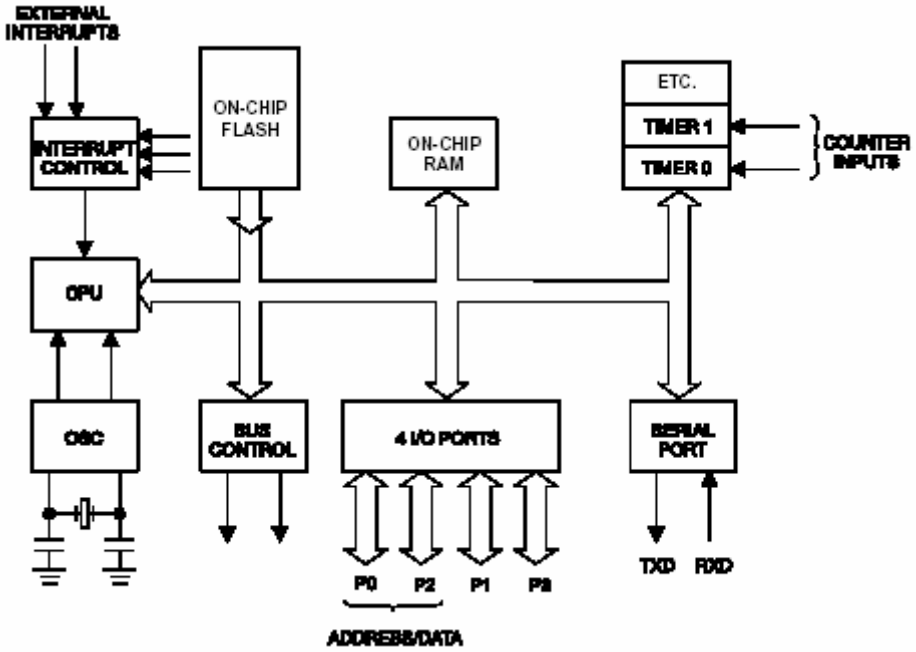
3. 8051 MİMARİSİ

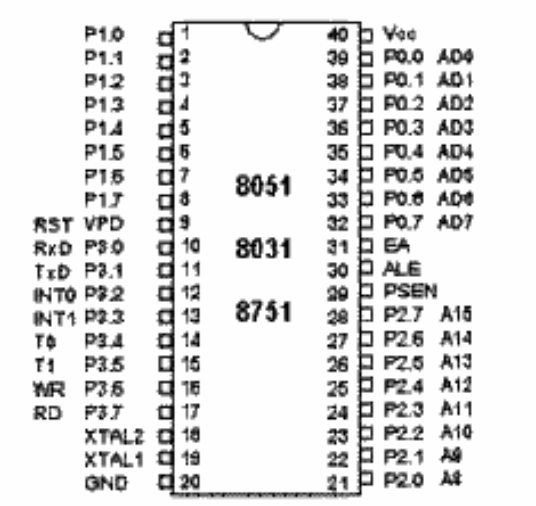
Temel mimari yapısı Şekil 3.1' de görülmekte olan 8051 mikrodenetleyici ailesinin başlıca özellikleri aşağıda verilmiştir.

- Kontrol uygulamaları için optimize edilmiş 8 bitlik CPU
- Genişletilmiş Boolean işleme komutları (tek bitlik lojik komutlar)
- On-chip program hafızası (Program ROM)
- On-chip veri hafızası (Data RAM)
- 4 adet 8 bitlik I/O portu
- Çift yönlü kullanılabilen ve tek tek adreslenebilen I/O pinleri
- 2 kanal 16 bitlik Timer/Counter (8052 de 3 kanal)
- Full Duplex UART (Seri haberleşme kanalı)
- Çok kaynaklı / vektörlü / öncelik seviyeli kesme yapısı (Interrupts)
- On-chip saat osilatörü



Şekil 3.1

3.1. 8051' İN 40 PİNLİ PİN KONFİGÜRASYONU



BACAK NO	SEMBOL	AÇIKLAMA
1,2,3,4,5,6,7,8	Port-1	Birden sekize kadar genel amaçlı giriş/çıkış portu pinleri paralel giriş veya çıkış olarak kullanılan port 1'e ait pinlerdir.
9	RST	Sıfırlama(Reset) pini Mikrodenetleyici resetlendiği anda program belleğinde kayıtlı olan sistem programları hafızasında çalışmaya başlar.
10,11,12,13,14,15,16,17	Port-3	Paralel giriş veya çıkış Port-3 'e ait port pinleridir. P3 'ün her pininin 2. bir fonksiyonu vardır.
10	P3.0 (RxD')	Asenkron seri haberleşmede seri bilginin alındığı port pinidir.
11	P3.1 (TxD')	Seri haberleşme esnasında bilginin gönderildiği port pinidir.
12	P3.2 (INT0)	0 nolu harici kesme girişi düşen kenar tetikleme ile aktif olur.
13	P3.3 (INT1)	Bir nolu harici kesme girişi düşen kenar tetikleme ile aktif olur.
14	P3.4 (T0)	Zamanlayıcı/Sayıcı - 0 Modunda çalışırken giriş olarak kulanılan bacaktır.
15	P3.5 (T1)	Zamanlayıcı/Sayıcı - 1 Modunda çalışırken giriş olarak kulanılan bacaktır.
16	P3.6 (WR')	Dış veri hafızasına veri yazılırken aktif olan kontrol işareti (WRITE) bu pinden üretilir.
17	P3.7 (RD')	Dış veri hafızasından veri okunurken aktif olan kontrol işareti (READ) bu pinden üretilir.
18	XTAL2	Clock Sinyal Giriş Ucu-2
19	XTAL1	Clock Sinyal Giriş Ucu-1
20	VSS	Mikrodenetleyicinin Referans Bacağı (Negatif besleme-Toprak) 0V uygulanır.
21,22,23,24,25,26,27,28	Port-2	Paralel giriş veya çıkış Port-2'nin uçları. Aynı zamanda dış çevre birrimleri ile iletişimde adres yolunun yüksek anlamlı (most significant) 8-bitlik bilgisini taşır.
29	PSEN'	Dış program hafızasından bilgi okunurken,dış hafıza elemanı bu kontrol sinyali ile adres bilgisi geldikten sonra bilgiyi veri yoluna çıkarır.
30	ALE	Mikrodenetleyicinin dış dünya ile haberleşmesinde Port-0 hem veri, hem adres yolunun düşük anlamlı (least significant) sekiz biti için kullanılır.Bu iki ayrı işaret gurubu ALE sinyali ile birbirinden ayrılır.

31	EA'	Mikrodenetleyici içindeki ROM'un Program belleği olarak kullanılıp kullanılmayacağını belirler.Bu giriş ucu lojik 1 olursa program reset sonrası çalışması dahili ROM ‘dan aksi takdirde harici ROM ‘ dan gerçekleştirilir.
32,33,34,35,36,37,38,39	Port-0	Paralel giriş çıkış portu.Bu port dış elemanlar kullanılmadığı zaman normal bir port görevi yapar.Dış hafıza birimleri kullanıldığı zaman ise veri yolu ile adres yolunun düşük anlamlı sekiz biti için kullanılır.Bu kullanım zamnalama ile olur,bunuda ALE sinyali sağlar.
40	VCC	Mikrodeneleyicinin pozitif beslemesi – Standart yapıda 5V uygulanır.

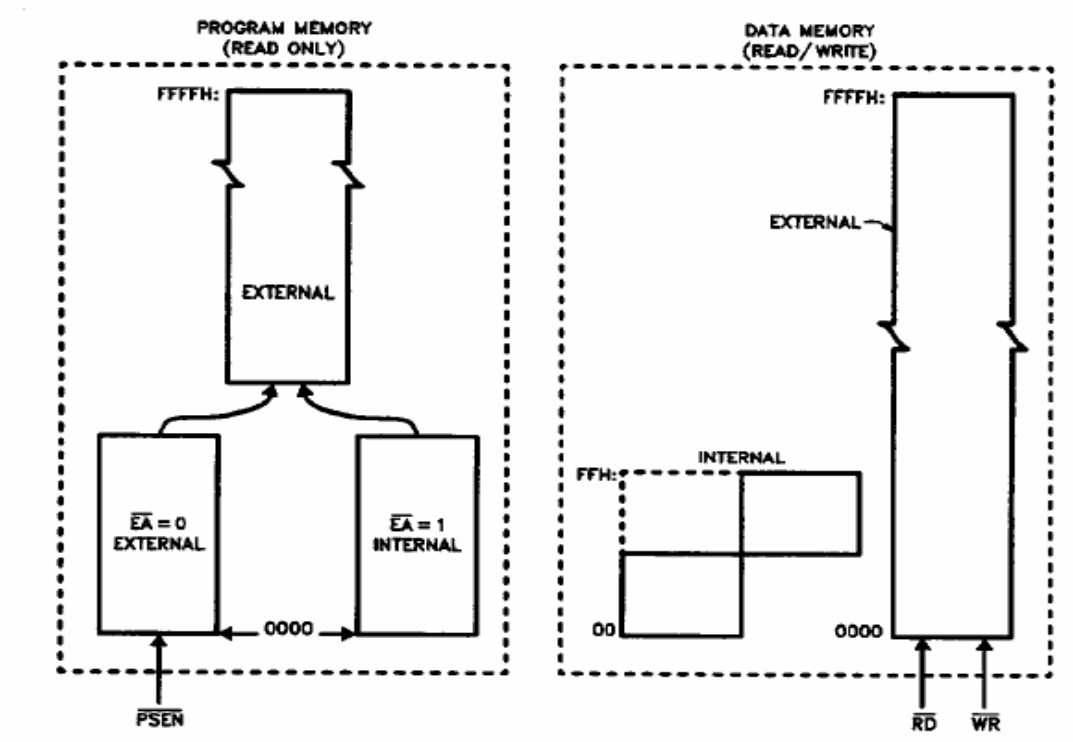
3.2. HAFIZA ORGANİZASYONU

8051 mikrodenetleyicisinin hafıza alanı Şekil 3.2’ de görüldüğü gibi, program ve veri hafızası olmak üzere iki ayrı kısımdan oluşur. 0000H – FFFFH arasında tanımlanmış olan program ve veri hafızalarına PSEN VE (RD’ & WR’) kontrol işaretleri ile 16 bit veri belleği adresleri DPTR registeri aracılığıyla aktarılır.

Program belleği (ROM,EPROM) yalnızca okunabilirdir. 64 Kbyte’a kadar program belleği olabilir. 8051, dahili 4 Kbyte ROM belleğe sahiptir. ROMsuz (ROMless) versiyonlarda tüm program belleği haricidir. PSEN’ ile harici program belleğinde belirtilen adresteki bilginin veri yoluna çıkmasına izin verilir. Harici program belleği ile 4 Kbyte-64 Kbyte’a kadar arttırılır.

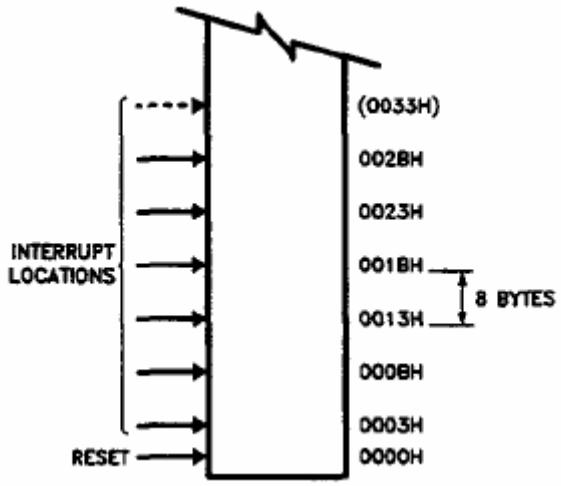
Veri belleği (RAM) program hafızasından ayrı olarak bir yer işgal eder. 8051 dahili 128 Byte RAM hafızasına sahiptir. Bu rakam harici veri belleği ile 64 Kbyte’a kadar artırılabilir. ROMsuz (ROMless) versiyonlarda dahili 128 Byte’tır. CPU, harici veri hafızasından bilgi okumak için RD’ pinini, harici veri hafızasına bilgi yazmak için ise WR’ pinini kullanır.

Harici program ve veri belleği istenirse birleştirilebilir. Bu işlem RD’ ve PSEN’ sinyalleri AND lojik kapısı girişlerine verilerek yapılır. AND kapısının çıkışı harici program ve veri belleği okunmasında kullanılır.



Şekil 3.2 Hafıza Organizasyonu

3.2.1. PROGRAM HAFIZASI



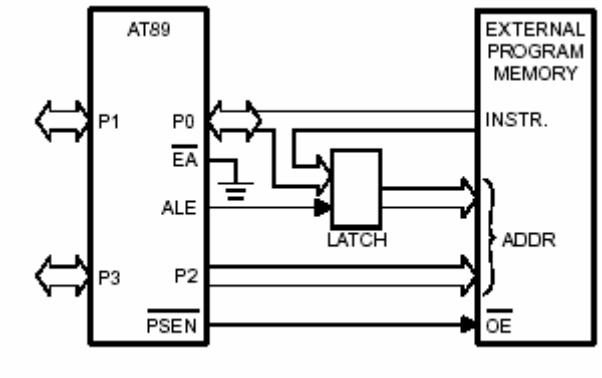
Şekil 3.3 Program Hafızası

Şekil 3.3'de program hafızasının düşük değerlikli kısmının haritası gösterilmektedir. Reset işleminden sonra CPU, 0000H adresinden komut yürütme işlemine başlar. Şekilde görüldüğü gibi her kesme program belleğindeki uygun bir yere atanır. Bu kesmeler, CPU'nun servis rutininin başladığı yere atlamasına sebep olur. Örneğin, harici kesme-0 (EXT-0) 0003H adresine atansın, eğer kullanılacaksa servis rutini, 0003H adresinden başlamak zorundadır. Eğer kesme kullanılmayacaksa kesmenin servis rutini program belleğinin genel amaçları için uygundur. Kesme servis adresleri 8 bytelık aralıkta yer kaplarlar. Harici kesme-0 için 0003H, Timer-0 için 000BH, Harici Kesme-1 (EXT-1) için 0013H ve Timer-1 için 001BH gibi. Eğer bir kesme servis rutini yeterince kısa ise (Kontrol uygulamalarında sık rastlanır) 8 byte aralığın tamamı kaplanabilir. Daha uzun servis rutinlerinde eğer diğer kesmeler kullanımda ise bir sonraki kesme adresine sıçramak için atlama komutu kullanılabilir.

Program belleğinin en az 4 Kbyte'lık kısmı dahili yada harici Rom içinde bulunabilir. Bu seçim yonganın 31 nolu pini olan EA' değerine bağlıdır. Eğer EA' lojik "0" ise program hafızası tamamen harici hafıza olarak tanımlanmış olur. EA' değerini lojik "1" yaparsak program hafızası iç bölgedeki 4Kbyte'lık ROM'dan itibaren başlar. ROMsuz (ROMless) elemanlarda (8031,80C31 gibi) harici program belleğini aktif etmek için bu pin lojik '0' ile (V_{SS}) dışarıdan bağlanmalıdır.

PSEN' harici ROM'un okunmasındaki uygulamalarda kullanılır. PSEN' dahili program uygulamalarında aktif edilemez.

Harici Program uygulaması için donanım konfigürasyonu Şekil 3.4' te verilmiştir.



Şekil 3.4 Harici Program Belleğinin Kullanımı

Unutulmaması gereken şey; 16 I/0 hattı (Port0-2) harici program belleği uygulamalarında BUS fonksiyonları olarak kullanılır. Port0 adres ve veri yolu (Address/Data BUS) gibi işlemlerde hizmet verir. Port2'yi otomatik olarak adres bilgisinin üst 8 biti (MSB) ile yüklenir. Program çevriminin ilk yarısında Port0 adres bilgisinin alt 8 biti (LSB) ile yüklenir ve ALE (Address Latch Enable) sinyalinin sıfıra düşmeşi ile tutucu (latch) tarafından tutularak adres yolunda 16 bitlik adres bilgisi elde edilmiş olur. Adres bilgisinin

elde edilmesi ile CS sinyali oluşur. PSEN' sinyalinin sıfıra düşmesinde belirli bir süre sonra veri yoluna şeçilmiş olan bilgi çıkmış olur. Çıkan bu bilgi bu komutun OP-CODE bilgisidir. Program hafıza adresleri her zaman 16 bit genişliğindedir. Buna rağmen program belleğin kullanılan gerçek miktarı 64 KByte'dan daha az olabilir. Harici Program uygulamalarında 8 bit portların ikisi olan P0 ve P2 program belleğinin adreslerinde kullanılır.

Mikrodenetleyici sisteminde program belleğinin kullanılmasında temel unsurları özetlersek;

EA: Program belleğinin nereden başlayacağını seçen bacaktır. Eğer Logic 1 olursa program belleği iç ROM'dan başlayacak ve 64kByte'lık harici hafızadan devam edecektir.

PSEN: Bu sinyal mikrodenetleyicinin 29 nolu bacağından çıkar ve harici hafıza elemanı ile belirtilen adresteki bilginin veri yoluna çıkmasına izin verir.

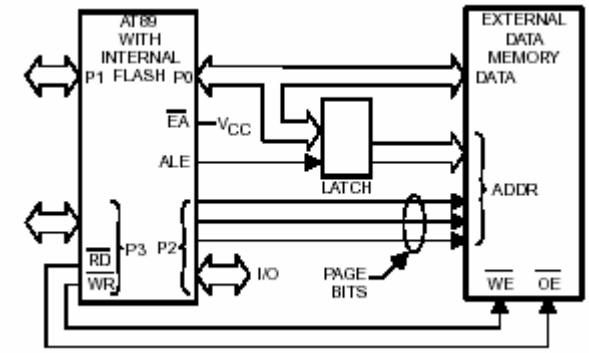
Veri Yolu: Bilginin iletildiği yoldur.
Adres Yolu: Bilginin hafızadaki yerini gösteren veridir.

CS: Birden fazla hafıza elemanı olduğu taktirde bilginin bulunduğu hafıza elemanını aktif yapar ve adres bilgisinin kodlanmasıyla elde edilir.

ALE: Mikrodenetleyici sistemi çevre birimlerle haberleşirken Port0 hem veri yolu için hem de adres bilgisinin alt 8 biti (LSB) için kullanılır. ALE sinyali bu işlem için gerekli olan zamanlamayı yapar. ALE sinyali ile Port0 üzerinden gönderilen veri bilgisi latchlenir, bu sayede Port0 zaman paylaşımı olarak hem adres bilgisinin alt 8 biti, hem de 8 bit veri için kullanılır.

3.2.2. VERİ HAFIZASI

Şekil 3.2' nin sağ tarafı dahili ve harici veri belleği alanlarını göstermektedir.



Şekil 3.5 Harici Veri Belleğine Ulaşım

Şekil 3.5 ise harici RAM'in 2KByte'a kadar olan erişimi için donanım konfigirasyonunu göstermektedir. Port0 çoklu adres/veri yolu (Address/data BUS) şeklinde RAM'e hizmet verir. Ayrıca Port2'nini üç hattı RAM'e ulaşmak için kullanılır. CPU, RD' ve WR' sinyallerini harici RAM' e geçişleri boyunca üretir. Harici Veri Belleği 64Kbyte'a kadar arttırılabilir. Harici veri belleği adresleri bir ya da iki byte genişliğinde olabilir. Bir bytelık adresler sıklıkla RAM' e ulaşmak için bir ya da daha fazla I/0 hattıyla beraber kullanılır (Şekil 3.5). İki bytelık adresler de Port2'den çıkarılan yüksek adresli byteları kullanabilirler.

Harici veri hafızası, program ve iç veri belleğinden tümüyle ayrı bir bölgedir ve kendine özgü komutları bulunmaktadır. Program belleği gibi Port0 ve Port2 üzerinden 64KByte' a kadar genişletilebilir.Yalnızca Port0 kullanılarak 256 byte'lık belleğe ulaşılabilir.

Harici veri hafızasına yalnız 2 komut ile erişilebilir.

```
MOVX  A,@DPTR
```

veya

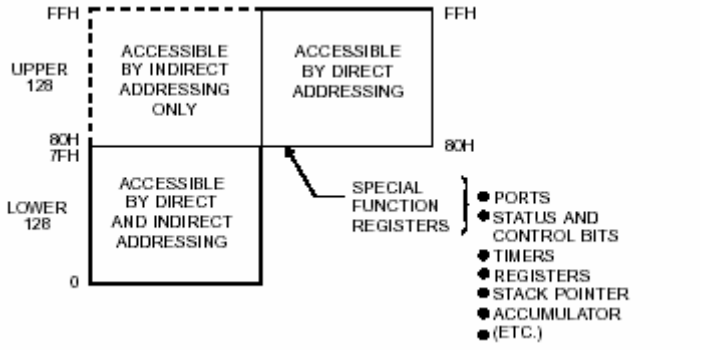
```
MOVX   @DPTR,A
```

Bu işlem iki aşamada yapılır:

a-) Port2 otomatik olarak DPTR'nin üst sekiz biti (DPH) ile yüklenir

b-) Port0 harici hafızaya erişim çevriminin ilk bölümünde DPTR' nin alt sekiz biti (DPL) dışarı çıkarır. ALE sinyalinin düşen kenarı ile DPL, tutucunun (latch) çıkışında tutulmuş olur. Bundan sonra harici belleğe ulaşıncaya kadar ALE lojik bir olmayacaktır. Bu andan sonra okuma/yazma işlemi yürütülür. Kullanılan komut `MOVX A,@DPTR` ise harici hafızadan okuma yapılacağı için RD' çıkışı 0 olur. Böylece Port0 vasıtasıyla DPTR' nin gösterdiği bilgi A' ya aktarılır. Eğer `MOVX @DPTR,A` komutu kullanılırsa harici belleğe yazma olayı yapılacağından WR' çıkışı lojik '0' olacaktır.

Dahili veri belleği şekil 3.6'da görüldüğü gibidir.

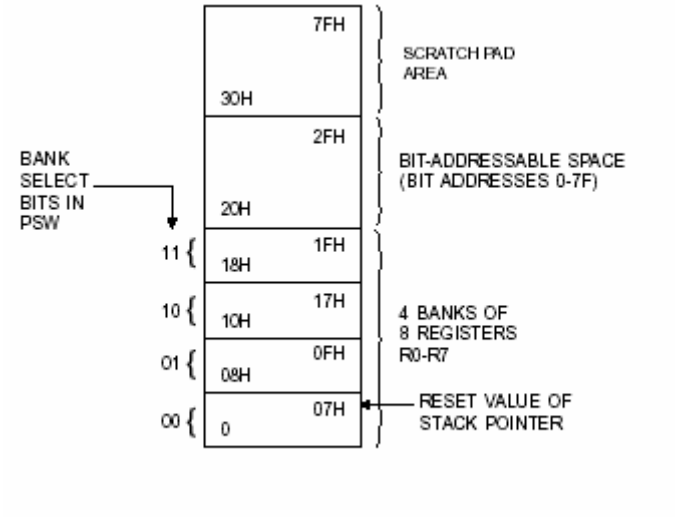


Şekil 3.6 Dahili Veri Belleği

Hafıza Alanı üç bloğa bölünmüştür. Bunlar düşük 128, yüksek 128 ve SFR (Special Function Register) alanlarında oluşur.

Dahili veri belleği adresleri her zaman bir byte genişliğindedir ve sadece 256 byte'lık adres alanını gösterir. Bununla beraber, dahili veri hafızasına yönelik adresleme modundaki basit bir yöntem ile 384 byte veri adreslenebilir. Veri hafızaya erişimlerde 7FH'den yüksek doğrudan (direct) adresler, bir hafıza alanına erişip 7FH'den yukarı dolaylı (indirect) adresler farklı hafıza alanına erişir. Şekil 3.5' te görülen yüksek 128 ve SFR alanları 80H ile FFH arasındaki adres bloklarında yer almasına karşı fiziksel olarak iki ayrı bölgedir.

RAM'ın düşük 128 byte'ı Şekil 3.7'de gösterilmiştir.



Şekil 3.7 Dahil RAM'ın Düşük 128 byte'ı

En düşük 32 byte'lık alanda tanımlanan 4 bank' dan her birinde 8 adet genel amaçlı register bulunmaktadır. Program komutlarıyla R0' dan R7' ye kadar olan registerlar çağrılır. Program Status Word (PSW) daki iki bit (RS0 ve RS1) hangi bankı kullanılacağını seçer. Register komutları direk adreslemede kullanılan komutlardan daha kısa olduğundan dolayı bu olay daha verimli kod alanı kullanımına izin vermiştir.

Register banklarının üzerindeki 16 byte bit adreslenebilir alanı (Bit Addressable Space) bloğu şeklinde biçimlendirilir. 8051 komut düzeni, bit-bit işlem yapabilen özel bit komutlarıyla büyük kullanım kolaylığı sağlar. Bu alandaki 128 bıt bu komutlar ile direk adreslenebilir. Bu alandaki bit adresleri 00H' den 07H' e kadardır.

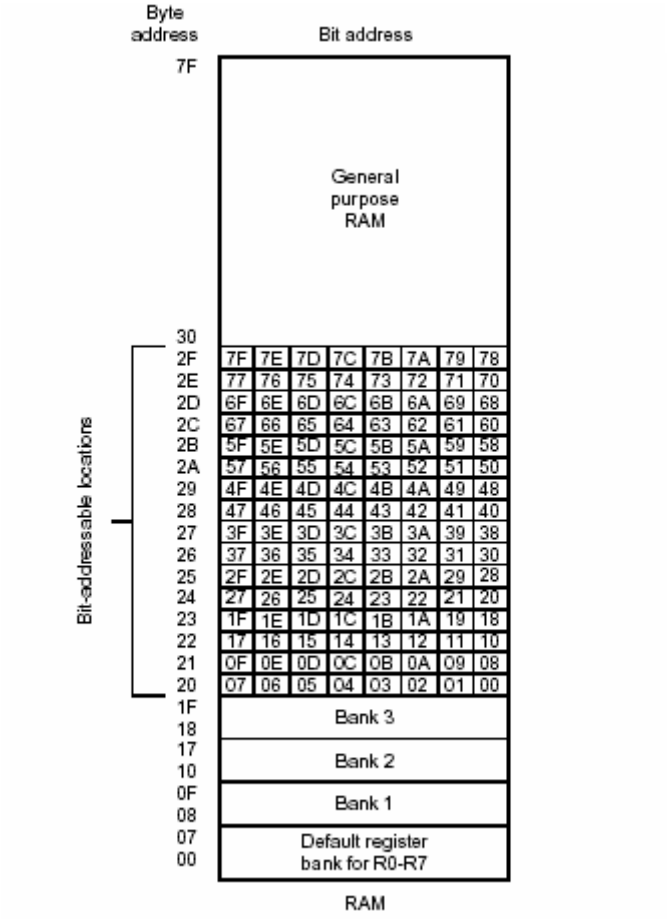
Düşük 128' de bulunan bütün bytelara doğrudan ya da dolaylı adresleme ile erişilebilir. Yüksek 128' e (Şekil 3.8) ise sadece dolaylı adresleme ile ulaşılabilir. Şekilde görüldüğü gibi bit-bit adreslenebilen bölge iki kısma ayrılmıştır. İlk kısım iç RAM'ın 20H adresinden başlayıp 2FH adresine kadar olan bölgedir. Bu bölgeye hem 8 bit normal adresleme yoluyla erişilebildiği gibi aynı zamanda da bit-bit adresleme yoluyla da erişilebilir.

Bit-bit adreslenebilen bu bölgenin 00H adresi iç RAM' in 20H adresinin 0. bitine karşılık gelmekte ve en son 7FH adresi ile RAM' ın 2FH adresinin 7. bitine karşılık gelmektedir. Örneğin;

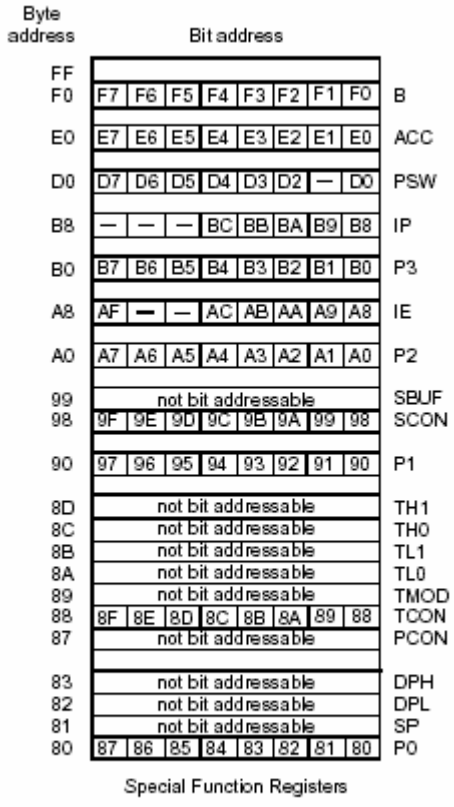
```
MOV     A,#5AH          ; A register'ına 5AH bilgisi atanır.

MOV     24H,A           ; 5AH bilgisi 24H adresine A üzerinden aktarmış
                        ;oluyoruz.

MOV     C,24H.4         ;Bu satırda 24H adresinin 4. biti bir bitlik Carry bitine
                        ;atanmış oldu. Bu işlem sonucunda C biti 1 ile
                        ;yüklenmiştir.
```



Şekil 3.8 Dahili RAM'ın yüksek 128 byte 'ı



Şekil 3.9 SFR Alanı

Şekil 3.9 Özel İşlem Register (SFR) alanının bir özetini göstermektedir. SFR' ler port latch' lerini, timerları, çevresel kontrolleri v.b. içerir. Bu registerlara doğrudan adresleme ile ulaşılabilir. SFR alanı içerisindeki 16 adres hem byte hem de bit adreslenebilirdir. Sonu 0 veya 8 ile biten SFR' ler bit adreslenebilirdir. SFR alanındaki bit adresleri 80H'den FFH'e kadardır.

Bu bölgenin farkı ise buradaki bitlerin değişimi programı direk olarak etkileyecektir. Örneğin ABH adresinde bulunan IE kesme register'inin 0. bitine "1" yüklenirse INT0 kesme izini verilmiş olur, "0" yüklenmesi durumunda ise verilen bu izin iptal edilmiş olur. Dolayısıyla bit-bit adreslenebilen hafıza bölgesine bu şekilde ulaşarak, programın akışı istenildiğinde değiştirilebilir.

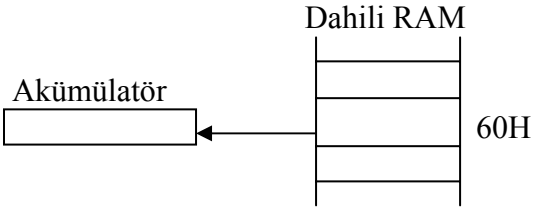
3.3. YAZILIM ALTYAPISI

3.3.1. ADRESLEME MODLARI

3.3.1.1. DOĞRUDAN ADRESLEME

Doğrudan adresleme, 8 bitlik bir adresteki 8 bitlik bir sayının sadece adres özel isimleri veya sayıları belirtilerek yer değiştirilmesidir. Bu modu işleyen kodda "#" simgesi kullanılmaz. Adresleri işaret eden sayılar 8 bitin üzerine çıktığında ise Assembler programı bir hata mesajı vermemesine karşın hatalı bir çalışma gerçekleşecektir.

```
MOV   A,60H       ;60H adresinde tutulan sayıyı register A'ya yaz
```



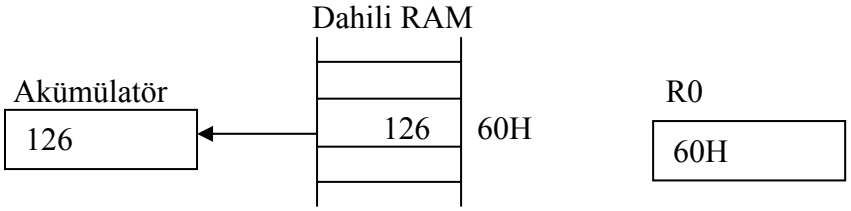
3.3.1.2. DOLAYLI ADRESLEME

Diğer genel amaçlı registerlardan farklı olarak R0 ve R1 ayrıca 16 bitlik bir register olan DPTR ile dolaylı adresleme için kullanılabilir. Bu işlemi adım adım anlatmak daha açıklayıcı olacaktır.

```
MOV   60H,#126    ;Bu komutla 60H adresinde 126 sayısını saklıyoruz

MOV   R0,#60H     ;Bu komutla R0 içerisine 60H sayısını yazarak bu
                  ;adresi işaret etmesini sağlıyoruz.

MOV   A,@R0       ;R0' ın gösterdiği  60H adresindeki 126 sayısını
                  ;A içerisine yazar
```



DPTR kullanarak da aynı işlem yapılabilir; ancak farklı olarak DPTR içine 16 bitlik adreslerde yazılabilir. Bu işlemde harici RAM kullanıldığında MOVX, dahili RAM kullanıldığında ise MOVC kullanılır

```
MOV   DPTR,#1200H       ;Harici adresi gösteren sayı DPTR'ye atılır

MOVX  A,@DPTR           ;DPTR' nin gösterdiği adres içeriği A' ya
                        ;aktarılır

MOV   DPTR,#0122H       ;Dahili adresi gösteren sayı DPTR'ye atılır
MOVC  A, @DPTR          ;DPTR' nin gösterdiği adres içeriği A' ya
                        ;aktarılır
```

3.3.1.3. REGISTER ADRESLEME

Önceden de belirtildiği gibi 8051 sisteminde R0-R7 gibi 8 farklı genel amaçlı register bulunur.Bu registerlara kullanıcının yazdığı program doğrultusunda veri atanabilir başka hiçbir şekilde içerikleri kendiliğinden değişmez. Bu registerlar ile doğrudan ve ivedi adresleme yapılabilir.

```
MOV         R1,#20H     ; R1 içerisine 20H yaz

MOV         R5,50H      ; 50H içeriğini R5 içerisine yaz

MOV         R2,A        ; Akümülatör içeriğini R2 içerisine yaz

MOV         A,R6        ; A içine R6 içeriğini ata
```

3.3.1.4. SFR ADRESLEME

Genel amaçlı registerlar gibi özel amaçlı registerlar da doğrudan veya dolaylı olarak erişilebilirler. Bu registerların adresleri 8051 mimarisinde özel bölgelerde tutulduğundan adresi gösteren sayıyı yazmak yerine bu isimler kullanılabilir. Örneğin ACC'nin 7FH-FFH aralığındaki adresi E0H dir; fakat bu şekilde okunabilir bir kod yazılamayacağından ACC ismi kullanılır ve assembler bu komutu direkt olarak adrese eşitler. Gereken sayı E0H'e yazılır ve yine ACC ile çağrılabilir.

3.3.1.5. İVEDİ ADRESLEME

Bu modda istenilen adrese aktarılacak olan veri komut içinde yazılır. Fakat sayı o registerin alabileceğinden büyük olduğunda assembler hata gösterecektir. Bu tip adreslemeye ait kodu yazarken MOV komutundan sonra birinci operand yazılacak olan adres veya register ikinci operand ise başında "#" işareti bulunan bir sayıdır. Eğer ivedi verinin ilk karakteri 0 – 9 arasında değil ise (örneğin hexadecimal gösterilimde A,B,C,..,F ise başına 0 koyulur.

```
MOV   A,#021H     ; A'nın içerisine 21H sayısını yaz.
```


```
MOV   A,#FFH      ;FFH Look Up Table ının başlangıç adresini A'ya ata
                  ;anlamına gelir bu işlem teknik olarak zaten
                  ;imkansız olmasının yanı sıra istediğimiz işlemin
                  ;yani FFH sayısı-nı A nın içerisine atma işlemi ile
                  ;hiçbir ilgisi yoktur.
```

3.3.1.6. INDEXED ADRESLEME

Bu tip adreslemede bir look-up table --verilerin sadece okunabilir özelikte olan program bellek içerisine daha önceden kaydedilmesiyle oluşur-- DPTR SFR (Special Function Register)' i yardımıyla okunarak istenilen adrese kopyalanabilir. Bunu sağlamak başta biraz zorlayıcı ve gereksiz görünse de bazı uygulamalarda vazgeçilmez bir rol oynar.

Kodun herhangi bir yerine yazılabilir olmasına rağmen, look-up table'ın tüm kodun sonuna eklenmesi kullanımda kolaylık getirecektir. Aşağıda veri bank (DB) komutu kullanılarak nasıl bir look-up table oluşturulduğu gösterilmiştir:

```
LOOK_UP_TABLE:          ;look-up table'dan veri çağırırken
                        ;kullanacağımız isim

      DB    00FFH       ;Bu bölgeye başında DB komutu ile gereken veriler
            gelir
      DB    10011100B   ;bu veriler direkt olarak sayılardır.Başka türlü
      DB    31          ;adreslenemeyeceğinden diğer adreslemelerdeki gibi
      DB    013H        ;istenen veri yazılırken başına '#' işareti
                        ;konmasına gerek yoktur.
```

Bu tarz bir tablo hazırlamasının alt program (subroutine) etiket (label) tasarımı ile karışabileceği için bu tip tabloların tüm program kodunun en altında yer almasının önemi bir kez daha anlaşılır. Sıra geldi bu tablodan ihtiyacımız olan bilgileri okumaya. Bunun için çok belirgin ve değişmez kalıplı bir kod dizisi kullanılabilir.

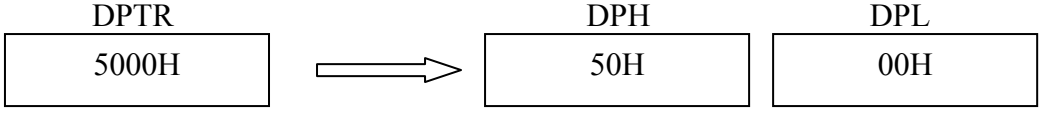
```
MOV   DPTR,#LOOK_UP_TABLE     ; Look-Up-Table başlangıç adresini DPTR
                              ; içerisine ata.
```

Haluk Turksoy Sk, No:11 ALTUNIZADE, 34662 ISTANBUL-TURKEY
TEL: +90 216 6518000 FAX: +90 216 6518009
http://www.elektro.com.tr info@elektro.com.tr

Look-Up-Table ‘ın başlangıç adresinin DPTR içerisine atanması aşamasında Look-Up-Table’ın ismini kullanabileceğimiz gibi; table’ın başlangıç adresini gösteren bir sayı da kullanabiliriz.

```
MOV   DPTR,#5000H            ;5000H adresi DPTR içerisine yerleştirildi.
```

Bu komut sonucunda :

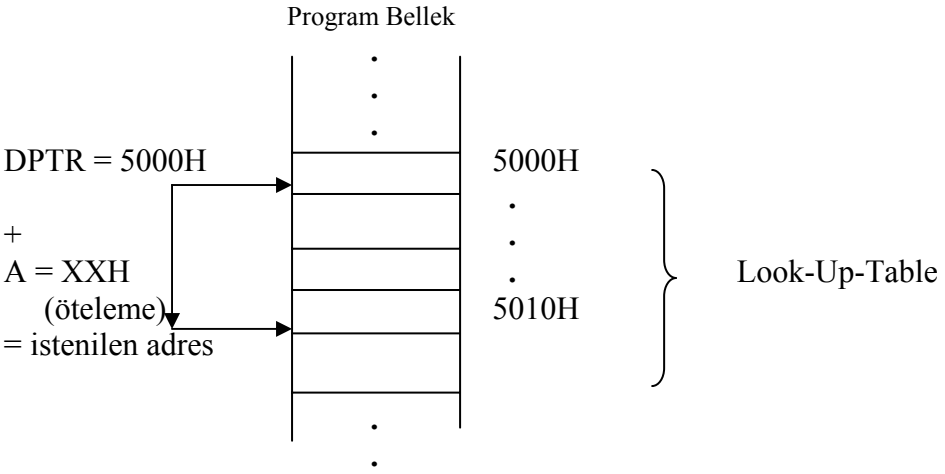


Böylece Look_Up_Table ‘ın başlangıç adresi DPTR registeri içerisine kaydedilir. Look – Up_table içerisinde istenilen kayda erişmek için ise program içerisinde değişen bir değişken kullanılır. Böylece Look-Up-Table içerisine istenilen adrese peogram kontrollü erişim sağlanmış olur.

```
MOV A , SANIYE          ;SANIYE program içinde deger alan bir değişkendir.
                        ;Bu değişken ile öteleme  değeri A registerına
                        ;atanmış olur.

      ; Bu aşamda Look-Up-Table' daki istenilen degere artık erişilebilir.

MOVC  A,@A+DPTR         ;Look-Up-Table'ın başlangıç adresi ile akümülatörde
                        ;bulunan tablo içerisinde istenilen değere gitmeni
                        ;sağlayan öteleme değeri toplanır. Bu toplam sonucu
                        ; gitmek istediğimiz adres değeridir.Komutun icrası
                        ;ile adreste yer alan bu bilgi akümülatore
                        ;aktarılmış olur.
```



Bu iki satırlık komut sayesinde look-up table çok basit bir şekilde okunabilir. Bu komutun en gerekli kullanım şekli kod örneklerinde bulunan "Seven Segment LED Display" sürme programında incelenebilir.

3.3.2. 8051 KOMUT SETİ, KOMUT GRUPLARI

8051 Ailesi MCU'lar komut seti yapısı açısından CISC (Complex Instruction Set Computer) olarak sınıflandırılmaktadır. RISC (Reduce Instruction Set Computer) mimarisi ile karşılaştırıldığında ilk bakışta daha az gelişmiş bir yapı olduğu düşünülmesine karşın güçlü komutları ile ciddi avantajlara sahiptir. RISC mimarisine sahip MCU'larda ancak birkaç komutla yapılabilen işlemler 8051 de tek bir komutla gerçekleştirilebilmektedir. Takip eden bölümlerde örnekler üzerinde açık bir şekilde görülebileceği gibi, özellikle CJNE, DJNZ gibi karşılaştırma ve dallanma; çevrim kontrol komutları vb sayesinde komut seti çok daha verimli bir şekilde kullanılabilmektedir.

Yanlış olan bir başka yaygın görüş ise RISC mimarisine sahip MCU'ların komut setlerinin daha kolay öğrenilebileceği düşüncesidir. Bir MCU üreticisi firma tarafından "Easy to learn 33 Instructions" sloganı ile bayraklaştırılan bu görüş konunun uzmanı olmayan MCU kullanıcıları ve öğrenciler arasında çok geniş bir illüzyona yol açmıştır. Gerçekte her hangi bir komut seti birkaç günlük çalışma ile öğrenilip, çeşitli uygulamalar ve pratik çalışmalar ile birkaç hafta içinde rahatlıkla iyi seviyede kullanılabilir. Esas önemli olan o komut setinin ne kadar rahat kullanılabildiği, arzu edilen işlemleri yerine getirmede ne gibi avantajlar sağladığı, nasıl hızlı çalışan kodlar hazırlanabildiğidir. Ayrıca MCU mimarisinin desteklediği ölçüde çarpma – bölme gibi komutlara sahip olabilmesi, bit seviyesinde kontrol işlemlerinde ne kadar kuvvetli olduğu da diğer önemli parametreler arasında anılabilir.

Esasen bütün komut setleri belli tip komut grupları içerirler ve büyük benzerlikler gösterirler. Veri Transferi, Veri İşleme ve Program Akış Kontrolü olmak üzere başlıca üç grup halinde sınıflandırılabilecek bu komut gruplarının fonksiyonları ve kullanımları aşağıda ayrıntılı olarak açıklanmıştır.

Çeşitli komut setlerinin benzer yapılara sahip olmasının genel olarak geçerli olmasının yanı sıra 8051 için özel olarak bakıldığında farklı mnemonic koda sahip 39 komut olduğu görülmektedir.

Veri Transfer Komutları	Aritmetik İşlem Komutları	Lojik İşlem Komutları	Program Akış Kontrol Komutları
MOV	ADD	ANL	CALL
MOVC	ADDC	ORL	RET
MOVX	SUB	XRL	RETI
PUSH	MUL	CLR	JMP
POP	DIV	CPL	JZ
XCH	INC	RL	JNZ
XCHD	DEC	RRC	JC
	DAA		JNC
			JB
			JNB
			JBC
			CJNE
			DCNZ

Tablo 3.1

3.3.2.1. VERİ TRANSFER KOMUTLARI

Veri transfer komutları, assembly dilinde hazırlanan yazılımlarda en sık kullanılan komut tiplerinden biridir. PUSH, POP, XCH gibi özel yapıda olanları ayrıca açıklamak üzere bırakırsak, MOV hedef, kaynak (MOV destination, source) yapısında oldukları söylenebilir.

MOV komutu, MOVC ve MOVX türevleri ile birlikte özel fonksiyon registerleri, dahili ve harici hafıza birimleri (SFR; Internal RAM, External program & data memory) arasında veri transferi gerçekleştirmek üzere kullanılmaktadır.

3.3.2.1. a) DAHİLİ VERİ HAFIZASI ve SFR'LER ARASI VERİ TRANSFERİ

Hedef ve kaynak değişkenlerinin alabileceği farklı operandlar ile toplam 16 farklı komut MOV mnemonic koduna sahiptir. Kaynak adresteki veriyi hedef adrese yazan komutun çalışma mantığı oldukça basittir. Ancak çeşitli operandların kullanılabilirliği zengin türevlere sahiptir.

MOV	A, source	;kaynak:	R_n ,direct, @R_i , #data
MOV	R_n , source	;kaynak:	A, direct, #data
MOV	direct, source	;kaynak:	A, R_n, direct, @R_i, #data
MOV	@R_i , source	;kaynak:	A, direct, #data
MOV	DPTR , #data		

Tablo 3.2

Program hafızasında işgal edilecek alanın azaltılması amacıyla dahili veri hafızasındaki genel amaçlı bölgelerin yerine Rn kodlu registerlerin kullanılması önerilebilir.

```
MOV     A, 50H  yerine MOV A, R2 gibi.
```

Ancak hazırlanacak yazılımların rahat okunulup anlaşılabilir olması açısından anlamlı isimler verilerek tanımlanmış değişkenlerin kullanımı tercih edilmelidir. Örneğin:,

```
MOV A, R4 yerine        SANIYE  DATA    50H
                        MOV     A, SANIYE   gibi.
```

Yukarıdaki örnekte Akümlatöre alınan R4 register içeriğinin hangi bilgiyi taşıdığı belli değilken, alternatif yöntemde saniye (zaman) bilgisi olduğu rahatlıkla anlaşılmaktadır. Her iki yöntemde işletim süresi 1 makine çevrimidir. (1µsn @ 12 MHz). Ancak Rn kullanımı halinde 1 byte olan kod , direct değişken kullanımı ile 2 byte'a çıkmıştır.

Sonuç olarak karar, yazılımı hazırlayan programcıya kalmıştır.

3.3.2.1. b)HARİCİ VERİ ve PROGRAM HAFIZASI İLE İLGİLİ KOMUTLAR

MOV komutunun benzeri fonksiyona sahip olmakla birlikte harici hafıza birimleri ile ilgili transfer işlemleri gerçekleştiren komutlar aşağıda açıklanmıştır.

Program hafızasında yer alan bir tablodan sayısal bir değerin okunması amacıyla kullanılacak olan MOVC komutu, Data Pointer (DPTR) veya Program Counter (PC)'ı tablo başlangıç adresi olarak kullanır. Bu değere akümülatörün (ACC) içeriğini ekleyerek ikisinin toplamından kaynak hafıza adresini hesaplar. Bu adreste yer alan sayısal değeri Akümlatörün içine yükler.

MOVC A, @A+ DPTR
MOVC A, @A+PC

Veri hafızasında yer alan herhangi bir adresten sayısal bir değerin okunması yada adrese herhangi bir veri yazılması amacıyla MOVX komutu kullanılmaktadır. Adres pointer olarak Data Pointer (DPTR) veya Ri (R0,R1) kullanılabilir. Ri 0000H – 00FFH aralığında yer alan ilk 256 byte 'lık alan içerisinde bilgi okuma ve yazmada kullanılırken; DPTR 64KB 'lık veri hafızası içerisinde bilgi okuma ve yazma işlemini gerçekleştirmek için kullanılır. Ri ve DPTR ile gösterilen adreste yer alan bilgi akümülatör içerisine yazılır.

MOVX A, @DPTR
MOVX A, @Ri

3.3.2.1. c) STACK TRANSFER KOMUTLARI

Verinin yığın (stack) içerisinde depolanmasını sağlamak için PUSH ve POP veri transfer komutları kullanılır. Yığına atmak yada çekmek istediğimiz veri için doğrudan adresleme modu kullanılır. PUSH komutu ile yığın işaretçisinin (Stack Pointer - SP) değeri 1 arttırılır ve veri yığın işaretçisinin gösterdiği yere veri yazılır. POP komutunda ise yığın işaretçisinin gösterdiği alandaki veri belirtilen adres alanına yazılır ve SP değeri 1 azaltılır.

PUSH	direct
POP	direct

Örneğin;

```
PUSH    DPH
```

Komutu ile DPH registerı içerisindeki değer yığın içerisine yazılır.

```
POP    35H
```

Komutu ile SP registeri ile gösterilen adresteki (yığının en üstünde) yer alan 1 byte 'lık bilgi 35H adresi içerisine yazılır.

Akümülatör içerisinde yer alan bilginin içeriğinin başka bir alanda yer alan veri ile değiştirilmesini sağlamak amacıyla XCH komutu kullanılır. Değiştirilecek olan veri Rn (R0..R7) içerisinde, Ri (R0,R1) ile gösterilen adres içerisinde yada verilen herhangi bir adres alanı içerisinde yer alabilir. Bazı özel durumlar için akümülatör içerisindeki verinin düşük 4 bitini başka bir verinin düşük 4 biti ile değiştirmemiz gerekebilir. Bu durumlarda ise XCHD komutu kullanılır. XCHD komutu akümülatör içerisindeki verinin düşük 4 biti ile Ri (R0,R1) ile gösterilen adres alanı içerisindeki verinin düşük 4 bitinin değiştirir.

XCH	A, Rn
XCH	A, @Ri
XCH	A, direct
XCHD	A, @Ri

3.3.2.2. ARİTMETİK İŞLEM KOMUTLARI

Aritmetik işlem komutları, adından da anlaşıldığı gibi toplama, çıkarma, çarpma, bölme gibi aritmetik işlemleri gerçekleştiren komut tiplerini içerir. Bu komutları, toplama-çıkarma, çarpma-bölme, arttırma-eksiltme işlerini gerçekleştirmelerine göre üç gruba ayırabiliriz.

3.3.2.2. a) Toplama – Çıkarma Komutları

Toplama ve çıkarma işlemini ADD, ADDC ve SUBB komutları gerçekleştirir. Bu komutların kullanımında akümülatör daima kaynak register olarak kullanılır.

ADD komutu ile kaynak alandaki veri, akümülatör ile toplanır ve sonuç akümülatör içerisine yerleştirilir. ADDC ile ise normal toplama işlemine ek olarak elde bayrağı içerisindeki bitin de toplamaya katılması sağlanmış olur. Kaynak registerlarının aldığı farklı değerler göre ADD ve ADDC komutu farklı mnemonic kodlara sahiptir.

ADD	A, source	;kaynak:	R_n ,direct, @R_i , #data
ADDC	A, sourec	;kaynak:	R_n ,direct, @R_i , #data

Çıkarma işleminde de toplama işleminde olduğu gibi sonuç yine akümülatör içerisinde saklanır. Kaynak byte ve elde bayrağı akümülatörden çıkartılır ve sonuç akümülatör içerisine yazılır. Kaynak registerlarının aldığı farklı değerler göre SUBB komutu farklı mnemonic kodlara sahiptir.

SUBB	A, source	;kaynak:	R_n ,direct, @R_i , #data

3.3.2.2. b) Çarpma – Bölme Komutları

Çarpma ve bölme işlemlerini gerçekleştirmek için MUL ve DIV komutları kullanılır. MUL komutu; akümülatör ve B register'larında yer alan 8-bit tam sayıları çarpar. 16-bit sonucun düşük değerli byte'ı akümülatöre, yüksek değerlikli byet'ı ise B register'ına yerleştirilir. DIV komutu; akümülatör içerisindeki değer B registeri içerisindeki değer ile bölünür. Tam sayı sonuç akümülatöre, kalan ise B registerına yerleştirilir. A ve registerlarının içerikleri, 8-bit işaretsiz tamsayı olarak işlem görür.

MUL	AB
DIV	AB

Aritmetik işlem komutları içerisinden MUL ve DIV komutu fazla makine çevrimi gerektiren komutlardır. Bu amaçla çarpma ve bölme işlemleri için ekstra donanım yapıları kullanılarak bu süreninin kısaltılması çalışılır.

3.3.2.2. c) Arrtıma – Eksiltme İşlemi Komutları

Herhangi bir adresteki verinin arttırılması yada eksiltilmesi için INC, DEC komutları kullanılır. Bu komutlar sayesinde herhangi bir adrese yada veri bloğuna erişilebilir.

INC ve DEC komutları farklı adresleme modlarında verilen 8-bit verinin arttırılması yada eksiltilmesinde kullanılır. Ayrıca; INC komutu ile DPTR registeri içerisideki 16-bit'lık sayının arttırılması da sağlanabilir.

INC	byte	;byte : A, Rn, @Ri, direct, DPTR
DEC	byte	;byet : A, Rn, @Ri, direct

3.3.2.3. LOJİK İŞLEM KOMUTLARI

Lojik işlemlerin gerçekleştirilmesinde kullanılan komutlar aldıkları operandlara göre iki farklı grup altında toplanabilir.

3.3.2.3. a) 8-Bitlik Lojik Fonksiyon Komutları

Veri üzerinde AND, OR, XOR işlemlerini gerçekleştiren komutlar sırasıyla ANL, ORL ve XRL 'dir. Kaynak ve Hedef registeri bitleri arasında istenilen lojik işlem gerçekleştirilir ve oluşan yeni değer kaynak registerı içerisine yerleştirilir. Bu komutlar kaynak ve hedef operandlarında aldıkları değerlere göre farklı mnemonic kodlara sahiptirler.

ANL	A, source	;source: Rn, @Ri, direct, #data
ANL	direct, source	;source: A, #data
ORL	A, source	;source: Rn, @Ri, direct, #data
ORL	direct, source	;source: A, #data
XRL	A, source	;source: Rn, @Ri, direct, #data
XRL	direct, source	;source: A, #data

3.3.2.3. b) Temizleme – Tümleme Komutları

Akümülatör içerisindeki veri üzerinde temizleme (clear), tümleyen alma (complement) lojik işlemlerini gerçekleştirmek üzere sırasıyla CLR, CPL komutları kullanılır.

CLR	A
CPL	A

3.3.2.3. c) Öteleme İşlem Komutları

Akümülatör içerisindeki verinin sola, sağa ötelenmesi için RL ve RR komutları kullanılır. Ayrıca elde bitini de öteleme işlemine dahil edebiliriz. Bunun için RRC ve RLC komutları kullanılır. Akümülatör içerisindeki verinin düşük anlamlı 4 bit ve yüksek anlamlı 4 bitin yer değiştirmesi lojik işlemlerini gerçekleştirmek için SWAP komutu kullanılır.

RL	A
RLC	A
RR	A
RRC	A
SWAP	A

3.3.2.3. d) 1-Bitlik Lojik Fonksiyon Komutları

Hedef elde biti olmak üzere AND, OR işlemlerini verilen kaynak biti ile bit düzeyinde gerçekleştiren komutlar sırasıyla ANL, ORL 'dir. Kaynak bitn önüne / koyularak bitin tersi ile bu lojik işlemlerin gerçekleştirilmesi sağlanabilir.

ANL	C, bit
ANL	C, /bit
ORL	C, bit
ORL	C, /bit

3.3.2.3. e) Bit Düzeyinde Temizleme – Tümleme Komutları

Elde biti yada verilen herhangi bit üzerinde temizleme (clear), tümleyen alma (complement) ve biti set etme lojik işlemlerini gerçekleştirmek üzere sırasıyla CLR, CPL, SETB komutları kullanılır.

CLR	source	;source : bit, C
CPL	source	;source : bit, C
SETB	source	;source : bit, C

3.3.2.4. PROGRAM AKIL KONTROL KOMUTLARI

Program akışı içerisinde programın farklı bir bölüme dallanmasını sağlayan komutlardır.

3.3.2.4. a) Koşulsuz Dallanma Komutları

Herhangi bir alt programa dallanma ve bu programdan dönüş için CALL ve RET komutları kullanılır. Alt programın bulunduğu yere göre CALL komutu ACALL ya da LCALL olarak iki gruba ayrılır. Alt program 11 bit ile adreslenebilen program belleğinde ise ACALL; 16 bit ile adreslenebilen program belleğinde ise LCALL kullanılır. Ancak; pratikte programcının sadece CALL kullanması yeterlidir. Assembly derleyici gerekli dönüşümleri gerçekleştirir. Alt programdan dönüşte ise RET komutu kullanılır. Eğer bu alt program bir interrupt ise RETI komutu kullanılır.

ACALL	addr11
ACALL	addr16
RET	
RETI	

Program içerisinde herhangi bir yere atlamak için JMP komutu kullanılır.Atlanılmak istenilen yere göre SJMP, LJMP ve AJMP olarak üç gruba ayrılır. Atlanılacak adres relative adres ise SJMP, 11 bitlik bir adres ise AJMP, 16 bitlik bir adres ise LJMP kullanılır. Ancak; pratikte programcının sadece JMP kullanması yeterlidir. Assembly derleyici gerekli dönüşümleri gerçekleştirir.

AJMP	addr11
LJMP	addr16
SJMP	
JMP	

3.3.2.4. b) Koşullu Dallanma Komutları

Belirli bir koşulun sağlanması durumunda programın bir yere atlamasını sağlayan komutlardır.

Elde bitinin, akümülatörün yada herhangi bir bitin 1 yada 0 olması durumuna göre program içerisinde bir bölgeye dallanma gerçekleştiren komutlar JC, JNC, JB, JNB, JZ, JNZ ‘dır. Ayrıca JBC komutu kontrol edilen bitin değeri 1 mı diye bakar, 1 ‘se biti 0 yapıp istenen yere dallanır.

JC	rel
JNC	rel
JB	bit, rel
JNB	bit, rel
JBC	bit, rel
JZ	rel
JNZ	rel

3.3.2.4. c) Karşılaştırma Komutları

Hedef ve kaynakta yer alan operandları karşılaştırarak istenilen değeri sağlaması durumunda dallanma gerçekleştiren komutlardır.

CJNE	A, direct, rel
CJNE	A, #data, rel
CJNE	Rn, #data, rel
CJNE	@Ri, #data, rel

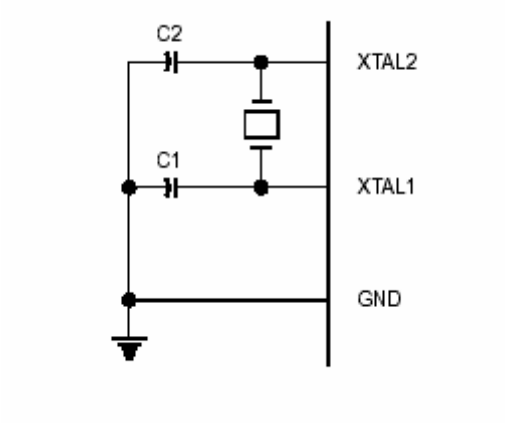
Karşılaşırıma işleminden önce azaltma işlemini de gerçekleştirebiliriz. Bu durum için DJNZ komutu kullanılır.

DJNZ	Rn,rel
DJNZ	direct, rel

Programın 1 makine çevrimi boyunca işlem yapmadan beklemesini istiyorsak NOP komutunu kullanırız.

3.3.3. İŞLEMCİ ZAMANLAMASI(CPU TIMING)

8051 mikrodenetleyicisinde, mikrodenetleyici içerisinde yapılan işlemlerin zaman uyumlu bir şekilde yapılmasını sağlayan bir iç osilatör bulunur. Bu osilatör sayesinde mikrodenetleyici donanımı yazılımla uyumlu bir şekilde gecikme yada başka zamanlama sorunu olmadan kararlı bir şekilde çalışır.



Şekil 3.10 : Osilatör Bağlantıları

Saat işaretinin üretilmesi için XTAL1 ve XTAL2 girişlerine bir kristal veya seramik rezonatör devresi bağlanır. Bu osilatör frekansı mikrodenetleyici içerisinde 12' ye bölünerek makine çevrimleri oluşturulur. Yani bir makine çevrimi 12 saat süresinden oluşur. Bu sürede mikrodenetleyici komutları yürütmek için kendi içinde birçok işlem yapar. Bu işlemler S olarak adlandırılan fazlarda yapılır. S1 den S6 ya kadar 6 S durumu vardır ve bu S durumları da kendi içinde P1 ve P2 fazlarından oluşur. Bir makine çevrimi 12 fazdan oluşur ve 12 MHz' lik kristalle bir makine çevrimi 1 μs dir.

3.3.4 MAKİNE ÇEVRİMLERİ

Her makine çevriminde yürütülen komutun operantları olarak program belleğinden iki okuma işlemi yapılır. Tek operantla çalışan komutlarda ikinci okuma işlemi dikkate alınmaz ve PC arttırılmaz. Şekil 3.11 a ve b de zamanlama diyagramında görülen bir makine çevrimlik komut yürütülmesinden önce komut, komut registerına (Instruction Register) gönderilen S1 durumuyla başlar. S4 durumunda yapılan ikinci okuma ve diğer işlemlerin ardından S6 durumundan sonra bir makine çevrimi tamamlanmış olur.

Veri belleğinden okuma işlemi, program belleğinden yapılan okuma işlemine göre iki kat daha fazla zaman alır. Şekil 3.12' de bu komut ve bu komut yürütülürken Port0 ve Port2' den üretilen adreslerin ALE, PSEN' ve RD' sinyallerinin zamanlaması görülmektedir. ALE

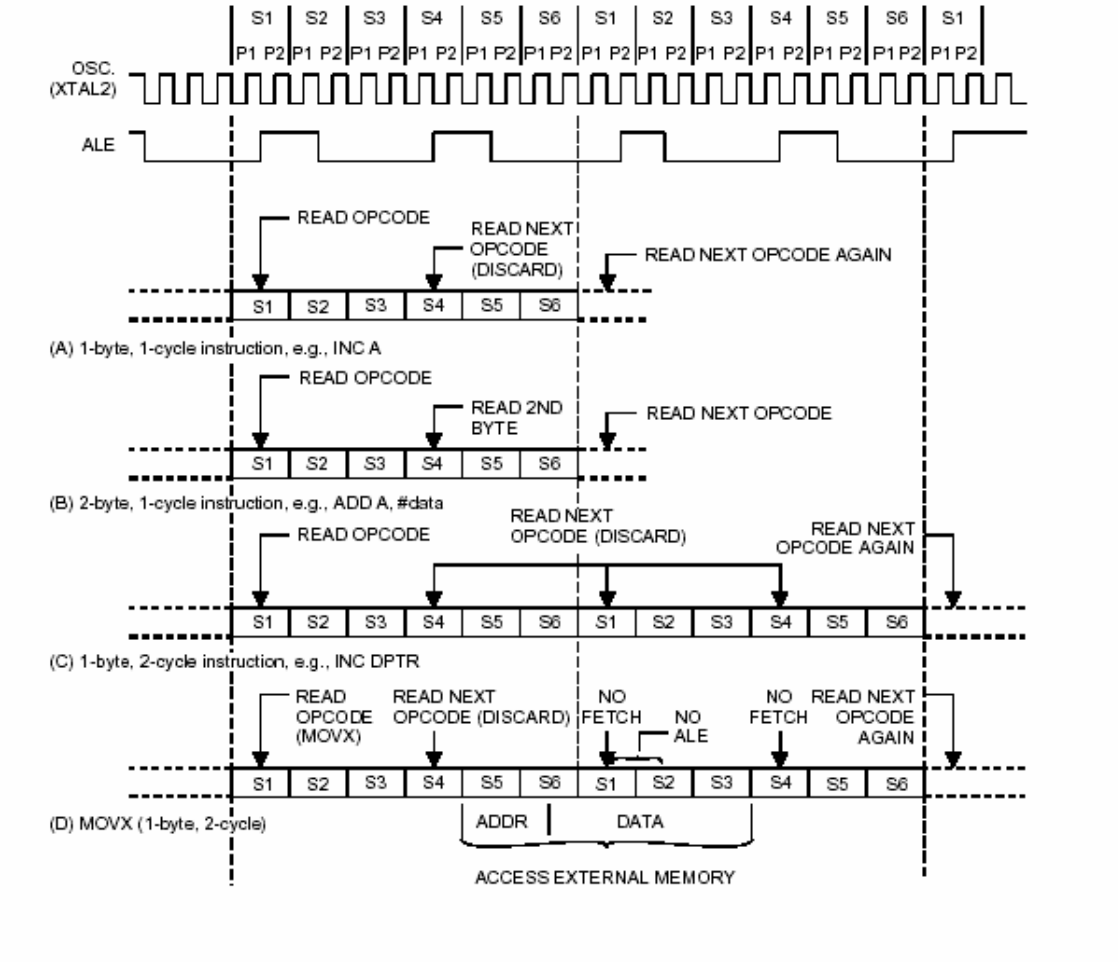
sinyali Port0 dan üretilen adresi, adres latch inde saklamak için kullanılır. Şekil 3.12 b de görüldüğü gibi MOVX komutunun yürütülmesinde ikinci makine çevriminde program hafıza okuma sinyali PSEN üretilmez. Program okumalarının dikkate alınmadığı tek an bu durumdur. Şekil 3.13 a ve b de harici program belleği okuma ve veri belleği okuma işlemlerinin her biri için gereken makine çevrimi gösterilmiştir.

Dahili ya da harici program belleğinden okuma yapmak için gereken süre aynıdır. Mikrodenetleyici dahili program belleğinden çalışırken, PSEN' sinyali aktif değildir ve mikrodenetleyicide Port0 ve Port2 den program adresleri üretilmez. Ancak her makine çevriminde ALE iki kez aktif olur. Bu özellik 12 MHz lik kristalli bir sistemde ALE ucundan 2 MHz lik saat sinyali üretildiği anlamına gelir. Şekil 3.12 b de görüldüğü gibi ALE sinyalinin bir tanesi, MOVX komutunun yürütülmesi sırasında üretilmez.

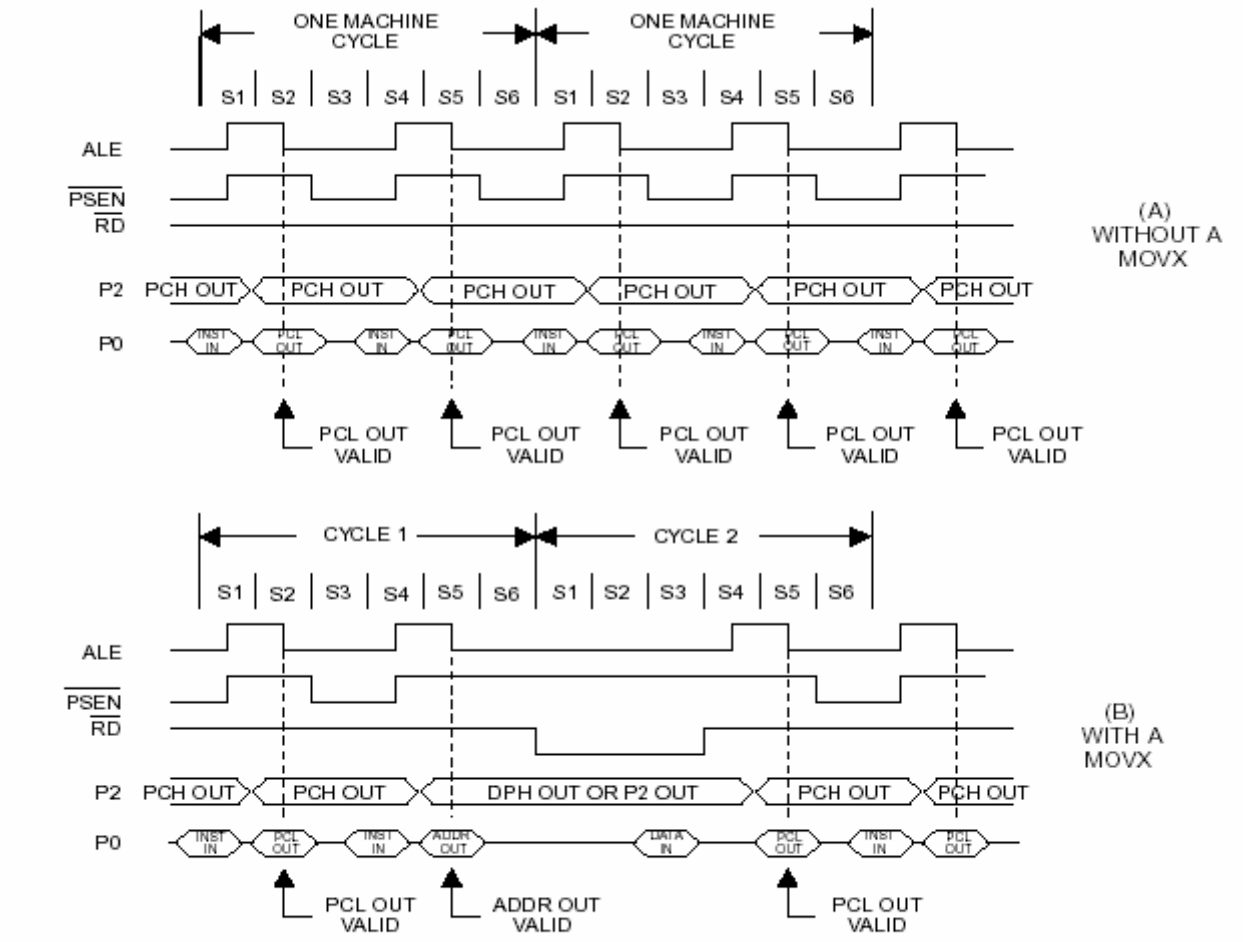
Mikrodenetleyicide komutlar yürütülürken öncelikle komutun ne olduğu çözülür (decoding), ardından bu komutun yapısına bağlı olarak, komuttan sonra gelen operantlarla komut yürütülür. Komutlar içerisinde, 1 byte ve 1 makine çevrimi süren komutlar olduğu gibi 3 byte ve iki makine çevrimi süren komutlarda vardır. Ayrıca bazı komutlar boyut olarak küçük olmasına karşın daha fazla makine çevrimi sürer. Bu fazla makine çevrimi süren komutlara MUL ve DIV komutları örnek verilebilir. Bu komutlar bölme ve çarpma işlemi yaptığından algoritmik olarak hem lojik hem de aritmetik işlemlerden oluştuğundan diğer komutlara göre daha uzun sürer. Ancak 8051 mikrodenetleyicisi DIV ve MUL komutları için optimize edilmiş bir yapıya sahiptir ve bu komutlar 4 makine çevrimi sürer.

Programların yürütülmesi için üretilen op-code (operation code) için 8051' de 8 bit olduğundan, 8051 mikrodenetleyicisinde 256 (2 ^8 = 256) komut vardır. Bu komutlardan sadece bir tanesi kullanılmaz, 255 komut ise bir mikrodenetleyici ile yapılabilecek aritmetik, lojik ve kontrol komutları için yeterlidir.

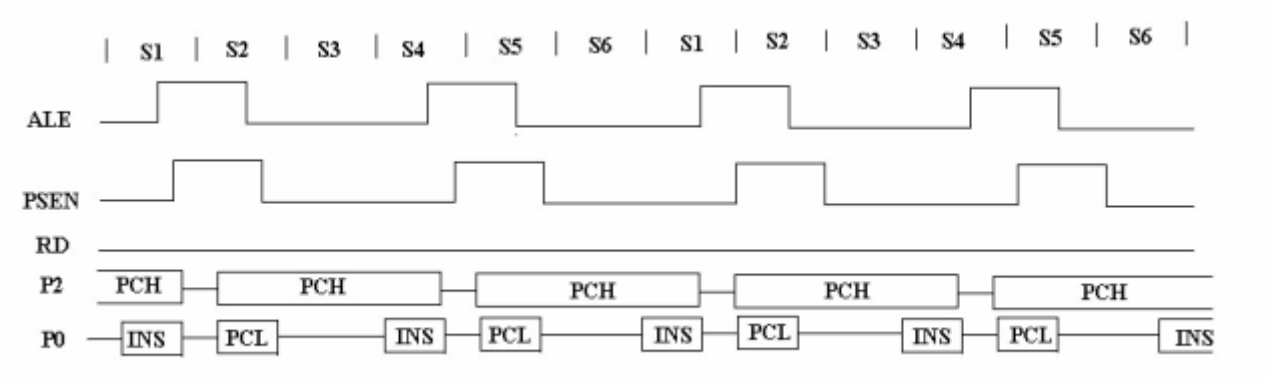
8051 mikrodenetleyicisi seri veri iletişim arabirimine sahiptir. Seri veri iletişim arabirimi için genellikle 11.0592' MHz lik kristal kullanılır. Bu kristalin kullanılmasının sebebi seri veri iletişimin yapısından kaynaklanır. Seri veri iletişiminde iki sistem arasında veri gönderimi için, gönderilecek veriye ek olarak başlangıç, bitiş ve kontrol sinyalleri de eklenir. Bu ek sinyallerle veri iletişiminde çerçeve boyutu daha fazla olur. 12 MHz frekansının çerçeve boyutuna küsuratlı olarak bölünmesinden dolayı seri veri iletişimi için optimize edilmiş 11.0592 MHz lik kristal kullanılır. Karşılaştırma yapmak gerekirse 12 MHz lik kristalle en fazla 4800 baud rate de iletişim yapılabilirken, 11.0592 MHz lik kristalle sıkça kullanılan 19200 baud rate de veri iletişimi yapılabilir.



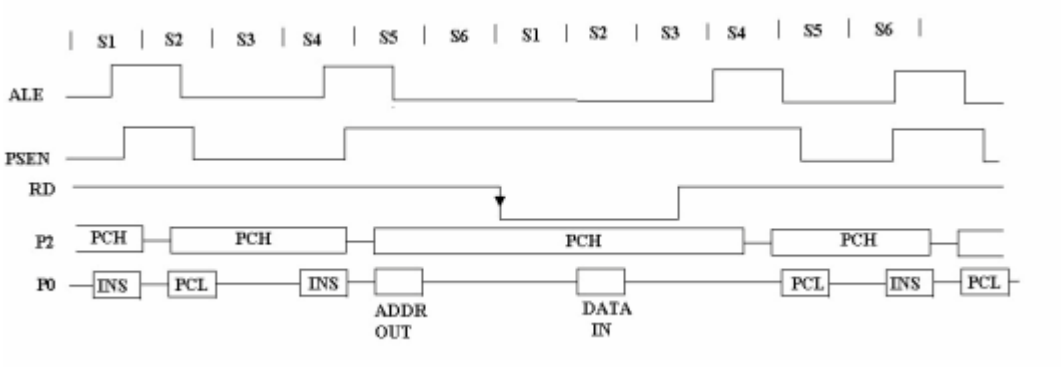
Şekil 3.11 Makine Çevrimlerinin durum (S) sinyalleri



Şekil 3.12 Harici Program Belleğinden yürütme sırasındaki yol çevrimleri.



Şekil 3.13 a


S1
S2
S3
S4
S5
S6
S1
S2
S3
S4
S5
S6
ALE
PSEN
RD
P2
PCH
PCH
PCH
PCH
P0
INS
PCL
INS
ADDR
OUT
DATA
IN
PCL
INS
PCL

Şekil 3.13 b