bulunan bu komut mikroişlemciye herhangi bir eylem yaptırmaz. Bu komut mikroişlemcinin durumu üzerinde hiçbir değişiklik yapmaz. Durum denetim kaydedicisindeki bayraklar etkilenmez. Mikroişlemci yalnızca birkaç clock palsı süresince boşa çalışır ve program sayıcı bir artarak bir sonraki komutla çalışmasına devam eder. Bu komut daha çok zaman geciktirme alt programlarında kullanılır. NOP komutunun her kullanılmasında, belli bir süre harcandığı için bu komutun özellikle bir döngü içerisinde kullanılması istenen sürenin elde edilmesinde büyük kolaylık sağlamaktadır. Şekil 8.42'de NOP komutunun işleyişi gösterilmektedir. Komutun işlem kodu "01" dir.



Şekil 8.42 : NOP komutunun işleyişi

Mikroişlemci (0101)₁₆ adresinde karşılaştığı NOP komutu ile hiçbir işlem yapmadan bir sonraki komutu işletmek üzere program sayıcıyı artırır. Aynı komutla (0102)₁₆ ve (0103)₁₆ adreslerinde de ardı ardına karşılaşacaktır ve her seferinde hiçbir işlem yapmadan yalnızca bir sonraki komutu çalıştırmak üzere program sayıcıyı arttıracaktır. Mikroişlemcinin (0104)₁₆ adresinde, işleyip eyleme dönüştürebileceği bir komut kodu bulunmaktadır. Bu komutun kodu "86"dır ve mikroişlemciye kendisinden sonra bulunan (A5)₁₆ verisini A kaydedicisine yüklemesini emreder. Böylelikle mikroişlemci, program kodlarını işleterek çalışmasına devam eder.

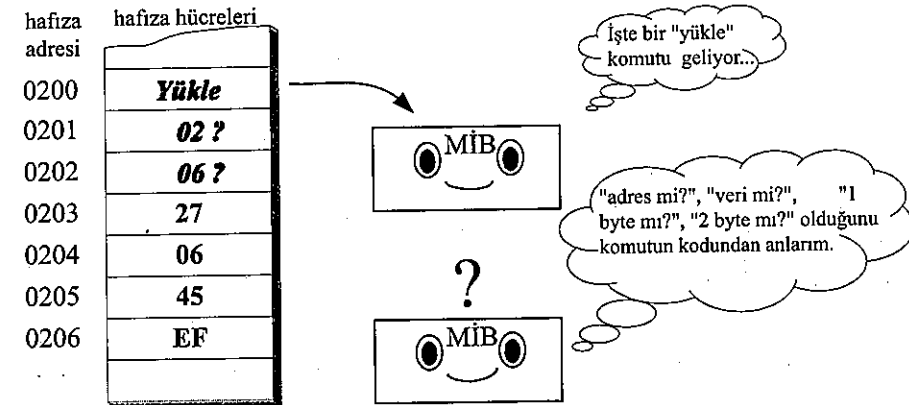
DEĞERLENDİRME ÇALIŞMALARI

- 1) Mikroişlemciler çalışırken neden komutlara ihtiyaç duyarlar?
- 2) Yükleme komutlarına örnek vererek bunları bir iki komutun çalışmasını anlatarak açıklayınız.
- 3) MC6802 mikroişlemcisindeki, 8 bitlik yükleme komutlarına örnek vererek çalışmasını bir komut üzerinde anlatınız.
- 4) Z80 mikroişlemcisindeki 16 bitlik yükleme komutları ile hangi kaydediciler üzerinde işlemler yapılmaktadır? Bu komutları örnek vererek açıklayınız.
- 5) Aritmetiksel ve mantıksal işlem gurubundaki komutlar ile yapılabilecek işlemleri yazınız.
- 6) Kaç tür dallanma komutu vardır?
- 7) Herhangi bir şarta bağlı olarak dallanan komutlar ile ileriye veya geriye doğru ne kadar dallanılır?
- 8) Göreceli dallanma komutlarının bağlı adresleri nasıl hesaplanır?
- 9) Kaydırma ve döndürme gurubu komutları ile ne tür işlemler yapılmaktadır? Bu gruptaki komutlardan bir örnek vererek komutun işleyişini anlatınız.
- 10) Bilgisayarlar programlanırken hangi durumlarda alt program kullanma gereği duyulur? Alt programları kullanmak programcıya ne gibi yararlar sağlar?
- 11) Alt programlardan ana programa dönüştü nele dikkat etmek gerekir?
- 12) Mikroişlemcilerde kaç çeşit yığın mekanizması vardır?
- 13) Verileri gerektiğinde yığına itmek veya çekmek programcıya ne gibi yararlar sağlar?
- 14) Kontrol komutları mikroişlemcinin çalışmasını nasıl etkiler?

ÜNİTE 9 ADRESLEME MODLARI

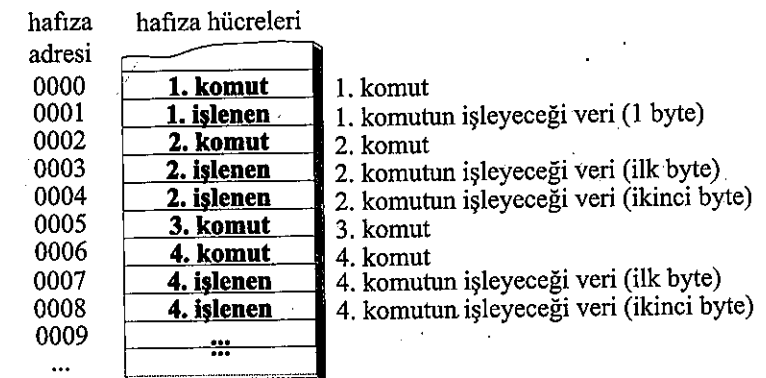
9.1 ADRESLEME MODU KAVRAMI

Adresleme Modu, mikroişlemcinin o anda yürütmekte olduğu programın komutlarına göre, komut ve verilerin, hafızadan hangi yöntemle getirileceğinin ve işleneceğinin genel bir tanımınıdır. Mikrobilgisayarlar için hazırlanan assembly program satırlarında, mikroişlemcinin tanıyıp işleyeceği komutlar ve veriler vardır. Herhangi bir komut, kendisi ve işleneni ile birlikte bir bütündür. Komuttan sonraki işlenen bir "adres" olabileceği gibi direkt olarak kullanılabilecek bir "veri" de olabilir. Ayrıca işlenenin 1 byte veya 2 byte büyüklüğünde olması duruma göre değişebilir. Komuttan sonra işlenen veri mi?, adres mi?, "adres" veya "veri" ise bu 1 byte mı?, 2 byte mı? mikroişlemci bunu nasıl tanımlayacak. Görüldüğü gibi, herhangi bir komutun işleneni, değişik türden bilgiler olabilmektedir. İşlenendeki bu çeşitlilik, komutlar açısından birtakım yöntemlerin kullanılmasını gerekli kılmıştır. Mikroişlemci açısından sorunun ne olduğu Şekil 9.1'de temsili olarak gösterilmiştir. Eğer komutlar için adresleme modu yöntemi kullanılsaydı, mikroişlemci komuttan sonraki işleneni tanımlayamayacaktı.



Şekil 9.1 : Mikroişlemciler komut kodlarına göre işleneni belirlerler.

Mikroişlemciler ilk çalışmaya başladığı andan itibaren, hafızadan önce komut okumak zorundadır. İlk komutla birlikte eğer komutun bir işleneni varsa hafızadan bu değer de okunarak komut işleme süreci başlamış olur. Komuttan sonra mikroişlemcinin tanımlaması gereken diğer bir durum ise işlenenin kaç byte'lık bir işlenen olduğudur. Daha sonra bu işlenenin bir veriyi mi? yoksa bir adresi mi? gösterdiği mikroişlemci tarafından tanımlanabilmelidir. Aslında bu noktada programcıların böyle bir problemin çözümüne kafa yormalarını gerektirecek bir durum yoktur. Zaten mikroişlemciler tasarlanırken daha işin başında bu problemler "adresleme modu" kavramı ile aynı işi yapan farklı komut kodları kullanılarak çözümlenmiştir.



Şekil 9.2 : Komutların ve işlenenlerin hafızadaki yerleşimi

Adresleme modu mantığını daha net kavramak için tablo 9.1'i inceleyelim. Örneğin, MC6802 mikroişlemcisinin komut setinde bulunan ve "A" kaydedicisine veri yüklemeye kullanılan "LDAA" komutunu ele alalım. A kaydedicisine bir veriyi yüklemek istediğimizde bu komutu kullanmak zorundayız. Fakat LDAA komutunun değişik durumlarda kullanılan birden fazla komut kodu vardır. Herhangi bir komutun, kodlarının değişik işlenenlerle nasıl kullanıldığı tablo 9.1'de görülmektedir. Tabloda görüleceği gibi herhangi bir komutun değişik işlenenler için farklı kodları bulunmaktadır. Herhangi bir komutun değişik işlenenler için farklı komut kodları kullanması, adresleme modu kavramı ile izah edilebilir.

Tablo 9.1 : MC6802'deki LDAA komutunun değişik işlenenlerle kullanılması

Adresleme modu	İş kodu	Assembly komutu	Açıklama
İvedi Adresleme (Immediate Addressing)	86 FE	LDAA #\$FE	A Kaydedicisine (FE) ₁₆ sayısını yükler.
Doğrudan Adresleme (Direct Addressing)	96 3F	LDAA \$3F	A Kaydedicisine (003F) ₁₆ adresinden 1 byte veri yükler.
Doğal Adresleme (Absolute Addressing)	B6 1F 00	LDAA \$1F00	A Kaydedicisine (1F00) ₁₆ adresinden 1 byte veri yükler.
Sıralı Adresleme (Indexed Addressing)	A6 05	LDAA \$05,X	X kaydedicisine (05) ₁₆ eklendiğinde elde edilen adresten 1 byte veri yükler.

Gerçekte LDAA komutunun her seferinde yaptığı iş sonuç olarak aynıdır ve bu iş "A" kaydedicisine bir veri yüklemektir. Fakat LDAA komutunun kullanabileceği dört farklı iş kodu bulunmaktadır. Bu durum işlenenlerin farklılığından ileri gelmektedir. Mikroişlemci, program hafızasından komutun "işlem kodunu" getirip "komut kaydedicisine" (instruction register) yükledikten sonra komutun işlevi, komut kod çözücüsü (instruction decoder) tarafından çözülür. Böylece işleneninde ne tür bir işlenen (1 byte veri, 2 byte adres, 2 byte veri gibi) olduğu mikroişlemci tarafından bilinecektir.

İvedi adresleme (Immediate Addressing) modunda; LDAA #\$FE komutuna karşılık "86" işlem kodu kullanılmaktadır. Mikroişlemci "86" işlem kodunun şifresini çözdüğünde, komuttan sonra "1 byte'lık veri" olduğunu (FE)₁₆ tanımlar ve bu veriyi A kaydedicisine yükler.

Doğrudan adresleme (Direct Addressing) modunda; LDAA \$3F komutunun işlem kodu ise "96" dir. Bu işlem kodu çözüldüğünde komuttan sonra "1 byte'lık adres" (3F)₁₆ olduğu tanımlanarak A kaydedicisine bu adresteki veriyi yüklemek için işlemlere başlar.

Doğal adresleme (Absolute Addressing) modunda; LDAA \$1F00 komutunun işlem kodu ise "B6" dir. Mikroişlemci, bu işlem kodunu çözdüğünde ise komuttan sonra "2 byte'lık adres" (1F00)₁₆ olduğunu tanımlar ve A kaydedicisine de bu adreste bulunan veriyi yükler.

Sıralı adresleme (Indexed Addressing) modunda; LDAA \$05,X komutunun işlem kodu ise "A6" dir. Mikroişlemci bu işlem kodunu çözdüğünde ise komuttan sonra "1 byte'lık veri" (05)₁₆ olduğunu tespit eder. A kaydedicisine yükleme yapmak için ise bu verinin indeks kaydediciye eklenmesiyle elde edilecek adresten getirileceğini belirler.

Assembly programlama tekniğinde, komutların farklı işlenenlerle (adres, veri) iş yapabilmelerini sağlamak için bir veya birkaç farklı işlem kodu kullanılması yoluna gidilmiştir. Mikroişlemci üreticileri, ürettikleri mikroişlemcilerin komut setlerinde, her komutun değişik adresleme modlarında kullanılan işlem kodlarını programcılara sunmuşlardır. Tablo 9.2'de birkaç mikroişlemcinin komut setinde bulunan adresleme modları listelenmiştir. Mikroişlemcilerin bazıları bu adresleme modlarından hepsini desteklerken bazıları da yalnızca bir mikroişlemci desteklemektedir. Programcı kullandığı mikroişlemcinin el kitabından veya değişik kaynaklardan kullanabileceği adresleme modlarını öğrenmesi gerekmektedir.

Tablo 9.2 : Mikroişlemcilerin komut setlerinde bulunan değişik adresleme modları

Adresleme modu			Bu adresleme modunu kullanan mikroişlemciler
İngilizce	Türkçe	Açıklama	
Immediate addressing	Veri tanımlı veya ivedi adresleme	İşlenen veri	MC6802, Mos. Tec. 6502, Intel 8085
Zero Page addressing Direct addressing	Sıfırıncı sayfa adresleme veya direkt adresleme	İşlenen (00) ₁₆ (FF) ₁₆ arasındaki bir adres (1 byte'lık)	Mos. Tec. 6502 MC6802
Indexed addressing	İndeksli adresleme	Sıralı işlemlerde kullanılan adresleme	MC6802, Mos. Tec. 6502, Intel 8085
Absolute addressing Extended addressing	Mutlak, genişletilmiş veya doğrudan adresleme	İşlenen 2 byte'lık adres	Mos. Tec. 6502 MC6802
Accumulator, Implied, Inherent addressing	Kaydedici veya imalı adresleme	Belli bir kaydedici için yapılan işlemler.	Mos. Tec. 6502, Intel 8085 MC6802
Relative addressing	Göreceli adresleme	Herhangi bir adresi referans olarak yapılan adreslemeler.	MC6802, Mos. Tec. 6502, Intel 8085
Indirect addressing	Dolaylı adresleme	Dolaylı yolla belirlen adresleme şekli.	Mos. Tec. 6502
Indexed-Indirect addressing	İndisli-dolaylı adresleme	Sıralı dolaylı adresleme	Mos. Tec. 6502
Indirect-Indexed addressing	Dolaylı-indisli adresleme	Dolaylı sıralı adresleme	Mos. Tec. 6502

Tablo 9.3 : MC6802'in bazı komutlarından değişik adresleme modlarındaki işlem kodları

Mnemonic komut	İvedi adresleme	Direkt adresleme	İndeks adresleme	Genişletilmiş adresleme	İmalı adresleme	Göreceli adresleme	Açıklama
ADDA	8B	9B	AB	BB	-	-	A+M → A
ABA	-	-	-	-	1B	-	A+B → A
ANDA	84	94	A4	B4	-	-	A.M → A
CLRB	-	-	-	-	5F	-	00 → B
DECB	-	-	-	-	5A	-	B - 1 → B
INC	-	-	6C	7C	-	-	M+1 → M
LDAA	86	96	A6	B6	-	-	
LDX	CE	DE	EE	FE	-	-	M _A X _H , (M+1) → X _L
BRA	-	-	-	-	-	20	
BEQ	-	-	-	-	-	27	Eğer Z=1 ise
NOP	-	-	-	-	-	01	İşlem yapma

Aynı türden işlenenler bir gruba dâhil edildiğinde "adresleme modu" (komut kodlama çeşidi) ortaya çıkar ve 6 temel adresleme modu vardır. Adresleme modları her işlemcilerde farklı sayıda ve özellikte olabilir. Motorola firmasının ürettiği MC6802 mikroişlemcisi 6 adresleme modu, MOS Tec. firmasının ürettiği 6502 mikroişlemcisi 9 adresleme modu, Zilog Z80 mikroişlemcisi ise 10 farklı adresleme modu kullanmaktadır. Fakat bir çok mikroişlemcideki fazladan bulunan adresleme modları hemen hemen bu 6 temel adresleme modundan türetilmiştir. Programcı kullandığı mikroişlemcinin desteklediği adresleme modlarını mikroişlemcinin teknik dokümanlarını inceleyerek araştırmalı ve bu adresleme modlarının nasıl kullanıldığını bilmelidir.

Motorola MC6802 işlemcisi 72 tane ana komuta sahiptir. Bu mikroişlemcinin 72 ana komutun, değişik adresleme modlarında kullanılan kodları sayıldığında, 197 tane farklı işlem kodu bulunduğu görülür. Teorik olarak 8 bitlik veri işleme kapasitesine sahip bir mikroişlemcinin kullanılabileceği komut sayısı en fazla $2^8=256$ tane olabilir. Zilog Z80 mikroişlemcisinin komut setinde 162 ana komut bulunmaktadır. Bu komutların farklı adresleme modlarındaki işlem kodlarının toplamı ise 1124 adettir. Bu sayının çok olmasının sebebi Zilog Z80 mikroişlemcisinde bazı komutların işlem kodları 2 byte'lıktır. Teorik olarak işlem kodlarının 2 byte ile gösterilmesi, toplam 65536 tane işlem kodunun kullanılabilmesi demektir. Motorola MC6800 ailesinin 6 adet adresleme moduna karşılık Zilog Z80 de 10 farklı adresleme modu kullanılmaktadır. MOS Tec. 6502 mikroişlemcisinde ise 13 temel adresleme modu kullanılmaktadır.

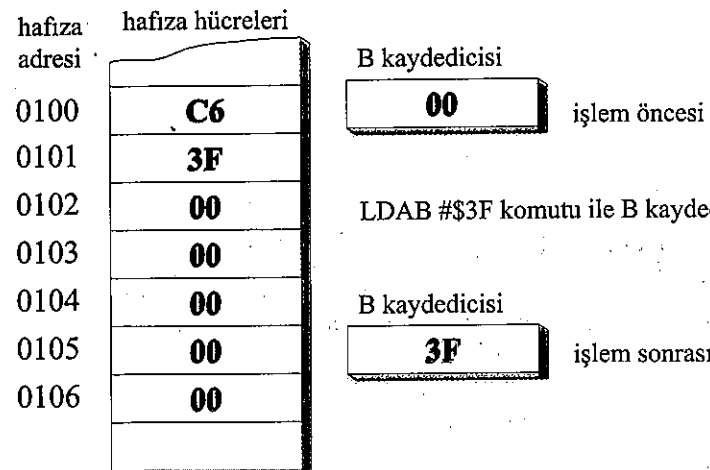
9.2 İVEDİ ADRESLEME (Immediate Addressing)

İvedi adresleme modunda kullanılan komutun işleneni bir veridir. Bu adresleme modu “veri tanımlı adresleme modu” şeklinde de söylenir. Kullanılan veri 1 veya 2 byte olabilir. İşlenecek verinin kaç byte olacağı komutun çeşidine göre değişir. Bazı komutlar 1 byte veri kullanırken bazıları 2 byte veri ile çalışırlar. Komutun kullanacağı verinin büyüklüğü, mikroişlemcinin üretimi sırasında, komut kodu çözücü biriminde tanımlanmıştır.

Tablo 9.4 : Bazı mikroişlemcilerin ivedi adresleme modunda kullanılan komutları

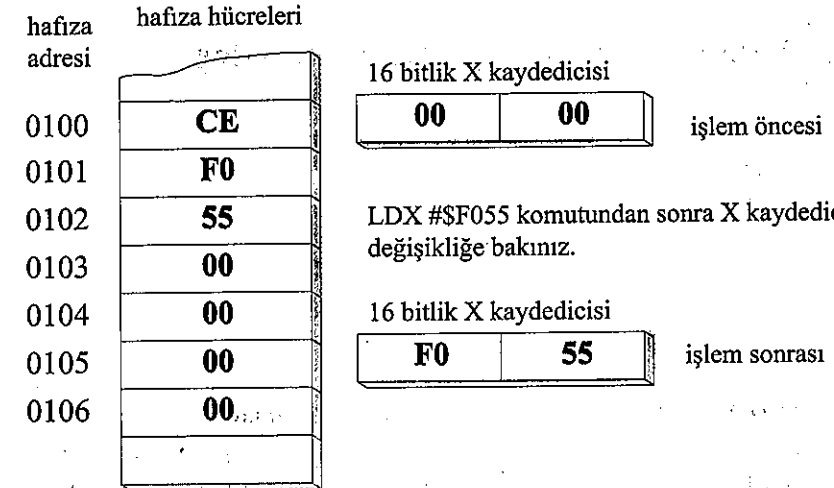
İş kodu	Assembly komutu	Açıklama
86 FE	LDAA #\$FE	(MC6802) A Kaydedicisine (FE) ₁₆ sayısını yükler
C6 3F	LDAB #\$3F	(MC6802) B Kaydedicisine (3F) ₁₆ sayısını yükler.
CE F0 55	LDX #\$F055	(MC6802) İndeks kaydedicisine (F055) ₁₆ sayısını yükler.
A9 36	LDAA #\$36	(MOS Tec. 6502) A kaydedicisine (36) ₁₆ sayısını yüklenir.

LDAB #\$3F komutunun işleyişi: LDAB #\$3F komutu B kaydedicisine (3F)₁₆ verisini yükler. Komutun işlem kodunun, hangi adresleme modundaki kod olacağı, işlenenden anlaşılır. LDAB #\$3F komutun işleneni bir veridir. Eğer komuttan sonra bir veri kullanılırsa bu komutun ivedi adresleme modunda kullanılmasına işaret eder. LDAB #\$3F komutunun ivedi adresleme modundaki işlem kodu “C6” dir. Komutun işleyişi şekil 9.3’de gösterilmiştir.



Şekil 9.3 : İvedi adresleme modunda LDAB #\$3F komutunun işleyişi

LDX #\$F055 komutunun işleyişi: Komutlar, ivedi adresleme modunda kullanıldığında 2 byte'lık veriler üzerinde de işlemler yapılabilirler. Örneğin, 2 byte'lık indeks kaydedicisine bir veri yüklemek istediğimizde, komuttan sonra 2 byte'lık bir işlenenin olması gerekir. İşlenenin bir veri olmasından dolayı da komut ivedi adresleme modunda kullanılmalıdır. Örneğin İndeks kaydedicisine (F055)₁₆ bir veri yüklemek istendiğinde bu işlem için LDX komutu kullanılmalıdır. LDX komutu eğer bir veri yükleyecekse bu komutun işlem kodu “CE” olacaktır. İndeks kaydedicisine (F055)₁₆ verisini yüklemek için yazılacak assembly komutu LDX #\$F055 şeklindedir. Şekil 9.4’de indeks kaydediciye 2 byte'lık verinin yüklenmesini göstermektedir.



Şekil 9.4 : İvedi adresleme modunda LDX #\$F055 komutunun işleyişi

9.3 KAYDEDİCİ ve İMALI ADRESLEME (Accumulator and Implied Addressing)

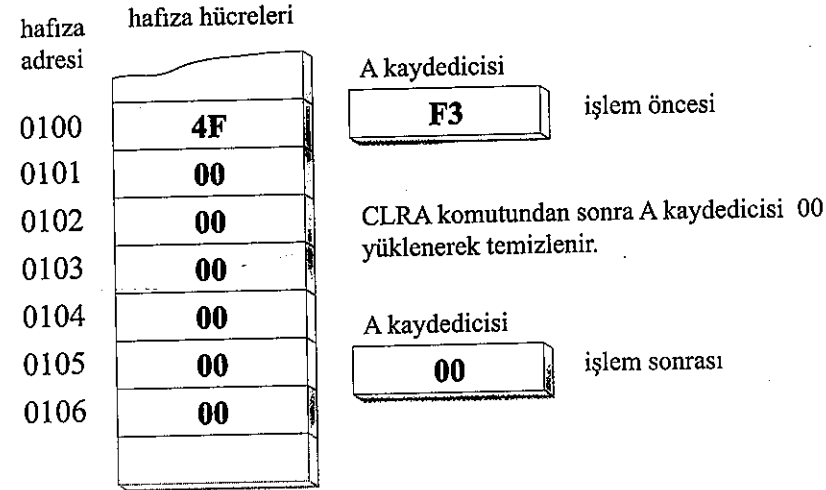
Bu adresleme modunda kullanılan komut, hiçbir işlenene gerek duymaz. Yani komuttan sonra ne bir “hafıza adresi” ne de bir “veri” belirtmeye gerek vardır. Bu adresleme modunda kullanılan komutlar 1 byte'lıktır. Tek başına kullanılan bu komutların, ne iş yapacağı daha üretim aşamasında tanımlanmıştır. Bu komutlara kendi görevlerinden başka bir şey yaptırılamaz. Örneğin kaydedicinin içeriğini 1 arttırmak için kullanılan bir komut, yalnızca bu işi yapar. Farklı olarak kaydedicinin içeriğini iki kez arttırmaz veya bir kez azaltamaz. Bazı komutlar, yalnızca kaydediciler üzerinde birtakım işlemler yaptıkları için kullanılan adresleme moduna “akümülatör adreslemesi” denilir.

“İmalı adresleme” modunda da yine aynı şekildedir, fakat bu komutlar kaydediciler üzerinde işlemler yapmazlar. Mikroişlemci içerisinde tek komutla yapılabilecek diğer işleri yaparlar. Örneğin NOP komutu, mikroişlemcinin işlem yapmadan bir sonraki komuta geçmesini sağlar.

Tablo 9.5 : Bazı mikroişlemcilerin kaydedici ve imalı adresleme modunda kullanılan komutları

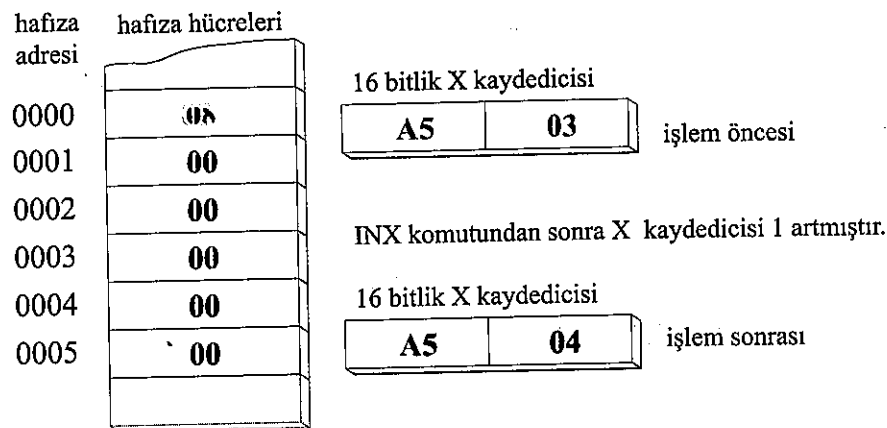
İş kodu	Assembly komutu	Açıklama
4F	CLRA	(MC6802) A kaydedicisine (00) ₁₆ yükleyerek sıfır.
08	INX	(MC6802) X kaydedicisini 1 artırır.
17	RLA	(Z80) A kaydedicisini 1 bit sola döndürür.
1B	DEC DE	(Z80) DE kaydedici çiftini 1 azaltır.
2F	CMA	(8080) A kaydedicisinin tümleyenini al.
00	NOP	(8085) işlem yapma.
1B	CLC	(MOS Tec. 6502) CCR ‘deki C bayrağı “0” yapılır.
AA	TAX	(MOS Tec. 6502) A kaydedicisi X kaydedicisine yüklenir.

CLRA komutunun işleyişi: CLRA komutu ile A kaydedicisinin içeriği sıfırlanır. CLRA komutu hafızada 1 byte yer kaplar ve komutun işlem kodu "4F" dir. Mikroişlemci bu komutu çözdüğünde üzerinde işlem yapacağı kaydediciyi ve ne tür bir işlem yapacağını bilir. CLRA komutu, A kaydedicisinin içeriğini temizlemekten başka bir şey yapmaz.



Şekil 9.5 : Kaydedici adresleme modunda CLRA komutunun işleyişi

INX komutunun işleyişi; Bu komut daha önceden belirlenmiş tek bir işi yapar. O da indeks kaydedicisinin içeriğini bir arttırmaktır. Komutun işlem kodu "08" dir. Dikkat edilirse komut bir byte'lıktır. Fakat 2 byte'lık bir kaydedici olan indeks kaydedicisinin içeriğini 1 arttırabilmektedir.



Şekil 9.6 : Kaydedici adresleme modunda kullanılan INX komutunun işleyişi

9.4 DOĞRUDAN ADRESLEME (Direct Addressing)

Komutlar, hafızanın ilk 256 byte'lık bir bölgesine bir veri yazmak veya okumak için kullanıldığında bu adresleme modunda kullanılmak zorundadır. Komuttan hemen sonra veri yazılmak veya okunmak istenen hafızanın adresi belirtilmelidir. Bu adresleme modu ile hafızanın ilk 256 byte'lık bölgesine erişim sağlanır. Mikroişlemci üreticileri özellikle, hafızanın ilk 256 byte'lık bölgesine erişim sağlanabilmesi için komut setlerinde bulunan komutlarını "doğrudan adresleme modunda" kullanım imkânı

sağlamışlardır. Bu durum programcıya hafızadan yer tasarrufu ve işlem hızı avantajı getirir. Hafızanın bu ilk 256 byte'lık bölgesi yani $(00)_{16} \dots (FF)_{16}$ arasında kalan bölge için kullanılan komutların doğrudan adresleme modunda kullanılmasına, bazı kaynaklarda ve mikroişlemci kataloglarında sıfır sayfa adreslemesi "zero page addressing" şeklinde ifade edilmiştir.

8 bitlik mikroişlemcilerin adresleyebileceği maksimum hafıza bölgesi $(00000)_{16} \dots (FFFF)_{16}$ adresleri arasındaki alan kadarlık bir bölgedir. Programcı eğer bu adresin ilk 256 byte'lık bölgesini kullanmak isterse (yani $(0000)_{16} \dots (00FF)_{16}$ adresleri arasındaki bölgeyi) o zaman kullandığı komutun işleneninde, fazladan bir byte'lık gereksiz bir kullanım söz konusu olacaktır $(0000)_{16} \dots (00FF)_{16}$. Aynı adresi, bu yararsız byte'ı $(00)_{16}$ kullanmadan da ifade edebilmek için komut, doğrudan adresleme modunda kullanılabilir. Bu şekildeki kullanımda $(0000)_{16} \dots (FFFF)_{16}$ arasındaki adresler hiçbir kısıtlama olmadan da $(00)_{16} \dots (FF)_{16}$ adresleri ile ifade edilebilmektedir. Burada fazladan kullanılan her $(00)_{16}$ değeri, hafızada 1 byte yer işgal edecektir. Aynı zamanda mikroişlemci aynı adres için önemsiz olarak kullanılan $(00)_{16}$ değerini tanımlamak için zaman harcayacaktır. Bunun çok kere tekrarlandığı durumlarda ise önemli süre ve yer kayıpları söz konusudur. Bu durumun dezavantajını ortadan kaldırmak için bazı komutların (hafızadaki ilk 256 byte'lık adres bölgesi için) "doğrudan adresleme modu"un kullanılmasında yarar vardır. Aslında bu adresleme modu "doğal adresleme modunun" alt bir üyesi gibidir. Hafızanın ilk 256 byte'lık bölgesine aynı şekilde "doğal adresleme modu" kullanılarak da erişilebilir o zamanda adresler $(0000)_{16} \dots (00FF)_{16}$ şeklinde gösterilerek 2 byte'lık şekilde tam olarak yazmayı gerektirir. Tablo 9.6'da aynı adresin iki farklı modda kullanılması gösterilmiştir.

Tablo 9.6 : $(5F)_{16}$ adresine doğrudan veya doğal adresleme modları ile aynı şekilde erişilebilir

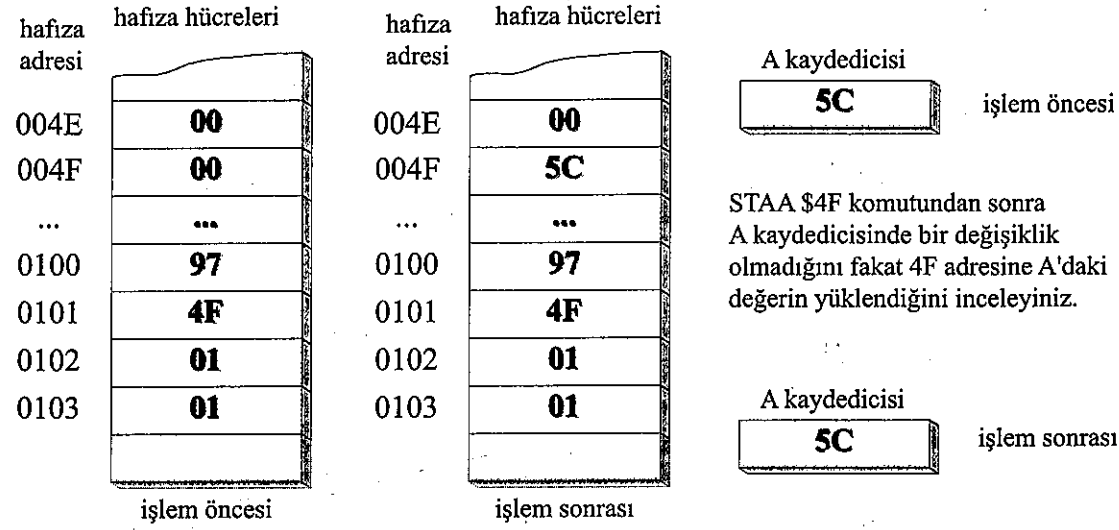
İş kodu	Assembly komutu	Açıklama
96 5F	LDAA \$5F	(MC6802) A kaydedicisine $(005F)_{16}$ adresindeki veriyi yükler.
B6 00 5F	LDAA \$005F	(MC6802) A kaydedicisine $(005F)_{16}$ adresindeki veriyi yükler.

Hafızanın ilk 256 adreslik bir bölgesi için "doğrudan adresleme modu" kullanıldığında, kullanılan komutun işlem kodu 1 byte ve erişilecek olan adres bilgisi de bir byte olacağı için hafızada toplam 2 byte yer işgal edilir (96 5F 'de olduğu gibi). Eğer, 3 byte'lık bir komutun toplam işlem süresi 7 periyot ise aynı adresin "doğrudan adresleme modunda" kullanılması ile işlem süresi 5 periyoda düşecek ve $7-5=2$ periyodluk süre kazanılacaktır. Dahası, hafızada fazladan 1 byte yer kazanılmış olacaktır. Tabii bu adresleme modunu ancak hafızanın ilk 256 byte'lık bölgesinin kullanılması durumunda kullanabiliriz. Bunu tek komut için düşündüğümüzde, önemsiz bir yer ve zaman kazanmak olarak görülebilir, fakat aynı komuttan program içerisinde 50 defa kullanıldığında toplam yer kazanma 50 byte ve toplam işlem süresinde $50 \times 2 = 100$ periyodluk bir süre kazanılacaktır.

Tablo 9.7 : Bazı mikroişlemcilerin doğrudan adresleme modunda kullanılan komutları

İş kodu	Assembly komutu	Açıklama
97 4F	STAA \$4F	(6802) A kaydedicisindeki veriyi $(4F)_{16}$ adresine yükler.
91 10	CMPA \$10	(6802) A kaydedicisindeki veri ile $(10)_{16}$ adresindeki veriyi karşılaştırır.
35 10	DEC \$10	(Z80) 10 adresindeki veriyi bir azaltır.
25 40	AND \$40	(MOS Tec. 6502) 40. adresde ki veri ile A kaydedicisindeki veriye AND işlemi uygulanır.

STAA \$4F komutunun işleyişi; Hafızanın ilk 256 byte'lık bölgesinde bulunan $(4F)_{16}$ adresine A kaydedicisindeki veriyi yüklemek için STAA \$4F komutunu doğrudan adresleme modunda kullanmak en mantıklı yol olacaktır. Komutun işlem kodu "97 4F" dir. Komut $(0100)_{16}$ adresinden itibaren başlamıştır. A kaydedicisindeki veri $(004F)_{16}$ adresine yüklenmiştir. Komutun işleyişi şekil 9.7'de gösterilmiştir. Komutun doğal adresleme yönteminde kullanımı için doğal adresleme modu konusundaki örneği inceleyiniz.



Şekil 9.7 : Doğrudan adresleme modunda kullanılan STAA \$4F komutunun işleyişi

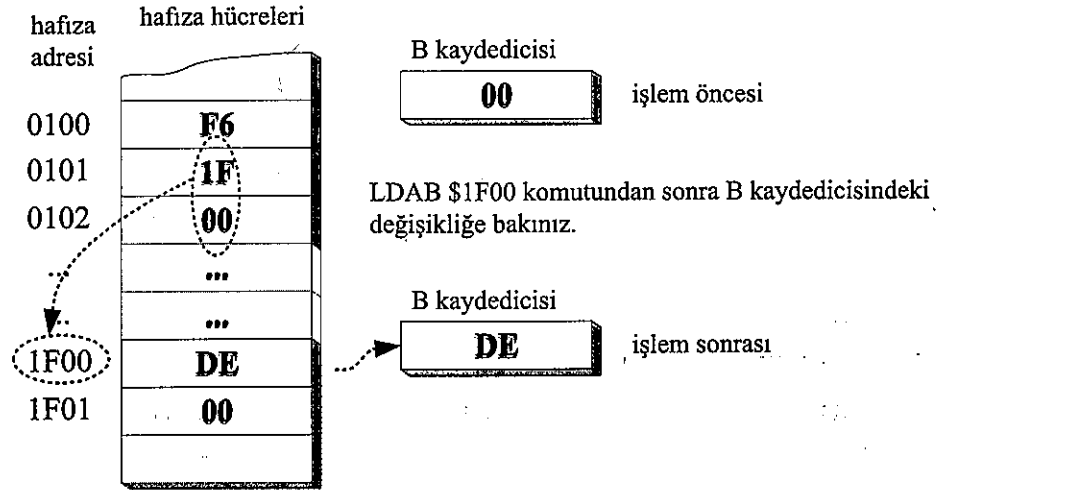
9.5 DOĞAL ADRESLEME (Absolute Addressing)

Bu adresleme yöntemi ile işlenenin bulunduğu veya bulunacağı hafıza hücrelerinin adresi belirtilir. Yani komuttan sonraki işlenen 2 byte'lık bir adres bilgisidir. Doğal adresleme modunda yazılan komutlar toplam 3 byte'tan oluşur. 1. byte komutun kendisi diğer 2 byte ise erişilecek olan adres bilgisidir. Bu adresleme yöntemini kullanarak (16 bitlik adres yolu olan işlemciler için) 65,536 byte'lık hafıza alanı içerisindeki herhangi bir hafıza hücrelerine erişilebilir.

Tablo 9.8 : Bazı mikroişlemcilerin doğal adresleme modunda kullanılan komutları

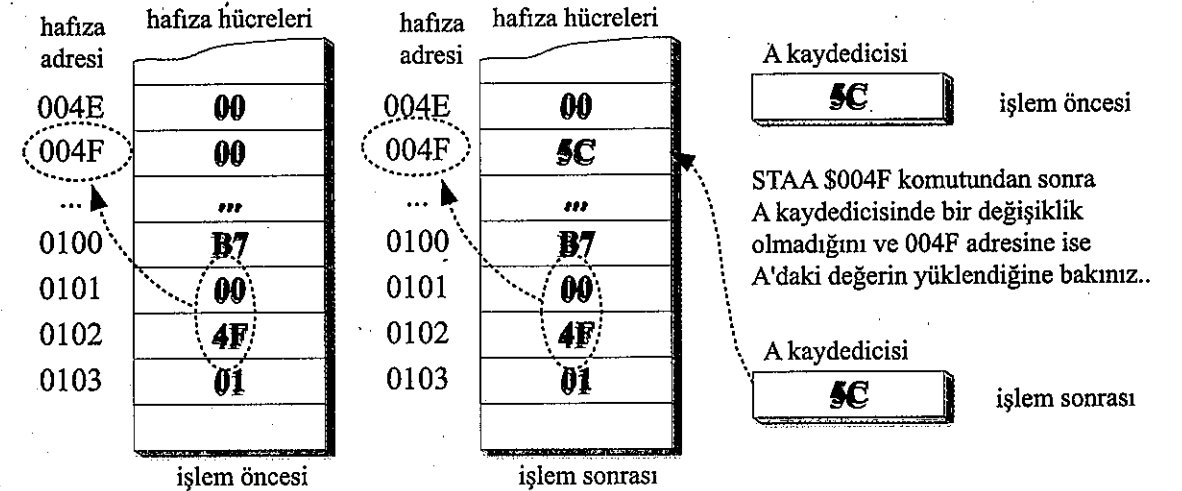
İş kodu	Assembly komutu	Açıklama
F6 1F 00	LDAB \$1F00	(MC6802) B kaydedicisine (1F00) ₁₆ adresindeki veriyi yükler.
B7 00 4F	STAA \$004F	(MC6802) A kaydedicisindeki veriyi (004F) ₁₆ adresine yükler.
2D 00 02	LDAA \$0200	(Mos Tec. 6502) (0200) ₁₆ adresindeki veriyi A kaydedicisini yükler.
C3 00 01	JP \$0100	(Z80) (0100) ₁₆ adresindeki komuttan devam et.

LDAB \$1F00 komutunun işleyişi; Bu komutla (1F00)₁₆ adresindeki veri B kaydedicisine yüklenir. Şekil 9.8'de komutun doğrudan adresleme modunda kullanımı, komutun hafızadaki yerleşimi ve yapılan işlem görülmektedir. Komut hafızanın (0100)₁₆ adresinde bulunmaktadır. Komut hafızada işlenenleri ile birlikte toplam 3 byte yer kaplar (F6 1F 00).



Şekil 9.8 : LDAB \$1F00 komutunun doğal adresleme modundaki kullanımı

STAA \$004F komutunun işleyişi; Bu komut A kaydedicisindeki veriyi (004F)₁₆ adresine yükler. Komutun işlem kodu "B7"dir. Komut hafızanın (0100)₁₆ adresinden itibaren başlamaktadır. Komutun hafızada yaptığı değişiklik ve komutun hafızadaki yerleşimi ve kaydedicilerin durumu Şekil 9.9'da görülmektedir. Bir önceki örnekle karşılaştırıldığında LDA komutu ile kaydedicilere yükleme yapılırken STA komutu ile de kaydedicilerdeki verilerin hafızaya yüklendiği görülür.



Şekil 9.9 : STAA \$004F komutunun doğal adresleme yönteminde kullanımı

9.6 SIRALI ADRESLEME (Index Addressing)

Hafızada sıralı şekilde bulunan veriler veya sıralı şekle sokulmak istenen veriler üzerinde daha kolay işlemler yapabilmek için "sıralı adresleme modu" kullanılır. Bu adresleme modunu kullanabilmek için mikroişlemcide sıralama kaydedicisi (index register) bulunmalıdır. Bir çok mikroişlemcide de bulunmaktadır. Bazı işlemcilerde sıralama kaydedicisinden iki tane bulunması, diziler üzerinde işlem yapmak için yazılan programları büyük ölçüde kısaltır ve sadeleştirir. Komutların sıralama adresleme modunda kullanılabilmesi için indeks kaydedicilerin de işin içine katılması gerekmektedir. Yani sıralı adresleme modu kullanılarak yazılan bir komuttan önce sıralama kütüğünü gereken şekilde düzenlemek gerekmektedir ve bu da tamamen programcının sorumluluğundadır.

Hafızada sıralı bir şekilde bulunan veriler üzerinde işlem yapabilmek için ilk verinin bulunduğu taban adresi sıralama kaydedicisine yüklenmelidir. Artık indeks kaydedicisi hafızada sıralı bir şekilde bulunan verilerden ilkinin adresini işaret etmektedir. Sıralı adresleme modunda kullanılacak diğer komutlara, bu veri zinciri içerisinde bulunan bir veriye erişebilmesi için verinin dizi içerisindeki sırasını belirtmek yeterlidir. Assembly komut satırında bir komutun sıralı adreslemede kullanıldığını “X” ile anlarız (LDAA #S05,X).

Tablo 9.9 : Bazı mikroişlemcilerin sıralı adresleme modunda kullanılan komutları

İş kodu	Assembly komutu	Açıklama
A6 05	LDAA #S05,X	(MC6802) A kaydedicisine indeks kayd edicisindeki değer (05) ₁₆ ekleyerek elde edilen adresten veri yükler..
35 40	AND \$40,X	(MOS Tec. 6502) İndeks kaydedicisinin taban adresine (40) ₁₆ ekleyerek elde edilen adresdeki veriyi A kaydedicisine yükler.

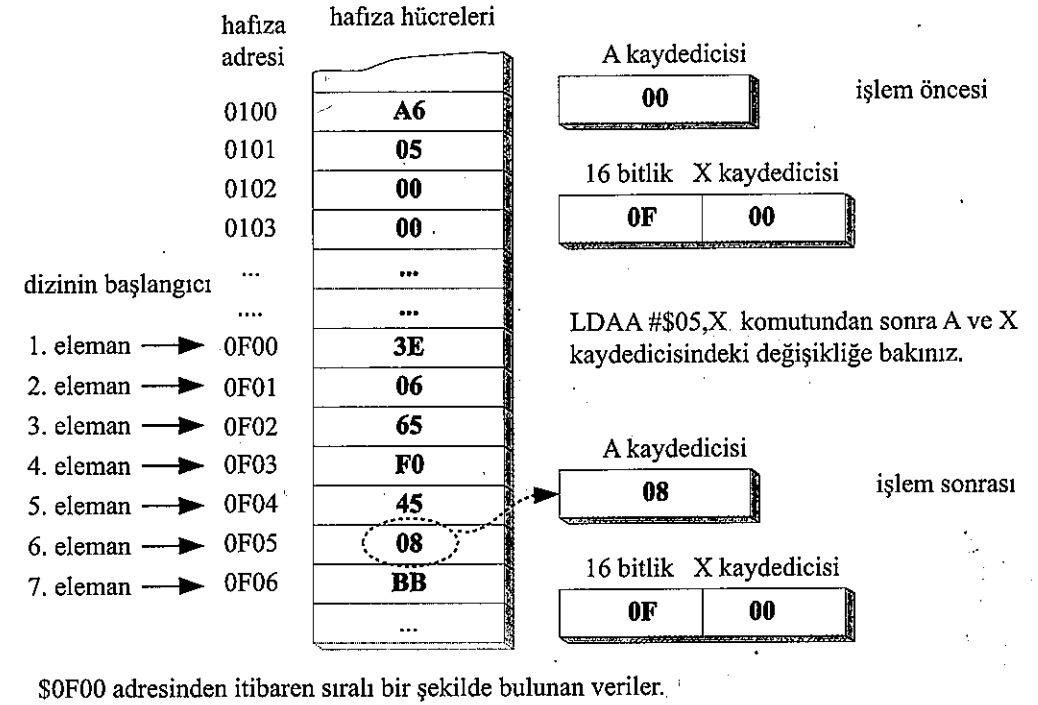
LDAA #S05,X komutunun işleyişi; (0F00)₁₆ hafıza hücresinden itibaren hafızada bulunan sıralı veriler üzerinde işlem yapmak için öncelikle dizinin ilk elemanının bulunduğu (0F00)₁₆ adres bilgisi indeks kaydedicisine yüklenir. Sıralı adresleme modunda kullanılan LDAA #S05,X komutu temelde A kaydedicisine bir veri yükleme komutudur. Fakat komut sıralı adresleme modunda kullanıldığı için yüklenecek verinin geçerli adresi indeks kaydedicisindeki taban adrese (05)₁₆ eklenmesi ile elde edilecektir. İndeks kaydediciye sıralı verinin başlangıç adresinin (0F00)₁₆ daha önceden yüklendiğini varsayarak LDAA #S05,X komutu ile (0F00)₁₆ adresindeki sıralı verinin 6. elemanı (#S05 işleneni dizideki 6. elemanı işaret eder.) A kaydedicisine yüklenir.

Tablo 9.10 : MC6802 komut setindeki komutlar ile hafızadaki sıralı verilerin 6. sırası A kaydedicisine yüklemek için yazılacak program satırları.

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	CE 0F 00		LDX #S0F00	İndeks kaydedicisi dizinin ilk verisinin bulunduğu adres ile yüklenir.
0102	A6 05		LDAA #S05,X	A kaydedicisine dizideki 6. veri yüklenir (A'ya yüklenmek istenen 6. verinin adresi, indeks kaydedicisindeki taban adrese (05) ₁₆ ekleyerek elde edilir.).
....				

İlk elemanın bulunduğu hafıza hücresinin adresi, indeks kaydedicisine yüklendikten sonra sıralı adresleme modunu kullanan herhangi bir komut işletilebilir. Şekil 9.10'da indeks kaydedicisine (0F00)₁₆ adres bilgisi yüklendikten sonra LDAA #S05,X komutunun işleyişi ve kaydedicilerdeki durum görülmektedir.

LDAA #S05,X komutunun işlem kodu “A6 05” dir. Komut (0100)₁₆ adresinde bulunmaktadır ve işleneni (05)₁₆ ile birlikte hafızada 2 byte yer tutar. Şekil 9.10'da görüldüğü gibi mikroişlemci LDAA #S05,X komutunu yürütürken komuttan sonraki (05)₁₆ verisini, indeks kaydedicisinde bulunan (0F00)₁₆ adres bilgisine ekleyerek elde edilen adresteki veriyi A kaydedicisine yükler. Geçerli adres bulunurken indeks kaydedicisindeki taban adresi değişmez. Yani LDAA #S05,X komutu işletildikten sonra X kaydedicisinin içeriği değişmez ve (0F00)₁₆ olarak kalır. X kaydedicisindeki veri özellikle değiştirilmediği sürece bu değer her zaman sıralı veriyi gösterecek şekilde saklı kalır.



\$0F00 adresinden itibaren sıralı bir şekilde bulunan veriler.

Şekil 9.10 : LDAA #S05,X komutunun işleyişi

9.7 BAĞIL ADRESLEME (Relative Addressing)

Bağıl adresleme modunu, şartlı veya şartsız dallanma komutları kullanır. Bu komutlar ile program, akışını belirtilen şartlar gerçekleştiğinde başka bir noktadan devam edecek şekilde programlanabilir. Bu komutlar şarta bağlı olarak hafızada işlenecek olan bir sonraki komutun etkin adresini, komutun bulunduğu yerden ya ileriye (127 byte) ya da geriye (-128 byte) olacak şekilde program sayıcısını değiştirerek belirler. Hafızada toplam 2 byte yer tutarlar. İlk byte dallanma komutunun işlem kodu, 2. byte ise dallanacak yerin bağlı adresini gösterir. Bağlı adres değeri 8 bitlik işaretli sayıdır. Bağlı adresin işaretli sayı olması en fazla +127 ileri, 128 geri sapma miktarları ile sınırlıdır. Daha fazla ileri veya geri dallanmalarda JMP (Jump; atla) veya JSR (Jump subroutine; alt programa atla) komutlarına benzer komutlar kullanılabilir.

Tablo 9.11 : Şartlı dallanma komutunda kullanılan bağlı adres değerinin kullanımı

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	86 55	BAS	LDAA #S55	
...	...	...		
0108	27 F7		BEQ BAS	(MC6802) Z=1 ise BAS etiketinin olduğu satıra dallan.

Bağıl adresleme modunu kullanan komutlar bir sonraki yürütülecek olan komutun etkin adresini belirlemek için PC'deki (Program Counter; program sayıcı) geçerli adrese, bağlı adres değerinin ekler veya çıkarır. PC'ye eklenen pozitif değerler, programın ileri dallanmasına, negatif değerler ise geri dallanmasına sebep olur. Tablo 9.11'deki program kodlarında, BEQ komutu Z=1 şartı sağlandığında BAS etiketinin bulunduğu (0100)₁₆ adresine yani geriye doğru bir dallanma yapacaktır. Buradaki (F7)₁₆ bağlı adres değeri negatif bir sayı olduğundan dallanmanın zaten geriye doğru olacağı açıktır. Bağlı adres değerinin doğru hesaplanması gerekir yoksa programın çalışması istenmeyen satırlara kayabilir. Dallanılacak adresin bağlı değerinin nasıl hesaplanacağı “Mikroişlemci komutları” bölümündeki “test ve dallanma komutları” konusunda detaylı olarak anlatılmıştır.

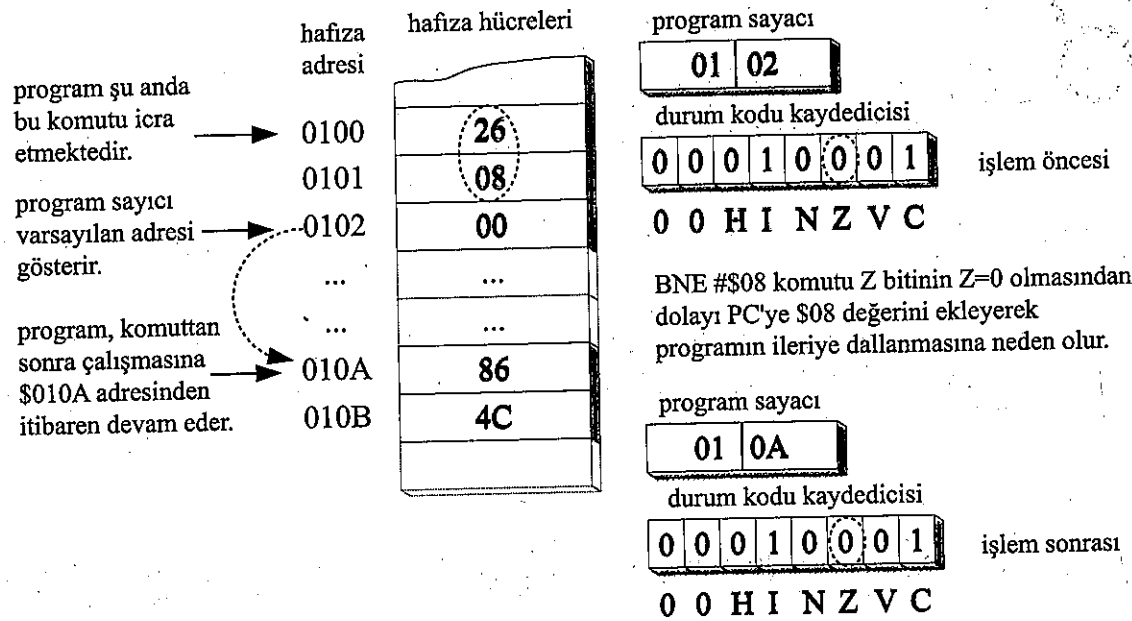
Tablo 9.12'de değişik mikroişlemcilerin komut setlerinde bulunan ve bağıl adresleme modunda kullanılan assembly komutları ve işlem kodları bulunmaktadır.

Tablo 9.12 : Bazı mikroişlemcilerin bağıl adresleme modunda kullanılan komutları

İş kodu	Assembly Komutu	Açıklama
26 08	BNE #08	(MC6802) Z=0 ise program (010A) ₁₆ adresine sapar.
20 80	BRA #80	(MC6802) Şartsız olarak geriye dallanır.
90 04	BCS \$04	(Mos Tec. 6502) C=1 ise (04) ₁₆ byte ileri sap.

BNE #08 komutunun işleyişi; Mikroişlemci BNE #08 komutunun işlem kodu olan "26 08" kodlarını işletirken program sayıcı normalde bir sonraki komutun bulunduğu adresi işaret etmek isteyecektir ve PC bir sonraki komutun adresini gösterecek şekilde artmış olacaktır yani PC'de (0102)₁₆ adresi olacaktır. Fakat yürütülmekte olan şartlı dallanma komutu gereken şartı (BNE komutu Z'nin Z=0 eşit olması şartını arar.) bulunduğu dolay, sapmanın gerçekleşmesini sağlayacaktır. Dolayısı ile program sayıcının içeriğine dallanılmak istenen adrese (010A)₁₆ ulaşabilmesi için (08)₁₆ değerinin eklenmesi gerekecektir. PC'de (0102)₁₆ bulunduğu dolay (08)₁₆ değerinin eklenmesi ile istenen adrese dallanma gerçekleşir. Eğer programcı "assembly komutlarından" "işlem kodlarına" çevirmeyi el ile yapıyorsa şartlı veya şartsız dallanma komutundan sonra kullanılacak bağıl adres değerini hesaplayarak komutun işlem koduna eklenmesi gerekmektedir. Fakat çevirici bir program kullanılıyorsa bu değer zaten otomatik olarak hesaplanacaktır (çevirici program, assembly komut satırlarında belirtilen etiketlerin bulunduğu adresleri hesaplayarak bağıl adres değerlerini bulur ve programın iş koduna bu değerleri yazar.).

MC6802 mikroişlemcisinin komut setinde bulunan bu komut hafızada (0100)₁₆ adresinde bulunmaktadır. Hafızada toplam 2 byte yer tutar. 1.byte komutun kodu olan "26" 2.byte ise "08" dallanma miktarıdır. Şekil 9.11'de komutun işleyişi ve kaydedicilerin durumu görülmektedir.



Şekil 9.11 : Bağıl adresleme de kullanılan BNE #08 komutunun işleyişi

9.8 DOLAYLI ADRESLEME (Indirect Addressing)

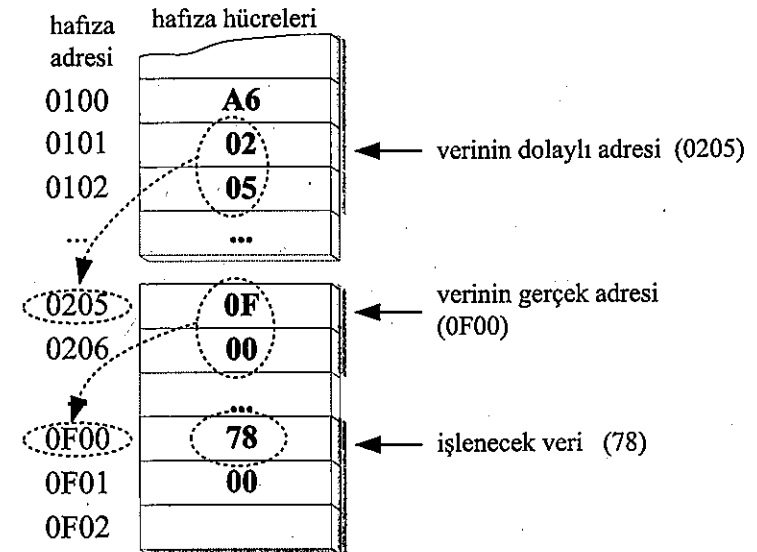
Komutların, doğrudan adresleme ve doğa adresleme modunda kullanılmasında, işlenecek verinin bulunduğu veya bulunacağı hafıza hücresinin adresinin, belirtilmesi gereklidir. Dolaylı adresleme modunda ise işlenecek verinin adresi başka bir hafıza hücresinde veya kaydedicide bulunur. Yani verinin gerçek adresi başka bir "hafıza hücresinde" veya bir "kaydedicide" saklıdır. Eğer bu adresleme modunda kaydediciler kullanılırsa bu adreslemeye "kaydedici-dolaylı adresleme" bir hafıza hücresi kullanılırsa "mutlak-dolaylı adresleme" adı da verilmektedir. Bu kaydedici veya hafıza hücresine çoğu kez adres işaretleyicisi de denilir. Bu tip bir adresleme modunu Intel 80xx serisi ve Zilog Z80 mikroişlemci ailelerinin komut setlerinde bulunmaktadır.



Motorola 6800 Ailesinde Dolaylı Adresleme Modu Yoktur:

Dolaylı adresleme şekline benzer bir adresleme, indeks kaydedici kullanarak gerçekleştirilebilir. Örneğin: LDAA #00,X komutu ile indeks kaydedicisi dolaylı adres kaydedicisi olarak kullanılarak A kaydedicisine yüklenecek verinin bulunduğu adres indeks kaydedicisinde belirtilmiş olur. Bu komutla A'ya yüklenecek verinin adresinin önceden indeks kaydedicisine yüklenmesi gerekecektir. Böylelikle A kaydedicisine yüklenecek verinin bulunduğu gerçek adres bilgisine, indeks kaydedicisindeki adres bilgisi üzerinden ulaşılarak dolaylı bir adresleme gerçekleştirilmiş olur.

Dolaylı adreslemenin başka bir hafıza adresini kullanarak nasıl yapıldığı şekil 9.12'de gösterilmiştir. Komutun (A6)₁₆ işlem kodundan sonra belirtilen adres, işlenecek olan verinin adresi değildir. Bu adres işlenecek verinin adresini gösteren adrestir. Böyle bir adresleme modunun programcıya nasıl bir avantajı olabilir? Bu adresleme yöntemi ile farklı yerlerde bulunan verilere erişmek için (0205)₁₆ adresindeki adresi değiştirmek yeterli olacaktır. (0205)₁₆ adresine biz (0F00)₁₆ yerine (0D00)₁₆ yazmış olsaydık komut (0D00)₁₆ daki veriyi işleme koyacaktı.



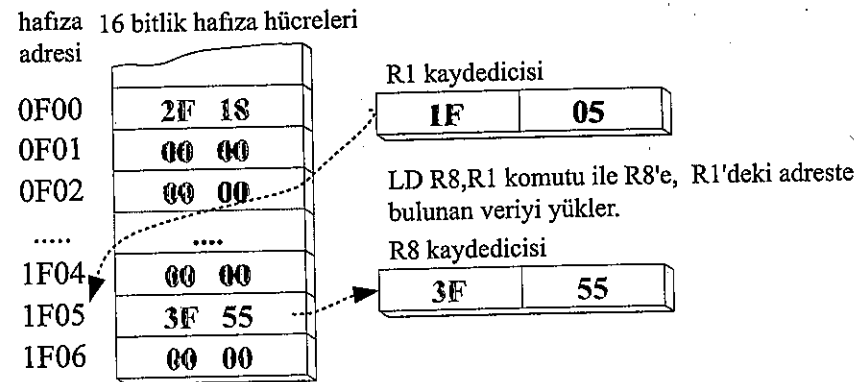
Şekil 9.12 : Dolaylı adresleme yöntemi

Tablo 9.13'de komut setlerinde dolaylı adreslemeyi kullanan bazı mikroişlemcilerin komutları verilmiştir.

Tablo 9.13 : Dolaylı adresleme modunda kullanılan mikroişlemci komutları

İş kodu	Assembly komutu	Açıklama
A1 40	LDA \$40,X	(MOS Tec. 6502) A kaydedicisine yüklenecek veri, X kaydedicisindeki taban adresine (40) ₁₆ adresindeki veri eklenerek elde edilen adresten yüklenir.
2F 18	LD R8,R1	(Z8000) R8'e R1 kaydedicisindeki adresten veri yükler.

LD R8,R1 komutunun işleyişi; Dolaylı adresleme modunun bir alt çeşidi olan “dolaylı-kaydedici adresleme modu” Z8000 işlemcisinin kullandığı komut adresleme moduna bir örnektir. LD R8,R1 komutu Zilog Z8000 mikroişlemcisinin komut setinde bulunmaktadır. Z8000 mikroişlemcisi 16 bitlik bir işlemcidir ve hafıza birimleri de 16 bit veriyi saklayacak şekilde seçilmiştir. Bu komut ile R8 kaydedicisine, R1 kaydedicisindeki bir adresten yükleme yapar. R8 kaydedicisine yüklenecek verinin adresi, R1 kaydedicisinde bulunmaktadır. Bu komutu kullanmadan önce R1 kaydedicisi doğru adres ile yüklenmek programcının sorumluluğundadır.



Şekil 9.13 : Kaydedici-Dolaylı adresleme modunu kullanan Z8000 işlemcisinde LD komutunun işleyişi

DEĞERLENDİRME ÇALIŞMALARI

- 1) Adresleme modu nedir? Mikroişlemciler programlanırken adresleme modu niçin gereklidir?
- 2) Mikroişlemci komutlarının kullandıkları “işlenen” nedir? kaç tür işlenen vardır?
- 3) Motorola MC6802 mikroişlemcisinin kullandığı adresleme modlarını yazın.
- 4) Zilog Z80 mikroişlemcisinin kullandığı adresleme modlarını yazın.
- 5) Mos. Tec. 6502 mikroişlemcisinin kullandığı adresleme modlarını yazın.
- 6) Intel 8085 mikroişlemcisinin kullandığı adresleme modlarını yazın.
- 7) İvedi adresleme modunu bir komutla birlikte açıklayınız.
- 8) Kaydedici veya imalı adresleme modunu bir komutla birlikte açıklayınız.
- 9) Doğrudan adresleme modunu bir komutla birlikte açıklayınız.
- 10) Doğal adresleme modunu bir komutla birlikte açıklayınız.
- 11) Sıralı adresleme modunda hangi kaydedici kullanılmak zorundadır? Neden?
- 12) Bağlı adresleme modunu bir komutla birlikte açıklayınız.

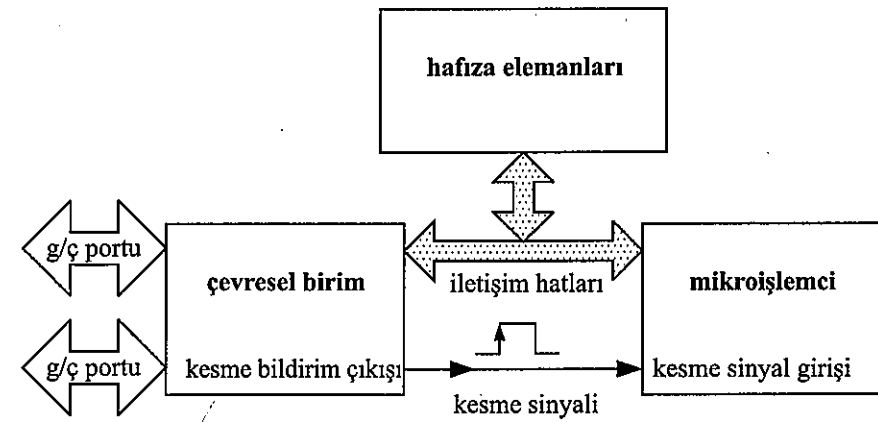
ÜNİTE 10 INTERRUPT (Kesme) ve RESTART (Yeniden Başlatma)

10.1 INTERRUPT (KESME)

Kesme; oluşan bir kesme sinyali ile birlikte mikroişlemcinin, yürütmekte olduğu programı bir süre için askıya alması (daha sonra işletmek üzere) ve denetimin kesme hizmet programına aktarılması olayıdır. Mikrobilgisayarlarda, kesme işlevini kullanmanın birçok sebebi ve yararı vardır. Kesmenin ne işe yarayabileceğini kavramak için bunu günlük hayattan bir örnekle açıklayalım. Örneğin, ev içerisinde gündelik hayatımızı sürdürürken kapı zili çaldığında o anki işimizi geçici bir süre bırakarak kapıyı açarız. Bizim kapıyı açmamızı işaret eden şey, zilin çalmasıdır. Hiçbir zaman zil çalmadan sürekli olarak kapıyı açarak gelip gelmeyenin olduğuna bakmayız. Sürekli olarak kapıyı açıp-kapamamız hiç pratik olmadığı gibi günlük yaşamımızı da olumsuz etkiler.

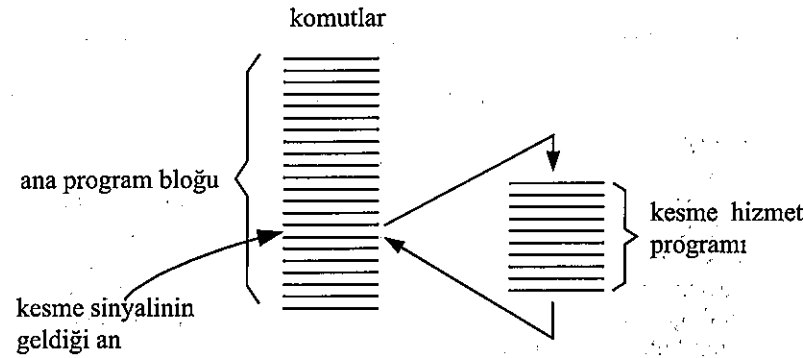
Mikrobilgisayarlar birçok çevresel birimler bulundurulur. Bu çevresel birimler, değişik nedenlerden dolayı mikroişlemci ile iletişim kurma gereği duyarlar. Mikroşlemciye ihtiyaç duyduklarında bunu bildirmek için mikroşlemcinin dikkatini çekecek bir sinyal üretirler. Mikroşlemci tarafından alınan sinyal ile hemen kesme hizmet programına dallanarak gerekli olan program satırları çalıştırılır. Böyle bir çalışma durumuna, bilgisayarlardaki klavyeleri örnek verebiliriz. Herhangi bir tuşa basılmadığı sürece klavyedeki tuşların okunmasına gerek yoktur. Eğer mikroşlemci sürekli olarak klavyede bir tuşa basılıp basılmadığını öğrenmek isteseydi çalışma zamanının büyük bir kısmını bu işe ayırmak zorunda kalacaktı. Bu da çok verimsiz bir çalışma demektir.

Aslında mikroşlemcilerin, sistemdeki çevresel birimlerin durumlarını öğrenmek için kullanılan iki yöntem bulunmaktadır. Tüm çevre birimlerinin durumunu tek tek iletişim kurarak öğrenmesi şeklindeki bir işleyişe “tarama yöntemli çalışma” denir. Fakat oluşan herhangi bir durumda bunun mikroşlemciye bir kesme sinyali ile bildirilmesi ve bunun da mikroşlemci tarafından hemen değerlendirip tekrar işine devam etmesi şeklindeki bir çalışmaya ise “kesme yöntemli çalışma” denilir.



Şekil 10.1 : Bir mikrobilgisayarda kesme sinyali bağlantısı

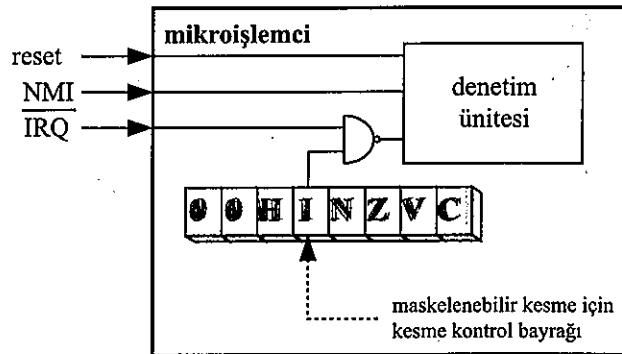
Öncelikle kesmenin oluşabilmesi için çevresel birimlerin birinden mikroşlemcinin kesme girişine bir kesme sinyalinin gelmesi gerekir. Mikroşlemci bu sinyali aldığı andan itibaren yürütmekte olduğu son komutun çalışmasını tamamlar. Daha sonra kaydedicilerdeki geçerli verileri otomatik olarak yığına saklar. Bu işlem tamamlandıktan sonra kesme hizmet programına dallanılır. Mikroşlemci kesme hizmet programı ile çalışmasını bitirdikten sonra ana programda kaldığı yerden devam edebilmesi için tekrar yığındaki bilgilerin alınıp kaydedicilerin içeriklerinin kesme olmadan önceki değerler ile yüklenmesi ve böylelikle ana programın kaldığı yerden çalışması sağlanmalıdır.



Şekil 10.2 : Ana programdan kesme hizmet programına geçiş.

Motorola 6800 serisi ve Zilog Z80 serisi mikroişlemciler, kesme sinyali ile birlikte ilk önce kaydedicilerdeki geçerli değerler otomatik olarak yığına atıldıktan sonra kesme hizmet programına dallanırlar. Fakat Intel 8080 mikroişlemcisine kesme sinyali geldiğinde, çalışılan hiçbir kaydedicinin içeriği yığına alınmaz. Bu iş programcının sorumluluğundadır. Intel 8080 ile çalışıldığında, kesme hizmet programına dallanıldığında eğer gerekiyorsa geçerli kaydedicilerin içerikleri PUSH komutu ile yığına atılması ve kesme hizmet programından çıkmadan önce de POP komutu ile yığındaki verilerin geriye çekilmesi gerekir.

Mikroişlemcilerde, maskelenebilir kesme girişi ve maskelenemez kesme girişi bulunabilir. Maskelenebilir kesme girişleri kontrol edilebilirken maskelenemez kesme isteklerine mikroişlemci her zaman cevap vermek durumundadır. Maskelenebilir ve maskelenemez kesmeleri kavramak için günlük hayattaki şu örneği inceleyelim. İki kapılı bir oda düşünelim. Kapının birisinden istendiği zaman giriş yapılabilirken diğer kapıdan ancak kapıdaki görevlinin izin vermesi hâlinde giriş yapılabilir. İlk kapıdan yapılan girişlere hiçbir zaman müdahale yapılamazken ikinci kapıdan yapılan girişler istendiği zaman durdurulabilir veya girişler sağlanabilir. Maskelenemez ve maskelenebilir kesmelerin temel mantığı da bu ilkeye göre çalışmaktadır.



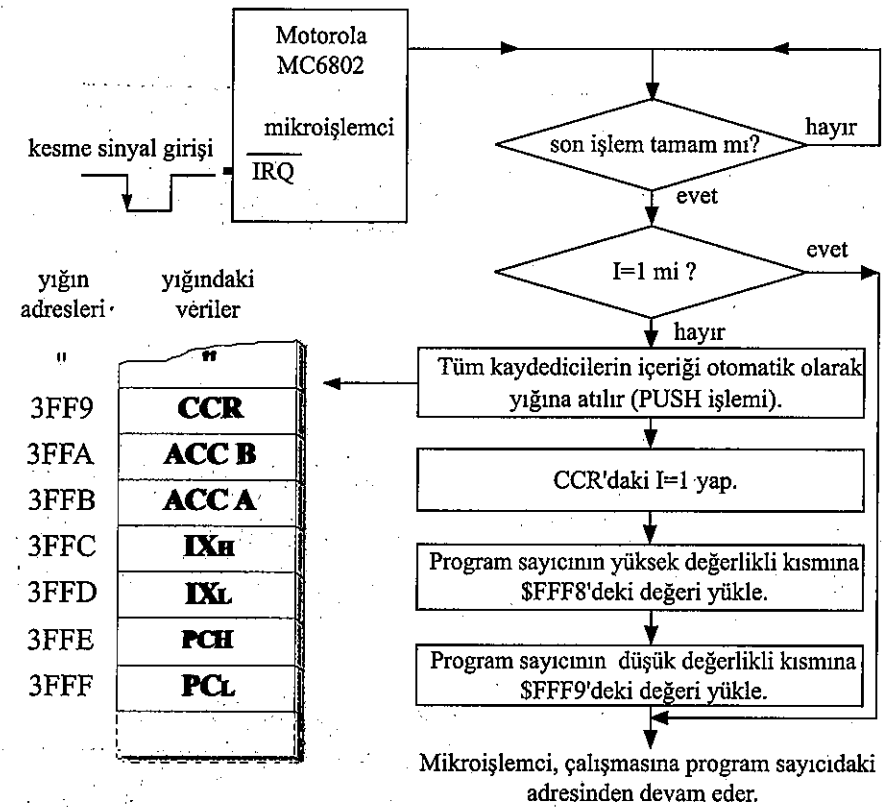
Şekil 10.3 : MC6802 mikroişlemcisinde maskelenebilir ve maskelenemez kesme girişleri

10.1.1 Maskelenebilir Kesmeler

Mikroişlemcilerdeki maskelenabilir kesme girişlerini kontrol edebilmek için durum kodu kaydedicilerinde kesme yetkilendirme bayrakları vardır. Bu bayraklar "1" veya "0" yapılarak, gelen kesmeler engellenebilir veya gelen kesmelere cevap verilebilecek şekilde ayarlanabilir. Mikroişlemcilerin çoğunda kesme denetim bayrağı gelen kesmeleri dikkate almayacak şekilde başlatılır. Eğer programcı sisteme gelen kesmelerin dikkate alınmasını istiyorsa bu bayrağı etkinleştirmelidir.

Intel'in 8080 mikroişlemcisinin durum kodu kaydedicisindeki kesme bayrağı; "EI" komutu kullanılarak kesme girişleri etkin kılınır, "DI" komutu ile de kesme istekleri göz ardı edilebilir. MC6802 mikroişlemcisinde CLI komutu kesme bayrağını "0", SEI komutu ise "1" yapar.

MC6802 mikroişlemcisinin IRQ girişi yeterli süre boyunca "0" da tutulursa şekil 10.4'deki işlemler yerine getirilmeye başlanır. IRQ girişinin "0" yapılması ile öncelikle yapmakta olduğu komutun işlemi tamamlanır. Daha sonra kesme bayrağı kontrol edilir. Eğer Durum kodu kaydedicisindeki I bayrağı "1" ise normal çalışmasına devam eder. "0" ise kesme işlemi başlatılır. Kaydedicilerin içerikleri yığına otomatik olarak atılır. Ardından program sayıcısına, kesme hizmet programının adresi yerleştirilerek mikroişlemcinin normal işleyişine bu adresten itibaren devam etmesi sağlanır. Tüm bu işlemler bir kesme sinyalinden sonra mikroişlemci tarafından otomatik olarak gerçekleştirilir. Programcıya kalan bir kesme sinyalinden sonra mikroişlemcinin kesme hizmet programının bulunduğu adresi, kesme vektörüne yazmış olması ve bu adreste kesme hizmet programını bulundurması gerekmektedir.



Şekil 10.4 : MC6802 mikroişlemcisindeki kesme sinyali ile birlikte gerçekleşen işlemler



MC6802 İçin İdeal Kesme Sinyali Bekleme Durumu:

MC6802 mikroişlemcisinde, $\overline{\text{IRQ}}$ kesmesinin maskesini kaldırmak ve kesmeyi bekle durumuna getirmek için NOP-CLI-WIA komut dizisi önerilir. WIA komutu ile kaydedicilerin içerikleri ve program sayıcı yığına itilir. Bundan sonra mikroişlemci adres yolunu, veri yolunu ve kontrol yolunu sistemden yalıtır. R/W kontrol ucu kayar hâledir ve VMA, low durumunda kesme sinyalinin gelmesi beklenir.

Kesme sinyali herhangi bir anda oluşabileceğinden, programcının ortaya çıkabilecek iki durum için önlem alması gerekir. Bunlardan birincisi ilk kesmenin ardından kesme hizmet programında çalışan mikroişlemci daha burada işini bitirmeden ikinci kesme sinyaline maruz kalabilir. İkincisi ise kesme sinyali daha süresini tamamlamadan mikroişlemcinin kesme hizmet programında işini bitirmesi ve aynı kesme sinyaline tekrar maruz kalabilmesi ihtimalidir. Bunlardan ilkinin çözümü yazılım ile ikincisinin çözümü ise donanım ile yapılabilir. Bunların nasıl olacağını inceleyelim.

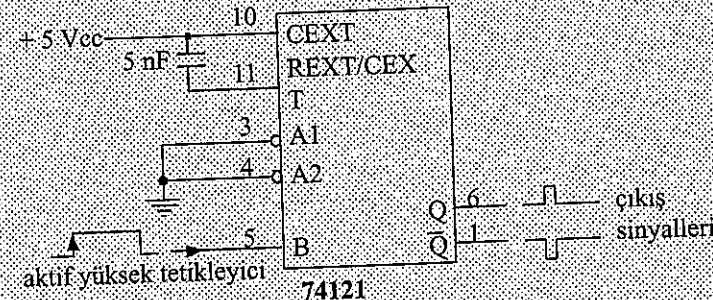
Eğer program bir kesme sinyali ile kesme hizmet programına dallanmış ise mikroişlemci kesme hizmet programını işletirken gelecek olan başka kesme sinyallerini önlemek gerekir. Bunun için kesme hizmet programının başında gelen kesmelerin dikkate alınmaması için kesme bayrağı yeniden düzenlenir. Kesme hizmet programından çıkıldığında ise bu bayrak gelecek olan diğer kesme sinyallerine cevap verebilmesi için yeniden düzenlenmelidir.

Eğer kesme hizmet programı, mikroişlemci girişindeki kesme sinyali daha ortadan kalkmadan işini tamamlarsa aynı kesme sinyaline maruz kalabilir. Bunun için aşağıdaki bilgi kutusundaki “kesme girişi darbe daraltıcı devre” kullanılabilir.



Kesme Sinyal Daraltıcı:

Kesme girişi için darbe daraltıcı devresi, mikroişlemci girişindeki kesme istek sinyalinin uzun süre etkin kalması durumunda, kesme hizmet programının aynı kesme sinyaline maruz kalmaması için kullanılabilir. Bu olumsuz durum aşağıdaki devre ile düzeltilebilir.

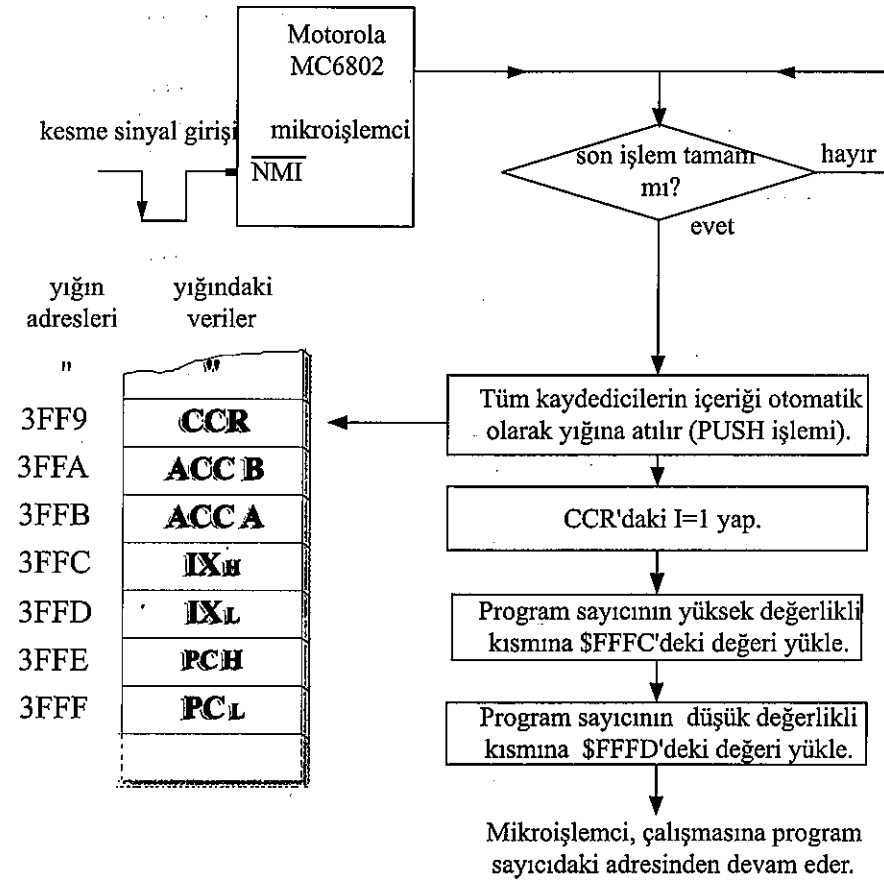


10.1.2 Maskelenemez Kesmeler

Maskelenemez kesmeler, hiçbir şarta bağlı olmadan mikroişlemcinin elindeki son komutu yürüttükten sonra kesme hizmet programına dallanmasıdır (Hatta IRQ kesme hizmet programında çalışılsa bile.) Bu tipteki bir kesme girişi çok özel durumlarda devreye girmesi istendiğinde kullanılır. Örneğin bir ısıtıcıyı kontrol eden bir mikrobilgisayar herhangi bir yangın meydana geldiğinde sistemin enerjisini kesip, yangın alarmını çalıştırmalı ve yangın söndürme sistemini devreye sokabilmelidir. Bu gibi durumlarda maskelenemez kesme kullanılarak acil ihtiyaçlar veya güç kesme gibi önemli işlemler için sistem devreye girebilmelidir. Maskelenemez kesme girişleri Motorola MC6802 ve Intel 8085 entegrelerinde mevcuttur. Motorola MC6800 mikroişlemci ailesinde 4 kesme bulunmaktadır. Bunlar öncelik seviyelerine göre sıralandığında;

- ❖ RST (Reset; yeniden başlat)
- ❖ NMI (Non maskable interrupt; maskelenemez kesme)
- ❖ IRQ (Interrupt; maskelenebilir kesme)
- ❖ SWI (Software interrupt; yazılım kesmesi)

Maskelenemez kesmelerden birisi de mikroişlemcinin “reset” girişi sayılabilir. Mikroişlemcinin reset girişi etkin yapıldığında tüm programlar kesintiye uğrar ve sistem, enerji verilmeden önceki konumundan tekrar başlar.



Şekil 10.5 : MC6802 mikroişlemcisindeki kesme sinyali ile birlikte gerçekleştirilen işlemler

SWI'da maskelenemez bir kesmedir. SWI kesmesi, aslında bir mikroişlemci komutudur. Kesme işlemi diğer donanım kesmelerinde olduğu gibi dışarıdan bir kesme sinyaline ihtiyaç duymaz. Buna yazılım kesmesi de denir. Komutun işlem kodu (3F)₁₆'dır. Bu komut uygulandığında mikroişlemci diğer kesmelerde olduğu gibi tüm kaydedicilerin içeriğini yığına atarak yazılım kesmesi hizmet programına dallanır.

10.1.3 Kesme Vektörleri

Mikroişlemciler ilk çalıştırıldıklarında, resetlendiğinde, kesme sinyali geldiğinde (maskelenebilir veya maskelenemez kesme) veya yazılım kesmesi komutu (SWI gibi) işletildiğinde yaptıkları ilk şey belirli sabit adreslere dallanmaktır. Bu adreslere “kesme vektörleri” denir. Kesme vektörleri sürekli aynı adreslerde bulunur. Her mikroişlemcinin kesme vektör adresleri farklıdır. Örneğin bir mikroişlemciye kesme sinyali uygulandığında, kesme hizmet programının hafızadaki yerini tespit etmek için IRQ kesme vektöründe yazılı olan adrese bakacaktır. Mikroişlemcinin bu refleksi her kesme sinyali geldiğinde gerçekleşecektir. Programcının da, kesme hizmet programının başlangıç adresini, IRQ vektör tablosundaki yere yazmış olması gerekmektedir. MC6802 'de, kesme hizmet programının başlangıç adresinin yüksek değerlikli kısmını (FFF8)₁₆ adresine, düşük değerlikli kısmı da (FFF9)₁₆ adresine yazılmalıdır. Tablo 10.1'de MC6802'nin kesme vektör tablosu bulunmaktadır. MOS. Tec. 6502 ve Motorola MC6802 mikroişlemcilerinde 1 tane IRQ (maskelenebilir kesme), 1 tane NMI (maskelenemez kesme) girişleri vardır.

Tablo 10.1 : MC6802 vektör tablosu

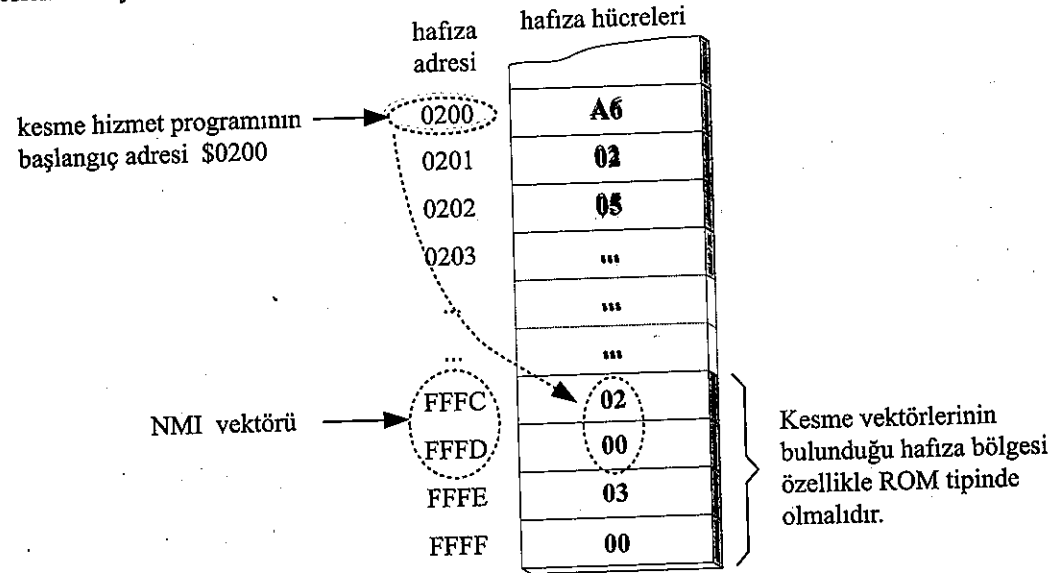
Kesme vektörü		Kesme türü
Yüksek değerlikli kısım	Düşük değerlikli kısım	
(FFFE) ₁₆	(FFFF) ₁₆	RESET (yeniden başlat)
(FFFC) ₁₆	(FFFD) ₁₆	NMI (maskelenemez kesme)
(FFFA) ₁₆	(FFFB) ₁₆	SWI (yazılım kesmesi)
(FFF8) ₁₆	(FFF9) ₁₆	IRQ (maskelenebilir kesme)

Tablo 10.2 : MOS Tec. 6502 vektör tablosu

Kesme vektörü		Kesme türü
Yüksek değerlikli kısım	Düşük değerlikli kısım	
(FFFD) ₁₆	(FFFC) ₁₆	RESET (yeniden başlat)
(FFFB) ₁₆	(FFFA) ₁₆	NMI (maskelenemez kesme)
(FFFF) ₁₆	(FFFE) ₁₆	SWI (yazılım kesmesi)
(FFFF) ₁₆	(FFFE) ₁₆	BRK (yazılım kesmesi)

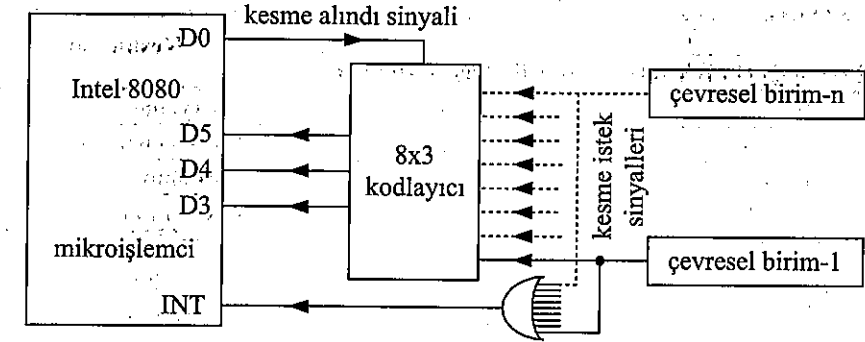
Örneğin MC6802 kullanılan bir mikrobilgisayarda, maskelenemez bir kesme sinyaline hizmet verecek kesme hizmet programının başlangıcı, hafızanın (0200)₁₆ adresinde olsun. (0200)₁₆ adresinin önceden sistemin NMI vektörüne yazılması gerekmektedir. MC6802 'de NMI kesme sinyali geldiğinde kesme hizmet programının başlangıç adresinin yüksek değerlikli kısmını almak için (FFFC)₁₆ adresine, düşük değerlikli kısmını almak için ise (FFFD)₁₆ adresine bakacaktır. NMI maskelenemez kesme girişi olduğundan, kesme vektör adresine dallanmadan önce CCR'deki kesme maskesi bayrağının kontrolünü yapmamaktadır. Daha sonra kesme vektörlerden alacağı adres bilgisini, program sayıcıya yükleyerek kesme hizmet programına dallanır.

Kesme vektörlerinin özellikle ROM tipindeki sabit hafızalara denk getirilmesindeki düşünce, sistemin enerjisi kesilse dahi, kesme vektörlerinde yazılı bulunan kesme hizmet programlarının başlangıç adreslerinin kaybolmamasının istenmesidir.



Şekil 10.6 : NMI kesme hizmet program adresinin NMI vektöründen alınışı

Tek kesme girişli mikroişlemciler üretildiği gibi, çoklu kesme girişine sahip mikroişlemciler de üretilmiştir. Intel 8080 mikro işlemcisi buna bir örnektir. Fakat Intel 8080 mikro işlemcisinde kesme işlemi MC6802 ve MOS Tec. 6502 mikro işlemcilerinden biraz farklıdır. Normalde birden fazla kesme girişi bulunan bir mikro işlemcinin, kesme isteğinde bulunan olası tüm çevre birimlerini tek tek taraması gereklidir. Fakat 8080 mikro işlemcisi, kesme isteğinde bulunan çevresel birimin, sistemin veri yoluna bir kod bırakmasını beklemektedir. Bu kod mikro işlemci tarafından okunarak hem kesme isteğinde bulunan çevresel birim tespit edilmiş hem de ilgili çevresel birimin ihtiyaç duyduğu kesme hizmet vektörüne dallanılmış olacaktır. Mikro işlemcinin veri yolundan okuyacağı kod aslında bir RST komutu kodudur. 8080 mikro işlemcisinde tanımlanmış 8 adet RST komutu vardır. Tablo 10.3'de RST komutları, komutların kodları ve adres vektörleri görülmektedir.



Şekil 10.7 : Çok kesme girişi çalışan 8080 mikro işlemcisinde kesme sinyallerinin alınışı

Örneğin çevresel bir arabirim kesme oluşturur ve kesme alındısına cevap olarak veri yoluna (D7)₁₆ kodunu bırakır ise program sayıcı (geri dönüş adresi olarak kullanılacağı için) yığına atılır ve (0010)₁₆ adresine atlama gerçekleştirilir. (0010)₁₆ adres vektöründe ilgili kesme hizmet programının ilk komutu olarak bir altyordam komutu yerleştirilir. Buradan alt programa dallanan mikro işlemci alt programdan dönüş komutu RET ile yığına dönüş adresi bulunduğundan önceki konumuna geri döner. Intel 8080'de kesme işlemleri tamamen alt program kullanma yöntemine benzerlik gösterir. 8080 mikro işlemcisinde çoklu kesme kullanmanın avantajının yanında çevresel birimlerden bir yeniden başlatma komutu sağlamasını istemesi ise bir dezavantajdır. Sistem tasarımcısı 8080 mikro işlemcisinin bu gereksinimini karşılayacak donanım çözümünü hesaba katmak zorundadır.

Tablo 10.3 : Intel 8080 mikro işlemcisinin kesme vektör tablosu

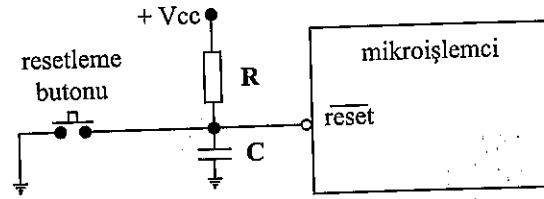
Komut	Komutun kodu	Dallanılacak vektör adresi
RST0	C7	(0000) ₁₆
RST1	CF	(0008) ₁₆
RST2	D7	(0010) ₁₆
RST3	DF	(0018) ₁₆
RST4	E7	(0020) ₁₆
RST5	EF	(0028) ₁₆
RST6	F7	(0030) ₁₆
RST7	FF	(0038) ₁₆

Kesmelerin çok kullanıldığı yoğun işlem hacmi olan sistemlerde kesmelerin öncelik seviyelerini ve art arda gelen kesmeleri sıraya sokmak gibi çok karmaşık işlemleri organize edebilmek için özel entegre elemanlar kullanılır. Öncelik seviyesi, yetki kontrol mekanizmasını ve kesme sırasını organize eden Intel 8259 programlanabilir kesme denetleyicisi gibi denetleyici entegreler vardır.

10.2 RESTART (Yeniden Başlatma)

Restart veya reset işlemi (yeniden başlama) mikroişlemcinin reset girişinin aktif edilmesi ile gerçekleşir. Bütün mikroişlemcilerde mutlaka bir resetleme ucu bulunmaktadır. Bazı mikroişlemcilerin reset girişi "0" da iken bazılarında "1" de aktif olur. Mikroişlemciler resetlendiklerinde tüm kaydedicilerdeki bilgiler sıfırlanır ve mikroişlemci, enerji verildiği ilk durumuna geçer. Bütün mikroişlemciler resetlendiklerinde çalışmaya, belirli bir adresten başlarlar. Bazı mikroişlemciler ilk çalışma adreslerini adres vektör tablosu olarak kullanırlarken bazıları çalışmaya bu adrese yazılmış programdan başlarlar. Eğer başlangıç adresi vektör tablosu olarak kullanılmış ise bu adres vektörüne asıl programın başlangıç adresi yazılır. Kesme vektörleri konusunda, reset için ayrılmış vektör adresleride belirtilmiştir.

Mikroişlemciler resetlendiğinde veya ilk enerji verildiğinde, hazır hâle gelebilmeleri için belirli bir süre geçirmektedirler. Bu süre boyunca mikroişlemcinin reset ucunun etkin kalması şarttır. Bu sürenin çok kısa olmasına karşılık, mikroişlemcinin kendisini toparlaması için gerekli minimum süre sağlanmadığı takdirde mikroişlemci çalışmaya başlamadan kilitlenebilir. Bu süre en güvenilir şekilde bir RC devresi ile sağlanabilir. Her mikroişlemcinin toparlanma süreleri farklıdır. RC zaman sabitini, bu süreler göz önüne alınarak hesaplamak gerekir. Şekil 10.8'de "0" sinyali ile etkin olan reset girişine sahip bir mikroişlemciye bağlanacak RC devresi görülmektedir.



Şekil 10.8 : Sıfırda etkin reset girişin bağlanacak resetleme devresi

Resetleme butonuna basıldığında mikroişlemci resetlenmiş olacaktır. Fakat çok küçük bir süre olmasına rağmen, butona yeterli süre basılmadığı durumlarda mikroişlemcinin resetleme işlemini tamamlaması için yeterli süreyi RC sağlayacaktır. Kondansatör belli bir seviyeye doluncaya kadar, resetleme girişi sıfıra çekilmiş olacaktır. R ve C'nin değerleri değiştirilerek gerekli süre sağlanmalıdır.



MC6802'nin Resette Beklerken Kontrol Bacaklarındaki Sinyal Seviyeleri:

MC6802 mikroşlemcisi sürekli reset'te tutulursa BA çıkışı "0", VMA çıkışı "0", R/ucu "1" ve adres yolunda (FFFF)₁₆ değeri görülecek şekilde bekler. Bu arada tüm kaydediciler sıfırlanmış olacaktır. Dikkat edilirse mikroşlemcinin reset ucu serbest bırakıldığı anda, reset vektörünün yüksek değerlikli (FFFF)₁₆ kısmını okuyacak şekilde hazır beklemektedir.

DEĞERLENDİRME ÇALIŞMALARI

- 1) Kesme ne demektir ve kaç çeşit kesme sinyali vardır.
- 2) Mikroşlemcilerde neden kesme girişi bulundurulmuştur?
- 3) Kesme vektörü ne demektir? Değişik mikroşlemcilerin kesme vektör adreslerini yazınız.
- 4) Kesmeleri öncelik seviyesine göre sıralayınız.
- 5) Mikroşlemciye maskelenebilir bir kesme geldiğinde nasıl bir yol izler? Anlatınız.
- 6) Maskelenebilir kesmeler nasıl önlenir?
- 7) Mikroşlemciye maskelenemez bir kesme geldiğinde nasıl bir yol izler? Anlatınız.
- 8) Mikroşlemci program içerisinde bir yazılım kesmesi komutu ile karşılaştığında nasıl bir yol izler?
- 9) Mikroşlemci girişine Reset sinyali uygulandığında nasıl bir yol izler? Anlatınız.

ÜNİTE 11 MİKROBİLGİSAYARLARDA HAFİZA ORGANİZASYONU

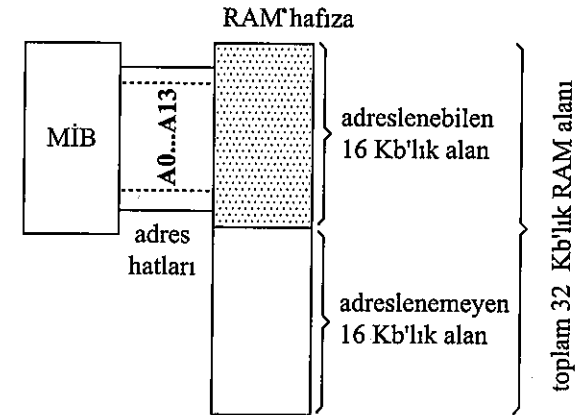
11.1 MİKROİŞLEMCİNİN HAFİZA ALANI

Mikroşlemcilerin fiziksel olarak adresleyebilecekleri maksimum adres alanı, o mikroşlemcinin "hafıza alanı" demektir. Bir mikroşlemcinin adresleyebileceği maksimum hafıza alanını, mikroşlemcinin adres bacağı sayısı belirler. Bir mikroşlemcinin adresleyebileceği en büyük hafıza alanını hesaplamak için aşağıdaki formülden yararlanılır.

$$2^{(\text{Mikroşlemcinin toplam adres hattı sayısı})} = \text{Adresleyebileceği maksimum hafıza alanı}$$

Basit mikrobilgisayarlarda bulunan mikroşlemcilerin adresleyebileceği hafıza alanı yaklaşık 64 Kb iken (küçük sistemler için yeterli bir hafıza alanı), kişisel bilgisayarlarda kullanılan mikroşlemciler 16 Mb (24 bitlik adres hattı olan mikroşlemciler), hatta 4 Gb'lık (32 bitlik adres hattı olan mikroşlemciler) bir hafıza alanını adresleyebilmektedirler.

Örneğin; bir mikroşlemcinin 14 adres bacağı ($A_0 \dots A_{13}$) varsa bu mikroşlemcinin adresleyebileceği maksimum hafıza alanı $2^{14} = 16384$ byte veya 16 Kb eder. Eğer bu mikroşlemciye yalnızca RAM hafıza bağlanacak olsaydı, bağlanacak RAM'ın kapasitesi en fazla 16 Kb olmalıydı. Çünkü daha büyük kapasitede bir RAM bağlanacak olsa mikroşlemci, bağlanan bu RAM'ın sadece 16 Kb'lık kısmına erişebilecek ve RAM'ın diğer alanları kullanılamayacaktı. Bunları Şekil 11.1'i inceleyerek kavramak daha kolay olacaktır.

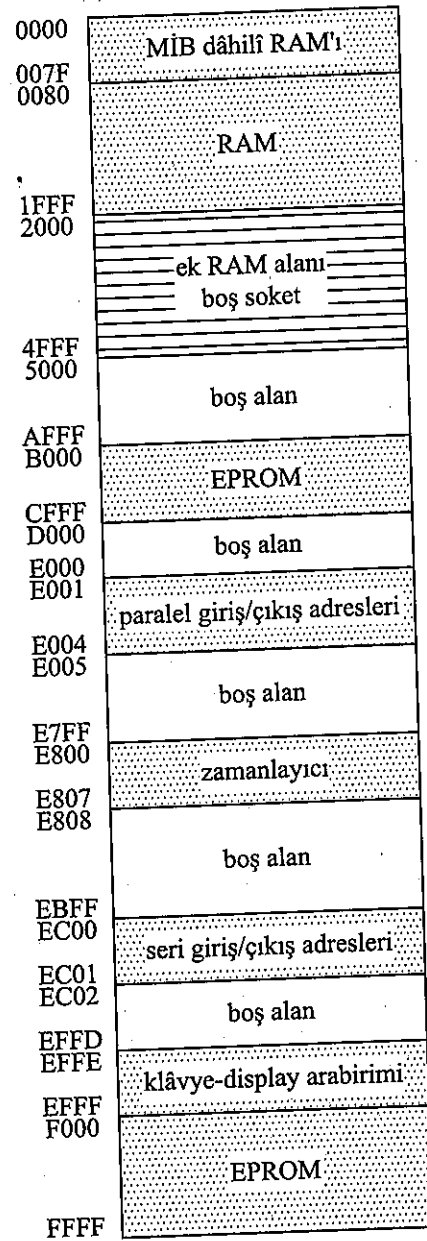


Şekil 11.1 : Tüm hafızayı adreslemeye yetmeyen bir MİB adres hatları

Mikrobilgisayarlarda RAM, EPROM, EEPROM gibi yalnızca hafıza elemanları bulunmaz. Giriş/çıkış elemanları, zamanlayıcılar, A/D dönüştürücüler gibi birçok çevresel birimler de bulunabilir. Mikroşlemciler, hafıza elemanları ve diğer çevresel birimler ile yapacağı tüm veri iletişimlerini veri yolu üzerinden gerçekleştirir. Ancak istediği birimden veri akışını sağlayabilmesi için hafıza elemanlarının ve diğer çevresel birimlerinin, hafıza alanı içerisinde olması şarttır. Hafıza alanının, hiçbir birimle çakışmayacak şekilde ayrı ayrı paylaştırılması gerekir. Toplam hafıza alanı, değişik çevre birimlerine paylaştırılırken dikkat edilecek en önemli nokta, hafıza birimlerinin ve diğer çevresel birimlerin, hiç birisinin diğerinin adresi ile çakışmayacak şekilde yapılmasıdır. Bu alanlar mikrobilgisayar sistemi tasarlanırken genel kurallara uymak kaydıyla istendiği şekilde yapılabilir.

Aşağıda, 16 adres bacağı bulunan bir mikroşlemcinin, adresleyebileceği maksimum alan olan 64 Kb'lık hafıza alanının, hafıza ve diğer çevresel birimlerine nasıl paylaştırıldığını gösteren Şekil 11.2 bulunmaktadır. Bu türden bir mikroşlemcinin adresleyebileceği maksimum hafıza alanının, çevresel birimlerine nasıl pay edildiğini gösteren diyagramlara "hafıza haritası" denilir.

Şekil 11.2'deki hafıza haritasında, mikroişlemcinin adresleyebileceği toplam 64 Kb'lık bir hafıza alanı vardır. Bu hafıza alanında bulunan her byte'a karşılık bir adres gelecek şekilde, $(0000)_{16}$ ile $(FFFF)_{16}$ arasındaki sayılar ile adreslenmiştir. Hafıza haritasında dikkat edilecek olursa hiçbir çevresel elemana atanmamış adres aralıkları vardır. Mikroişlemci için bu alanlar, hiçbir anlam ifade etmez, bu alanlara veri yazılamaz ve okunamaz. Daha sonra bu alanlara yeni çevresel birimler veya hafızalar eklemek sureti ile sistem geliştirilebilir. Şekil 11.2'de görüldüğü gibi sistemdeki çevresel elemanlardan veya hafıza birimlerinden hiçbirinin adresi diğeri ile aynı değildir. Bu çok önemli bir noktadır. Eğer çevresel elemanlar ve hafıza birimleri bu şekilde adreslenmeseydi adres çakışmaları oluşacaktı.



Şekil 11.2 : Tipik bir mikrobilgisayar sisteminde hafıza haritası

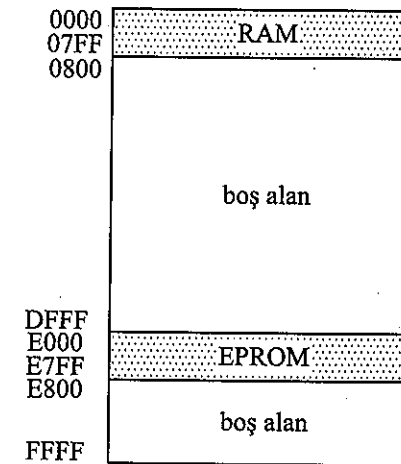
Eğer sistemde bir adres çakışması olsaydı örneğin; adres yoluna gönderilen $(0FFF)_{16}$ adresi aynı anda hem EPROM'daki bir hafıza hücreğini hem de RAM'daki bir hafıza hücreğini adresleyecekti. Böyle bir durumda sistemin sağlıklı çalışması mümkün değildir. Tasarımcı, herhangi bir elemanın "toplam hafıza alanında" işgal edeceği alanı da hesaba katarak tüm sistemi bir bütünlük içinde ele almalı sistemi önce kâğıt üzerinde şekillendirmelidir. Tüm çevresel elemanların ve hafıza birimlerini, sistemin hafıza haritasında uygun yerlere yerleştirdikten sonra bu elemanların adres yolundan gönderilen bir adrese karşılık etkin olmasını sağlayacak tedbirler alınmalıdır. Bu işlemin nasıl yapılacağı "hafıza paylaşımı" konusunda incelenmiştir.

11.2 HAFIZA PAYLAŞIMI

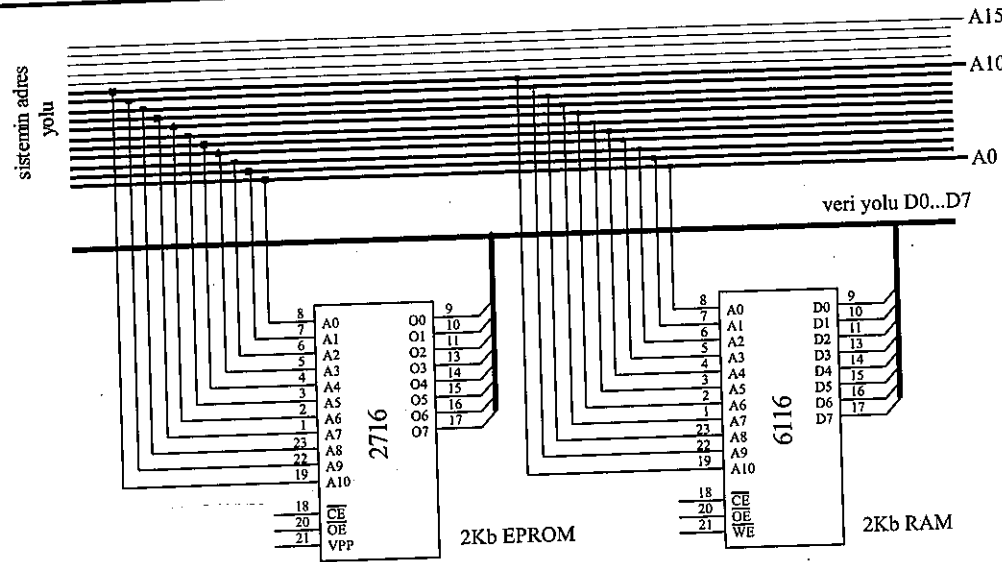
Bir mikroişlemci sistemindeki hafıza paylaşımındaki amaç, kapasiteleri belli olan hafıza elemanlarından gerektiği kadarını kullanarak sistemin hafıza haritasındaki belirlenen yerlerine yerleştirmek olacaktır. Aynı zamanda mikroişlemcinin toplam hafıza alanına, diğer çevresel birimlerin hafıza haritasındaki yerlerine yerleştirilmesi yapılacaktır. Mikrobilgisayar sistemindeki adres ve veri yoluna, tüm birimlerin fiziksel bağlantıları, mikroişlemcinin adresleyebileceği maksimum alan içerisinde bulunması şarttır.

11.2.1 Tek RAM ve ROM'dan Oluşan Sistemler

Bir mikrobilgisayarda 2 Kb RAM ve 2 Kb EPROM hafıza birimleri kullanılacaksa bunların sistem yollarına bağlantısını yapmadan önce RAM ve EPROM'un genel hafıza alanında, hangi adreslerde bulunacağına karar vermek gerekir. Bunu yaparken sistemin genel bir hafıza haritasını çıkarmak en akılcı yaklaşım olacaktır. Daha sonra RAM ve EPROM, hafıza haritasında belirlenen bu adreslere yerleştirilebilir. Şekil 11.3'de, RAM'ın $(0000)_{16}$, EPROM'unda $(E000)_{16}$ adreslerinden başlayarak adreslenmesi için yapılan yerleştirmenin, hafıza haritasındaki durumu görülmektedir. Her ikisi de 2'şer Kb olduğu için RAM $(0000)_{16} \dots (07FF)_{16}$, EPROM'da $(E000)_{16} \dots (E7FF)_{16}$ adresleri arasındaki bölgeyi kaplayacaktır.



Şekil 11.3 : 2 Kb'lık RAM ve EPROM'un hafıza haritasındaki yerleşimi



Şekil 11.4 : 1 Kb'lık RAM ve 1 Kb'lık EPROM'un, sistemin adres yoluna bağlantıları

Hafıza haritasında belirlenen yerlere göre, RAM ve EPROM'un adres bacakları ile sistemin adres yolu arasındaki bağlantılar yapılır. Kullanılan RAM ve EPROM'un kapasitesi 2 Kb'tır ve adres bacağı sayısı 11 tanedir. Dikkat edilirse RAM ve EPROM'un adres bacakları aynı adres hatlarına bağlanmıştır ve ilk bakışta bir çakışma görülmektedir. Mikroişlemcinin, adres yoluna bıraktığı $(0000)_{16} \dots (07FF)_{16}$ veya $(E000)_{16} \dots (E7FF)_{16}$ arasındaki adres değerlerinin RAM ve EPROM'daki aynı hücreleri işaret etmesini önlemek gerekecektir. Ayrıca programcının, program içerisinde kullanacağı $(E000)_{16} \dots (E7FF)_{16}$ arasındaki adreslerin EPROM'u, programcının, program içerisinde kullanacağı $(0000)_{16} \dots (03FF)_{16}$ arasındaki adreslerin de RAM hafızadaki bir hücreyi işaret ettiğinden emin olması gerekmektedir. Gerçekten EPROM'un veri hücreleri ile RAM'daki veri hücreleri, aynı $(0000)_{16} \dots (07FF)_{16}$ adresleri arasındadır. Fakat biz EPROM'un adres yolunda $(E000)_{16} \dots (E7FF)_{16}$ adresleri arasında görünmesini istiyoruz. Bu gibi çakışmaları gidermek için RAM ve EPROM'larda bulunan çip seçme (CS; chip select) uçlarını kullanarak sistemin hafıza organizasyonunu yapmak gerekir.

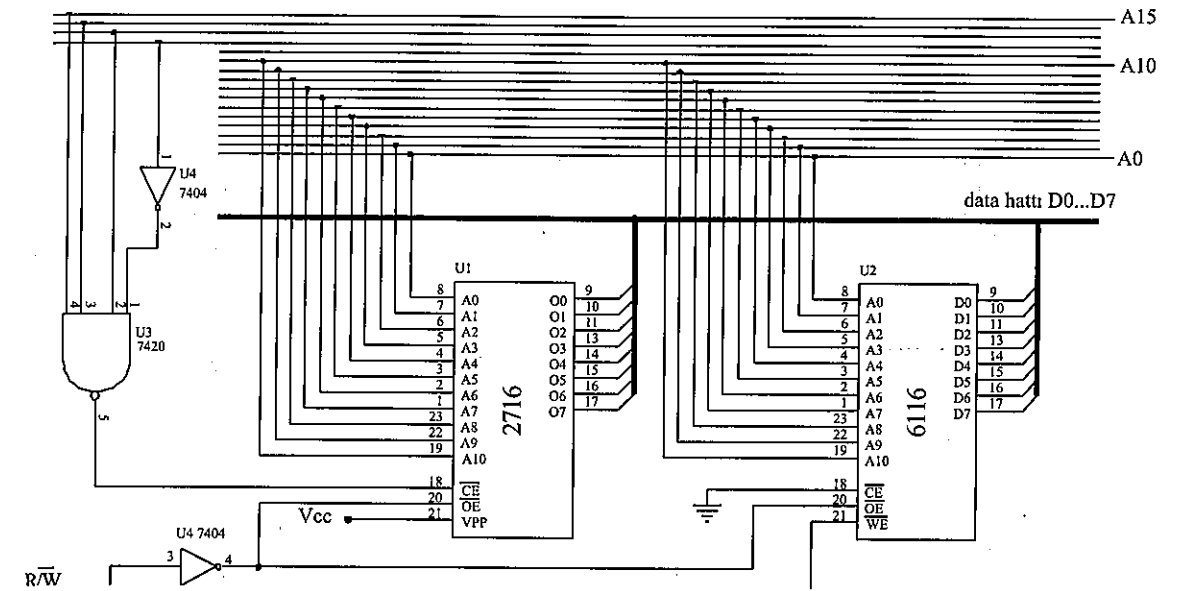
Tasarımcının sistemdeki bu tip adres çakışmalarını önlemek için donanımda birtakım düzenlemeler yapması gerekmektedir. Genellikle adres yolundaki yüksek değerlikli bitler, çip seçme uçlarında değerlendirildiklerinde bu tip çakışmalar giderilebilmektedir. Bu sistemde de adres yoluna bırakılan $(E000)_{16}$ adresindeki yüksek değerlikli bitler çip seçme ucunda kullanılırsa sorun halledilmiş olacaktır. Mikroişlemci EPROM hafızaya erişmek için adres yoluna $(E000)_{16} \dots (E7FF)_{16}$ adresleri arasında bir değer bıraktığı zaman, hem veri iletişiminin EPROM ile yapılması hem de EPROM'un seçilmesi sağlanmış olacaktır.

EPROM, sistemin adres yoluna bırakılan $(E000)_{16} \dots (E7FF)_{16}$ arasındaki adres değerlerini her zaman için $(0000)_{16} \dots (07FF)_{16}$ arasındaki adresler gibi okuyacaktır. Çünkü sistemin adres yolundaki $A_{11} \dots A_3$ hatlarındaki değerler, EPROM tarafından okunmadığından, bu hatlarda oluşan adres değerleri EPROM'un CS (chip select) ucunu etkileyen bir kontrol sinyali olarak kullanılmıştır. $A_{10} \dots A_0$ adresleri ile EPROM veya RAM hafızalarının tüm hücrelerine erişilebilirken $A_{15}, A_{14}, A_{13}, A_{12}$ hatları ile de EPROM'un seçilmesi için gerekli olan sinyaller elde edilmiştir.

Tablo 11.1 : Hafıza birimlerini adreslemede kullanılan adres hatları

	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
RAM	X	X	X	X	X	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
EPROM	1	1	1	0	X	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

0/1; EPROM veya RAM'ın adres hattı bağlantıları
1 veya 0; EPROM'u seçmede kullanılan adres hatları
X; Seçmede etkisi olmayan hatlar



Şekil 11.5 : Tek RAM ve tek EPROM'un, adres yoluna bağlantısı

EPROM'da bulunan seçme girişi, bu elemanın etkin yapılmasında kullanılır. Bu giriş lojik "0" yapılmadan EPROM'dan herhangi bir veri okumak veya yazmak (EPROM'un programlanması sırasında) mümkün değildir. Adres yolundaki $A_{15}, A_{14}, A_{13}, A_{12}$ hatları ile EPROM'un ucu için gerekli olan lojik "0" sinyali bir NAND kapısı ve bir NOT kapısı ile sağlanır. Lojik "0" in elde edilmesi için adres yolundaki A_{15}, A_{14}, A_{13} hatlarında lojik "1" sinyalinin bulunması gerekmektedir. Ancak bu üç hattın lojik "1" olması yeterli değildir. Ayrıca A_{12} hattında lojik "0" olması bu sinyalinde terslenerek NAND kapısına uygulanması bize gerekli olan $(EXXX)_{16}$ adreslerini sağlayacaktır.

Bu duruma göre EPROM'un seçilebilmesi ancak, A_{15}, A_{14}, A_{13} hatlarında lojik "1" ve A_{12} hattında da lojik "0" olması ile mümkündür. Yani adres yoluna $(EXXX)_{16}$ değerlerinden başka bir değer gönderildiğinde EPROM'un seçilmesi ve adreslenmesi mümkün olmayacaktır. EPROM ve RAM'ın seçilmesi, daha birçok yöntemle sağlanabilir. Bu tasarımcının sistemi ve hafıza organizasyonunu ne kadar iyi yapabildiğine bağlıdır. Bu örnekler üzerinde anlatılanlar, hafıza organizasyonu hakkındaki yaklaşımın, nasıl olması gerektiği hakkında bilgi vermelidir.



Hafıza Birimlerinin Seçme Uçları Yüksek Değerlikli Adres Hatlarından Elde Edilmelidir

Mikrobilgisayar sistemlerinde kullanılan tüm hafıza birimlerinin adres hatları (özel bir tasarım düşünülmeyişi sürece) sistemin adres yoluna, A_0 adres hattından başlanarak bağlanmalıdır. Böylece mikroşlemci, adres yoluna bıraktığı $(0000)_{16}$ dan başlayan adresler ile tüm hafıza birimlerindeki verilere, sıralı bir şekilde erişebilir. Hafızaya erişirken kullanılan en büyük adres değeri sistemde bulunan hafıza biriminin kapasitesi ile ilişkilidir. Örneğin kullanılan hafıza birimi 8 Kb ise bu hafızanın son veri hücrelerinin adresi $(1FFF)_{16}$ olacaktır.

Hafıza birimlerinin adreslenmesinde kullanılmayan, en yüksek değerlikli adres hatları (A_{15}, A_{14}, A_{13} gibi) genellikle hafıza birimlerinin seçiminde kullanılır. Hafıza birimlerinin seçimi ve yetkilendirilmesi hafıza organizasyonunda ayrıntılı olarak düşünülmesi gereken bir noktadır.

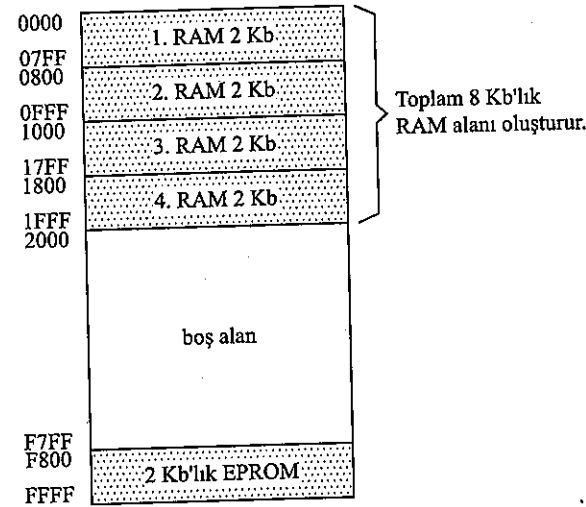
11.2.2 Birden Fazla RAM ve ROM'dan Oluşan Sistemler

Mikrobilgisayarlarda bazen fazla miktarda hafıza alanı gerekebilir. Piyasada ihtiyacımızı karşılayacak büyüklükte tek bir hafıza elemanı bulunmadığı takdirde, bu ihtiyacımızı birden çok hafıza elemanını kullanarak karşılama yoluna gidebiliriz. Fakat birden çok hafıza elemanı kullanıldığında, hafıza elemanlarının iyi organize edilmesi şarttır. İyi bir organizasyon için tek RAM ve tek EPROM'un bulunduğu sistemdeki gibi, işe önce hafıza haritasını çıkarmakla başlamamız gerekir. Hafıza haritasında hangi hafıza elemanının hangi aralıkta olacağına karar verilir. Kullanılan hafıza elemanlarının, kaç tane seçme girişinin bulunduğu ve bu girişlerin hangi durumda etkin olacağı el kitaplarından araştırılmalıdır. Konunun daha iyi anlaşılması için örnek bir hafıza organizasyonu aşağıda anlatılacaktır.

Elimizde kapasiteleri 2'şer Kb olan 4 adet RAM ve 2 Kb olan 1 adet EPROM olsun. Kullanılan RAM'larda ve EPROM'da tek seçme ucu bulunmaktadır. Her hafızanın seçme ucu "0" olduğunda etkin olmaktadır. Her biri 2 Kb olan hafızaların adresleme aralığı $(0000)_{16} \dots (07FF)_{16}$ olacaktır. 16 bit adres yoluna sahip bir mikroişlemci toplam $2^{16} = 64$ Kb'lık bir aralığı adresleyebilir. Kullandığımız hafıza elemanlarının toplam kapasitesi ise; her biri 2 Kb olan RAM'ların toplamı 8 Kb ve EPROM'un kapasitesi de 2 Kb olduğundan, tüm hafıza elemanlarının sistemde toplam kapasitesi 10 Kb olacaktır. Toplam bu kapasitedeki hafızayı 16 bitlik bir mikroişlemci rahatlıkla adresleyebilecektir. Bu hafıza birimlerinin hafıza haritasındaki yerleşimi şekil 11.6'deki gibi olacaktır.

Şekil 11.6'daki hafıza haritası incelendiğinde RAM'lar art arda bağlanarak $(0000)_{16} \dots (1FFF)_{16}$ arasında sürekli bir hafıza alanı elde edildiği görülür. Mikroişlemci $(0000)_{16} \dots (1FFF)_{16}$ aralığında 2'şer Kb'lık 4 adet RAM hafızayı, tek bir 8 Kb'lık RAM hafıza gibi adresleyebilecektir.

Hafıza birimlerinin sisteme bağlanabilmesi için öncelikle bunların hafıza haritasındaki yerlerine göre kod çözümünde kullanılacak adres hatları belirlenir. Hafıza birimlerinin bacak bağlantıları tablo 11.2'deki gibi adres yoluna bağlanır. Yüksek değerlikli adres hatları ise EPROM ve RAM'ların seçimi için kullanılmaktadır.



Şekil 11.6 : Sistemin hafıza haritası

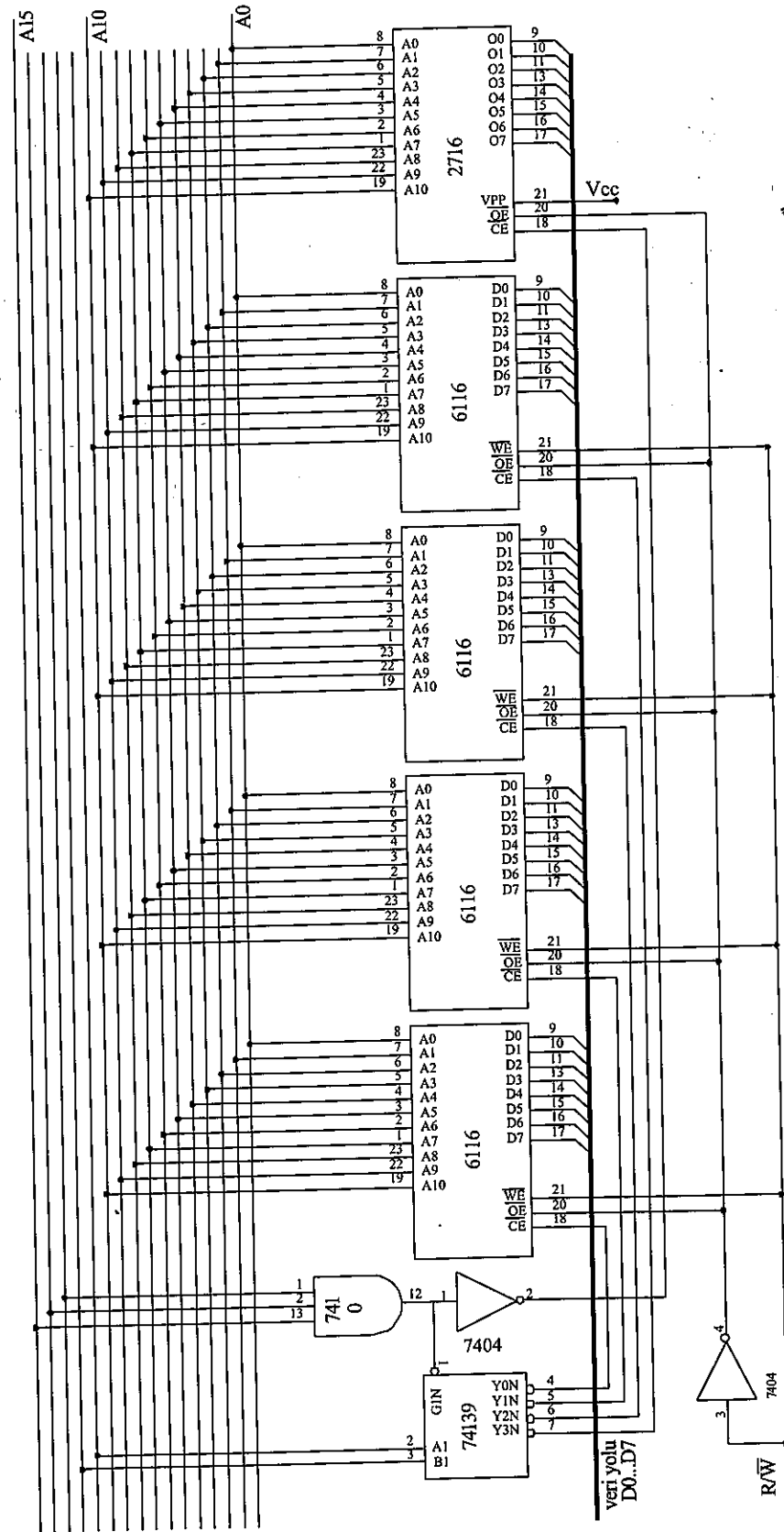
A_{15} , A_{14} , A_{13} hatlarına bağlanan AND kapısı, EPROM için gereken seçme sinyalini üretmede kullanılacaktır. A_{15} , A_{14} , A_{13} hatlarının üçünün de aynı anda "1" olması ile AND kapısının çıkışı "1" olacaktır. AND kapısından hemen sonra bağlanan NOT kapısı EPROM'un CE girişi için gerekli olan "0" bilgisini üretmiş olacaktır. Tablo 11.2 ve şekil 11.7'deki devre beraber incelendiğinde bu durum açıkça görülmektedir. EPROM ve RAM adres çakışmalarını engellemek için AND kapısının çıkışı kodlayıcının yetkilendirme girişine bağlanmıştır. Bu durumda EPROM seçildiğinde kodlayıcı çıkışlarının "1" yapılacaktır. Kodlayıcının tüm çıkışlarının "1" olması, EPROM adreslendiğinde herhangi bir RAM hafızanın da seçilmesini ve adreslenmesini engelleyecek bir önlemdir.

Şekil 11.7'deki devrede görüleceği gibi A_{11} ve A_{12} hatları 74139 kod çözücü (2'den 4'e kod çözücü) entegresine bağlanmıştır. Kod çözücünün çıkışları RAM'ların CE bacaklarına bağlanarak kod çözücünden elde edilen seçme bilgisi ile RAM'ların belirli adres aralıklarında seçilmesi sağlanmış olur. A_{11} ve A_{12} hatlarında oluşabilecek dört durum vardır ve bu durumları ayrı bir doğruluk tablosu olarak düşünebiliriz. Bu tablodan elde edilen bilgiler ile RAM'ların tek tek seçilmesi sağlanacaktır.

Tablo 11.2 : RAM ve EPROM'un sisteme bağlanan adres bacağı bağlantıları

	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
1. RAM $(0000)_{16} \dots (07FF)_{16}$	X	X	X	0	0	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
2. RAM $(0800)_{16} \dots (0FFF)_{16}$	X	X	X	0	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
3. RAM $(1000)_{16} \dots (17FF)_{16}$	X	X	X	1	0	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
4. RAM $(1800)_{16} \dots (1FFF)_{16}$	X	X	X	1	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
EPROM $(F800)_{16} \dots (FFFF)_{16}$	1	1	1	X	X	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

0/1; EPROM veya RAM'ın adres hattı bağlantıları
1 veya 0; RAM ve EPROM'u seçmede kullanılan adres hatları
X; Adreslemede etkisi olmayan hatlar



Şekil 11.7 : EPROM ve RAM'ların mikrobilgisayar sistemindeki bağlantıları

DEĞERLENDİRME ÇALIŞMALARI

- 1) Mikroişlemcilerin hafıza alanları ne demektir?
- 2) 16 adres hattı olan bir mikroişlemcinin adresleyebileceği maksimum hafıza alanı nedir?
- 3) Mikrobilgisayarların hafıza haritaları sistem hakkında ne tür bir bilgiler içerir?
- 4) Hafızanın $(0000)_{16}$ adresinden başlayan 1 Kb'lık RAM ile hafızanın $(F000)_{16}$ adresinden başlayan 1 Kb'lık EPROM'un hafıza haritasındaki yerini belirtiniz.
- 5) Her biri 1 Kb olan dört adet RAM hafızayı Hafızanın $(4000)_{16}$ adresinde itibaren yerleştirecek şekilde hafıza haritasını hazırlayıp devresini çizin.
- 6) Hafıza birimlerinin seçiminde, adres yollundaki hangi hatlar daha çok tercih edilmelidir?
- 7) 2 Kb'lık 6116 RAM hafızanın veri, adres ve kontrol bacakları hangileridir? Kontrol bacakları bir mikrobilgisayar sisteminde nerelere bağlanmalıdır?

ÜNİTE 12 MİKROİŞLEMCİLERDE TEMEL GİRİŞ/ÇIKIŞ KAVRAMLARI

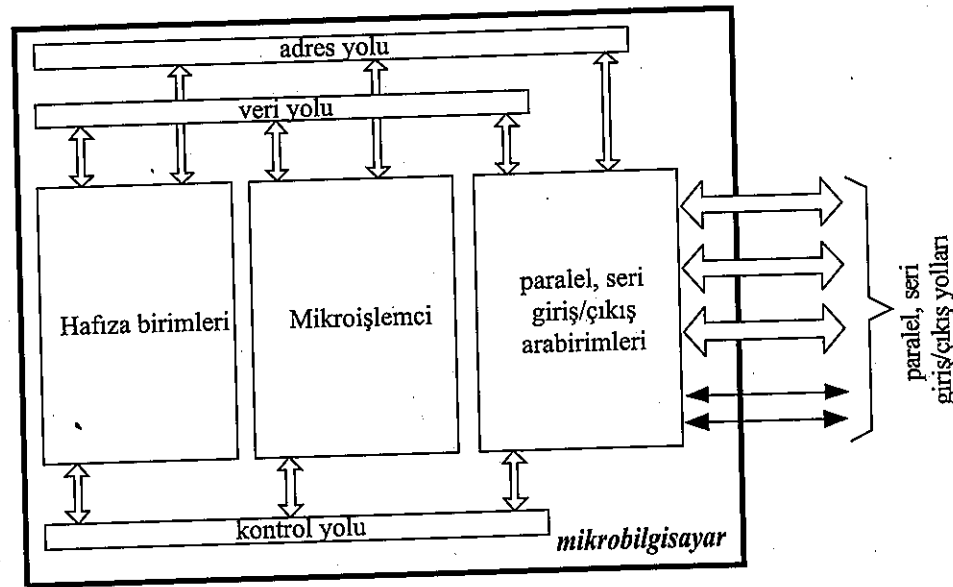
12.1 GİRİŞ/ÇIKIŞ ARABİRİMLERİ

Mikrobilgisayarlar genel olarak çevre birimlerinden aldığı verileri çok hızlı bir şekilde işleyerek sonuçlandıran ve sonuçları da yine çevre birimleriyle dış dünyaya aktararak bulunduğu sistemlerin işleyişini düzenleyen birimlerdir. Mikrobilgisayarlar dış dünya ile sürekli iletişim içerisinde bulunurlar. Mikrobilgisayarların dış dünya ile iletişimini sağlamada kullanılan birimlere genel olarak “giriş/çıkış arabirimleri”¹ denir. Mikrobilgisayarlar giriş/çıkış arabirimleri ile yazıcı, klavye, dijital/analog veya analog/dijital çeviriciler, kod çözücüler ve bunlar gibi daha birçok harici birimle bağlantı sağlarlar. Bu bağlantıları sağlayan giriş/çıkış arabirimleri iki çeşittir.

1-Paralel arabirimler

2-Seri arabirimler

Bu iki iletişim arabirimleri kullandıkları iletişim yöntemlerinden dolayı bu isimleri almışlardır. Mikroişlemcilerin dış dünya ile bağlantısını sağlayan giriş/çıkış arabirimlerinden paralel arabirlere “PIA” (parallel interface adaptor), seri arabirimlere ise “ACIA” (asynchronous communication interface adaptor) denilmektedir.



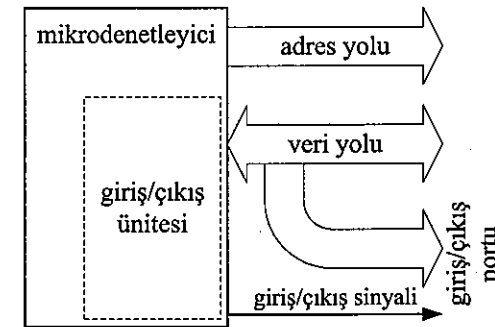
Şekil 12.1 : Bir mikrobilgisayarın dış çevre ile bağlantısını sağlayan giriş/çıkış arabirimleri

12.2 MİKROİŞLEMCİLERDE GİRİŞ/ÇIKIŞ ARABİRİMLERİNİN ADRESLENMESİ

Mikrobilgisayarların, dış dünya ile bağlantısını sağlayan giriş/çıkış arabirimlerinin adreslenmesinde üç yöntem kullanılmaktadır. Bu yöntemler;

- ❖ Tek yongalı mikroişlemciler üretilerek giriş/çıkış yapma tekniği
- ❖ Port yaklaşımli giriş/çıkış tekniği
- ❖ Hafıza haritalı giriş/çıkış tekniği

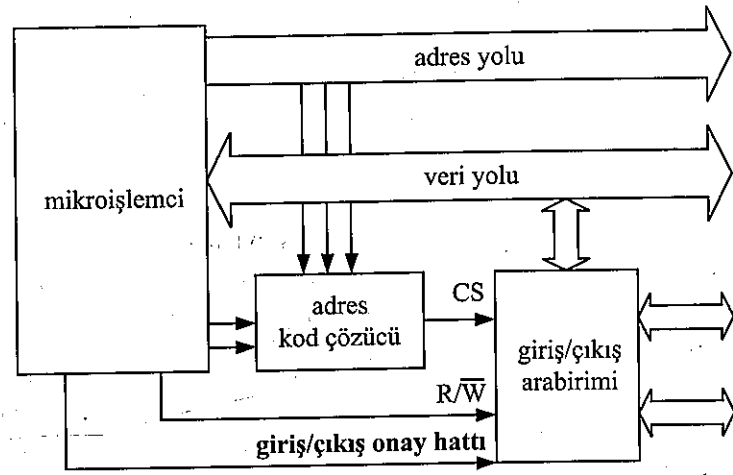
Tek yongalı mikroişlemcilerde giriş/çıkış tekniği: Bu mikroişlemcilerin çevre birimleri ile iletişimini sağlamak için kullanılan PIA ve ACIA gibi iletişim elemanları mikroişlemci içerisine entegre edilmiş hâldedir. Böylelikle sistem daha basit hâle getirilmiştir. Bu tip mikroişlemcilere “mikrodenetleyici” de denilmektedir. Bu mikroişlemcilere en iyi örnek olarak Intel'in 8051 ve COP420L mikroişlemcileri verilebilir. Bu işlemcinin seri ve paralel giriş/çıkış için kullandığı iletişim arabirimleri entegre içerisindedir ve giriş/çıkış uçları entegrenin üzerindedir.



Şekil 12.2 : Tek yongalı mikroişlemcilerde giriş/çıkış tekniğinin kullanılması

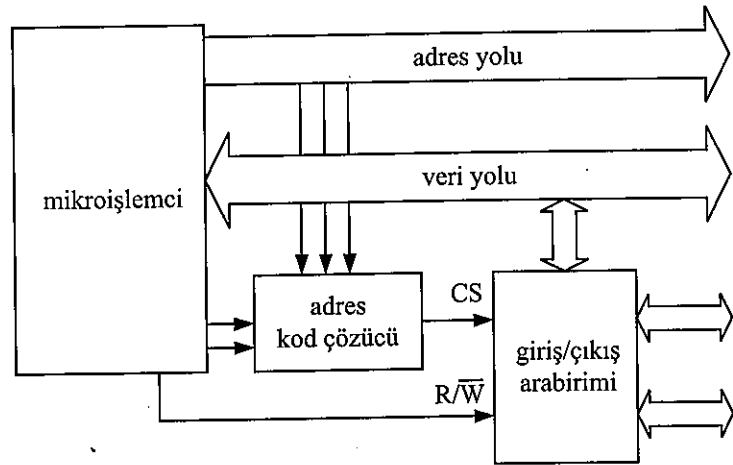
Port yaklaşımli giriş/çıkış tekniği: Bu tekniğe göre üretilen mikroişlemcilerin giriş/çıkış işlemlerini sağlayan özel komutları (IN, OUT) vardır. Bu komutlar ile istenen veri giriş/çıkış portundan dış dünyaya aktarılır veya dış dünyadan alınır. Bu tip bir yaklaşımı destekleyen mikroşlemcilerde veri yoluna bırakılan veya okunan verinin bir hafıza elemanına mı veya giriş/çıkış arabirimine mi aktarılacağını bir kontrol sinyali ile bildirir. Örneğin Z80 mikroşlemcisi veri yoluna “IN” veya “OUT” komutuyla bir veri bıraktığında mikroşlemcisinin IRQ sinyali “0” olur. Bu sinyal giriş/çıkış elmanını yetkilendirerek verinin giriş/çıkış kaydedicileri üzerinden dış dünyaya taşır veya dış dünyada okur. Giriş/çıkış elemanları ile hafıza elemanları mikroşlemcisinin veri yolunu ortak kullandıklarından okunan/yazılan verinin, hafızaya mı yoksa bir giriş/çıkış birimine mi iletileceği bu kontrol sinyali ile belirlenir. Bu kontrol sinyali giriş/çıkış komutları (IN, OUT) ile birlikte otomatikman oluşur ve programcının bunu sağlamak için yapması gereken özel bir işlemi yoktur. Fakat sistemin donanım mimarisinde bu kontrol sinyali, mutlaka giriş/çıkış işlemi yapan arabirimler ile irtibatı sağlanmalıdır. 16 bitlik adres hattına sahip bir mikroşlemcisinin adresleyebileceği maksimum hafıza alanının 65536 olduğunu biliyoruz. Port yaklaşımli giriş/çıkış tekniğini kullanan mikroşlemciler, port işlemlerini hafıza işlemlerinden ayırmak için ayrı bir kontrol hattı kullandığından teorik olarak giriş/çıkış arabirimlerinin adreslenmesinde de tıpkı hazıraların adreslenmesinde olduğu gibi 65536 byte'lık bir alanın adreslenebileceği düşünülebilir. Fakat çoğu mikroşlemcide bu iş için adres yolunun ilk 8 biti giriş/çıkış işlemlerinde kullanıldığından genellikle adreslenecek çevre birimi sayısı 256'yı geçmez.

¹ “Giriş/çıkış arabirimi” ifadesi özel bir arabirim için belirtilmediği sürece seri veya paralel giriş/çıkış arabirimleri için kullanılmaktadır.



Şekil 12.3 : Port yöntemi giriş/çıkış tekniğini kullanan mikroişlemci

Hafıza haritalı giriş/çıkış tekniğini kullanan mikroişlemcilerde özel olarak giriş/çıkış işlemlerinde kullanılan komutlar bulunmamaktadır ve ayrıca giriş/çıkış sinyali oluşturulmamaktadır. Bu tip mikroişlemcilerde herhangi bir veriyi hafıza hücresine yazmak/okumak ile bir iletişim arabirimine yazmak/okumak arasında hiçbir fark yoktur. Bu yöntemde mikroişlemci her çevre birimini sanki hafızanın bir hücresiymiş gibi adresler. Tasarımcı, hafıza haritasının bir bölümünü giriş/çıkış arabirimlerinin adreslerini kullanmak için ayırır. Bu da sistemin genel hafızasından yer işgal ettiği için bir dezavantaj olarak düşünülebilir.



Şekil 12.4 : Hafıza haritalı giriş/çıkış tekniğini kullanan mikroişlemci

12.3 PARALEL GİRİŞ/ÇIKIŞ ARABİRİMİ

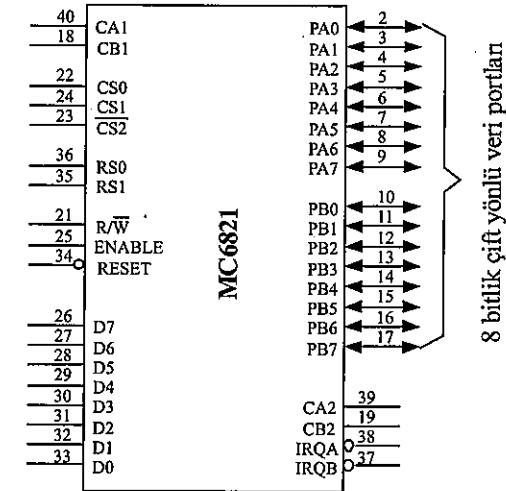
Mikroişlemci üreticileri, ürettikleri mikroişlemciler ile tam uyumlu giriş/çıkış elemanlarını da piyasaya sunmaktadırlar. Aşağıdaki giriş/çıkış arabirimleri için bahsedilen kural ve yöntemler diğer giriş/çıkış arabirimleri için de ortak noktaları içermektedir. Kullanıcılar seçtikleri giriş/çıkış arabirimlerinin detaylı bilgilerini, üreticilerin internet sitelerinden veya ürünün katalog bilgilerinden faydalanarak öğrenebilirler.

12.3.1 MC6821 Paralel Giriş/Çıkış Entegresi

MC6821 entegresi Motorola firmasının ürettiği bir paralel giriş/çıkış entegresidir ve MC6800 ailesinin bir üyesidir. Kullanıcıya 8 bitlik iki adet programlanabilir giriş/çıkış portu ile çalışma imkânı sağlar. MC6821 entegresinin diğer kontrol uçları ve görevleri tablo 12.1'de listelenmiştir.

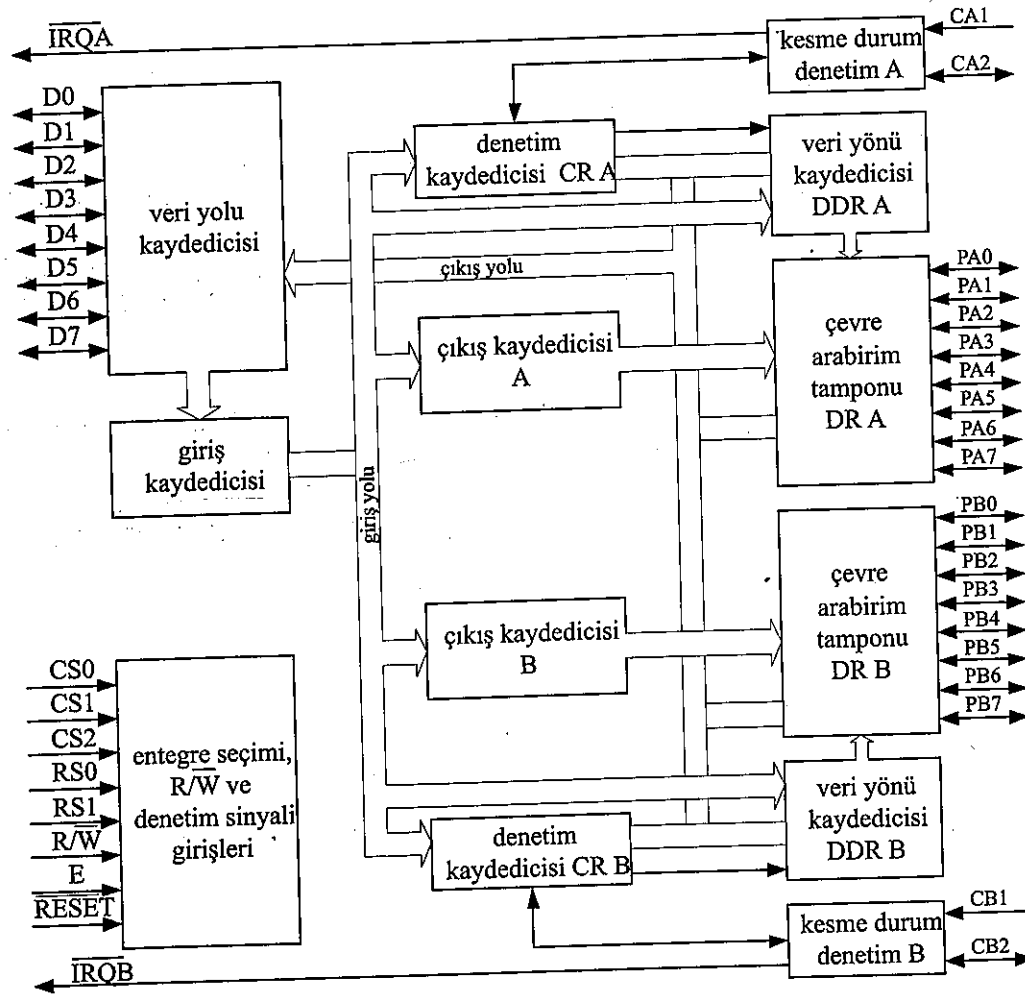
Tablo 12.1 : MC6821'nin bacak bağlantıları ve görevleri

Bacak	Açıklama
CA1 (CB1)	PIA, çevre birimin bu girişe gelen sinyal ile hazır olup olmadığını belirler.
CA2 (CB2)	Çevre birimlerle el sıkışmada kullanılan bu ucun ne şekilde bir işlev göreceğini kontrol kaydedicisindeki D ₃ , D ₄ , D ₅ bitleri belirler.
E (Enable)	PIA'nın içerisindeki birimlerin birbirleri ile ve mikroişlemci ile uyumlu çalışması için gerekli sinyal girişidir. Bu uç MC6800 mikroişlemcilerindeki E çıkışına bağlanır.
CS0, CS1, CS2	PIA'nın adreslenmesinde kullanılan seçme girişleridir. CS0 ve CS1 lojik "1" de etkinken CS2 lojik "0" da etkindir.
R/W	PIA'nın kaydedicilerinin okunması ve yazılması için kullanılan oku/yaz sinyali girişi
Reset	Bu girişin "0" yapılması ile PIA'nın içerisindeki tüm kaydediciler sıfırlanır.
RS0, RS1	Kaydedicilerin seçilmesi için kullanılan seçme girişleri. Bu girişlerin durumlarına göre PIA'nın kaydedicilerine erişim sağlanır.
IRQA, IRQB	Harici çevre birimlerinin durumlarını, mikroişlemciye bildirmek için kullanılan kesme çıkışlarıdır.



Şekil 12.5 : MC6821 entegresinin bacak bağlantıları

MC6821 paralel giriş/çıkış entegresinin içerisinde 6 adet kaydedici vardır. Bu kaydediciler A ve B olarak 3'er kaydedicili 2 grup hâlinde. Her gruptaki kaydediciler ilgili giriş/çıkış portunun programlanmasında ve erişilmesinde kullanılır. Her grupta bulunan bu üç adet kaydedici, veri kaydedicisi (DR; Data Register), veri yönü kaydedicisi (DDR; Data Direction Register) ve kontrol kaydedicisidir (CR; Control Register).



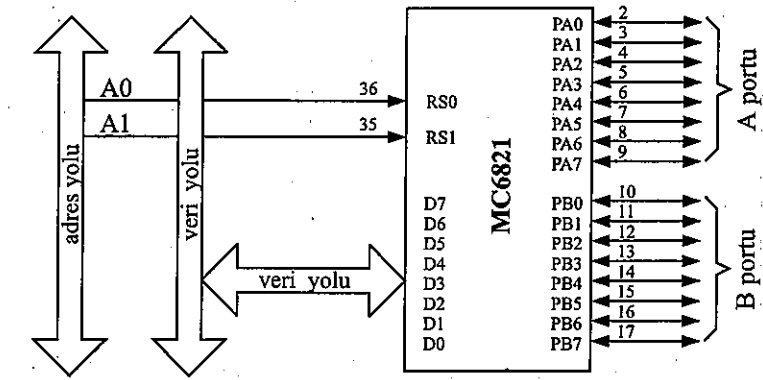
Şekil 12.6 : MC6821 PIA'nın iç yapısı

Veri Kaydedicisi (DR; Data Register): PIA'nın çevresel birimlerle ile bağlantısını yapan kaydedicidir. Bu kaydedicilere yazılan veya okunan veriler çevre birimlerine aktarılır veya çevre birimlerinden okunan verilerdir. Bu kaydedicideki her bit çıkışa direkt olarak gönderilir veya alınır.

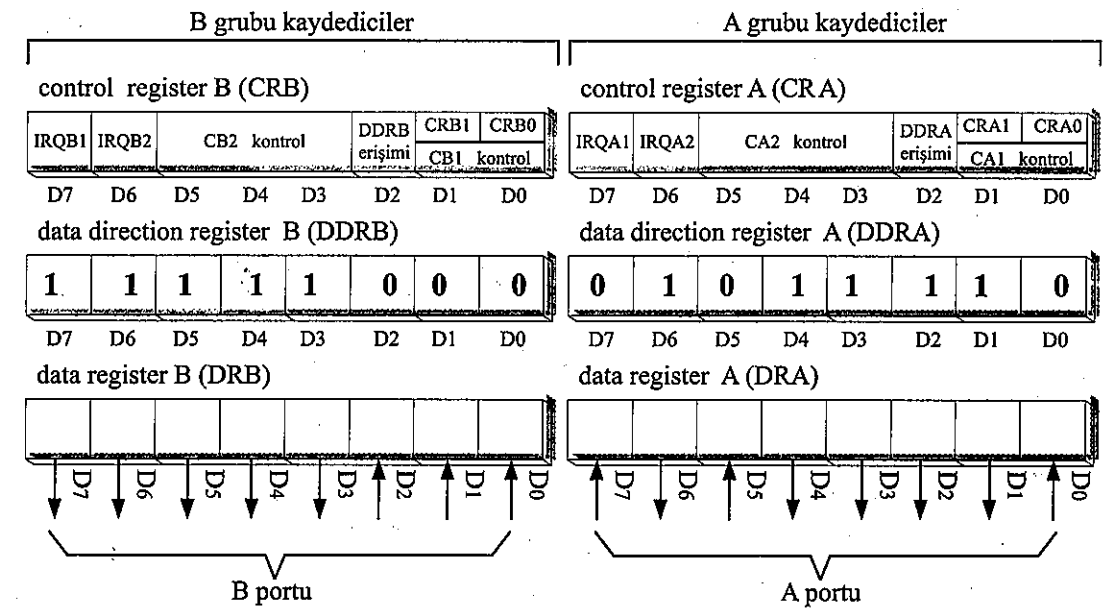
Veri Yönü Kaydedicisi (DDR; Data Direction Register): Veri yönü kaydedicisi isminden de anlaşılacağı gibi A veya B portundaki giriş/çıkış uçlarının veri yönünü tayin etmek için kullanılır. PIA'nın giriş/çıkışındaki her uca, "veri yönü kaydedicisinde" 1 bit karşılık gelir. Bu portlardaki her uç ayrı ayrı giriş veya çıkış olarak programlanabilir. Eğer herhangi bir giriş/çıkış ucunu çıkış olarak programlamak istiyorsak veri yönü kaydedicisindeki ilgili bite "1", giriş olarak programlamak istiyorsak veri yönü kaydedicisindeki ilgili bite "0" yazmamız gerekir. Bu kaydedicideki bitlerin "1" veya "0" oluşuna göre PIA'nın giriş/çıkış uçlarının veri yönü belirlenmiş olur. Şekil 12.8'de veri yönü kaydedicisinde bulunan "1" ve "0" verilerine göre giriş/çıkış uçlarının yön durumu görülmektedir (giriş için: "1", çıkış için "0").

Kontrol Kaydedicisi (CR; Control Register): MC6802 paralel giriş/çıkış entegresinin durumu ve denetiminin yapıldığı kaydedicidir. Bu kaydedici ile hem PIA'nın denetlenmesi yapılır hem de giriş/çıkış durumları öğrenilir. PIA'nın ne şekilde kullanılacağı (standart giriş/çıkış, kesme kontrollü veya el sıkışmalı) bu kaydediciye yüklenecek veriler ile yapılmaktadır.

Entegrede kullanılan tüm giriş/çıkış uçlarının programlanabilmesi ve dolayısıyla kaydedicilerine erişilmesi için RS₀ ve RS₁ giriş uçlarının adres yolundaki A₀ ve A₁ uçlarına bağlanması gerekir. RS₀, A₀ hattına ve RS₁, A₁ hattına bağlanmalıdır. Entegrenin 8 bitlik veri hatları da sistemin veri yoluna bağlanmalıdır.



Şekil 12.7 : MC6821 paralel giriş/çıkış entegresinin sistem yollarına bağlantısı



Şekil 12.8 : PIA'nın A ve B portlarına bağlı kaydedicileri

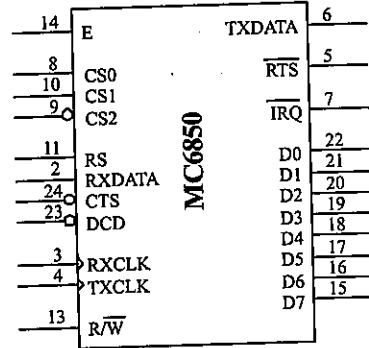
12.4 SERİ GİRİŞ/ÇIKIŞ ARABİRLERİ

12.4.1 MC6850 Seri Giriş/Çıkış Entegresi

Motorola firmasının üretmiş olduğu MC6850 entegresi, MC6802, R6551 ve diğer mikroişlemcilerle uyumlu çalışabilecek bir asenkron seri iletişim arabirimidir. Bu arabirim, sistem veri yoluna bağlı iki yönlü 8 bitlik veri hatlarıyla mikroişlemci arasında veri alışverişinde bulunur. MC6850 entegresi "alıcı grubu" ve "verici grubu" olarak iki alt birimden oluşmuştur. MC6850'nin diğer bacak bağlantıları şöyledir;

Tablo 12.2 : MC6850'nin bacak bağlantıları ve görevleri

Bacak	Açıklama
E (Enable)	ACIA'nın içerisindeki bir imlerin birbirleri ile ve mikroişlemci ile uyumlu çalışması için gerekli sinyal girişidir. MC68xx mikroşlemcili mikrobilgisayarlarda sistem saati olarak kullanılan E hattı ACIA'nın E çıkışına bağlanır.
CS0, CS1, CS2	ACIA'nın adreslenm esinde kullanılan seçme girişleridir. CS0 ve CS1 lojik "1" de etkinken CS2 lojik "0" da etkindir.
RS	Durum ve kontrol kaydedicilerin seçilmesi için kullanılan seçme girişleri. RS girişinin R/W ile birlikte kullanıldığında hangi kayd edicilere erişim sağlandığı tablo 12.3'de gösterilmiştir.
RXDATA, TXDATA	Seri iletişim yapıldığı giriş/çıkış uçları. RXDATA veri girişinin, TXDATA veri çıkışının yapıldığı uçlardır.
CTS, DCD, RTS	Çevre birimlerle el sıkışmada kullanılması için üç uç bulu nmaktadır. Bu uçlar; CTS (Clear To Send; Göndermek için sil), DCD (Data Carrier Detect; Taşıyıcı görüldü), RTS (Request To Send; Göndermek için sil).
RXCLK, TXCLK	Alıcı ve verici iletişim hızlarına uygun standart sinyal uçlarıdır. TXCLK vericinin iletişim hızını belirleyen saat girişidir. RXCLK alıcının iletişim hızını belirleyen saat girişidir. MC6850 farklı iletişim hızlarında alma ve gönderme yapabilir. Alıcı ve verici hızları kontrol kaydedicisinin D ₀ ve D ₁ bitlerine yazılan değerlere göre değiştirilebilmektedir.
R/W	ACIA'nın kaydedicilerinin okunması ve yazılması için kullanılan oku/yaz sinyali girişi. Aynı zamanda bu uç kaydedicilerin seçiminde de rol alır.
IRQ	Kesme oluşturan bir durumda mikroşlemciye kesm e sinyali göndererek durumu bildirmek için kullanılan çıkıştır. Bu çıkış mikroşlemcinin kesme girişine uygulanır. ACIA'nın hangi durumlarda kesme üreteceği kaydedicilerin programlanma durumuna göre belirlenir.

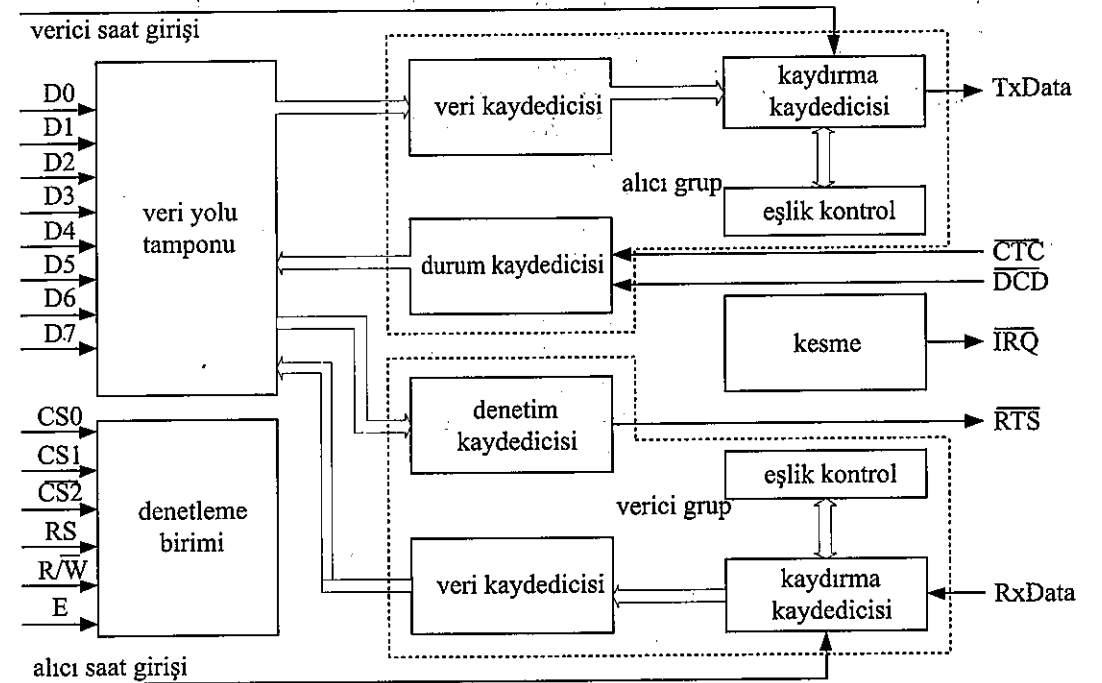


Şekil 12.9 : MC6850 entegresinin bacak bağlantıları

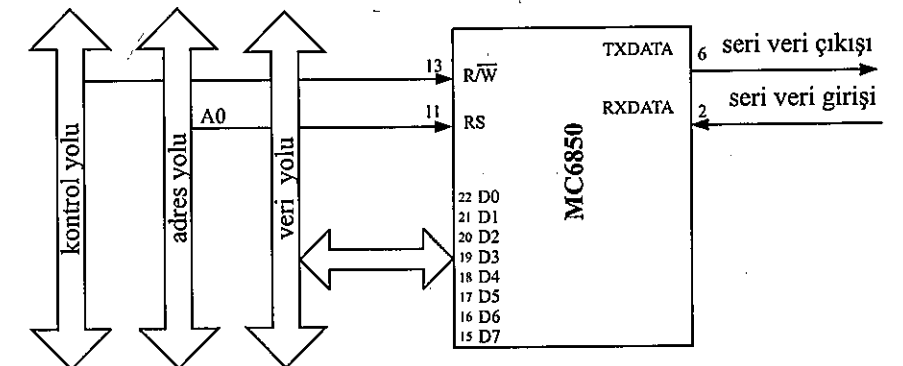
MC6850 seri arabirim entegresinde 4 adet kaydedici bulunmaktadır. Bunlar “durum kaydedicisi”, “kontrol kaydedicisi”, “gönderici veri kaydedicisi”, “alıcı veri kaydedicisi” dir. Entegrede bu kaydedicilerin seçiminde RS girişi ve R/W hattı beraber kullanılır. RS girişi genellikle adres yolundaki A₀ hattına bağlanır. Bu kaydedicilere erişmek için RS ve R/W uçlarına uygulanması gereken sayısal değerler tablo 12.3'de verilmiştir.

Tablo 12.3 : ACIA'nın kaydedicilerine erişmek için RS ve R/W uçlarına uygulanacak sayısal değerler

RS	R/W	Seçilen kaydedici
0	0	Kontrol kaydedicisi
0	1	Durum kaydedicisi
1	0	Gönderici veri kaydedicisi
1	1	Alıcı veri kaydedicisi

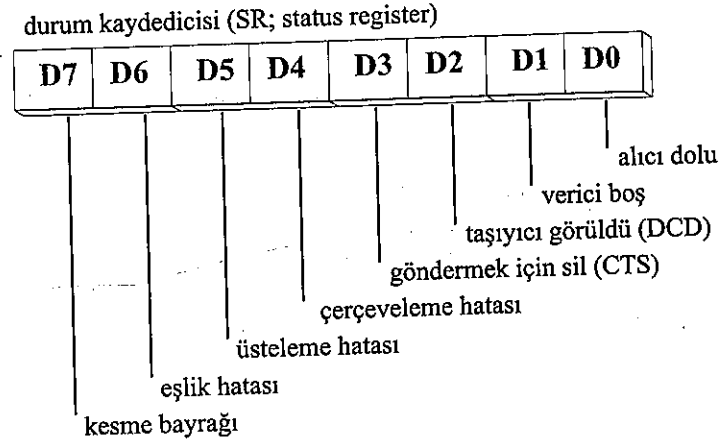


Şekil 12.10 : MC6850 ACIA'nın iç yapısı



Şekil 12.11 : MC6850 paralel giriş/çıkış entegresinin sistem yollarına bağlantısı

Durum Kaydedicisi (SR; Status Register): ACIA'nın alıcı ve vericiye ilişkin durumlarını içerir. Durum kütüğü 8 bitlik bir kaydedicidir. Bu kaydedici yalnız okunabilir bir kaydedicidir. Bu kaydedicideki bitlerin ifade ettiği anlamlar şekil 12.12'de gösterilmiştir.



Şekil 12.12 : MC6850 ACIA'nın durum kaydedicisindeki bitlerin anlamları

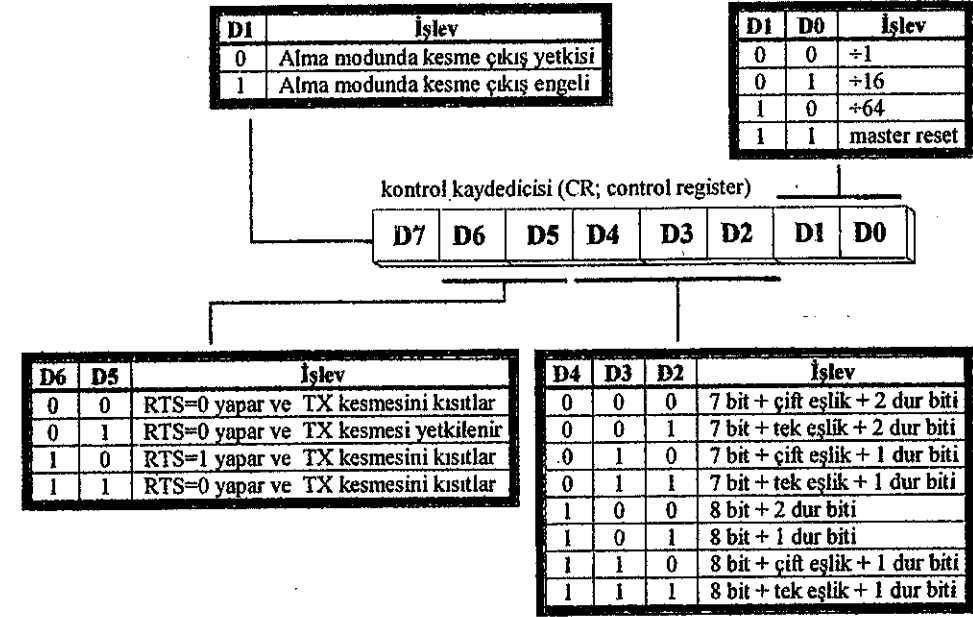
Kontrol Kaydedicisi (CR; Control Register): Alıcı ve vericiyi denetlemek üzere kullanılır. Bu kaydedicideki D₆ ve D₁ bitleri iletişim hızını belirlemeye yarar. Şekil 12.13'de bu bitlerin durumuna göre iletişim hızının alacağı değerler gösterilmektedir. D₄, D₃, D₂ bitleri iletişimde kullanılacak verinin uzunluğunun, eşlik bitinin ve dur bitinin şeklini belirlemede kullanılır. D₆, D₅ bitleri vericinin kesme işlemlerini denetler. D₇ biti kesme kontrol bitidir. Bu bit "1" yapılır ve aşağıdaki durumlar oluşursa bu bir kesme sinyali üretilmesine neden olur. Bunlar:

- 1-Alıcı dolu olduğunda veri akışı devam ederse,
- 2-Alıcıdan veri okunmadan gelen yeni veri çığneme oluştuğunda,
- 3- DCD girişinin 0'dan 1'e geçmesidir.

Gönderici ve Alıcı veri kaydedicileri (TxDR, RxDR; Transmitter, Receiver Data Register): Seri kanaldan aktarılabilecek veya alınacak veriler bu kaydediciler üzerinden yapılır. TxDR kaydedicisi yalnızca yazılabilir, RxDR kaydedicisi ise yalnızca okunabilir özellikte kaydedicilerdir.

MC6850 ACIA'nın kaydedicilerinin;

- 1-Durum ve kontrol kaydedicileri RS ve R/W girişleri ile erişilebilirken okuma/yazma çevrimleri sırasında bu kaydedicilerdeki değerler artırma, azaltma, döndürme gibi komutlarla kullanılmaz.
- 2-Kontrol kaydedicisi mikroişlemci tarafından okunamadığından bu kaydediciye yazılan değer bir kopyası hafızaya yüklenmelidir.
- 3-ACIA'nın bacaklarında ayrı bir reset girişi yoktur. Kontrol kaydedicisindeki D₆ ve D₁ bitleri "1" yapılarak ACIA resetlenmiş olur.
- 4- Durum kaydedicisinin D₀ biti "veri hazır", D₁ biti "çevre birimi hazır" bayraklarıdır.



Şekil 12.13 : MC6850 ACIA'nın kontrol kaydedicisindeki bitlerin anlamları

12.5 GİRİŞ/ÇIKIŞ ARABİRİMLERİ İLE İLETİŞİM YÖNTEMLERİ

12.5.1 Giriş/Çıkış Arabirimlerinin Adreslenmesi

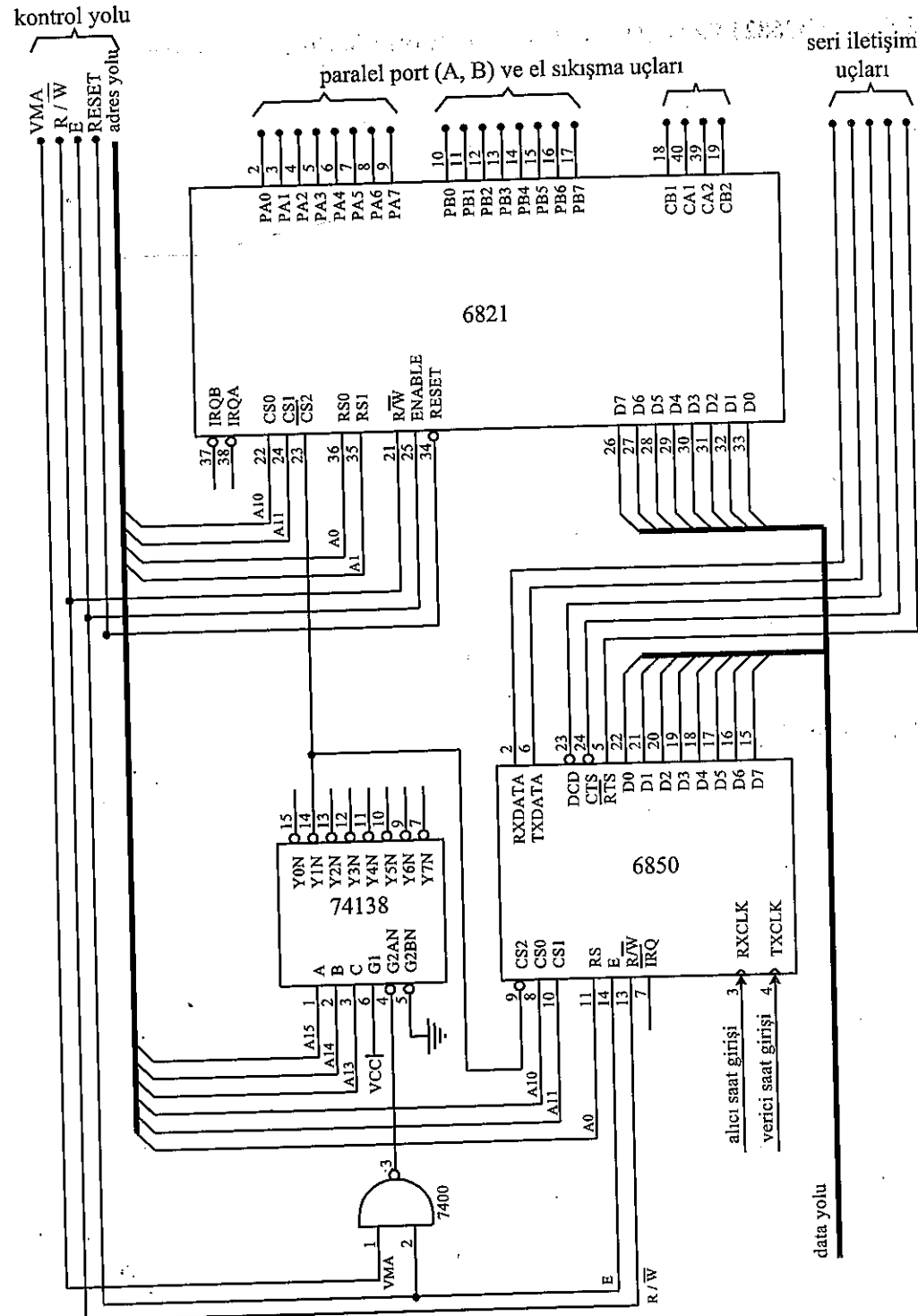
Burada bir mikrobilgisayarın seri ve paralel giriş/çıkış iletişimini sağlamak için kullanılan MC6821 ve MC6850 entegrelerinin, sistemin hafıza haritasında önceden belirlenmiş adreslere göre kod çözümünün ve diğer bağlantılarının nasıl yapıldığını inceleyeceğiz. MC68xx mikroişlemci ailesinde giriş/çıkış adresleri hafıza haritasında yer almaktadır. Yani bu mikroişlemci hafıza haritalı giriş/çıkış tekniği kullanılmaktadır.

Daha önce de belirtildiği gibi PIA'nın (MC6821) sistem hafıza haritasında 4 adrese, ACIA'nın (MC6850) ise 2 adrese ihtiyacı vardır. Bu adreslerin sistem haritasında PIA için (8000)₁₆...(8003)₁₆ ve ACIA içinde (8C00)₁₆...(8C01)₁₆ adreslerine göre yapılması için nasıl bir yol izlendiği aşağıda anlatılmıştır. Tablo 12.4'de PIA'nın ve ACIA'nın bu adreslerle hangi kaydedicilerine erişim sağlandığı gösterilmiştir.

Tablo 12.4 : Giriş/çıkış birimlerindeki belirtilen aralıkta adreslenen kaydediciler

Adres	Birim	Erişilen kaydedici
(8000) ₁₆	MC6821 PIA	DRA Veri kaydedicisi (Kontrol kaydedicisindeki D2= "1" ise)
		DDRA Veri yönü kaydedicisi (Kontrol kaydedicisi (CRA) D2= "0" ise)
(8001) ₁₆		CRA Kontrol kaydedicisi
(8002) ₁₆		DRB Veri kaydedicisi (Kontrol kaydedicisi (CRB) D2= "1" ise)
	MC6850	DDRB Veri yönü kaydedicisi (Kontrol kaydedicisi (CRB) D2= "0" ise)
(8003) ₁₆		CRB Kontrol kaydedicisi
(8C00) ₁₆		CR Kontrol kaydedicisi ((8C00) ₁₆ adresine yazıldığında)
		SR Durum kaydedicisi ((8C00) ₁₆ adresine okunduğunda)
(8C01) ₁₆		TxDR Çıkış kaydedicisi ((8C01) ₁₆ adresine yazıldığında)
		RxDR Giriş kaydedicisi ((8C01) ₁₆ adresine okunduğunda)

MC6800 mikroişlemci ailesinde geçerli adres yolundaki adres bilgisinin geçerli adres olması VMA ve E çıkışının aynı anda "1" olması ile gerçekleşmektedir. Bu durum dikkate alındığında, 74138 entegresinin G2AN seçme ucu VMA ve E sinyalleri kullanılarak AND kapısı ile sürülmüştür. Böylece adres yolu üzerindeki değerlerin gerçekten bir adres olması ve seçimin, E sinyalinin "1" olduğu durumda yapılması sağlanmış olacaktır. Bu iki koşul sağlandığında 74138'in PIA ve ACIA adreslemesi doğru yapılır.

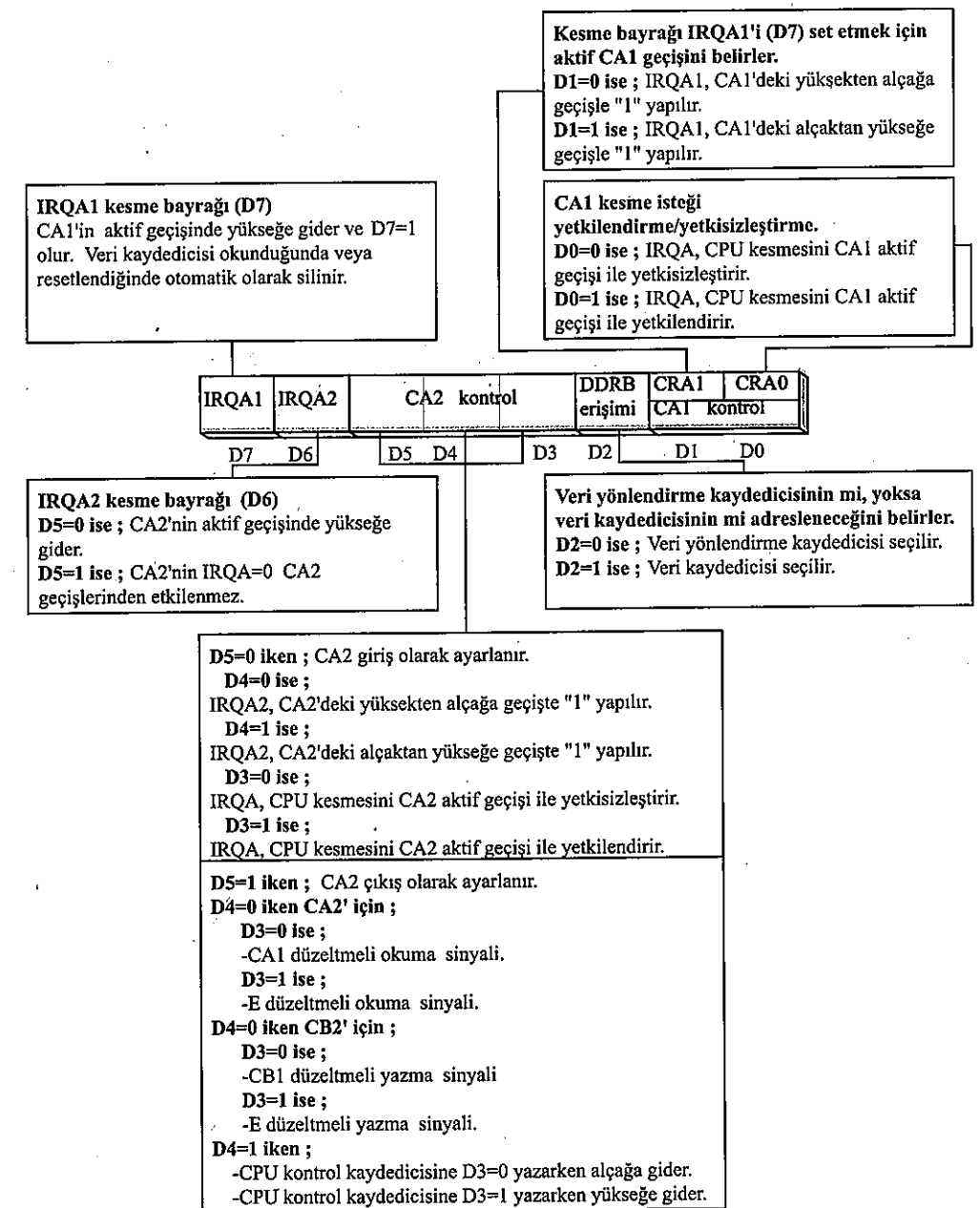


Şekil 12.14 : PIA ve ACIA'nın mikrobilgisayarda belirlenen adreslere göre bağlantısı

Adres yolunun A_{11}, A_{10}, A_9, A_8 , hatları ile $(X0XX)_{16} \dots (XCXX)_{16}$ arası adreslerin, A_{11}, A_{10}, A_9 hatları ile $(8XXX)_{16}$ adreslerinin kod çözümü yapılmaktadır. Burada kullanılan hatlar bir arada düşünüldüğünde istediğimiz adres aralığının seçilmesi sağlanmış olacaktır. Adres yolunun A_1, A_0 , hatları ACIA ve PIA'daki kaydedicilerin adreslenmesinde görev yapmaktadırlar. Burada kullanılan 74138 entegresinin diğer çıkışları sistemdeki RAM, EPROM gibi diğer elemanların adreslenmesinde de kullanılabilir.

12.5.2 MC6821 Paralel Giriş/Çıkış Entegresinin Programlanması

MC6821 paralel giriş/çıkış entegresi programlanarak iki şekilde çalışır. Birincisi standart giriş/çıkış için kullanımı diğeri ise el sıkışma yöntemi ile kullanımıdır. PIA'nın el sıkışma yöntemi ile çalışması için bağlandığı diğer çevre birimlerden (yazıcı gibi) gelen durum/denetim sinyallerinin kullanılması ile mümkündür. Buna **"onaylı giriş/çıkış tekniği"** de denir.



Şekil 12.15 : MC6821 PIA kontrol kaydedicisindeki bitlerin görevleri

PIA'nın onaylı giriş/çıkış tekniğine göre çalışması için kontrol kaydedicilerine (CRA ve CRB) uygun değerler yükleyerek gerçekleştirilebilir. Şekil 12.15'de PIA'nın onaylı giriş/çıkış tekniğine göre kullanımı için kontrol kaydedicisindeki bitlerin ne şekilde yükleneceği gösterilmiştir.

PIA'nın diğer çalışma şekli, yalnızca temel giriş/çıkış işlemleri için programlanması ile mümkündür. Her portun bitleri birbirinden bağımsız olarak giriş veya çıkış olarak programlanabilmektedir. A veya B portundaki bu uçların çift yönlü olması aynı anda hem giriş hem de çıkış olarak kullanılacağı anlamına gelmez. Her uç programlandıktan sonra ya giriş ya da çıkış olarak tek yönlü kullanılabilir. PIA'yı bu şekilde programlayarak kullanabilmek için üç temel adım uygulanmalıdır.

- 1-Kontrol kaydedicisindeki D₂ biti "0" yapılarak veri yönü kaydedicisi erişime açılır.
- 2-Veri yönü kaydedicisine çıkış hatlarının iletişim yönlerini belirten bit düzeni yüklenir.
- 3-Kontrol kaydedicisindeki D₂ biti "1" yapılarak "veri kaydedicisi" iletişime açılır.

PIA'nın bağlı olduğu adreslerin bilinmesi, kaydedicilere erişerek PIA'nın istenilen çalışma şekline göre programlanabilmesi ve giriş/çıkış portlarına veri yazma/okuma işlerinde kullanabilmek için gereklidir. Örneğin PIA sistemin hafıza haritasında (8000)₁₆... (8003)₁₆ adresleri arasında olsun. Dikkat edilirse verilen adres aralığı ile ancak 4 farklı hafıza hücresine erişmek mümkündür. Fakat MC6821 entegresinde 6 adet kaydediciye erişebilmek için dört farklı adres kullanmak yeterlidir. (8000)₁₆ adresi ile A grubu "veri yönü kaydedicisi" ve "veri kaydedicisine", (8002)₁₆ adresi ile B grubu "veri yönü kaydedicisi" ve "veri kaydedicilerine" aynı adreslerden erişmek mümkündür. Bu "kontrol kaydedicilerindeki" D2 biti kullanılarak yapılmaktadır. Bu şekilde dört farklı adres ile toplam 6 kaydediciye erişebilmek mümkündür.

Tablo 12.5 : PIA'nın kaydedicilerine erişme yöntemi

Adres	(A ₁) RS1	(A ₀) RS0	Erişilen kaydedici
(8000) ₁₆	0	0	(DRA) Veri Kaydedicisi (Kontrol kaydedicisindeki D2= 1 ise) (DDRA) Veri Yönü Kaydedicisi (Kontrol Kaydedicisi (CRA) D2= 0 ise)
(8001) ₁₆	0	1	(CRA) Kontrol Kaydedicisi
(8002) ₁₆	1	0	(DRB) Veri Kaydedicisi (Kontrol Kaydedicisi (CRB) D2= 1 ise) (DDRB) Veri Yönü Kaydedicisi (Kontrol Kaydedicisi (CRB) D2= 0 ise)
(8003) ₁₆	1	1	(CRB) Kontrol Kaydedicisi

Tablo 12.5'deki gibi A grubundaki kaydediciler (8000)₁₆ ve (8001)₁₆, B grubundaki kaydediciler ise (8002)₁₆ ve (8003)₁₆ adreslerinde bulunmaktadır. PIA programlanırken A veya B grubundaki kaydediciler arasında hiçbir fark yoktur, yalnızca erişim adresleri farklıdır. Örneğin PIA'nın A ve B portlarındaki giriş/çıkış uçlarını tablo 12.6'daki gibi programlamak istiyorsak yapılacak işlemler adım adım şu şekilde olacaktır;

Tablo 12.6 : Örnek giriş/çıkış uçları

B grubu giriş/çıkış								A grubu giriş/çıkış							
PB ₇	PB ₆	PB ₅	PB ₄	PB ₃	PB ₂	PB ₁	PB ₀	PA ₇	PA ₆	PA ₅	PA ₄	PA ₃	PA ₂	PA ₁	PA ₀
Çıkış	Çıkış	Çıkış	Çıkış	Çıkış	Giriş	Giriş	Giriş	Giriş	Çıkış	Giriş	Çıkış	Çıkış	Çıkış	Çıkış	Giriş
1	1	1	1	1	0	0	0	0	1	0	1	1	1	1	0
F				8				5				E			

1. Adım: A grubundaki "veri yönü kaydedicisine" erişebilmek için (8001)₁₆ adresindeki "kontrol kaydedicisinin" D2 bitini "0" yapacak değeri yüklememiz gerekmektedir. A grubundaki "kontrol kaydedicisindeki" D2 biti "0" yapılarak (8000)₁₆ adresi ile "veri yönü kaydedicisine" erişim sağlanmış olur.
2. Adım: A portundaki giriş/çıkış uçlarının Tablo 12.6'daki gibi olması için (8000)₁₆ adresindeki "veri yönü kaydedicisine" yazmamız gereken bilgi (01011110)₂=(5E)₁₆ olmalıdır. Buradaki "0" lar "giriş", "1" ler "çıkış" a karşılık gelmektedir.
3. Adım: A portundaki giriş/çıkış uçlarının yönü belirlendikten sonra "A portunun" iletişime açmamız gerekecektir. Bunun için (8001)₁₆ adresindeki kontrol kaydedicisinin D2 biti "1" yapılması yeterlidir. Böylelikle (8000)₁₆ adresi ile "veri kaydedicisini" iletişime açmış oluruz.

Artık PIA'nın A portu ile çevre birimleri arasındaki bağlantı (8000)₁₆ adresindeki "veri kaydedicisi" üzerinden gerçekleştirilmeye hazırdır. Tablo 12.6'daki verilere göre bu işlemlerin aynısını B portu için de yapmamız gerekmektedir. Bununla ilgili hazırlanan assembly programını 13. bölümün "giriş/çıkış ilişkili program yazımı" konusunda hazırlanmıştır.

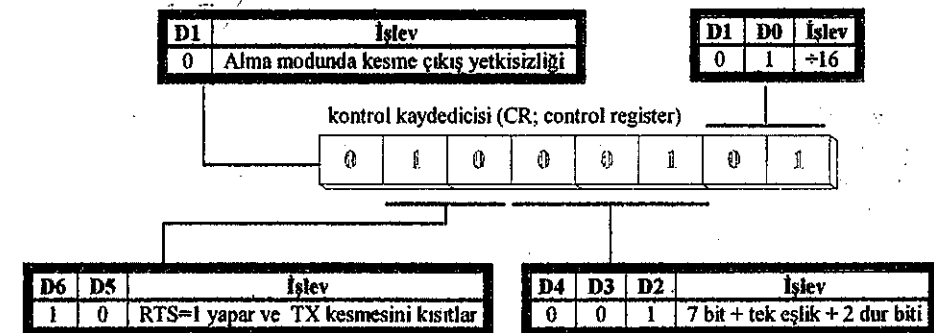
12.5.3 MC6850 Seri Giriş/Çıkış Entegresinin Programlanması

MC6850 seri giriş/çıkış entegresi üzerinden çevre birimleri ile iletişimin sağlanması için sistemin hafıza haritasında yer alan ACIA kaydedicilerinin programlanması gerekmektedir. ACIA'da bilindiği gibi 4 adet kaydedici bulunmaktadır. Bu kaydedicilerden kontrol kaydedicisi programlanarak ACIA'nın istendiği şekilde çalışması sağlanır. MC6850'i harici saat darbeleri kullanır. Eğer ACIA'ya 38400 Hz. frekansında saat darbeleri uygulanırsa ve kontrol kaydedicisindeki osilatör bölme oranı 16 seçilirse 2400 baud'luk bir sinyal akışı sağlanmış olacaktır. ACIA kaydedicileri tablo 12.7'deki gibi hafıza haritasında (8C00)₁₆ ve (8C01)₁₆ adreslerinde bulunacak şekilde gerekli bağlantıları sağlanmış olsun.

Tablo 12.7 : ACIA'nın kaydedicilerine erişme yöntemi

Adres	RS	R/W	Erişilen kaydedici
(8C00) ₁₆	0	0	CR Kontrol kaydedicisi
(8C00) ₁₆	0	1	SR Durum kaydedicisi
(8C01) ₁₆	1	0	TxDR Gönderici veri kaydedicisi
(8C01) ₁₆	1	1	RxDR Alıcı veri kaydedicisi

MC6850 entegresi ile 2400 baud'luk, 7 veri biti, tek eşlik ve iki dur biti ile iletişim sağlayabilmek için kontrol kaydedicisine (45)₁₆ verisini yazmamız gerekmektedir. (45)₁₆ verisinin kontrol kaydedicisindeki bit düzenine göre entegrenin çalışma fonksiyonunu nasıl belirlediği şekil 12.16'da görülmektedir.



Şekil 12.16 : Kontrol kaydedicisinin işlevleri

Aşağıda (tablo 12.8 ve tablo 12.9) seri kanaldan (ACIA'dan) veri okuyup (0100)₁₆ adresine yazan ve (0100)₁₆ adresindeki veriyi seri olarak ileten alt programlar MC6802 mikroişlemci komut setindeki komutlar kullanılarak hazırlanmıştır.

Amaç; ACIA'dan bir byte veri okuyarak bunu hafızanın (0100)₁₆ adresine yazmak.

Açıklama: ACIA'nın kesme girişi bulunmadığı için önce tüm kaydedicilerin sıfırlanması için kontrol kaydedicisinin D₁ ve D₀ bitlerine "11" verileri yazılır. Daha sonra ACIA'nın kontrol kaydedicisi "7 veri biti, tek eşlik, 2 dur biti" şartlarını sağlaması için (45)₁₆ verisi ile yüklenir. Bundan sonra ACIA'nın RxDR kaydedicisi bir döngüde okutulur. Bu döngüden ancak gelen veri RxDR kaydedicisini doldurup durum kaydedicisindeki D₀ biti "1" olduğunda çıkılır. Gelen seri veri RxDR kaydedicisine yüklendiğinde döngüden çıkılarak okunan veri hafızanın (0100)₁₆ adresine yazılarak ana programa RTS komutu ile dönülür.

Tablo 12.8 : ACIA'dan seri veriyi okuyup (0100)₁₆ adresine yazan alt program

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	86 03		LDAA #\$03	A'ya (03) ₁₆ verisini yükle.
0002	B7 8C 00		STAA \$8C00	Kontrol kaydedicisine A'daki değeri yükleyerek ACIA'yı resetle.
0005	86 45		LDAA #\$45	A'ya (45) ₁₆ verisini yükle.
0007	B7 8C 00		STAA \$8C00	ACIA'yı "7 veri biti, tek eşlik, 2 dur biti" için kur.
000A	B6 8C 00		LDAA \$8C00	Durum kaydedicisini oku.
000D	44	Wait:	LSRA	Durum kaydedicisini sağa kaydır.
000E	24 FD		BCC Wait	Eğer C=0 ise Wait etiketine dallan.
0010	B6 8C 01		LDAA \$8C01	Veri alma kaydedicisindeki veriyi A'ya yükle.
0013	B7 01 00		STAA \$0100	A'daki veriyi (0100) ₁₆ adresine yükle.
0016	39		RTS	Ana programa dön.

Amaç; hafızanın (0100)₁₆ adresindeki veriyi ACIA üzerinden seri olarak çevre birimlerine aktarmak.

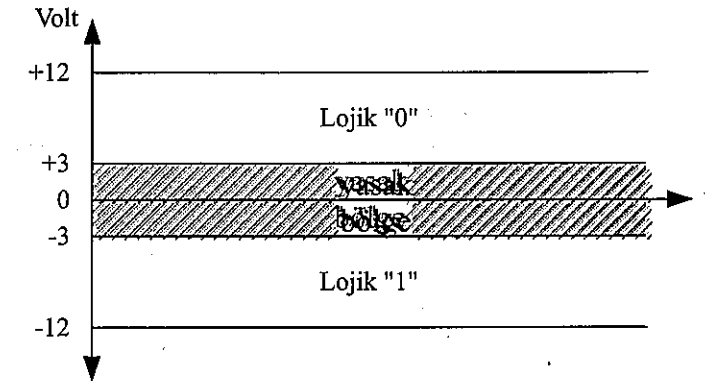
Açıklama: ACIA'nın kesme girişi bulunmadığı için önce tüm kaydedicilerin sıfırlanması için kontrol kaydedicisinin D₁ ve D₀ bitlerine "11" verileri yazılır. Daha sonra ACIA'nın kontrol kaydedicisi "7 veri biti, tek eşlik, 2 dur biti" şartlarını sağlaması için (45)₁₆ verisi ile yüklenir. Bundan sonra ACIA'nın durum kaydedicisindeki D₁ biti okunarak "0" olup olmadığı kontrol edilir. Eğer veri gönderici kaydedici boş ise D₁ biti "1" olacaktır. D₁=1 olduğunda yani veri gönderici kaydedicisi boş ise (0100)₁₆ adresindeki veri okunarak ACIA'nın TxDR kaydedicisine yazılır ve RTS komutu ile ana programa dönülür.

Tablo 12.9 : Hafızanın (0100)₁₆ adresindeki veriyi ACIA üzerinden seri olarak ileten alt program

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	86 03		LDAA #\$03	A'ya (03) ₁₆ verisini yükle.
0002	B7 8C 00		STAA \$8C00	Kontrol kaydedicisine A'daki değeri yükleyerek ACIA'yı resetle.
0005	86 45		LDAA #\$45	A'ya (45) ₁₆ verisini yükle.
0007	B7 8C 00		STAA \$8C00	ACIA'yı "7 veri biti, tek eşlik, 2 dur biti" için kur.
000A	B6 8C 00		LDAA \$8C00	Durum kaydedicisini oku.
000D	44	Wait:	LSRA	Durum kaydedicisi bir bit sağa kaydır.
000E	44		LSRA	Durum kaydedicisi bir bit sağa kaydır.
000F	24 FC		BCC Wait	Eğer C=0 ise Wait etiketine dallan.
0011	B6 01 00		LDAA \$0100	Veri alma kaydedicisindeki veriyi A'ya yükle.
0014	B7 8C 01		STAA \$8C01	A'daki veriyi (0100) ₁₆ adresine yükle.
0017	39		RTS	Ana programa dön.

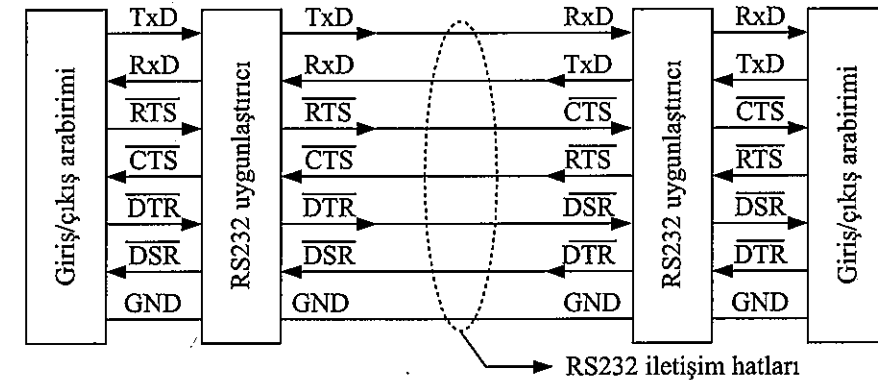
12.6 RS232 STANDARDI

TTL lojik seviyeleri ile RS232 standardının lojik seviyeleri farklıdır. RS232 standardında +3 V ve yukarısı lojik "0", -3 V ve aşağısı lojik "1" kabul edilmektedir. +3 V ve -3 V aralığındaki sinyal seviyeleri sayısal bilginin yanlış dönüştürülmesine sebep olacağından bu gerilim seviyelerine yakın veri iletiminden kaçınılmalıdır. Lojik seviyelerin yüksek gerilim seviyelerine çekilmesinin sebebi, uzun mesafeli iletişimlerde kullanılan hatlarda oluşabilecek elektriksel kayıpları telâfi etmek içindir. Normal sayısal bilgileri RS232 standardına uygun gerilim seviyelerine dönüştürmek için Max 232, MC1489 ve MC1488 gibi entegreler kullanılmaktadır. Bunlar TTL lojik seviyelerini RS232 standardına çeviren entegrelerden birkaçıdır.



Şekil 12.17 Standart RS232 sinyal seviyeleri

Seri iletimde 20000 baudluk bir hızdaki veri iletimi 15 metre, daha düşük hızlardaki veri iletimi 610-915 metre arasında yapılırsa daha güvenli (hata oranı çok düşük) olarak iletilir.

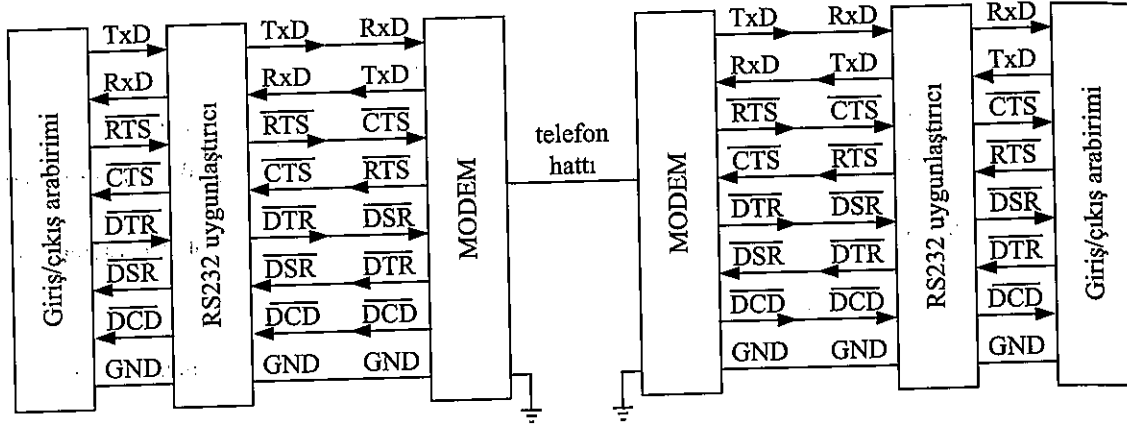


Şekil 12.18 : Temel seri giriş/çıkış entegresi ile RS232 dönüştürücülerinin bağlantısı

Uzak mesafelerdeki iki bilgisayarın telefon hatlarını kullanarak haberleşebilmesi için modemler kullanılır. Modemler standart RS232 bağlantı noktaları ile haberleşmektedirler. Şekil 12.18'den farklı olarak RS232 ile modem arasında bulunan (data carrier detect) bağlantısı, telefon hattında taşıyıcı sinyalin varlığını sisteme bildiren bir kontrol hattıdır. RS232 ve modem arasındaki bağlantı şekil 12.19'da verilmiştir.

DEĞERLENDİRME ÇALIŞMALARI

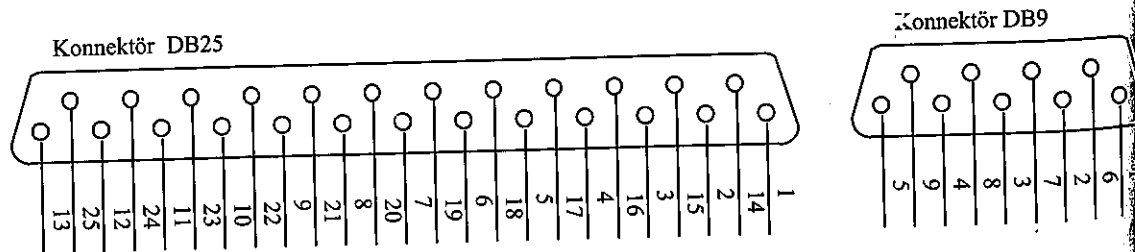
- 1) Mikrobilgisayarların çevre birimleri ile iletişimi kaç çeşit arabirimle gerçekleştirilir?
- 2) Mikrobilgisayarların çevre birimleri ile bağlantısını sağlayan giriş/çıkış arabirimlerinin adreslenmesinde kaç çeşit yöntem vardır?
- 3) Basit bir mikrobilgisayar devresinde giriş/çıkış arabirimlerin bağlantısını blok diyagram olarak çiziniz.
- 4) MC6821 paralel giriş/çıkış entegresinin iç yapısını blok diyagram olarak çiziniz.
- 5) MC6821 paralel giriş/çıkış entegresinde bulunan kaydedicilerin görevlerini yazınız.
- 6) MC6850 seri giriş/çıkış entegresinin iç yapısını blok diyagram olarak çiziniz.
- 7) MC6850 seri giriş/çıkış entegresinde bulunan kaydedicilerin görevlerini yazınız.
- 8) MC6850 entegresi ile "7 veri biti, tek eşlik, 2 dur biti" şeklinde bir seri iletişim protokolü hazırlamak için gerekli program kodlarını MC6802 komutları ile hazırlayınız.
- 9) Hafıza haritasının $(8000)_{16} \dots (8003)_{16}$ adresleri arasında bulunan MC6821 entegresinin A portunu çıkış, B portunu giriş olacak şekilde MC6802 mikroişlemcisi komutlarını kullanarak programlayınız.
- 10) RS232 standartına göre lojik "1" ve lojik "0" değerleri hangi gerilim aralığında gösterilir?



Şekil 12.19 : RS232 ile modem bağlantısı

Tablo 12.10 : RS232 bağlantı noktalarının isimleri ve DB25 ve DB9 bacak numaraları

RS232 konnektör bağlantı noktaları			Açıklama	
Ad	DB9	DB25		
TxD	3	2	Veri gönderici çıkış	(Transmit data)
RxD	2	3	Veri alan giriş	(Receive data)
RTS	7	4	Göndermek için iste	(Request to send)
CTS	8	5	Göndermek için sil	(Clear to send)
DTR	4	20	Veri terminali hazır	(Data terminal ready)
DSR	6	6	Veri kümesi hazır	(Data set ready)
DCD	1	8	Veri taşıyıcısı görüldü	(Data carrier detect)
GND	5	1	Şase	(Ground)



Şekil 12.20 : RS232 standartına göre DB25 ve DB9 konnektörlerinin bacak bağlantıları



Assembly Programlamada Sayı Tabanlarını Gösteren Semboller:

Assembly komutlarından sonra bulunan değerlere "işlenen" denir. İşlenenler farklı sayı tabanlarında olabilmektedir. Ayrıca işlenen, bir "adres" olabileceği gibi doğrudan kullanılacak bir "veri" de olabilir. İşte tüm bu farklılıkları ayırt etmek için assembly komutları ile birlikte kullanılan işlenenlerin ne tipte bir veri olduğunu göstermesi için birtakım semboller kullanılır.

Sembol	İşlenenin Sayı Tabanı	Örnek
D	Onluk	27 veya 27D
% veya B	İkilik	%10011101 veya 10011101B
@ veya O	Oktal	@56 veya 56O
\$ veya H	Hexadecimal	\$3F veya 3FH

NOT: Eğer işlenen doğrudan kullanılan bir veri ise bu sayı tabanı sembollerinin önüne bir de "#" diyerek işaret konur. Örneğin \$F0 veya \$3F27 adresi gösterirken #F0 veya #3F27 doğrudan kullanılacak verileri temsil eder.

13.1.2 Komutların Hafıza Gereksinimleri ve İşlem Süreleri

Assembly programlama dilinde kullanılan komutlar bir veya iki byte olabilmektedir. Çoğu komut ise bir işlenen ile birlikte kullanılmaktadır. Bu komutun görevine ve kullanım amacına bağlı olarak değişebilmektedir. Eğer komutla birlikte bir işlenen kullanılıyorsa işlenen de ayrıca bir veya iki byte olabilmektedir. Tüm bunlar bir araya getirildiğinde herhangi bir komut hafızada toplam 1, 2, 3 veya 4 byte'lık yer kaplayabilmektedir. Tablo 13.1'de büyüklükleri değişik olan komutlardan örnekler verilmiştir.

Tablo 13.1 : Komutların hafıza gereksinimleri

İş kodu				Assembly komutu	Açıklama
1. byte	2. byte	3. byte	4. byte		
49				ROLA	(MC6802) A kaydedicisindeki veri bir bit sola döndürülür. Toplam bir byte'lık bir komuttur. İşleneni yoktur.
97	10			STAA \$10	(MC6802) A kaydedicisindeki veriyi (0010) ₁₆ adresine yükle. Toplam iki byte'lık bir komuttur. İlk byte komutun kendisi, ikinci byte bir adrestir.
86	F0			LDAA #F0	(MC6802) A kaydedicisine (F0) ₁₆ verisini yükle. Toplam iki byte'lık bir komuttur. İlk byte komutun kendisi, ikinci byte doğrudan kullanılacak veridir.
7F	1F	00		CLR \$1F00	(MC6802) (1F00) ₁₆ adresini temizle. Toplam üç byte'lık bir komuttur. İlk byte komutun kendisi, ikinci ve üçüncü byte'lar bir adrestir.
FE	0F	00		LDX #F0F0	(MC6802) X kaydedicisine (0F00) ₁₆ verisini yükler. Toplam üç byte'lık bir komuttur. İlk byte komutun kendisi, ikinci ve üçüncü byte'lar doğrudan kullanılan veridir.
DD	21	00	1F	LD IX, 1F00H	(Z80) X kaydedicisine (1F00) ₁₆ verisini yükler. Dört byte'lık bir komuttur. İlk iki byte komutun kendisi, ikinci ve üçüncü byte'lar yüklenecek veridir.
ED	43	00	01	LD (0100H), BC	(Z80) BC kaydedicisini (0100) ₁₆ ve (0101) ₁₆ adreslerine yükler. Dört byte'lık bir komuttur. İlk iki byte komutun kendisi, ikinci ve üçüncü yüklenecek verinin adresidir.

Mikroişlemciler, hafızada yüklü olan programları adım adım işleyerek çalıştırırlar. Mikroişlemciler her adımda yalnızca bir komutun icrasını gerçekleştirebilir. Mikroişlemci önce hafızadan sıradaki komutu içeriye alıp komut kaydedicisine yükler. Daha sonra bu komutun görevinin tanımlanması için komut kod çözücü tarafından kodu çözülür. Mikroişlemci, komutun görevini icra ederek bir sonraki komutu hafızadan getirir. Tüm bunlar her komut için rutin olarak yapılır. Her komutun görevinin yerine getirilmesi belirli bir zaman almaktadır. Komutların görevlerini tamamlaması için farklı sayıda saat periyodları gerekir. Hangi komutun kaç saat periyodunda tamamlandığı mikroişlemci üreticileri tarafından kataloglarında yayınlanmıştır.

Örneğin MC6802 mikroişlemcisinde LDAA #F0 komutu ile A kaydedicisine (F0)₁₆ verisi yüklenmek istenmektedir. Mikroişlemci katalogunda bu komutun görevini tamamlayabilmesi için gereken süre 2 saat periyodu olarak belirtilmiştir. LDAA #F0, X komutunun görevini tamamlayabilmesi için gereken süre ise 5 saat periyodu olarak belirtilmiştir. Osilatör frekansı 1 MHz (her saat periyodu 1 µsn'dir.) olan mikroişlemci, 2 saat periyodunda tamamlanan bir komutu 2x1 µsn = 2 µsn, 5 periyotta tamamlanan bir komut ise 5x1 µsn = 5 µsn'de bitirir.

13.1.3 Assembler Yönlendiricileri (Pseudo; Sahte)

Assembly program satırlarında kullanılan yardımcı komutlardır. Aslında tam olarak komut değildirler, dolayısıyla bunlara sözde komut da denebilir. Çünkü assembler çevirici programlarına yön veren veya yardımcı olan etiketlerdir. Çevirici programlar, pseudo (okunuşu: su'do) komutları için gerçek bir makine kodu üretmezler. Örneğin END bir pseudo komutudur ve assembler çeviricisine program satırlarının bittiğini işaret eder. Çevirici, END komutundan sonraki komutları çevirmez ve bu komutla çevirme işlemine son verir. Pseudo komutları, assembler çeviricilerini şu noktalarda etkilemektedirler:

- ❖ Program kodlarının hafızadaki başlangıç yerlerini bildirmede
- ❖ Sabitleri, sembolleri veya etiketleri tanımlamada
- ❖ Geçici veriler için hafızada yer ayırmada
- ❖ Tablo veya sıralı verileri hafızaya yerleştirmede

Pseudo komutları ve tanımları tablo 13.2'de örnekleri ile birlikte verilmiştir.

Tablo 13.2 : Pseudo komutlar

Pseudo komutları	Açıklama	Örnek	
EQU	Eşitle; bir etikete değer atar.	PORT_A EQU \$0800	Program içerisinde (0800) ₁₆ yerine PORT_A etiketi kullanılabilir.
SET	Yeniden eşitle; Bu ifade ile programın herhangi bir yerinde aynı etikete yeniden değer atanabilir.	DURUM SET \$00 DURUM SET \$01	EQU'dan farkı, aynı etikete istendiği zaman yeni değer atanabilmesidir.
ORG	Programın başlangıç adresini gösterir. Program belirtilen bu adresten ardışık olarak sıralanır. ORG ile belirtilmezse program satırları (0000) ₁₆ adresinden itibaren sıralanır.	0100 ORG \$0100 LDAA #F0	ORG komutu ile belirtilen adresten sonraki ilk program komutu (0100) ₁₆ adresinden itibaren hafızaya yüklenir.

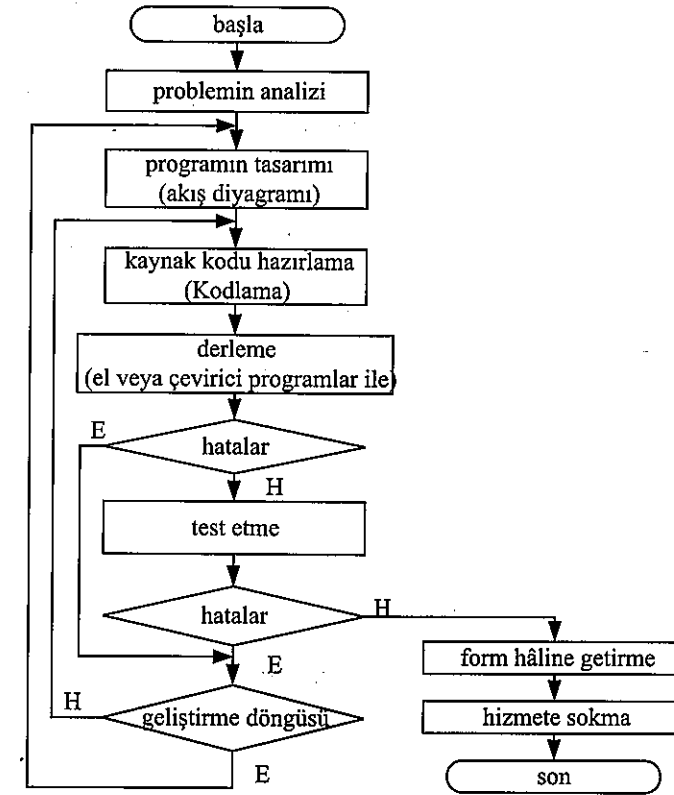
Tablo 13.2'nin devamı

DB veya DFB	Belirlenen veri veya verileri program sayıcının o anda gösterdiği yerden başlayarak ardışık hafıza konumlarında saklar. DB den önceki etiket bu verilerin ilkinin adresini tutar. DB aynı zamanda metinleri hafızada ASCII kodları ile saklar.	AD DB #0E DRM DB #00,#0F MSJ DB "HATA"	AD: (0E) ₁₆ olarak tanımlandı. DRM: (00) ₁₆ ve bir sonraki adreste (0F) ₁₆ saklanır. MSJ etiketi 'HATA' yazısının ASCII kodlarını ardışık olarak hafızada saklar. Yani hafızada (72) ₁₆ , (65) ₁₆ , (84) ₁₆ , (65) ₁₆ verileri saklanır.
DW veya DWM	Hafızada 2 byte'lık veri veya etiket tanımlanır. Genellikle tablo adreslerinin tutulmasında kullanışlıdır.	TABLO DW \$0100 TEST DW #01FF	(0100) ₁₆ adresini TABLO olarak belirtir. TEST (01FF) ₁₆ olarak tanımlandı.
DS veya DFS	Değişik veriler yükleyebilmek için geçici hafıza bölgesi ayırır.	BUFFER DS #7F	Hafızada bu komutun bulunduğu adresten itibaren 127 byte'lık (7F) ₁₆ boş alan ayırır.
MACRO	Sabit bir etiketle belirli işlerin yapılmasında kullanılır.	YUKLE MACRO PUSH HL PUSH BC PUSH DE ENDM	YUKLE etiketi ile tanımlanmış bir makro. Bu makro program satırlarında her kullanıldığında yandaki işlemleri yapacaktır. MACRO'nun sonunu bildirir.
ENDM	Makro sonu bildirimi.		
END	Assembly program satır sonu bildirimi.	END	Assembly program sonu.

13.2 ASSEMBLY PROGRAM YAZMANIN TEMELLERİ

Her işte olduğu gibi bilgisayar programcılığında da program yazmaya başlamadan önce plânlama yapmak zaman kaybını azaltır, hata oranını düşürür, kısacası kalite ve verimliliği artırır. Bunun için herhangi bir program hazırlama aşamasında aşağıdaki basamakların sıra ile yapılmasında büyük faydalar vardır. Bu bölümde program geliştirme basamaklarının neler olduğu ve her basamakta nelerin yapılması gerektiği anlatılmıştır. Şekil 13.3'de bilgisayar programcılığında genel olarak kabul görmüş bu işlem basamakları akış diyagramı şeklinde verilmiştir.

- ❖ Problemin açık bir tanımı yapılmalıdır.
- ❖ Bu problemde bize verilenler ve istenenler açıkça yazılmalı veya çizilmelidir.
- ❖ Problemin ayrıntılı analizi yapıldıktan sonra birden fazla çözüm yöntemi araştırılmalıdır.
- ❖ Bulunan çözüm yöntemleri arasında değişik kriterlere bakarak en uygunu seçilmelidir.
- ❖ Seçilen çözüm yönteminin akış diyagramı veya işlem basamakları oluşturulmalıdır.
- ❖ Çözüm için en uygun programlama dili seçilir.
- ❖ Akış diyagramına veya işlem basamaklarına göre belirlenen dille program kodları yazılmalıdır.
- ❖ Program çalışır hâle getirilmelidir (derlenme aşaması).
- ❖ Programın test edilmesi. Bu aşamada hata tespitleri yapılmalıdır.
- ❖ Varsa hatalar düzeltilmeli ve test aşaması tekrar edilmelidir.
- ❖ En son programın, problemin çözümündeki yeterlilikleri ve sınırlılıkları gibi değerlendirmeler yapılmalı varsa ihtiyaç duyulan eklentileri yapılmalıdır.



Şekil 13.3 : Temel programlama tekniğinin akış diyagramı ile gösterilmesi

13.2.1 Problem Analizi

İyi bir tasarımcının herhangi bir ürün üzerindeki çalışmalarını her zaman için en ucuz, en az zamanda (yapılan işin toplam süresi üzerinden) ve en yalın olacak şekilde yapması gerekir. Hiçbir zaman basit bir problemin çözümünde çok karmaşık bir yol izlenmesi ve lüzumsuz birçok eleman kullanması kabul edilebilir bir şey değildir. Bu daha çok tasarımcının işi iyi kavrayamadığını gösterir. Bunun için öncelikle bir problemin bilgisayarla çözümünün kullanıcıya getireceği avantajlar tespit edilmelidir. Eğer problemin çözümünde bilgisayarın kullanılması, diğer yöntemlere göre birtakım avantajlar sağlayacaksa o zaman bu yöndeki çalışmalara başlanmalıdır. Daha işin başında bu tespitte bulunmak yapılacak çalışmalar için harcanan zaman ve paranın boşa gitmemesi açısından çok önemlidir.

Problemin ne olduğunu ve nasıl çözümlemeyi düşündüğümüzü, yapacağımız basit şekil, şema ve karalamalar ile boş bir kağıda dökmek büyük ölçüde problemi kavramamıza yarar sağlayacaktır. Konumuz bilgisayar programlama olduğu için herhangi bir problemin bilgisayar ile çözülebilirliğine karar vermek gerekecektir. Kahve makinesi varken kahveyi bilgisayara yaptırmaya kalkışmak hiç de akıllıca olmayacaktır.

13.2.2 Akış Diyagramı

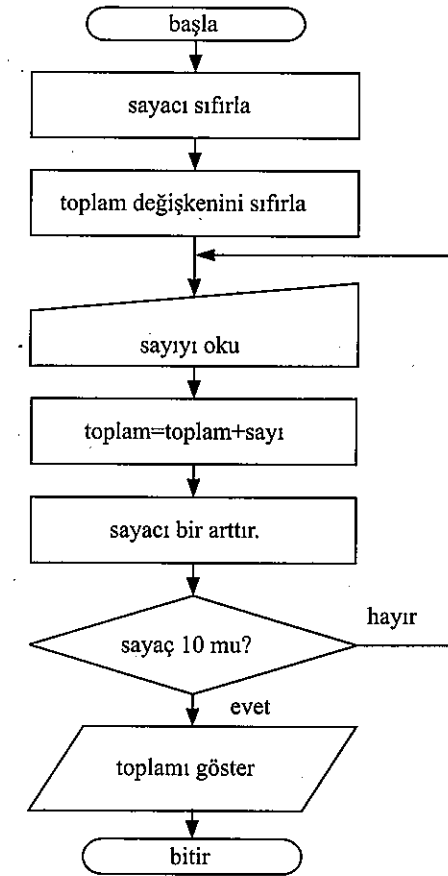
Programlamaya başlamadan önce kafamızda tasarladığımız düşüncelerimizi belirli mantık sırasına göre düzenleyip programın işleyişini anlatan bir diyagram çizeriz. Problemin çözümünde, belirli bir sıra takip edileceğine göre bunu standart bir diyagrama aktarmamız, program kodlarını yazmaya başladığımızda büyük kolaylıklar sağlayacaktır. Akış diyagramları, problemin çözümünde izlenen yolun veya yöntemin yeniden gözden geçirilmesine de büyük katkılar sağlayacaktır. Aynı zamanda akış diyagramı hazırlamak, problemin çözümünde izlenen yolun başkaları tarafından da değerlendirilebilmesi açısından evrensel bir yöntemdir.

Akış diyagramlarında kullanılan sembollerin her biri ayrı bir işlemi ifade eder. Akış diyagramındaki semboller, problemin çözümünde izlenen yöntemi ifade edecek şekilde seçilmelidir. Örneğin bir işlemin sonucuna göre program akışı değişiyorsa kullanılacak sembol baklava dilimi şeklinde olan “karar” sembolü olmalıdır. Veya tekrarlı bir işlem için “döngü” sembolü kullanılır. Tablo 13.3’de her akış sembolünün anlamı yanında belirtilmiştir.

Tablo 13.3 :Akış diyagramı sembolleri

	İşlem
	Önceden tanımlı işlem
	Karar
	Döngü
	Giriş/çıkış
	Elle girdi
	Elle işlem
	Program başı veya sonu
	Yazıcı

Şekil 13.4 10 adet sayının sıra ile toplanmasında izlenen yolun, akış diyagramı ile nasıl gösterildiğine bir örnektir. Diyagramda ilk önce “sayaç” ve “toplam” değişkenleri sıfırlanmaktadır. Sonra ilk sayı okunmaktadır. Okunan sayı toplam değişkenine eklenir. Sayaç 10 adet girişin kontrolünü yapabilmek için bir artırılır. Sayaç kontrolü yapılır, eğer sayaç değeri 10 olmamış ise yeni sayı girişine dallanılır ve yeniden giriş istenir. Eğer on adet giriş yapılmış ise toplam değişkeninin değeri ekranda gösterilerek program bitirilir.



Şekil 13.4 : 10 sayının toplamını yapan ve sonucu gösteren akış diyagramı

Şekil 13.4'deki akış diyagramına göre hazırlanmış BASIC programı görülmektedir.

```

' Bu program girilen 10 adet sayının toplamını verir.
SAYAC=0
TOPLAM=0
YENIDEN: INPUT "Bir sayı giriniz =";A
TOPLAM=TOPLAM+A
SAYAC=SAYAC+1
IF SAYAC<10 THEN GOTO YENIDEN
PRINT "10 adet sayının toplamı =", TOPLAM
END
  
```

13.2.3 Assembly Kodlama

Assembly kodlama, akış diyagramında belirlenen aşamaların assembly dili komutları ile kaynak programın hazırlanmasıdır. Kodlama, kullanılan mikroişlemcinin komut setinden, istenen işe uygun assembly komutunun kullanılmasıdır. Assembly komutların yazılacağı yer herhangi bir metin editörü olabileceği gibi çeviricinin kendi program geliştirme editörü de olabilir.

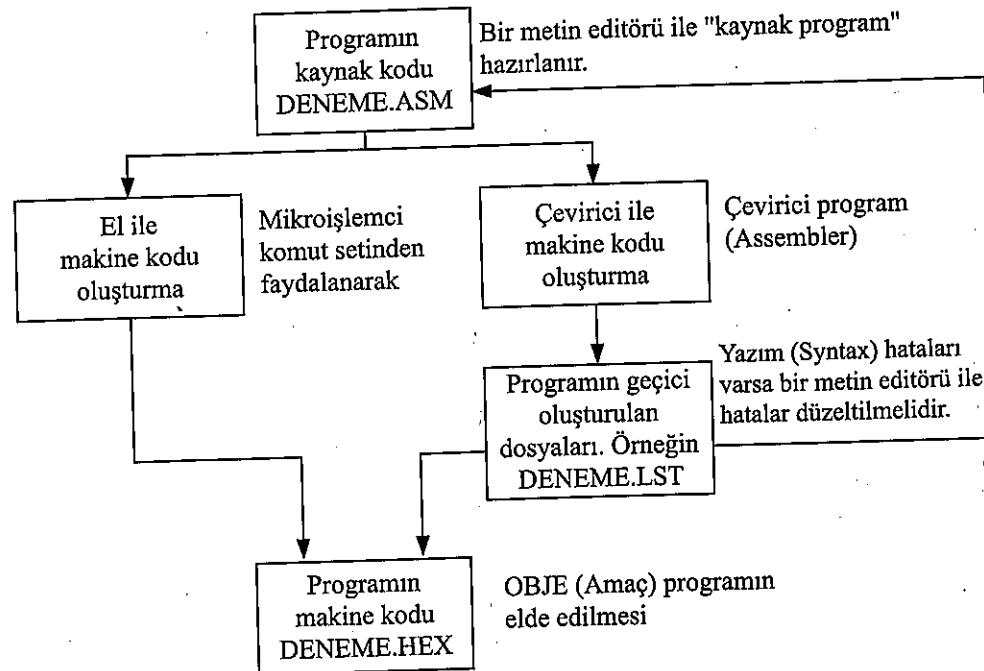
Mikrobilgisayarın hafızasının (0100)₁₆ adresinden (0109)₁₆ adresine kadar olan aralıkta bulunduğu varsayılan 10 adet sayının toplanmasını assembly programlama dili kullanarak yazmak istediğimizde tablo 13.4'deki program satırlarını yazmamız gerekecektir. Programın kendisi (0200)₁₆ adresinden itibaren bulunmaktadır. Toplamın sonucu ise (0300)₁₆ adresine yazılacaktır. (0100)₁₆ ve (0109)₁₆ adresleri arasında bulunan sayıların toplam değeri 255'den büyük olmadığı varsayılacaktır. Program MC6802 mikroişlemcisinin komut setindeki assembly komutları kullanılarak hazırlanmıştır.

Tablo 13.4 : 10 sayının toplamını bulan assembly programı (MC6802 komutları ile)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0200	CE 01 00		LDX #0100	İndeks kaydedicisi (0100) ₁₆ adresi ile yüklenir.
0203	4F		CLRA	A kaydedicisi sıfırlanır.
0204	E6 00	TEKRAR:	LDAB \$00,X	B kaydedicisine indeks kaydedicisinin gösterdiği adresten bir değer yüklenir.
0206	1B		ABA	A=A+B
0207	08		INX	İndeks kaydedicisini 1 artırır.
0208	8C 01 09		CPX #0109	İndeks kaydedicisi (0109) ₁₆ değeri ile karşılaştırılır.
020B	26 F7		BNE TEKRAR	İndeks kaydedicisi (0109) ₁₆ adrese ulaşmadysa TEKRAR etiketine dallanılır.
020D	B7 03 00		STAA \$0300	A kaydedicisindeki toplam sonuç (0300) ₁₆ adresine yüklenir
0210	3F		SWI	Program bitirilir.

Herhangi bir metin editöründe hazırlanan programın kaynak kodunda, doğal olarak assembly dilindeki (LDA, STX...) komutlar bulunacaktır. Mikroişlemci ancak makine dilindeki komutlar ile çalışır. Bundan dolayı assembly dili ile hazırlanmış programın kaynak kodundaki komutların makine diline çevrilmesi gerekmektedir. Bunun için kullanılan iki yöntem vardır: “el ile çevirme” veya “assembly çeviricisi” yani “assembler” programı kullanmak. Sonuçta iki yolla da mikroişlemcinin çalışması için gereken makine komutları oluşturulabilmektedir.

Makine kodlarının el ile çevrilmesinde yapılacak hatalar ancak amaç programın çalışmasındaki hatalarla tespit edilebilir. Fakat çevirici program kullanıldığında yazılan yazım hataları programcıya hemen bildirilerek daha amaç program oluşturulmadan düzeltme imkânı sağlanmış olur.



Şekil 13.5 : “Kaynak programdan” “amaç programın” elde edilmesi yöntemleri

13.2.4 Assembly Komutlarının El ile Makine Diline Çevrilmesi

Assembly dili ile hazırlanmış kaynak programın, makine diline çevrilmesinde “el ile çevirme” diye bilinen yöntemde kullandığımız mikroişlemcinin “komut tablosunu” bulundurmamız gerekecektir. Bir mikroişlemcinin komut tablosunda, kullanılan assembly komutlarının makine dilindeki karşılıkları (makine kodu) bulunacaktır. Bizim yapmamız gereken; hazırladığımız kaynak programda bulunan her assembly komutu için mikroişlemcinin komut setindeki makine kodunu bulmak ve yazmak olacaktır. Burada dikkat etmemiz gereken nokta kullandığımız assembly komutunun hangi adresleme modunda kullanıldığıdır. Çünkü aynı assembly komutunun farklı adresleme modlarındaki makine kodları farklı olmaktadır. Bunun nedeni “Adresleme modları” bölümünde anlatılmıştır.

Tablo 13.5 : MC6802'nin komut setinden örnek komutlar

Mnemonic Komut	İvedi adresleme	Direkt adresleme	İndeks adresleme	Genişletilmiş adresleme	İmalı adresleme	Göreceli adresleme	
ABA	-	-	-	-	1B	-	A+B → A
CLRA	-	-	-	-	4F	-	00 → A
LDAB	C6	D6	E6	F6	-	-	
STAA	-	97	A7	B7	-	-	A → M
CPX	8C	9C	AC	BC	-	-	(X _H / X _H) - (M, M+1)
INX	-	-	-	-	08	-	X + 1 → X
LDX	CE	DE	EE	FE	-	-	M → X _H , (M+1) → X _L
BNE	-	-	-	-	-	26	Eğer Z=0 ise
SWI	-	-	-	-	-	3F	

MC6802 mikroişlemcisi için hazırlanan “10 sayının toplamını bulan programda” (tablo 13.4) kullanılan komutların makine kodları tablo 13.5’de verilmiştir. Programdaki assembly komutlarının, makine kodlarına (makine koduna) çevrilmesinde mikroişlemcinin “komut tablosundan” faydalanılmıştır. Programdaki komutların ne iş yaptığı aşağıda anlatılmıştır.

LDX #0100: Bu komut ile indeks kaydedicisi (0100)₁₆ değeri ile yüklenir. (0100)₁₆ adresi sıralı şekilde bulunan 10 adet verinin ilk adresidir. İndeks kaydedicisine doğrudan veri yüklemesi yapılacağı için kullanılan adresleme modu ivedi adreslemedir. LDX komutunun ivedi adresleme modundaki makine kodu tablodan da görüleceği gibi (CE)₁₆’dır. Hafızada üç byte yer tutar.

CLRA: Program, hafızadan okunan sayıları birbirine eklenerek toplar ve toplamı da A kaydedicisinde tutmaktadır. A kaydedicisi toplama sonucunu tutacağından işleme başlamadan önce içeriği silinmiştir. CLRA komutu A kaydedicisini silmek için kullanılmıştır. CLRA komutu yalnızca imalı adresleme modunda kullanılır ve işlem kodu (4F)₁₆’dır. Hafızada bir byte yer tutar.

LDAB \$00,X: Bu komut, indeks kaydedicisinin göstermiş olduğu adresten veri okuyarak B kaydedicisine yükler. Komut indeks adresleme modunda kullanıldığı için makine kodu (E6)₁₆ olacaktır. Hafızada 3 byte yer tutar.

ABA: B kaydedicisinde bulunan değer ile A kaydedicisinde bulunan değerler toplanır. Toplamın sonucu yine A kaydedicisinde tutulur. Komutun makine kodu (1B)₁₆’dır. Hafızada bir byte yer tutar.

INX: Bu komutla İndeks-kaydedicisinin içeriği bir artırılır. İndeks kaydedicisi hafızada sıralı şekilde bulunan 10 adet verinin adreslerini saklamaktadır. Her INX komutundan sonra indeks kaydedicisi bir sonraki verinin adresini işaret edecektir. Komut tek adresleme modunda kullanılabilmektedir. İmalı adresleme modunda kullanılan komutun makine kodu (08)₁₆’dır.

CPX #0109: İndeks kaydedicideki değer ile (0109)₁₆ değeri karşılaştırılmaktadır. Bu karşılaştırmanın sonucunda durum kodu kaydedicisindeki N, Z ve V bitleri etkilenir. Eğer karşılaştırma

sonucunda sayılar birbirine eşit ise Z biti Z=1 olacaktır. Komut ivedi adresleme modunda kullanıldığı için işlem kodu (8C)₁₆'dır.

BNE TEKRAR: Bir önceki karşılaştırmanın sonucuna göre dallanır. Eğer bir önceki son komutun etkilediği Z biti Z=0 ise TEKRAR etiketine dallanacaktır. Eğer Z=1 ise dallanma gerçekleşmeyecek ve program bir sonraki komuttan devam edecektir. Komut göreceli adresleme modunda kullanıldığından makine kodu (26)₁₆'dır.

STAA \$0300: A kaydedicisindeki değeri (0300)₁₆ adresine yükler. A kaydedicisi hafızada bulunan 10 adet verinin toplamını içermektedir. STAA \$0300 komutu ile 10 adet verinin toplamı (0300)₁₆ adresinde saklanacaktır. Komut genişletilmiş adresleme modunda kullanıldığı için makine kodu (B7)₁₆'dir.

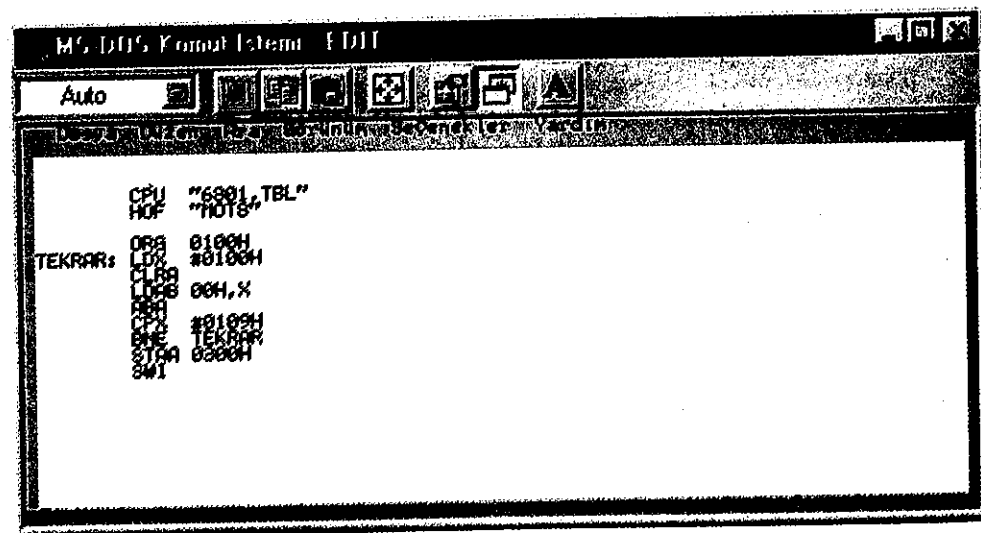
SWI: Bu komut ile program bitirilir. Mikroişlemci bu noktadan sonraki çalışmasına, SWI vektöründe belirtilen adresten devam edecektir. Komutun makine kodu (3F)₁₆'tir.

Hazırladığımız assembly programındaki komutların makine dilindeki karşılıklarını, mikroişlemcinin komut setinden faydalanarak assembly program tablosundaki ilgili yere yazarız. Şimdi elde ettiğimiz makine kodlarını sistemin hafızasına girerek programı çalıştırmamız gerekecektir. Kullandığımız mikrobilgisayarın klavye arabirimini kullanarak sistemin hafızasına bu kodları tek tek girebileceğimiz gibi bir kişisel bilgisayar aracılığı ile de mikrobilgisayarın hafızasına yüklenebilir. Kodları, mikrobilgisayarın hafızasına klavye ve gösterge arabirimini kullanarak girmek, çok uzun satırlı programlar için oldukça zor bir iştir ve hata yapmak ve hataları bulmak programcıyı bir hayli uğraştırır. Hazırladığımız programın bir kişisel bilgisayar aracılığı ile mikrobilgisayara aktarılması daha profesyonel bir yaklaşım olacaktır.

13.2.5 Assembly Komutlarının Assembler Programı ile Makine Diline Çevrilmesi

Hazırlanan assembly program komutlarının el ile makine koduna çevrilmesi programcının çok büyük zaman kaybına ve hata yapmasına sebep olur. Elle kodlama programlamaya yeni başlayanlar ve kısa programlar için uygun olsa da tecrübeli bir programcı için "assembly çeviricileri" vazgeçilmez bir program geliştirme ortamıdır. Assembly dilinde yazılan kaynak programın çeviriciler ile makine diline dönüştürülmesi çok hızlı ve hatasız olmaktadır. Aynı zamanda çeviriciler, kaynak programdaki yazım hatalarını tespit ederek programcıya hızlı hata düzeltme imkânı da sağlamış olacaktır.

"Cross assembler" birçok programcının kullandığı bir çeviri programdır. Bu programın kullanılması için öncelikle bir kişisel bilgisayara ihtiyaç vardır. Çevirici program, kişisel bilgisayara yüklendikten sonra herhangi bir metin editöründe kaynak program hazırlanır. Yazılan kaynak program, uzantısı ".ASM" olacak şekilde kaydedilir. Örneğin "DENEME.ASM". Resim 13.1'de, MS-DOS metin editöründe yazılmış örnek bir kaynak program görülmektedir.



Resim 13.1 : Metin editöründe yazılmış kaynak dosya "DENEME.ASM"

Hazırlanan kaynak program ".ASM" uzantılı bir dosya olarak kaydedildikten sonra DOS ortamında şekil 13.6'daki gibi bir komut satırı yazılır.

```
C:\Cross>C32 DENEME.ASM -L DENEME.LST -H DENEME.HEX
```

Şekil 13.6 : Cross Assembler çeviricisinin (C32.EXE) komut satırı

Bu komut satırı ile DENEME.ASM dosyasındaki assembly komutları C32 çevirici programı tarafından makine diline çevrilir. Eğer kaynak programda yazım hatası (syntax error) varsa hatalı satır numaraları ile birlikte rapor edilir. Şekil 13.7'de görüleceği üzere, çevirici tarafından tespit edilmiş iki hatalı satır bulunmaktadır. Assembler tarafından "CLRA" ve "STAA \$0300" komutların bulunduğu satırların hatalı yazıldığı rapor edilmektedir.

```
C:\Cross>C32 DENEME.ASM -L DENEME.LST -H DENEME.HEX
Cross-32 Meta-Assembler PC/MS-DOS Version 1.13
Copyright (C) 1989 Universal Cross-Assemblers
```

Starting pass number 1

Starting pass number 2

```
E0103 00    CLR
E010C       STAA $0300
```

End of Assembly -- 2 Errors

Şekil 13.7 : Cross Assembler'in hata raporu

Eğer program yazımında hiçbir yazım hatası yoksa çevirme işlemi tamamlanır (şekil.13.8). Çevirme işleminin sonucunda, DENEME.LST (şekil 13.9) ve DENEME.HEX (şekil 13.10) dosyaları oluşturulur. DENEME.LST bir rapor dosyası niteliğindedir. DENEME.HEX dosyası ise kaynak program dosyasının makine diline çevrilmiş hâlidir. Deneme.hex dosyası aynı zamanda seri iletimde kullanılabilecek "S" formatında düzenlenmiştir. DENEME.HEX dosyasına "obje dosyası" denir.

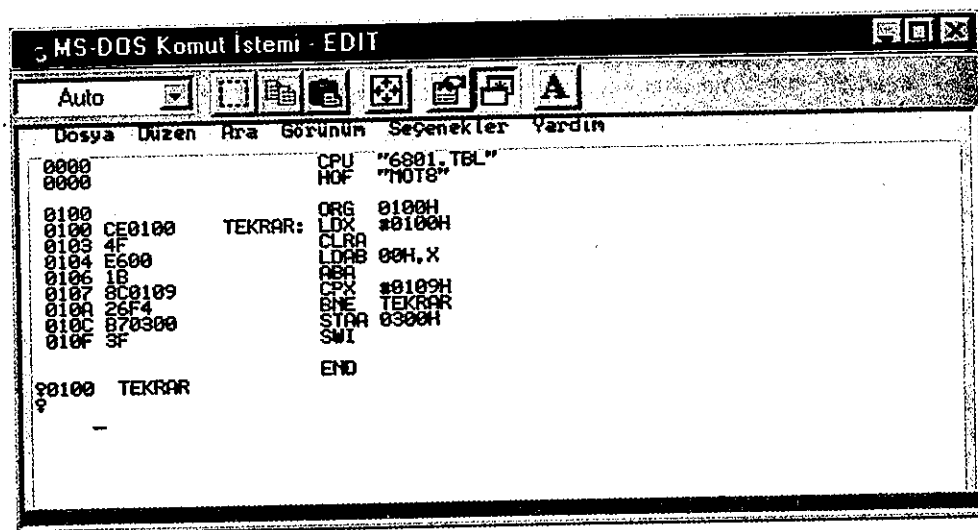
```
C:\Cross>C32 DENEME.ASM -L DENEME.LST -H DENEME.HEX
Cross-32 Meta-Assembler PC/MS-DOS Version 1.13
Copyright (C) 1989 Universal Cross-Assemblers
```

```
Starting pass number 1
```

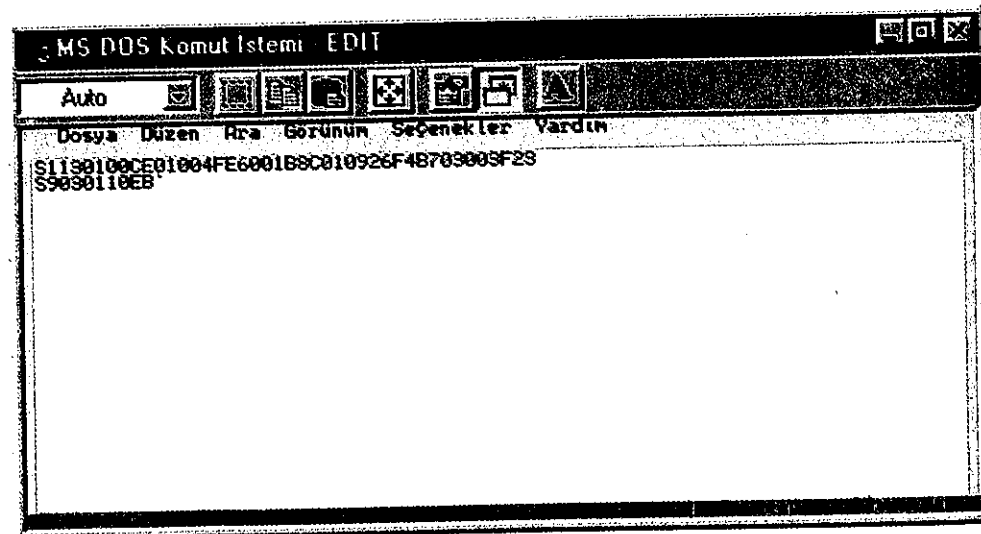
```
Starting pass number 2
```

```
End of Assembly -- No Errors
```

Şekil 13.8 : Hatasız bir assembly kaynak dosyasının Cross Assembler tarafından sonuçlandırılması



Şekil 13.9 : Cross Assembler tarafından oluşturulan rapor niteliğindeki ".LST" uzantılı dosya



Şekil 13.10 : Cross Assembler tarafından oluşturulan ".HEX" uzantılı obje dosya

13.2.6 Hataların Düzeltilmesi (Debugging)

Programcılar için hazırlanan bir programın test edilmesi ve test sırasında bulunan hataların yeniden düzenlenmesi olağan bir şeydir. Çünkü yüzlerce program komutlarının yazımı sırasında hata yapmak kadar doğal bir şey yoktur. Hata yapmak bu işin bir parçası olarak kabul edilmelidir. Önemli olan hataların tespitinde doğru test veya yöntemlerin seçilmesidir. Program hazırlanırken, genel olarak aşağıdaki noktalara dikkat edilirse hem hata oranı azalacaktır hem de olağan hataların tesbiti kolaylaşacaktır.

- ❖ Gereksiz detaylardan kaçınılmalıdır.
- ❖ Özel durumlar en aza indirgenmelidir. Özel durumlar özel test ve kontrol gerektirebilir.
- ❖ Programın üzerinde işlem yapacağı veriler kontrol edilmelidir.
- ❖ İşaretili sayıların kullanımında işaret bitlerine dikkat edilmelidir.
- ❖ Özellikle kaydedicilerin ve hafızaların sınır değerlerine dikkat edilmelidir.
- ❖ Yığın kullanımında, yığına itilen ve yığından çekilen veriler en çok hata yapılan yerlerdir.
- ❖ Programlar mümkün olduğunca genel kurallara göre hazırlanmalıdır.
- ❖ Her komutun yanına kısa açıklamalar eklenmelidir.
- ❖ Çok uzun programlar olabildiğince modüler yapıda hazırlanmalıdır. Modüllerin tek başına hata tespiti daha kolay olabilmektedir.
- ❖ Alt programa parametre aktarımlarına ve geri dönüş değerlerine dikkat edilmelidir.
- ❖ Bir sistemde birden çok hatalar bulunabilir. Bu hatalar genel olarak iki türden olabilir: "yazılım hataları" ve "donanım hataları".

Bilgisayar sistemlerinde donanım hatalarının tespiti için öncelikle çalışması ve güvenilirliği daha önceden sınanmış basit test programları kullanmak çok kullanışlı ve geçerli bir yöntemdir. Hazırlanan test programları ile sistemin giriş değerlerine karşılık doğru cevap verip vermediği ölçü aletleri (osilaskop, lojik analizör, lojik prop,...) ile ölçülerek yapılabilir. Donanım hatalarının tespitinde dijital elektronik devrelerindeki arıza arama yöntemlerinin hepsi kullanılabilir.

Yazılım hatalarının tespiti için öncelikle donanımın doğru çalıştığından emin olmak gerekir. Sağlıklı çalışmayan bir bilgisayar sisteminde, hatasız bir program da doğru çalışmayabilir. Yazılım hataları iki türden olur;

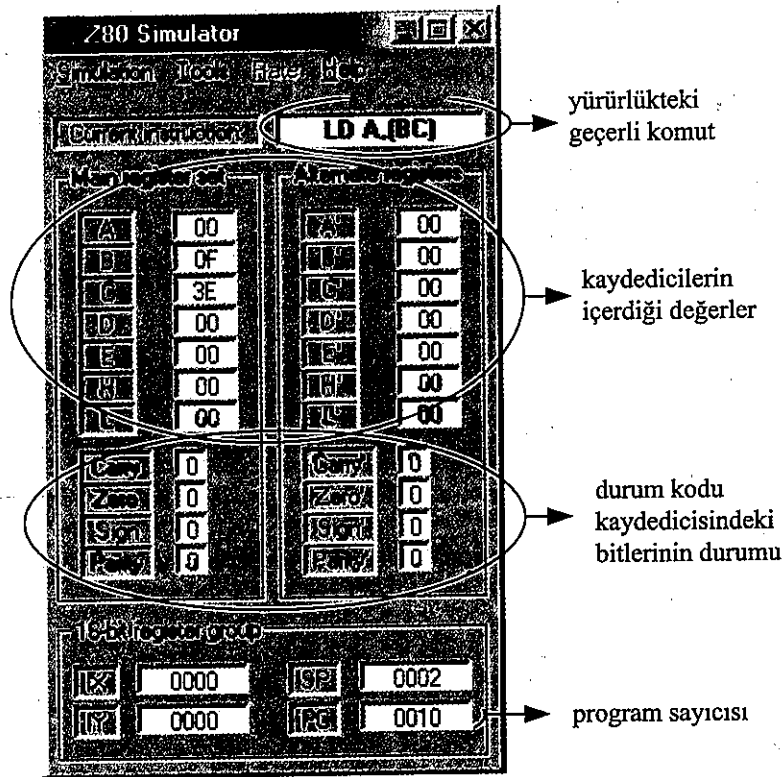
- ❖ Yazım (syntax) hataları,
- ❖ Mantık hataları,

Yazım hatalarını tespit etmek, mantık hatalarını tespit etmekten çok daha kolaydır. Yapılan yazım hatalarının tespiti kullanılan çevirici (assembler) programları tarafından programcıya listelenmektedir. Yazım hataları herhangi bir assembly komutunun yanlış yazılması (LDA yerine LA yazılması gibi) ya da yanlış bir etiket kullanılması olabilir. Bu gibi hatalar çevirici veya düzenleyici programlar tarafından tespit edilerek programcıya bu hatalar rapor edilir.

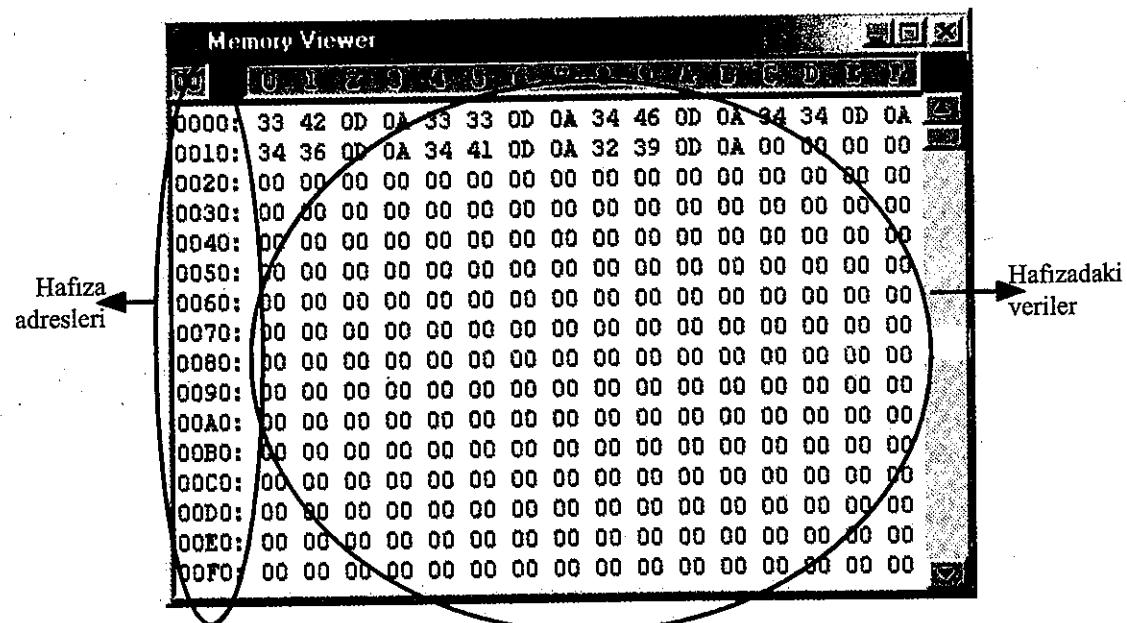
Fakat yapılan mantık hatalarının örneğin; bir azaltma "DECA"komutu yerine, arttırma "INCA" komutunun kullanılmasında çevirici veya düzenleyicinin bulacağı ve rapor edeceği bir yazım hatası yoktur. Ancak bu tip hatalar, programın istenen sonucu vermemesine sebep olacaktır. Veya doğru kurulmayan bir akış diyagramına göre hazırlanmış bir program, doğru sonuçlar üretmez. Yazılımların test edilmesinde ve hataların tespitinde kullanılan yöntemler;

- ❖ Simülasyon programları
- ❖ Hata ayıklama programları (Debug)
- ❖ Emülatör

Simülasyon programları, kişisel bilgisayarda çalışan ve seçilen mikroişlemcinin çalışmasını taklit eden programlardır. Her mikroişlemcinin simülasyon programları farklıdır. Şekil 13.11'de Z80 için hazırlanmış bir simülasyon programının ara yüzü görülmektedir. Simülasyon programlarıyla kaynak programın çalışması sırasında, mikroişlemcinin kaydedicilerindeki değişiklikler adım adım izlenebilir. Aynı zamanda mikroişlemcinin adresleyebileceği hafıza alanı da izlenebilmektedir (şekil 13.12). Bunların tamamı simülasyon programlarının bir mikroişlemci ve hafıza üzerinde yaptığı hayali çalışmadır.



Şekil 13.11 : Simülasyon programı (Z-80 mikroişlemcisi için)



Şekil 13.12 : Simülasyon programının hafıza içeriği

Debug program hatalarını bulmada kullanılan bir programdır. Debug programı ayrıca kısıtlı program komutları yazma, hafıza içeriğini gösterme, hafıza üzerinde değişiklik yapma ve programın hafızada istediğimiz bir adrese taşınmasına imkân sağlar. Tablo 13.6'da debug programında kullanılan komutların listesi verilmiştir.

Tablo 13.6 : Debug programında kullanılan komutların listesi

Debug komutu	Komut parametreleri	Açıklama
A	[adres]	birleştir
C	aralık adresi	karşılaştır
D	[aralık]	göster
E	adres [liste]	giriş
F	aralık listesi	doldur
G	[=adres] [adresler]	git
H	değer1 değer2	hex
I	bağlantı noktası	giriş
L	[adres] [sürücü] [ilk kesim] [sayı]	yükle
M	aralık adresi	taşı
N	[yol adı] [arg listesi]	ad
O	bağlantı noktası byte'ı	çıkış
P	[=adres] [sayı]	devam
Q		çık
R	[yazmaç]	yazmaç
S	aralık listesi	ara
T	[=adres] [değer]	izle
U	[aralık]	dağıt
W	[adres] [sürücü] [ilk kesim] [sayı]	yaz

Komut "D"nin çalıştırılması sonucu elde edilen veriler aşağıda verilen (şekil 13.13) monitör ekranındaki gibi görülmektedir. Komut "D" belirtilen adres aralığındaki hafıza içeriğini gösterir.

```
C:\>DEBUG
-D 0000:0100
0000:0100 59 EC 00 F0 D5 EF 00 F0-65 F0 00 F0 2C E7 FF FF Y.....e.....
0000:0110 E7 E7 00 F0 E7 E7 00 F0-E7 E7 00 F0 E7 E7 00 F0 .....
0000:0120 E7 E7 00 F0 E7 E7 00 F0-E7 E7 00 F0 37 26 4E FD .....7&N.
0000:0130 E7 E7 00 F0 E7 E7 00 F0-E7 E7 00 F0 FC 04 70 00 .....p.
0000:0140 6F EF 00 F0 E7 E7 00 F0-E7 E7 00 F0 E7 E7 00 F0 O.....
0000:0150 E7 E7 00 F0 E7 E7 00 F0-E7 E7 00 F0 E7 E7 00 F0 .....
0000:0160 E7 E7 00 F0 E7 E7 00 F0-E7 E7 00 F0 E7 E7 00 F0 .....
0000:0170 03 00 39 10 E7 E7 00 F0-E7 E7 00 F0 E7 E7 00 F0 ..9.....
```

Şekil 13.13 : D 0000:0100 komutunun bilgisayar ekranındaki sonucu

Komut "R", mikroişlemcilerin kaydedicilerindeki değerleri ekranda gösterir. Komut R'nin etkisi şekil 13.14'de gösterilmektedir.

```
C:\>DEBUG
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1D27 ES=1D27 SS=1D27 CS=1D27 IP=0100 NV UP EI PL NZ NA PO NC
1D27:0100 F6450101 TEST BYTE PTR [DI+01],01 DS:0001=20
```

Şekil 13.14 : R komutunun bilgisayar ekranındaki sonucu

DEĞERLENDİRME ÇALIŞMALARI

- 1) Program nedir?
- 2) Bir bilgisayarın şematik yapısını çiziniz.
- 3) Yazılım ve donanım kavramları nedir? Açıklayınız.
- 4) Assembly programlama dilinin diğer dillerden farkını anlatınız.
- 5) Assembly program satırlarında kullanılan verilerin tanımlanmasında kullanılan sembollerin anlamlarını yazınız.
- 6) Komutların hafıza gereksinimleri ve işlem sürelerini nasıl belirleriz?
- 7) Assembler yönlendiricilerini yazınız. Bunların komutlardan farkı nedir?
- 8) Assembler yönlendiricileri programcıya ne tür kolaylıklar sağlayabilir? Anlatınız.
- 9) Bir problemin bilgisayarda çözümünü sağlıklı bir şekilde yapabilmek için hangi adımlar izlenmelidir?
- 10) Assembly komutları sistem hafızasına kaç türlü yüklenebilir?
- 11) Mikrobilgisayar hataları kaç tür olabilir?
- 12) Yazılım hataları kaç tür olabilir?
- 13) Yazılım hatalarının tespitinde kullanılan yöntemler nelerdir?

ÜNİTE 14 ASSEMBLY PROGRAM ÖRNEKLERİ

14.1 BASİT PROGRAM ÖRNEKLERİ

Basit program örnekleri ile assembly komutlarının program içerisinde kullanımı hakkında temel bilgi sahibi olunur. Basit program örnekleri için akış diyagramları çizmeye gerek duyulmamıştır.

14.1.1 İki Sayının Toplanarak Sonucun Başka Bir Hafızaya Yükleme

Amaç; (0100)₁₆ ve (0101)₁₆ adreslerindeki sayıların toplanarak sonucun (0200)₁₆ adresine yüklenmesi.

Açıklama: Toplanacak sayıların (0100)₁₆ ve (0101)₁₆ adreslerine önceden yüklendiğini varsayalım.

Önce bu adreslerdeki sayılar LDAA ve LDAB komutu ile A ve B kaydedicilerine yüklenir. Daha sonra ABA komutu ile bu iki sayı toplanır. ABA komutu toplanan sayıların sonucunu A kaydedicisinde tutar. Toplamın sonucu A kaydedicisinden (0200)₁₆ adresine yüklenir. Doğru sonuç için toplamın 255'ten küçük olması gerekir. Sonuç 8 bitten büyük ise bu sayıların toplamı için 16 bitlik toplama programları kullanılmalıdır. 16 bitlik toplama programları toplamın sonucunu hafızanın art arda iki hücrene yüklerler.

Tablo 14.1 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	B6 01 00		LDAA \$0100	A kaydedicisine (0100) ₁₆ 'deki veriyi yükle
0003	F6 01 01		LDAB \$0101	B kaydedicisine (0101) ₁₆ 'deki veriyi yükle
0006	1B		ABA	A ve B kaydedicilerini topla
0007	B7 02 00		STAA \$0200	A'daki veriyi (0200) ₁₆ 'e kopyala
000A	3F		SWI	Programı bitir.

14.1.2 Bir Hafıza Hücresinin veya Kaydedicinin Silinmesi

Amaç; A kaydedicisinin ve hafızanın (0100)₁₆ adresindeki hücrenin silinmesi.

Açıklama: CLR komutu belirtilen (0100)₁₆ adresindeki veriyi sıfırlar yani siler. CLRA komutu ile de A kaydedicisinin içeriği sıfırlanır.

Tablo 14.2 : Programın kaynak kodları (MC6802 mikroişlemcisi için) Programı bitir.

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	7F 01 00		CLR \$0100	(0100) ₁₆ adresindeki veri silinir.
0003	4F		CLRA	A kaydedicisini siler (0 ile yükler).
0004	3F		SWI	Programı bitir.

14.1.3 Hafızadaki Verilerin Yer Değiştirilmesi

Amaç; hafızanın (0100)₁₆ ve (0101)₁₆ adreslerindeki verilerin yer değiştirilmesi.

Açıklama: (0100)₁₆ adresindeki veri A kaydedicisine yüklenir. (0101)₁₆ adresindeki veri de B kaydedicisine yüklenir. Daha sonra A kaydedicisindeki veri (0101)₁₆ adresine B kaydedicisindeki veride (0100)₁₆ adresine yüklenir.

Tablo 14.3 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	B6 01 00		LDAA \$0100	A kaydedicisine (0100) ₁₆ 'deki veriyi yükle
0003	F6 01 01		LDAB \$0101	B kaydedicisine (0101) ₁₆ 'deki veriyi yükle
0006	F7 01 00		STAB \$0100	B'deki veriyi (0100) ₁₆ 'e kopyala
0009	B7 01 01		STAA \$0101	B'deki veriyi (0101) ₁₆ 'e kopyala
000C	3F		SWI	Programı bitir.

14.1.4 Kaydedicideki Bir Verinin İlk Dört Bitinin Maskelenmesi

Amaç; hafızanın (0100)₁₆ adresinde bulunan bir verinin düşük değerlikli ilk dört bitinin maskelenmesi.

Açıklama: (0100)₁₆ adresindeki veri A kaydedicisine yüklenir. A kaydedicisindeki veri ile ANDA komutunun önündeki #11110000B verisi mantıksal olarak AND işlemine tâbi tutulur. Bu işlemin sonucunda A kaydedicisindeki verinin düşük değerlikli ilk dört biti sıfırlanmış ya da maskelenmiş olur (XXXX0000)₁₆. A kaydedicisindeki maskelenmiş veri tekrar (0100)₁₆ adresindeki hafıza hücreğine yazılır.

Tablo 14.4 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	B6 01 00		LDAA \$0100	A kaydedicisine (0100) ₁₆ 'deki veriyi yükle
0003	84 F0		ANDA #11110000B	A kaydedicisindeki veri ile #11110000B verisini mantıksal AND işlemi uygular.
0005	B7 01 00		STAA \$0100	A'daki sonucu tekrar (0100) ₁₆ 'e kopyalar.
0008	3F		SWI	Programı bitir.

14.1.5 Hafızadaki İki Sayının Karşılaştırılması

Amaç; hafızanın (0100)₁₆ ve (0101)₁₆ adreslerindeki sayıların karşılaştırılması.

Açıklama: (0100)₁₆ adresindeki veri A kaydedicisine yüklenir. (0101)₁₆ adresindeki veri de B kaydedicisine yüklenir. Daha sonra A kaydedicisindeki veri ile B kaydedicisindeki veri karşılaştırılır. Karşılaştırma sonucunda kaydedicilerin içerikleri değişmez, fakat durum kodu kaydedicisindeki bitleri etkilenir. Karşılaştırma sonucunda A ve B kaydedicilerindeki sayılar birbirine eşit ise Z biti "1" aksi hâlde "0" olur. B kaydedicisindeki sayı A kaydedicisindeki sayıdan büyük ise N biti "1" değilse "0" olur.

Tablo 14.5 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	B6 01 00		LDAA \$0100	A kaydedicisine (0100) ₁₆ 'deki veriyi yükle
0003	F6 01 01		LDAB \$0101	B kaydedicisine (0101) ₁₆ 'deki veriyi yükle
0006	11		CBA	A ve B Kaydedicileri karşılaştırılır. Kaydedicilerdeki içerik değişmez. Durum kodu kaydedicisindeki N, Z, V, C bitleri etkilenir.
0007	3F		SWI	Programı bitir.

14.1.6 Eğer Z Biti "0" ise Dallan

Amaç; durum kodu kaydedicisindeki sıfır "Z" bitinin bir önceki komuttan dolayı "1" olması durumunda, programın başka bir satıra dallanmasını sağlamak.

Açıklama: SBA komutu ile A kaydedicisi veriden B kaydedicisindeki veri çıkarılır. Çıkan sonuç tekrar A'ya kaydedilir. Eğer A=(00000000)₁₆, olduğunda yani A kaydedicisi sıfırlandığında Z biti'de "1" olur. Hemen sonraki bağıl dallanma komutu, Z=1 iken belirtilen etiketin olduğu satıra dallanır. Bu program kodlarına göre A ve B'deki değerler birbirine eşit ise SON etiketine dallanacak, değilse program normal akışında devam edecektir.

Tablo 14.6 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	10	FARK:	SBA	A-B; A'daki değerden B çıkarılır. Çıkan sonuç A'ya yüklenir. Aynı zamanda Durum kodu bitleri de etkilenir. (N, Z, V, C) A=B ise çıkan sonuç sıfır olacağından Z=1 olacaktır.
0001	27 06		BEQ SON	Z biti kontrol edilir. Z=1 ise SON'a dallanır.
0003	5C		INCB	B'yi bir artırır. B=B+1
0004	F7 01 00		STAB \$0100	B'yi (0100) ₁₆ adresine yükle.
0007	20 F7		BRA FARK	Şartsız "FARK" etiketine dallanır.
0008	3F	SON:	SWI	Programı bitir.

14.1.7 Alt Programa Dallanma ve Alt Programdan Geri Dönme (JSR ve RTS Komutları)

Amaç; ana programdan alt programa dallanma, alt programda işin bitmesi ile birlikte ana programa kaldığı yerden devam etme. Bu komutlar ile alt programa gidip-gelme söz konusudur.

Açıklama: Bu komuttan sonra program çalışmasını belirtilen adresteki alt programa yönlendirir. JSR komutunun JMP komutundan farkı, ana programdan JSR komutu bir alt programa dallanıldığında, alt programda işin bitmesinden sonra ana programa tekrar dönebilmek için RTS komutunun kullanılmasıdır. JMP komutunda geri dönüşü sağlayan bit komutu yoktur. JSR komutu kullanıldığında, mikroişlemci geri dönüş adresini yığına atar. RTS komutu ile birlikte geri dönüş adresi yığından alınarak kullanılır.

Tablo 14.7 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	01		NOP	Hiçbir işlem yapmaz, PC bir artar.
0001	BD 05 00		JSR \$0500	Program çalışmasına (0500) ₁₆ adresindeki alt programdan devam eder.
0004	4F		CLRA	Alt program dönüşünde çalışmasına bu adresten devam edecektir.
...	...	...	...	...
			ORG \$0500	Bundan sonraki program kodları (0500) ₁₆ adresinden itibaren başlar (pseudo komutu).
0500	01		NOP	Hiçbir işlem yapmaz, PC bir artar.
0501	01		NOP	Hiçbir işlem yapmaz, PC bir artar.
0502	39		RTS	Ana programda kaldığı (0004) ₁₆ adresine geri döner.

14.1.8 Herhangi Bir Adrese Dallanma (JMP Komutu)

Amaç; programın akışını başka bir adrese yönlendirebilmek için JMP komutu kullanılabilir.

Açıklama: JMP komutu ile programın akışı bir anda başka bir adrese yönlendirilebilir. Bu komut hiçbir şarta bakmaksızın program sayıcının içeriğini (0500)₁₆ ile yükler. Komutun işleyişinden sonra program (0500)₁₆ adresinden itibaren çalışmasına devam eder.

Tablo 14.8 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

H. Adr.	İş kodu	Etiket	Assembly komutu	Açıklama
0000	01		NOP	Hiçbir işlem yapmaz, PC bir artar.
0001	7E 05 00		JMP \$0500	PC (0500) ₁₆ adresine dallanır.
...	...	...	...	...
0500	01		NOP	İşlem yapma.
0501	3F		SWI	Programı bitir.

14.1.9 Hafızadaki Veriyi Sola Kaydırma

Amaç; hafızanın 40H adresindeki veriyi A kaydedicisine alarak bir bit sola kaydırmak ve sonucu tekrar 40H adresine yazmak.

Açıklama: I8085 mikroişlemci komutları kullanılarak yazılan programda hafızanın 40H adresindeki veri önce A kaydedicisine yüklenir. RLC komutu ile bir bit sola kaydırılır. A'daki veri tekrar 40H adresine STA 40H komutu hafızaya yazılır.

Tablo 14.9 : Programın kaynak kodları (8085 microişlemcisi için)

H. Adr.	İş kodu	Etiket	Assembly komutu	Açıklama
0000	3A 40 00		LDA 40H	40H adresinden veriyi A kaydedicisine al
0003	07		RLC	Bir bit sola kaydırılır.
0004	32 40 00		STA 40H	A kaydedicisindeki veriyi 40H adresine yaz.
0007	3F		SWI	Programı bitir.

14.2 ARİTMETİK VERİLİ PROGRAM YAZIMI

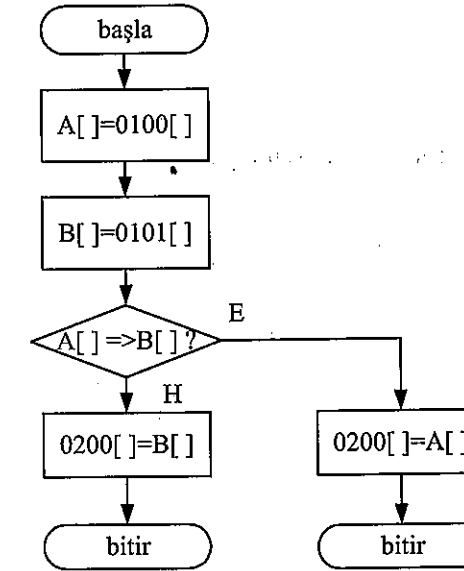
Akış Diyagramlarında Kullanılan Kısaltmaların Anlamları:

- ❖ A[]; A kaydedicisi (8 bitlik bir kaydedici)
- ❖ 0200[]; (0200)₁₆ Adresindeki hafıza hücresi
- ❖ 1F[]; (001F)₁₆ ve (1F)₁₆ adresindeki hafıza hücresi
- ❖ DE[]; DE kaydedici çifti (16 bitlik bir kaydedici)
- ❖ A[2F]; A kaydedicisinde bulunan değer (2F)₁₆ tir.
- ❖ A[] = A[] - 1; A kaydedicisindeki değeri bir azalt.
- ❖ A[] = #1F; A kaydedicisine 1F₁₆ değeri yüklenir.
- ❖ A[] = B[]; A kaydedicisine B kaydedicisindeki değer yüklenir.
- ❖ 0200[] = A[]; A'daki değer 0200[] adresine yüklenir.
- ❖ A[] = 0200[]; 0200[] adresindeki değer A[] kaydedicisine yüklenir.
- ❖ X[]; İndeks kaydedicisi (16 bitlik)
- ❖ X[01FF]; İndeks kaydedicisinde bulunan değer (01FF)₁₆ dir.
- ❖ X[] = X[] + 1; İndeks kaydedicisi 1 artırılır.
- ❖ X[] = 0200[]; İndeks kaydedicisine \$0200 ve \$0201 adreslerindeki değerler yüklenir.
- ❖ X[] = #S0200; İndeks kaydedicisine (0200)₁₆ yüklenir.
- ❖ A[] = #S00, X[]; İndeks kaydedicisinin işaret ettiği adreste bulunan veri A kaydedicisine yüklenir.
- ❖ #S00, X[] = A[]; A'daki veri indeks kaydedicisinin işaret ettiği adrese yüklenir.

14.2.1 İki Sayıdan Büyüğün Bulunması

Amaç; hafızanın (0100)₁₆ ve (0101)₁₆ adreslerindeki verilerden büyük olanı (0200)₁₆ adresine yazmak.

Akış diyagramı



Diyagram 14.1 : İki sayıdan büyüğünün bulunması

Açıklama: Hafızada bulunan pozitif sayılar (0...255) karşılaştırılarak büyük ve küçük sayılar belirlenebilir. Önce hafızanın (0100)₁₆ ve (0101)₁₆ adresindeki veriler A ve B kaydedicilerine yüklenir. CBA komutu ile A ve B'deki sayılar karşılaştırılır. Eğer A'daki sayı B'deki sayıdan daha büyük veya eşit ise BGE komutu programı BYK etiketine dalandırarak A kaydedicisindeki sayıyı STAA \$0200 komutu ile (0200)₁₆ adresine yazar ve program bitirilir. Şayet CBA komutundan sonra B'nin daha büyük olduğu sonucu çıkarsa BGE komutu ile dallanma gerçekleşmez ve hemen alttaki STAB \$0200 komutu işletilerek B kaydedicisindeki veri (0200)₁₆ adresine yazılır ve program bitirilir.

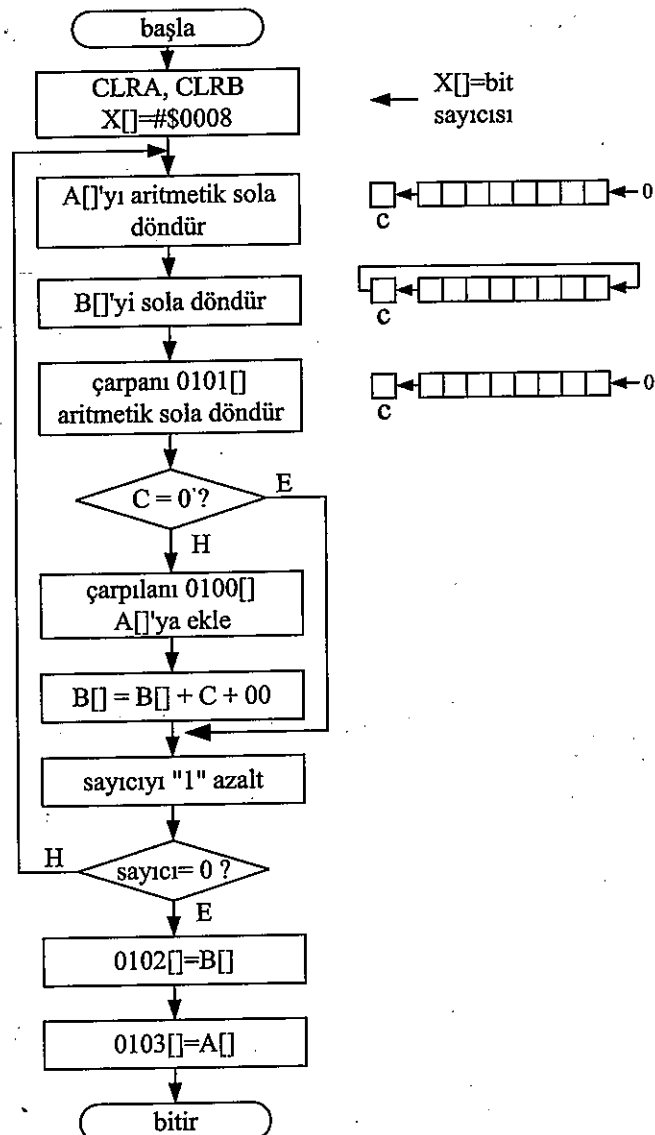
Tablo 14.10 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	B6 01 00		LDAA \$0100	(0100) ₁₆ 'deki sayı A 'ya yüklenir.
0003	F6 01 01		LDAB \$0101	(0101) ₁₆ 'deki sayı B 'ye yüklenir.
0006	11		CBA	(A-B) A ile B'deki sayılar karşılaştırılır.
0007	2C 04		BGE BYK	CBA komutunun sonucunda CCR'nin Z ve N bitlerine göre dallanma gerçekleşir. A'nın B'den büyük veya eşit olması durumunda BYK etiketine dallanır.
0009	F7 02 00		STAB \$0200	B büyük olduğundan (0200) ₁₆ 'e B'deki yazılır.
000C	3F		SWI	Program bitirilir.
000D	B7 02 00	BYK:	STAA \$0200	A Büyük olduğundan (0200) ₁₆ 'e A'daki yazılır.
0010	3F		SWI	Program bitirilir.

14.2.2 8 Bitlik İki Sayının Çarpımı

Amaç; birçok mikroişlemcide bulunmayan çarpma komutunun yerine kullanılabilecek bir çarpma programının hazırlanması. Çarpılan sayı hafızanın (0100)₁₆, çarpan sayı (0101)₁₆ adresine yazılmalıdır. Program çarpımın sonucunun yüksek değerlikli byte'ını (0102)₁₆, düşük değerlikli byte'ını (0103)₁₆ adreslerine yazacaktır.

Akış diyagramı



Diyagram 14.2 : 8 bitlik çarpma algoritması

Açıklama: A ve B kaydedicileri çarpmanın sonucunu saklayacakları için başlangıçta sıfırlanmıştır. X kaydedicisi çarpmanın tüm bitlerini test etmek için $(0008)_{16}$ ile yüklenmiştir. Program başında kullanılan ASLA ve ROLB komutlarının ilk seferde bir etkisi görülmemesine karşın daha sonraları her döngüde A ve B kaydedicilerini döndürerek çarpım sonucunun düşük ve yüksek değerlikli kısımlarının elde edilmesini sağlamaktadır. ASL \$0101 komutu ile $(0101)_{16}$ adresindeki çarpan "1" bit sola döndürülür. Eğer elde biti "1" ise $(0100)_{16}$ adresindeki çarpılan ile A kaydedicisindeki değer toplanır, B kaydedicisi ile elde biti toplanır, indeks kaydedici "1" azaltılır ve $X=0$ değilse SHFT etiketine dallanılır. Elde biti "0" ise DECR etiketine dallanılır burada indeks kaydedici "1" azaltılır $X=0$ değilse SHFT etiketine dallanılır. Bu işlem $X=0$

oluncaya kadar devam eder. $X=0$ olduğunda çarpımın yüksek değerlikli byte'ı için B kaydedicisi $(0102)_{16}$ adresine, düşük değerlikli kısmı için A kaydedicisi $(0103)_{16}$ adreslerine yüklenir.

Tablo 14.11 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	4F		CLRA	A kaydedicisi temizlenir.
0001	5F		CLRB	B kaydedicisi temizlenir.
0002	CE 00 08		LDX #0008	İndeks kaydediciye #0008 verisi yüklenir.
0005	48	SHFT:	ASLA	B'yi aritmetik sola kaydır.
0006	59		ROLB	A'yı sola döndür.
0007	78 01 01		ASL \$0101	(0100) ₁₆ adresindeki veriyi aritmetik sola kaydır.
000A	24 05		BCC DECR	C=0 ise DECR etiketine dallan.
000C	BB 01 00		ADDA \$0100	B ile (0101) ₁₆ adresindeki değer ile topla.
000F	C9 00		ADCB #\$00	A+00+C (Elde biti) toplar.
0011	09	DECR:	DEX	İndeks kaydediciyi 1 azalt.
0012	26 F1		BNE SHFT	X=0 değilse (Z=0) ise dallan. (X=0 olduğunda Z=1 olur.)
0014	F7 01 02		STAB \$0102	A'daki değeri (0102) ₁₆ adresine yükle.
0017	B7 01 03		STAA \$0103	B'deki değeri (0103) ₁₆ adresine yükle.
001A	3F		SWI	Programı bitir.

İkilik Tabandaki Sayıların Çarpımı:

8 bitlik iki sayının çarpımını, çarpılan sayının, çarpan sayı kadar kendisi ile toplanması sonucunda elde edebiliriz. Assembly komutları ile hazırlanan çarpma programı bu temel mantık üzerine kurulmuştur. Örnek olarak (00101010)₂ ile (00101100)₂ sayılarının çarpımında, “çarpandaki” “1” sayısı kadar “çarpılanın” her seferinde bir basamak kaydırılarak toplandığı aşağıda gösterilmiştir. Çarpımdaki sayıların 1'er byte'lık olmasına rağmen çarpımın 2 byte'lık olması olağandır.

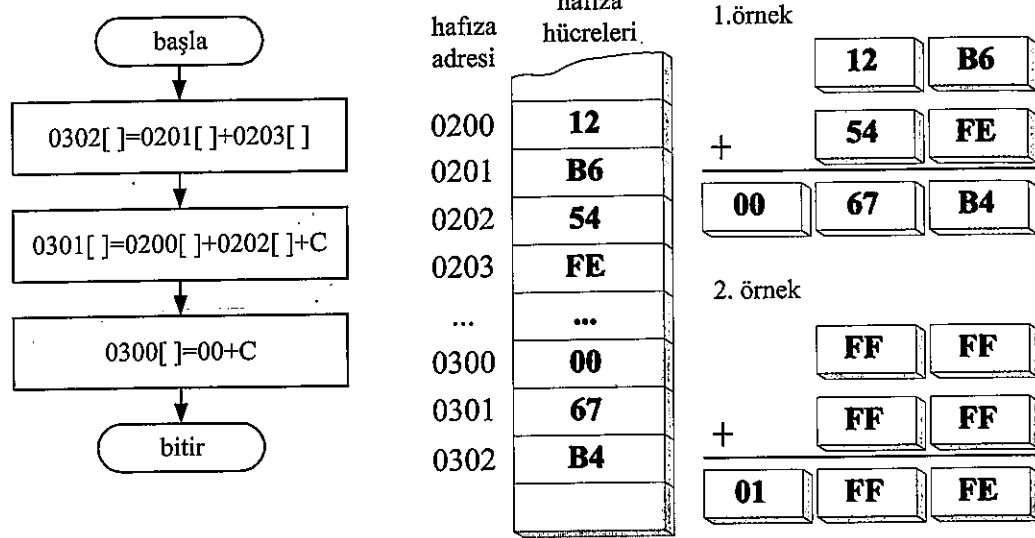
[illegible]

Çarpımın düşük değerlikli bitinden başlayarak bitler tek tek tetes edilir. Eğer bit “0” ise, çarpım “1” bit sola kaydırılır, eğer “1” ise çarpılan çarpıma eklenerek çarpım “1” bit sola kaydırılır. Çarpımın 8 biti de test edilinceye kadar devam edilir.

14.2.3 8 Bitlik Mikroİşlemcilerde 16 Bitlik Sayıların Toplanması

Amaç; ardışık olarak (0200)₁₆ ve (0201)₁₆ adreslerinde bulunan 16 bitlik bir sayı ile (0202)₁₆ ve (0203)₁₆ adreslerinde bulunan 16 bitlik bir sayının toplamından elde edilen sonucun (0300)₁₆, (0301)₁₆ ve (0302)₁₆ adreslerine yüklenmesi.

Akış diyagramı



Diyagram 14.3 : 16 bitlik sayıların toplamı

Açıklama: 8 bitlik mikrobilgisayarlar büyüklüğü en fazla 255 olan sayıları hafızalarında tutabilirler. 255'den daha büyük sayılar ile işlem yapabilmek için hafıza çiftleri kullanmak gerekir. Bu örnekte 16 bitlik iki sayıdan birisi hafızada art arda (0200)₁₆ ve (0201)₁₆ adreslerinde diğeri ise (0202)₁₆ ve (0203)₁₆ adreslerinde yüklü bulunmaktadır. Bu 16 bitlik iki sayının toplamından elde edilecek sonucun alabileceği en büyük değer hafızada en fazla 3 byte yer tutacağından toplamın sonucu için (0300)₁₆, (0301)₁₆, (0302)₁₆ adresleri kullanılmıştır.

Önce sayıların en düşük değerlikli kısımları ((0201)₁₆ ve (0203)₁₆) toplanarak sonucun en düşük değerlikli kısmı elde edilerek hafızanın (0302)₁₆ adresine yazılır. Daha sonra sayıların yüksek değerli kısmını elde etmek için bir önceki toplamdan oluşan elde varsa bununla birlikte üçü toplanır (0200)₁₆, (0202)₁₆ ve eğer var ise bir önceki toplamdan oluşan elde) ve sonuç (0301)₁₆ adresine yüklenir. Eğer sayıların yüksek değerlikli kısımları toplanırken elde oluşmuş ise bunu da hesaba ilâve etmek için önce A kaydedicisinde oluşan eldeyi etkilemediğinden (00)₁₆ yüklenir ve bir önceki işlemde oluşan eldeyle toplanır (ADCA komutu ile) ve sonuç (0300)₁₆ adresine yazılır.

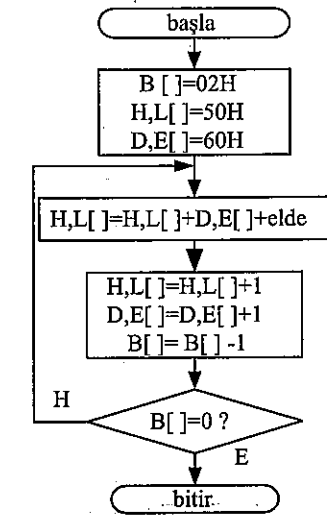
Tablo 14.12 : Programın kaynak kodları (MC6802 mikroİşlemcisi için)

El. Adr.	İş. Kodu	Etiket	Assembly komutu	Açıklama
0000	B6 02 01		LDAA \$0201	A'yı (0201) ₁₆ adresindeki değer ile yükle
0003	BB 02 03		ADDA \$0203	A'yı (0203) ₁₆ adresindeki değer ile topla
0006	B7 03 02		STAA \$0302	A'yı (0302) ₁₆ adresine sakla
0009	B6 02 00		LDAA \$0200	A'yı (0200) ₁₆ adresindeki değer ile yükle
000C	B9 02 02		ADCA \$0202	A'yı (0202) ₁₆ adresindeki değer ve C ile topla
000F	B7 03 01		STAA \$0301	A'yı (0301) ₁₆ adresine sakla
0012	86 00		LDAA #\$00	A'yı (00) ₁₆ ile yükle, bu komut bir önceki işlemde kalan C'yi etkilemez.
0014	89 00		ADCA #\$00	A'yı (00) ₁₆ ve C (elde) biti ile topla
0016	B7 03 00		STAA \$0300	A'yı (0300) ₁₆ adresinde sakla
0019	3F		SWI	Programın sonu

14.2.4 Dört Haneli İki BCD Sayının Toplamı

Amaç; hafızanın (50)₁₆ ve (51)₁₆ adreslerinde bulunan BCD kodlanmış bir sayı ile (60)₁₆ ve (61)₁₆ adreslerinde bulunan BCD kodlanmış diğeri sayıyı toplayıp sonucu (50)₁₆ ve (51)₁₆ adreslerine yazan programın yapmak. (30)₁₆ adresine program çalıştırılmadan önce toplanacak sayıların adeti yüklenmelidir. Örneğimize uygun olarak (30)₁₆ adresine 02 yüklenmesi olmalıdır. Program Intel 8085 komut satına uygun hazırlanacaktır.

Akış diyagramı



Diyagram 14.4 : Dört haneli 2 BCD sayının toplamın bulunması

Açıklama: Toplanacak sayıların adeti A kaydedicisine yüklenikten sonra bu sayıların başlangıç adresleride H,L ve D,E kaydedici çiftlerine yüklenir. Ele bayrağı temizlenir. A kaydedicisine D,E'nin gösterdiği adresten veri yüklenir. ADC M komutu ile A'daki veriye H,L'nin gösterdiği adresten alınan veri ve elde biti eklenir. Yapılan toplama işleminden sonra sonuç DAA komutu ile onluk sayı düzenine getirilir. A'daki sonuç H,L'nin işaret ettiği adrese yüklenir. H,L ve D,E kaydedicileri birer arttırılarak bir sonraki hanelerin toplamına geçilir. Bu işlem B kaydedicisi sıfır oluncaya kadar devam eder.

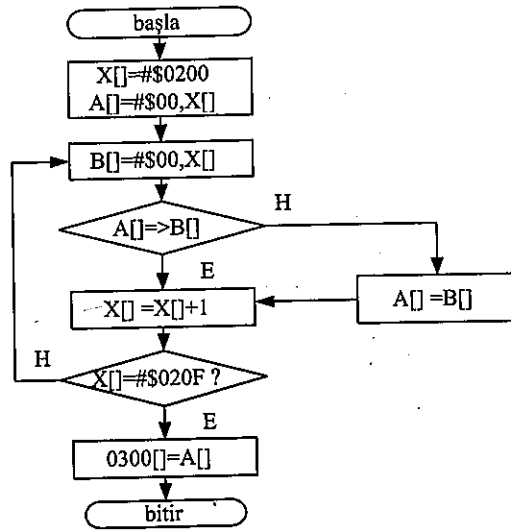
Tablo 14.13 : Programın kaynak kodları (Intel 8085 mikroİşlemcisi için)

Hafıza adresi	İş. kodu	Etiket	Assembly komutu	Açıklama
0000	3A 30 00		LDA 30H	Toplam sayısını 30 adresinden A'ya al.
0003	47		MOV B,A	A'daki değeri B'ye yükle.
0004	21 51 00		LXI H,50H	H,L kaydedicisine 50H değerini yükle.
0007	11 61 00		LXI D,60H	D,E kaydedicisine 60H değerini yükle.
000A	A7		ANA A	A'ya AND işlemi yaparak eleği temizle.
000B	1A	TEKRAR:	LDAX D	D,E kaydedicisinin gösterdiği adresten A'ya veri yükle.
000C	8E		ADC M	H,L'nin gösterdiği adresdeki veriyi elde biti ile birlikte A'ya ekle.
000D	27		DAA	A'yı ondalık tabana ayarla
000E	77		MOV M,A	A'yı H,L'nin gösterdiği adrese yükle
000F	13		INX D	D,E kaydedicisini bir arttır.
0010	23		INX H	H,L kaydedicisini bir arttır.
0011	05		DCR B	B'yi bir azalt
0012	C2 0B 00		JNZ TEKRAR	Sıfır değilse tekrar etiketine dallan.
0015	C3 15 00	SON:	JMP SON	

14.2.5 Bir Sayı Dizisinde En Büyük Sayının Bulunması

Amaç; hafızanın (0200)₁₆ ve (020F)₁₆ adresleri arasında kalan bölgede bulunan pozitif sayılar içerisindeki en büyük sayıyı bularak (0300)₁₆ adresine yazmak.

Akış diyagramı



Diyagram 14.5 : Bir dizi içerisindeki en büyük sayının bulunması

Açıklama: Program bir sayı dizisi içerisindeki en büyük sayıyı bulur. Öncelikle dizinin başlangıç adresi indeks kaydedicisine yüklenir. A kaydedicisi büyük sayıyı tutan kaydedici olarak kullanılmıştır. B kaydedicisi ise dizi içerisindeki sayıları alarak karşılaştırma yapmada kullanılmıştır. Karşılaştırma A ile B kaydedicileri arasında yapılır. Eğer B'deki sayı A'dakinden büyükse B'deki değer A'ya aktarılır değilse bir sonraki dizi elemanına geçilir. Dizi sonunun gelip gelmediği programın (000C)₁₆ adresindeki karşılaştırma komutu ile yapılmaktadır. Dizi sonuna gelindiğinde dizideki tüm sayılar birbirleri ile karşılaştırılmış olacaktır ve bu karşılaştırmaların sonucunda en büyük sayı A' kaydedicisinde olduğu için A kaydedicisindeki değer (0300)₁₆ adresine kopyalanarak program bitirilir.

Tablo 14.14 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

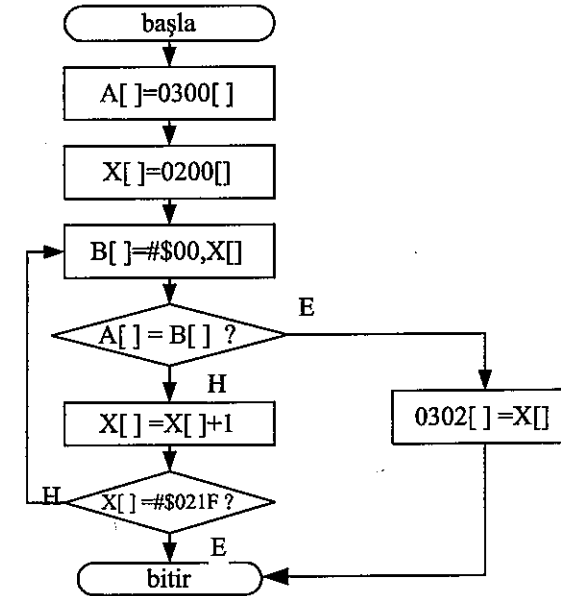
Hafıza Adresi	İş Kodu	Etiket	Assembly Komutu	Açıklama
0000	CE 02 00		LDX #0200	Dizinin başlangıç adresini X'e yükle.
0003	A6 00		LDAA #00,X	İlk değeri A'ya kaydet.
0005	E6 00	ARA:	LDAB #00,X	İlk değeri B'ye de kaydet.
0007	11		CBA	A ile B kaydedicilerini karşılaştı.
0008	24 01		BCC DVM	A=>B'den ise dallan (C=0 iken dallanır.)
000A	17		TBA	A=B B'yi büyük sayıyı tutan A'ya aktar.
000B	08	DVM:	INX	Dizi kaydedicisini bir arttır.
000C	8C 02 0F		CPX #020F	Dizi sonunu kontrol et.
000F	26 F4		BNE ARA	Dizi sonu mu ? Hayır ise aramayı tekrarla.
0011	B7 03 00		STAA \$0300	Evet ise A'yı (0300) ₁₆ adresine kopyala.
0014	3F		SWI	Programı bitir.

14.3 KARAKTER VERİLİ PROGRAM YAZIMI

14.3.1 Belirli Bir Hafıza Bölgesinde Bir Byte Veri Arama Programı

Amaç; hafızanın (0200)₁₆ ve (021F)₁₆ adresleri arasındaki bölgede belirlenen bir byte'lık veriyi aramak. Program, aranan veri bulunduğunda verinin adresini (0301)₁₆ ve (0302)₁₆ adreslerine yükleyecektir. Program çalıştırılmadan önce aranacak veri (0300)₁₆ adresine girilmelidir.

Akış diyagramı



Diyagram 14.6 : Bir dizi içerisindeki bir byte veriyi arama

Açıklama: Program, bir sayı dizisi içerisinde bir byte veri arar. Aranacak veri hafızanın (0300)₁₆ adresine girilir. Program hafızanın (0200)₁₆ ile (021F)₁₆ adresleri arasında arama yapacağı için dizinin başlangıç adresi indeks kaydedicisine girilir. B kaydedicisine, indeks kaydedicisinin işaret ettiği adresteki veri yüklenir. A kaydedicisindeki veri ile B kaydedicisindeki veri karşılaştırılır. Eğer aranan (A'daki veri) veri ile hafızadan alınan veri (B'deki veri) eşit ise program indeks kaydedicisinin işaret ettiği adresin yüksek değerlikli kısmını (0301)₁₆ adresine düşük değerlikli kısmını ise (0302)₁₆ adresine yükleyerek program sonlandırılır. Aranacak değeri bulunmadığı takdirde dizinin son adresi (0301)₁₆ ve (0302)₁₆ adreslerine yüklenerek program sona erer.

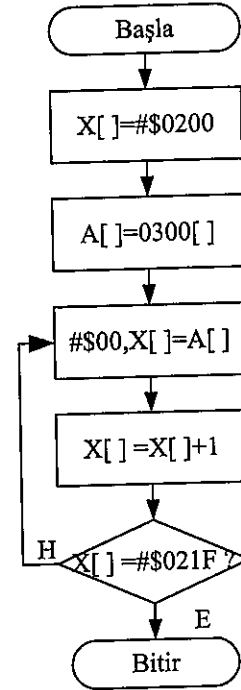
Tablo 14.15 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza Adresi	İş Kodu	Etiket	Assembly Komutu	Açıklama
0000	B6 03 00		LDAA \$0300	A'ya (0300) ₁₆ adresindeki aranan veriyi kaydet.
0003	CE 02 00		LDX #0200	Dizinin başlangıç adresini X'e yükle
0005	E6 00	ARA:	LDAB #00,X	İlk değeri B'ye kaydet.
0007	11		CBA	A ile B kaydedicilerini karşılaştı.
0008	27 06		BEQ SON	A=B ise dallan (Z=1 iken dallanır.)
000A	08		INX	Dizi kaydedicisini bir arttır.
000C	8C 02 1F		CPX #021F	Dizi sonu mu?
000F	26 F5		BNE ARA	Dizi sonu değilse aramayı tekrarla
0011	FF 03 01	SON:	STX \$0301	X'deki adresi (0301) ₁₆ adresine kopyala.
0014	3F		SWI	Programı bitir.

14.3.2 Belirli Bir Hafıza Bölgesini Bir Sabit İle Doldurma Programı

Amaç; hafızanın (0200)₁₆ ve (021F)₁₆ adresleri arasındaki bölgeyi belirlenen bir sabit değer ile doldurmak. Hafızaya doldurulacak veri hafızanın (0300)₁₆ adresine yazılmalıdır.

Akış diyagramı



Diyagram 14.7 : Bir diziyi belirli bir sabit veri ile doldurma

Açıklama: Hafızaya doldurulacak veri (0300)₁₆ adresinden alınarak A kaydedicisine yüklenir. İndeks kaydediciye doldurulacak bölgenin başlangıç adresi yüklenir. A'daki veri indeks kaydedicisinin işaret ettiği ilk hafıza hücresine yazılır ve indeks kaydedici bir sonraki hafıza hücresini göstermesi için bir artırılır. Her arttırmanın sonunda dizi sonuna gelinip gelinmediğine bakılır. Dizi sonuna gelinceye kadar A'daki veri belirlenen hafıza aralığına yazılır. Dizin son adresine (021F)₁₆ gelindiğinde program bitirilir.

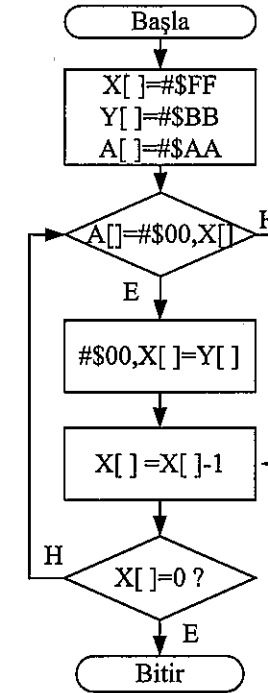
Tablo 14.16 : Programın kaynak kodları (MC6802 mikro işlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	CE 02 00		LDX \$0200	Dizin başlangıç adresini X'e yükle
0003	B6 03 00		LDAA \$0300	Yüklenilecek değeri (0300) ₁₆ adresinden A'ya yükle.
0006	A7 00	YKL:	STAA #\$00,X	A'daki veriyi İndeksin işaret ettiği yere yükle.
0008	08		INX	X kaydedicisini bir arttır.
0009	8C 02 1F		CPX \$021F	X (021F) ₁₆ değerine eşit mi ? (eşitse Z=1 olur.)
000C	26 F8		BNE YKL	Z=0 olduğu sürece dallanır.
000E	3F		SWI	Programı bitir.

14.3.3 Dizi İçerisindeki Belirli Bir Veriyi Başka Bir Veri İle Değiştirme

Amaç; Hafızanın \$0000 ile \$00FF adres aralığında bulunan bir dizi içerisindeki "AA" verisini "BB" ile yer değiştirmek. Bu program Mos. Tec. mikro işlemcisinin komut seti ile hazırlanacak.

Akış diyagramı



Diyagram 14.8 : Dizideki bir veriyi başka bir veri ile yer değiştirme

Açıklama: Dizinin toplam boyutu FF olarak X kaydedicisine yüklenir. Aranacak veri (AA)₁₆ A kaydedicisine değiştirilecek veri (BB)₁₆ Y kaydedicisine yüklenir. Sonra X indeks kaydedicisinin göstermiş olduğu adresteki veri ile A kaydedicisindeki veri CMP komutu ile karşılaştırılır. Eğer veriler birbirine eşitse Z bayrağı "1" olur. BNE komutu Z bayrağı "0" olduğunda dallanacağından bir sonraki komuta geçilir. STY komutu ile Y kaydedicisindeki değer X indeks kaydedicisinin işaret ettiği adrese yani karşılaştırma yapılan adrese (BB)₁₆ verisini yükleyerek bir sonraki komuta geçilir. Eğer karşılaştırma işlemi sonucunda Z bayrağı "0" olsaydı DEX komutu ile indeks kaydedicisi bir azaltılarak aramaya devam edilirdi. Bu işlem X=0 oluncaya kadar devam eder. Böylece dizi içerisindeki tüm (AA)₁₆ verileri (BB)₁₆ yapılmış olur.

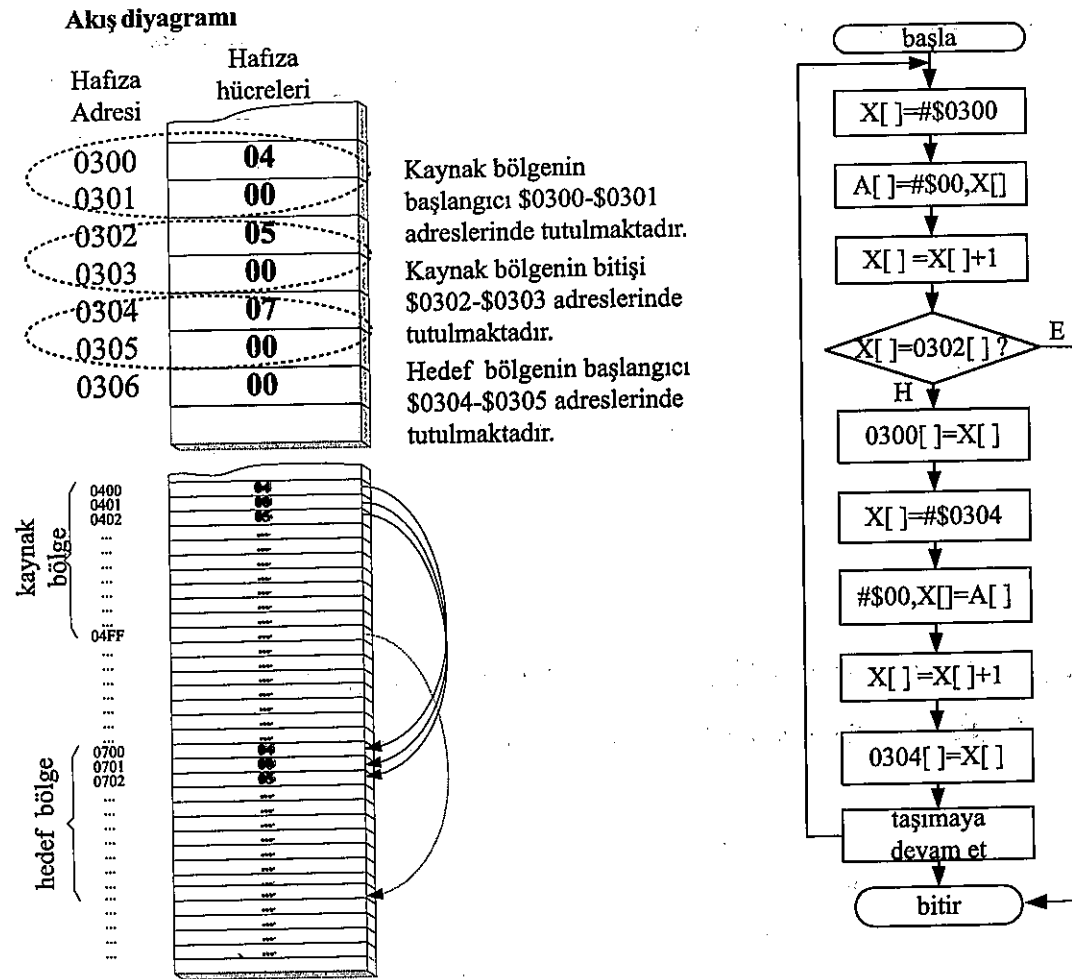
Tablo 14.17 : Programın kaynak kodları (Mos. Tec. 6502 mikro işlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0100	A6 FF		LDX \$FF	X kaydedicisine (FF) ₁₆ yükler.
0102	A0 BB		LDY \$BB	Y kaydedicisine (BB) ₁₆ yükler.
0104	A9 00		LDA \$AA	A kaydedicisine (AA) ₁₆ yükler.
0106	D5 00	TARA	CMP #\$00,X	İndeks kaydedicisinin gösterdiği adresteki veri ile A kaydedicisindeki veriyi karşılaştırılır.
0108	D0 02		BNE ARA	
010A	94 40		STY #\$00,X	
010B	CA	ARA	DEX	
010D	D0 F8		BNE TARA	

14.3.4 Belirli Bir Hafıza Aralığını Başka Bir Hafıza Bölgesine Taşıma

Amaç; hafızada bulunan bir veri bloğunun hafızadaki başka bir bölgeye taşınması. Taşınacak veri bloğuna "kaynak bölge", verinin taşınacağı yere "hedef bölge" denir. Program çalıştırılmadan önce kaynak bölgenin başlangıç adresi (0300)₁₆ ve (0301)₁₆, bitiş adresi (0302)₁₆ ve (0303)₁₆ adreslerine yazılmalıdır. Aynı şekilde hedef bölgenin başlangıç adresini de (0304)₁₆, (0305)₁₆ adreslerine yazmamız gerekmektedir.

Kullandığımız mikroişlemcinin kaydedicileri yetmediği durumlarda değişik bilgileri saklamak için hafıza hücrelerini kaydedici gibi kullanabiliriz. Bu programda da kaynak bölgenin başlangıç ve bitiş adresi ile hedef bölgenin başlangıç adresleri için hafıza hücreleri kullanılmıştır. Hafızada adres bilgileri tutulacağı için her veri için ardışık 2'şer byte'lık alan ayrılmıştır.



Diyagram 14.9 : Bir dizi veriyi başka bir bölgeye taşımak

Açıklama: Taşınacak verinin başlangıcını (0300)₁₆, (0301)₁₆ adreslerine (0400)₁₆ olarak bitiş adresini de (0302)₁₆, (0303)₁₆ adreslerine (0500)₁₆ olarak girmekle kaynak bölgenin başlangıç ve bitiş adreslerini belirtmiş olmaktadır. Hedef bölgenin başlangıç adresini ise hafızanın (0304)₁₆, (0305)₁₆ adreslerine (0700)₁₆ olarak girmekle tüm başlangıç değerlerini tamamlamış oluruz. Burada amaç hafızanın (0400)₁₆ ile (0500)₁₆ adresleri arasındaki bir dizi veriyi (0700)₁₆ adresinden itibaren taşımak. Bu başlangıç değerleri girildikten sonra program çalıştırılabilir.

Öncelikle kaynak bölgenin başlangıç adresi (0300)₁₆ adresinden indeks kaydediciye yüklenir. İndeks kaydedici iki byte'lık bir kaydedici olduğundan hafızanın (0300)₁₆ ve (0301)₁₆ adreslerindeki verileri

art arda alır. İndeks kaydedicinin işaret ettiği adresteki veri A kaydedicisine kopyalanır. İndeks kaydedici bir artırılarak böylelikle bir sonraki kaynak veriyi işaret etmesi sağlanır. Kaynak bölgenin sonuna gelinip gelinmediği (0302)₁₆, (0303)₁₆ adreslerinde yüklü bulunan kaynak bölgenin son adresi ile karşılaştırılır. Eğer kaynak bölgenin sonuna gelinmedi ise indeks kaydedicideki adres değeri (0300)₁₆ adresinde saklanır. Şimdiye kadar A kaydedicisine kaynak bölgeden veri alınmış oldu daha sonra bu veriyi hedef bölgeye taşımak için indeks kaydedicisine hedef bölgenin adresi (0304)₁₆'den alınır. A indeks kaydedicinin işaret ettiği adrese yüklenir. İndeks kaydedici bir artırılarak tekrar (0304)₁₆ adresinde saklanır. BRA komutu ile taşınan veri bloğunun sonuna ginceye kadar döngü tekrarlanır.

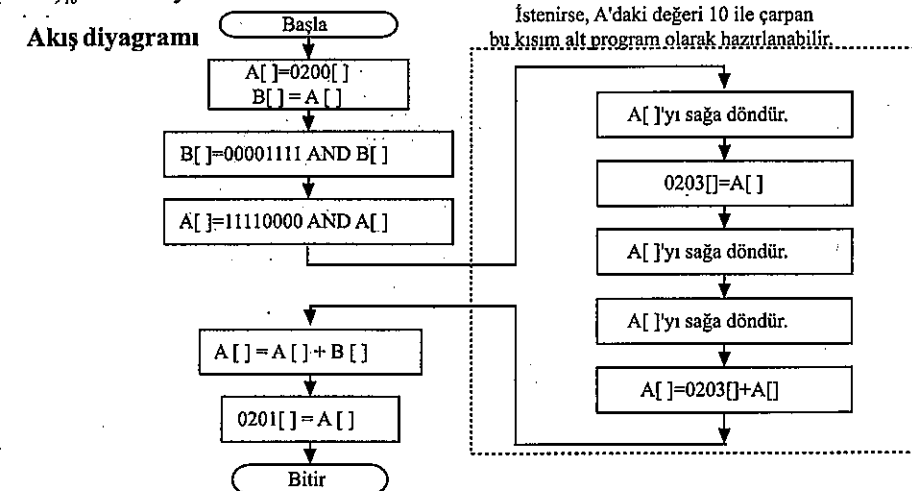
Tablo 14.18 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza Adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	FE 03 00	TASI:	LDX \$0300	Kaynak bölgenin başlangıç adresi (0300) ₁₆ 'den alınır.
0003	A6 00		LDAA #\$00,X	X'in işaret ettiği adresteki veri A'ya yüklenir.
0005	08		INX	X kaydedicisi bir artırılır.
0006	BC 03 02		CPX \$0302	Kaynak verinin sonu olup olmadığı kontrol edilir.
0009	27 0E		BEQ SON	Eğer blok sonu değilse bir sonraki komutla devam edilir. Bloğun sonuna gelinmiş ise SON etiketine dallanılır.
000B	FF 03 00		STX \$0300	İndeks kaydedicideki değer (0300) ₁₆ adresinde saklanır.
000E	FE 03 04		LDX \$0304	İndeks'e hedef bölgenin adresi yüklenir.
0011	A7 00		STAA #\$00,X	Veri X'in işaret ettiği adrese yüklenir.
0013	08		INX	İndeks kaydedici bir artırılır.
0014	FF 03 04		STX \$0304	Hedef bölgenin adresi (0304) ₁₆ adresinde saklanır.
0017	20 E7		BRA TASI	Taşımaya devam etmek için dallanır.
0019	3F	SON:	SWI	Taşıma işlemi tamamlandığında program bitirilir.

14.4 KOD DÖNÜŞTÜRÜCÜ PROGRAM YAZIMI

14.4.1 BCD Kodlu Sayının İkilik Karşılığına Dönüşümü

Amaç; hafızanın (0200)₁₆ adresinde bulunan BCD kodlanmış bir sayının ikilik karşılığını bularak sonucu (0201)₁₆ adresine yazmak.



Diyagram 14.10 : BCD kodlanmış sayıyı onluk tabana dönüştürme



1. Onluk tabandaki sayının BCD kodu ile gösterilmesi:

Bir sayının BCD kodlamasını yapmak için sayıdaki geçen her rakamın (0,1,...,9) ikilik karşılığını yazarız. BCD kodlanmış bir sayı dörderli haneler halinde gruplandırılır. Bu parçalar BCD₁, BCD₂ diye gösterilebilir. Her parça BCD kodlanmış sayının bir hanesine karşılık gelmektedir. Örneğin (76)₁₀ sayısının rakamları ayrı ayrı BCD kodlayarak yazıldığında ikilik karşılığı görülmektedir.

(76)₁₀ sayısının BCD karşılığı; (7)_{BCD2}-(6)_{BCD1} → (0111)₂-(0110)₂

(76)₁₀ sayısının ikilik karşılığı; 76₁₀ → (0100 1100)₂'dir.

Sayının ikilik tabandaki karşılığı ile BCD kodlanmış karşılığı arasındaki farka dikkat ediniz. Onluk tabanda yazılan tüm sayılar BCD kodlama tekniği ile kodlanacağı zaman, sayıdaki her rakamın ayrı ayrı ikilik tabanda yazılması gerekir.

Onluk tabandaki (576)₁₀ sayısının ikilik karşılığı; (1001000000)₂

BCD kodlanmış (576)₁₀ sayısı; (0101 0111 0110)₂

76 → (70)₁₀ + (6)₁₀ → (01000110)₂ + (00000110)₂

76 → (7)_{BCD2}-(6)_{BCD1} → (0111000)₂ OR (00000110)₂

2. BCD kodu ile gösterilen sayının onluk tabanda gösterilmesi:

BCD kodlanmış bir sayı, sağdan başlanarak dörderli gruplara ayrılır. Örneğin (01110110)_{BCD} sayısı iki parçadan oluşmaktadır. Bunlar (0111)_{BCD2} ve (0110)_{BCD1}'dir. 76₁₀ sayısı onluk sistemde 70+6 şeklinde toplanarak elde edilir. BCD kodlanmış bir sayıdan onluk sayı karşılığını elde etmek için sıra ile onlar hanesindeki sayıyı "10" ile birler hanesindeki sayıyı "1" ile çarpıp bu sayıların toplanması gerekir. BCD kodlanmış bir sayıdan onluk sayıyı elde etmek için örnekte verilen sayıya göre, BCD₂ hanesindeki "7" rakamını eğer 10 ile çarpıp (7.10=70), BCD₁ hanesindeki 6 rakam ile topladığımızda 76₁₀ sayısını elde edebiliriz.

Özetle; BCD kodlanmış bir sayıdan onluk tabandaki sayıyı elde etmek için ...+100.BCD₂+10.BCD₁+BCD₀ şeklinde formüle edebiliriz.

Yöntem: Assembly programı hazırlarken iki haneli BCD kodlanmış bir sayının BCD₂ hanesini 10 ile çarpmak için önce sayının BCD₁ hanesi sıfırlanmalıdır. Bunu BCD kodlanmış sayı ile "11110000" sayısına AND işlemi uygulayarak yapabiliriz. Örneğin (01110110)₂ AND (11110000)₂ = (01110000)₂ olur. Artık bu sayıyı BCD kodlanmış bir sayı değil ikilik tabandaki bir sayı olarak değerlendirmemiz gerekir. Bundan sonra sağa döndürme komutu (ROR komutu) kullanarak BCD₂ sayının 10 ile çarpımını elde edebiliriz. Sayıyı bir kez sağa döndürdüğümüzde 2'ye bölünmüş oluruz. Sayıyı 2 kez daha sağa döndürdüğümüzde 8'e bölünmüş oluruz. Sayının 2'ye bölümü ile 8'e bölümünden elde edilen sonuçlar toplandığında BCD₂ hanesindeki sayıyı 10 ile çarpmış oluruz.

Aşağıda BCD kodlanmış (76)_{BCD} sayısının BCD₂ hanesindeki 7 rakamından sağa döndürme komutu kullanılarak 70 sayısının nasıl elde edildiği adım adım aşağıda gösterilmiştir.

1) BCD kodlanmış (76)_{BCD} sayısındaki BCD₁=6 hanesi sıfırlanır. (01110110)₂ AND (11110000)₂ = (01110000)₂

2) Geriye (70)_{BCD} sayısı kalır. (Bu sayı onluk tabanda 112'dir. Biz (01110000)_{BCD}=(70)_{BCD} bölme yolu ile 112 sayısından 70 sayısını elde edeceğiz.). Böylece (01110000)₂=112 BCD₂ hanesini 10 ile çarpmış oluruz.

3) Sayı bir kere sağa döndürülürse 2'ye bölünmüş olur. (00111000)₂=56

4) 56 sayısı iki kere daha sağa döndürülürse, "112" 8'e bölünmüş olur. (00001110)₂=14

Sayı bir kez sağa döndürülürse 8'e bölünmüş olur. 00001110₂=14₁₀

Sayının 2'ye bölümü ile 8'e bölümleri toplandığında 70 elde edilir. 56+14=70

Açıklama: Hafızanın (0200)₁₆ adresinde bulunan BCD kodlanmış bir sayı önce A kaydedicisine yüklenir. Bu sayının aynı TAB komutu ile B kaydedicisine de kopyalanır. BCD kodlanmış sayıdan onluk sayıyı elde etmek için kullandığımız formülde (10.BCD₂+BCD₁) 10.BCD₂'yi ve BCD₁'i ayrı ayrı elde etmemiz gerekecektir. Bunun için B kaydedicisindeki BCD kodlanmış sayının BCD₂ tarafı sıfırlanır. A kaydedicisindeki BCD kodlanmış sayının da BCD₁ tarafı sıfırlanır. A kaydedicisindeki BCD₂'den 10.BCD₂'yi elde etmek için sayı bir kez sağa döndürülerek ikiye bölünmüş olur. Bu sayı hafızanın (0203)₁₆ adresinde saklanır. A kaydedicisindeki değer 2 kez daha sağa döndürülerek sayının 8'e bölümü elde edilir. Sayının 2'ye bölümünden elde edilen değer ile (hafızanın (0203)₁₆ adresine saklanmıştı.) 8'e bölümünden elde edilen değer toplandığında BCD₂ hanesinin 10 ile çarpımı elde edilmiş olacaktır. Daha sonra B kaydedicisindeki değer ile bu değer toplanarak sonuç hafızanın (0201)₁₆ adresine yüklenmiştir.

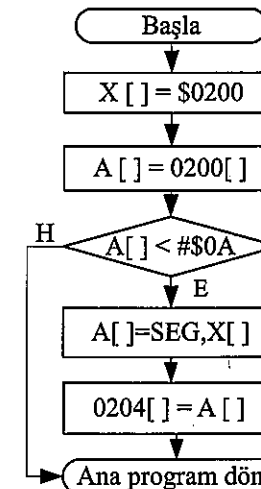
Tablo 14.19 : Programın kaynak kodları (MC6802 mikroişlemcisi için)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	B6 02 00		LDAA \$0200	BCD sayı 0200 adresinden A'ya yüklenir.
0003	16		TAB	A'daki değer B'ye kopyalanır.
0004	C4 0F		ANDB #\$0F	B'deki değer yüksek değerlikli kısmı sıfırlanır
0006	84 F0		ANDA #\$F0	A'deki değer düşük değerlikli kısmı sıfırlanır.
0008	46		RORA	A bir kez sağa döndürülür.
0009	B7 02 03		STAA \$0203	A'daki değer geçici olarak (0203) ₁₆ 'de saklanır.
000C	46		RORA	A bir kez sağa döndürülür.
000D	46		RORA	A bir kez daha sağa döndürülür.
000E	BB 02 33		ADDA \$0203	(0203) ₁₆ 'deki değer ile A'daki değer toplanır.
0011	1B		ABA	A'daki değer ile B'deki değer toplanır.
0012	B7 02 01		STAA \$0201	Elde edilen sonuç (0201) ₁₆ adresine yüklenir.
0015	3F		SWI	Program bitirilir.

14.4.2 BCD Kodlu Sayının 7 Segment Display İçin İkili Koda Dönüştürülmesi

Amaç; hafızanın (0201)₁₆ adresinde bulunan BCD kodlanmış sayının, 7 segment displayler için gereken ikilik koda dönüştürülmesi. Dönüştürülen kod hafızanın (0204)₁₆ adresinde saklanacaktır. (Dönüştürülecek BCD kodlu veri tek haneli olmalıdır. Yani 0...9 arasındaki rakamlar 0201 adresinde bulunmalıdır. Program bu şartlarda doğru sonuç üretir.)

Akış diyagramı



Diyagram 14.11 : BCD 7 segment dönüştürme algoritması

Tablo 14.20 : Programın kaynak kodları (MC6802)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
			ORG \$0100	Bundan sonraki program kodları 0100'den adresinden itibaren başlar (pseudo komutu).
0100	FE 02 00		LDX \$0200	BCD kodlu veriyi 0200 adresinden alır.
0103	B6 02 01		LDAA \$0201	BCD kodlu veriyi test için A'ya da alır.
0106	81 0A		CMPA #\$0A	A'daki BCD kodlu veri (0A) ₁₆ 'dan küçük mü?
0108	24 05		BCC SON	Hayır ise SON'a dallan.
010A	A6 00		LDAA SEG,X	A'ya 0201 adresindeki BCD kodlu sayının karşılığını SEG tablosundan alır.
010C	B7 02 04		STAA \$0204	BCD'nin 7 segment karşılığı 0204 adresine yüklenir.
010F	3F	SON:	RTS	Ana programa dön.
			ORG \$0000	Bundan sonraki program kodları 0000'den adresinden itibaren başlar.(Pseudo komutu)
0000		SEG:	DFB #\$3F, #\$06, #\$5B, #\$4F, #\$66, #\$6D, #\$7D, #\$07, #\$7F, #\$6F	

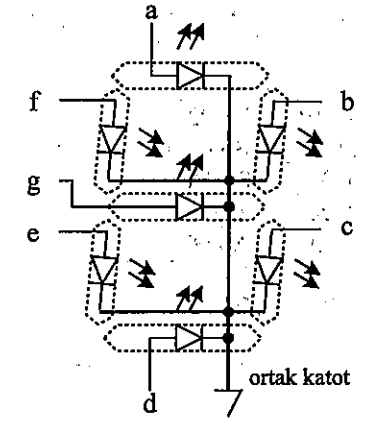
Açıklama: BCD kodlanmış bir sayının 7 segmentli displaylerde görüntülenmesi için displaydeki led bağlantılarına belirli bir düzende ikilik sayının oluşturulması gerekmektedir. Bir bakıma dijital elektronikte sıkça kullanılan 7447 kod dönüştürme entegresinin yaptığı işi burada yazılım yolu ile gerçekleştirmekteyiz. Hemen hatırlatalım; BCD kodlanmış bir veriyi 7 segment displaylerde görüntüleyebilmek için kod dönüştürücü bir entegre kullanmak gerekirken bir mikrobilgisayar sisteminde bu işi hafızanın beş on hücreğine yazacağımız bir program ile çözümlenebilmektedir. Böyle bir program ne sisteme ek bir maliyet getirecektir ne de donanımda fazla bir değişikliğe gidilecektir. Bu program mikrobilgisayarların esnekliğine iyi bir örnek olarak verilebilir.

Program hafızanın (0100)₁₆ adresine yerleşmiş bir alt program olarak hazırlanmıştır. Alt programın son satırında ana programa dönüş için RTS komutunun kullanıldığına dikkat edin. ORG \$0100 komutu ile alt programın hafızaya yerleşeceği adresin başlangıç değeri belirtilmiştir. Sonra LDX \$0200 komutu ile indeks kaydedicisine, hafızanın (0200)₁₆ ve (0201)₁₆ adreslerindeki veriler yüklenir. İndeks kaydedici 2 byte'lık bir kaydedici olduğundan bu kaydedicinin düşük değerlikli kısmına (0201)₁₆ adresindeki veriyi yüklemek için komut LDX \$0200 şeklinde yazılmıştır. LDAA \$0201 komutu ile A kaydedicisine (0201)₁₆ adresindeki BCD kodlanmış değer alınır. Bu değer 10'dan küçük olup olmadığına bakılır. Eğer alınan değer 10'dan büyük ise program SON etiketine oradan da ana programa dallanır. Eğer küçük ise işleme devam edilir.

Programa başlamadan önce 7 segment displayler için gerekli kodların tümü (bu kodlar tablo 14.19'de verilmiştir) (0000)₁₆ adresinden itibaren bir kod tablosu olarak hafızaya girilmelidir. Bu tablonun başlangıç adresi önemlidir. Çünkü dönüştürülecek BCD sayının 7 segment kodu, tablonun başlangıç adresine eklenen kayma değeri ile belirlenmiştir. Okunan BCD değerinin 7 segment display için gerekli kodu tablodan alındıktan sonra sonuç (0204)₁₆ adresine yazılacaktır. Eğer sistemimizde bir görüntüleme arabirimi mevcut ise dönüştürülen bu değer, görüntüleme arabirimine gönderilerek elde edilen rakamın displayde görüntülenmesi de sağlanabilir.

Tablo 14.21 : 7 segment displaylerde rakam görüntülemek için gereken ikilik kodlar

Rakam	Hex kod	İkili kod						
		D ₆ g	D ₅ f	D ₄ e	D ₃ d	D ₂ c	D ₁ b	D ₀ a
0	3F	0	1	1	1	1	1	1
1	06	0	0	0	0	1	1	0
2	5B	1	0	1	1	0	1	1
3	4F	1	0	0	1	1	1	1
4	66	1	1	0	0	1	1	0
5	6D	1	1	0	1	1	0	1
6	7D	1	1	1	1	1	0	1
7	07	0	0	0	0	1	1	1
8	7F	1	1	1	1	1	1	1
9	6F	1	1	0	1	1	1	1



Şekil 14.1 : Ortak katot display

14.5 ZAMAN GECİKTİRME PROGRAM YAZIMI

14.5.1 Zaman Geciktirme Alt Programlarına Duyulan İhtiyaç

Mikroişlemciler, kendi içerisinde saniyede yüzlerce komutu sonuçlandırabilecek hızda ve yapıdadırlar. Fakat mikroişlemciler ile birlikte kullanılan bazı elemanlar mikroişlemci ile aynı hızda çalışmamaktadırlar. Mikroişlemciler zamanının bir kısmını kullanıcıdan veya diğer birimlerden gelecek bilgiyi beklemekle geçirir. Bu gibi durumlarda hız uyumsuzluğu ortaya çıkacaktır. Örneğin düşük hızlı bir hafıza biriminden bir verinin veri yoluna bırakılması için mikroişlemcinin bir süre beklemesi gerekecektir. Veya düşük hızlı bir D/A çeviriciye art arda gönderilen verinin hızı, D/A çeviricinin **çevrim hızından**¹ yüksek olmamalıdır.

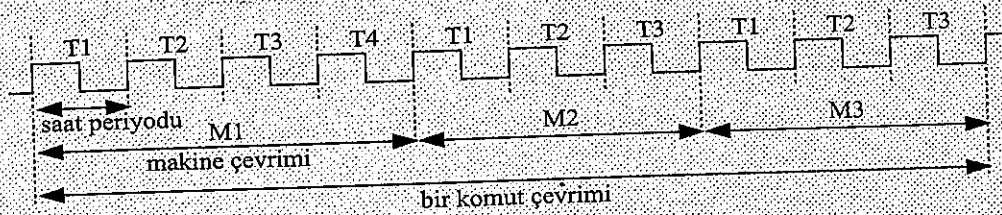
Mikrobilgisayarlarda zaman geciktirmek için en çok kullanılan yöntem mikroişlemcinin belirli bir döngüde bir süre kalmasını sağlamaktır. Gereken zaman geciktirme mikroişlemcinin bu döngüde kalma süresi ile hesaplanır. Bazı uygulamalarda çok hassas zaman geciktirme sürelerine ihtiyaç duyulmaz. Hassas zaman geciktirme gerektiği durumlarda ise döngüde bulunan komutların işletim süreleri tek tek hesaplanarak toplam süre bulunmalıdır.

¹ Çevrim Hızı; D/A veya A/D çeviricilerin girişindeki bilgiyi çıkış bilgisine dönüştürmek için harcayacağı zamanın süresidir.



Saat Periyodu, Makine Çevrimi ve Komut Çevrimi:

Bir "komut çevrimi" mikroişlemcinin, bir komutun gerekli bütün parçalarını alıp getirmesi ve o komutu yürütmesi için gerekli zamandır. Bir komut çevrimi mikroişlemcinin türüne göre 1 ile 5 arasında "makine çevriminden" oluşur. Yani bir komut çevrimi değişik bölümlerden oluşur ve bunlar M1, M2, M3, ... diye adlandırılmışlardır. M1, M2, M3, ... şeklinde adlandırılan makine çevrimleri kitabın komut çalışma evrelerinde anlatılmıştır. Bir mikroişlemcinin kalp atışları onun saatidir. Her makine çevrimi de belirli sayıda "saat periyotlarından" (saat sinyalinin) oluşur. Eğer bir komutun çevrimini tamamlaması için gereken toplam saat periyodunun sayısı ve her saat periyodunun süresi (T) bilinirse komutu yürütmek için gerekli zaman hesaplanabilir.



Intel 8080A, Intel 8085A ve Z80 mikroşlemcilerinde her komut çevrimi, komutun türüne göre 1 ile 5 makine çevriminden oluşur. Her makine çevriminde yine komutun türüne göre 3 ile 6 saat periyodundan oluşur. 4 MHz. saat giriş olan bir Z80 mikroşlemcisinde her saat periyodu

$$T = \frac{1}{f} = \frac{1}{4MHz} = 0,25\mu s. \text{ dir.}$$

Intel 8080A mikroşlemcisinde bir "OUT" komutu 3 makine periyodundan oluşmaktadır. Bunlar M1, M2 ve M3 olarak etiketlenmiştir. M1 makine çevrimi komut getirme (fetching, decoding) safhasıdır ve 4 saat periyodundan oluşur. M2 komut çevrimi 3 saat periyodundan oluşur. Bu da bir "fetching" olayıdır fakat bu safhada hafızadan veri getirildiğinden yani çözülecek bir komut olmadığından 3 saat periyodunda bitmektedir. M3 makine çevriminde, komut işlevini gerçekleştirir ve veri yoluna kaydedicisindeki veriyi bırakır. M3 komut çevrimi de, 3 saat periyodunda tamamlanmaktadır. Görüldüğü gibi "OUT" komutunun "bir komut çevrimi", 3 "makine çevriminden" ve 4+3+3=10 "saat periyodundan" oluşmaktadır. 8080A da çok kullanılan saat frekansı 2 MHz.dir ve bu frekansta bir saat periyodu 0,5 µs. eder.

Toplam saat periyodu sayısı x Bir saat periyodu süresi = Bir komut çevrim süresi.

Dolayısıyla bir OUT komut çevrim süresi toplam olarak 10x0,5=5 µs.dir.

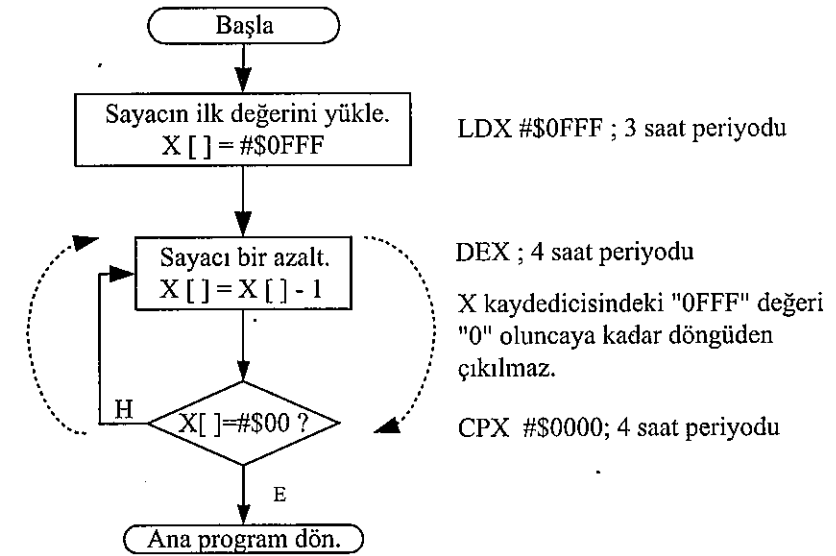
MC6800 ve R6502 mikroşlemcilerinde bir komut çevrimi 2 ile 8 arasında makine çevriminden oluşmaktadır. Makine çevrim sayısı, saat periyodu sayısına eşit olduğundan, makine çevrim sayısı ile saat periyodunu çarparak herhangi bir komutun "komut çevrim süresi" hesaplanmaktadır. MC6800 ve R6502 mikroşlemcileri, "boru hattı" (pipelining) denilen bir tekniktan yararlandıkları için daha düşük frekanslarda daha kısa komut çevrim sürelerine sahip olabilmektedirler. Boru hattı, son getirilen byte yürütülürken bir sonraki byte'ın hafızadan getirilmesi tekniğidir.

Mikroşlemci üreticileri, ürettikleri mikroşlemcilerin komut setinde her komutun işletilmesi için gereken "makine çevrimi sayısını" veya "saat periyodu sayılarını" verirler. MC6802¹ mikroşlemcisinin komut tablosun incelendiğinde her komutun kaç "saat periyodunda" tamamlandığı görülecektir. Örneğin 4 MHz hariç saat frekansı olan MC6802 mikroşlemcisinin her saat periyodu aşağıdaki gibi hesaplanır.

$$T = \frac{1}{f} = \frac{1}{4MHz} = 0,25\mu s.$$

MC6802 mikroşlemcisinde "LDAA #0F" komutunun tamamlanması 2 saat periyodu gerektirdiğinden bu komutun işlevini gerçekleştirmesi için geçen süre de 0,5 µsn olacaktır. Aynı mikroşlemcide "LDX \$00FF" komutu 6 saat sinyaline ihtiyaç duyduğundan harcanan süre 1,5 µsn olacaktır. MC6802 mikroşlemcisinde NOP komutu 2 makine periyodunda tamamlandığından hiçbir işlem yapmadan 0,5 µsn'lik bir gecikme sağlar. NOP komutunun art arda kullanılması belirli oranda bir gecikme sağlayacaktır. Fakat uzun zaman geciktirme gerektiren durumlarda bu komutun bir döngü içinde kullanılması daha gerçekçi bir yaklaşımdır.

Zaman geciktirmek için kullanılan yöntemin sonlu döngülerle yapıldığını söylemiştik. Bu tip döngülerde genel olarak bir sayıcı, sayıcı değerini arttıran bir komut ve sayıcının belirlenen değere gelip gelmediğine bakan bir karşılaştırma komutu içerir. Diyagram 14.10'da örnek bir zaman geciktirme alt programlarında kullanılan akış diyagramı gösterilmiştir.



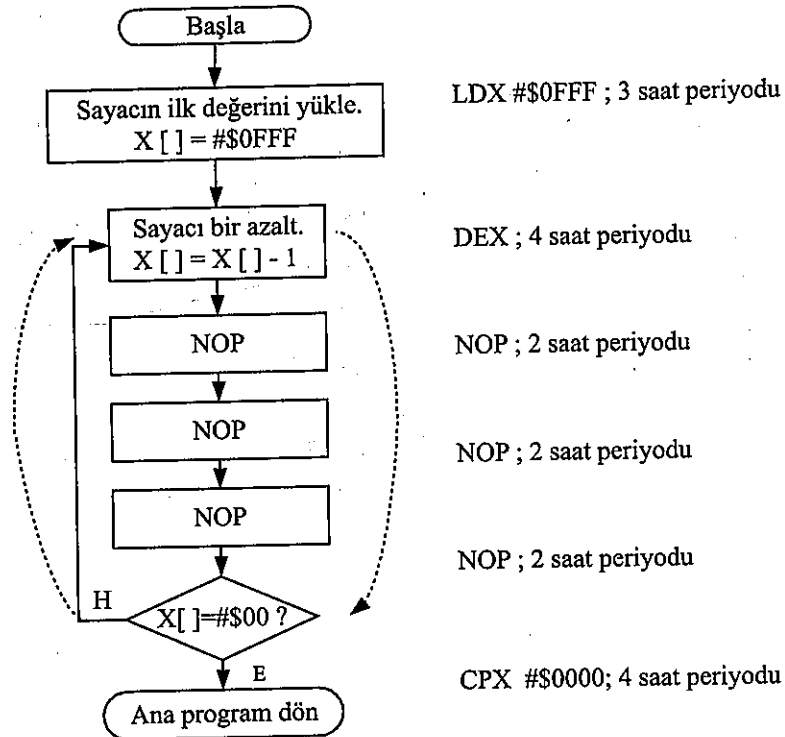
Diyagram 14.12 : Zaman geciktirme algoritması

Zaman geciktirme algoritmasında da görüldüğü gibi ilk komut 3 saat periyodu, ikinci komut 4 saat periyodu, üçüncü komut 4 saat periyodu süresince işletilmektedir. İkinci ve üçüncü komutlar indeks kaydedicisindeki (0FFF)₁₆ değeri "0" oluncaya kadar tekrar etmektedir, dolayısıyla ikinci ve üçüncü komutlar (0FFF)₁₆ = (4095)₁₀ kez tekrar etmektedir. Bu tekrarlar neticesinde geçen toplam süre;

$$[3 \text{ saat periyodu} + (4 \text{ saat periyodu} + 4 \text{ saat periyodu}) \times 4095] = 32763 \text{ saat periyodu}$$

¹ MC6802 ve R6502 mikroşlemcilerinde "makine çevrim sayıları" ve "saat periyotları" aynıdır.

4 MHz osilatör frekansı olan MC6802 mikroişlemcisinin her saat periyodu 0,25 µsn olduğundan $32,763 \times 0,25 \mu\text{sn} = 8190,75 \mu\text{sn}$ bu da yaklaşık 8,2 msn edecektir. Bu sürenin daha uzun olması istendiğinde döngü içine NOP komutu kullanılabilir.



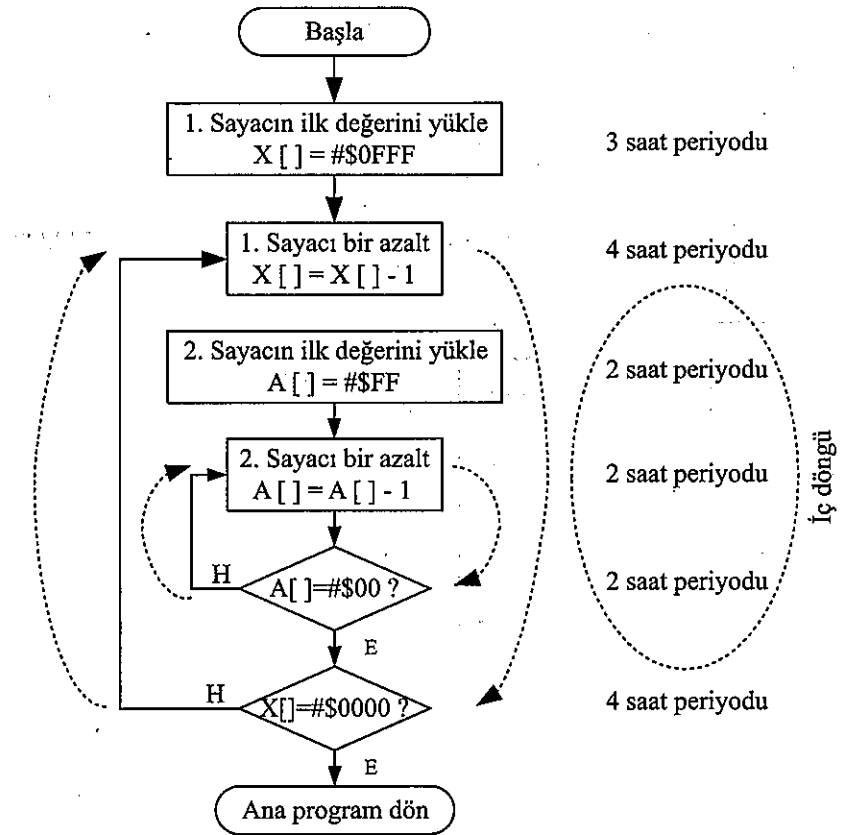
Diyagram 14.13 : Süresi uzatılan gecikme döngüsü

Diyagram 14.11'de görülen algoritmada, döngü içerisinde NOP komutu kullanıldığında süre biraz daha uzatılmış olacaktır. NOP komutu kullanıldığında bu alt program için geçen toplam süre;

$$[3 \text{ periyot} + (4 \text{ periyot} + 2 \text{ periyot} + 2 \text{ periyot} + 2 \text{ periyot} + 4 \text{ periyot}) \times 4095] = 57333 \text{ periyot eder.}$$

57333 periyot 4 MHz hızındaki bir mikroişlemcide toplam $57333 \times 0,25 \mu\text{sn} = 14333,25 \mu\text{sn}$ bu da yaklaşık olarak 14,4 msn eder.

Döngü içinde NOP komutu kullanılması ile geciktirme döngüsünde kısa bir fark oluşturulabilmekteyiz. Daha uzun süreli geciktirme alt programları elde edebilmek için iç içe döngüler kullanmak gerekir.



Diyagram 14.14 : Uzun zaman geciktirme algoritması

Diyagram 14.12'deki akış diyagramında, uzun süreli zaman geciktirme elde etmek için iç içe döngülerin kullanımı gösterilmiştir. İç döngüde A kaydedicisindeki değer (FF)₁₆'den (00)₁₆ oluncaya kadar (255)₁₀ kez tekrarlanmaktadır. İçerideki döngüde oluşan gecikme süresi yaklaşık olarak;

$$[2 \text{ saat periyodu} + (2 \text{ saat periyodu} + 2 \text{ saat periyodu}) \times 255] = 1022 \text{ saat periyodu eder.}$$

Dışarıdaki döngü her seferinde içerideki döngüye gireceği için alt programın süresi önemli ölçüde uzamaktadır. Dış döngüde X kaydedicisindeki değer (0FFF)₁₆'den (0000)₁₆ oluncaya kadar (4095)₁₀ kez tekrarlanmaktadır. Döngünün tamamı yaklaşık olarak;

$$[3 \text{ saat periyodu} + (1022 \text{ saat periyodu}) \times 4095] = 4185093 \text{ periyodu eder.}$$

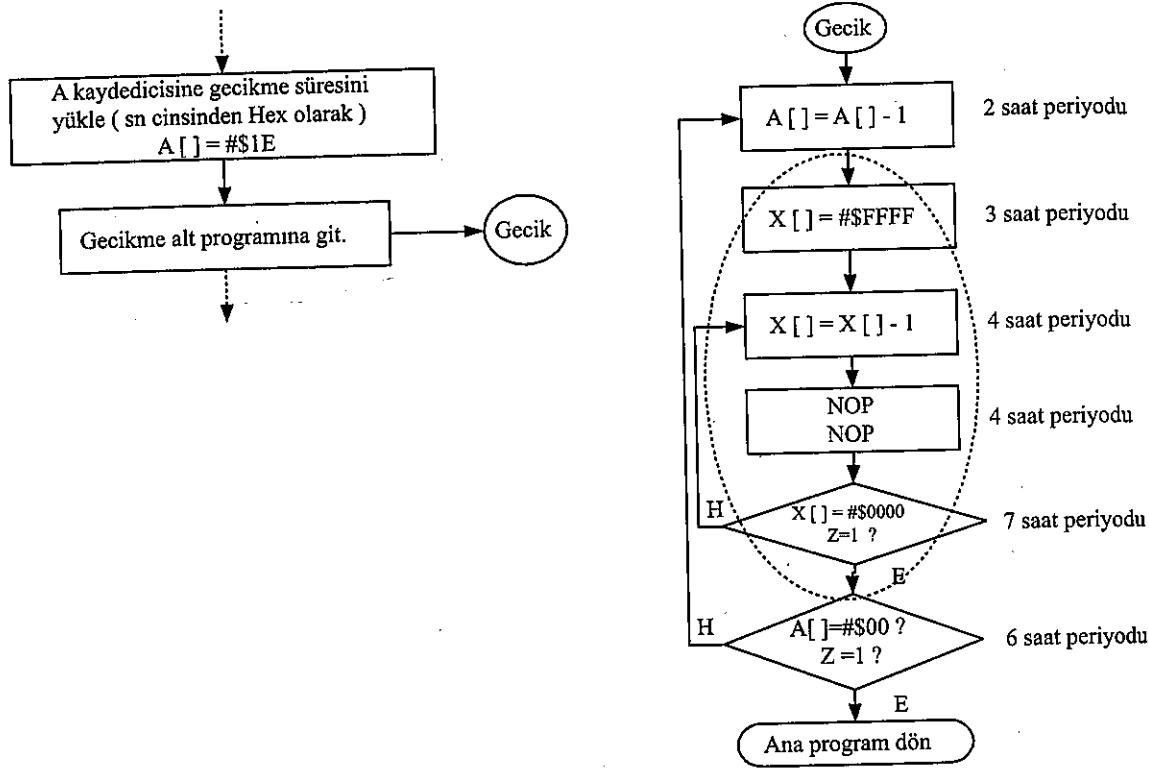
Bu da 4 MHz hızındaki bir mikroişlemcide yaklaşık olarak $0,25 \times 4185093 = 1046273,25 \mu\text{sn}$ bu da yaklaşık 1,05 sn'lik bir gecikme demektir.

Gecikme döngülerini, alt program şeklinde hazırlamak daha profesyonel bir yaklaşımdır. Gereken gecikmeyi saniye cinsinden alt programa aktardığımızda istediğimiz süredeki gecikmeyi her zaman kullanabilmek olanağı bulmuş olacağız.

14.5.2 30 Saniyelik Uzun Zaman Geciktirme Alt Programı

Amaç; hafızanın (0200)₁₆ adresinde bulunan 0,25 saniyelik gecikme alt programını kullanarak 30 sn'lik gecikme elde etmek.

Akış diyagramı



Diyagram 14.15 : 30 sn'lik zaman geciktirme algoritması

Tablo 14.22 : Programın kaynak kodları (MC6802)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	86 1E		LDAA #1E	A'ya 30 sn'lik gecikme için (1E) ₁₆ yükle
0002	BD 02 00		JSR \$0200	Gecikme alt programına dallan.
0005	3F		SWI	Programı bitir.
...	...		...	...
0200	4A	GECIK:	DECA	A'yı bir azalt
0201	CE FF FF		LDX #FFFF	X'e (FFFF) ₁₆ değerini yükle
0204	09	AZL:	DEX	X'i bir azalt
0205	01		NOP	İşlem yapma
0206	01		NOP	İşlem yapma
0207	8C 00 00		CPX #0000	X[]=(0000) ₁₆ oldu mu ?
020A	26 F8		BNE AZL	X[]=(0000) ₁₆ değilse AZL etiketine dallan
020C	81 00		CMPA #00	A[]=(00) ₁₆ oldu mu ?
020E	26 F0		BNE GECIK:	A[]=(00) ₁₆ değilse GECIK etiketine dallan
0210	39		RTS	Ana programa dön

Açıklama: Program 4 MHz osilatör frekansı olan bir MC6802 mikroişlemcisine göre yazıldığından komut süreleri de buna göre hesaplanmıştır. Buna göre her saat periyodu 0,25 sn olmaktadır. Hafızanın (0200)₁₆ adresinde bulunan 0,25 sn'lik gecikme alt programını kullanarak 30 sn'lik bir gecikme elde edebilmek için A kaydedicisine 30x0,25=120, (78)₁₆ yükledikten sonra gecikme alt programını çağırmak yeterli olacaktır. Gecikme alt programı A kaydedicisinde bulunan değer kadar gecikme döngüsünde kalacaktır. Gecikme alt programına A kaydedicisi üzerinden aktarılan değer (04)₁₆ olduğunda yaklaşık olarak 1 sn'lik bir gecikme sağlanacaktır.

$$[3 \text{ periyot} + (4 \text{ periyot} + 4 \text{ periyot} + 7 \text{ periyot}) \times 65.535] = 983.028 \text{ saat periyodu eder.}$$

Diyagramda daire içine alınan iç döngü 983028 x 0,25=245757 µsn'lik bir gecikme sağlar. Bu süre yaklaşık olarak 0,25 sn olarak kabul edilirse, dış döngü bu süreyi 120 kez tekrar edeceği için yaklaşık olarak toplam 30 sn'lik bir gecikme sağlanmış olacaktır.

$$[(2 \text{ periyot} + 983028 \text{ periyot} + 6 \text{ periyot}) \times 120] = 117964320 \text{ periyot eder.}$$

117964320 saat periyodu 4 MHz osilatör frekansına sahip bir mikroişlemcide 29491080 µsn'lik bir süreye karşılık gelir. Bu da yaklaşık olarak 29,5 sn eder.

İstenilen gecikme süresini tam değerinde elde edebilmek için döngülerde kullanılan komutların sayılarının ve sürelerinin çok hassas ayarlanması gerekir. Örneğin çok hassas süreler için alt programa gidiş ve alt programdan dönüş sürelerini de hesaba katmak gerekir. Fakat çoğu uygulamalarda bu süreler dikkate alınmayacak ölçüde önemsizdir.



Zamanlayıcı (timer) Entegreler:

Zamanın çok önemli olduğu uygulamalarda bu süre ayarlamalarını yazılımla gerçekleştirmek, istenmeyen zamanlama hatalarına sebep olabilmektedir. Aynı zamanda bu süreleri yazılım yoluyla elde etmek mikroişlemcinin zamanını sırf bu döngülerde geçirmesine neden olacaktır. Bunun yerine mikrobilgisayarlarda kullanılan "zamanlayıcı" veya "timer" entegreleri profesyonel çözümler sunmaktalar. Programlanabilen bu zamanlayıcılar ile belirlenen sürelerde zamanın dolduğunu mikroişlemciye bir kesme sinyali göndererek bildirirler. Motorola ailesinde MC6840 zamanlayıcı entegrelerine verilebilecek bir örnektir.

14.6 GİRİŞ/ÇIKIŞ İLİŞKİLİ PROGRAM YAZIMI

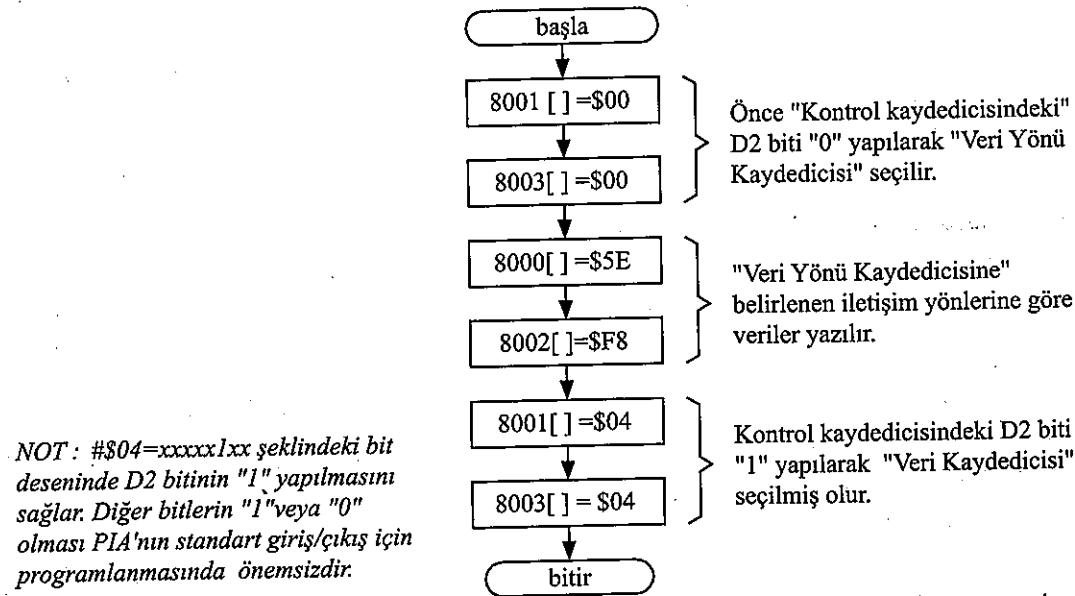
14.6.1 PIA'nın Temel Giriş/Çıkış İşlemleri için Programlanması

Tablo 14.23 : Örnek giriş/çıkış uçları

B grubu giriş/çıkış								A grubu giriş/çıkış							
PB ₇	PB ₆	PB ₅	PB ₄	PB ₃	PB ₂	PB ₁	PB ₀	PA ₇	PA ₆	PA ₅	PA ₄	PA ₃	PA ₂	PA ₁	PA ₀
Çıkış	Çıkış	Çıkış	Çıkış	Çıkış	Giriş	Giriş	Giriş	Giriş	Çıkış	Giriş	Çıkış	Çıkış	Çıkış	Çıkış	Giriş
1	1	1	1	1	0	0	0	0	1	0	1	1	1	1	0
F				8				5				E			

Amaç; temel bağlantı adresi (8000)₁₆...(8003)₁₆ arasında olan MC6821 paralel iletişim entegresinin standart giriş-çıkış şeklinde kullanımı için Tab' o 14.21'de belirtilen giriş/çıkış düzeninde programlamak.

Akış diyagramı



Diyagram 14.16 : MC6821 PIA'nın programlanması için akış diyagramı

Tablo 14.24 : Programın kaynak kodları (MC6802 mikroişlemcisi komutları ile)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	7F 80 01		CLR \$8001	(8001) ₁₆ adresi silinerek A portunun kontrol kaydedicisindeki D2 bitini "0" yapılır.
0003	7F 80 03		CLR \$8003	(8003) ₁₆ adresi silinerek B portunun kontrol kaydedicisindeki D2 bitini "0" yapılır.
0006	86 5E		LDAA #\$5E	A kaydedicisine belirlenen iletişim yönlerindeki bit deseni yüklenir.
0008	B7 80 00		STAA \$8000	A'daki veri (8000) ₁₆ 'deki A portunun veri yönü kaydedicisine yüklenir.
000B	86 F8		LDAA #\$F8	A kaydedicisine belirlenen iletişim yönlerindeki bit deseni yüklenir.
000D	B7 80 02		STAA \$8002	A'daki veri (8002) ₁₆ 'deki B portunun veri yönü kaydedicisine yüklenir.
0010	86 04		LDAA #\$04	A kaydedicisine D2 bitini "1" yapacak değer yüklenir.
0012	B7 80 01		STAA \$8001	(8001) ₁₆ adresine (04) ₁₆ verisini yükleyerek A portu kontrol kaydedicisinin D2 bitini "1" yapılır.
0015	B7 80 03		STAA \$8003	(8003) ₁₆ adresine (04) ₁₆ verisini yükleyerek B portunun kontrol kaydedicisindeki D2 bitini "1" yapılır.

Açıklama: MC6821 PIA'nın veri yönü kaydedicilerine erişmek için kontrol kaydedicisindeki D₂ bitinin "0" yapılması gerekmektedir. PIA standart giriş/çıkış şeklinde kullanılacağı için kontrol kaydedicisindeki diğer bitlerin durumunu önemsemeden D₂ biti ile birlikte tüm bitler CLR \$8001 ve CLR \$8002 komutları ile sıfırlanmaktadır. Böylece (8000)₁₆ adresi ile A portunun, (8002)₁₆ adresi ile de B portunun "veri yönü kaydedicilerine" erişim sağlanabilmektedir. A portunun tablo 14.21'deki gibi bir iletişim düzeninde olması için A kaydedicisi (5E)₁₆ verisi ile yüklenir. Sonra bu veri STAA \$8000 komutu ile A portu "veri yönü kaydedicisine" yüklenir. Aynı şekilde B portunun istenen iletişim düzeninde olması için B kaydedicisi (F8)₁₆ verisi ile yüklendikten sonra bu veride STAA \$8002 komutu ile B portu "veri yönü kaydedicisine" yüklenir. Son olarak A ve B portlarına çevre birimlerine veri alma/gönderme işlerini yapabilmek için (8001)₁₆ ve (8003)₁₆ adreslerindeki D₂ bitleri "1" yapılmıştır. Bu LDAA #\$04 komutu ve ardından STAA \$8001 ve STAA \$8003 komutları kullanılarak yapılmıştır.

14.6.2 Anahtar ile Led Yakıp Söndürme Programı

Amaç; bu programın hazırlanması ve anlaşılması oldukça basittir. Bu program bizi bir mikrobilgisayarda PIA ile temel giriş/çıkış işlemlerinin nasıl yapıldığı hakkında bilgi sahibi yapacaktır.

Temel bağlantı adresi (8000)₁₆...(8003)₁₆ adresleri arasında olan MC6821 paralel iletişim entegresinin A portuna bağlı anahtarlardan alınan bilgiye göre B portundaki led diyotları yakmak (Anahtar "1" konumunda iken led yanar, "0" konumunda iken led söner.).

NOT: PIA'nın A portundaki tüm giriş/çıkış uçlarının giriş olacak şekilde programlandığı ve B portunun tüm giriş/çıkış uçlarının çıkış olacak şekilde programlandığı varsayılarak led yakma söndürme programı hazırlanmıştır. A portunun giriş adresi (8000)₁₆ ve B portunun çıkış adresi (8002)₁₆'dir.

14.6 GİRİŞ/ÇIKIŞ İLİŞKİLİ PROGRAM YAZIMI

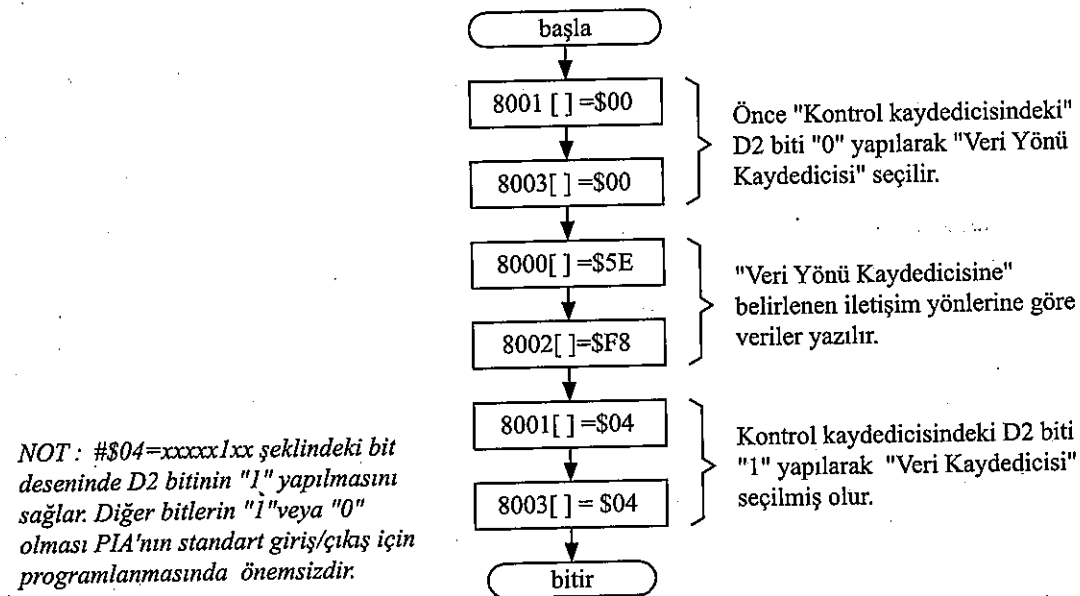
14.6.1 PIA'nın Temel Giriş/Çıkış İşlemleri için Programlanması

Tablo 14.23 : Örnek giriş/çıkış uçları

B grubu giriş/çıkış								A grubu giriş/çıkış							
PB ₇	PB ₆	PB ₅	PB ₄	PB ₃	PB ₂	PB ₁	PB ₀	PA ₇	PA ₆	PA ₅	PA ₄	PA ₃	PA ₂	PA ₁	PA ₀
Çıkış	Çıkış	Çıkış	Çıkış	Çıkış	Giriş	Giriş	Giriş	Giriş	Çıkış	Giriş	Çıkış	Çıkış	Çıkış	Çıkış	Giriş
1	1	1	1	1	0	0	0	0	1	0	1	1	1	1	0
F				8				5				E			

Amaç; temel bağlantı adresi (8000)₁₆... (8003)₁₆ arasında olan MC6821 paralel iletişim entegresinin standart giriş-çıkış şeklinde kullanımı için Tab'o 14.21'de belirtilen giriş/çıkış düzeninde programlamak.

Akış diyagramı



Diyagram 14.16 : MC6821 PIA'nın programlanması için akış diyagramı

Tablo 14.24 : Programın kaynak kodları (MC6802 mikroişlemcisi komutları ile)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	7F 80 01		CLR \$8001	(8001) ₁₆ adresi silinerek A portunun kontrol kaydedicisindeki D2 bitini "0" yapılır.
0003	7F 80 03		CLR \$8003	(8003) ₁₆ adresi silinerek B portunun kontrol kaydedicisindeki D2 bitini "0" yapılır.
0006	86 5E		LDAA #\$5E	A kaydedicisine belirlenen iletişim yönlerindeki bit deseni yüklenir.
0008	B7 80 00		STAA \$8000	A'daki veri (8000) ₁₆ 'deki A portunun veri yönü kaydedicisine yüklenir.
000B	86 F8		LDAA #\$F8	A kaydedicisine belirlenen iletişim yönlerindeki bit deseni yüklenir.
000D	B7 80 02		STAA \$8002	A'daki veri (8002) ₁₆ 'deki B portunun veri yönü kaydedicisine yüklenir.
0010	86 04		LDAA #\$04	A kaydedicisine D2 bitini "1" yapacak değer yüklenir.
0012	B7 80 01		STAA \$8001	(8001) ₁₆ adresine (04) ₁₆ verisini yükleyerek A portu kontrol kaydedicisinin D2 bitini "1" yapılır.
0015	B7 80 03		STAA \$8003	(8003) ₁₆ adresine (04) ₁₆ verisini yükleyerek B portunun kontrol kaydedicisindeki D2 bitini "1" yapılır.

Açıklama: MC6821 PIA'nın veri yönü kaydedicilerine erişmek için kontrol kaydedicisindeki D₂ bitinin "0" yapılması gerekmektedir. PIA standart giriş/çıkış şeklinde kullanılacağı için kontrol kaydedicisindeki diğer bitlerin durumunu önemsemeden D₂ biti ile birlikte tüm bitler CLR \$8001 ve CLR \$8002 komutları ile sıfırlanmaktadır. Böylece (8000)₁₆ adresi ile A portunun, (8002)₁₆ adresi ile de B portunun "veri yönü kaydedicilerine" erişim sağlanabilmektedir. A portunun tablo 14.21'deki gibi bir iletişim düzeninde olması için A kaydedicisi (5E)₁₆ verisi ile yüklenir. Sonra bu veri STAA \$8000 komutu ile A portu "veri yönü kaydedicisine" yüklenir. Aynı şekilde B portunun istenen iletişim düzeninde olması için B kaydedicisi (F8)₁₆ verisi ile yüklendikten sonra bu veride STAA \$8002 komutu ile B portu "veri yönü kaydedicisine" yüklenir. Son olarak A ve B portlarına çevre birimlerine veri alma/gönderme işlerini yapabilmek için (8001)₁₆ ve (8003)₁₆ adreslerindeki D₂ bitleri "1" yapılmıştır. Bu LDAA #\$04 komutu ve ardından STAA \$8001 ve STAA \$8003 komutları kullanılarak yapılmıştır.

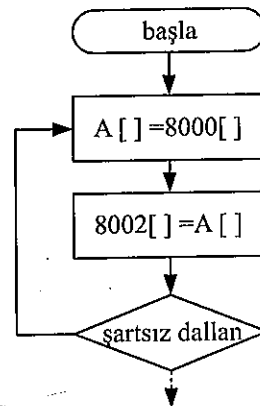
14.6.2 Anahtar ile Led Yakıp Söndürme Programı

Amaç; bu programın hazırlanması ve anlaşılması oldukça basittir. Bu program bizi bir mikrobilgisayarda PIA ile temel giriş/çıkış işlemlerinin nasıl yapıldığı hakkında bilgi sahibi yapacaktır.

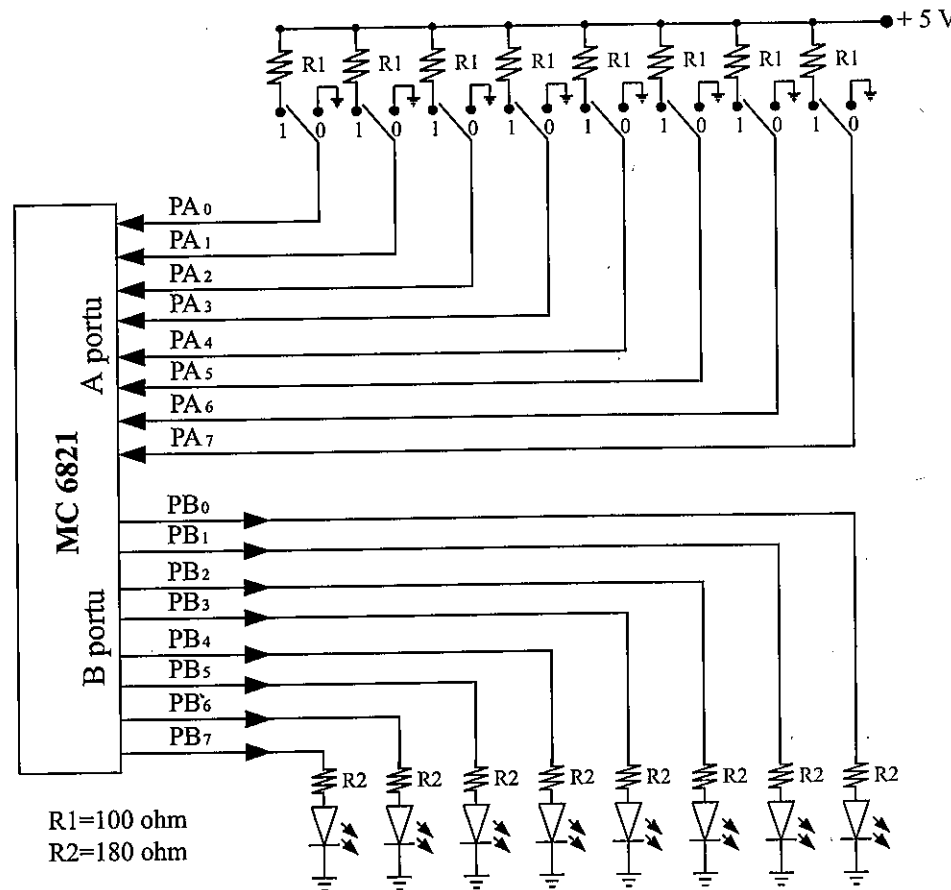
Temel bağlantı adresi (8000)₁₆... (8003)₁₆ adresleri arasında olan MC6821 paralel iletişim entegresinin A portuna bağlı anahtarlardan alınan bilgiye göre B portundaki led diyotları yakmak (Anahtar "1" konumunda iken led yanar, "0" konumunda iken led söner.).

NOT: PIA'nın A portundaki tüm giriş/çıkış uçlarının giriş olacak şekilde programlandığı ve B portunun tüm giriş/çıkış uçlarının çıkış olacak şekilde programlandığı varsayılarak led yakma söndürme programı hazırlanmıştır. A portunun giriş adresi (8000)₁₆ ve B portunun çıkış adresi (8002)₁₆'dir.

Akış diyagramı



Diyagram 14.17 : Anahtar bilgisine göre led yakma söndürme programı akış diyagramı



Şekil 14.2 : PIA kullanarak "anahtar ile led yakıp söndürme" devre şeması

Tablo 14.25 : Programın kaynak kodları (M6802)

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
0000	B6 80 00	OKU:	LDAA \$8000	(8000) ₁₆ adresindeki A portundan veri oku.
0003	B7 80 02		STAA \$8002	(8002) ₁₆ adresindeki B portuna, A portundan okuduğun bilgiyi gönder.
0006	20 F8		BRA OKU	Şartsız olarak OKU etiketine dallan.
0008	3F		SWI	Programı bitir. SWI komutundan önce BRA komutu ile şartsız dallanma gerçekleştiğinden program sayıcı hiçbir zaman SWI'nın olduğu satıra gelemes. Dolayısı ile bu komut burada görevsizdir.

Açıklama: Program A portundan okuduğu veriyi B portuna gönderecek şekilde hazırlanmıştır. Önce A portundan LDAA komutu ile veri okunarak A kaydedicisine gönderilir. Daha sonra A kaydedicisine alınan bu veri (8002)₁₆ adresindeki B portundan dışarı çıkarılır ve bu işlem program resetleninceye kadar A portundan okunan veri B portundan dışarı çıkarılır. A portundan okunan anahtarların konumuna göre B portundaki ledler ışık verecektir.

NOT : Assembly programları; üzerinde çalıştığı sistemin fiziksel özelliklerine göre hazırlanmak zorundadır. Bundan dolayı sistemde bulunan tümü giriş/çıkış birimlerinin, hafızaların ve diğer birimlerin (timer, ADC gibi) adreslerinin bilinmesi gerekmektedir. Giriş/çıkış arabirimlerinin nasıl adreslendiği 11. bölümde anlatılmıştır. Bölüm 10'daki hafıza organizasyonu da bu kapsamda incelenebilir.

DEĞERLENDİRME ÇALIŞMALARI

- 1) Hafızada bulunan iki sayının toplanarak başka bir hafıza hücresine yüklenmesi için MC6802 komutları ile bir program hazırlayınız.
- 2) Farklı hafıza adreslerinded bulunan verilerin yerlerini değiştiren bir programı Z80 komutları ile hazırlayınız.
- 3) İki sayıdan büyüğünü bulun bir programı MC6802 komutları ile hazırlayınız.
- 4) 8 bitlik iki sayının çarpımını hesaplayan bir programı 6502 komutları ile hazırlayınız.
- 5) Hafızada sıralı şekilde bulunan bir grup verinin içerisinde en küçüğünü bulan programı MC6802 komutları ile hazırlayınız.
- 6) Hafızada sıralı şekilde bulunan bir grup verinin içerisinde bizim belirlediğimiz bir veriyi arayan bir programı Z80 komutları ile hazırlayınız.
- 7) Belirli bir hafıza bölgesini "00" ile dolduran programı MC6802 komutları ile hazırlayınız.
- 8) Belirli bir hafıza bölgesini başka bir yere taşıyan programı Z80 mikroişlemcisi komutları ile hazırlayınız.
- 9) BCD kodlanmış bir veriyi ikilik tabana çeviren programı MC6802 komutlarını kullanarak hazırlayınız.
- 10) 2 dk. gecikme sağlayan alt programı 6502 komutları ile hazırlayınız.
- 11) Sistemin (8000)₁₆...(8003)₁₆ adreslerinde bulunan MC6821 paralel giriş/çıkış entegresinin A portunu giriş, B portunu çıkış olacak şekilde MC6802 mikroişlemcisi komutlarını kullanarak program hazırlayınız.

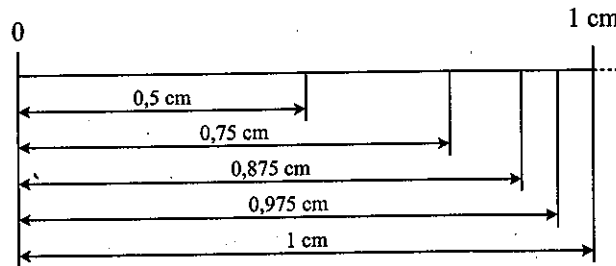
ÜNİTE 15 ANALOG/DİJİTAL VE DİJİTAL/ANALOG SİSTEMLER

15.1 ANALOG VE DİJİTAL NİCELİKLER

Fiziksel büyüklükler ölçülebilen ve bir değeri olan büyüklüklerdir. Bunlardan bazıları;

- ❖ Sıcaklık
- ❖ Uzunluk
- ❖ Işık şiddeti
- ❖ Hız
- ❖ Ağırlık
- ❖ Yoğunluk
- ❖ Ses şiddeti
- ❖ Elektriksel potansiyel

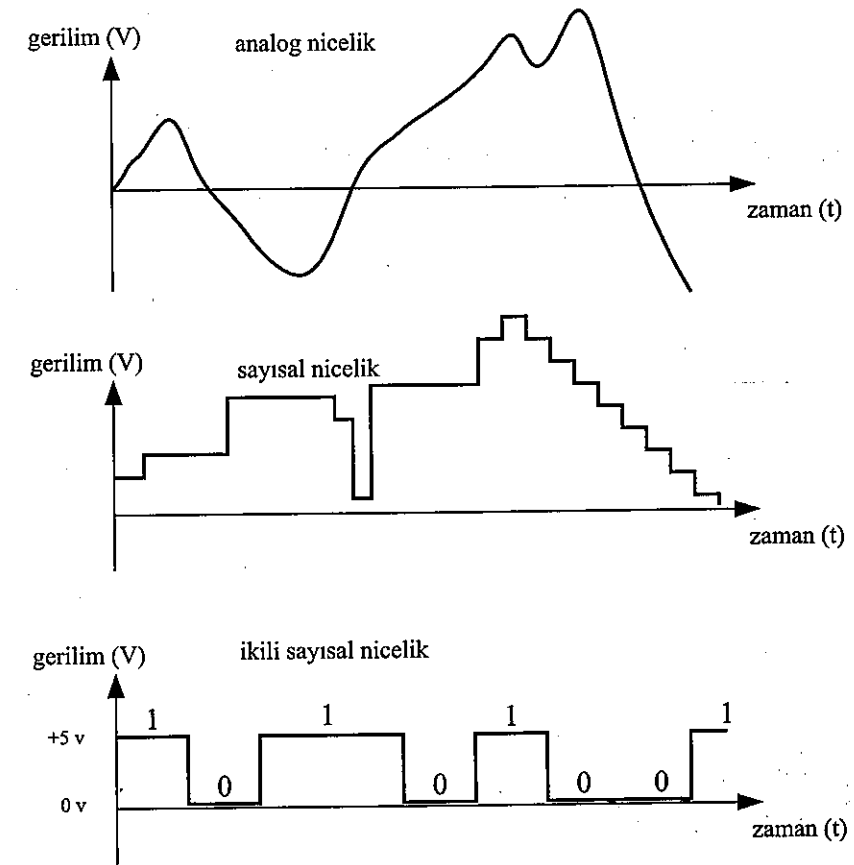
Bunlar gibi daha birçok fiziksel nicelikler sayılabilir. Bu fiziksel nicelikler kendi birimleri ile ifade edilir ve iki ölçüm değeri arasında sonsuz sayıda başka değerler bulunabilir. Örneğin 0 ile 1 cm uzunluk arasında sonsuz sayıda uzunluk vardır. Diğer fiziksel nicelikler içinde aynı durum söz konusudur. Bu tür fiziksel niceliklere, “analog” nicelikler de denir.



Şekil 15.1 : Analog bir nicelik; Uzunluk

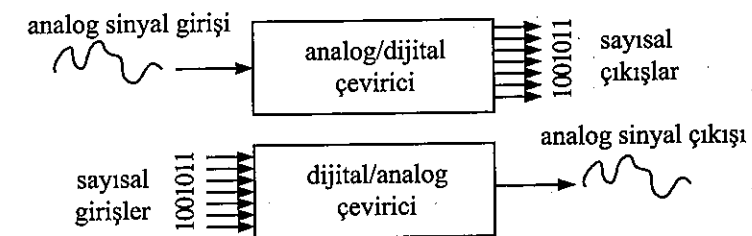
Sayısal nicelikler, analog niceliklerin tam sayılar ile ifade edilmesi şeklinde düşünülebilir. Örneğin bir sepetteki yumurta sayısı, sayısal bir nicelik. Sepetteki yumurta ya 10 tanedir ya da 11 tanedir. Sepette 10,5 tane yumurta olmaz. Başka bir örnek; eğer bir eğimli yerden yukarı çıkıyorsak basabileceğimiz sonsuz sayıda nokta vardır fakat bir merdivenden çıkıyorsak basabileceğimiz yer yalnızca basamaklardır.

Analog nicelikler zaman içerisinde herhangi bir anda herhangi bir değeri alabilirken sayısal nicelikler herhangi bir anda yalnız belirli değerleri alabilirler. Analog ve sayısal nicelikler arasındaki fark ta budur. Sayısal nicelik, yalnız iki değerden birisi ile ifade edilebiliyorsa buna “ikili sayısal nicelik” denilir. Lâmbanın açık veya kapalı olması gibi.



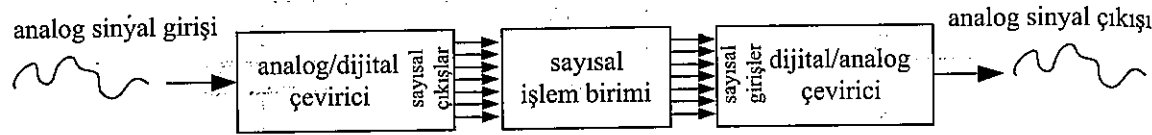
Şekil 15.2 : Analog, sayısal ve ikili sayısal nicelikler

Analog sinyalleri kullanarak çalışan elektronik devrelere “analog devreler” denilir. Elektronik teknolojinin ilk yıllarında, kullanılan elektrik sinyallerinin tamamı “analog sinyaller” idi. Analog sinyaller ile çalışan devrelerden en tanınmış olanları; radyo verici ve alıcıları, ses frekans yükselteçleri, televizyonlar sayılabilir ve bunlar hâlâ günümüzde yaygın olarak kullanılmaktadır. Dijital elektronikteki gelişmeler elektronik devre ve cihazlara çok fazla fonksiyonellik kazandırdığı, dijital verilerin saklanması ve işlenmesinin daha kolay olması nedeniyle tercihler analog devrelerden dijital devrelere doğru kaymaya başlamıştır. Buna paralel olarak analog sinyallerin de dijital sinyallere dönüştürülmesi önem kazanmıştır. Aynı şekilde sayısal sinyallerinde analog sinyallere çevrilmesinin gerektirdiği durumlar oluşmuştur. Şekil 15.3’de, analog elektrik sinyalinin sayısal elektrik sinyaline ve sayısal elektrik sinyalinin analog elektrik sinyaline çevrilmesinde kullanılan A/D (analog/dijital) ve D/A (dijital/analog) çeviricilerin blok diyagramı verilmiştir.



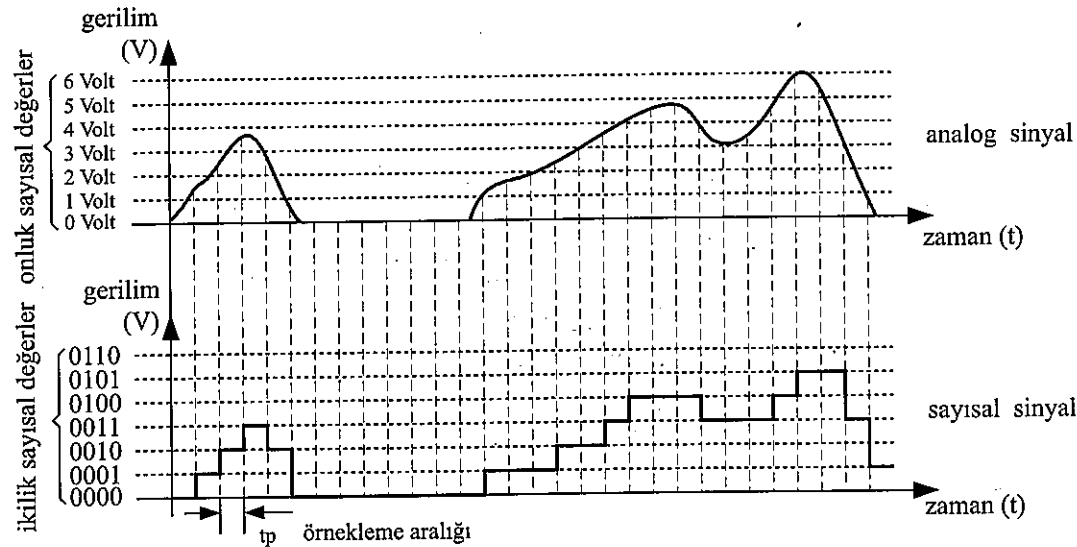
Şekil 15.3 : A/D ve D/A Çeviricilerin blok diyagramı

Bazı sistemlerin analog girişi ve analog çıkışı olmasına rağmen, sistem içerisinde yapılan tüm işlemler dijital olabilmektedir. Böyle durumlarda sisteme girilen analog sinyal önce sayısala çevrilir. Daha sonra sayısal sinyal üzerinde istenen işlemler gerçekleştirilerek çıkışta tekrar analog sinyale çevrilir. Şekil 15.4'de bu işleri yapan bir sistemin blok diyagramı verilmiştir. Böyle bir çalışmaya en iyi örnek, kişisel bilgisayarlarda kullanılan ses kartlarını verilebiliriz.



Şekil 15.4 : Analog sinyallerin, sayısal olarak işlenmesi ve tekrar analog sinyale dönüştürülmesi

Pratikte kullandığımız tüm fiziksel değerlerin birbirine dönüştürülmesi sırasında oluşabilecek kayıplar A/D ve D/A çevirme işlemlerinde de olmaktadır. Analog/dijital çevirici kullanılarak analog bir sinyalin dijital bir sinyale dönüştürüldükten sonra oluşan dalga şekli, şekil 15.5'de görülmektedir.



Şekil 15.5 : Analog bir sinyalin, dijital sinyale dönüştürülmesi grafiği

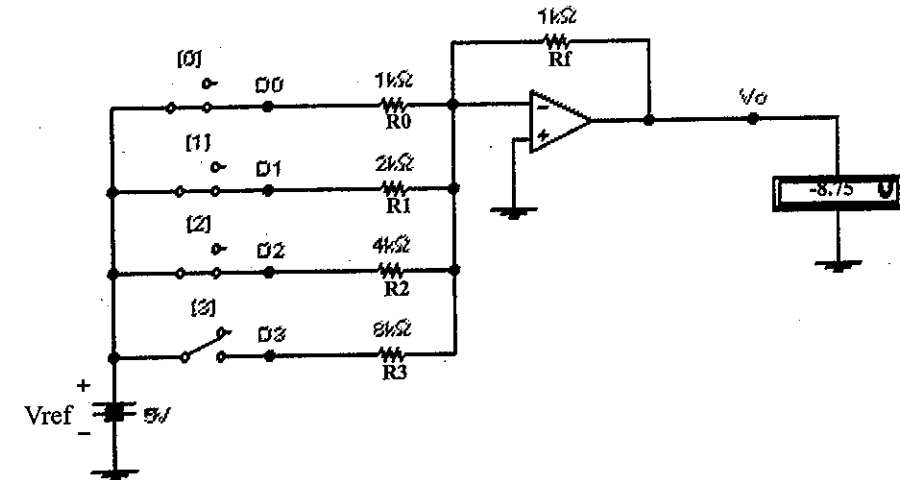
15.2 DİJİTAL/ANALOG ÇEVİRİCİLER

Dijital/analog çeviriciler, girişine uygulanan ikilik sayının büyüklüğü ile orantılı çıkış gerilimi verebilen devrelerdir. Aşağıda dijital bilginin analog bir gerilime çevrilmesinde, en çok kullanılan üç yöntem anlatılmıştır;

- ❖ Paralel girişli D/A çeviriciler,
- ❖ R-2R Merdiven tipi D/A çeviriciler,
- ❖ Entegre¹ elemanlar kullanılarak yapılan D/A çeviriciler.

¹ D/A çeviricilerde kullanılan entegre elemanlar "monolitik" (Monolitik: tek silikon yonga üzerine yerleştirilmiş mikro devrelere verilen addır.) veya "hibrit" (Hibrit: birkaç silikon yonga veya mikro elemanların tek bir paket hâline getirilmesiyle elde edilen elemanlara verilen addır.) şeklinde olabilir.

15.2.1 Paralel Girişli D/A Çevirici



Şekil 15.6 : Paralel girişli dijital/analog çevirici devresi

Dijital sinyallerin, analog sinyallere dönüştürülmesinde en kolay yöntem, paralel girişli D/A çevirici veya diğer adıyla ağırlıklı (değerlikli) direnç D/A çeviriciler kullanmaktır. Şekil 15.6'da, dijital girişlerin değerliklerine göre sıralanmış dirençler üzerindeki gerilimlerin bir toplayıcı opamp devresi ile nasıl gerçekleştirildiği görülmektedir. Opamp, V_{ref} gerilimini, dirençler üzerinden toplayarak çıkışta belli bir gerilime dönüştürecektir. $R_0, \dots, R_3$ dirençlerinin değerleri dijital bilgi girişindeki bitlerin değerliklerine (8421) göre belirlenmiştir. 4 bitten daha fazla giriş gerektiğinde devre yeni direnç değerleri ile genişletilebilir fakat bu da her giriş için değişik değerlerde direnç bulmak zorluğunu beraberinde getirir. En düşük değerlikli girişe D_0 bağlanacak direnç 10 K Ω ise D_3 girişine bağlanacak direnç değeri 1280 K Ω =1,2 M Ω olacaktır. Dirençler arasındaki bu denli dengesiz farklar D/A çeviricinin doğruluğunu düşüreceklerdir.

$$V_0 = -R_f \left(\frac{V_{ref} \cdot D_0}{R_0} + \frac{V_{ref} \cdot D_1}{R_1} + \frac{V_{ref} \cdot D_2}{R_2} + \frac{V_{ref} \cdot D_3}{R_3} \right)$$

Formülün genel hâli $V_0 = -R_f \cdot V_{ref} \left(\frac{D_0}{R_0} + \frac{D_1}{R_1} + \frac{D_2}{R_2} + \frac{D_3}{R_3} \right)$ şeklinde olur.

Örnek: Şekil 15.6'daki devrenin $D_0=1, D_1=1, D_2=1, D_3=0$ dijital girişlerine karşılık, çıkışında kaç volt gerilim oluşur? ($V_{ref}=5$ V, $R_f=1$ K Ω) Değerler yerine konulduğunda;

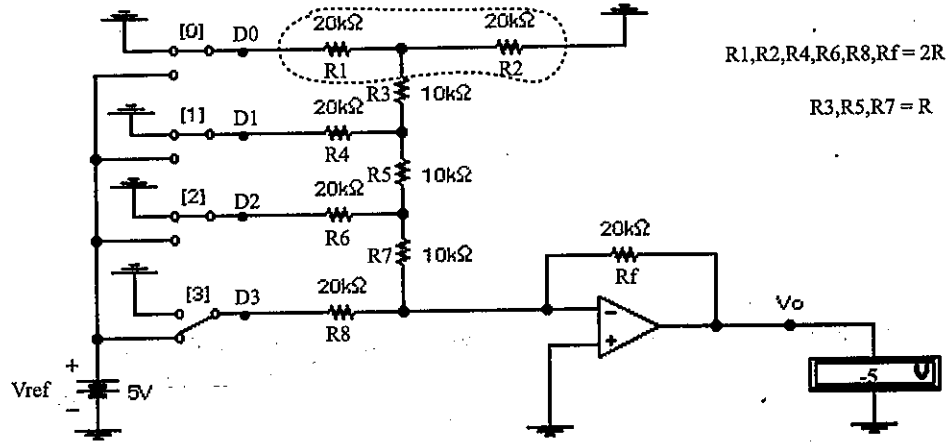
$$V_0 = -1.5 \cdot \left(\frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{0}{8} \right) \text{ bulunur.}$$

Sonra paydalar eşitlenip hesaplandığında;

$$V_0 = -5 \cdot \left(\frac{4}{4} + \frac{2}{4} + \frac{1}{4} + 0 \right) \rightarrow V_0 = -5 \cdot \left(\frac{7}{4} \right) \rightarrow V_0 = -\left(\frac{35}{4} \right) = -8,75 \text{ V olur.}$$

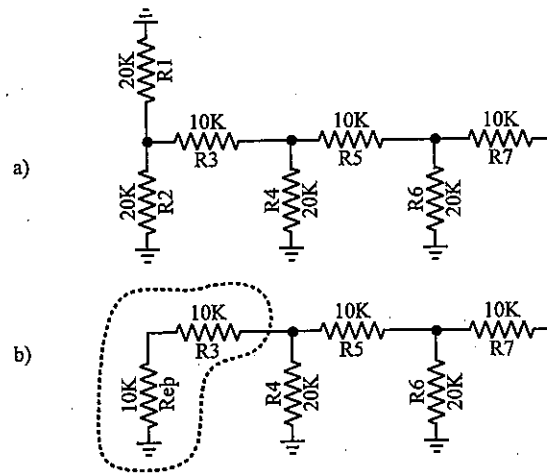
Devrede $D_0=1, D_1=1, D_2=1, D_3=0$ dijital girişlerine karşılık $V_0 = -8,75$ V'luk çıkış gerilimi oluşmuştur.

15.2.2 Merdiven tipi (R/2R) D/A Çevirici



Şekil 15.7 : R-2R Merdiven tipi dijital/analog çevirici devresi

Sayısal bilginin analog bilgiye çevrilmesinde kullanılan diğer bir yöntem R-2R merdiven tipi dijital/analog çevirici devreleridir. R/2R D/A çevirici devresinin, paralel girişli D/A çevirici devresinden avantajlı tarafı, yalnızca iki değerde direncin kullanılmasıdır. Şekil 15.7'de R/2R D/A çeviricinin devresi görülmektedir. Devrenin nasıl çalıştığını anlamak için en değerlikli bit girişinin dışındaki tüm girişlerin sıfır olduğunu kabul edelim (şekil 15.7'deki gibi).



Şekil 15.8 : R/2R D/A çevirici devresinde eş değer direncin bulunması

Şekil 15.8-a'daki devrede bulunan R_3 ve R_4 dirençleri birbirine paraleldir ve eş değer direnci $10\text{ K}\Omega$ dur. Bulunan eş değer direnci R_3 direncine seridir ve toplam eş değer direnç $20\text{ K}\Omega$ dur. Böylece tüm dirençler sıra ile çözümlendiğinde, dirençlerin eş değerinin $20\text{ K}\Omega$ olduğu görülecektir. Devrede eş değer direncin bir ucu şaseye, diğer ucu opamp'ın V girişine bağlı olduğundan (opamp'ların özelliğinden dolayı V girişi V^+ şaseye bağlı olduğu için 0 potansiyele sahiptir.) R_{eq} 'ten herhangi bir akım akmayacaktır ve R_{eq} hesaplamada dikkate alınmaz. Devredeki düğüm noktalarında V_{ref} gerilimi dijital girişlerin değerlikleri oranında bölünür. Değerler yerine konulup hesaplandığında çıkış;

$$V_0 = -R_f \left(D_3 \frac{\left(\frac{V_{ref}}{2R} \right)}{2R} + D_2 \frac{\left(\frac{V_{ref}}{2R} \right)}{2R} + D_1 \frac{\left(\frac{V_{ref}}{2R} \right)}{2R} + D_0 \frac{\left(\frac{V_{ref}}{2R} \right)}{2R} \right)$$

$$V_0 = -R_f V_{ref} \left(\frac{D_3}{2R} + \frac{D_2}{4R} + \frac{D_1}{8R} + \frac{D_0}{16R} \right) \text{ olur.}$$

Paydalar eşitlendiğinde formülün genel hâli;

$$V_0 = -\frac{R_f V_{ref}}{16R} (8D_3 + 4D_2 + 2D_1 + 1D_0)$$

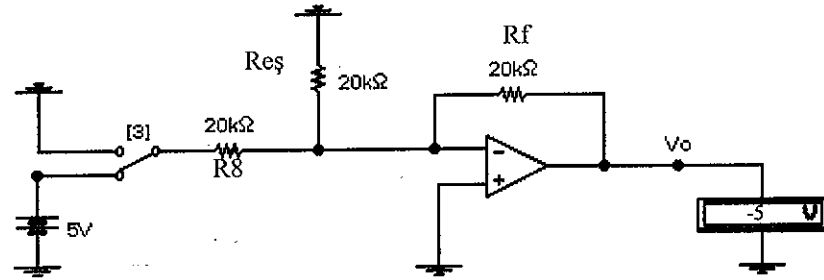
şeklinde olur.

Örnek: Şekil 15.7'deki devrenin $D_3=1$, $D_2=0$, $D_1=0$, $D_0=0$, dijital girişlerine karşılık, çıkışında kaç volt gerilim oluşur? ($V_{ref}=5\text{ V}$, $R_f=20\text{ K}\Omega$, $R=10\text{ K}\Omega$)

Verilen değerler yerine konulduğunda;

$$V_0 = -\frac{20.5}{16.10} (8.1 + 4.0 + 2.0 + 1.0) \quad V_0 = -\left(\frac{5}{8} \right) 8 \quad V_0 = -5 \text{ volt olarak bulunur.}$$

Eğer aynı işlemler yalnızca D_2 girişi 1 yapıldığında hesaplanırsa çıkışın $V_o = -2,5$ volt olduğu, yalnızca D_1 girişi 1 yapılsa çıkışın $V_o = -1,25$ volt olduğu ve yalnızca D_0 girişi 1 yapılsa çıkışın $V_o = -0,625$ volt olduğu görülecektir. D_0, D_1, D_2, D_3 girişlerinin aynı anda değişik olasılıkları hesaplandığında ise 0 ile -5V arasında çok farklı çıkış değerleri ile karşılaşılacaktır.



Şekil 15.9 : Devrenin eş değer direnç bulunduğundan sonraki hâli

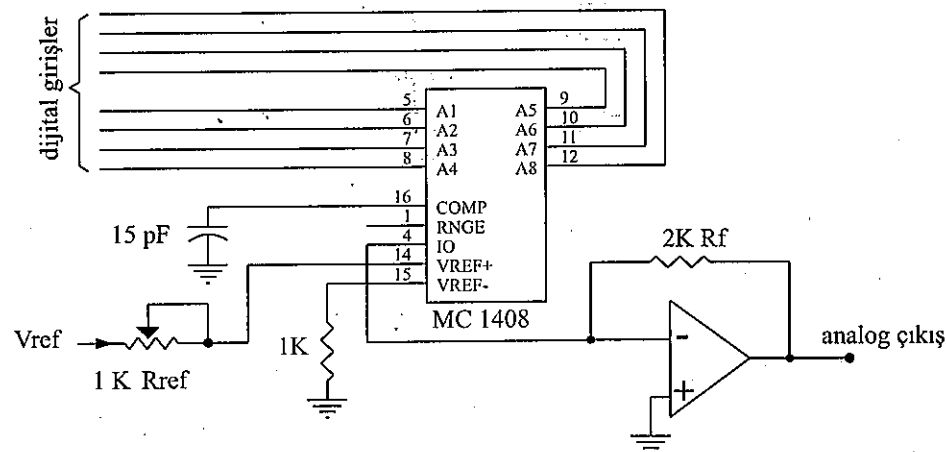
Şimdi, hemen şu soru akla gelebilir: her farklı dijital girişe karşılık farklı bir çıkış değeri elde edilebiliyorsa acaba çıkışta 0 ile -5 volt gerilimleri arasında kaç farklı değer görülebilir? Bu değerlerin sayısı, D/A çeviricilerin çözünürlüğü ile açıklanabilir. D/A çeviricilerde çözünürlük ve buna benzer bilinmesi gerekli bazı kavramların ne olduğu aşağıda verilmiştir.

- ❖ Çözünürlük
- ❖ Doğruluk
- ❖ Lineerlik
- ❖ Çevrim süresi

15.2.3 Entegre D/A Çeviriciler

Dijital sinyallerin analog sinyallere dönüştürülmesinde yaygın olarak kullanılan diğer bir yöntem entegre devre elemanları kullanmaktır. Entegre D/A çeviricilerin çalışma prensipleri, MC1408L entegresi kullanılarak yapılan D/A çevirici üzerinde anlatılmıştır.

MC1408L entegresi 8 dijital girişi olan 16 bacaklı bir entegredir. Daha önceki bilgilerimize dayanarak bu entegrenin çıkışındaki analog gerilim en fazla $2^8 = 256$ basamağa ayrılabilir. Buradan entegrenin $\left(\frac{1}{256}\right) \cdot 100 = 0,39\%$ yani % 0,39 çözünürlükte olduğu söylenir.



Şekil 15.10 : MC1408 entegresi ile yapılan D/A çevirici devresi

Teorik olarak D/A çevirici entegresi kullanılarak yapılan bir devrede çıkışın hesaplanmasında kullanılan formül aşağıdaki gibidir ($R_f = 2\text{ K}\Omega$, $V_{ref} = 5\text{ V}$, $R_{ref} = 1\text{ K}\Omega$).

$$V_o = -\frac{V_{ref}}{R_{ref}} R_f \left(\frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256} \right)$$

$$V_o = -\frac{5}{1} \cdot 2 \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256} \right)$$

$$V_o = -\frac{5}{1} \cdot 2 \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256} \right)$$

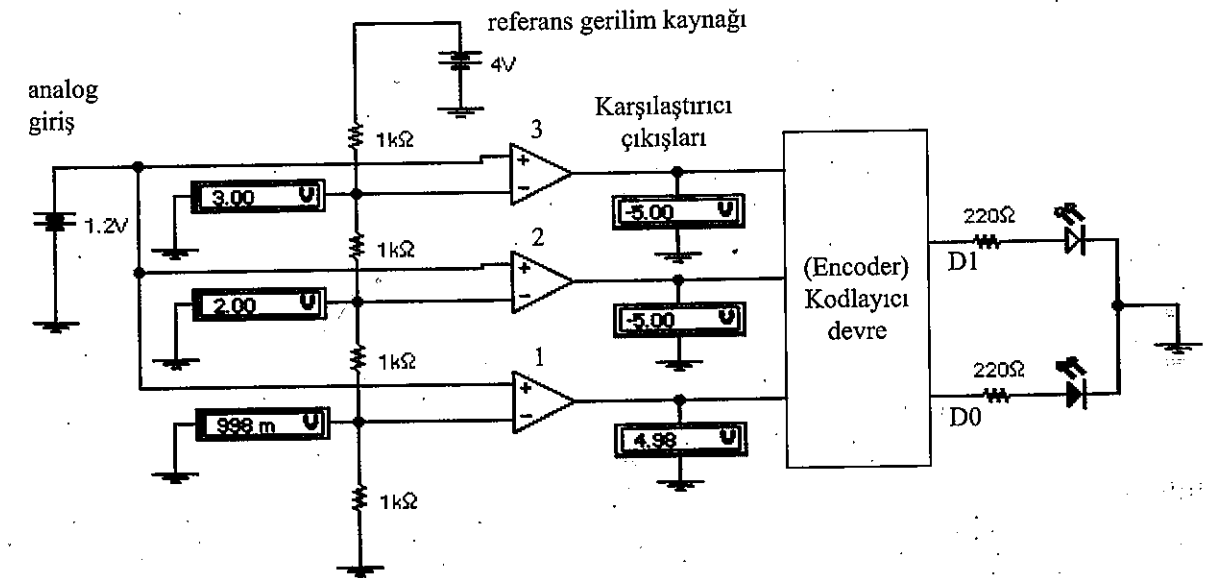
$$V_o = -\frac{5}{1} \cdot 2 \left(\frac{255}{256} \right) \quad V_o = -10 \left(\frac{255}{256} \right) \quad V_o = -9,961\text{ v.}$$

Bütün sayısal girişler 1 iken çıkıştaki gerilimin değeri -9,961 volt oluncaya kadar, V_{ref} , R_{ref} , R_f değerleri ile kalibre edildikten sonra devre kullanılmalıdır.

15.3 ANALOG/DİJİTAL ÇEVİRİCİLER

15.3.1 Kodlayıcı (Encoder) ile A/D Çevirici

Şekil 15.11'de encoder entegresi kullanılarak yapılan A/D çeviricinin prensip şeması görülmektedir. Şekildeki devre iki kısımda incelenebilir. Birinci kısım girişteki analog sinyali, referans gerilimden elde edilen sabit değerlerle karşılaştıran karşılaştırmacılar, ikincisi ise karşılaştırma sonucunda elde edilen değerleri bir encoder entegresi kullanarak dijital forma sokan kısımdır. Birbirlerine seri bağlı dirençlerin her birisine 1 volt gerilim düşecek şekilde referans gerilimin 4 volta ayarlandığını varsayalım. Her düğüm noktasındaki gerilimler 3 V, 2 V, 1 V. olarak görülecektir.



Şekil 15.11 : Encoder ile yapılan A/D çeviricinin prensip şeması

Analog giriş	Karşılaştırıcı çıkışları			Dijital çıkış	
	1.	2.	3.	D1	D0
0 ile 1 volt arası	-5 V	-5 V	-5 V	0	0
1 ile 2 volt	5 V	-5 V	-5 V	0	1
2 ile 3 volt	5 V	5 V	-5 V	1	0
3 ile 4 volt	5 V	5 V	5 V	1	1

NOT: 5 volt dijital "1", -5 volt dijital "0" ile ifade edilir.

Karşılaştırıcıların (comparator) eksi girişlerine, seri bağlı dirençlerden elde edilen gerilimler uygulanmıştır. Artı girişleri birleştirilerek analog sinyalin girişi yapılmıştır. Şekildeki devrede analog girişin bir an için 1.2 volt olduğunu düşünürsek 1. karşılaştırıcının çıkışı +5 volt olurken diğerleri -5 volt olur. Bu değerler kodlayıcı devrede değerlendirilerek giriş gerilim seviyesi ile orantılı bir dijital değer olarak çıkar.

İki bitlik çözünürlük için en az üç adet karşılaştırıcı kullanmak gerekli iken üç bitlik çözünürlük elde etmek için 7 adet karşılaştırıcı kullanmak gereklidir. İstenen çözünürlükte bir A/D çevirici için gereken karşılaştırıcı sayısı $2^n - 1$ formülü ile hesaplanabilir. Eğer 8 bitlik çözünürlükte bir A/D çevirici için $2^8 - 1 = 255$ adet karşılaştırıcı gerekecektir.

Özellikle 3 bitlik çözünürlüğün üzerindeki ihtiyaçlarda, çok fazla sayıda karşılaştırıcıya gereksinim duyulması bu A/D çeviricilerin en önemli dezavantajlarıdır. Diğer dezavantajı ise karşılaştırıcıların çıkışından elde edilen bilgilerin standart encoder için gereken kod bilgisine dönüştürülmesi için ek kapılarla yapılan devrelere ihtiyaç duymasıdır. Fakat bunlara karşılık, girişindeki analog sinyali diğer A/D çeviricilere göre en hızlı şekilde dijital bilgiye dönüştürmesi bunların diğerlerine göre avantajlı yanındır. Hızlarından dolayı bu tip dönüştürücülere "flaş" da denilmektedir.

15.3.2 Dirençli Merdiven Dalgalı A/D Çevirici

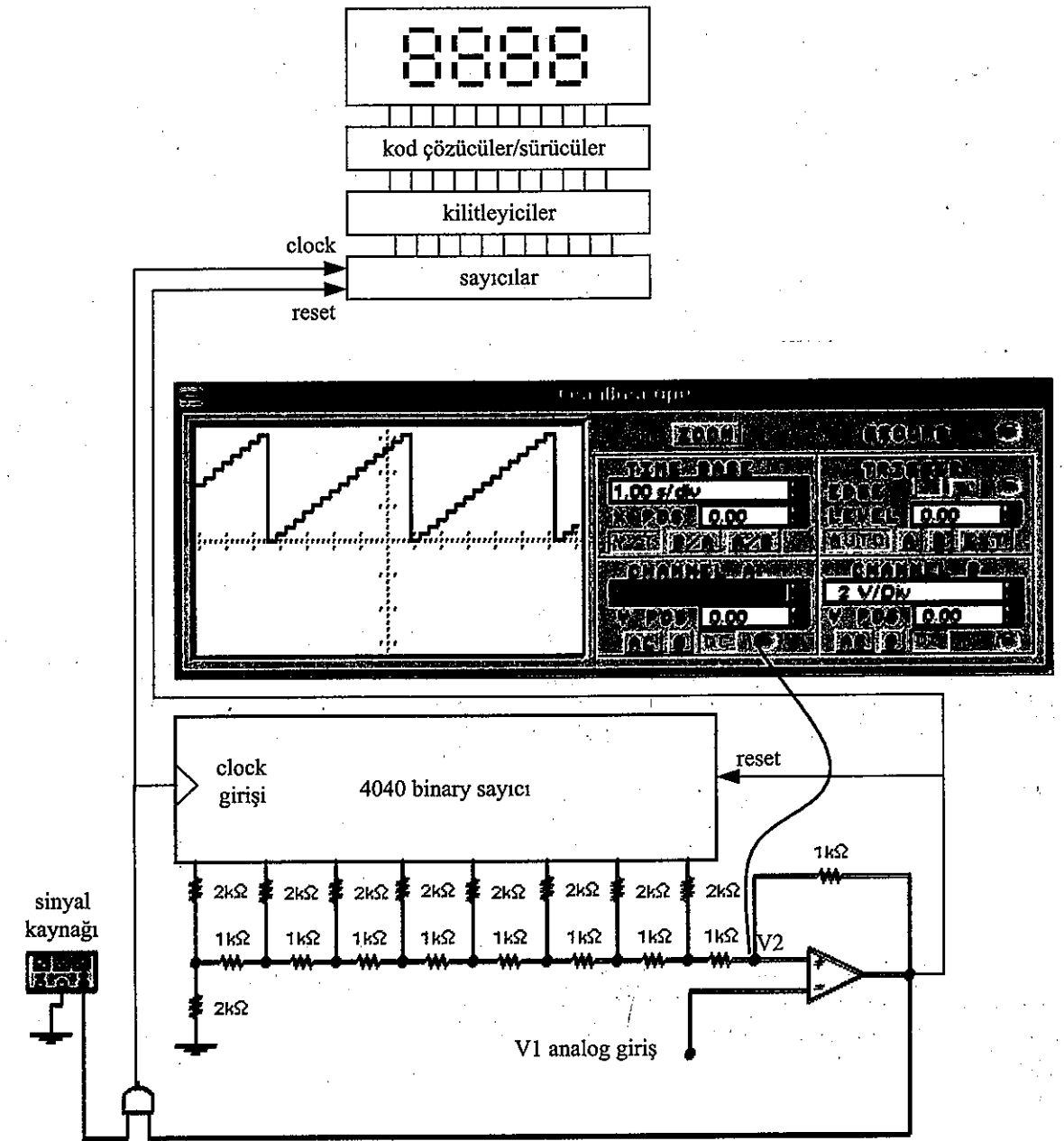
Şekil 15.12'de dirençli merdiven dalgalı analog/dijital çeviricinin prensip şeması görülmektedir.

Devre dört ana kısımdan meydana gelmektedir.

- ❖ Binary sayıcı
- ❖ Gerilim bölücü dirençler
- ❖ Karşılaştırıcı opamp
- ❖ Sayıcı ve display modülü

Binary sayıcı olarak 4040 entegresi kullanılmıştır. Bu entegre 12 bitlik binary çıkış verebilir. Yani 12 bitlik bir binary sayıcıdır. Sayıcının çıkışına bağlanan dirençler üzerinde, sayıcının binary çıkış bilgisine göre değişik gerilim düşümleri olur. Opamp sayıcı çıkışından elde edilen gerilim ile ölçülmek istenen analog gerilimi karşılaştırır. Sayıcı ve gösterge modülü 4040 sayıcı ile aynı clock işaretlerini kullanan yani birlikte hareket eden fakat bağımsız bir sayıcıdır.

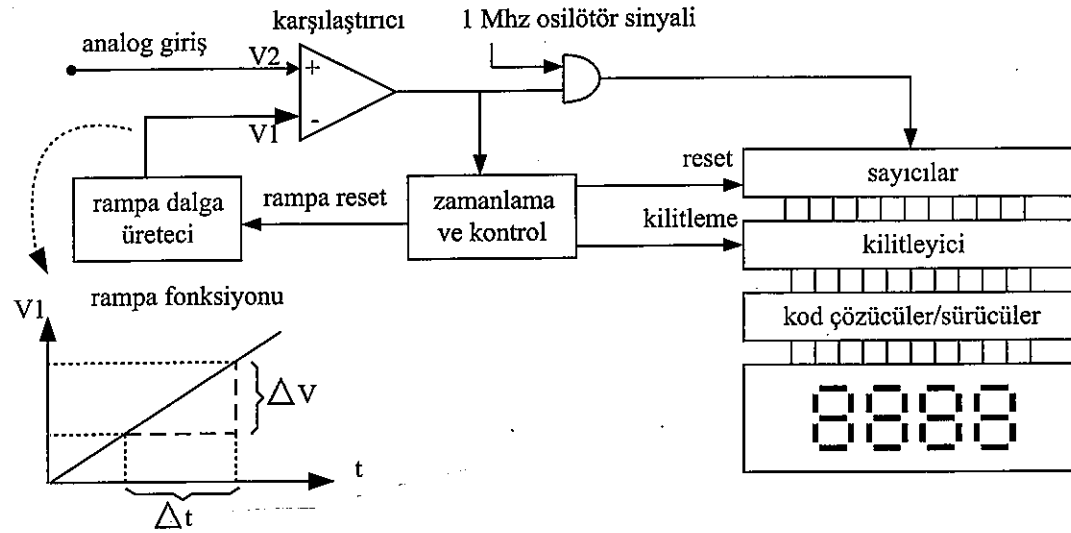
Devrenin çalışmasına gelince; önce 4040 binary sayıcının (00000000)₂ çıkış verdiğini düşünelim. Bu durumda opamp çıkışı pozitif bir değer bu da lojik "1" olduğundan bu çıkış AND kapısına girmektedir. And kapısının diğer girişine clock osilatörün çıkışı bağlanmıştır. AND kapısı clock palslerini geçirerek sayıcıların saymasına yetki verecektir. Bu sırada 4040 binary sayıcı saymaya başladığından opamp'ın V_2 girişinde giderek yükselen bir merdiven dalga görülecektir. Çıkan dalganın şekli osilaskop simülasyonunda görüldüğü gibi olacaktır. Merdiven sinyalinin seviyesi, uygulanan analog sinyal seviyesine eşit oluncaya kadar merdivendeki basamaklar sayıcılar tarafından sayılacaktır. Gerilim seviyeleri eşitlendiğinde opamp çıkışı negatif gerilim üreterek lojik "0" sinyalin oluşmasını sağlayacaktır. Lojik "0" seviyesi AND kapısının kapanmasına yani Osilatör sinyallerinin AND kapısı üzerinden geçmesini engelleyecektir. Böylece hem 4040 binary sayıcının hem de gösterge sayıcısının durmasına sebep olacaktır. Böylelikle displayde görülen değer, uygulanan analog sinyalin karşılığı olacaktır. Bu işlemin sürekli tekrarlanması için aynı çıkış sayıcıyı da resetleyerek yeniden saymasını sağlar. Bu tip devreler çok hızlı değişen veya frekansı yüksek analog sinyalleri ölçmede pek yeterli değildir.



Şekil 15.12 : Dirençli merdiven dalgalı analog/dijital çevirici

15.3.3 Tek Meyil Tamamlaması ile Yapılan A/D Çevirici

Tek meyil tamamlaması ile yapılan A/D çeviricilerin çalışması, prensip olarak merdiven dalgalı A/D çeviricilerin çalışması ile aynıdır. Merdiven dalgalı A/D çeviricilerde merdiven dalga sinyali kullanılırken tek meyil tamamlaması ile çalışan A/D çeviricilerde "Rampa fonksiyonu" kullanılır. Rampa fonksiyonu ile daha hassas karşılaştırmalar yapılabilirken merdiven tipi dalga kullanıldığında A/D çeviricinin çözünürlüğü merdiven sinyalin basamak sayısı kadar olacaktır bu da çeviricinin çözünürlüğünü düşürmektedir.



Şekil 15.13 : Tek meyil tamamlaması ile yapılan A/D çeviricinin prensip şeması

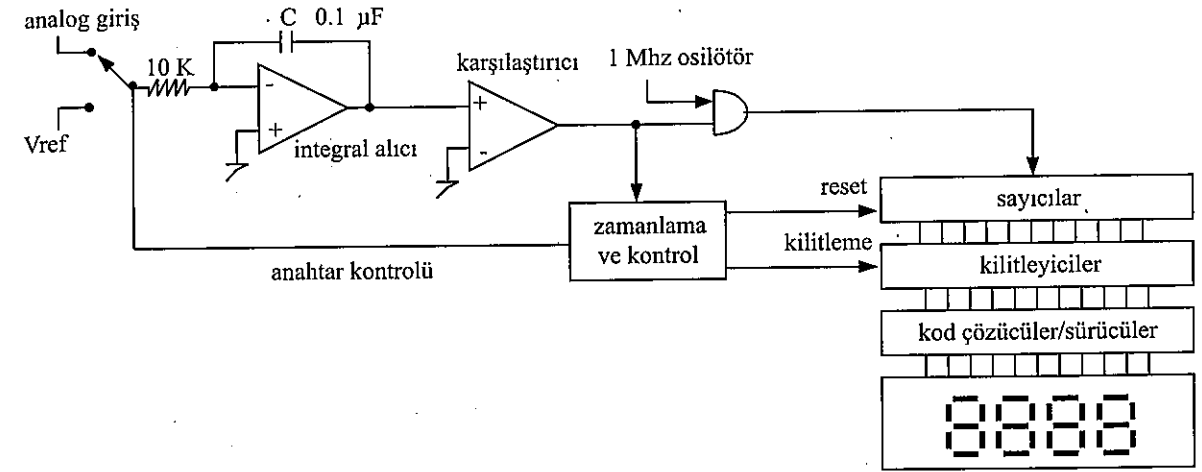
Devre; rampa fonksiyon generatörü, karşılaştırıcı opamp, ana saat sinyali (1 MHz kare dalga sinyal) sayıcılar ve kontrol devreleri şeklinde kısımlardan oluşmaktadır. Devrenin çalışması, karşılaştırıcı girişine uygulanan analog sinyal ile rampa fonksiyonun karşılaştırılması esasına dayanır. Devreye gerilim uyguladığımız anda karşılaştırıcı çıkışı lojik "1" olur ve rampa sinyali yükselmeye başlar. Aynı anda karşılaştırıcı çıkışının lojik "1" olması ile AND kapısı 1MHz'lik osilötör sinyalini geçirmeye başlar ve sayıcı saymaya başlar. Rampa fonksiyonu girişe uygulanan gerilim ile aynı olduğu anda, karşılaştırıcı çıkışı lojik "0" olur ve AND kapısı kapanır, sayıcılar durur. Displayde görüntülen değer analog sinyalin karşılığı olur.

Devrenin çalışmasını daha iyi anlayabilmek için saat frekansının 1MHz ve uygulanan analog sinyalin bir an için 1,2 volt olduğunu varsayalım. Rampa fonksiyonunda eğiminin $\Delta V/\Delta t = 1 \text{ V/msn}$ olduğunu varsayalım. Rampa geriliminin 1,2 volta ulaşması için 1,2 msn zaman geçmesi gerekecektir. Bu süre boyunca sayıcı 1MHz'lik osilötörden 1200 saat darbesi alacak ve göstergede 1200 rakamları belirecektir. Uygun yere virgül konulduğunda okunan değer gerilim cinsinden $1,200 = 1,2$ volta karşılık gelecektir.

Tek meyil tamamlaması ile yapılan A/D çeviriciler, sıcaklık ve giriş gerilimindeki parazitler, rampa fonksiyonunda kararsızlığa neden olduğundan dolayı yeterince hassas ve güvenilir değillerdir. Çift meyil tamamlaması ile yapılan A/D çeviricilerde bu tip olumsuzluklar ortadan kaldırılmıştır.

15.3.4 Çift Meyil Tamamlaması ile Yapılan A/D Çevirici

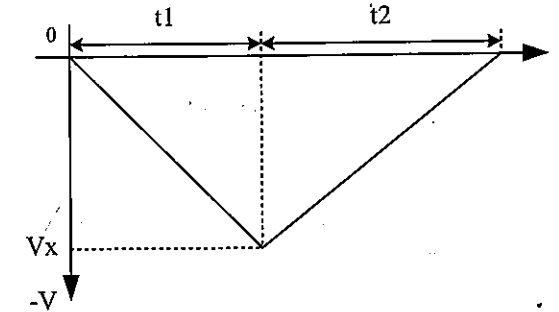
Çift meyil tamamlaması ile yapılan A/D çeviricinin tek meyilden farkı, giriş gerilimini veya referans gerilimini seçen anahtardan başka bir şey değildir. Bu anahtar devreyi giriş gerilimindeki olumsuz etkilerden yalıtıdığı için daha sağlıklı ölçüm sonuçları vermektedir. Tabii bu anahtar, çok hızlı transistörlerle yapılan elektronik bir anahtardır.



Şekil 15.14 : Çift meyil tamamlaması ile yapılan A/D çeviricinin prensip şeması

Devre çalışmasına, Integral alıcı girişine uygulanan analog sinyalin kondansatörü şarj etmesi ile başlar. Bu şarj sırasında integral alıcı devrenin çıkışında düzgün ve sabit eğimli bir rampa oluşturur. Girişe uygulanan pozitif gerilim integral alıcı çıkışında negatif eğimli bir rampa oluşmasını sağlar bu aynı zamanda karşılaştırıcı çıkışını pozitif yapacağından sayma işlemini başlatır. Fakat bu "belirli sabit bir sayım süresince, negatif eğimli rampa üretilmesi için" gerekmektedir. Asıl ölçme henüz başlamamıştır ve sayıcılar asıl ölçme için resetlenir. Şimdiye kadar, sabit sayma süresince girişe uygulanan gerilimle orantılı olarak negatif eğimli bir gerilim elde etmek için yapılanlar anlatılmıştır. Sabit sayma süresince elde edilen rampa gerilim tamamlanınca (V_x) kontrol devreleri integral alıcı devrenin girişindeki anahtarı $-V_{ref}$ gerilimine çevirir. Bu sefer negatif giriş gerilimi integral alıcının pozitif rampa üretmesine sebep olur aynı zamanda dolan kondansatör düzgün ve pozitif eğimli bir rampa oluşturarak deşarj olmaktadır. Integral alıcı çıkışı sıfırı hemen geçince, karşılaştırıcı çıkışı lojik "0" olur ve kontrol devreleri bu geçişi algılayarak elde edilen sayıcı değerlerini tutuculara ve displaylere gönderir. Kontrol devresi bundan sonra yeni bir sayma için tekrar sayıcıları sıfırlar ve anahtarı $V_{giriş}$ gerilimine çevirir.

Devrenin dayandığı mantık, belli bir süre boyunca elde edilen negatif eğimli gerilimi, tekrar sıfıra ulaşmak için pozitif eğimli bir gerilimle karşılamak zorundadır. Bu ifadeler eşitlendiğinde RC sabiti her iki tarafta olduğu için bu elemanlarda oluşan olumsuz etkiler nötralize edilmiş olacaktır. Bu da çift meyil tamamlaması ile yapılan ölçümleri daha kararlı ve güvenli kılar.



Şekil 15.15 : Çift meyilli rampa fonksiyonunun zamana göre değişimi

t_1 = Sabit bir süre

t_2 = Deşarj süresi, aynı zamanda sayıcının sayma süresi

V_x = Kondansatörün t_1 süresince dolduğu gerilim.

$$\frac{V_{giriş}}{RC} t_1 = \frac{V_{REF}}{RC} t_2 \quad V_{giriş} \cdot t_1 = V_{REF} \cdot t_2$$

RC her iki tarafta da olduğundan etkisi sıfırdır. t_1 ve V_{ref} gerilimleri sabit olduğundan buradan çıkan sonuç şudur; t_2 sayıcı çıkışı, $V_{giriş}$ gerilimi ile doğru orantılıdır. Sonuç olarak bilinmeyen giriş gerilimi t_1 süresince integral alıcı devreye uygulanır. Sayıcı sıfıra resetlenir ve integral girişi bu sefer referans gerilime bağlanır. Çıkış sıfıra ulaşmaya kadar geçen süre (t_2) boyunca gerekli sayım yapılır. Elde edilen sayım miktarı giriş gerilimi ile doğru orantılı bir değerdir.

Örneğin; t_1 , 1 MHz'lik osilatör frekansının 1000 sayımına eşit olsun bu da 1 msn eder. $V_{ref} = 1$ V olsun. 3 voltluk bir girişe karşılık t_2 'nin değeri;

$$3V \cdot 1ms = 1V t_2 \quad \text{dir.}$$

Buradan $t_2 = 3$ msn bulunur. 3 msn boyunca 1 MHz'lik osilatör sayıcıyı 3000'e kadar saydırır. Uygun yere virgöl konularak okunan değer 3 volt olarak kalibre edilmiş olur.

Çift meyil tamamlaması ile yapılan A/D çeviriciler tek meyilliye göre daha yavaştır fakat R ve C değerlerinin pasifize edilmesi ile daha doğru ölçümler yapılabilir.

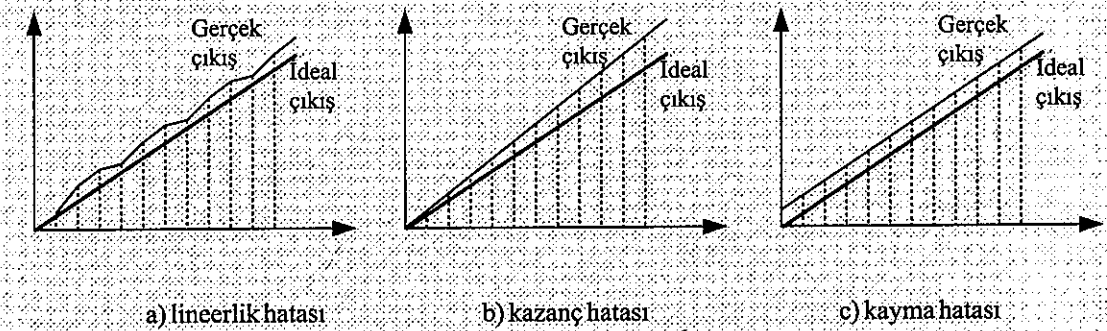


D/A Çeviricilerde Çözünürlük, Doğruluk, Lineerlik ve Çevrim Süresi Kavramları:

Çözünürlük (Hassasiyet): D/A çeviricinin dijital giriş sayısına göre belirlenir. 8 bit dijital girişe sahip bir D/A çeviricinin çözünürlüğü $2^8=256$ 'dır. Yani çıkışında görülecek maksimum gerilim 256 'ya bölünecektir ve her değer bir adıma (basamak) karşılık gelecektir. Örneğin 4 bitlik bir girişi olan D/A çeviricinin çözünürlüğü $2^4=16$ 'dır. Çıkışında, maksimum 5 volt gerilim görülebiliyorsa bu D/A çeviricinin her basamağı ise $5/16=0,312$ volt olacaktır. Eğer 5 voltluk çıkış 8 bitlik girişi olan bir D/A çevirici ile elde edilmiş olsaydı bu D/A çeviricinin çözünürlüğü $2^8=256$ olacağından her basamak $5/256=0,019$ volt olacaktı. Elde edilen değerlere göre 8 girişli bir D/A çevirici, 4 girişli bir D/A çeviriciye göre daha hassas adımları olan çıkış vereceği açıktır. Bu değerler % ile ifade edilecek olursa 4 girişli D/A çeviricinin çözünürlüğü $100/16=6,25$ yani %6,25 iken 8 girişli D/A çeviricinin çözünürlüğü $100/256=0,39$ yani %0,39 olacaktır. 12 girişli D/A çeviricinin çözünürlüğü ise $100/4096=0,024$ yani %0,024 olacaktır.

Doğruluk : Hesap edilen çıkış ile ölçülen çıkış arasındaki benzeşme bize D/A çeviricinin doğruluğu hakkında bilgi verir. Doğruluk bir D/A çeviricinin maksimum çıkış geriliminin yüzdesi ile belirtilir. Örneğin maksimum çıkışın $\pm 0,3$ 'ü gibi. Örneğin maksimum çıkış gerilimi 5 volt olan bir D/A çeviricinin doğruluğu $\pm 0,3$ ise çıkışındaki gerilim 5,003 volt veya 4,997 volt olabilir. Bir D/A çeviricinin ideal doğruluk değeri; çıkışındaki herhangi bir basamak değerinin yarısı, kabul edilebilir bir değerdir (Basamak değeri; birbirini izleyen iki dijital girişe karşılık, çıkıştan elde edilecek iki gerilim arasındaki fark kadardır.). Örneğin $\pm 0,2$ doğruluğu olan bir D/A çeviricinin herhangi bir çıkışı için oluşabilecek hata $0,002 \times 5V = 10$ mV olacaktır. Yani herhangi bir dijital girişe karşılık çıkışında görülen gerilimden 10 mV fazla veya az olabilir. Her basamağı 19 mV olan bir D/A çevirici için 10 mV normal bir doğruluk değeridir. Basamak aralığı 19 mV olan bir D/A çeviricide bu değerin en fazla 9,5 mV değeri idealdir.

Lineerlik (Doğrusallık) : Bir D/A çeviricinin gerçek çıkış eğrisi ile ideal çıkış eğrisinin arasındaki fark bize D/A çeviricinin lineerliği hakkında bilgi verir. Çıkış eğrileri değişik sebeplerden dolayı farklılık gösterebilir. Bu farklılıklar D/A çeviricinin çıkışına bağlı opamp'daki kazanç hatası, akım kaynağındaki hatalar, dirençlerdeki hatalar vb. Bu gibi sebeplerin doğurduğu hataların grafiksel gösterimi aşağıdaki şekillerdedir.



Çevrim süresi : Bir D/A çeviricinin girişindeki dijital bilginin, analog bir gerilime dönüştürülünceye kadar geçen süredir. Eğer girişteki dijital veri, D/A çeviricinin çevrim süresinden daha hızlı değiştirilirse çıkışta doğru değerler görülmeyebilir. Bunun için kullanılan D/A çeviricinin girişine uygulanacak dijital bilgilerin, D/A'nın çevrim süresinden daha az olacak şekilde gönderilmesi gerekir.

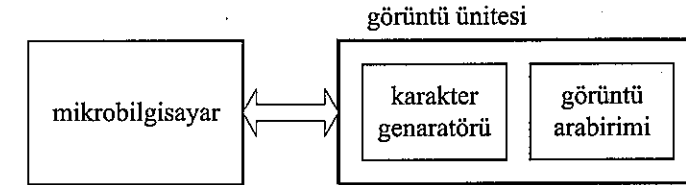
DEĞERLENDİRME ÇALIŞMALARI

- 1) Sayısal sistemler neden analog sinyalleri doğrudan kullanamazlar?
- 2) Sayısal sinyallerin analog sinyallere çevrilmesinde kaç yöntem kullanılmaktadır?
- 3) Sayısal sinyaller analog sinyallere dönüştürülürken ne tür hatalar olabilmektedir?
- 4) Çözünürlük, doğruluk, lineerlik ve çevrim süresi kavramlarını açıklayınız.
- 5) Analog sinyallerin sayısal sinyallere dönüştürülmesinde kaç yöntem kullanılmaktadır?
- 6) Analog sinyallerin sayısal sinyallere dönüştürülürken çevrim süresinin önemi nedir?
- 7) Analog bir sinyal sayısal bir sinyale dönüştürülürken örnekleme frekansı en az kaç olmalıdır.
- 8) Tek meyil tamamlaması ile yapılan A/D çeviricinin prensip şemasını çizerek çalışmasını açıklayınız.

ÜNİTE 16 KARAKTER GENARATÖRLERİ

16.1 KARAKTER GENARATÖRLERİ

Bilgisayarlar, genel olarak bilgileri çok hızlı işleyebilen, sıralayabilen, hesaplayabilen cihazlar olarak ifade edilir. İnsanların, bilgisayarlarda işlenen sonuçları izleyebilmesinin değişik yöntemleri vardır. Bu bilgilerin kağıda dökülmesi, bir görüntü arabiriminde gösterilmesi veya endüstriyel uygulamalarda birtakım işlerin yaptırılması olarak söylenebilir. Bilgilerin, görüntü sistemleri kullanılarak gösterilmesi çoğu sistemlerde kullanılan bir uygulamadır. Bilgisayarlarda işlem görmüş verilerin, display, monitör veya başka görüntü ünitelerinde görülebilecek bir şekle dönüştürülmesi tamamen görüntü ünitesinin kapsamı içerisinde.



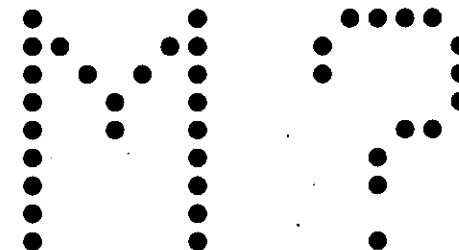
Şekil 16.1 : Görüntü sisteminin temel kısımları, "karakter generatörü" ve "görüntü arabirimi"

Bilgisayar sistemlerinde bilgileri göstermede kullanılan görüntü sistemleri iki temel kısımdan oluşur. Birincisi, görüntü arabirimi diğeri ise karakter generatörleridir. Görüntü arabirimleri için alfasayısal karakterleri (rakamlar, harfler, işaretler, semboller...) üreten devrelere "karakter generatörleri" denir. Görüntü arabirimleri ise kullanıcıların bilgisayarlarda yaptıkları işlemlerin görüntülenmesi amacı ile kullanılır. Görüntü arabirimlerinin değişik yapı ve tipte olanları vardır. Bunlardan bazıları; monitör, likit kristal display, 7 segment display, nokta matris display şeklinde olanlar sayılabilir. Karakter generatörleri ise görüntü arabirimlerinde gösterilecek olan karakterleri üretmek için tasarlanmışlardır. Bir karakter generatörü, karakter üretebilmek için birçok temel devrelerden yararlanır. Bunlardan bazıları; ROM, kod çözücü, kodlayıcı, seri veya paralel kaydırmalı kaydediciler, ring sayıcılar, dijital/analog çeviriciler, frekans bölücü devreler bunlardan bazılarıdır.

16.2 NOKTAMATRİSİ

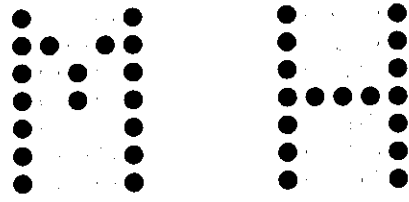
ROM tipi hafıza entegrelerinde, değişik standarttaki (ASCII, EBCDIC...) alfasayısal karakterlerin bit desenleri yüklenmiştir. Bu entegrelerde alfasayısal karakterler, genellikle 7x5 ve 9x7 nokta matrisi düzeninde yüklenmiştir. Piyasada bulunan ve karakterlerin bit deseninin yüklü olduğu ROM entegreler, Amerika ve Avrupa standartlarına göre belirlenmiş karakterlerin desenlerini içermektedirler. Bu entegrelerde Türk alfabesinde bulunan "Ğ, Ü, Ş, İ, Ö, Ç, ğ, ü, ş, ö, ç" gibi karakterler bulunmamaktadır. Türk alfabesinde bulunan karakterleri kullanabilmek için bu karakterlere uygun bit desenlerini EPROM tipindeki entegrelere yükleyerek elde edebiliriz.

Şekil 16.2'de MCM 6576 standart ROM entegresinde yüklü olan "M" harfi ve "?" soru işareti karakterlerinin 9x7 matris tipindeki düzeni gösterilmiştir.



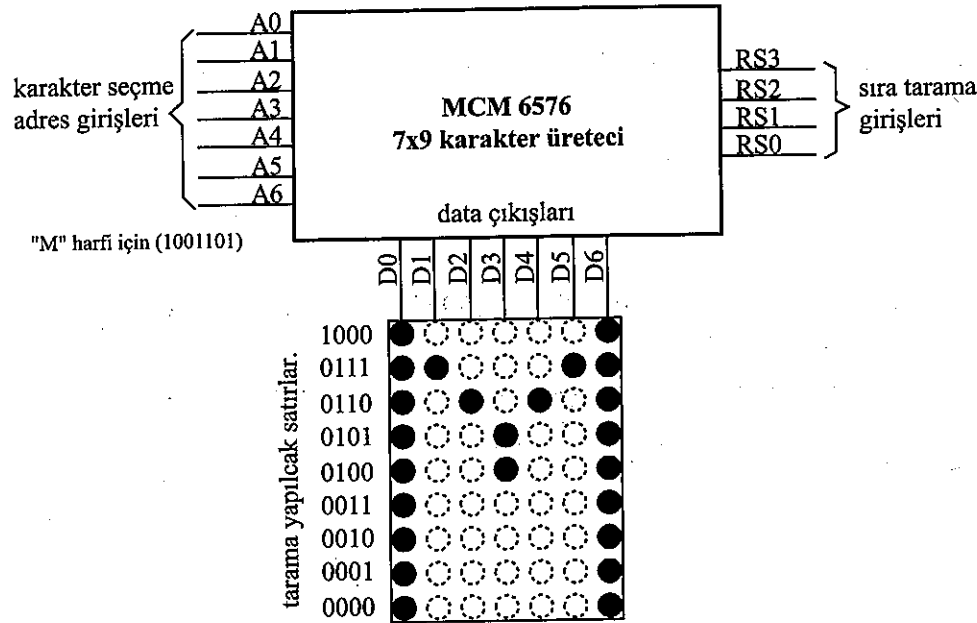
Şekil 16.2 : 9x7 matris düzeninde karakterlerin gösterilmesi

Şekil 16.3'de 7x5 matris düzeninde "M" ve "H" karakterlerinin elde edilmesi gösterilmiştir.



Şekil 16.3 : 7x5 matris düzeninde karakterlerin gösterilmesi

Şekil 16.4'de MCM 6576 karakter generatöründen karakter üretebilmek için gerekli olan bağlantı görülmektedir. Karakter seçme adres girişinden, görüntülenmek istenen karakterin kodu girilir. Örneğin "M" harfi görüntülenmek istenirse $(1001101)_2$ kodu kullanılmalıdır. Hangi karakter için hangi kodun kullanılacağı, kullanılan ROM entegrenin katalog bilgilerinden elde edilebilir. Karakterin tamamının displayde görüntülenmesi için satır tarama girişinin (RS3..RS0) ve display satırlarının aynı anda, en az 25 Hz (İnsan gözü 25 Hz üzerindeki değişiklikleri algılayamaz.) olacak şekilde taraması gerekmektedir. Böylece displayde sabit görünen bir "M" harfi elde edilir.



Şekil 16.4 : MCM 6576 ROM entegreden karakter elde edilmesi

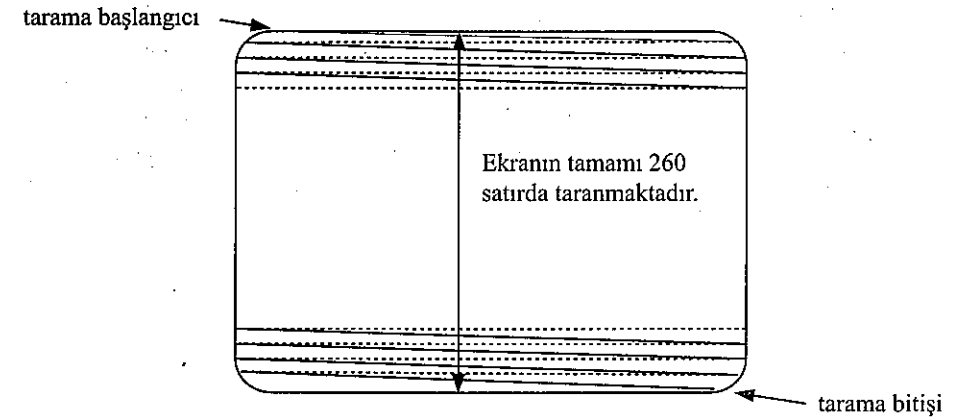
16.3 CRT GÖSTERGELİ KARAKTER GENARATÖRLERİ

Mikrobilgisayar sistemlerinde kullanılan birçok görüntüleme arabirimleri vardır. CRT (cathode ray tube; katot ışın tüpü) nokta matrisli display, 7 segmentli display, LCD (Liquid Crystal Display) bunlardan bazılarıdır.

Bilgisayarlarda kullanılan CRT görüntüleme arabirimleri, televizyon ve osilaskop gibi cihazlarda kullanılan görüntüleme arabirimlerinin çalışma ilkesi ile aynıdır. Video kuvvetlendiricisi aydınlık ve karanlık bölgelerden oluşan görüntüyü elde edebilmek için tarama sırasında ışığın şiddetini, görüntülenecek video

bilgisine göre değiştirir. CRT tipi ekranlarda görüntü satır satır taranarak elde edilir. Elektron tabancasından çıkan ışın görüntülenecek bilginin her satırı için tüpün yüzeyindeki fosforlu tabakayı tarayarak görüntüyü oluşturur. Bir satırın taraması tamamlandıktan sonra hızla ekranın soluna geçer ve bir alt satırdan yeniden taramaya başlanır. Böylece her seferinde ekranın üstünden altına doğru tüm ekran taranmış olur. Böylelikle ekranın tüm yüzeyinde görüntü elde edilir.

Bu yöntemde, kararlı bir görüntü elde edebilmek için yatay tarama frekansının, dikey tarama frekansının ve video bilgisinin senkronize olması gerekir. Bilgisayar sistemlerinde kullanılan CRT'lerde 15600 Hz'lik yatay tarama frekansı ve 60 Hz'lik dikey tarama frekansları kullanılır. 15600/60 hesaplandığında 260 satırlık bir taramanın yapılacağı görülecektir.

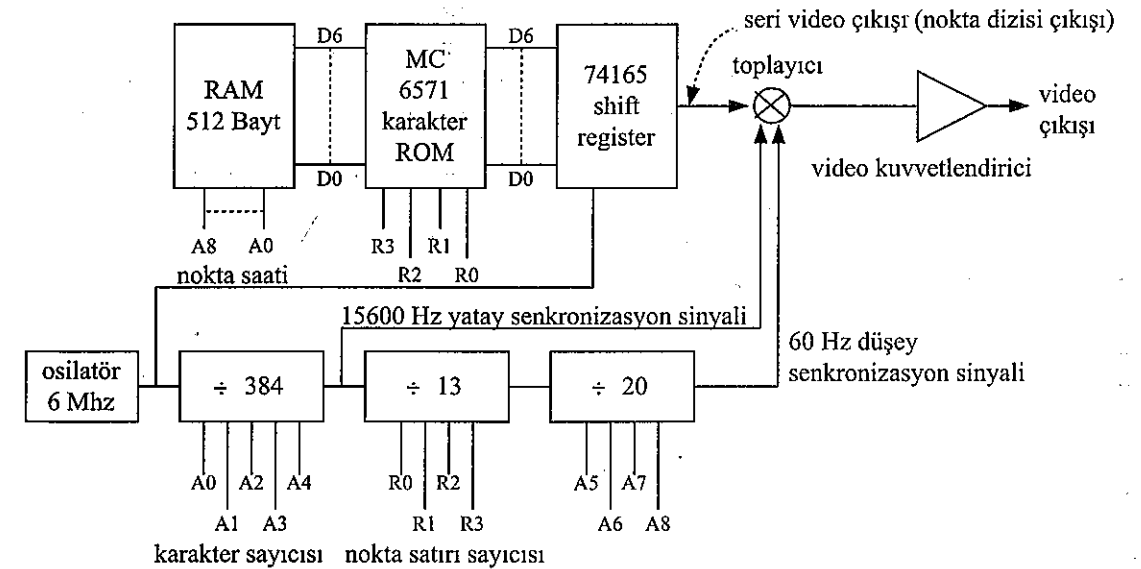


Şekil 16.5 : CRT taraması

16.4 KARAKTER GENERATÖRLERİNİN ÇALIŞMA PRENSİPLERİ

16.4.1 CRT Tipindeki Görüntü Arabirimlerinde Karakter Görüntülenmesi

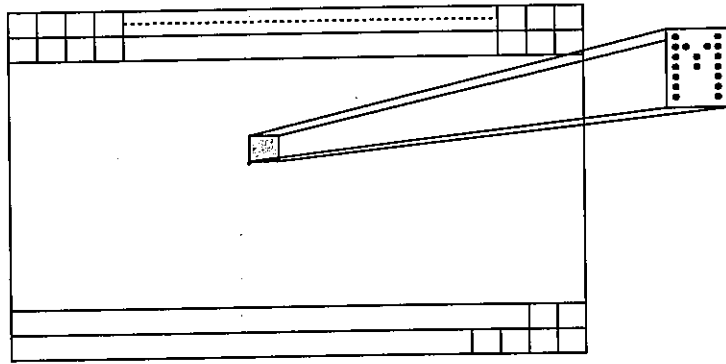
Şekil 16.6'da, CRT tipindeki bir görüntüleme arabiriminde karakterlerin görüntülenmesini sağlamak için kompozit video sinyalinin nasıl elde edildiğini gösteren devre şemasının blok diyagramı görülmektedir.



Şekil 16.6 : CRT karakter generatörünün blok diyagramı

RAM'da görüntülenecek karakterin kodu bulunmaktadır. Bu kod karakter üretici tarafından okunarak karakterin bit desenine ulaşılır. Bu bit deseninde, ilk satırın noktaları kayma kaydedicisine yüklenir ve kayma kaydedicisinden her seferinde bir nokta olacak şekilde dışarı çıkarılır. CRT tipi ekranlarda katod ışını, ekran boyunca kayarken ilk karakterin üst satırının nokta dizisini üretir. Hemen ardından ikinci karakterin üst satırı elde edilerek nokta dizisi şeklinde üretilir. 6571 ROM entegresindeki karakterlerin bit desenini elde etmek ve herhangi bir satırdaki karakterlerin tamamının bit desenini ekranda görüntüleyebilmek için her satırın 13 kez tekrarlanması gerekmektedir. Böylelikle her satırdaki 32 karakterin tamamını görüntülemiş olacaktır. Birinci satırda bulunan karakterlerin bit deseni tarandıktan sonra bir alt satıra geçilerek 16 satırın tamamı taranır (20 ile bölme devresi 16 satırı taramak için gerekli frekansı böler, geriye kalan 4 satırlık süre ise geri dönüş zamanı için kullanılmıştır.).

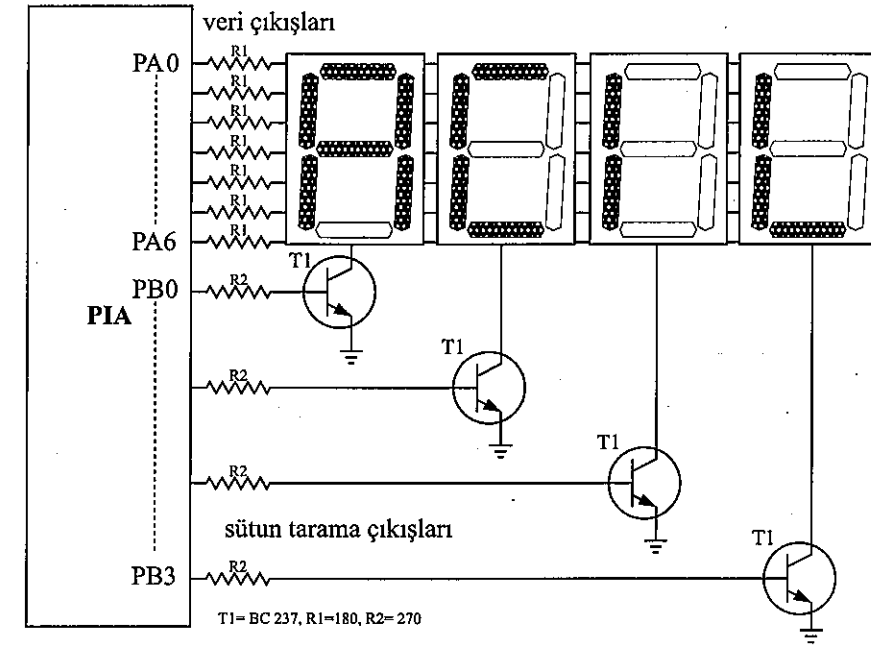
Bu taramalar böylelikle devam ederek ekranda görüntü elde edilmiş olur. Mikrobilgisayar ekrandaki görüntüde değişiklik yapmak istediğinde RAM'daki ASCII kodları değiştirmesinin hemen ardından, görüntüleme sistemi yeni bilgileri ekrana basmak için gereken değişiklikler bir sonraki taramada bitirilmiş olacaktır. Burada 32x16 (16 satır 32 sütun) modunda çalışan siyah/beyaz bir görüntüleme arabiriminin çalışma şekli anlatılmıştır. Değişik satır ve sütun sayısına sahip görüntüleme sistemlerin bulunduğu gibi renkli ve grafik modunda çalışan görüntüleme birimlerinin bulunduğu ve bunların çalışma şekillerinin çok daha detaylı ve karmaşık olacağı açıktır.



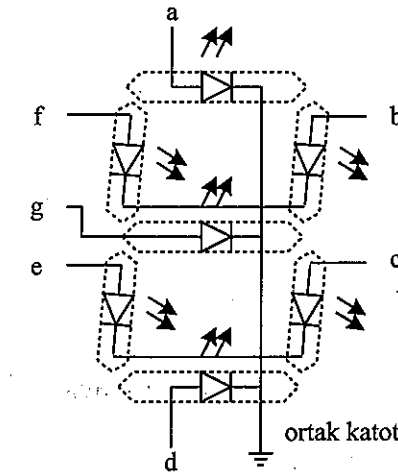
Şekil 16.7 : CRT ekranında bir karakterin bit deseni

16.4.2 7 Segment ve Nokta Matris Displaylerde Karakter Görüntülenmesi

Mikrobilgisayar sistemlerinde en çok kullanılan görüntüleme sistemleri, 7 segment display veya nokta matris displayler kullanılarak yapılan görüntüleme sistemleridir. Bu tipteki görüntüleme arabirimleri, daha çok endüstriyel çalışmalar için geliştirilen mikrobilgisayar uygulamalarında kullanılmaktadır. 7 segment veya nokta matris displaylerin likit kristal tipinde olanları da vardır. Şekil 16.8'deki devre 7 segment displayler ile yapılan görüntüleme aygıtının temel şemasıdır. 7 segment displayler ile elde edilebilecek karakter sayısı azdır. Örneğin 7 segment displayler ile "M", "G", "K", "T" gibi daha bir çok karakteri göstermek için yetersiz kalır. Daha fazla karakter görüntüleyebilmek için nokta matris displayler kullanmak gerekir.



Şekil 16.8 : Multiplexer kontrollü 7 segment display devresi

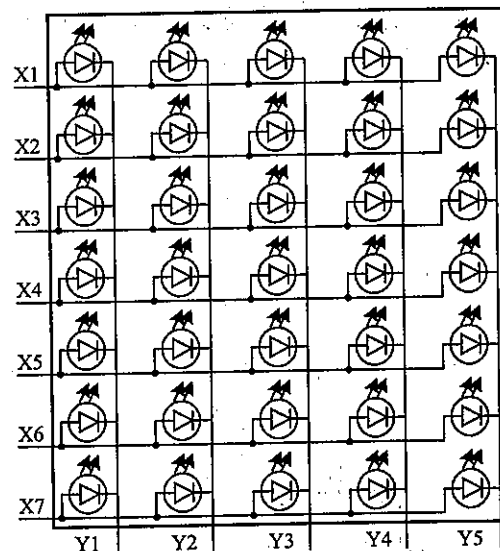


Şekil 16.9 : Ortak katotlu 7 segment display iç bağlantısı

Şekil 16.8'deki görüntüleme sistemlerindeki displaylerde, görüntü oluşturabilmek için görüntülenmesi istenen bilgilerin displaylerde sıra ile görüntülenmesi gerekir. PIA, paralel giriş/çıkış işleminde kullanılan herhangi bir veri portu olabilir. Devrede kullanılan PIA'nın, iki adet (A ve B portu) 8 bitlik paralel çıkış portu bulunmaktadır. PA0....PA6 port çıkışında, görüntülenecek verinin kodu, displaylere gönderilir. Devredeki tüm displaylerin veri girişi bağlantıları paraleldir yani 1. displayin a,b,c,d,e,f,g bacağı ikinci displayin a,b,c,d,e,f,g, bacakları ve diğerleri aynı şekilde birbirine bağlıdır. Tüm displayler paralel bağlantılı olduğu için PA0..PA6 portundaki veriler hepsinde görüntülenecektir. Fakat tüm displaylerin (displayler ortak katotlu) katot bağlantıları bir transistör ile kontrol edilmektedir. Hangi displayin etkin olması istenirse o displayin katodunu kontrol eden transistörü ilettime geçirmek yeterli olacaktır. Aynı anda bu karakterin diğer displaylerde görüntülenmesini önlemek için diğer transistörler de kesimde tutulur.

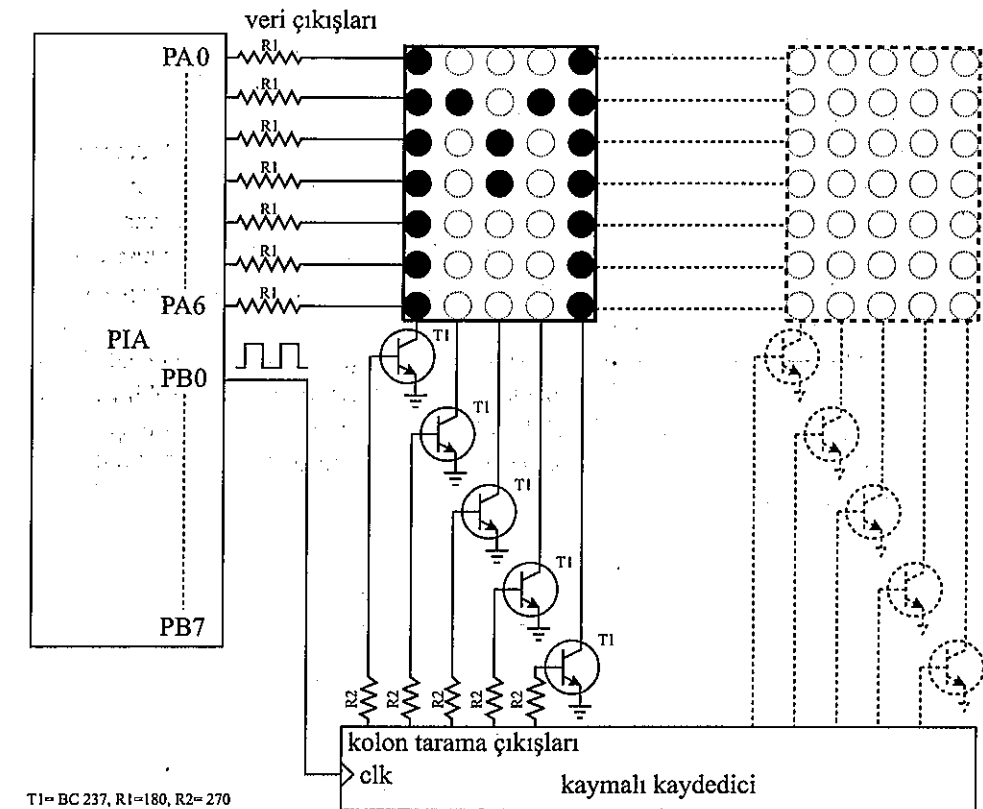
Şekil 16.8'deki devrede eğer “ACIL” mesajı yazdırılmak istenirse, ilk displayde “A” harfinin görüntülenmesi için gerekli bit deseni A portuna (PA₀, PA₇) yüklenir. 1. displayin “A” harfini gösterebilmesi için katot ucunda bağlı olan transistörün ilettime geçirilmesi gerekir. Yalnızca 1. displayin katoduna bağlı olan transistör iletimde olduğu için diğer displaylerde “A” harfi görüntülenmeyecektir. Ardından 2. displaye “C” harfinin kodu gönderilir ve bu seferde 2. displayin katoduna bağlı transistör ilettime geçirilir. Böylece istenen tüm veriler sıra ile displaylere gönderilmiş olur. Eğer bu işlem saniyede 25 kereden fazla tekrar ederse displaylerin tümünde, duran bir “ACIL” yazısı görüntülenecektir. Bu tip bir tarama “multiplexer kontrollü tarama” diye adlandırılır. Burada dikkat edilecek nokta gönderilen verinin her seferinde aynı displaye gelecek şekilde katoduna bağlı transistörün ilettime geçirilmesidir.

Nokta matris görüntüleme arabirimleri: daha önce değinildiği gibi 7 segment displayler ile sınırlı sayıda karakter kullanılmaktadır. Bu yetersizlik, nokta matris displayler ile aşılabılır. Şekil 16.8'deki devre nokta matris displayler ile yeniden düzenlendiğinde şekil 16.11'deki gibi olur. Nokta matris displaylerdeki her sütun bağımsızdır ve her sütundaki ledlerin katotları veya anotları ortaktır. Şekil 16.10'da, 7x5 tipinde nokta matrisin displayin iç bağlantısı görülmektedir.

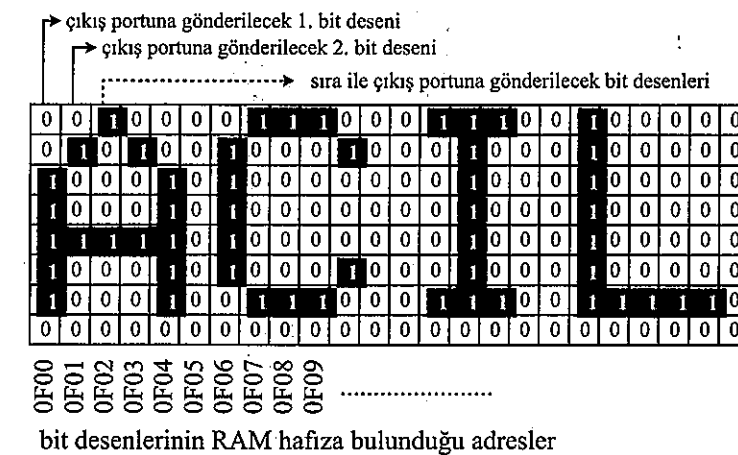


Şekil 16.10 : 7x5 tipindeki ortak katot uçlu nokta matris displayin iç bağlantısı

Nokta matris tipindeki displayler kullanıldığında her sütun ayrı ayrı transistör ile sürülmelidir. Böyle olunca her 7x5 tipindeki bir display için toplam 5 tane transistör kullanılması gerekir. Nokta matris displayler kullanıldığında çok sayıda sütun ve beraberinde transistör olacağı için aktif edilmek istenen sütundaki transistörün iletme geçirilmesinde kayan nokta kaydedicilerini kullanmak, 8 bitlik portu kullanmaktan daha avantajlıdır. 7x5 tipindeki nokta matris display kullanarak yapılmak istenen 8 karakterli bir görüntüleme arabirimi için yaklaşık (her displayin 5 sütunu var x 8 bitlik paralel port=40) 40 adet port çıkışına bu da $40/8=5$ adet paralel porta ihtiyaç var demektir. Mikrobilgisayarlar da zaten sınırlı sayıdaki 8 bitlik paralel portları bu işe ayırmak akıllıca değildir. En iyisi her sütunu sıra ile etkinleştirmek için kayan noktalı kaydedici veya başka tip çözümler düşünülmelidir. Sütunları taramak için kayan noktalı kaydedicinin clock girişine her seferinde bir clock sinyali vermek gerekir. Bunun için B portunun herhangi bir çıkışı kullanılabilir (PB₀). Burada dikkat edilmesi gereken nokta, veri çıkış portundan gelen bilginin her seferinde aynı sütunda kalabilmesi için clock sinyalini dikkatli göndermek gereklidir yani senkron çalışmalıdır.



Şekil 16.11 : Mikrobilgisayar sistemlerinde nokta matris display kullanımı



Şekil 16.12 : 7x5 matris displayler için karakterlerin RAM'daki görüntüsü

Bu devrede herhangi bir anda yalnız bir sütun etkin olacağı açıktır. Eğer görüntülenecek karakterlerin bit deseni saniyede 25 kez displaylere gönderilirse bu displaylerdeki görüntüntülerde duruyor izlenimi verecektir. Dot matris displayler kullanılarak yapılan görüntüleme sistemlerinde kullanılan yöntem; displaylerde görüntülenmek istenen karakterlerin bit desenleri şeklinde kullanımıdır. ROM entegredeki karakter tablosundan okunarak RAM hafızaya aktarılır (display hafızasına). Dot matris displaylere gönderilecek veriler RAM hafızadan çıkış portuna aktarılır. Tüm bu işlemleri sıra ile yazmak gerekirse aşağıdaki şekilde olmalıdır.

- ❖ Görüntülenecek karakterlerin bit desenlerini ROM tablosundan okuyarak RAM hafızaya kopyalanır,
- ❖ RAM hafızadaki karakterlerin bit desenindeki her byte sıra ile çıkış portuna aktarılır.
- ❖ Displayde görüntüleneni veri çok kısa bir süre bekletildikten sonra ikinci veri çıkış portuna aktarılır.
- ❖ RAM'daki bit desenlerinin tamamı dislaye gönderildikten sonra bu işlem en az saniyede 25 kez tekrarlanır.
- ❖ RAM hafızadaki veri değişinceye kadar dislaydeki görüntü sürekli tarandığı için kalacaktır.

Buraya kadar 7 segment ve nokta matris displayler ile yapılan temel görüntüleme devreleri anlatılmıştır. Bu devrelerin pratikte kullanılması mümkündür, fakat çok kullanışlı değildir. Bu devrelerin mikrobilgisayarlarda kullanılması için özel hazırlanmış yazılımlara da ihtiyacı vardır. Ayrıca bu tip görüntüleme sistemlerinde, displaylerdeki görüntünün sürekli kalabilmesi için mikroişlemcin bu işe büyük zaman harcaması gerekir. Pratikte daha profesyonel çözümler mevcuttur. Örneğin, Intel'in 8279 entegresi hem klavye hem de görüntüleme arabirimini içinde barındıran ve çalışmasını mikroişlemciden bağımsız yürüten bir entegredir. Displayler ve klavye tuşları için gerekli tüm kontrol ve tarama sinyallerini kendisi üretebilmektedir. Programcının, görüntülenmesini istediği mesajı 8279 entegresinin görüntü hafızasına yazması yeterli olacaktır. Entegrenin nasıl kullanıldığı ve bacak bağlantıları katalog bilgilerinden bulunabilir.

DEĞERLENDİRME ÇALIŞMALARI

- 1) Mikrobilgisayarlarda kaç çeşit görüntüleme arabirimi vardır?
- 2) Görüntüleme arabirimlerinde neden karakter genaratörleri kullanılır?
- 3) CRT'lerde görüntü nasıl oluşturulur? Çalışma prensibini anlatınız.
- 4) Multiplexer kontrollü 7 segment görüntüleme arabiriminin çalışma prensibini anlatınız.
- 5) Görüntüleme arabirimlerindeki "tarama" kavramını anlatınız.
- 6) Görüntü hafızası ne demektir? Görüntü hafızasındaki veriler görüntüleme arabirimlerine nasıl aktarılır?
- 7) Taramalı çalışan görüntüleme arabirimlerinde insan gözü sn'deki en az kaç tekrarı titreşimsiz görüntü olarak algılayabilir?

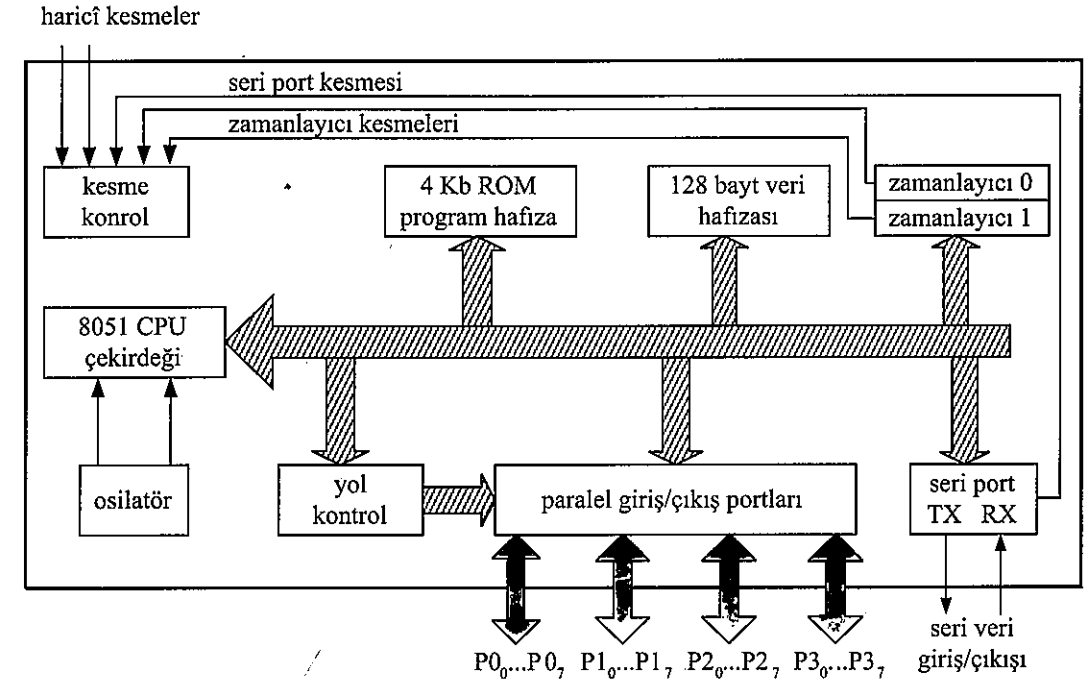
ÜNİTE 17 MİKRO DENETLEYİCİLERLE KONTROL

17.1 MİKRO DENETLEYİCİLERİN YAPISI

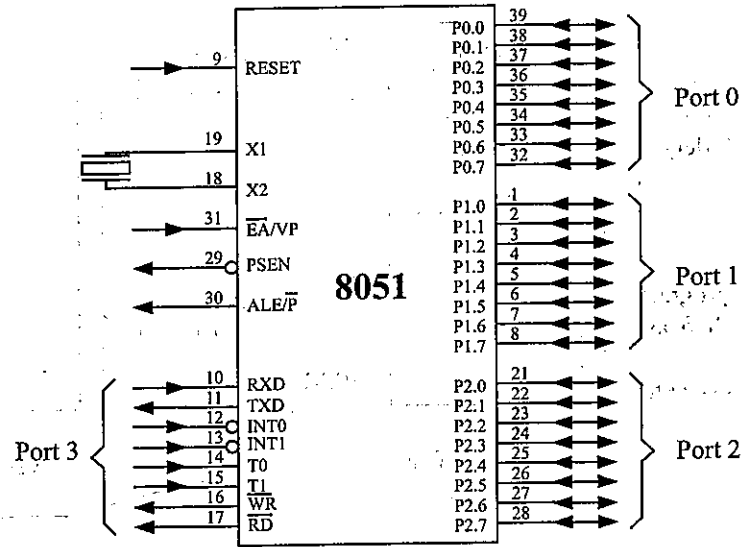
Mikroişlemcilerin özel bir türü olan mikrodenetleyiciler özellikle endüstriyel uygulamalarda sıkça kullanılmaktadır. Mikrodenetleyicilerin kullanım yerlerinden bazıları şunlardır; yazar kasalar, fotokopi makineleri, faks cihazları, yazıcılar, dâhilî telefon santralleri, TV, teyp, CD, oyuncaklar, robotlar, kameralar. Mikrodenetleyiciler günlük hayatımızda kullandığımız elektronik cihazlar ve birçok endüstriyel uygulamalarda denetim ve kontrol amaçlı olarak kullanılmaktadır.

8051 kodlu mikrodenetleyici, Intel firmasının 1980 yılında piyasaya sunduğu 8 bitlik bir mikrodenetleyicidir. Bu mikrodenetleyicinin piyasada çok tutulmasından dolayı birçok üretici firma aynı yapı üzerine çok değişik özellikleri olan mikrodenetleyiciler geliştirip piyasaya sunmuşlardır. Bu ürünlerin birbirlerinden çok farkı olmasa da ROM/EPROM kapasiteleri, hızı, giriş/çıkış sayısı, analog/dijital girişleri, zamanlayıcıları gibi alt elemanlarında farklılıklar vardır. Mikrodenetleyicilerin mikroşlemcilerden farkı, içerisinde ROM/EPROM, RAM, paralel ve seri giriş/çıkış üniteler ve zamanlayıcı/sayıcı gibi elemanlar içermeleridir. "MCS" kelimesi, içerisinde birçok eleman bulunduran mikrodenetleyicilerin tanımlanmasında kullanılan genel bir ifadedir. MCS, çekirdeği 8051 mikrodenetleyicisi olan Intel mikrodenetleyici ailesindeki tüm mikrodenetleyiciler için kullanılmaktadır.

17.1.1 İç Yapısının Blok Diyagram Üzerinde İncelenmesi



Şekil 17.1 : MCS-51 serisi mikrodenetleyicisinin iç mimarîsi

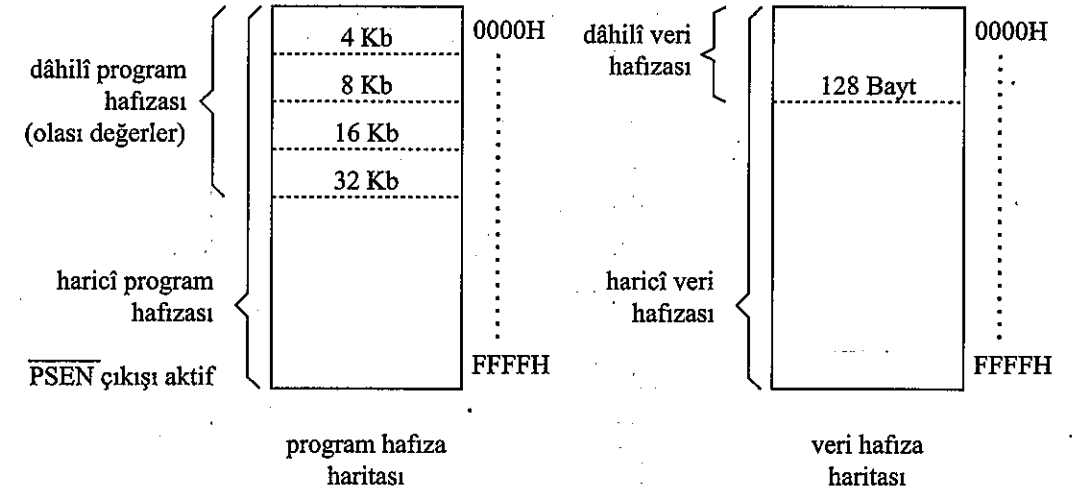


Şekil 17.2 : MCS-51 Mikrodenetleyicisinin bacak bağlantıları

Mikrodenetleyicilerin temel özellikleri şunlardır;

- ❖ Kontrol uygulamaları için özelleştirilmiş mikroişlemci
- ❖ Program hafızası (ROM/EPROM)
- ❖ Veri hafızası (iç ve dış veri hafızası)
- ❖ Çift yönlü paralel giriş/çıkış portları (8 bitlik)
- ❖ Çift yönlü seri giriş/çıkış portu
- ❖ Zamanlayıcı/sayıcı
- ❖ Kesme girişleri

Program hafızası: Program hafızası yalnızca okunabilir hafızalardır. Program hafızasında mikrodenetleyicinin istenen şekilde çalışmasını sağlayan programlar bulundurulur. Mikrodenetleyici teoride program hafızası için toplam 64 Kb'lık büyüklüğündeki bir hafıza alanına erişebilir, fakat kullanılan mikrodenetleyicide, bu büyüklükte bir program hafızası olmayabilir. Piyasada bulunan MCS-51 serisi mikrodenetleyicilerde, modeline göre 4Kb, 8Kb, 16 Kb, 32 Kb büyüklüğünde program hafızası içermektedir. (MCS-51AH mikrodenetleyicisinde 4Kb ROM bulunmaktadır.). Mikrodenetleyicilerin içerdiği program hafızasından daha büyük bir program hafızası kullanmak istendiğinde haricî bağlanan ROM/EPROM gibi hafıza elemanları PSEN okuma sinyali ile kontrol edilebilir. Eğer MCS-51 ile birlikte haricî program hafızası kullanılacak ise EA girişi "0" yapılarak mikrodenetleyiciye bildirilir. Böylece haricî program hafızasının varlığı mikrodenetleyiciye bildirilerek program hafıza erişimlerinin haricî hafızadan yapılması sağlanır. EA girişi "1" yapıldığında mikrodenetleyici kendi içerisindeki ROM hafızaya erişecektir.



Şekil 17.3 : Program ve veri hafıza haritaları



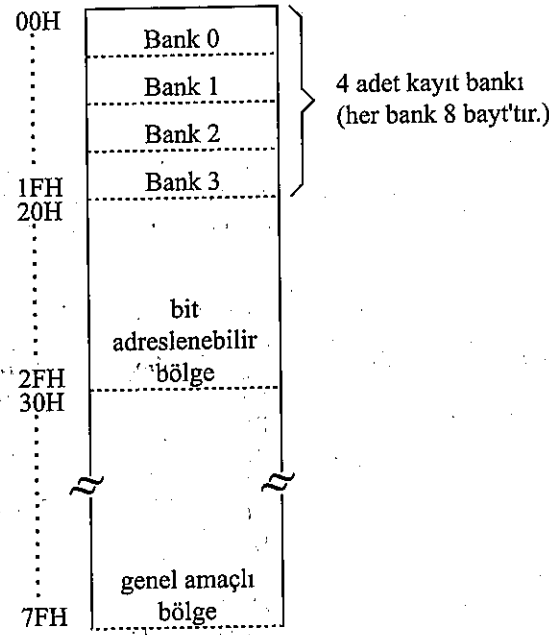
MCS-51 Mikrodenetleyicilerin Vektör Tablosu:

MCS-51 mikrodenetleyici ailesindeki mikrodenetleyiciler resetlendiğinde veya sisteme ilk defa enerji uygulandığında başlangıç programını yürütmek için belirli 0000H adresinden başlar. Bu başlangıç adreslerinin bulunduğu alana daha önce vektör tablosu demiştik. Şimdi bu denetleyiciler resetlendiğinde veya herhangi bir kesme sinyali uygulandığında, programın dallanacağı vektör adresleri aşağıdaki tabloda görülmektedir.

Kesmeler	Kesme vektörü
Seri port	0023H
Zamanlayıcı 1	001BH
Haricî 1	0013H
Zamanlayıcı 0	000BH
Haricî Kesme 0	0003H
Reset	0000H

Veri hafızası: MCS-51 ailesinde verileri yazabilmek için oku/yaz hafızalar kullanılabilir. MCS-51 serisi mikrodenetleyicilerinde bu oku/yaz belleğin teorik olarak toplam alanı 64 Kb olabilir. MCS-51 ailesindeki mikrodenetleyicilerde genellikle dâhilî veri hafızası 128 veya 256 byte olmaktadır. Tasarımcı isterse bu alanı dışarıdan bağlayacağı haricî RAM hafızalar ile arttırabilir (Eklenilecek toplam haricî RAM kapasitesi 64 Kb ile sınırlıdır.).

İki kısım veri hafızası mevcuttur; tıpkı program hafızasında olduğu gibi dâhilî ve haricî veri hafızası bulunmaktadır. Dâhilî veri hafızası değişik kullanımları için birtakım bölümlere ayrılmıştır. Şekil 17.3'te 128 byte'lık dâhilî veri hafızasının kısımları görülmektedir.



Şekil 17.4 : Dâhilî veri hafızasının kısımları

Paralel giriş/çıkış portları: MCS-51 mikrodeneleyicilerin 4 tane paralel giriş/çıkış portu vardır. Bu portlara veri yazmak ve okumak için SFR (special function registers; özel fonksiyon kaydedicileri) kayıtları kullanılır. Port 0 portu haricî program veya veri hafızasına erişimde düşük değerlikli adres/veri yolu ($A_7...A_0$, $D_7...D_0$) bilgisi olarak ta kullanılır. Port 2 portu haricî program veya veri hafızasına erişimde yüksek değerlikli adres yolu ($A_{15}...A_8$) bilgisi olarak ta kullanılır. Port 1 ve Port 3 genel amaçlı olarak kullanıldığı gibi Port 3 alternatif olarak özel çıkışlar vermektedir.

Tablo 17.1 : Port 3'ün alternatif fonksiyon özellikleri

Bacak numarası	Alternatif fonksiyon	Sembolü
P3 ₀ (10)	Seri giriş	RxD
P3 ₁ (11)	Seri çıkış	TxD
P3 ₂ (12)	Haricî kesme 0	INT0
P3 ₃ (13)	Haricî kesme 1	INT1
P3 ₄ (14)	Zamanlayıcı/sayıcı 0 haricî giriş	T0
P3 ₅ (15)	Zamanlayıcı/sayıcı 1 haricî giriş	T1
P3 ₆ (16)	Haricî hafıza yazma sinyali	WR
P3 ₇ (17)	Haricî hafıza okuma sinyali	RD

**Özel Fonksiyon Kaydedicileri (SFR):**

8051 mikrodeneleyicisinin giriş/çıkış portları, zamanlayıcı/sayıcılar, seri port, yığın kaydedicisi gibi özel fonksiyonların kontrol edebilmek için veri hafızasının 80H...FFH adresleri arasında ayrılmış özel bir tampon bölgesi vardır. Bu bölgedeki veri hücrelerine "Özel Fonksiyon Kayıtları" (SFR; Special Function Registers) denilir.

SFR kaydedicileri, kısaltması ve adresi

SFR Kaydedicilerin Adı	Sembolü	Dâhilî Adresi
Port 0	P0	80H
Port 1	P1	90H
Port 2	P2	A0H
Port 3	P3	B0H
ACC veya A kaydedicisi	ACC	E0H
B kaydedicisi	B	F0H
Yığın kaydedicisi	SP	81H
Program durum kaydedicisi	PSW	D0H
Veri göstericisi	DPTR _L -DPTR _H	82H-83H
Güç kontrol kaydedicisi	PCON	87H
Kesme öncelik kaydedicisi	IP	B8H
Kesme izin kaydedicisi	IE	A8H
Seri port kontrol kaydedicisi	SCON	98H
Seri haberleşme kaydedicisi	SBUF	99H
Zamanlayıcı/sayıcı kontrol kaydedicisi	TCON	88H
Zamanlayıcı/sayıcı mod kontrol kay.	TMOD	89H
Zamanlayıcı/sayıcı 0 (düşük byte)	TL0	8AH
Zamanlayıcı/sayıcı 1 (düşük byte)	TL1	8BH
Zamanlayıcı/sayıcı 0 (yüksek byte)	TH0	8CH
Zamanlayıcı/sayıcı 1 (yüksek byte)	TH1	8DH

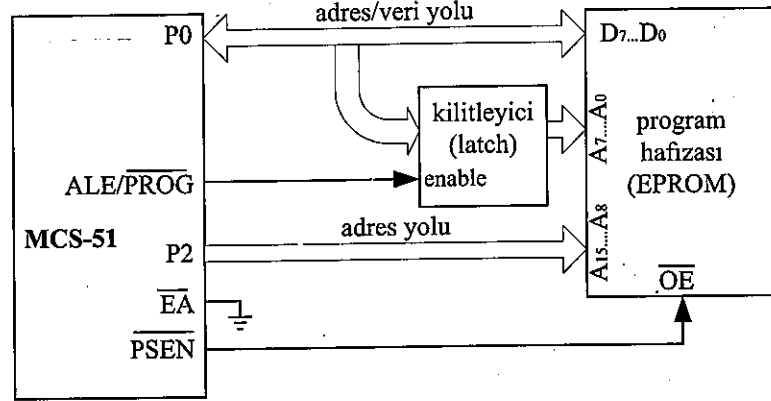
Seri giriş/çıkış portları: Seri giriş/çıkış port uçları, veri gönderme ve alma işlerini aynı anda yapabilmektedir. Bu şekildeki iletişime "full duplex" denilmektedir. Seri portla yapılacak iletişim kuralları SFR kaydedicilerinden SCON düzenlenerek belirlenir. Seri porttan gönderilecek veri veya alınacak veri SFR kaydedicilerinden SBUF kaydedicisi üzerinden gerçekleştirilir. Aynı anda veri gönderme ve alma tek başına SBUF kaydedicisi üzerinden yapılıyor görünmesine rağmen gerçekte bu iş donanım olarak iki ayrı kaydedici ile yapılmaktadır. SBUF üzerinden yapılan okuma bir kaydediciden yapılırken SBUF kaydedicisi üzerinden okuma diğer kaydedici ile yapılmaktadır. Fakat kullanıcı açısından gönderme ve alma işlemi aynı kaydediciyi üzerinden "SBUF" yapıldığından bu ayrımı çok önemli değildir. Programcı seri iletişimi başlatmadan önce mutlaka SCON ve PCON kaydedicilerini düzenlemek zorundadır.

Zamanlayıcı/sayıcı: Zamanlayıcı/sayıcı, sistem saatini kullanarak sayma ve zamanlama işlerinde kullanılır. Bu sayıcıların dışarıdan gelen haricî palsleri de kullanarak sayması sağlanabilir. Bunun için port3'ün T0 ve T1 girişleri kullanılabilir. Zamanlama/sayma işlemlerinin durumunu ilgili zamanlayıcı/sayıcı bir kesme sinyali üretirek mikrodeneleyiciye bildirir.

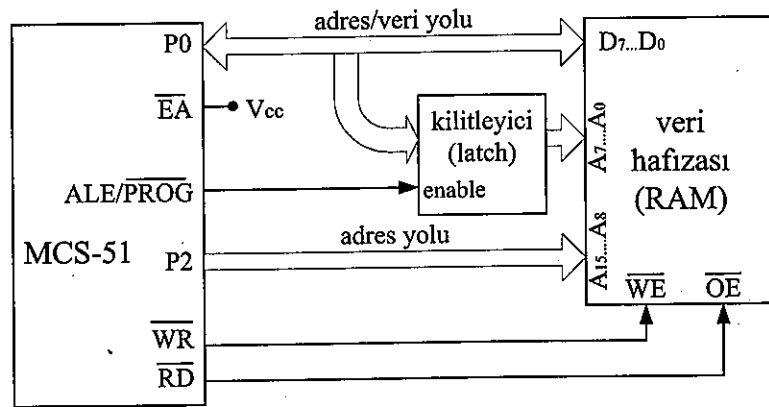
Kesme girişleri: MCS-51 mikrodeneleyicilerinde herhangi bir kesme sinyalini kullanmak için önce IE (kesme yetkilendirme, şekil 17.9) kaydedicisindeki EA biti "1" yapılarak gelen kesmelere karşılık verecek durumda kalması sağlanır. Ayrıca hangi kesmelere izin verilecekse IE kaydedicisindeki ilgili bit "1" yapılır. Daha sonra kesme oluşmasında bu kesmeye cevap verecek kesme alt programının hazırlanıp program hafızasına yüklenmesi gerekir. Kesme hizmet alt programının başlangıç adresi, kesme vektöründe mutlaka belirtilmelidir.

17.1.2 Hafıza Bağlantıları (RAM ve EPROM)

Mikrodenetleyicilerin bazı modellerinde her ne kadar dâhilî program ve veri hafızaları bulunsada bazı durumlarda bu hafızalar yeterli gelmeyebilir. Mikrodenetleyicilerin bazı modellerinde ise hiç program ve/veya veri hafızası olmayabilmektedir. Bunun için MCS-51 ailesindeki tüm mikrodenetleyiciler haricî veri ve program hafıza bağlantılarına imkân sağlarlar. Haricî veri veya program hafızalarının prensip şemaları aşağıda (şekil 17.5, şekil 17.6) verilmiştir. Haricî veri veya program hafızaları bağlantıları yapılırken kullanılan, $\overline{EA}$, ALE/PROG uçları kullanılmaktadır. $\overline{EA}$ girişi ucu mikrodenetleyiciye, haricî bir program hafızasının varlığını bildirir. Mikrodenetleyici EA girişi "0" iken 0000H...FFFFH arasında haricî bir program hafızasına erişecektir. giriş "1" ise haricî program hafızasının kullanılmadığı anlaşılar. ALE/PROG çıkışı ucu "1" olduğunda P0 portunun çıkışındaki adres/veri yolunda bir adres bilgisi olduğunu işaret eder. Bu sinyal bir kilitleyici elemanı ile birlikte kullanılarak program veya veri hafızasının düşük değerlikli adres bilgisi elde edilir.



Şekil 17.5 : Program hafızası için bağlantı şeması



Şekil 17.6 : Veri hafızası için bağlantı şeması

17.1.3 Aritmetik Birim

MCS-51 mikrodenetleyici ailesindeki ürünlerde bulunan ALU (Aritmetik Logic Unit) biriminde aritmetik işlemlerden toplama, çıkarma, çarpma, bölme gibi dört işlemin yanında AND, OR, XOR, döndürme/kaydırma ve bit tabanlı işlemler gibi lojik işlemleri yapılabilmektedir. Bu işlemleri yapabilmek için kullanılan komutlar da ACC kaydedicisi kullanılmaktadır.

17.2 MİKRO DENETLEYİCİ KOMUTLARI

MCS-51 mikrodenetleyici ailesinde kullanılan komutlar 5 ana grupta toplanmıştır;

- ❖ Aritmetik işlem komutları
- ❖ Mantıksal (lojik) işlem komutları
- ❖ Veri transfer komutları
- ❖ İkili işlem komutları
- ❖ Program dallanma komutları

Aşağıda bu gruplarda toplanmış komutların işleyişleri hakkında genel bilgi verilmiştir. Burada mikro işlemcilerin ve mikrodenetleyicilerin kullandıkları komutların tümünü ve işleyişlerini vermek mümkün değildir. Bu tip bilgilere, kullanılan mikro işlemci veya mikrodenetleyicilerin kataloglarını elde ederek ulaşmak gerekir. Örnek olarak kullanılan mikrodenetleyicinin bazı bilgileri bu kitabın "mikro işlemcilerin dış yapısı" bölümünde bulunmaktadır.

17.2.1 Aritmetik İşlem Komutları

MCS-51 mikrodenetleyicilerinin komut tablolarında aritmetik işlemler için kullanılan toplama, çıkarma arttırma, azaltma, çarpma, bölme ve ondalık ayar komutları bulunmaktadır.

Tablo 17.2 : MCS-51 mikrodenetleyici ailesinde aritmetik işlem komutları

İş kodu	Assembly komutu	Açıklama
28	ADD A,R0	A Kaydedicisi ile R0 kaydedicisi toplanır. Sonuç A'da saklanır.
38	ADDC A,R0	A Kaydedicisi ile R0 kaydedicisi elde biti ile birlikte toplanır. Sonuç A'da saklanır.
98	SUBB A,R0	A kaydedicisinden R0 kaydedicisi elde biti ile birlikte çıkarır. Sonuç A'da saklanır.
04	INC A	A kaydedicisini 1 arttırır.
14	DEC A	A kaydedicisini 1 azaltır.
A3	INC DPTR	Veri göstericisini 1 arttırır.
A4	MUL A,B	A ile B kaydedicisini çarpar. Sonuç A'da saklanır.
84	DIV A,B	A'yı B'ye böler. Sonuç A'da saklanır.
D4	DA A	A'yı onluk tabana göre düzenler.

17.2.2 Lojik İşlem Komutları

MCS-51 mikrodenetleyici komut kümesinde bulunan lojik işlem komutları ile AND, OR ve XOR işlemleri ile temizleme, silme, döndürme, değiştirme (takas) işlemleri yapılabilmektedir. Bunların yanında bu mikrodenetleyicilerin komut kümesinde bulunan birçok komut ile bit tabanlı işlemler yapabilmek mümkündür.

Tablo 17.3 : MCS-51 mikrodeneleyici ailesinde lojik işlem komutları

İş kodu	Assembly komutu	Açıklama
58	ANL A,R0	A kaydedicisi ile R0 kaydedicisine mantıksal AND işlemini uygular. Sonuç A'da saklanır.
48	ORL A,R0	A kaydedicisi ile R0 kaydedicisine mantıksal OR işlemini uygular. Sonuç A'da saklanır.
68	XRL A,R0	A kaydedicisi ile R0 kaydedicisine mantıksal XOR işlemini uygular. Sonuç A'da saklanır.
E4	CLR A	A kaydedicisini sıfırlar.
F4	CPL A	A kaydedicisinin komplementini (eşleniğini) alır.
23	RL A	A kaydedicisini sola döndürür.
33	RLC A	A kaydedicisini elde üzerinden sola döndürür.
03	RR A	A kaydedicisini sağa döndürür.
13	RRC A	A kaydedicisini elde üzerinden sağa döndürür.
C4	SWAP A	A kaydedicisindeki alt ve üst bitlerini (nibble) yer değiştirir.

17.2.3 Veri Transfer Komutları

17.2.3.1 Yükleme komutları

Yükleme komutları kaydedicilere veri yükleme ve kaydedicilerdeki verileri hafızalara yüklemede kullanılır. Yükleme komutları yapı bakımından aşağıdaki formattadır. Burada "hedef" yükleme yapılacak yeri, "kaynak" ise yüklenecek veriyi ya da verinin yerini göstermektedir.

Yükleme komutu <hedef>, <kaynak>

Tablo 17.4 : MCS-51 mikrodeneleyici ailesinde yükleme komutları

İş kodu	Assembly komutu	Açıklama
E8	MOV A,R0	R0 kaydedicisindeki değeri A'ya yükle.
F8	MOV R0,A	A kaydedicisindeki değeri R0'a yükle.
78 3F	MOV R0,#3FH	3FH değerini R0 kaydedicisine yükle.
83	MOVC A,[A+PC]	Program hafızasından veri yükle.
F0	MOVX [DPTR],A	DPTR ile işaretli yere A'yı yükle.
C0 E0	PUSH A	A'daki değeri yığına atar.
C8	XCH A,R0	R0 kaydedicisi ile A kaydedicisindeki değerler yer değiştirir.

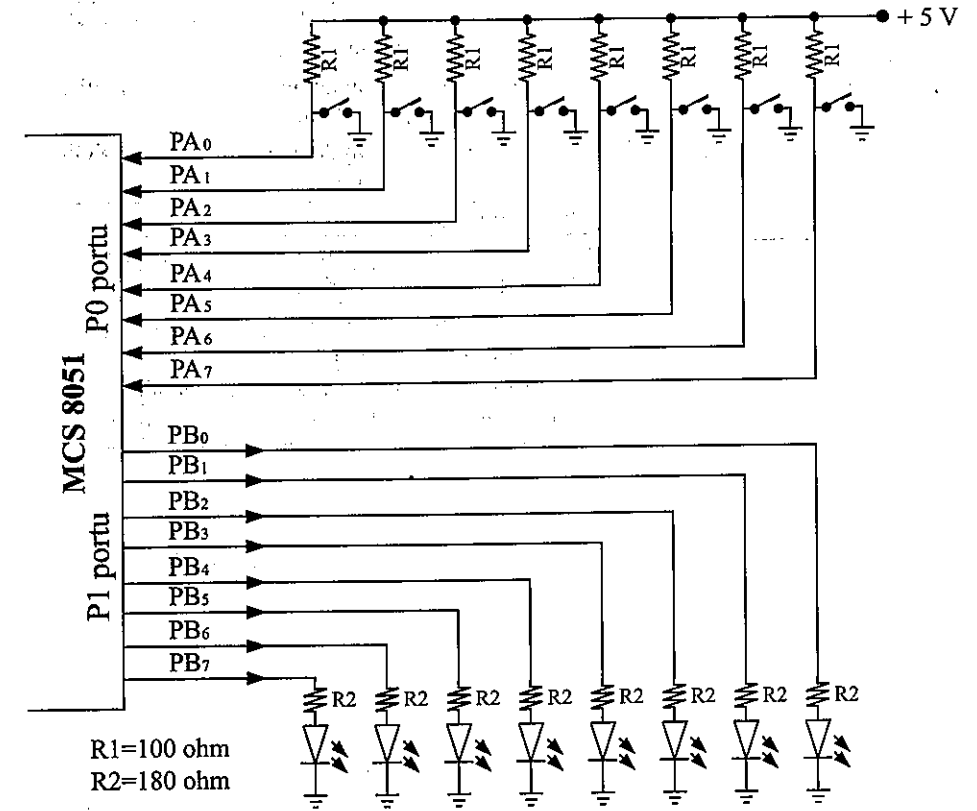
17.2.3.2 Giriş/çıkış komutları

MCS-51 ailesindeki mikrodeneleyicilerde giriş/çıkış işlemleri için bulunan IN/OUT gibi komutları yoktur. Giriş/çıkış işlemleri mikrodeneleyicinin giriş/çıkış portlarına donanım olarak bağlantılı olan SFR kaydedicileri (P0, P1, P2, P3) üzerinden gerçekleştirilir. Örneğin P0 portuna veri göndermek istediğimizde, göndereceğimiz veriyi SFR kaydedicilerinden P0 kaydedicisine yazmamız yeterli olacaktır.

17.2.3.3 Basit bir giriş/çıkış uygulaması

Bu örnek uygulama ile P0 portundan okunan giriş değerini, P1 portundaki ledlere göndererek basit bir giriş/çıkış uygulamasının donanım ve yazılımı incelenmiş olacaktır. Giriş olarak kullanılacak porta önce FFH değeri yazılarak o portun giriş portu olarak ayarlanması sağlanmış olur

Şekil 17.7'de normalde tüm girişler lojik "1" dedir ve tüm ledler yanacak şekilde program çalışmaktadır. P0 portundaki anahtarlara basıldıkça, bu girişe karşılık gelen P1 çıkış portundaki led de sönecektir.



Şekil 17.7 : MCS-51 mikrodeneleyici için basit giriş/çıkış uygulaması

Tablo 17.5 : Basit giriş/çıkış uygulamasının için yazılımı

Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
2200	75 80 FF		MOV P0,#FF	P0 Portunu giriş olarak ayarlar.
2203	E5 80	BAS	MOV A,P0	Anahtar konumlarını P0 portundan A'ya yükle
2205	F5 50		MOV P1,A	Anahtar konumlarını A'dan P1 portuna gönder.
2207	80 FA		SJMP BAS	Bas etiketine dallan. Sonsuz döngü.

17.2.4 İkili işlem Komutları

İkili işlem komutları bit seviyesinde işlem yapabilmemize imkân tanır. MCS-51 mikrodeneleyicilerinin bazı SFR kaydedicileri ve 20H...2FH arasındaki veri hafızasında bulunan hücreler üzerinde bit seviyesinde işlem yapabilmek mümkündür.

Tablo 17.6 : MCS-51 mikrodnetleyici ailesinde ikili işlem komutları

Hafıza adresi	İş kodu	Assembly komutu	Açıklama
2200	C3	CLR C	Elde'yi sıfırla
2201	D3	SETB C	Elde'yi "1" yap.
2202	D2 91	SETB P1.1	SFR kaydedicilerinden P1'in 1. bitini "1" yap.
2204	B3	CPL C	Elde'nin eşleniğini al.
2205	82 E7	ANL C,A.7	A kaydedicisinin 7. biti ile elde'yi AND'le
2207	A2 B3	MOV C,P3.3	P3 kaydedicisindeki 3.biti elde'ye yükler.

1.1.1 Program Dallanma Komutları

Programın, belirli durumlara göre işleyişini yönlendirebildiği komutlardır. Örneğin bir kontrol sinyaline göre bir ısıtıcının kapatılması veya açılması gibi. Dallanma komutları program sayıcısının içeriğini değiştirerek dallanmayı gerçekleştirir. Normalde her komuttan sonra program sayıcı bir sonra işletilecek komutun bulunduğu adresi gösterecek şekilde artar. Dallanma komutları şartlı ve şartsız dallanma komutları diye iki kısma ayrılabilir. Alt programa dallanma komutları, diğer dallanma komutlarından biraz farklıdır. Alt program dallanma komutları (CALL) kullanıldığında ana programda kullanılan kaydediciler ve geri dönüş değerleri yığına atılır ve bir geri dön komutu (RET) ile ana programda kaldığı yerden devam ederler. Bu tip bir işleyiş normal dallanma komutlarında yoktur.

Tablo 17.7 : MCS-51 mikrodnetleyici ailesinde program dallanma komutları

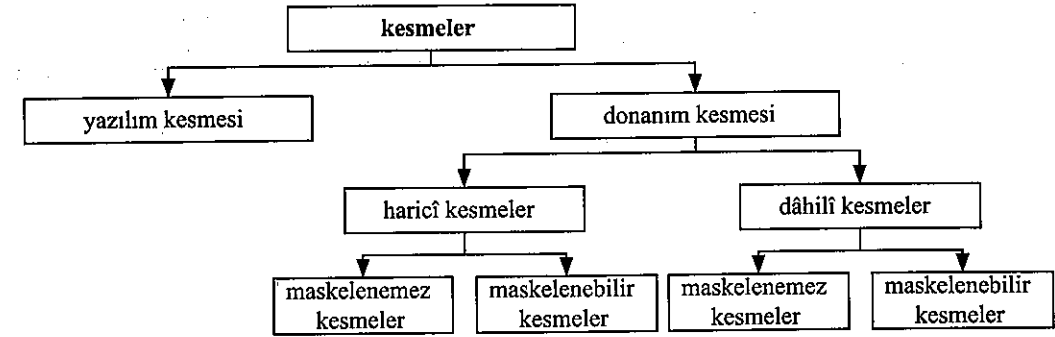
Hafıza adresi	İş kodu	Etiket	Assembly komutu	Açıklama
2200	12 F0 00		LCALL \$F000	F000H adresindeki alt programa dallan.
2203	22		RET	Alt programdan geri dön.
2204	32		RETI	Kesme alt programından geri dön.
2205	60 07		JZ LABEL	A kaydedicisi sıfır ise LABEL adresine dallanır.
2207	70 05		JNZ LABEL	A kaydedicisi sıfır değil ise LABEL adresine dallanır.
2209	D8 03		DJNZ R0,LABEL	Kaydediciyi azalt, sıfır değilse LABEL adresine dallanır.
220A	B8 20 00		CJNE R0,\$20,LABEL	Kaydedicideki veri 20H değerini karşılaştır. Eşit değil ise LABEL hafıza adresinden devam eder.

17.3 KESME KONTROLLÜ İŞLEMLER

Kesme işlemi bir sistemdeki mikrodnetleyicinin veya mikroişlemcinin çok öncelik isteyen bir duruma cevap vermesi diye tanımlanabilir. Mikrodnetleyici veya mikroişlemci rutin çalışması esnasında, dışarıdan veya içeriden acil cevap bekleyen bir durum olduğunda bunu hemen cevaplaması için kesme kontrollü çalışma şekli ile tasarlanmalıdır. Örneğin bir motorun hızı belirli bir değerin altına düştüğünde bu durumun hemen değerlendirilmesi için bir kesme sinyali ile sistem uyarılır. Mikrobilgisayar öncelikle kesme sinyalini gönderen birimi tespit eder ve ilgili kesme hizmet programını hemen devreye sokar.

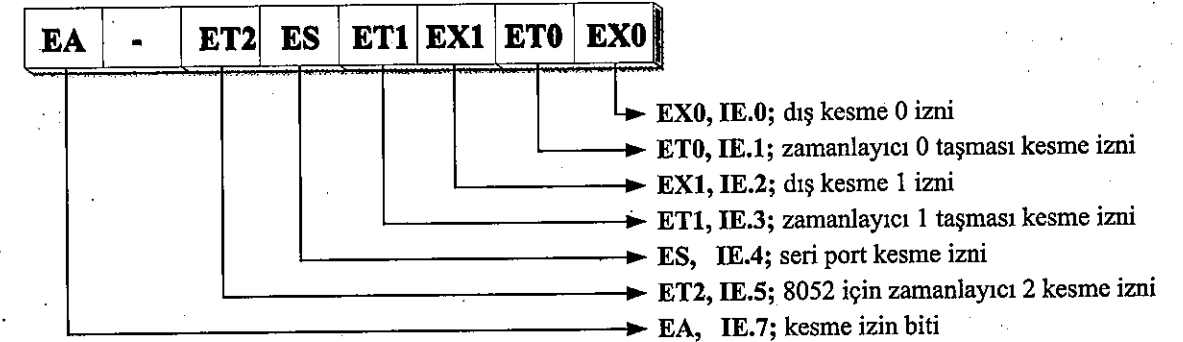
Kesme sinyalleri dışarıdan gelebileceği gibi sistemin kendisinden de oluşturulabilir. Hangi kesme girişi kullanılırsa kullanılsın mutlaka o kesmeye ait bir kesme hizmet programı bulunmak zorundadır. Kesme hizmet programları da tıpkı alt programlar gibi hafızanın belirli bir yerinde bulunan ve gerektiğinde kullanılan program parçalarıdır. Kesme hizmet programlarının alt programlardan farkı; programcı tarafından

program hazırlanırken istediği bir yerde "CALL" komutu ile çağırdığı bir program modülü olmasıdır. Bu program parçalarının kullanım yeri ve zamanı tamamen program kodlarında belirtilmiştir. Alt programların ne zaman ve nerede kullanıldığı programın kaynak kodunda görülebilir. Fakat kesme hizmet programları ancak bir haricî veya dâhilî kesme sinyali olduğunda kullanılan program parçalarıdır. Yani kesme hizmet programlarının ne zaman kullanılacağını haricî veya dâhilî çevresel birimler belirler. Dâhilî kesmeler bir zamanlayıcı/sayıcı taşması olabilir. Sayıcı belirli bir değere ulaştığında bir kesme sinyali ile mikroişlemciyi uyararak ilgili kesme hizmet programına dallanmasını sağlar. Kesmelerin hiyerarşik yapısı şekil 17.8'de gösterilmiştir.



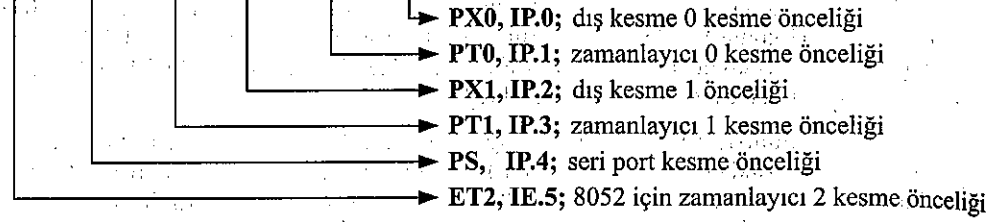
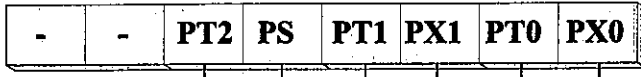
Şekil 17.8 : Kesme sinyallerinin hiyerarşik düzeni

Kesme izin kaydedicisi (şekil 17.9) mikrodnetleyicinin gelen kesmelere cevap verip vermeyeceğinin belirlendiği yerdir. Buradaki bitlerin durumuna göre gelen kesmelere cevap verilir veya verilmeyebilir. Bir dış kesmenin cevaplanabilmesi için ilgili kesmeye ait kontrol bitinin (IE0 veya IE1) yetkilendirilmesi, dış kesme girişine (INT0 veya INT1) bir kesme sinyalinin gelmesi ve o kesmeye ait kesme vektör tablosundaki adreste bir kesme hizmet programının bulunması gerekmektedir. Kesmenin düşen kenarda mı yoksa yükselen kenarda mı etkin olacağını TCON kaydedicisindeki TCON.0 ve TCON.2 bitlerinin durumu belirler.



Şekil 17.9 : IE; Kesme izin kaydedicisi (Interrupt Enable Register)

Kesme öncelikleri ile oluşan kesmelere verilecek hizmet sırasının belirlenmesidir. Oluşan kesmelerin öncelik sırası yazılım ile IP kaydedicisindeki bitlere göre belirlenebilmektedir. Böylece bir kesme sinyaline hizmet verilirken aynı öncelik seviyesine veya daha düşük öncelik seviyesine sahip kesmelere cevap verilmez. Eğer öncelik seviyesi daha düşük bir kesme oluşmuş ise o sırada işlem gören kesme hizmet programı bir süre askıya alınarak öncelik seviyesi yüksek kesmeye cevap vermek için ilgili kesme hizmet programına dallanılır. Bu kesme hizmet programı tamamlandığında önceki kesme hizmet programına devam edilir.



Şekil 17.10 : IP; Kesme öncelik kaydedicisi (Interrupt Priority Register)

Kesme öncelik biti "1" yapılan kesme, daha öncelik kazanacaktır. Eğer kesmeler aynı önceliğe sahip ise öncelik sırası yüksekten alçağa doğru aşağıdaki sıralama geçerlidir.

1. IE0
2. TF0
3. IE1
4. TF1
5. RI veya TI
6. TF2 veya EXF2

17.4 MİKRODENETLEYİCİLERİN SİMÜLÂTÖR PROGRAMI

Mikrodenetleyicilerin simülâtör programları, bize hazırladığımız programların sonuçlarını bilgisayar ortamında görsel olarak izlememize imkân sağlarlar. Simülâtör programları, her mikrodenetleyici için özel olarak hazırlanmış programlardır. Bu programlarda, mikrodenetleyicinin tüm kaydedicileri ve giriş/çıkış portları izlenebilir. Hazırlanan programın akışı bu kaydedicilerin ve çıkışların durumuna bakılarak izlenebilir.

Simülasyon ortamında donanım ve yazılımlarda görülen hatalar hemen düzeltilebildiğinden çoğu zaman gerçek ortama aktarılacak harcanan zaman ve ara kaybının önlenmesine önemli katkılar sağlamaktadır. Simülasyon ortamında geliştirilen ve test edilen donanım ve yazılım tasarımcıyı gerçek ortamında daha az hata ile karşı karşıya bırakır. Tasarımcıların özellikle çok karmaşık sistemler üzerinde çalışırken simülasyon programları kullanmasında fayda vardır.

17.5 EPROM YAZMA YÖNTEMİ İLE PROGRAMAKTARIMI

EPROM hafızalar en çok kullanılan kalıcı hafıza birimleridir. Mikrobilgisayarlar veya mikrodenetleyicilerin monitör programları¹ bu tip hafızalara yüklenir. Sistemin enerjisi kesilse dahi EPROM'a yüklü olan program silinmez.

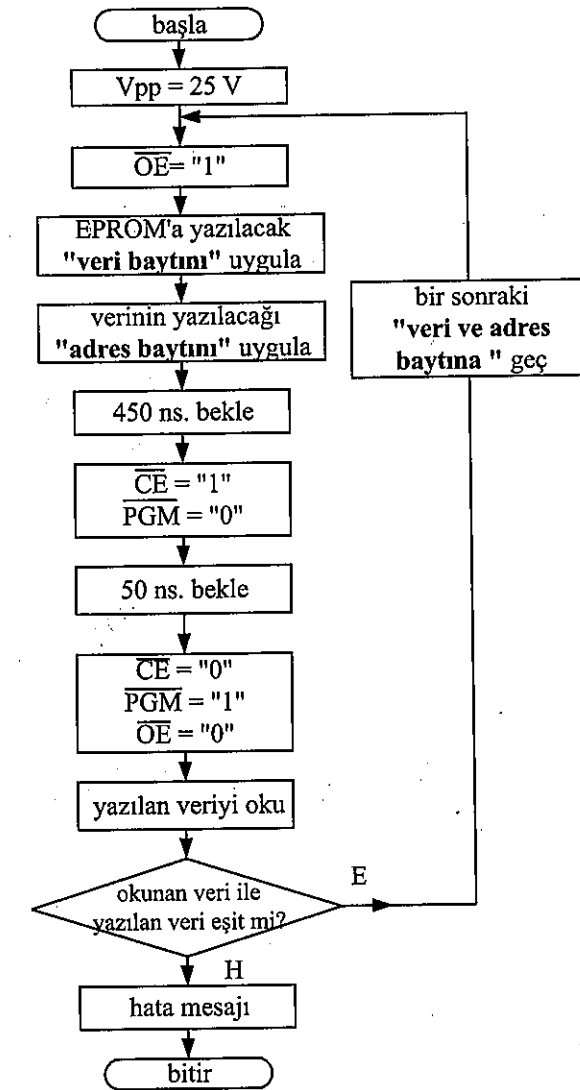
EPROM'lar, dalga boyu $2.537.10^{-7}$ m (2537 angström) olan mor ötesi ışık altında 2,5 cm mesafede bırakıldığında yaklaşık 15 ile 20 dakika arasında silinirler. EPROM'un silinmesi demek bütün hücrelerinin "1" yapılması demektir. EPROM programlanması ise yazılacak verinin bit deseninde bulunan "0" lara karşılık EPROM'daki "1" hücrelerini "0" yapmaktır.

EPROM'ların programlanması iki yöntemle yapılmaktadır. Birincisi, EPROM'a yazılacak verilerin ek bir devre yardımı ile adres/veri değerlerinin tek tek belirlenerek yazıldığı el ile programlama yöntemidir. Bu yöntem küçük program veya verileri yüklemeye kullanılabilir fakat çok uzun program veya

verileri yazmak için hiç kullanışlı değildir. Çok uzun program veya verileri yazmada daha çok bilgisayar destekli EPROM programlama cihazları kullanılmaktadır. Bu cihazlar programcıya EPROM'u okuma ve yazma noktasında çok büyük kolaylık ve hız kazandırır. Aynı zamanda EPROM'un programlanması esnasında oluşabilecek hatalar anında programcıya bildirir. Piyasada en çok kullanılan genel amaçlı EPROM'lar Intel'in 2716, 2732, 2764 kodlu EPROM'larıdır. Bu EPROM'ların hakkında daha teknik bilgiler için üretici firmanın katalog veya Internet sayfalarına bakılabilir. Şekil 17.11'de 27xx serisindeki bir EPROM'un programlanması için kullanılan programlama algoritması görülmektedir.

Açıklama: EPROM'un Vpp bacağı +25 V'a çekilir (Bu değer EPROM'un tipine göre değişiklik gösterebilir. Bazılarında bu değer +21 V' tur.). EPROM'un seçme girişleri aktif yapılarak EPROM etkin hâle getirilir (CS="1"). Yazılacak "veri" ve "adres" byte'ı EPROM'a uygulanarak kısa bir süre beklenir. Programlamayı gerçekleştirmek için PGM (programming) bacağı 50 ns süresince "1"de tutulur. Sonra PGM "0" yapılarak program darbesi kesilir. EPROM'un programlanması için bu işlem tekrar edilir.

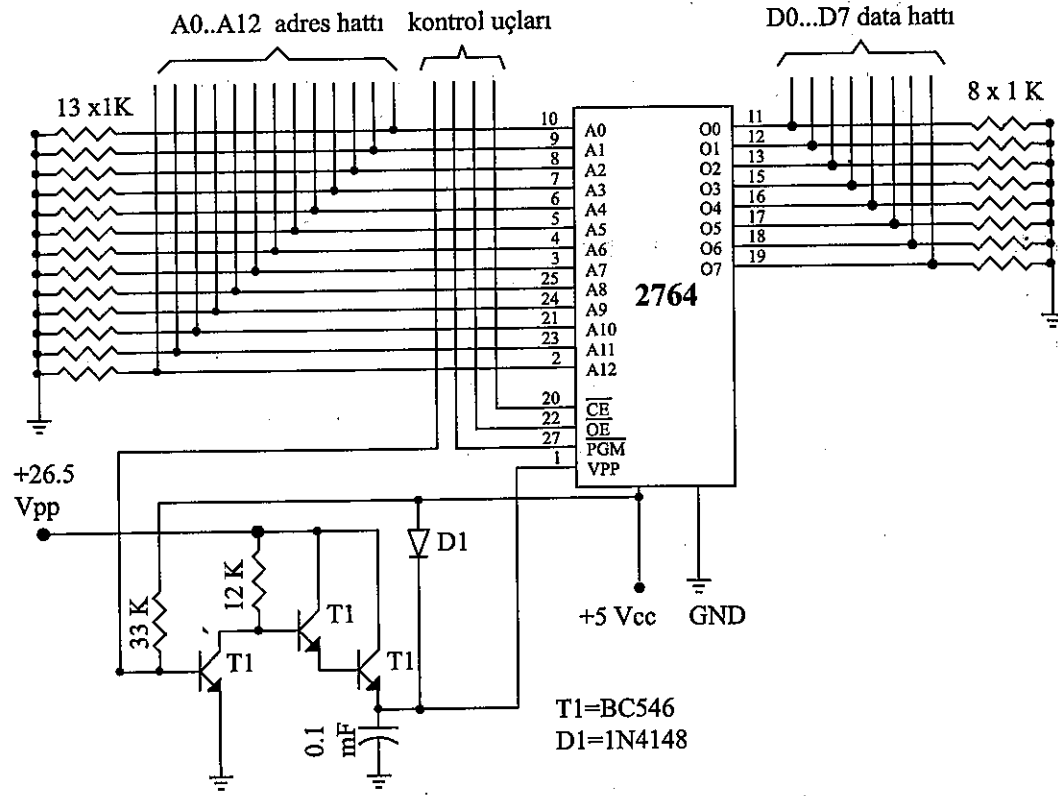
Programlamada, adresler ardışık sıra ile yapılabileceği gibi değişik adresler karmaşık şekilde programlanabilir. Eğer herhangi bir hücre programlanmadan önce test için okunup tüm bitleri "1" olup olmadığı kontrol edilirse programlama daha güvenli olacaktır. Yanlış yazılmış bir hücre için EPROM'un tamamen silinmesi gerekmektedir.



Şekil 17.11 : EPROM programlama akış diyagramı

¹ Monitör programı: Bir mikrobilgisayarın çalışmasını ve izlenmesini sağlayan temel yazılım, işletim sistemi.

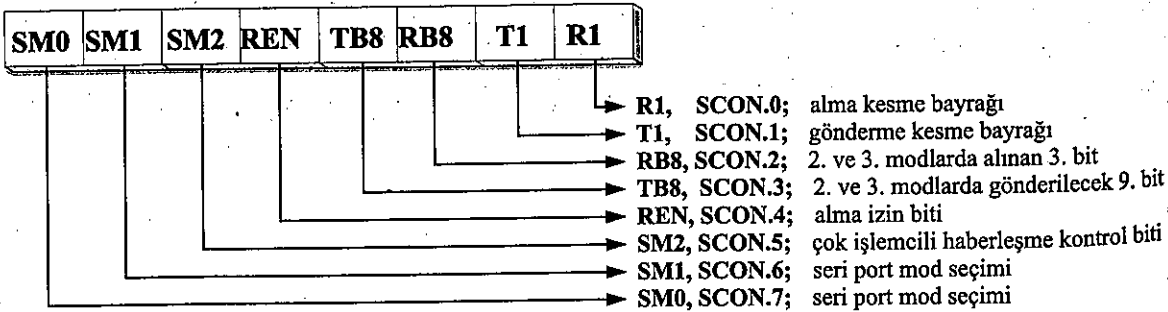
Şekil 17.12'de EPROM programlama devresi görülmektedir. Devre 26.5 voltluk ve 5 voltluk kaynakla beslenmektedir. Devrenin transistorlu kısmı programlama gerilimi olan 25 V'luk anahtarlama yapmaktadır. 26.5 voltluk gerilim transistorlar üzerindeki gerilim düşümleri çıktıktan sonra Vpp gerilimi için gereken 25 V elde edilmiş olur. Kontrol uçlarına programlama sinyali verildiği zaman 25 voltluk programlama gerilimi EPROM'a uygulanmış olur. Programlama sinyali uygulanmadığı zamanlarda, diyot iletme geçerek Vpp bacağına EPROM'un normal kullanımı için gerekli "1" sinyali sağlanmış olur. Kondansatör EPROM programlanırken oluşabilecek elektriksel parazitleri filtre eder.



Şekil 17.12 : 2764 EPROM Programlama Devresi

17.6 SERİ PORT'TAN PROGRAMIN AKTARILMASI

MSC-51 mikrodeneleyicilerinde seri port, tıpkı paralel portun kullanımında olduğu gibi SFR kaydedicilerini kullanır. Seri portun SFR kaydedicileri, "SCON" ve "SBUF" kaydedicileridir. SCON seri iletişim ile kontrol edilir. Seri porttan gönderilen veya seri porta gelen bilgiler "SBUF" kaydedicisi üzerinden yapılır.



Şekil 17.13 "SCON" seri iletişim kontrol kaydedicisi

R1; seri veri alma işleminin tamamlandığını belirtmek için donanım tarafından "1" yapılır.

T1; seri veri gönderme işleminin tamamlandığını belirtmek için donanım tarafından "1" yapılır.

RB8; 9. veri bitinin alınacağı mod2 ve 3 çalışmada alınan son bit RB8'e yüklenir.

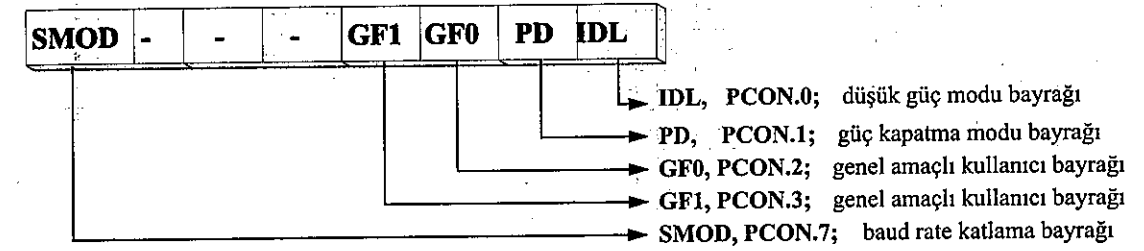
TB8; mod 2 ve mod 3 çalışmada gönderilecek 9. veri bitini belirler. Yazılım ile bu bit "1" veya "0" yapılabilir.

REN; seri veri alıp almama durumunu tayin eder. Yazılım ile "1" yapılarak gelen seri veri alınır.

SM2; mod 2 ve mod 3 çalışmada çok işlemcili haberleşmede kullanılır.

SM1 SM0; seri portun çalışma modunu belirler.

Seri iletişimde kullanılan diğer bir kaydedici PCON (Power Control Register) kaydedicisidir. Eğer baud rate üretimi için "zamanlayıcı 1" kullanılıyorsa ve SMOD "1" ise seri port'un baud rate'i mode 1, 2, 3'te 2 kat artırılır.



Şekil 17.14 "PCON" güç kontrol kaydedicisi

MCS-51 seri port iletişimi 4 ayrı modda çalışmaktadır. Bunlar:

Tablo 17.8 : Seri port modları

SM1	SM0	Çalışma modu (SMOD)	İletişim hızı	Açıklama
0	0	0	$F_{osc}/12$	Kayıtlı kaydedici
0	1	1	"Zamanlayıcı/sayıcı 1" ile belirlenir.	8 bit UART
1	0	2	$F_{osc}/64$ veya $F_{osc}/32$	9 bit UART
1	1	3	"Zamanlayıcı/sayıcı 1" ile belirlenir.	9 bit UART

Tablo 17.9 : Seri port kurulumu

Çalışma modu (SMOD)	SM2	SCON
0	0 (tek işlemcili)	10H
	1 (çift işlemcili)	-
1	0	50H
	1	70H
2	0	90H
	1	B0H
3	0	D0H
	1	F0H

Mod 0: Bu modda iletişim hızı osilatör frekansının $F_{osc}/12$ 'sidir ve değiştirilemez. Seri portu bu modda kullanmak için zamanlayıcıları kurmaya gerek yoktur. Gönderilecek veriyi SBUF'a yazmak yeterlidir.

$$\text{BaudRate} = F_{osc} / 12$$

Mod 1: İletişimin hızı zamanlayıcı 1 ile belirlenir. Zamanlayıcı 1, mod 2 konumuna alınır ve zamanlayıcının frekansına göre iletişimin hızı aşağıdaki formül ile hesaplanır. Formüldeki "SMOD" PCON kaydedicisinde baud rate katlama değerini belirleyen bayraktır. "TH1" ise bir SFR olup zamanlayıcı 1'in yüksek değerlikli baytını gösterir.

$$BaudRate = \frac{2^{SMOD} \cdot F_{OSC}}{384 \cdot (256 - TH1)}$$

Genellikle kullanılacak olan baud rate ve osilatör frekansına uygun TH1'in değerini hesaplama yoluna gidilir. Bu durumda formül;

$$TH1 = 256 - \frac{2^{SMOD} \cdot F_{OSC}}{384 \cdot BaudRate} \text{ şekli ile kullanılır.}$$

Mod 2: İletişimin hızı osilatör frekansının $F_{osc}/32$ ' si veya $F_{osc}/64$ 'ü olarak sabittir. Bunu PCON kaydedicisindeki SMOD biti belirler.

PCON.SMOD="0" ise Baud Rate = $F_{osc}/32$

PCON.SMOD="1" ise Baud Rate = $F_{osc}/64$ 'tür.

Mod 3: İletişimin hızı değişkendir, zamanlayıcı/sayıcı frekansına bağlıdır. Seri veri iletimin şekli Mod 1 kullanımında olduğu gibidir.

DEĞERLENDİRME ÇALIŞMALARI

- 1) Mikrodenetleyicilerin mikroişlemcilerden farkı nedir?
- 2) Intel 8051 mikrodenetleyicinin özellikleri nelerdir?
- 3) 8051 mikrodenetleyicideki SFR kaydedicileri nelerdir?
- 4) 8051 mikrodenetleyicinin vektör adresleri nelerdir?
- 5) 8051 mikrodenetleyicide hafıza bölümleri kaç ayrılmıştır?
- 6) 8051 mikrodenetleyicide program ve veri hafızası bağlantılarını çiziniz.
- 7) 8051 mikrodenetleyici komut grupları nelerdir?
- 8) Aritmetik işlem grubu komutlarını bir örnek üzerinde açıklayınız.
- 9) Lojik işlem grubu komutlarını bir örnek üzerinde açıklayınız.
- 10) Veri transfer işlem grubu komutlarını bir örnek üzerinde açıklayınız.
- 11) 8051 mikrodenetleyicisinde giriş/çıkış işlemlerinde kullanılmak üzere kaç çeşit port vardır?
- 12) 8051 mikrodenetleyicisinde giriş/çıkış işlemlerini bir örnek üzerinde açıklayınız.
- 13) 8051 kesme izin kaydedicisinde bulunan bitlerin fonksiyonlarını yazınız.
- 14) Simülâtör programları ne işe yarar?
- 15) EPROM'a veri yazma ve silme işlemlerini anlatınız.
- 16) EPROM'a veri yazmak için hazırlanacak programın akış diyagramını inceleyiniz.
- 17) 8051 mikrodenetleyicisi ile seri iletişim gerçekleştirebilmek için hangi kaydediciler kullanılmaktadır?
- 18) Seri port mod'ları nelerdir?

ÜNİTE 18 ROBOTLAR

18.1 ROBOTUN TANIMI

İnsanlar her zaman işlerini kolaylaştırıcı araçlar yapmak ve geliştirmek ihtiyacı hissetmişlerdir. Bu ihtiyaçlar ilk insanlarda kesici, delici, kırıcı aletleri yaparak günümüzde ise otomobil, bulaşık makinesi, çamaşır makinesi gibi çok sayıda araç ve gereçler üreterek kendini göstermiştir.

İşlerin kolaylaştırılması ve hızlandırılması düşüncesi endüstrinin değişik kollarında da kendini her zaman göstermiştir. Bunun neticesinde; klâsik makineler, yarı otomatik makineler ve tam otomatik makinelerden sonra artık hata denetimi yapan, verimliliğin artırılması ve belli kalitede mal üretilmesini sağlayabilmek için endüstriyel çalışmalar giderek bilgisayara dayalı otomasyona yönlendirmiştir. Bu nedenle günümüzde bir çok üretim dalında sadece belli fonksiyonları çok hassas, hızlı ve devamlı aynı kalitede yerine getiren özel makineler ve sistemler kullanılmaya başlanmıştır. Şimdilerde fabrikalarda en az insan gücüyle üretim yapılmakta, üretimde merkezi kontrol sistemleri kullanılmakta ve robotlar, üretimde insan işgücünün yerini almaktadırlar.

Tüm bunlar bir arada düşünüldüğünde; malzemeleri, parçaları, takımları hareket ettirmek ve taşımak için tasarlanmış programlanabilir çok fonksiyonlu manivelâ veya çeşitli özel işleri uygulamak için programlanmış hareketleri yapan özel cihazlara "robot" denir.

Endüstride robotlara ihtiyaç duyulmasındaki gerekçelerden biri insan gücü ile yapılamayacak işleri yaptırma düşüncesidir. Örnek vermek gerekirse elektronik devrelere pirinç tanesi büyüklüğündeki yüzlerce parçanın yerleştirilmesi insanlarca yapıldığında yorucu, zahmetli, sıkıcı ve hata oranı yüksek bir iştir. Oysa bir robot kolu bu işi süratli ve hatasız bir biçimde gerçekleştiriyor, üstelik dinlenme molası bile vermeden ! Ayrıca rutin işlerin yanında insanların uzun süre çalışamayacakları ortamlarda da robotlara iş düşüyor. Deniz dibinde yüksek basınç altında, yüksek sıcaklıkta, Mars yüzeyi gibi elverişsiz ortamlarda, hatta insan vücudunun sığamayacağı genişlikteki havalandırma borularında robotlar kullanılıyor.

Robotların klâsik otomasyon makinelerinden farklı bazı yanları vardır. Robotları, değişik ortam ve işler için yeniden programlanarak çok amaçlı kullanılan portatif makineler olarak tanımlarsak klâsik otomasyon makineleri (ne kadar akıllı ve yetenekli yazılım ve donanıma sahip olurlarsa olsunlar) bir tek amaç için tasarlanmış makinelerdir. Bu duruma göre robotlar, mekanik işleyişinden çok robotun çalışmasını sağlayan programın yeniden düzenlenerek çok amaçlı kullanıma imkân sağlaması yönü ile diğer makinelerden farklıdır. Bunun yanında robotların daha akıllı ve tanımlayıcı olmaları için "yapay zeka" çalışmaları hızla devam etmektedir. Yapay zeka ile robotlar öğrenme, görüntü işleme, hareket, birden fazla robotların kullanıldığı sistemler, davranış kontrolü ve yer belirleme gibi konularda daha çok yetenek kazanacaklardır.



Resim 18.1 Ark kaynağı yapan robot

Robotlar üzerinde yapılan araştırmalar iki kolda devam etmektedir. Birincisi değişik ortam ve işlere göre yeniden programlanarak çok amaçlı kullanılan portatif makineler olarak “endüstriyel robotlar”, diğeri ise daha akıllı, insan gibi düşünüp ve karar verebilen, geçmiş deneyimlerini kullanabilen daha zeki makinelerdir. Endüstriyel robotlar endüstrinin her kolunda geniş bir uygulama alanı bulmuş ve kullanılmaktadır. İkincisi üzerinde daha çok araştırma çalışmaları devam etmektedir. Programlanarak çalışan bir makine ile algılayan, düşünen ve düşündüğünü eyleme dönüştüren bir makine arasında çok büyük fark vardır. Biz daha çok endüstrinin değişik üretim ve kontrol alanlarında kullanılan robotları tanımaya ve incelemeye çalışacağız.



Robotlarda “Algılama”, “Düşünme”, “Davranma” Döngüsü:

Carnegie Mellon Üniversitesi'nin ünlü Robot Cihazlar Enstitüsü'nden Chuck Thorpe “Zeki bir robotun kalitesinin göstergesi, “algılama-düşünme-davranma” döngüsüdür. Ve en zor kısmı da ‘algılama’dır” diyor.

Isaac Asimov da bilim kurgu romanlarında teknolojik açıdan tutarlı bir robot kavramı tanımlar ve robotların amacının insana hizmet olduğunu, bir robotun kendi amaçlarını insanların amaçlarına hiçbir zaman tercih edemeyeceğini koyduğu üç robot kuralı ile belirler;

1. Bir robot bir insana zarar veremez veya pasif kalmak suretiyle zarar görmesine izin veremez.
2. Bir robot kendisine insanlar tarafından verilen emirlere 1. kural ile çelişmediği sürece itaat etmek zorundadır.

18.2 ROBOTUN ENDÜSTRİDEKİ YERİ VE İNSANLA KARŞILAŞTIRILMASI

Endüstriyel alanlarda, çalışmaların daha çok insan işgücü ile yapılması durumunda üretim hızının ve kalitesinin düştüğü, konsantre ve güç gerektiren alanlar ile insan sağlığının ve hayatının tehlikeye düştüğü durumlarda robotların kullanılması tercih edilmektedir.

Endüstrinin her kolunda endüstriyel robotlar kullanılabilir. Bu tip robotlar, yapacağı işe uygun olarak programlanarak rutin işlerin hızlı, hatasız ve sistematik bir şekilde yapılmasını sağlamaktadır. Bu özelliği ile robotlar, aynı üretim ortamının her kademesine yeniden programlanarak bir çok yerde ve alanda kullanılabilir. Robotların kullanıldığı endüstriyel uygulama alanlarından bazıları aşağıda verilmiştir.

- ❖ Dökme, püskürtme ve presleme uygulamaları
- ❖ Paketleme, istifleme, seçme, sıralama, yükleme ve boşaltma uygulamaları
- ❖ Kaynak yapma, taşlama, parlatma, çapak temizleme ve kesme uygulamaları
- ❖ Daldırma, boyama, buharlama, kurutma uygulamaları
- ❖ Montaj uygulamalarında
- ❖ Radyoaktif ve X-ışını olan uygulamalarda
- ❖ Çok sıcak, çok soğuk veya basınçlı ortamlarda

Kısaca insan sağlığının veya hayatının tehlikede olduğu ortamlar ile hız ve hassasiyet gerektiren durumlarda robotların kullanılması tercih edilmektedir.

Tablo 18.1 : Robotların insanlarla karşılaştırılması

Çok hızlı ve sürekli çalışmaz.	Çok hızlı ve sürekli çalışır.
Dinlenme ihtiyaçları vardır.	Dinlenme ihtiyaçları yoktur.
Yorgunluk ve psikolojik nedenlerden dolayı hata yapma oranı yüksektir.	Hata yapma oranı düşüktür.
Duygusaldır. İşverimliliği duygularına göre değişir.	Tehlikeli ve sağlıksız ortamlarda insanların yerini alır.
Tekrarlı ve monoton işler kişiye olumsuz yönde etkiler ve çalışma verimini düşürür	Tekrarlı işleri doğruluk ve karalılıkla yerine getirir.
Olağa dışı durumlara karşılaştığında çok hızlı ve etkili karar verebilir.	İnsan gücüne nazaran daha kuvvetli ve seridir Performansları çalıştıkça düşmez
Robotları insanlar tasarlar. Robotların nasıl çalışacağını insanlar programlar ve arıza vermedikleri sürece programın dışına çıkmaları mümkün değildir (Robotların yapabilecekleri yüklenen programın yeterliliğine bağlıdır.)	Olağan dışı durumlarda çok hızlı ve etkili karar verebilmesi, bilgisayar programının yeterliliğine bağlıdır. Günümüzde, bilgisayarların kendi kendine öğrenmelerini (yapay zeka) sağlayacak program geliştirme çalışmaları hızla devam etmektedir.
	Kendi kendini programlayamaz. Arıza ve periyodik bakımları operatörlerce yapılır.

18.3 ROBOT SENSÖRLERİ

18.3.1 Duyular

İnsan duyuları robot araştırmalarında en sıklıkla kullanılan duyular olmuştur. Aneak bir robotta bu tip duyuların kullanılabilmesi için öncelikle insan üzerinde bu duyuların çalışma prensiplerinin son derece iyi bir biçimde analiz edilmesi gereklidir. İnsan duyuları bir veya birden fazla organ kullanılarak elde edilir.

Görme, gözler aracılığı ile gerçekleşir. Optik süreçler bu duyuda rol oynar. **Duyuma**, kulaklar aracılığı ile olur. Akustik süreçler sonucunda gerçekleşir. **Dokunma**, deri aracılığı ile merkezi değil dâginik bir algı olarak gerçekleşir. Temel mekanik ve ısı algılayıcılarının etkisi ile oluşturulur. **Koku**, burun aracılığı ile elde edilir. Gaz fazındaki kimyasal süreçler bu duyuyu oluşturur. **Tat**, dil aracılığı ile elde edilir. Sivi fazda gerçekleşen kimyasal süreçlerin sonucudur.

Duyular, tek bir organ kaynaklı olarak elde edilmedikleri gibi bir organ da sadece bir duyunun elde edilmesi için çalışmaz. Denge duyumu için kulaklar yardımcı olurken dil dokunma duyusu için önemli bir alan, deri de sıcaklık duyumu için en önemli bölgelerden biridir.

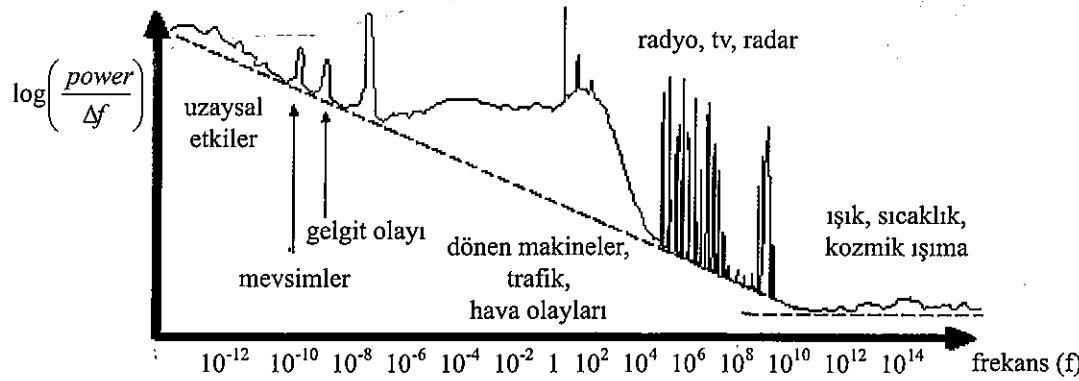
Robotlara insanlarda bulunan temel duyu organlarının işlevini yapacak duyargaları yerleştirmek önemli bir sorun olmasına rağmen bunların dışında, robotlara insanların sahip olmadığı diğer duyargalar eklenebilir. Bunlar, görülebilir tayfın dışındaki alanda görme yetisi, radar ve lazer kullanarak mesafe ölçümü, insanların duyarlı olduğu 20 Hz-20 KHz sinirinin dışında da duyabilme, tatma ve koku alma dışındaki kimyasal inceleme, radyasyona duyarlılık olabilir.

18.3.2 Ölçüm ve Ölçerler

Robotların çalışmasında “Algıla-Düşün-Davran” döngüsü temel bir çalışma sürecini oluşturmaktadır. Bu süreçte bir robotun çevresel şartları algılaması ve algıladığı niceliği değerlendirmesi robotların işleyişi açısından çok önemlidir.

İnsanlar arasında tercih edilen ölçüm yöntemleri, ölçümlerin görsel olarak yapılmasıdır. Bu ölçümler termometrelerdeki sıcaklığın, barometrelerde hava basıncının uzunluğa dönüştürülmesi, tartılarda ağırlığın, hız ölçerlerde hızın açıya dönüştürülmesi, indikatörlerde (gösterge) kimyasal süreçlerin renklere dönüştürülmesi olabilir. Sayılabilecek dönüşümler içerisinde ısı dirençler sıcaklığı dirence, elektro kimyasal algılayıcılar, kimyasal süreçleri elektriksel gerilime çevirir. Fotoakım devreleri ışık şiddetini akıma, piezoelektrik devreleri de termal ışımayı gerilime çevirir. Nem ölçerler, nemi kapasitansa dönüştürür. Mikrofonlar da ses basıncını çeşitli elektriksel yapılara dönüştürürler.

Genellikle ölçülen nesne ve ölçümü yapan aynı çevre içerisinde yer alır. Dolayısı ile ölçüm işlemi de ölçülen değer üzerinde etki yapmaktadır. Ayrıca çevre faktörleri ölçümler üzerinde karışıklık yaratmaktadır. Ölçüm sırasında girişim yaratan sinyallerin varlığı ve ölçü aletlerinin sınırlamaları ölçme değerlerini etkiler.



Şekil 18.1 : Doğadaki gürültü faktörleri

Sıcaklık, doğadaki tüm ölçümleri etkileyen bir gürültü tipidir. Buna "johnson gürültüsü" adı da verilir. Tüm cihazları etkilemektedir. Bunun en temel örneği dirençlerdir. Sıcaklığın etkisi ile rastgele hareket eden elektronlar gerilimde farklılıklar yaratırlar.

Sonuç olarak robotların geliştirilmesi sürecinde ölçme yöntemleri ve ölçücülerin geliştirilmesi önemli bir yer tutar. Robotlarda kullanılan ölçücüler genel olarak "algılayıcı veya sensör" olarak tanımlanmaktadır.

18.3.3 Sensörler/Ölçerler/Dedektörler

Robot araştırmalarında son derece zengin bir tayfa dağılan algılayıcılar kullanılmaktadır. Bunların bir bölümü insan temelli iken bir bölümü tümüyle insana yabancıdır. Algılayıcıları şu şekilde sınıflandırabiliriz:

Manyetik Algılayıcılar

- ❖ Pusula
- ❖ Dış uzaydaki yıldızlardan gelen manyetik alanı ölçen cihazlar
- ❖ Akı kapısı
- ❖ Manyeto indüktif algılayıcılar
- ❖ Manyetik direnç algılayıcıları (mr)
- ❖ Dev manyetik direnç algılayıcıları (gmr) ve dönme valfleri
- ❖ Bobin antenleri
- ❖ Hall etkisi algılayıcıları
- ❖ Manyetik kesinlik algılayıcıları
- ❖ Nükleer manyetik rezonans (nmr)
- ❖ Magnetoensefalograf (meg) & mcg
- ❖ Hırsızlığa karşı kullanılan algılayıcılar

Basınç Ölçerler

- ❖ Silikon zar yapılı algılayıcılar
- ❖ Basınçla değişir dirençli yapıda olanlar

Sıcaklık Ölçerler

- ❖ Sıcaklıkla değişir dirençli yapıda olanlar (termistörler)
- ❖ Silikon yarı iletkenler

İvme Ölçerler

- ❖ Mikromekanik yapıda olanlar
- ❖ Silikon basınçla değişir-dirençli yapıda olanlar
- ❖ Sarsıntı ölçerler
- ❖ Yerçekimi ölçerler
- ❖ Sismometreler (deprem ölçerler)

Nem Ölçerler

Işık Algılayıcıları

- ❖ Fotometreler
- ❖ Mor ötesi algılayıcılar
- ❖ Kızıl ötesi dedektörleri
- ❖ Bulut algılayıcıları
- ❖ Parlaklık algılayıcıları

Eğim Ölçerler

- ❖ Sıvı temelli

Gaz Fazı Algılayıcılar

- ❖ Koku algılayıcıları
- ❖ Gaz algılayıcıları

Akustik Algılayıcılar

- ❖ Ses algılayıcıları
- ❖ Konuşma tanıma cihazları
- ❖ Ultrasonik algılayıcılar

Yatay gözler (Gyroscope)

- ❖ Döner kütle sistemleri
- ❖ Fiber optik sistemler
- ❖ Magneto hidrodinamik sistemler
- ❖ Titreşen çatallar, tekerlekler ve basınçla çalışanlar

Mesafe Ölçerler

- ❖ Lazer
- ❖ Ultrason temelli algılayıcılar

Bir algılayıcıyı etkileyen en önemli çevre faktörleri depolama koşulları, kısa ve uzun dönemli kararlılık, çalışma sıcaklığı ve kendini ısıtmadan kaynaklanan hatalardır. Birçok algılayıcı sadece dirençler, kapasitör veya indüktörlerden oluşur ve aldıkları değerler çevredeki bir özellik ile ilgilidir. Termistörlerde direnç; sıcaklıkla, nem ölçerlerde kapasitans; nem ile değişir. Yakınlık ölçerlerde L yüzeye olan yakınlığa bağlıdır, manyetik direnç ölçerlerde direnç manyetik alan akısı değişimi ile değişirken ışığa bağlı iletkenlerde direnç gelen ışık yoğunluğunun bir fonksiyonudur. Birçok diğer algılayıcı ise temelde gerilim kaynağıdır. Elektrokimyasal algılayıcılar kimyasal süreçlere bağlı olarak gerilim kaynağı görevi üstlenirler, fotovoltaiik algılayıcıların gerilimleri ışık yoğunluğuna bağlıdır. Algılayıcının voltaj kaynağı olduğu durumlarda yüksek giriş empedansına sahip bir sistem tercih edilir.

Diğer önemli algılayıcılar, robotların kendileri ile ilgili ölçümler yapmalarını sağlayan algılayıcılardır. Bunlardan en yaygın enerji seviyesi göstergeleridir. Pillerin sıklıkla kullanıldığı robot uygulamalarında akım veya gerilim ölçülerek pilin enerji seviyesi hakkında bir fikir edinilebilir. Bu şekilde robotun derhal enerji bulması için gerekli işlemin başlatılması sağlanabilir.

Tıkanma algısı, bu durumda robot hareket edememekte, ancak motorlar çalışmaktadır. Motorlar, tam gerilimde ve kesilme akımı denen miktarda akım çekmektedirler. Bu durumun fark edilmesi, sadece robotun işlevini yerine getirebilmesi açısından değil, aynı zamanda motorların ve işlemcinin yanmasının engellenmesi için de önemlidir. Bazı motorlarda bunun fark edilebilmesi için özel sıcaklık algılayıcıları bulunur. Bu duruma müdahale edilmesinde çekingen davranmak gereklidir zira tekerleklerin dönmediği her durum tıkanmaya işaret değildir.

18.4 ROBOTUN PROGRAMLANMASI

Robotlar, değişik eksenlerde hareket eden parçaları olan makineler olmakla beraber daha önce belirttiğimiz gibi robotların diğer otomasyon makinelerinden farkı, değişik ortam ve koşullara göre yeniden programlanarak kullanılabilir olmasıdır.

Robotların kullanım alanlarının genişlemesi ile beraber yazılım teknolojisindeki gelişmeler değişik amaçlar için yeniden programlanması ihtiyacını da beraberinde getirmiştir. Bu nedenle değişik firmalar, ürettikleri robotlar için farklı yazılımlar geliştirmiş ve kullanmışlardır. Günümüzde robotlar için yapılan yazılımlar artık ayrı birer ürün değil; robotların birer parçası hâline geldiler.

Bazı robot üreticileri ürettikleri robotların programlanıp gerçek çalışma ortamında kullanımına gelmeden önce operatöre, robot için hazırlanan programın test edilmesine imkân sağlamak için bilgisayar ortamında simülasyon programlarını sunarlar. Simülasyonlar ile robot için hazırlanan programın, robotun hareketlerini nasıl etkilediği izlenmiş olacak ve çalışma sırasında meydana gelebilecek hatalar robotlar yerlerinden çıkarılmadan, simülasyon yardımıyla görülebilmekte ve düzeltilebilmektedir. Bu da işletmeye zaman ve maddi bakımdan büyük yararlar sağlayacaktır. Robotların programlanmasında üç yöntem kullanılmaktadır.

1. Gezinerek Programlama: Robot kolu belirli bir referans noktasında başlayarak gitmesi gereken yere bir kontrol konsolu yardımı ile (joystick, klavye veya benzeri kumanda araçları) götürülerek konumlandırılır. Bu pozisyonu robotun hafızasına yüklenir. Daha sonra bir sonraki pozisyona gidilir ve her pozisyonda kalma süresi de hafızaya girilerek robot programlanır. Robot çalışmaya başlatıldığında adım adım hafızasındaki pozisyonlara giderek çalışmasını sürdürür.

2. Hafızaya yükleyerek programlama: Bir operatör yardımı ile robota istenen çalışmayı öğretme ilkesine dayanır. Operatör robotu programlayabilmesi için ilk olarak robotla birlikte istenen işi tekrar eder. Tıpkı bir çocuğun elinden tutup yazı yazdırmaya benzer şekilde programlanır. Robot kayıt sistemi operatörün tüm hareketlerini hafızasına alarak çalışmaya başladığında aynen tekrar eder. Bu tip uygulamalara örnek olarak sprej (püskürtme) boyama verilebilir.

3. Uzaktan programlama: Robotun tüm hareketleri bir programlama dili kullanılarak uzak bir ofiste programlanabilir. Bunun için kullanılan birçok programlama dilleri mevcuttur. Aynı zamanda robot için hazırlanan programların test edilebilmesi için simülasyon programları da kullanılmaktadır. Robotların programlansın da kullanılan birçok programlama dili vardır. Aşağıda robotların programlanmasında kullanılan bazı programlama dilleri verilmiştir.

- ❖ **Robotscript:** Robotscript Evrensel Robot Denetimcisi (Universal Robot Controller) tarafından kontrol edilen bütün robot modellerini sanal olarak programlamak için kullanılır.
- ❖ **ARAC (Aritmetik Robot Uygulama Denetimi):** Yazılımı birçok esnek üretim sisteminin çekirdeğini oluşturur.
- ❖ **AML:** Hareketi tanımlayabilen bir program ya pozisyonu ya da zamanı referans alır. AML, PASCAL benzeri yüksek seviyeli bir programlama dilidir. Program, PASCAL'daki gibi "Begin" ve "End" komutları arasına yazılır. Ancak AML nesneye yönelik (object oriented) bir programlama dilidir.

- ❖ **RoboML:** Bazı robot programlama dilleri internet aracılığıyla insan robot etkileşimini sağlamayı amaçlamaktadır. Ancak burada eşzamanlı iletişim için gerekli olan yüksek kaliteli bağlantı, alıcıdaki (client) programın uyumluluğu gibi bazı sorunlar çıkmaktadır. Bunun için XML bazlı bir dil olan RoboML kullanılmaktadır.

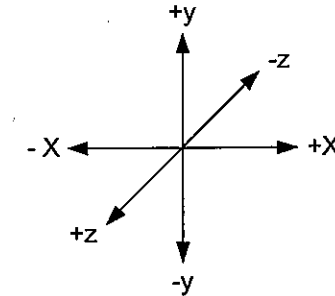
18.5 ROBOTLARIN HAREKET BİÇİMLERİ

18.5.1 Eksenler ve Hareket Biçimleri

Robotların hareket eksenleri ve hareket biçimleri kullanılacağı yere göre önemli bir kriter olarak karşımıza çıkmaktadır. Robotların hareket alanı "çalışma zarfını" oluşturur. Robotların hareket eksenleri;

- ❖ **Düşey hareket (doğrusal hareket) :** Kolun yukarı/aşağı hareketleri. Bu hareket, bütün kolun yatay eksen etrafında açılmal hareketi sağlayan bir eklemler veya kolun düşey bir yatak boyunca yukarı/aşağı hareketi ile yapılır.
- ❖ **Radyal hareket (doğrusal hareket) :** Kolun uzaması ve kısılması (içeri ve dışarı hareket).
- ❖ **Dönme hareketi (dairese hareket) :** Düşey eksen etrafında dönme (robot kolunun sağa veya sola dönmesi). Bilek hareketleri:
 1. Bilek dönme hareketi; bileğin kol/dirsek eksen etrafında dönmesi.
 2. Bilek eğilme hareketi; bileğin yukarı/aşağı açılmal hareketi.
 3. Bilek sağa/sola hareketi; bileğin sağa veya sola dönmesi.

Dairesel ve doğrusal olmak üzere iki temel hareket ekseninde çalışırlar. Bazı gelişmiş robotlar hem dairese hem de doğrusal hareketleri birlikte bulundurabilirler. Robotlar bu hareket eksenleri X,Y ve Z koordinat uzayında gerçekleştirebilirler.



Şekil 18.2 : Robotların üç boyutlu hareket eksenleri

Hareket Sistemleri: NC takım tezgahlarına benzer şekilde, endüstriyel robotların hareket sistemleri noktadan-noktaya (point-to-point PTP) veya çevresel (contouring) hareket olarak sınıflandırılır. Çevresel hareket sürekli hareket diye de adlandırılır.

Noktadan-noktaya harekette robotun hareketi bir nokta konumundan diğerine doğru kontrol edilir. Her nokta robotun kontrol hafızasında programlanır ve sonra çalışma anında bu noktalar işlem görür. Noktadan-noktaya robotlar bazı işlerde çok elverişli olabilirler, örneğin tezgah yükleme ve boşaltılması, al-yerleştir eylemleri ve nokta kaynağı uygulamaları.

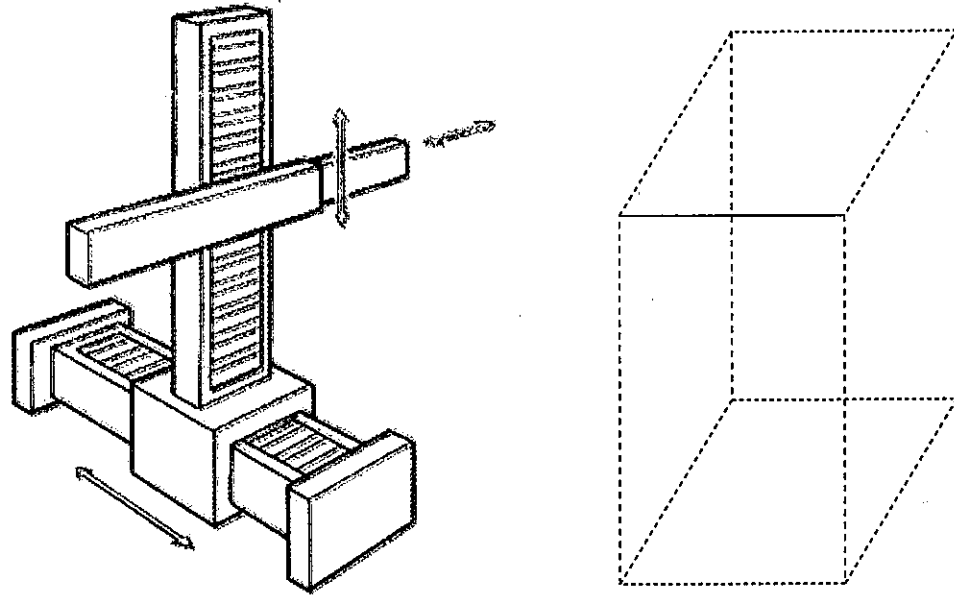
Çevresel robotlar, düzgün birleşik bir eğriyi meydana getiren birbirine çok yakın yerleştirilmiş olan noktaları takip etme kabiliyetine sahiptirler. Çevresel robotların hafıza ve kontrol kapasitesi, noktadan-noktaya olanlarından daha büyüktür. Çünkü robot tarafından takip edilen yolun sadece hareket dizilerinin uç noktaları değil tüm yol göz önünde bulundurulur. Bazı endüstriyel operasyonlarda sürekli kontrol gereklidir. Örneğin; püskürtme (sprej) boyama, sürekli kaynak operasyonları, hareket eden bir konveyör üzerindeki nesneleri tutma, vs.

Bir robot hareketinin kapasitesi, kontrol edilebilmesi mümkün olan eksenlerdeki hareketlerle belirlenir. Sayısal denetimdeki hareketlere çok benzerdir. Endüstriyel robotlar değişik tip ve boyutlarda yapılmaktadırlar. Çeşitli kol hareketlerini yapabilirler ve farklı hareket sistemlerine sahiptirler. Endüstriyel robotlar genel olarak şu konfigürasyonlarda üretilmektedirler

- ❖ Kartezyen koordinat konfigürasyonu
- ❖ Silindirik koordinat konfigürasyonu
- ❖ Küresel (polar coordinate) konfigürasyonu
- ❖ Mafsallı kol (jointed arm) konfigürasyonu

18.5.2 Kartezyen Koordinat

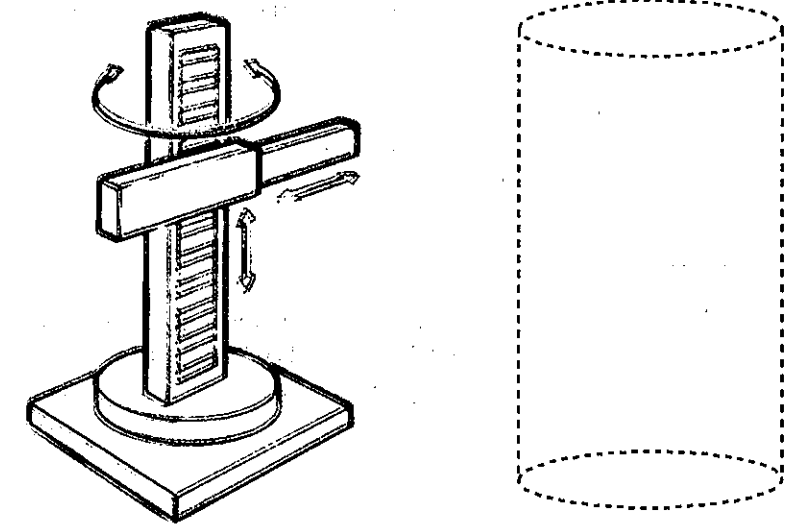
Bu konfigürasyon en kısıtlı hareket serbestine sahip robot tasarım şeklidir. Dörtgen konfigürasyon olarak da adlandırılır. Bu robot şekli birbirine dik üç eksende hareket eden kısımlara sahiptir. Hareketli kısımlar X, Y ve Z kartezyen koordinat sistemi eksenlerine paralel hareket ederler. Robot, üç boyutlu dikdörtgen prizması hacmi içindeki tüm noktalara kolunu hareket ettirebilir. Bu koordinat sistemi, dairesel hareket gerektiren uygulamalarda kullanılamaz.



Şekil 18.3 : Kartezyen koordinat çalışma özelliğine sahip robot taslağı ve çalışma zarfı

18.5.3 Silindirik Koordinat

Bu konfigürasyonda, robot gövdesi düşey eksen etrafında dönebilen düşey bir kolondur. Kol kısmının bir kaç hareketli kısmı vardır. Bu hareketli kısımlar kolu yukarı/aşağı ve içeri/dışarı hareket ettirir ve bilek kısmının dönmesini sağlarlar. Silindirik kol tasarımı, çalışma bölgesinde engellerin olmadığı uygulamalarda kullanılır. Silindirik konfigürasyonunun tipik işi nesneleri bir yerden başka bir yere taşımaktır (al ve yerleştir).

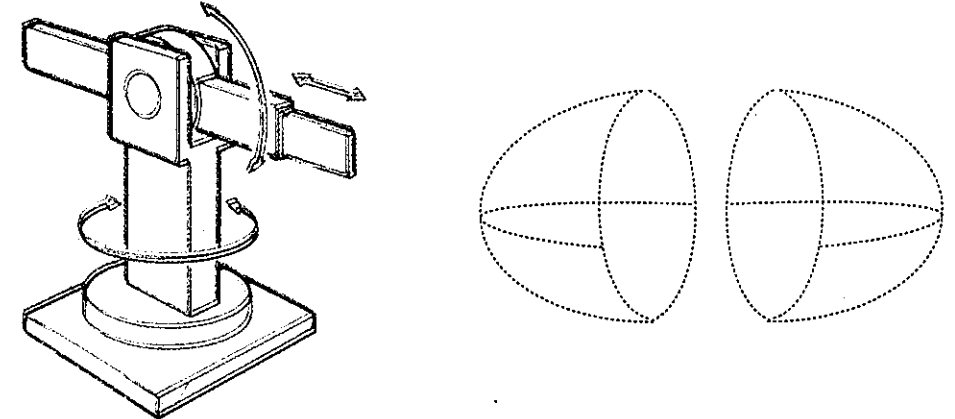


Şekil 18.4 : Silindirik koordinat çalışma özelliğine sahip robot taslağı ve çalışma zarfı

18.5.4 Diğer Koordinatlar

18.5.4.1 Küresel koordinat konfigürasyonu

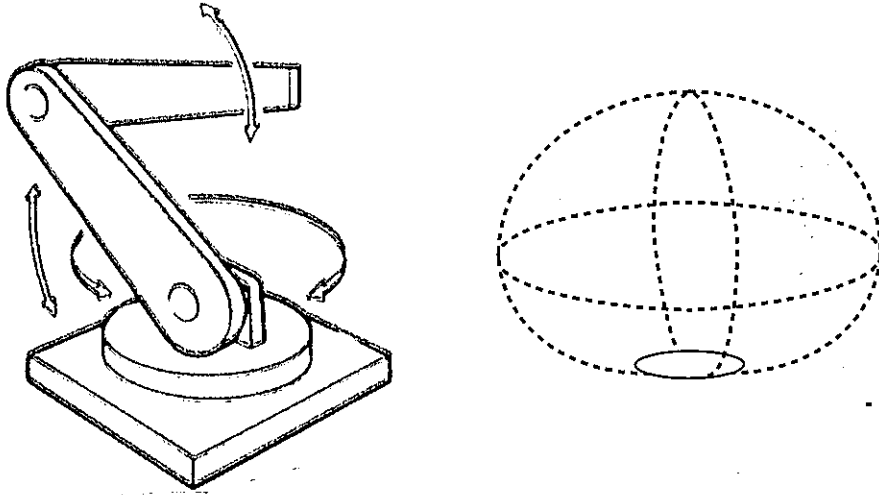
Kutupsal veya polar konfigürasyon diye adlandırılan küresel koordinat konfigürasyonu, silindirik koordinat konfigürasyonunun değişik bir uyarlamasıdır. Her türlü uygulamada robot konfigürasyonunun sağladığı çalışma hacmi (çalışma zarfı) önemli bir yer tutar. Kol, küresel bir hacim içinde hareket eder. Robotun bir döner tabanı vardır ve teleskobik kolu yukarı/aşağı hareket ettirmek için bir mafsal kullanılır.



Şekil 18.5 : Küresel koordinat çalışma özelliğine sahip robot taslağı ve çalışma zarfı

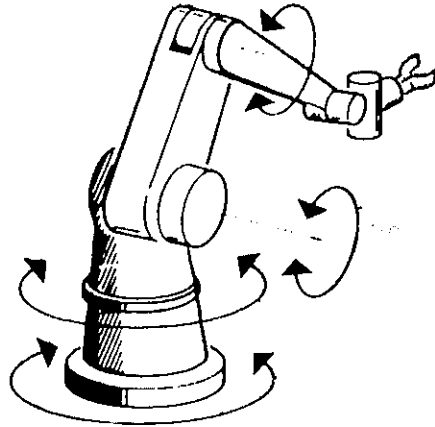
18.5.4.2 Mafsallı kol konfigürasyonu

Bağlantı parçalarından oluşan robot kol konfigürasyonu genellikle açısız veya mafsallı robot diye adlandırılır. Yaygın bir şekilde kullanılır. Çünkü otomotiv endüstrisindeki nokta kaynağı ve boyama işlemlerinde olduğu gibi maharetli hareketleri yapabilecek kabiliyete sahiptirler. Mafsallı kol konfigürasyonu, insan kolunun yapısına benzer şekilde tasarlanmıştır. Robotun, insan omuzu, dirsek ve bilek mafsallarını yaptığı hareketlere benzer hareketleri yapabilmesi için hareketli parçaları mafsallarla birleştirilmiştir. Robot kol bir tabana bağlıdır. Bu taban kendi ekseninde dönebilir.



Şekil 18.6 : Mafsallı kol koordinat çalışma özelliğine sahip robotun taslağı ve hareket zarfı

Yukarıdaki konfigürasyonlar sadece üç serbestlik derecesi sağlamalarına rağmen “uç eklentiler” ve ek açılar sağlayarak daha geniş bir çalışma alanı sağlanabilmektedir.



Şekil 18.7 : Mafsallı kol konfigürasyonuna sahip robot taslağı

18.6 ROBOT TIPLERİ

Endüstriyel robot bir dizi hareketi otomatik olarak yapmak için programlanabilen bir makinedir. Programlanabilir olduğu için kullanımı esneklik. Eski geleneksel otomasyonda bu esneklik yoktur. Endüstriyel robot değişik işler yapmak amacı ile iş parçalarını, malzemeleri, takımları veya özel cihazları taşımak için programlanabilir. Endüstriyel robotun CNC tezgahlarına benzer yönleri vardır. CNC tezgahlarında hareket için kullanılan teknoloji robotun mekanik kollarının hareketi için de kullanılır. Robotlar genellikle parçaları tutmak için programlanırlar. Robotların programlanması CNC tezgahların kullanılmasından farklıdır.

Robot kullanımının pratik ve ekonomik olup olmadığı araştırıldıktan sonra robot kullanmaya karar verilmelidir. Çok farklı kapasiteli, değişik tipte robotlar üretilmektedir. Hangi robotun, hangi iş için daha elverişli olduğuna, araştırıldıktan sonra karar verilmelidir.

Robotların sanayiye girmesiyle üretimin bir çok sektörü, büyük değişikliklere uğramıştır. Robotlar yaygın olarak otomotiv sanayiinde, esnek imalat sistemlerinde kullanılmaktadır. Ayrıca tezgah yükleme-boşaltma, malzeme taşımacılığı, nokta kaynağı, ark kaynağı, püskürtme boyama-kaplama, işleme (delik delme, taşlama vs.), elektrik montaj, makine-imalat montaj, kalite kontrol, kimya sanayinde, nükleer enerji santrallerinde, döküm sanayinde, uzay sanayi gibi işlerde fazlaca kullanılmaktadır.

Robotlar, üretim hatlarında ve esnek üretim sistemlerinde olduğu gibi malzeme taşıma görevleri ile bir grup cihazı entegre etmede de önemli rol oynamaktadır. Ayrıca robot, parçaları otomatik olarak takım tezgahına yükleyip almak için çok esnek özelliğe sahip bir sistemdir. Bir çok karmaşık ve otomatik sistemler, parçaların tezgahlara yüklenmesi için kullanılmaktadır. Ancak bunlar esnek olmayabilir. Endüstriyel robotlar çeşitli şekillerde sınıflandırılabilir. Yaptıkları işlere göre sınıflandırılabilirler, örneğin spre yayama, kaynak, montaj, vb. çeşitli işlere göre üç genel sınıflandırma yapılabilir.

18.6.1 Basit Robotlar

Bunlar “al-yerleştir” robotlarıdır ve genellikle malzeme taşımacılığında kullanılır. Bunların genellikle maliyeti ucuz, bakımı kolay ve hızlıdır. Orta ve uzun süreli üretim uygulamalarında üretkenliği büyük ölçüde artırır. Otomobil endüstrisi bu robotların kullanımına güzel bir örnektir. Son zamanlarda robotların sıkça kullanıldığı paketlemede de bunlar kullanılır. Geri besleme sistemi olmadığı için genellikle mafsal hareketleri için adım motorlar kullanılır. Bu robotlar, servosuz üç veya dört serbestlik derecelidir. Her eksen mekanik stoplar hareket sınırlarını ayarlamak için kullanılır. Hareketlerde 0,1mm'lik bir hassasiyet sağlanır. Basit robotlar, besleyici ve parça konumlayıcılar gibi yan destekleyici cihazlara çok bağlı olarak çalışırlar. Çünkü hareketleri sınırlıdır. Bu robotlar için nispeten daha basit bir kontrol sistemi gereklidir. Çünkü ana kontrol, çok az veya hiç geri beslemesiz açık/kapalı anahtarlarla yapılmaktadır.

18.6.2 Orta Teknoloji Robotları

Basit robotlardan daha fazla özelliklere sahip robotlardır. Geniş öğrenme kapasiteli ve daha fazla hafızaya sahiptirler. Bu robotlar 4 ile 6 arasında serbestlik derecelidirler ve genellikle bu eksenlerdeki hareketler servo kontrollüdür. Hareketler için doğru akım motorları kullanılır. Orta teknoloji robotları tek tezgahı yükleme, boşaltma ve benzeri işleri yaparlar. Elektrik kaynağı gibi sürekli yol operasyonlarını yapamazlar. Birçok işler bu robotlarla otomatikleştirilir.

18.6.3 Yüksek Teknoloji Robotları

Bu robotlar, endüstri otomasyonunda yüksek seviyeli kontrol üniteleri olan esnek ve programlanabilen manivelâlara sahiptirler. Bu kontrol üniteleri sensör ve kontrol cihazlarına bağlanabilir. Böylece, montaj gibi karmaşık işler robotlara öğretilir. Bu gelişmiş endüstriyel robotlar, bilgisayar kontrollü iş hücrelerine ve üretim sistemlerine bağlanabilir. Yüksek teknoloji robotlar birkaç programı saklayabilecek büyük hafızaya sahiptirler ve birlikte çalıştığı sistemin ihtiyaçlarına göre programlarını otomatikleştirebilirler. Bu robotlar, en karmaşık işleri yapabilecek güçlü yazı diline benzer programlama dillerine sahiptirler. Bunları programlamak için CAD teknikleri kullanılabilir.

18.7 ROBOT KONTROL VE ÇALIŞMA SİSTEMLERİ

Robotun fiziksel konfigürasyonları ve temel hareket kabiliyetlerine ek olarak diğer bazı teknik özellikler vardır. Bu özellikler verilen bir işi yapmak için endüstriyel robotun verimlilik ve etkinliğini tespit ederler. Bu teknik özelliklerden önemlileri şunlardır;

İş hacmi (çalışma zarfı): İş hacmi (çalışma zarfı), robotun çalıştığı hacme denir. Diğer bir deyişle robot bileği ucunun ulaşabildiği noktaları içeren hacimdir. Robot üreticileri iş hacmini, el veya takım olmadan bilek ucunun hareket edebileceği noktaların oluşturduğu hacim olarak tanımlamaktadırlar. Bir endüstriyel robotun çalışma hacmi, onun fiziksel konfigürasyonu, boyutu, kol ve mafsallık hareketlerinin sınırları tarafından tespit edilir. Bir kartezyen koordinat robotunun çalışma hacmi, dikdörtgen prizması şeklinde olur. Silindirik koordinat robotun çalışma hacmi silindirikdir. Polar koordinat veya küresel koordinat konfigürasyonu bir küresel hacim oluşturur. Mafsallı bir robotun çalışma hacmi ise düzensiz bir yapıdadır, kısmen küresel bir hacme sahiptir. Robot üreten firmalar genellikle özel iş hacimlerinin üstten ve yandan görünüşlerini ve ölçülerini kataloglarında belirtirler

Hareket hassasiyeti: Çoğu uygulamalarda robotun bilek ucunun hareket ettirilebilme hassasiyeti önemli yer tutar. Robotikte hareket hassasiyeti karmaşık bir konudur ve üç özelliği aşağıda verilmektedir

1. Uzaysal çözünürlük: Bilek ucunda robot tarafından kontrol edilebilen en küçük artışa (harekete) denir. Büyük ölçüde bu, robotun kontrol sisteminin çözünürlüğü tarafından tespit edilir. Ayrıca, robotun mafsallarındaki hassasiyet bozuklukları kolunu konumlama kabiliyetini etkiler. Kontrol çözünürlüğünü tespit eden faktörler, kolun hareket aralığı ve bu hareket için kontrol hafızasındaki 'bit' saklama kapasitesidir. Kol hareketi onun temel hareketlerine veya serbestlik derecesine bölünmelidir ve her serbestlik derecesinin çözünürlüğü ayrı ayrı değerlendirilmelidir.

2. Hassasiyet: Bilek ucunun (veya bileğe takılı birtakımın) verilen bir hedef noktaya çalışma hacmi içinde gidebilme kapasitesidir. Hassasiyet, uzaysal çözünürlük ile yakından ilgilidir çünkü uzaydaki bir noktaya ulaşmak için robotun kabiliyet, mafsallık hareketlerini küçük artışlara bölme özelliğine bağlıdır. Bir robotun hassasiyeti birbirine komşu iki çözünürlük noktası arasındaki mesafenin iki katıdır. Robotun hassasiyeti, mekanik hassasiyet bozukluklarından etkilenir. Örneğin, elemanların esnemesi, dişliler arasındaki bozukluk.

3. Tekrarlanabilirlik: Robotun bilek ucunun önceden öğretilmiş olan uzaydaki bir noktaya tekrar konumlayabilme kabiliyetine denir. Tekrarlanabilirlik hassasiyetten farklıdır. Tekrarlanabilirlik ve hassasiyet arasındaki ilişki; ilk olarak robot T hedef noktasına bileği hareket ettirmek için programlanmış olsun. Hassasiyeti kendini sınırladığı için robot A noktasına gelmiş olsun. A ve T arasındaki mesafe hassasiyettir. Daha sonra robota önce programlandığı noktaya geri dönmesi için komut verilsin. Ancak tekrarlanabilirlik sınırlaması nedeni ile robot eli R noktasına gelmiş bulunmaktadır. Sonraki safhalarda robota aynı noktaya geri dönmesi için komut verildiğinde her zaman R noktasına geri dönmeyecektir fakat T etrafındaki noktalara gidecektir. Tekrarlanabilirlik hataları rastgele bir değer alır. Genellikle tekrarlanabilirlik hassasiyetten daha iyidir. Robot kolu ve bileğindeki elemanların hassasiyet bozukluğu tekrarlanabilirlik hatalarının başlıca nedenlerini teşkil eder.

Hareket hızı: Robotun uç efektörü hareket ettirme hızı en fazla yaklaşık 1.5 m/sn'ye kadar değişir. Bu hız robotların yaptıkları işe göre ayarlanabilir. Hız şu faktörler göz önüne alınarak tespit edilmelidir: Hareket ettirilen nesnenin ağırlığı, hareket edilen mesafe ve nesnenin konumlanması gereken hassasiyeti. Atalet problemlerinden dolayı ağır nesneler, hafif nesneler kadar hızlı hareket ettirilemez. Ayrıca, yüksek konumlama hassasiyeti gerektiğinde nesneler daha yavaş hareket ettirilmelidir.

Yük taşıma kapasitesi: Ticari robotların taşıma kapasiteleri çok çeşitlidir. 500 kg'a kadar ağırlık kaldırabilen robotlar bulunmaktadır. "FC Versatran" robotu 1000 kg'lık yükü taşıyabilecek kapasitededir. Bunun yanında "PUMA 250" sadece 1,2 kg yük taşıma kapasitesine sahiptir.

Hareket sistemleri: Ticari robotlarda kullanılan üç temel hareket sistemi vardır. Bunlar; hidrolik, elektrik motoru ve pnömatik yolla hareket veren sistemleridir. Hidrolik tahrik sistemleri genellikle büyük kapasiteli robotlarda kullanılır. Fabrikalarda daha fazla yer kaplarlar. Bu sistemin avantajı robotun mekanik olarak basit olmasını sağlar, yüksek dayanıma sahip ve hızı da yüksektir. Elektrik motorları (DC, step motorlar veya servomotorlar) ile hareket eden robotlar, hidrolik robotlar kadar fiziksel dayanım ve hıza sahip değildirler fakat bunların hassasiyeti ve tekrarlanabilirlikleri daha iyidir. Hidrolik güç ünitesine gerek olmadığı için daha az yer kaplarlar. Pnömatik sistemli robotlar tipik olarak daha küçüktür ve diğer iki tipten daha az karmaşık bir yapıya sahiptir. Genellikle, al/yerleştir ve diğer basit hareketleri gerektiren işler için bu robotlar kullanılır.

DEĞERLENDİRME ÇALIŞMALARI

- 1) Robot nedir?
- 2) Robotların endüstrideki yeri nedir?
- 3) Robotlar daha çok nerelerde tercih edilmektedir?
- 4) Robotla insanı değişik açılardan karşılaştırınız.
- 5) Robot sensörlerini yazınız. Bu sensörlerin kullanım amaçlarını belirtiniz.
- 6) Robotlar nasıl bir iş yapacağına kendisi mi karar verir?
- 7) Robotların istenilen şekilde çalışması için ne şekilde programlanırlar?
- 8) Robot programlama dillerinden birkaçını yazınız.
- 9) Robotlar ne tür hareketleri yapabilirler?
- 10) Robotlarda çalışma zarfı ne demektir?
- 11) Robotların hareket koordinatları nelerdir?
- 12) Yapı ve kullanım alanlarına göre değerlendirildiklerinde genel olarak robot çeşitleri nelerdir?
- 13) Robotlar endüstrinin hangi alanlarında ne amaçla kullanılmaktadır?
- 14) Robotların çalışma ve hareket kapasiteleri hakkında bilgi veriniz.

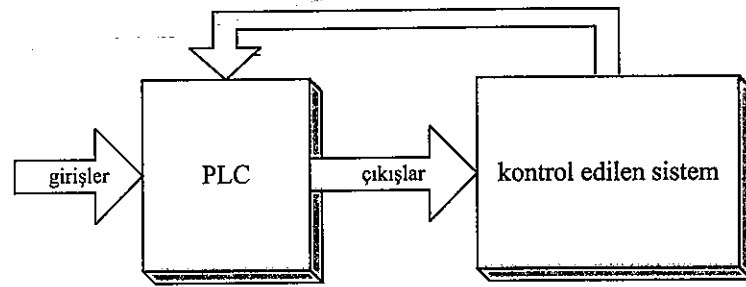
ÜNİTE 19 PROGRAMLANA BİLEN MANTIKSAL DENETLEYİCİLER (PLC)

19.1 PLC SİSTEMİ

19.1.1 PLC Nedir?

Mikroişlemciler, mikrodeneleyiciler ve dijital elemanlarla yapılan sistemlerin geliştirilebilir esnekliğe sahip oluşu, modülerliği ve birçok fonksiyonun küçük bir hacimde toplanması bu tip sistemlerin daha çok tercih edilmesine neden olmuştur.

PLC; hafızasına yüklenmiş olan programa göre girişleri ve geri besleme işaretlerini değerlendirip istenilen denetim ve kontrol sinyallerini üreten, bu sinyallere göre çıkışındaki sistemleri kontrol eden “özel amaçlı” mikrobilgisayardır.



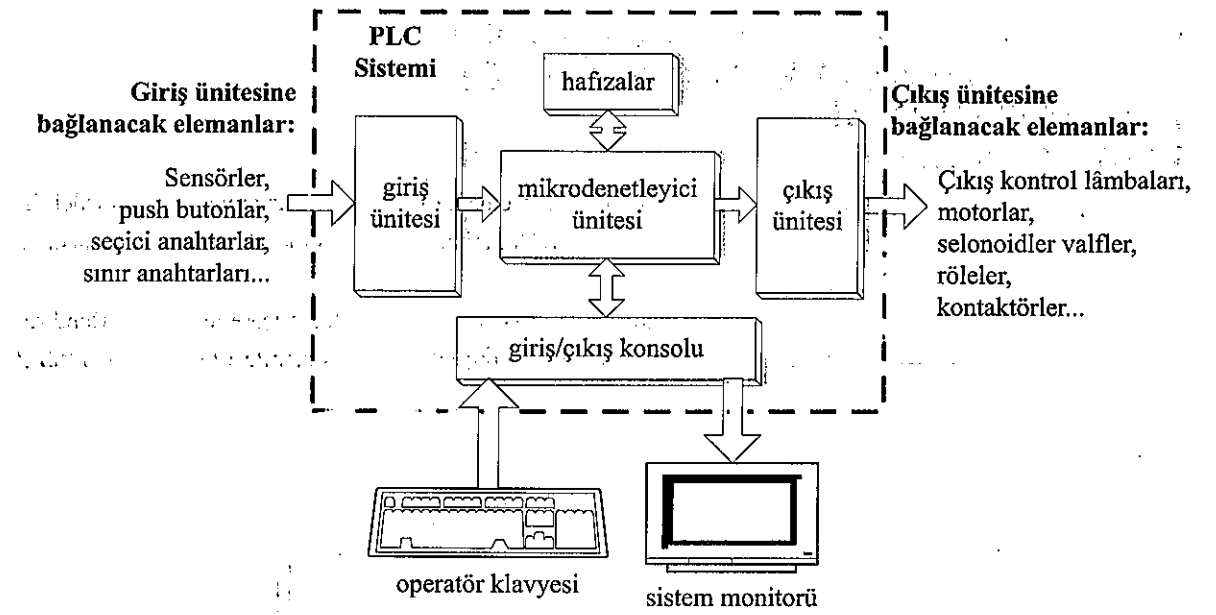
Şekil 19.1 : PLC'nin genel kullanım diyagramı

PLC sistemini kullanmak için gelişmiş bir programlama dili bilmeye, içyapısını veya kullanılan mikroişlemcinin komut kodlarını (makine dilini) bilmeye gerek yoktur. PLC sistemlerinde önemli olan, giriş ve çıkış bağlantılarının dikkatlice yapılması, ihtiyaca uygun programın sisteme girilmesi yeterli olacaktır (Bu programı hazırlayabilmek için genellikle çok basit komutlardan oluşan bir programlama dilini bilmek gereklidir.). PLC cihazları programlanırken kullanıcı hatalarını azaltan özelliği olduğundan dolayı teknik elemanlar endüstriyel uygulamalarda çok rahatlıkla programlayıp kullanabilir. PLC cihazının kullanılan yere ve işlem kapasitesine cevap verecek ve gelişmelere açık olacak şekilde seçilmesi önemlidir. PLC cihazlarının tercih edilmelerinin en önemli sebepleri ise güvenilir olması, düşük güç harcamaları ve değişik endüstriyel uygulamalara cevap verecek esneklikte olmalarıdır.

19.1.2 PLC'nin Yapısı

Bir PLC cihazı mikroişlemci, hafıza, giriş/çıkış birimleri, programlayıcı birim ve güç kaynağı gibi temel ünitelerden oluşur. PLC'ler programlandıktan sonra bir monitöre, disket gibi hafıza elemanlarına veya bir bilgisayara (PC) ihtiyaç duymadan sonsuz bir çalışma döngüsüne girerler.

Şekil 19.2'de bir PLC'deki temel üniteler ve bunlar arasındaki sistem yolları (veri yolu, adres yolu, kontrol yolu) görülmektedir.



Şekil 19.2 : PLC'nin blok diyagramı

Mikrodeneleyici ünitesi PLC sisteminin beynidir denebilir. İçerisinde çeşitli lojik devreler vardır. Mikrobilgisayar sisteminin hafızasında depolanmış PLC programı, sayıcı/zamanlayıcı gibi alt sistemlerin yanında, çeşitli sensör devrelerinden gelen giriş bilgilerini okuyup istenilen çalışmayı gerçekleştirmek için gerekli bilgileri çıkış veya kontrol ünitesine gönderir.

Giriş ve çıkış aygıtlarında dışarıdan gelen giriş ve dışarıya verilen çıkış sinyallerini, mikrobilgisayarın tanıyıp işleyebileceği seviyelere dönüştüren ve sistem ile çevre birimleri arasında tampon görevi gören, sistemi zararlı elektriksel ve manyetik etkilerden koruyan devreler içerir. Sistemin girişine, geri besleme gerilimleri, sensörler ve kontrol anahtarlarından gelen sıcaklık güç tüketimi, hız gibi duruma ve ortama göre değişen analog veya dijital giriş bilgileridir. Çıkışta ise motorlara, ısıtıcılara elektrikle kontrol edilen mekanik, pnömatik veya hidrolik hareket sistemlerine gönderilen elektriksel çıkış bilgileridir. Buradaki giriş birimlerinde elde edilen verilere göre kontrol sistemi cihazı çalıştırır veya durdurur. Sistemdeki tüm giriş ve çıkış uçları özel adreslere sahiptirler. Bu adresler mikrobilgisayar tarafından bilinir.

Giriş ve çıkış aygıtlarında bulunan uçlarda çeşitli AC ve DC voltaj değerleri ve dijital çıkış seviyeleri bulunabilir. Tablo 19.1'de tipik bir PLC'de bulunan giriş/çıkış gerilim seviyeleri bulunmaktadır.

Tablo 19.1 : PLC cihazlarında tipik giriş/çıkış gerilim seviyeleri

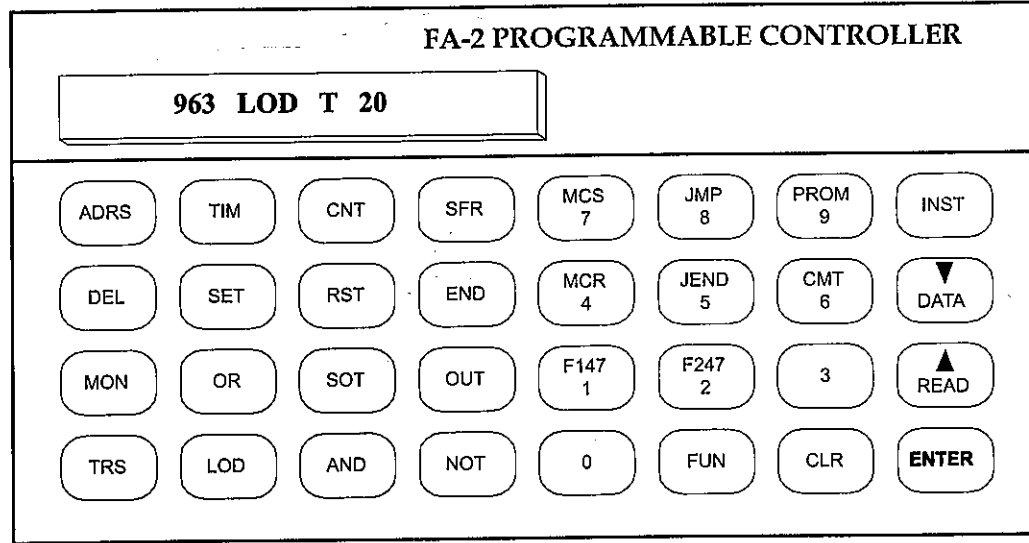
Giriş gerilim seviyeleri	Çıkış gerilim seviyeleri
12 volt AC/DC	12/48 volt AC
24 volt AC/DC	120 volt AC
48 volt AC/DC	230 volt AC
120 volt AC/DC	120 volt DC
230 volt AC/DC	230 volt DC
5 volt TTL	5 volt TTL

PLC'nin belleği, sayısal işlemlerin gerçekleştirilmesini sağlayacak mantık, zamanlama, karşılaştırma, sıralama, sayma ve aritmetiksel işlemler için kullanılan komutların istenilen işlemleri yapması için programlandığı ve depolandığı yerdir. PLC'lerde, bilgileri saklı tutmak için kalıcı veya geçici hafıza birimleri kullanılmaktadır. PLC'lerde kullanılan hafıza birimlerinden bazıları şunlardır; RAM, EPROM, EEPROM, PROM. PLC'lerde kullanılan hafızaların özelliklerine göre kullanım yerleri;

EPROM veya PROM: Sistemin kontrol, denetim ve programlanmasını sağlayan işletim sisteminin (PLC sistem programı) tutmaya yarayan hafıza elemanıdır.

RAM: PLC çalışırken dış çevreden veya programın ürettiği verileri geçici olarak saklamada kullanılır.

EEPROM: Sistem kapatıldığında dahi saklanması gerekli görülen verilerin tutulması için kullanılan elektriksel olarak silinip yazılabilen ve enerjisi kesildiğinde verileri saklı tutan bir hafıza elemanıdır.



Şekil 19.3 : Tipik bir PLC cihazının programlama konsolu

PLC cihazlarını programlamak için iki yöntem kullanılır. Bunlardan birincisi “ladder diyagram dili” diğeryse “PLC sembolik komutları”. Günümüzde birçok PLC kullanıcıya PC (kişisel bilgisayar) destekli program hazırlama ortamı sunmaktadır. Programcı kişisel bilgisayarda çalışan bu paket programlar ile PLC cihazının programlanmasını ve kontrolünü gerçekleştirebilmektedir. Ayrıca bu paket programlar (PLC programlama, programı) ladder diyagram dili ile hazırlanan kaynak programdan sembolik komutların elde edilmesine veya sembolik komutlar ile yazılmış PLC programından ladder diyagram dilinin elde edilmesine imkân sağlayabilmektedirler.

19.2 MERDİVEN (LADDER) DİYAGRAMI

Ladder diyagramı, kumanda sisteminin (kontaktör ile yapılan uygulamalar) şalt resimlerine benzemektedir. Temel mantık elemanları, kontaklar, bobinler ve kutular şeklindedir. Ladder diyagram sembolleri ile oluşturulan şemalara da burada devre denilmektedir. Fiziksel olarak PLC'ye bağlanmış bir giriş ya da kumanda sistemdeki yardımcı röle, zaman rölesi, kontaktör yardımcı kontakları, burada da kontak olarak adlandırılan bir sembolle gösterilir. Normalde açık bir kontak kapandığı zaman enerji akışını sağlar. Bir kontak, normalde kapalı da olabilir. Enerji akışı kontak açtığı zaman kesilir. Fiziksel olarak bağlı bir çıkış ya da kumanda sistemdeki yardımcı röle, zaman rölesi bobinleri, burada da bobin olarak gösterilir. Bu bobine enerji geliyorsa bobin kontakları konum değiştirir.

Tablo 19.2 : Merdiven diyagramında kullanılan bazı semboller

Semboller	Açıklama
I0.0 — — —	Normalde açık kontak
I0.0 — / —	Normalde kapalı kontak
Q0.0 —()—	Çıkış
—(END)—	Son
— NOT —	Değil
CU C0 R PV CTU	Yukarı sayıcı
IN T32 PT TON	Çekmede gecikmeli zaman rölesi

CPU'daki karmaşık işlemler için kutu şeklinde semboller kullanılır. Kutuların zaman rölesi, sayıcı, matematik işlemler gibi değişik işlevleri vardır ve program yazımını kolaylaştırır. Tablo 19.2'de ladder diyagram sembolleri görülmektedir.

19.3 PROGRAMLAMA DİYAGRAM DİLİ

PLC'ler, giriş, çıkış, zamanlayıcı, sayıcı, veri işleme ve program denetiminde kullanılan kontrol komutları ile programlanırlar. PLC'ler kullanılarak bir sistemin kontrolü yapılmak istendiğinde, önce sistemin klâsik yöntemler ile bilinen kumanda devresi çizilir. Daha sonra bu kumanda devresini gerçekleştirmek için PLC'lerin programlanmasında kullanılan ladder diyagramı hazırlanır. Son olarak da hazırlanan ladder diyagramı baz alınarak PLC komutları yazılır. PLC'nin programlanması direkt olarak sembolik komutlar kullanarak da yapılabilir.

PLC programları, istenilen işlevleri yerine getiren bir dizi komut satırlarında oluşur. Ladder programlarındaki grafik gösterimler yerine, burada metin formatı kullanılır.

19.3.1 PLC sembolik (mnemonic) komutları

Otomasyon endüstrisinde birçok PLC cihazı üreten firmalar bulunmaktadır. PLC komutlarının belirli bir standardı bulunmadığı için her firmanın kendi cihazlarında kullandığı komutlar farklıdır. Buna rağmen PLC'lerin çoğunda benzer işi yapan komutlar vardır ve bu komutların yalnızca kısaltmalarında küçük farklar vardır. Örneğin bir firma “YÜKLE” işlevini karşılamak için “LOD” komutu kullanırken diğer bir firma bunu “LD” ile ifade etmiştir. Burada komutlar hakkında genel bir fikir vermek maksadı ile örnek bir PLC cihazının komutları tanıtılmıştır.

Tablo 19.3 : PLC 'de kullanılan örnek komutlar

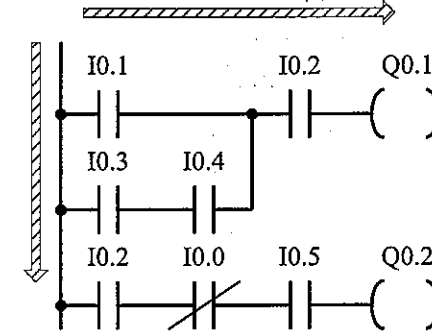
Komut kısaltması	Açık yazılımı	Açıklama
= n	Output	Çıkış komutu, lojik yığının tepe değerini n ile belirtilen adrese kopyalar.
A n	And	Bu komut, süregelen lojik sonucuyla n adresindeki bit değerini lojik olarak AND'ler. Sonuç, yeni süregelen lojik sonuç olarak lojik yığının en üstüne yazılır.
ALD	And Load	Bu komut, lojik yığının en üst ve bir altındaki bitlerinin lojik olarak AND'lenmesini sağlar. Sonuç yığının en üstüne yazılır ve yığının derinliği 1 azalır.
AN n	And Not	Bu komut, süregelen lojik sonucuyla n adresindeki bit değerini lojik olarak AND NOT'lar. Sonuç, yeni süregelen lojik sonuç olarak lojik yığının en üstüne yazılır.
CTU cxxx,pv	Counter	"Yukarı say" komutu (CTU), CU girişinin (lojik yığını n ikinci biti) her pozitif yükselen kenarında (0'dan 1'e dönüşünde) sayar. Eğer cxxx >= Ayar Değeri (pv) ise çıkış biti set edilir. Reset (R) girişi (lojik yığının birinci biti) gelirse sıfırlanır. Maksimum sayma ve ayar değeri 32767'dir.
JMP n	Jump	Bu komut, program içerisinde belirtilen etikete (n) atlama sağlar.
LD n	Load	Bu komut, n adresindeki bitin değerini lojik yığının en üstüne kopyalar. Yığındaki diğer bit değerleri "1" aşağıya kayar.
LDN n	Load Not	Bu komut, n adresindeki bitin değerinin "değilini" lojik yığının en üstüne kopyalar. Yığındaki diğer bit değerleri "1" aşağıya kayar.
LPP	Lojik kesme	Bu komut lojik yığının en tepesindeki bit değerini atar. İkinci bit değeri yığının tepesine yükselir, diğer değerler "1" yukarı kayar.
LPS	Lojik itme	Bu komut, lojik yığının tepesindeki değeri kopyalarak üste yazar ve yığındaki diğer bitleri "1" aşağı kaydırır. Yığının en altındaki bit itilerek atılır ve yok olur.
MEND	Main End	Kullanıcı programını bitirmek için kullanılır. Her zaman için ana programın (alt programlar hariç) son komutunu oluşturur.
NOT	Not	Bu komut lojik yığının en üstündeki biti değiller; 0'sa 1, 1'se 0 yapar.
O n	Or	Bu komut, lojik yığının en üstteki bitini n adresindeki bitle lojik olarak OR'lar. Sonuç, lojik yığının yeni tepe değeri olur.
OLD	Or Load	Bu komut, lojik yığının birinci ve ikinci bitleri arasında OR işlemi yapar. Sonuç lojik yığının tepesine yazılır. İşlem sonucunda lojik yığın derinliği bir azalır.
ON n	Or Not	Bu komut, n adresindeki bitin değeriyle lojik yığının üst değeri arasında lojik OR NOT işlemi yapar. Sonuç lojik yığının tepe değerine yazılır.
R s bit,n	Reset	Bu komut, s_bit adresinden başlayan N adet biti reset eder (sıfırlar). Eğer s_bit değeri bir sayıcı (C) ya da zaman rölesine (T) aitse hem sayıcı/zaman rölesi biti hem de anlık değerleri sıfırlanır.
S s bit,n	Set	Bu komut, s_bit adresinden başlayan N adet biti set eder (1 yapar).
TON txxx,pt	Timer	Çekmede gecikmeli zaman rölesi (TON) kutusu, lojik yığının birinci biti 1 olduğunda maksimum zaman değerine kadar çalışır. Eğer txxx >= Ayar Değeri (pt) ise çıkış biti T set edilir. Sayma esnasında lojik yığının birinci biti 0 olursa sayma durur ve röle sıfırlanır.

NOT: SIMENS firmasının SIMATIC S7-200 model PLC cihazına ait örnek komutlar

19.3.2 Ladder diyagramının PLC komutlarına dönüştürülmesi yöntemi

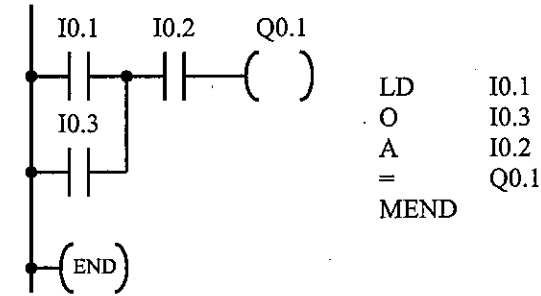
Ladder diyagramındaki sembollerden program kodlarına dönüşüm yapmanın iki yöntemi vardır. Birincisi bilgisayar yazılımlarını kullanarak ladder diyagramındaki sembolleri program kodlarına dönüştürmek, diğeri ise diyagramdaki sembollere ve bağlantılara bakarak program kodlarını yazmak. Burada ladder diyagramına bakarak PLC program kodlarına nasıl dönüşüm yapıldığını inceleyeceğiz.

Ladder diyagramlarından PLC program kodlarını elde etmek için ladder diyagramını soldan sağa ve yukarıdan aşağıya doğru takip etmemiz gerekmektedir. Ladder diyagramındaki kollar, solunda dikey çizgi ile birbirine bağlanır.



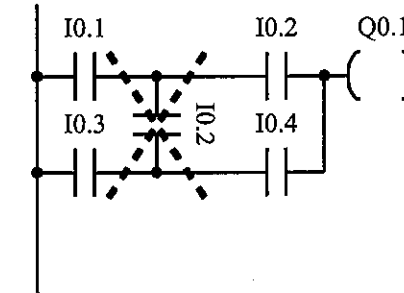
Şekil 19.4 : Sinyal akışı soldan/sağa yukarıdan aşağıya doğrudur

Ladder diyagramından elde edilecek program kodu yükle "LOAD" komutu ile başlar, "END" komutu ile biter. Şekil 19.5'de S7 için hazırlanmış ladder diyagramı ve program komutları görülmektedir.



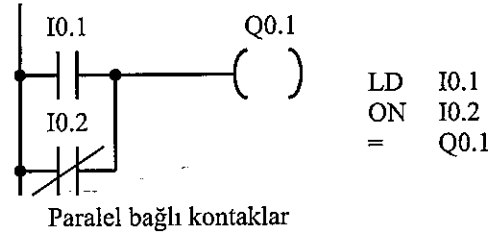
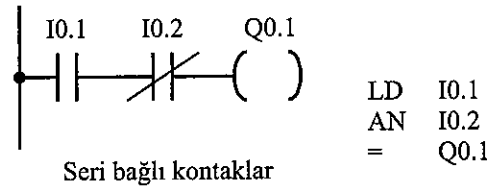
Şekil 19.5 Ladder diyagramında başlangıç ve bitiş komutları

Ladder diyagramlarında dikey kontak bağlanmaz.



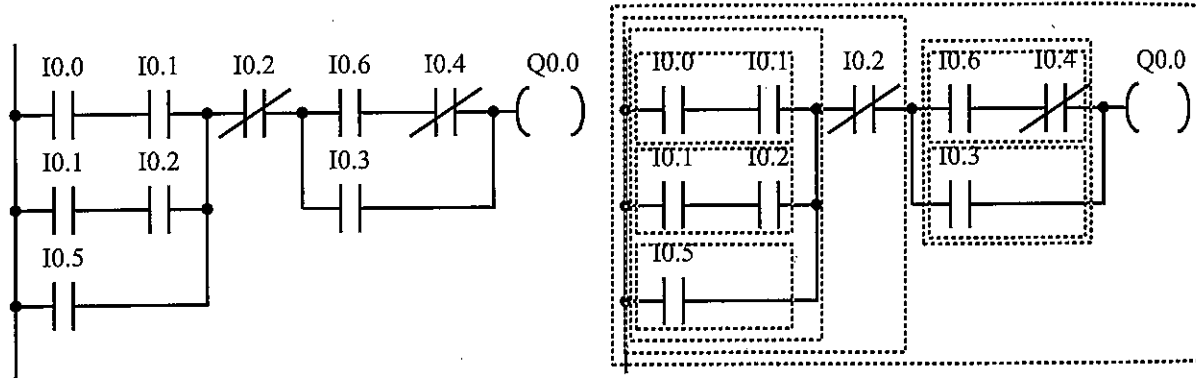
Şekil 19.6 : Yanlış kontak bağlantısı

Seri bağlı iki kontak "AND" komutu ile paralel bağlı iki kontak "OR" komutu ile birbirine bağlanır.



Şekil 19.7 : Kontakların seri ve paralel bağlantısı

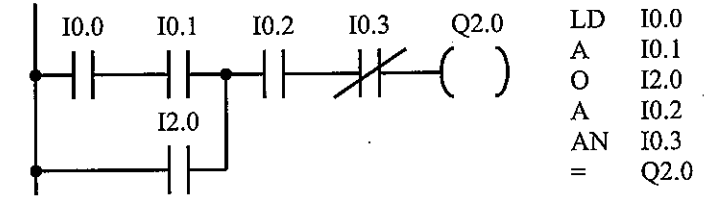
Devre seri ve paralel temel parçalara ayrılır.



Şekil 19.8 : Devre seri ve paralel temel parçalara ayrılır

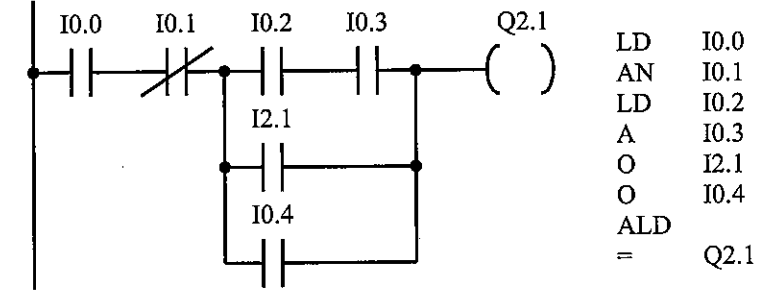
Şekil 19.8'deki karmaşık devrenin PLC komutlarını yazabilmek için devrenin gruplara ayrılmayısıyla daha kolay olacaktır.

LD	I0.0
A	I0.1
LD	I0.1
A	I0.2
OLD	
O	I0.5
AN	I0.2
LD	I0.6
AN	I0.4
O	I0.3
ALD	
=	Q0.0



Şekil 19.9 : Paralel ve Seri bağlantılı devre

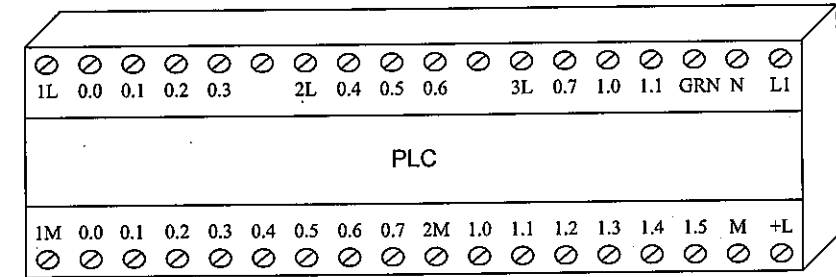
Seri/paralel bağlantılı ladder diyagramından PLC komutlarının elde edilmesi;



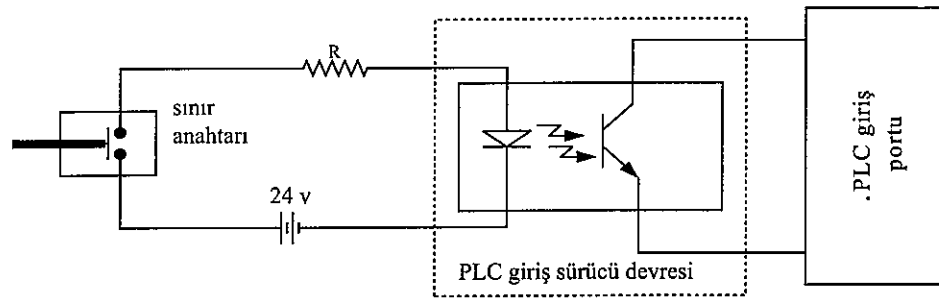
Şekil 19.10 : Seri/paralel bağlantılı devre

19.5 GİRİŞ/ÇIKTI CİHAZLARI

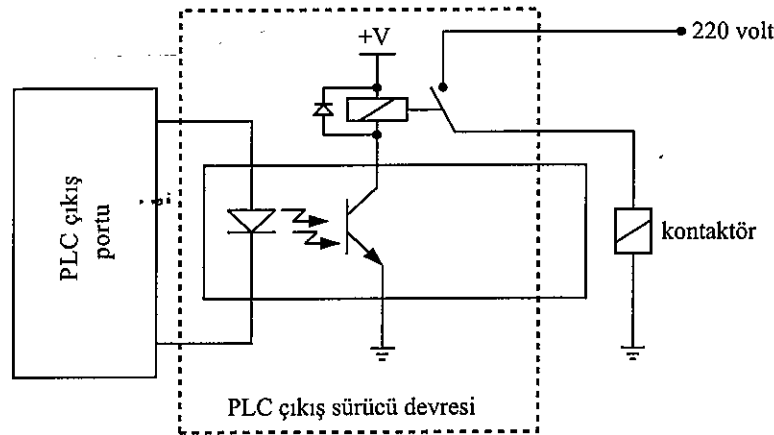
Giriş/çıkış modülleri, PLC'yi motorlar, lâmbalar, limit anahtarları ile çevre birimlerine bağlantı sağlamaktadır. PLC'ler programlandıktan sonra kontrol edecekleri sistemle giriş ve çıkış bağlantıları yapılır. Bu bağlantılar PLC üzerindeki bağlantı noktaları ile yapılır. PLC üzerindeki bu bağlantılar PLC cihazının içerisinde bulunan mikrobilgisayar ile elektriksel yalıtımı yapılmış giriş/çıkış noktalarıdır.



Şekil 19.11 : Tipik bir PLC cihazının giriş/çıkış bağlantı noktaları



Şekil 19.12 : PLC giriş ünitesi



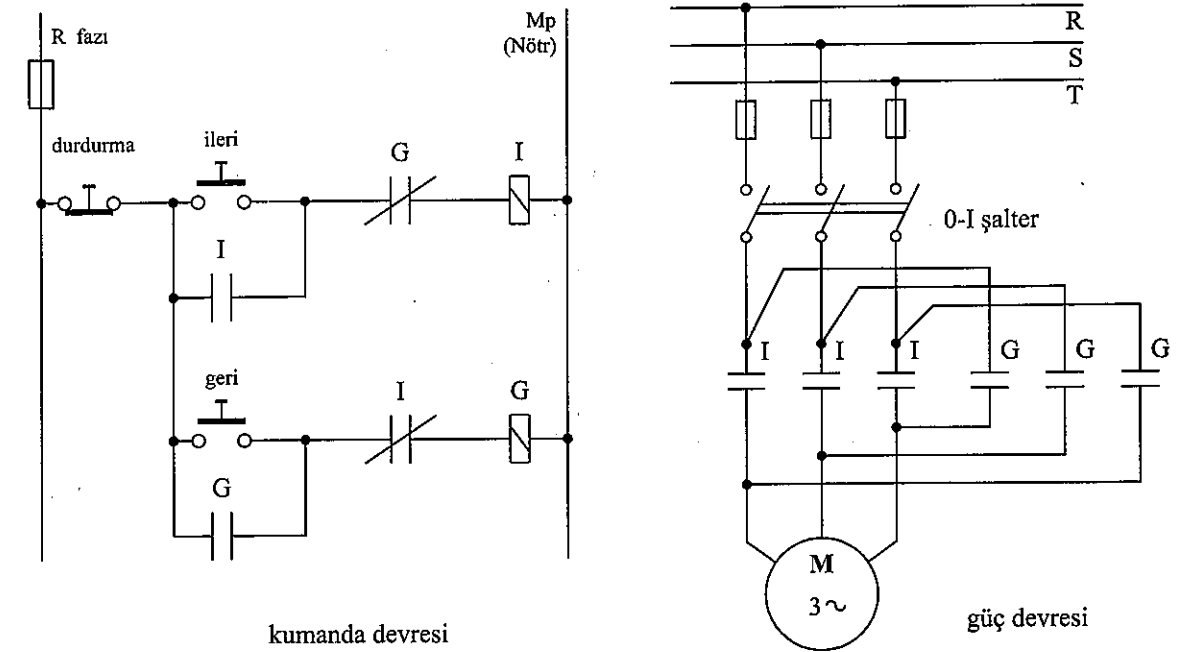
Şekil 19.13 : PLC çıkış ünitesi

PLC cihazlarında her giriş ve çıkış noktalarının belirli bir adresi vardır. PLC'ler genel olarak bu giriş/çıkış noktalarındaki elektronik (tiryak, transistör, vb.) veya mekanik rölelere kumanda eder. PLC programlarında değişen giriş değerlerine göre çıkışın kontrolünü sağlar. PLC cihazlarının giriş noktalarına konum ve mesafe sensörleri, sınır anahtarları (Optik, mekanik veya magnetik olabilir). PLC çıkışlarına ise kontaktörler, ikaz-uyarı lambaları gibi çıkış elemanları bağlanabilir.

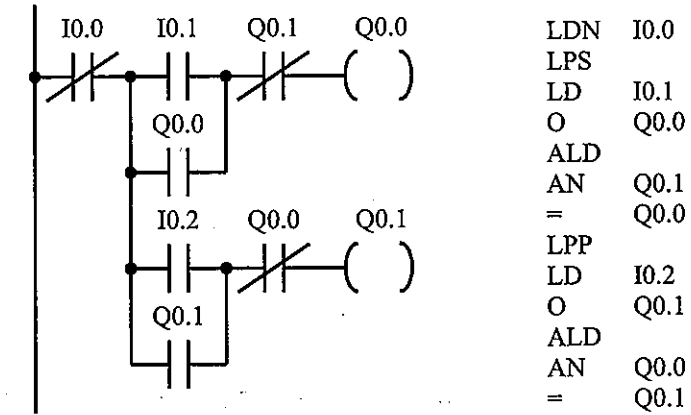
19.5 PLC İLE ÜÇ FAZLI ASENKRON MOTORUN DEVİR YÖNÜNÜ DEĞİŞTİRME

Bu devrenin amacı 3 fazlı asenkron motoru ileri/geri yönde çalıştırmaktır. Motorların ileri/geri yönde çalıştırılması için herhangi iki fazının yerinin değiştirilmesi devir yönünün de değişmesine neden olur. Şekil 19.14'deki kumanda devresi, iki fazın değiştirilmesini sağlamakta ve güç devresi ile de motora kumanda etmektedir.

İleri butonuna basıldığında akım geri kontaktörünün kapalı kontağı üzerinden ileri kontaktörünü enerjilendirmekte, kontaktör güç devresindeki ve kumanda devresindeki kontaklarının konumu değişmektedir (Açık kontaklar kapanır, kapalı kontaklar açılır). Motor ileri yönde çalışırken geri yönde çalışmasına mâni olmaktadır. (İleri kontaktörünün kapalı kontağı ile) Motoru geri yönde çalıştırmak için ise önce durdurma butonuna basılmalı, sonra geri butonuna basılmalıdır. Geri kontaktörü çalıştığında güç devresinde R ve T fazlarının yeri değişmekte dolayısı ile motorun devir yönü değişmektedir. Aynı şekilde geri kontaktörünün kapalı kumanda kontağı ileri kontaktörünün çalışmasına engel olmaktadır. İki kontaktör aynı anda çalışırsa R ve T fazları kısa devre olur. Bu istenmeyen bir durumdur. Gerektiğinde kısa devre veya aşırı akım durumlarında bu devreye koruma röleleri de eklenebilirdi.



Şekil 19.14 : Üç fazlı asenkron motorun ileri/geri yönde çalıştırılmasına ait kumanda ve güç devresi



Şekil 19.15 : Kumanda devresinin ladder diyagramı ve komut listesi

DEĞERLENDİRME ÇALIŞMALARI

- 1) PLC nedir? PLC'nin mikrobilgisayarlardan farkı nedir?
- 2) PLC'nin iç yapısını genel olarak blok diyagramda gösteriniz.
- 3) Merdiven diyagramı nedir?
- 4) Merdiven diyagramında kullanılan sembollerin görevlerini belirtiniz.
- 5) PLC'ler kaç şekilde programlanabilmektedir?
- 6) PLC'lerin giriş/çıkış bağlantılarını süren devrelerin çalışmasını şekil çizerek anlatınız.

MOTOROLA 6802 KOMUT SETİ

DURUM KODU KAYDEDİCİSİ KOMUTLARI		Adres. Modu			Mantıksal işlemler	Durum kod. kay.					
		İNHER				S	4	3	2	1	0
İşlemler Mnemonic		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0→C	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	2	1	0→I	●	R	●	●	●	●
Clear Overflow	CLV	0A	2	1	0→V	●	●	●	●	R	●
Set Carry	SEC	0D	2	1	1→C	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	2	1	1→I	●	S	●	●	●	●
Set Overflow	SEV	0B	2	1	1→V	●	●	●	●	S	●
Accmltr A → CCR	TAP	06	2	1	A→CCR	A'nın değeri					
CCR → Accmltr A	TPA	07	2	1	CCR→A	●	●	●	●	●	●

285

ZILOG Z80 komut seti

Sembolik Kod	Hex Kod	Sembolik Kod	Hex Kod	Sembolik Kod	Hex Kod	Sembolik Kod	Hex Kod	Sembolik Kod	Hex Kod
ADC A, A	8F	ADD IY, IY	FD 29	CPDR EDB9	ED B9	IN C, (C)	ED 48	JP PO, ss	E2 16-bit
ADC A, B	88	ADD IY, SP	FD 39	CPI EDA1	ED A1	IN D, (C)	ED 50	JP Z, ss	CA 16-bit
ADC A, C	89	AND A	A7	CPL EDB1	ED B1	IN E, (C)	ED 58	JR C, e	38e
ADC A, D	8A	AND B	AO	DAA	27	IN F, (C)	ED 70	JR NC, e	30e
ADC A, E	8B	AND C	A1	DEC A	3D	IN H, (C)	ED 60	JR NZ, e	20e
ADC A, H	8C	AND D	A2	DEC B	05	IN L, (C)	ED 68	JR Z, e	28e
ADC A, L	8D	AND E	A3	DEC BC	0B	INC A	3C	JR e	18e
ADC A, s	CE 8-bit	AND H	A4	DEC C	0D	INC B	04	LD A, A	7F
ADC A, (HL)	8E	AND L	A5	DEC D	15	INC BC	03	LD A, B	78
ADC A, (IX+d)	DD 8E d	AND S	E6 8-bit	DEC DE	1B	INC C	OC	LD A, C	79
ADC A, (IY+d)	FD SE d	AND (HL)	A6	DEC E	1D	INC D	14	LD A, D	7A
ADC HL, BC	ED 4A	AND (IX+d)	DD A6 d	DEC H	25	INC DE	13	LD A, E	7B
ADC HL, DE	ED 5A	AND (LY+d)	FD A6 d	DEC HL	2B	INC E	1C	LD A, H	7C
ADC HL, HL	ED 6A	BIT b, s		DEC IX	DD 2B	INC H	24	LD A, I	ED 57
ADC HL, SP	ED 7A	CALL ss	CD 16-bit	DEC IY	FD 2B	INC HL	23	LD A, L	7D
ADD A, A	87	CALL C, ss	DC 16-bit	DEC L	2D	INC IX	DD 23	LD A, s	3E 8-bit
ADD A, B	80	CALL M, ss	FC 16-bit	DEC SP	3B	INC IY	FD 23	LD A, R	ED 5F
ADD A, C	81	CALL NC, ss	D4 16-bit	DEC (HL)	35	INC L	2C	LD A, (BC)	OA
ADD A, D	82	CALL NZ, ss	C4 16-bit	DEC (IX+d)	DD 35 d	INC SP	33	LD A, (DE)	1A
ADD A, E	83	CALL P, ss	F4 16-bit	DEC (IY+d)	FD 35 d	INC (HL)	34	LD A, (HL)	7E
ADD A, H	84	CALL PE, ss	EC 16-bit	DI F3	F3	INC (IX+d)	DD 34 d	LD E, (IX+d)	DD 7E d
ADD A, L	85	CALL PO, ss	E4 16-bit	DJNZ E	10 e	INC (IY+d)	FD 34 d	LD E, (IY+d)	FD 7E d
ADD A, s	C6 8-bit	CALL Z, ss	CC 16-bit	EI FB	FB	IND	ED AA	LD A, (ss)	3A 16-bit
ADD A, (HL)	86	OCF	3F	EX (SP), HL	E3	INDR	ED BA	LD B, A	47
ADD A, (IX+d)	DD 86 d	CP A	BF	EX (SP), IX	DE E3	INI	ED A2	LD B, B	40
ADD A, (IY+d)	FD 86 d	CP B	B8	EX (SP), IY	FE E3	INR	ED B2	LD B, C	41
ADD HL, BC	09	CP C	B9	EX AF, AF	08	JP ss	C3 16-bit	LD B, D	42
ADD HL, DE	19	CP D	BA	EX DE, HL	EB	JP (HL)	E9	LD B, E	43
ADD HL, HL	29	CP E	BB	EXX	D9	JP (IX)	DD E9	LD B, H	44
ADD HL, SP	39	CP H	BC	HALT	76	JP (IY)	FD E9	LD B, L	45
ADD IX, BC	DD 09	CP L	BD	IM 0	ED 46	JP C, ss	DA 16-bit	LD B, s	06 8-bit
ADD IX, DE	DD 19	CP s	FE 8-bit	IM 1	ED 56	JP M, ss	FA 16-bit	LD B, (HL)	46
ADD IX, IX	DD 29	CP (HL)	BE	IM 2	ED 5E	JP NC, ss	D2 16-bit	LD B, (IX+d)	DD 46 d
ADD IX, SP	DD 39	CP (IX+d)	DD BE d	IN A, (s)	DB 8-bit	JP NZ, ss	C2 16-bit	LD B, (IY+d)	FD 46 d
ADD IY, BC	FD 09	CP (IY+d)	FD BE d	IN A, (C)	ED 78	JP P, ss	F2 16-bit	LD BC, ss	01 16-bit
ADD IY, DE	FD 19	CPD EDA9	ED A9	IN B, (C)	ED 40	JP PE, ss	EA 16-bit	LD BC, (ss)	ED 4B 16-bit

ZILOG Z80 komut seti

Sembolik Kod	Hex Kod	Sembolik Kod	Hex Kod
LD H, B	60	LD (HL), A	77
LD H, C	61	LD (HL), B	70
LD H, D	62	LD (HL), C	71
LD H, E	63	LD (HL), D	72
LD H, H	64	LD (HL), E	73
LD H, L	65	LD (HL), H	74
LD H, s	26 8-bit	LD (HL), L	75
LD H, (HL)	66	LD (HL), 5	36 8-bit
LD H, (IX+d)	DD 66 d	LD (IX+d), A	DD 77 d
LD H, (IY+d)	FD 66 d	LD (IX+d), B	DD 70 d
LD HL, ss	21 16-bit	LD (IX+d), C	DD 71 d
LD HL, (ss)	2 A 16-bit	LD (IX+d), D	DD 72 d
LD I, A	ED 47	LD (IX+d), E	DD 73 d
LD IX, ss	DD 21 16-bit	LD (IX+d), H	DD 74 d
LD IX, (ss)	DD 2A 16-bit	LD (IX+d), L	DD 75 d
LD IY, ss	FD 21 16-bit	LD (IX+d), s	DD 36 d 8-bit
LD IY, (ss)	FD 2A 16-bit	LD (IY+d), A	FD 77 d
LD L, A	6F	LD (IY+d), B	FD 70 d
LD L, B	68	LD (IY+d), C	FD 71 d
LD L, C	69	LD (IY+d), D	FD 72 d
LD L, D	6A	LD (IY+d), E	FD 73 d
LD L, E	6B	LD (IY+d), H	FD 74 d
LD L, H	6C	LD (IY+d), L	FD 75 d
LD L, L	6D	LD (IY+d), s	FD 36 d 8-bit
LD L, s	2E 8-bit	LD (ss), A	32 16-bit
LD L, (HL)	6E	LD (ss), BC	ED 43 16-bit
LD L, (IX+d)	DD 6E d	LD (ss), DE	ED 53 16-bit
LD L, (IY+d)	FD 6E d	LD (ss), HL	22 16-bit
LD R, A	ED 4F	LD (ss), IX	DD 22 16-bit
LD SP, HL	F9	LD (ss), IY	FD 22 16-bit
LD SP, IX	DD F9	LD (ss), SP	ED 73 16-bit
LD SP, IY	FD F9	LDD	ED A8
LD SP, ss	31 16-bit	LDDR	ED B8
LD (BC), A	02	LDI	ED AO
LD (DE), A	12	LDIR	ED BO
		NEC	ED 44

Sembolik Kod	Hex Kod	Sembolik Kod	Hex Kod	Sembolik Kod	Hex Kod
RR B	CB 18	SBC A, L	9D		
RR C	CB 19	SBC A, s	DE 8-bit		
RR D	CB 1A	SBC A, (HL)	9E		
RR E	CB 1B	SBC A, (IX+d)	DD 9Ed		
RR H	CB 1C	SBC A, (IY+d)	FD 9Ed		
RR L	CB 1D	SBC HL, BC	ED 42		
RR (HL)	CB 1E	SBC HL, DE	ED 52		
RR (IX+d)	DD CB d IE	SBC HL, HL	ED 62		
RR (IY+d)	FD CB d IE	SBC HL, SP	ED 72		
RRA	IF	SCF	37		
RRC A	CB 0F	SET bs(h)	Bk Ek Table		
RRC B	CB 08	SLA A	CB 27		
RRC C	CB 09	SLA B	CB 20		
RRC D	CB 0A	SLA C	CB 21		
RRC E	CB 0B	SLA D	CB 22		
RRC H	CB 0C	SLA E	CB 23		
RRC L	CB 0D	SLA H	CB 24		
RRC (HL)	CB 0E	SLA L	CB 25		
RRC (IX+d)	DD CB d 0E	SLA (HL)	CB 26		
RRC (IY+d)	FD CB d 0E	SLA (IX+d)	DD CB d 26		
RRCA	0F	SLA (IY+d)	FD CB d 26		
RRD	ED 67	SRA A	CB 2F		
RST 00H	C7	SRA B	CB 28		
RST 08H	CF	SRA C	CB 29		
RST 10H	D7	SRA D	CB 2A		
RST 18H	DF	SRA E	CB 2B		
RST 20H	E7	SRA H	CB 2C		
RST 28H	EF	SRA L	CB 2D		
RST 30H	F7	SRA (HL)	CB 2E		
RST 38H	FF	SRA (IX+d)	DD CB d 2E		
SBC A, A	9F	SRA (IY+d)	FD CB d 2E		
SBC A, B	98	SRA C	CB 29		
SBC A, C	99	SRA D	CB 2A		
SBC A, D	9A	SRA E	CB 2B		
SBC A, E	9B	SRA H	CB 2C		
SBC A, H	9C	SRA L	CB 2D		

KISALTMALARIN VE SEMBOLLERİN ANLAMLARI

Sembol	Açıklaması
S	İşaret bayrağı. Eğer sonucun en önemli biti 1 ise S=1.
Z	Sıfır bayrağı. Eğer işlemin sonucu 0 ise Z=1.
P/V	Eşlik veya taşıma bayrağı. Eşlik (P) ve taşıma (V) aynı bayrağı paylaşır. Mantık işlemlerinde sonucun eşliği bu bayrağı etkilerken aritmetik işlemlerde sonucun taşınması bu bayrağı etkiler. Eğer işlemin sonucu tek ise P/V=0; sonuç çift ise P/V=1; eğer taşıma varsa P/V=1 eğer taşıma yoksa P/V=0.
H	Yarım elde bayrağı (Ara Elde Bayrağı denilebilir). Yapılan aritmetik işlemlerde akümülatörün 4.bitünden 1 elde oluşur veya 1 borç alınursa H=1'dir.
N	Toplama çıkartma bayrağı. Eğer bir önceki işlem çıkartma ise N=1'dir.
C	Elde bayrağı. Sonucun en önemli bitünden elde oluşursa C=1'dir.
Ø	İşlemin sonucuna göre bayrak etkilenir.
●	İşlemin sonucundan bayrak etkilenmez.
0	İşlemin sonucundan bayrak sıfırlanır.
1	İşlemin sonucundan bayrak 1 olur.
X	Bayrağın durumu belirsizdir.
V	İşlem sonucundaki taşımadan P/V bayrağı etkilenir.
P	İşlemi sonucunun eşliğinden P/V bayrağı etkilenir.
r	- CPU yazaçlarından biri. A, B, C, D, E, H, L
s	Komutla ilgili tirm adresleme modları için 8-bitlik adres veya 8-bitlik sayı (0-255)
ss	16-bit adres veya 16-bitlik sayı (0-65535)
ii	IX veya IY indeks yazaçlarından biri.
R	Tazeleme (refresh) sayacı (özel amaçlı yazaç)
d	7-bit kayma
e	Program sayıcağına göre 7-bit kayma

	A	B	C	D	E	H	L
0	C7	C0	C1	C2	C3	C4	C5
1	CF	C8	C9	CA	CB	CC	CD
2	D7	D0	D1	D2	D3	D4	D5
3	DF	D8	D9	DA	DB	DC	DD
4	E7	E0	E1	E2	E3	E4	E5
5	EF	E8	E9	EA	EB	EC	ED
6	F7	F0	F1	F2	F3	F4	F5
7	FF	F8	F9	FA	FB	FC	FD

	A	B	C	D	E	H	L
0	C7	C0	C1	C2	C3	C4	C5
1	CF	C8	C9	CA	CB	CC	CD
2	D7	D0	D1	D2	D3	D4	D5
3	DF	D8	D9	DA	DB	DC	DD
4	E7	E0	E1	E2	E3	E4	E5
5	EF	E8	E9	EA	EB	EC	ED
6	F7	F0	F1	F2	F3	F4	F5
7	FF	F8	F9	FA	FB	FC	FD

289

Komutlar		IMMED.		ABSOLUTE.		ZERO PAGE		ACCUM.		IMPLIED		INDEX X		INDEX Y		ZERO PAGE X		ABSOLUTE X		ABSOLUTE Y		RELATL.		INDIRE.		ZERO PAGE Y		Durum kodu kaydedildi						
Numero.	İşlemler	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	N	Z	C	E	D	V
ADC	A←A+C→A (40)	69	2	2	6D	4	3	65	3	2			61	6	2	71	5	2	75	4	2	7D	4	3				✓	✓	✓	-	-	✓	
AND	A←A∩A→A (41)	79	2	2	7D	4	3	75	3	2			71	6	2	71	6	2	75	4	2	7D	4	3				✓	✓	-	-	-	-	
ASL	C←127.....DA←0				9E	6	3	06	5	2									16	6	2	1E	7	3				✓	✓	✓	-	-	-	
BCC	BRA ON C=0 (2)																								90	2	2	-	-	-	-	-	-	-
BCS	BRA ON C=1 (2)																								B0	2	2	-	-	-	-	-	-	-
BEO	BRA ON Z=1 (2)																								F0	2	2	-	-	-	-	-	-	-
BIT	A←A∩1				7C	4	3	24	3	2																		M7	✓	-	-	-	-	M6
BMI	BRA ON N=1 (2)																								30	2	2	-	-	-	-	-	-	-
BNE	BRA ON Z=0 (2)																								D0	2	2	-	-	-	-	-	-	-
BPL	BRA ON N=0 (2)																								10	2	2	-	-	-	-	-	-	-
BRK											00	7	1															-	-	-	-	-	-	-
BVC	BRA ON V=0 (2)																								30	2	2	-	-	-	-	-	-	-
BVS	BRA ON V=1 (2)																								70	2	2	-	-	-	-	-	-	-
CLC	0←C										18	2	1															-	-	-	-	-	-	0
CLED	0←D										D8	2	1															-	-	-	-	-	-	0
CLI	0←I										58	2	1															-	-	-	-	-	-	0
CLV	0←V										88	2	1															-	-	-	-	-	-	0
CMP	A←A-M (40)	09	2	2	0D	4	3	05	3	2			C1	6	2	D1	5	2	D5	4	2	D0	4	3				✓	✓	✓	✓	✓	✓	
CPX	X←M	10	2	2	1C	4	3	14	3	2																		✓	✓	✓	✓	✓	✓	
CPY	Y←M	20	2	2	2C	4	3	24	3	2																		✓	✓	✓	✓	✓	✓	
DIC	M←M-M				0F	6	3	06	5	2							D6	6	2	D1E	7	3						✓	✓	✓	✓	✓	✓	
DIX	X←X-X										C8	2	1															✓	✓	✓	✓	✓	✓	
DIT	Y←Y-Y										88	2	1															✓	✓	✓	✓	✓	✓	
FOR	A←M→A (41)	49	2	2	4D	4	3	45	3	2			41	6	2	51	5	2	55	4	2	50	4	3				✓	✓	✓	✓	✓	✓	
INC	M←M+1				1E	6	3	16	5	2							E6	6	2	E1E	7	3						✓	✓	✓	✓	✓	✓	
INX	X←X+1										E8	2	1															✓	✓	✓	✓	✓	✓	✓
INY	Y←Y+1										C8	2	1															✓	✓	✓	✓	✓	✓	✓
IRP	Jump to new loca.																											-	-	-	-	-	-	
ISRR	Jump to sub.																											-	-	-	-	-	-	
LDPA	M←A (41)	49	2	2	4D	4	3	45	3	2			A1	6	2	B1	5	2	B5	4	2	B0	4	3				✓	✓	✓	✓	✓	✓	

Mnemonic	İstenenler	INMEDI	ABSOLU.	ZERO PAGE	ACCUM.	IMPLD	INDEX N	INDEX Y	ZERO PAGE X	ABSOLU. X	RELATL	INDRE.	ZERO PAGE Y	Durum kodu kaydedici			
														N	Z	C	D V
LDX	M→X	(D) A2 2 2	AE 4 3	A6 3 2						BE 4 3			BE 4 2	✓	✓	-	-
LDX	M→X	(D) A0 2 2	AC 4 3	A4 3 2					B4 4 2	BC 4 3				✓	✓	-	-
LSR	B→D		4E 6 3	4E 3 2	4A 2 1				5E 6 2	5E 7 3				0	✓	✓	-
NOP	No operation					EA 2 1			15 4 2	ID 4 3				✓	✓	-	-
ORA	A→A	(D) 09 2 2	0D 4 3	05 3 2			01 6 2	11 5 2	15 4 2	ID 4 3				✓	✓	-	-
PSHA	R→S					48 3 1								-	-	-	-
PHP	R→S					08 3 1								-	-	-	-
PLA	S→R					68 4 1								-	-	-	-
PLP	S→R					28 4 1								✓	✓	-	-
ROL	D→D		2E 6 3	2E 3 2	2A 2 1				3E 6 2	3E 7 3				✓	✓	-	-
ROR	D→D		6E 6 3	6E 3 2	6A 2 1				7E 6 2	7E 7 3				✓	✓	-	-
RUI	Return int.					40 6 1								-	-	-	-
RUS	Return sub.					60 6 1								-	-	-	-
SBC	A→A	(D) E9 2 2	ED 4 3	E5 3 2			E1 5 2	F1 5 2	F5 4 2	FD 4 3				✓	✓	3	-
SBC	I→C					38 2 1								-	-	-	-
SHD	I→D					F8 2 1								-	-	-	-
SHI	I→I					78 2 1								-	-	-	-
STA	R→M		3D 4 3	35 3 2			31 6 2	91 6 2	95 4 2	9D 5 3				-	-	-	-
STX	X→M		3E 4 3	3E 3 2					94 4 2				96 4 2	-	-	-	-
STY	Y→M		3C 4 3	3C 3 2										-	-	-	-
TAX	A→X					A4 2 1								✓	✓	-	-
TAY	A→Y					A6 2 1								✓	✓	-	-
TSX	S→X					BA 2 1								✓	✓	-	-
TXA	X→A					BA 2 1								✓	✓	-	-
TXS	X→S					9A 2 1								-	-	-	-
TYA	Y→A					98 2 1								✓	✓	-	-

(1) Add it to n if page boundary is crossed

(2) Add it to n if branch occurs to same page. Add 2 to n branch occurs to different page

(3) Carry not borrow

(4) If in decimal mode, z flag is invalid. Accumulator must be checked for zero result

X: Index X
Y: Index Y
A: Accumulator
M/S: Memory per stack pointer
V: Exclusive OR

+ADD
-Subtract
V: AND
M/S: Memory bit 7
M/S: Memory bit 6

V: Modified
-Not modified
M/S: Memory bit 7
M/S: Memory bit 6

Mnemonic	Açıklama	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
MOV r1,r2	Kaydediciden kaydediciye taşı	0	1	D	D	D	S	S	S
MOV M,r	Kaydediciden belleğe taşı	0	1	1	1	0	S	S	S
MOV r,M	Bellekten kaydediciye taşı	0	1	D	D	D	1	1	0
HLT	Durdur	0	1	1	1	0	1	1	0
MVI r	Kaydediciye dolaysız taşı	0	0	D	D	D	1	1	0
MVIM	Belleğe dolaysız taşı	0	0	1	1	0	1	1	0
INR r	Kaydediciyi artır	0	0	D	D	D	1	0	0
DCR r	Kaydediciyi azalt	0	0	D	D	D	1	0	1
INRM	Bellek içeriğini artır	0	0	1	1	0	1	0	0
DCRM	Bellek içeriğini azalt	0	0	1	1	0	1	0	1
ADD r	Kaydediciyi A'ya ekle	1	0	0	0	0	S	S	S
ADC r	Kaydediciyi, elde ile birlikte A'ya ekle	1	0	0	0	1	S	S	S
SUB r	Kaydediciyi A'dan çıkart	1	0	0	1	0	S	S	S
SBB r	Kaydediciyi, ödünç ile birlikte A'dan çıkart	1	0	0	1	1	S	S	S
ANA r	Kaydediciyi A ile Ve isteminden geçir	1	0	1	0	0	S	S	S
XRA r	Kaydediciyi A ile Özel VEYA'la	1	0	1	0	1	S	S	S
ORA r	Kaydediciyi A ile VEYA'la işleminden geçir	1	0	1	1	0	S	S	S
CMP r	Kaydediciyi A ile karşılaştır	1	0	1	1	1	S	S	S
ADD M	Belleği A'ya ekle	1	0	0	0	0	1	1	0
ADC M	Belleği, elde ile birlikte A'ya ekte	1	0	0	0	1	1	1	0
SUB M	Belleği A'dan çıkart	1	0	0	1	0	1	1	0
SBB M	Belleği, ödünç ile birlikte A'dan çıkart	1	0	0	1	1	1	1	0
ANA M	Belleği A ile VE'le	1	0	1	0	0	1	1	0
XRA M	Belleği A ile Özel VEYA'la	1	0	1	0	1	1	1	0
ORA M	Belleği A ile VEYA'la	1	0	1	1	0	1	1	0
CMP M	Belleği A ile karşılaştır	1	0	1	1	1	1	1	0
ADI	A'ya dolaysız ekle	1	1	0	0	0	1	1	0
ACI	A'ya, elde ile dolaysız ekle	1	1	0	0	1	1	1	0
SUI	A'dan dolaysız çıkart	1	1	0	1	0	1	1	0
SBI	A'dan ödünç ile dolaysız çıkart	1	1	0	1	1	1	1	0
AMI	A ile dolaysız VE'le	1	1	1	0	0	1	1	0
XRI	A ile dolaysız Özel VEYA'la	1	1	1	0	1	1	1	0
ORI	A ile dolaysız VEYA'la	1	1	1	1	0	1	1	0
CPI	A ile dolaysız karşılaştır	1	1	1	1	1	1	1	0
RLC	A'yi sola döndür	0	0	0	0	0	1	1	1
RRC	A'yi sağa döndür	0	0	0	0	1	1	1	1
RAL	A'yi, elde üzerinden sola döndür	0	0	0	1	0	1	1	1
RAR	A'yi elde üzerinden sağa döndür	0	0	0	1	1	1	1	1
JMP	Koşulsuz atla	1	1	0	0	0	0	1	1
JC	Elde varsa atla	1	1	0	1	1	0	1	0
JNC	Elde yoksa atla	1	1	0	1	0	0	1	0
JZ	Sıfırda atla	1	1	0	0	1	0	1	0
JNZ	Sıfır değilse atla	1	1	0	0	0	0	1	0
JP	Pozitifse atla	1	1	1	1	0	0	1	0
JM	Negatifse atla	1	1	1	1	1	0	1	0
JPE	Eşlik çiftse atla	1	1	1	0	1	0	1	0
JPO	Eşlik tekse atla	1	1	1	0	0	0	1	0
CALL	Koşulsuz çağır	1	1	0	0	1	1	0	1
CC	Elde varsa çağır	1	1	0	1	1	1	0	0
CNC	Elde yoksa çağır	1	1	0	1	0	1	0	0
CZ	Sıfırda çağır	1	1	0	0	1	1	0	0
CNZ	Sıfır değilse çağır	1	1	0	0	0	1	0	0
CP	Pozitifse çağır	1	1	1	1	0	1	0	0
CM	Negatifse çağır	1	1	1	1	1	1	0	0
CPE	Eşlik çiftse çağır	1	1	1	0	1	1	0	0
CPO	Eşlik tekse çağır	1	1	1	0	0	1	0	0
RET	Geriye don	1	1	0	0	1	0	0	1
RC	Elde varsa geriye don	1	1	0	1	1	0	0	0
CP	Pozitifse çağır	1	1	1	1	0	1	0	0

Mnemonik	Açıklama	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
CM	Negatifse çağır	1	1	1	1	1	1	0	0
CPE	Eşlik çiftse çağır	1	1	1	0	1	1	0	0
CPO	Eşlik tekse çağır	1	1	1	0	0	1	0	0
RET	Geriye don	1	1	0	0	1	0	0	1
RC	Elde varsa geriye don	1	1	0	1	1	0	0	0
RNC	Elde yoksa geriye don	1	1	0	1	0	0	0	0
R2	Sıfırda geriye don	1	1	0	0	1	0	0	0
RNZ	Sıfır değilse geriye don	1	1	0	0	0	0	0	0
RP	Pozitifse geriye don	1	1	1	1	0	0	0	0
RM	Negatifse geriye don	1	1	1	1	1	0	0	0
RPE	Eşlik çiftse geriye don	1	1	1	0	1	0	0	0
RPO	Eşlik tekse geriye don	1	1	1	0	0	0	0	0
RST	Yeniden başlat	1	1	A	A	A	1	1	1
IN	Giriş	1	1	0	1	1	0	1	1
OUT	Çıkış	1	1	0	1	0	0	1	1
LXIB	Kaydediciye dolaysız yükle BC çifti	0	0	0	0	0	0	0	1
LXID	Kaydediciye dolaysız yükle DE çifti	0	0	0	1	0	0	0	1
LXIH	Kaydediciye dolaysız yükle HL çifti	0	0	1	0	0	0	0	1
LXISP	Yığın işaretçisine dolaysız yükle	0	0	1	1	0	0	0	1
PUSHB	BC kaydedici çiftini yığına it	1	1	0	0	0	1	0	1
PUSHD	DE kaydedici çiftini yığına it	1	1	0	1	0	1	0	1
PUSHH	HL kaydedici çiftini yığına it	1	1	1	0	0	1	0	1
PUSHPSW	A'yi ve Bayrakları yığına it	1	1	1	1	0	1	0	1
POPB	BC kaydedici çiftini yığından çek	1	1	0	0	0	0	0	1
POPD	DE kaydedici çiftini yığından çek	1	1	0	1	0	0	0	1
POPH	HL kaydedici çiftini yığından çek	1	1	1	0	0	0	0	1
POP PSW	A'yi ve Bayrakları yığından çek	1	1	1	1	0	0	0	1
STA	A'yi dolaysız sakla	0	0	1	1	0	0	1	0
LDA	A'yi dolaysız yükle	0	0	1	1	1	0	1	0
XCHG	DE ve HL kaydedicilerini karşılıklı değiştir	1	1	1	0	1	0	1	1
XTHL	Yığının tepesiyle HL'i karşılıklı değiştir	1	1	1	0	0	0	1	1
SPHL	HL'in içeriğini yığın işaretçisine yükle	1	1	1	1	1	0	0	1
PCHL	HL'in içeriğini program sayacına yükle	1	1	1	0	1	0	0	1
DADB	BC'yi HL'e ekle	0	0	0	0	1	0	0	1
DADD	DE'yi HL'e ekle	0	0	0	1	1	0	0	1
DADH	HL'i HL'e ekle	0	0	1	0	1	0	0	1
DADSP	Yığın işaretçisini HL'e ekle	0	0	1	1	1	0	0	1
STAXB	A'yi dolaylı sakla	0	0	0	0	0	0	1	0
STAXD	A'yi dolaylı sakla	0	0	0	1	0	0	1	0
LDAXB	A'yi dolaylı yükle	0	0	0	0	1	0	1	0
LDAXD	A'yi dolaylı yükle	0	0	0	1	1	0	1	0
INXB	BC kaydedici içeriklerini artır	0	0	0	0	0	0	1	1
INXD	OE kaydedici içeriklerini artır	0	0	0	1	0	0	1	1
INXH	HL kaydedici içeriklerini artır	0	0	1	0	0	0	1	1
INXSP	Yığın işaretçisini artır	0	0	1	1	0	0	1	1
DCXB	BC'yi azalt	0	0	0	0	1	0	1	1
DCXD	DE'yi azalt	0	0	0	1	1	0	1	1
DCXH	HL'i azalt	0	0	1	0	1	0	1	1
DCXSP	Yığın işaretçisini azalt	0	0	1	1	1	0	1	1
CMA	A'nin tümleyenini al	0	0	1	0	1	1	1	1
STC	Elde bayrağını kur	0	0	1	1	0	1	1	1
CMC	Elde bayrağının tümleyenini al	0	0	1	1	1	1	1	1
DM	A'yi ondalık tabana ayarla	0	0	1	0	0	1	1	1
SHLD	HL'i dolaysız sakla	0	0	1	0	0	0	1	0
LHLD	HL'i dolaysız yükle	0	0	1	0	1	0	1	0
EI	Kesmeleri yetkilendir	1	1	1	1	1	0	1	1
DI	Kesmeyi yetkisizle	1	1	1	1	0	0	1	1
NOP	Hiç işlem yapma	0	0	0	0	0	0	0	0
RIM	Kesme maskesini oku	0	0	1	0	0	0	0	0
SIM	Kesme maskesini kur	0	0	1	1	0	0	0	0

Sembol	Anlamı
akümülatör	A kaydedicisi
adr	16 bitlik adres değeri
veri	8 bitlik veri değeri
veri 16	16 bitlik veri değeri
byte 2	Komutun 2. byte'ı
byte 3	Komutun 3. byte'ı
port	Bir G/Ç biriminin 8-bitlik adresi
r, r1, r2	A,B,C,D,E,H,L kaydedicilerinden biri
DDD, SSS	A,B,C,D,E,H,L kaydedicilerinden birini gösteren bit deseni (DDD= varış yeri, SSS=kaynak yeri)
	DDD veya SSS Kaydedici adı
	111 A
	000 B
	001 C
	010 D
	011 E
	100 H
	101 L
rp	B; BC çiftini B yüksek değerlikli C düşük değerlikli kaydedici olarak gösterir. D; DE çiftini D yüksek değerlikli E düşük değerlikli kaydedici olarak gösterir. H; HL çiftini H yüksek değerlikli L düşük değerlikli kaydedici olarak gösterir. SP 16 bitlik yığın göstericisi
RP	B,D,H,SP kaydedici çiftlerinden birini gösteren bit deseni.
	RP Kaydedici Çifti
	00 B,C
	01 D,E
	10 H,L
	11 SP
rh	Gösterilen kaydedici çiftinin ilk (üst sıralı) kaydedicisi
rl	Gösterilen kaydedici çiftinin ikinci (alt sıralı) kaydedicisi
PC	16 bitlik program sayıcısı (yüksek değerlikli kısmı PCH, düşük değerlikli kısmı PCL).
SP	16 bitlik yığın göstericisi (yüksek değerlikli kısmı SPH, düşük değerlikli kısmı SPL).
r _m	r kaydedicisinin m biti (bitler soldan sağa doğru 7'den 0'a kadar numaralanır).
Bayraklar (flags)	Z: Sıfır, S:İşaret, P:Eşlik, CY:Elde, AC:Yarım elde
() veya []	Parentez içeriği hafıza konumu veya kaydedici içeriği
←	'e aktar.
^	VE işlemi
∨	Özel VEYA işlemi
Λ	VEYA işlemi
+	Toplama
-	İkiye tümleyen çıkarma
*	Çarpma
↔	... ile karşılıklı deęiş.
-	Bire tümleyen
n	Yeniden başlatma sayısı
NNN	0-7 arası yeniden başlatma sayısının sırasıyla 000-111 arasında ikili gösterimi

TEKNİK TERİMLER SÖZLÜĞÜ

A

Adres yolu (address bus)	: Üzerinden adres bilgisinin taşındığı iletken teller.
Akümülatör	: Mikroişlemcilerde temel işlemler için kullanılan bir kaydedicidir.
Alfasayısal	: Alfabetik ve sayısal karakterler.
Algılayıcı (sensor)	: Bir fiziksel büyüklüğün şiddetini ölçen eleman.
Algoritma (algorithm)	: Belirli bir işi gerçekleştirmek için adım adım tanımlanan yönerge.
Altıyordam (subroutine, procedure)	: Ana program içerisinde tekrarlı işleri yapmak için çağrılan ve işlemleri bitirdikten sonra denetimi tekrar ana programa aktaran program modülleri.
Amaç program (object program)	: Bilgisayar hafızasına yüklenmeye hazır derlenmiş ya da çevrilmiş program dosyası.
Aritmetik mantık birimi (ALU; arithmetic and logical unit)	: İkilik tabandaki sayılar üzerinde bir takım matematiksel (toplama, çıkarma, çarpma,...) ve mantıksal işlemler (AND, OR,...) yapan birim.
Assembler (çevirici)	: Assembly dili ile yazılmış kaynak programı derleyip makine iline çeviren program.
Assembly	: Mikroişlemcilerin programlanmasında kullanılan sembolik komutlarla hazırlanan programlama dili.

B

Baud rate	: Bir bitin aktarılması için geçen süre. Baud rate= 1/1 bitin geçiş zamanı.
Bayrak (flag)	: İşlemcinin veya ALU'nun o anki durumu gösteren register.
BCD (binary code decimal)	: Onluk sayı sistemindeki rakamların ikilik kodlanmış şekli.
Bit (binary digit)	: Sayı sisteminin ikili tabanındaki rakamlarından her biri.
Blok transfer	: Bir grup verinin bir küme olarak transfer edilmesi.
Bps (bits per second)	: Bir saniyede iletilen bit sayısı. Bps=1/bit sayısı.
Byte	: 8 bitle ifade edilen bilgi kümesi.

Ç

Çapraz çevirici (cross assembler)	: Belirli bir bilgisayar için hazırlanmış bir programı, başka bir sistemde kullanılabilir hâle dönüştüren çevirici program.
Çevre birimi (peripheral)	: Mikrobilgisayarlara sonradan bağlanan harici cihazlar veya elemanlar (yazıcı, monitör vb.).
Çoklu işlemci (multiprocessor)	: Aynı yolu paylaşan birden fazla mikro işlemciyi içeren sistem.

D

Derleyici (Compilers)	: Yüksek seviyeli dillerle yazılmış kaynak programı makine kodlarına dönüştüren program.
Donanım (hardware)	: Bilgisayarı oluşturan Fiziksel parçaların tümü.
Düzenleyici (editör)	: Bilgisayarlarda program kodları yazabilmemizi sağlayan metin düzenleme programları.

E

El sıkışma (hand shaking)	: Karşılıklı iki sistemin arasında veri transferinin yapıldığını doğrulamak için yapılan haberleşme.
Emülasyon (emulation)	: Bir bilgisayar sisteminin, başka bir bilgisayar sistemi için hazırlanmış veri ve programları kullanabilmesine ve aynı sonuçları üretebilmesine imkân sağlayan yöntem.
Erişim süresi (access time)	: Herhangi bir birimden gerekli bir bilginin istenmesi ve gelmesi arasında geçen süre.
Eşlik biti (parity bit)	: Seri iletilen verilerin doğru gelip gelmediğinin kontrolünü yapmak için verinin sonuna eklenen sağlama biti. Tek eşlik ve çift eşlik yöntemleri kullanılmaktadır (bk. seri iletişim).

F

FIFO (first in first out)	: Bilgisayar hafızasının belirli bir bölümünü depo (yığın) olarak kullanan yöntemde yığına giren ilk verinin ilk çıkması.
Firmware	: Kalıcı olarak ROM tipi hafızalarda saklı bulunan ve bilgisayarın ilk çalışmasını sağlayan, sistemin temel işlevlerini karşılayan yazılım.

G

Gerçek zaman (real time)	: İlgili fiziksel olayın çıktığı anda işlem yapabilen bilgisayar sistemi ya da program.
--------------------------	---

H

Hafıza (memory)	: Bilgisayarlardaki sayısal verileri geçici veya kalıcı olarak saklamakta kullanılan birimler.
Hafıza haritası (memory map)	: Sistemdeki erişilebilir tüm hafızaların ve giriş/çıkış elemanlarının tümünün adreslerinin bir tablo üzerindeki yerlerinin belirtildiği tablo veya diyagram.
Hafıza hücresi (memory cell)	: İkilik tabandaki sayıların tutulabildiği hafıza birimlerinde her bir byte'lık bilginin tutulduğu yer, hücre.
Hata ayıklama (debug)	: Bir programdaki hataları tespit etmek. Hataları tespit etmek ve düzeltmek için kullanılan programlara da genel olarak "debugger" (okunuşu: dibâg'r) denir.
Hibrit	: Birden fazla monolitik ve diğer elektronik elemanların (direnç, transistor...) bir arada kullanarak bir paket hâline getirilmesi.

İ

İşlem kodu (op-code)	: Mikro işlemcinin belirli bir işlemi yapmasını sağlayan bir veya birkaç byte'lık makine kodu.
İşlenen (operand)	: Bir işlem kodunun üzerinde işlem yapacağı veri veya yer.

K

Kalibre (calibrate)	: Referans değerine göre yapılan ayarlama.
Kompozit video sinyali (composite video signal)	: Video, silme, senkronizasyon, referans ve renk işaretlerini kapsayan toplam televizyon işareti.
Kapı (port)	: Giriş veya çıkış işlemlerinde kullanılan ara bağlantı aracı.
Katot ışınlu tüp (CRT; cathode ray tube)	: Katottan anoda doğru akan ışık huzmesinin kullanıldığı kapalı tüp, ekran.
Kaydedici (register)	: Mikro işlemci içerisinde, üzerinde işlem yapılacak verinin geçici olarak saklandığı adres veya veri hücreleri.
Kaynak program (source program)	: Bilgisayarların programlanmasını sağlayabilmek için hazırlanan kodların tümü. Bu kodlar değişik programlama dillerinde (Assembly, C, Basic) hazırlanabilir. Kaynak program hazırlandıktan sonra derlenmesi, yorumlanması veya çevrilmesi gereklidir. Bu işlemler, kaynak programı mikro işlemcinin çalıştırabileceği makine kodlarına dönüştürür.
Kesme (interrupt)	: Program akışının değişik sebeplerle kesintiye uğraması. Kesmeler yazılı ve donanım kesmeleri diye ikiye ayrılır. Kesintiye uğrayan ana program, çalışmayı geçici olarak kesme hizmet programlarında yürüterek ana programda kaldığı yerden devam eder.
Komut (instruction)	: Mikro işlemcilerin tanımlayıp uyguladığı sözcük.
Komut saykılı (instruction cycle)	: Bir komutun işlenmesi için gerekli temel makine periyodu sayısı.
Komut takımı (instruction set)	: Bir mikro işlemcinin tanımlayıp uygulayabildiği komutların bulunduğu tablo. Bu tablo, mikro işlemci üretildikten sonra kullanıcılara sunulur.
Kontrol yolu (control busses)	: Kontrol sinyallerini taşıyan ortak elektriksel hatlardan oluşan yol.

L

LIFO (last in first out)	: Bilgisayar hafızasının belirli bir bölümünü depo (yığın) olarak kullanan yöntemde yığına giren son verinin ilk çıkması.
--------------------------	---

M

Makro çevirici (makro-assembler)	: Standart assemblerin özel bir çeşidi. Makro assembler, çok kullanılan komut gruplarının makro olarak tanımlanmasına imkân sağlar.
Maskelenemez kesme (NMI; non-maskable interrupt)	: Bir programın yürütülmesini kesebilen ve diğer kesmeleri devre dışı bırakabilen, öncelik seviyesi yüksek engellenemeyen kesme türü.
Mikrodenetleyici (microcontroller)	: Mikro işlemci çekirdeği, hafıza birimleri (RAM, EPROM...) ve giriş/çıkış birimlerini içeren tümleşik eleman.
Mikrobilgisayar (microcomputer)	: Yapısında mikro işlemci, hafızalar, giriş ve çıkış birimleri ile klavye/display modülü içeren temel bilgisayar.

Mikroişlemci (MIB; Merkezi İşlem Birimi) Microprocessor (CPU; Central Processing Unit) Mnemonic	: Bilgisayarlar için hazırlanan program komutlarını işleterek aritmetiksel, mantıksal ve diğer işlemleri yapan merkezi ünite.
Modem (MODulator-DEMulator)	: Sayısal verilerin uzak yerlere iletilmesi için kullanılan iletişim yöntemi. Bu yöntemde iletilen veri ilk önce modüle edilir. Modüle edilmiş veri bozulmadan karşı tarafa iletdikten sonra alıcı taraf modüleli sinyali demodüle ederek veriyi eski hâline dönüştürür.
Monolitik	: Bir mikro devrenin, tek silikon yonga üzerine yerleştirilmiş hâline verilen addır.
O	
Offset	: Bir segment içerisindeki veri hücrelerinin adresi.
P	
Paralel iletişim	: Verilerin birden fazla hat üzerinde aynı anda iletildiği iletişim şekli.
Program sayıcı (program counter)	: Otomatik olarak bir sonraki işletilecek komutun adresini tutan kaydedici.
Programlanabilen mantıksal denetleyici (PLC; Programable Logical Control)	: Sensörden aldığı bilgiyi kendine verilen programa göre işleyen ve iş elemanlarına ulaştıran mikrobilgisayar.
R	
Reset	: Sistemin yeniden başlatılması.
S	
Segment	: Hafızanın 64 Kb parçalara ayrılmış bölümlerden her biri.
Seri iletişim	: Verilerin tek hat üzerinden art arda iletildiği iletişim şekli.
Servo	: Çıkışı, o andaki çıkış değeri ile belirli bir referans değerine göre ayarlanabilen, otomatik kontrol sistemleri için kullanılan genel bir terim.
Simülasyon (simulation)	: Fiziksel sistemlerin tüm işlev ve özelliklerinin bilgisayar ortamında modellenmesi ile bu model üzerinde yapılan çalışmalar.
T	
Tampon (buffer)	: a) Bilgisayar hafızasında geçici olarak ayrılan bölge. b) Devreler arasındaki sinyal akışını düzenleyen ara devre.
Tek yonga (monolitik)	: Tek bir silikon yonga üzerinde hazırlanmış mikro devrelerdir.
Transducer	: Bir fiziksel büyüklüğü diğer bir fiziksel büyüklüğü dönüştüren eleman.
V	
Veri yolu (ata buses)	: Verilerin elektriksel sinyal olarak taşındığı ortak yol. Genellikle iki yönlü iletim için kullanılır.
Y	
Yalnızca okunabilir hafıza (ROM; read only memory)	: İçerisindeki bilgilerin enerjisi kesilse de silinmediği hafıza elemanları. Örneğin PROM, EPROM, EEPROM.
Yazmaç	: Bk. Kaydedici.
Yazılım (software)	: Bilgisayarın bir amaca yönelik işleyişini sağlayan tüm programlardır.
Yığın (stack)	: Verilerin geçici bir süre hafızalarda tutulması. Bu mekanizma belirli bir sıra ile veri giriş-çıkış işlerini düzenler.
Yığın göstericisi (stack pointer)	: Verileri, hafızanın belirlenen bir bölgesinde sıralı olarak saklanmasını sağlayan yığın yönteminde en üstteki verinin adresini tutan kaydedici.
Yüksek seviyeli diller (High Level Language)	: Bilgisayar programları hazırlamak için geliştirilen, anlaşılması ve kullanılması kolay programlama dilleri.
Z	
Zamanlayıcı (Timer)	: Belirli zaman aralıklarında sinyal üretmesi için programlanabilen entegre.

DİZİN

2

2764 71, 79, 277, 279

8

8085 23, 44, 77, 82, 97, 139, 142, 152, 156, 212

A

Adres yolu 24, 41, 71, 79, 83, 95, 166, 182
Adresleme modu 24, 138, 139, 151
Akış diyagramı 194, 196, 213, 220, 227, 235
Alt programlar 127, 136, 275
Aritmetik mantık birimi 46
ASCII 13, 14, 20, 63, 194, 257, 260
Assembler 193, 201, 207
Assembly programlama 20, 47, 113, 138, 191, 198, 207

B

Baud 77, 281
Bayrak 33
BCD 35, 116, 224, 241
Blok transfer 107, 108
Buffer 41

C

CD-ROM 81
Cross Assembler 200
CRT 258, 264

Ç

Çevirici 193, 201, 245, 249
Çevre birimi 75

D

Dallanma komutu 118
Debug 204
Decoding 50
Dinamik hafızalar 65, 73
Disket 61
Donanım 134, 204
Durum kodu kaydedici 32, 39, 110, 155, 211

E

EPROM 21, 44, 64, 71, 79, 162, 168, 183, 257, 265, 270, 277, 297
Eşlik biti 69
Executing 51

F

Fetching 49
First In First Out 131

G

Genel amaçlı kaydediciler..... 35, 71, 85
Giriş/çıkış..... 33, 56, 73, 92, 126, 136, 162, 171, 180, 240, 272, 302

H

Hafıza..... 23, 44, 59, 63, 73, 87, 91, 101, 127, 133, 147, 162, 170, 186, 192, 209, 227, 235, 240, 273
Hafıza haritası..... 163, 189
Hafıza hücresi..... 44
Hafızalar..... 59, 63, 81
Hata ayıklama..... 204
Hibrit..... 244

I

Instruction..... 85
Interrupt..... 33, 36, 92, 125, 156, 275, 276
IRQ..... 71, 156, 172

İ

İndeks kaydedici..... 31, 102, 105, 116, 140, 198, 200, 217, 221, 229
İşlem kodu..... 47
İşlenen..... 16, 23, 47, 48, 139

K

Kapı..... 10, 11
Kaydedici..... 44, 110, 139, 142, 152
Kaynak program..... 199
Kesme..... 24, 33, 71, 82, 92, 134, 153, 160, 177, 266, 274, 276
Komut..... 16, 23, 37, 40, 50, 71, 82, 99, 109, 111, 122, 130, 144, 160, 200, 206, 299
Kontrol yolu..... 45

L

Ladder..... 297, 298, 300
Last In First Out..... 105, 131

M

Maskelenemez kesme..... 73, 74, 82, 96, 156, 157
Memory..... 36, 63, 64
MIB..... 21, 44, 50, 77, 162
Mikrobilgisayar..... 10, 21, 59, 66, 70, 164, 207, 258, 260, 275, 296
Mikroişlemci..... 16, 21, 30, 40, 50, 70, 86, 96, 118, 121, 133, 142, 151, 161, 173, 191, 200, 231
Mod..... 280, 281
Monolitik..... 244
Motorola 6802..... 11, 23, 28, 52
Motorola 6821..... 71

N

Nibble..... 10, 20
NMI..... 71, 73, 86, 93, 96, 158, 159

O

Operand..... 93

Ö

Özel amaçlı kaydediciler..... 30, 37, 125

P

Paralel iletişim..... 13, 66, 81
PLC..... 295, 300, 305
Port..... 76, 127, 172, 268

R

RAM hafıza..... 5, 64, 71, 131, 162, 170, 264, 267
Restart..... 160
Robotlar..... 282, 287, 291, 294

S

Seçme ucu..... 76
Segment kaydedici..... 37, 87
Seri iletişim..... 13, 66, 70, 81, 177, 280
Statik hafızalar..... 63, 65, 81

V

Veri..... 11, 20, 41, 50, 69, 71, 80, 91, 139, 175, 180, 219, 224, 266, 270, 281
Veri yolu..... 14, 23, 24, 45, 46, 50, 55, 72, 74, 75, 77, 82, 95

W

Word..... 11, 20

Y

Yığın..... 31, 32, 37, 85, 86, 130, 203
Yığın kaydedici..... 31, 85, 132, 133

Z

Z-80..... 113, 205
Zamanlama..... 53, 269

KAYNAKÇA

1. Adalı, Eşref, Mikroişlemciler Mikrobilgisayarlar, İstanbul, 1991.
2. Albert Ali Salah, Robot Sunumları, Algılayıcılar, 2000.
3. Clements, Alan, Bilgisayar Donanımının Temelleri, Ankara, 1994.
4. Cooke, Phil, Mikroelektronik Sistemler Kitap I, Ankara, 1994.
5. Doç.Dr. Akın,Levent, Autonomous Robots Dersi Algılayıcılar Sunumu, Boğaziçi Üniversitesi
6. Dr. Gümtüşkaya, Haluk, Mikroişlemciler Ve 8051 Ailesi, İstanbul, 1999.
7. Dr. Gümtüşkaya, Haluk, Mikroişlemciler Ve Bilgisayarlar, İstanbul, 1999.
8. Hall, V. Douglas, Mikroişlemciler Ve Sayısal Sistemler, Eskişehir, 1994.
9. Halsall, Fred, Mikroelektronik Sistemler Kitap II, Ankara, 1994.
10. Heathkit Educational Systems, Robotics And Industrial Electronics, USA,1985.
11. Leatham-Jones, Barry, Bilgisayarlı Nümerik Kontrol Konusuna Giriş, Eskişehir, 1994.
12. Martin, G., Heap, N., Mikroişlemci İlkeleri Kitap IV, Ankara, 1994.
13. Meadows, R., Parson, A.J., Mikroişlemciler Esaslar, Elemanlar ve Sistemler, Eskişehir, 1994.
14. Morse, M.J., Mikroişlemci-Tabanlı Sistemler Kitap V, Ankara, 1994.
15. Özkan, Turhan,Mikroişlemciler, Mikrobilgisayarlar ve Assembly Programlama II, İstanbul, 1994.
16. Potton, A., Mikroişlemci-Tabanlı Sistemlerkitap IV, Ankara,1994.
17. Ronald C. Arkin, Behavior-Based Robotics, 1998.
18. Siegel,Mel, Algılar ve Algılayıcılar Ders Notları, Carnegie Mellon Üniversitesi Robot Enstitüsü
19. Simmons, Peter, İleri Mikroelektronik Uygulamaları, Ankara, 1994.
20. Topaloğlu, Nurettin, Mikroişlemci Mimarisi ve Assembly Dili, Ankara, 2001
21. Topaloğlu, Nurettin, Mikroişlemciler ve Assembly Dili, Ankara, 1999.
22. Uçar, Fehmi,Yaşar, Hasan, µp 6802, İzmir, 1992
23. Prof. Dr. Varol, Asaf,Robotik, İstanbul, 2002
24. Woollons, David, Mikroelektronik Sistemler Kitap III, Ankara, 1994.

İnternet kaynakları :

-
- | | |
|--|------------------------------------|
| 1. http://hlakin.cmpe.boun.edu.tr/593/robopl/ | Bogaziçi Üniversitesi Robot sitesi |
| 2. http://eniacc.sclcu.edu.tr/webders/robotik/robotik.htm | Selçuk Üniversitesi Robot sitesi |