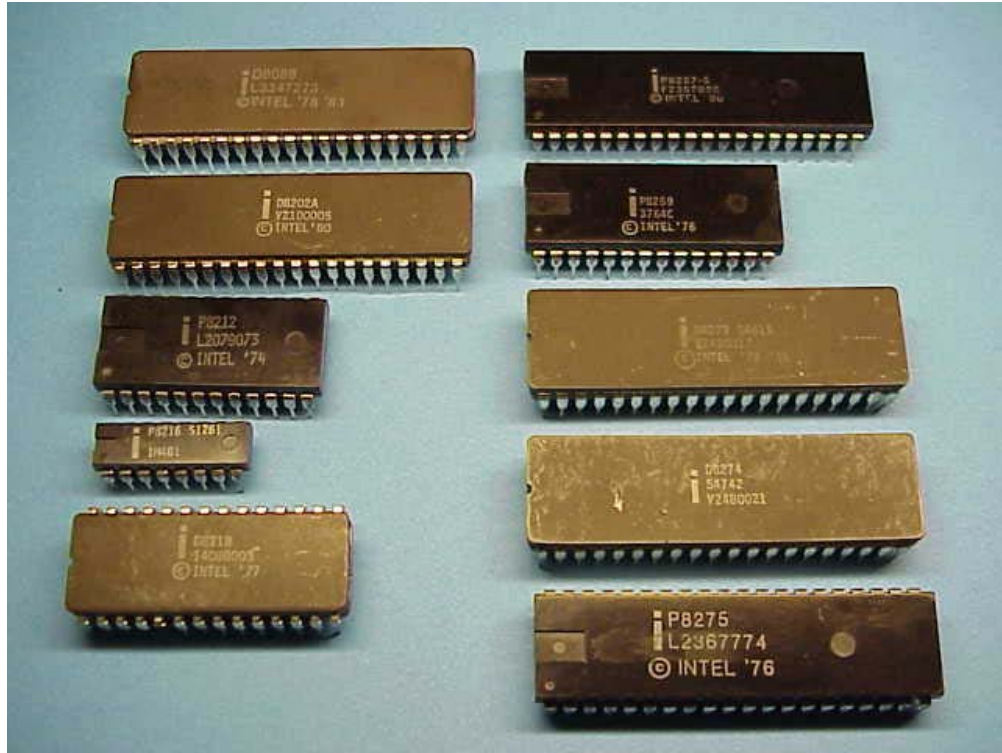


# *Mikroiřlemciler ve Mikrobilgisayarın Geliřimi*



## *Giriř*

Mikroiřlemci (microprocessor) icat edileli kısa bir sre olmasına raėmen modern hayatın bir parçası haline gelmiřtir. Gnlk yařamda otomobilde, televizyonda, telefonda, kapı otomatiėinde, asansrde, trafik ıřıklarında, hesap makinesinde, mzik aletlerinde, daktilolarda, oyuncaklarda, cep telefonlarında ve benzeri birėok cihazda farkında olmadan mikroiřlemcileri kullanmaktayız. Mikroiřlemciler daha

---

Mikrodenetleyiciler 8051 Uygulamaları

önce hayal edemeyeceğimiz hızla mektuplarımızı dünyanın her noktasına ulaştırmakta, hiç bir kimse ile karşılaştırılamayacak kadar kısa sürede karmaşık sayısal işlemleri yapabilmektedir.

## Mikroişlemcinin Gelişimi

İlk mikroişlemci 1971 yılında 4004 adıyla Intel firması tarafından üretilmiştir. 4004 çok güçlü bir işlemci değildi, bir adımda sadece 4 bitlik verileri işleyebiliyordu, buna rağmen birçok kişi için heyecan vericiydi çünkü tüm birimler tek bir tümdevre içinde toplanmıştı. Benzer teknolojiyi kullanarak Intel 1974 yılında 8 bitlik 8080 mikroişlemcisini üretti. Gerçek anlamda bilgisayarın boyutunu küçülten mikroişlemci, 8088 ise 1979 yılında üretildi ve 1982 yılında IBM firması bu işlemciyi kullanarak ilk kişisel bilgisayarı (PC) pazara sundu. İlerleyen yıllarda kişisel bilgisayar pazarına yönelik 80286, 80386, 80486, Pentium, Pentium II, Pentium III ve Pentium 4 işlemcileri Intel tarafından pazara sunuldu.

Adı	Yıl	Transistor Adedi	Frekans	Kelime Uzunluğu	Hız
8080	1974	6,000	2 MHz	8	$64 \times 10^4$
8088	1979	29,000	5 MHz	16, 8-bit Veri Yolu	$33 \times 10^4$
80286	1982	134,000	6 MHz	16	$1 \times 10^6$
80386	1985	275,000	16 MHz	32	$5 \times 10^6$
80486	1989	1,200,000	25 MHz	32	$20 \times 10^6$
Pentium	1993	3,100,000	60 MHz	32, 64-bit Veri Yolu	$100 \times 10^6$
Pentium II	1997	7,500,000	233 MHz	32, 64-bit Veri Yolu	$\sim 300 \times 10^6$
Pentium III	1999	9,500,000	450 MHz	32, 64-bit Veri Yolu	$\sim 510 \times 10^6$
Pentium 4	2000	42,000,000	1.5 GHz	64 bit	$\sim 1,700 \times 10^6$

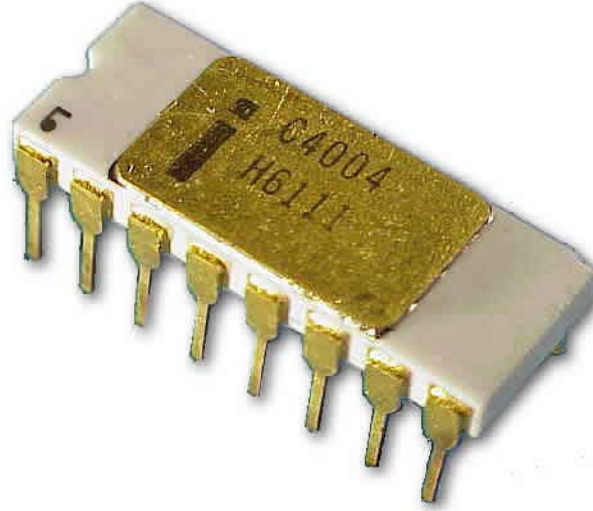
Çizelge–1.1 Intel firması tarafından üretilen mikroişlemcilerin gelişimi ve özellikleri.<sup>1</sup>

IBM firmasının bilgisayar mimarisini kullanan birçok firma bu işlemciler ile daha ucuz kişisel bilgisayarlar ürettiler. 8088'den başlayarak Pentium 4'e kadar tüm işlemciler geriye doğru tüm işlevleri gerçekleştirebilmektedir. 8088'de çalışmak

<sup>1</sup> Kaynak:www.intel.com

## Mikrodenetleyiciler 8051 Uygulamaları

üzere yazılmış bir program aynı zamanda Pentium 4'te de çalışmaktadır. Intel firması tarafından üretilen işlemcilerin gelişimi Çizelge–1.1'de verilmiştir. Intel firması dışında Motorola 6800, RCA 1801, MOS Technology 6502 ve Zilog Z80 mikroişlemcilerini ürettiler.



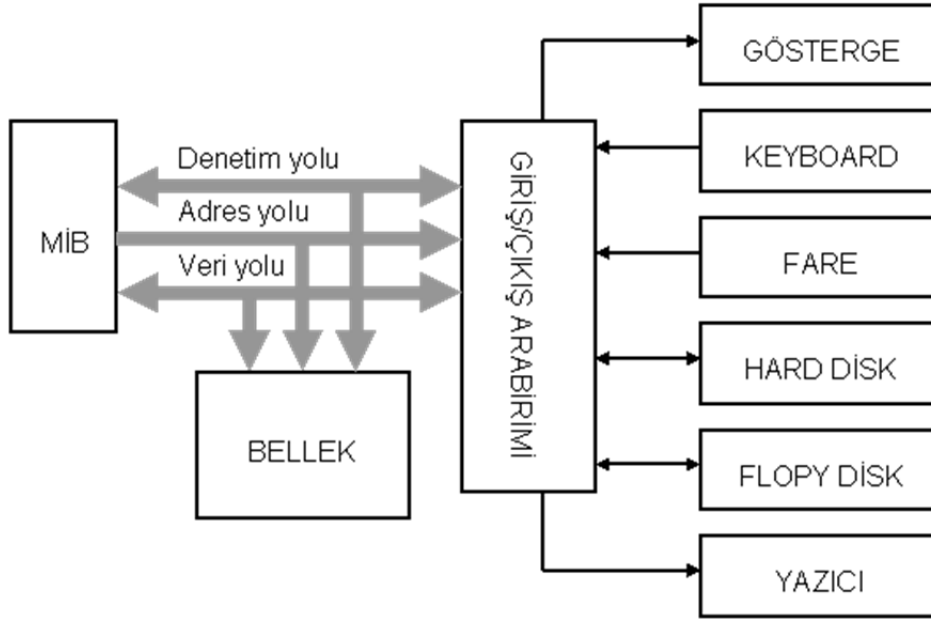
Şekil–1.1 Intel'in ürettiği ilk mikroişlemci 4004'ün görünümü.

## Mikrobilgisayarlar

Mikroişlemcinin yararlı bir işte kullanılması kalıcı ve geçici veri saklama belleklerinin ve insan veya makinelerle iletişim sağlayan çevre birimlerinin (peripheral devices) bağlanması ile mümkün olur. Elde edilen bu cihaza bilgisayar (computer) adı verilir. Birden fazla bilgisayarın standart bir iletişim ağı ile birbirlerine bağlanması ile elde edilen sisteme bilgisayar ağı (computer network) adı verilir. Bilgisayarı oluşturan elemanlara donanım (hardware), bilgisayarın ne yapacağını belirleyen komutlar zinciri olan programlara da yazılım (software) adı verilir.

Şekil–1.2'de temel bilgisayarın blok şeması gösterilmiştir. Yukarıda bahsedilen birimlerin yanı sıra adres, veri ve denetim bilgilerini birimler arası taşıyan hatların oluşturduğu adres, veri ve denetim yolları da yer almaktadır. Yollardaki hat sayısı ve özellikleri her mikroişlemcide farklıdır. Çevre birimleri bilgisayarın kullanılacağı işe göre farklılık gösterir. Kişisel bilgisayar olarak kullanıldığında en temel çevre

birimleri veri ve program saklama cihazları ( hard disk, floppy disk driver, CD driver gibi), görüntüleyici, keyboard, fare ve yazıcıdır. Bu birimlerin bağlantısı için ana kart üzerinde ara birim yongaları yerleştirilmiştir. Bağlantı uçları standart hale gelen soketler ile kasa üzerine veya ana kart üzerine yerleştirilmişlerdir. Endüstriyel bilgisayarlarda genellikle ana kart üzerine yerleştirilen genişleme soketleri kullanılarak arabirim kartı takılır ve cihaza bu karttan çıkış alınır. Bazı endüstriyel cihazlar ise doğrudan kasa üzerinde bulunan soketlerden biri kullanılarak bilgisayarla iletişim kurması sağlanır. İkinci yöntem yavaş olan sistemlerde kullanılabilir.



Şekil–1.2 Mikrobilgisayarın birimleri.

## Merkezi İşlem Birimi

Aritmetik ve mantık işlemleri yapabilen, veri saklayabilen, yaptığı işlemlerin sonucuna göre karar verebilen ve belleğe veri yazıp okuyabilen sayısal elektronik devrelerine işlemci (processor) adı verilir. İşlemciler ilk zamanlarda elektron tüpleri ile elde edildiklerinden, boyutları çok büyüktü. 1970’li yılların başında transistörlerle tek bir yongada elde edilen işlemciler elektron tüpleri ile elde edilenlere oranla çok küçük olduğundan işlemci kelimesinin başına küçük anlamına gelen mikro kelimesi eklenerek mikroişlemci (microprocessor) kelimesi elde edilmiştir. Bazı kaynaklar

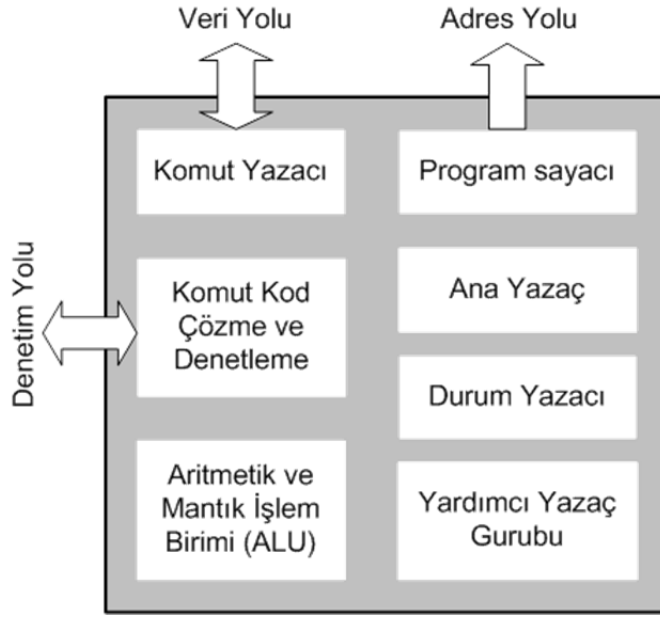
## Mikrodenetleyiciler 8051 Uygulamaları

işlemciyi merkezi işlem birimi kısaca MİB, (central processing unit kısaca CPU) olarak adlandırmışlardır. Günümüzde her iki isim de kullanılmaktadır.

Merkezi işlem birimi bilgisayarın beynidir, bilgisayardaki aritmetik mantık ve karar verme işlemleri ile bağlı birimlerin denetimi bu birim tarafından yapılır. MİB mantık devrelerinin birleşiminden oluşur ve sürekli yaptığı işlem komut getirme (fetching) ve komut yürütmedir (executing). MİB ikilik kodları yürütme yeteneğine sahiptir. Bu kodların her biri basit bir işlemi temsil eder örneğin toplama, çıkarma VE, VEYA, DEĞİL gibi. Tüm bu ikilik kod kümesine komut kümesi (instruction set) adı verilir. Her mikroişlemcinin kendine özgü bir komut kümesi vardır.

Şekil-1.3'de MİB'in basitleştirilmiş iç yapısı gösterilmiştir. Aritmetik ve mantık işlem birimi (arithmetic logic unit, ALU) aritmetik ve mantık işlemlerin gerçekleştiği yerdir. Komut kod çözme ve denetim birimi gelen komutun niteliğini belirler ve bu komut için gerekli denetim işaretlerini üreterek içerideki ve dışarıdaki birimlere gönderir. Ana yazaç aritmetik ve mantık işlem yapılırken birinci sayıyı ve işlem sonunda sonucun yazıldığı yazaçtır. Birçok mikroişlemcide bu yazaca akümülatör adı verilir. Durum yazacı her bitine ayrı bir görev verilmiş bir yazaçtır. Örneğin toplama işlemlerin sonucunun ana yazaca sığmayan kısmı durum yazacında yer alan elde bitinde saklanır. Bu yazaçtaki diğer bitlere ise mikroişlemci üreticisine göre farklı görevler verilmiştir, negatif biti, sıfır biti, ondalık işlem biti gibi. Yardımcı yazaç gurubu geçici veri saklamak için kullanılır. Yardımcı yazaç gurubunda yer alan yazaç sayısı mikroişlemci üreticisine bağlı olarak değişir. Program sayacı işletilecek komutun adres bilgisinin oluşturulduğu bir ikili sayıcıdır. Dışarıda doğrudan adres yoluna bağlıdır. Komut yazacı program sayacı tarafından adresi belirlenen, bellekten veri yolu ile getirilen komutun yazıldığı yerdir. Belleğe yazılacak veri yine bu yazaç yoluyla yanin veri yoluna yazmak için yine bu yazaç kullanılır.

Mikroişlemcinin bir makine saykalında işleyebileceği verinin uzunluğu ALU'nun bit sayısı ile sınırlıdır. Akümülatörün bit sayısı da ALU'ya eşittir. Akümülatördeki bit sayısına mikroişlemcinin kelime uzunluğu adı verilir. Kelime uzunluğu 8 bit olan mikroişlemcilerde akümülatör 8 bittir ve bir makine saykalında 8 bitlik iki sayıyı toplar ve en fazla 9 bit olarak sonucu akümülatör ve durum yazacında yer alan elde bitine yazar. 32 bit olan mikroişlemcilerde bir makine saykalında iki 32 bitlik sayı toplanabilir ve 33 bitlik sonucun 32 biti akümülatöre yazılırken bir biti elde bayrağına yazılır.



Şekil-1.3 Merkezi işlem biriminin blok şeması.

## Yarı İletken bellekler

Bilgisayarın işleteceği programlar ve işleyeceği veriler ile sonuç verileri bellekte saklanır. Sürekli kullanılan program parçaları kalıcı tür belleklerde saklanır, bu belleğin içeriği özel durumlar dışında bilgisayar tarafından değiştirilemez. İşleyeceği veriler ve işlem sonucunda elde ettiği veriler geçici tür belleklerde saklanır. Geçici ve kalıcı tür belleklerin yapısı MİB'ne doğrudan bağlanabilecek şekilde olmalıdır. Bu özelliği taşıyan bellekler kalıcı tür olan ROM, geçici tür olan RAM yarı iletken belleklerdir.

ROM bellek sadece okunabilen içeriği özel donanım olmadıkça değiştirilemeyen yarı iletken belleklerdir. Üretim sırasında programlanan mask ROM belleklerin, sonradan içeriği özel donanım olsa bile değiştirilemez. İçeriği sınırsız adet okunabilir, kullanım ömrü 50 yıl olarak kataloglarda verilmektedir. Özel programlayıcı ile bir defa programlanabilen sadece okunur belleklere ise PROM adı verilir. Bu belleğin diğer özellikleri mask ROM ile aynıdır. ROM'lar aynı program ile daha az sayıda cihaz üretilecek ise mask ROM'a göre daha ekonomik olmaktadır. Diğer bir ROM çeşidi ise silinebilir ve tekrar programlanabilir EPROM belleklerdir. EPROM bellekler özel programlayıcı ile programlanır, kullanıldıktan sonra içeriği değiştirilmek istendiğinde üzerinde bulunan pencereden 15 dakika ultraviyole ışık uygulanarak

## Mikrodenetleyiciler 8051 Uygulamaları

silinebilir. Silme işlemi sonrası tüm hücrelerin içerikleri mantık 1 olacaktır. Bu tür belleklerin programlanması ve silinmesi özel donanım ve süre gerektirmesi yeni tür ROM üretimine sebep olmuştur. Flash EPROM olarak adlandırılan belleklerin bilgisayar sistemlerinde kullanımı hızla yaygınlaşmıştır. Flash EPROM'lar birkaç bağlantı değişikliği ile kullanıldıkları devre üzerinde silinebilir ve aynı devre üzerinde programlanabilir. Silme işlemi belleğin tüm hücrelerini mantık 1 yapar, silinecek hücreyi seçme şansımız yoktur. İstenildiğinde programın kopyalanmasını engellemek için şifre konabilir. Diğer bir bellek türü ise elektrik ile silinebilir EEPROM'dur. EEPROM ile Flash EPROM'dan özelliği her satırının özel düzeneğe gerektirmeden diğerlerinden bağımsız siline bilmesidir. Hızlı silme işlemi kullanıldığında ise Flash EPROM'dan farkı yoktur. Bellekler hakkındaki daha geniş bilgiyi sayısal elektronik kitaplarından elde edebilirsiniz.

Bilgisayarda ROM bellek ilk açılışta işlemcinin çevre birimleri ile anlaşabilmesi için gerekli altprogramları saklamak için kullanılır. Bu alt programlar ilk açılışta işlemcinin adını, kullanılan arabirim yongalarının numaralarını ve RAM belleğin sağlamlık denetimini görüntüleyiciye yazar. Bu işlemlerin sonunda kullanıcıdan yeni bir komut bekler hale gelir. RAM bellekte ise kullanıcının sonradan girdiği veriler ve bu verilere ait sonuçlar yer alır. Büyük programlar ROM ve RAM bellekte saklanamaz, bu işlem için hard disk adını verdiğimiz manyetik bellekler kullanılır. Büyük program parçaları sıra ile RAM belleğe aktarılır ve işlemci buradan komutları getirir ve yürütür. İşlemi tamamlanan program parçası bir sonraki bölüm ile değiştirilir. MiB doğrudan hard diskten işlem yapamaz.

## Adres, Veri ve Denetim Yolları

Yol (bus) belli bir amaçla veri taşımak için kullanılan iletkenler kümesine verilen addır. MiB çevresindeki birimlere adres, veri ve denetim yolları ile bağlıdır. MiB işlem yapacağı birimi adres yoluna yazdığı adres bilgisi ile seçer. Yapacağı işlemin ne olduğunu denetim yoluna yazdığı bilgi ile bildirirken işlemin sonucunda oluşan bilgiyi veri yolu ile sonucun yazılması gereken birime taşır.

Adres yolu adresin mikroişlemci tarafından üretilmesinden dolayı tek yönlüdür ve bu yolda veri akışı MiB'den bellek veya arabirim tümdevrelerine doğrudur. Küçük ölçekli bilgisayarlar 16 veya 20 adres hattına sahiptirler. 1 hat iki bellek satırını adresleyebilir, 16 adres hattı ise;

$$2^{16}=65536 \text{ adet bellek satırını adresleyebilir.}$$

## Mikrodenetleyiciler 8051 Uygulamaları



$2^{10}=1024$  Satır K ile kısaltılırsa;

$$2^{16}=2^6 \cdot 2^{10}=2^6 K=64 K$$

Olarak adlandırılır. 20 adres hattına sahip işlemcilerde;

$2^{20}=1048576$  Bellek satırı adreslenebilir. Kısalttığımızda 1 Mega bellek satırı olarak söyleyebiliriz. Formül ile ifade edecek olursak;

$$\text{Belleme kapasitesi} = 2^{\text{Adres Hattı Sayısı}}$$

Şeklinde yazabiliriz.

Veri yolu mikroişlemcinin işleyeceği komutları bellekten komut yazacına taşır veya akümülatörde oluşan sonuçları geçici belleğe taşır. Veri yolundaki bilgi akışı çift yönlüdür. Veri yolundaki hat sayısı mikroişlemcinin kelime uzunluğuna eşit olur. Araştırmalar sonucuna göre, mikro işlemcilerin yaptığı işlemlerin üçte ikisi kendi yazaçları ile bu birimler arası veri aktarma işlemleri olduğu görülmüştür. Performansı yüksek işlemci elde etmek istiyorsak veri yolunun mümkün olduğu kadar hızlı ve geniş tutulması gerekmektedir. Veri yolunun genişliği mikro işlemcinin komut sayısını belirler. Komut sayısının fazla olması mikro işlemcinin yeteneğini arttırır.

$$\text{Mikro işlemci komut sayısı} = 2^{\text{veri hattı sayısı}}$$

Denetim yolu mikro işlemcinin bellek ve arabirim tümdevreleri ile bilgi alış verişinde kullandığı eşleme işaretleri ile zamanlama ve kesme işaretlerinden oluşur. Veri ve adres yolunda olduğu gibi tüm hatlarının görevleri ve yönleri aynı değildir. Her hattın görevi farklıdır. Genellikle zamanlama amaçlı olanlar MİB tarafından üretilirler. Bunlardan en önemlileri saat (clock), yazma (write), okuma (read) hatlarıdır. Bunlardan oku hattı tek yönlü bir hattır ve bellekten okuma işlemi sırasında mikro işlemci tarafından üretilir. Yazma hattı da aynı yönlüdür ve yazma sırasında mikro işlemci tarafından yazma işlemi sırasında üretilir. Kesme (interrupt) hattı çevre birimleri veya programcı tarafından üretilir. Bu işaret etkin olduğunda işlemci normal program akışını keser ve özel bir program işletir. Denetim yolundaki hatların sayısı ve çalışma şekilleri mikro işlemci üreticisine göre çok farklılık gösterir.



## Giriş/Çıkış Aygıtları

Giriş/çıkış cihazları veya diğer adıyla bilgisayar çevre elemanları MİB ile gerçek dünya arasındaki iletişimi sağlayan birimlerdir. Bu birimler olmasaydı büyük bir olasılıkla bilgisayarlar kimse tarafından kullanılmazdı. Üç çeşit giriş/çıkış cihazı vardır. Bunlar, veri saklama cihazları, insan ile iletişimi sağlayan cihazlar, denetim ve gösterge cihazlarıdır.

### Veri Saklama Aygıtları

Manyetik veri saklama cihazları bellek teknolojisi arenasında RAM ve ROM gibi yarıiletken belleklerle birlikte anılırlar. Fakat aslında yapı olarak çok farklıdır. Bu cihazlar çok geniş kapasiteye sahiptirler, fakat mekanik olduklarından güvenilirlikleri düşüktür. Pazar gün geçtikçe büyümesine rağmen performanslarında fazla gelişme olmamıştır. RAM de olduğu gibi MİB'e yakın bağlanamaz, arabirim kullanmak zorundadır ki bu da veri iletimini yavaşlatır. Boyutlarının sürekli büyümesine rağmen teknolojisi bu ölçüde gelişmemektedir. RAM ve ROM gibi yarıiletken belleklere sığmayan programların saklanmasında kullanılır. Bu cihazlardan MİB doğrudan program işletemez, ancak parçalar halinde RAM belleğe taşınır ve işlem bittikten sonra tekrar bu cihazlara kaydedilir. Bu cihazlar (online) sürekli hatta veya arşiv amaçlı olmak üzere iki türlü kullanılabilirler. Genellikle manyetik ortamda veri saklayan hard diskler sürekli hatta çalışırlar. CD-ROM adını verdiğimiz optik ortamda veriyi saklayan bellekler ise arşiv amaçlı kullanılırlar. Bunların dışında teyp kasetleri de veri saklama amaçlı kullanılırlar.

### İnsan İle İletişimi Sağlayan Cihazlar

Bilgisayar ile insanı kaynaştırmak insan ile makineyi anlaştıran cihazların çokluğuna bağlıdır. En çok kullanılan arabirim video display terminal (VDT) olarak adlandırılan klavye ve katot ışınlı tüpten oluşan ekrandır. Günümüzde görüntüleme cihazları oldukça çeşitlenmiştir. LCD ekranların yüksek renk kalitesi ve kapladıkları az alan dolayısıyla kullanımı yüksek fiyatlarına rağmen artmaktadır. Diğer cihazlar ise yazıcı, çizici, mikrofon, hoparlör, fare, joystick, light pen'dir.

### Denetim Ve Monitör Cihazları

Mikroişlemciler endüstride de yaygın olarak kullanılmaktadır. Özellikle seri üretim bantlarında üretimin hatasız ve hızlı olması için birçok işi robotlar yapmaktadır. Robotları ise mikroişlemciler denetlemektedir. Mikroişlemci denetleyeceği değişkenin değerini öğrenmek için sensör kullanır. Değişkeni denetlemek için motor, röle gibi elemanları kullanır. Sensörler genellikle basınç, sıcaklık, ışık, hareketi gibi

ölçtükleri parametreleri elektrik işaretine dönüştürürler. Bu analog elektrik işareti ADC kullanılarak sayısallaştırılır. Bu bilgi işlendikten sonra tekrar kontrol elemanına gönderilmeden DAC kullanılarak analog işarete dönüştürülür. Endüstride yaygın olarak mikroişlemci yerine mikrodenetleyici kullanılmaktadır.

## Yazılım

İlk yıllarda donanım yazılımdan daha önemliydi ve maliyeti yüksekti. Son yıllarda ise yazılım daha önemli hale gelmiştir. Maliyet olarak donanımın üzerine çıkmıştır. Şekil-1.4'te yazılım türleri basitleştirilmiş olarak gösterilmiştir. Yazılım üç seviyeden oluşur: en dışta uygulama programı, onun içinde işletim sistemi programı, en içte giriş/çıkış alt programları.

En içteki giriş/çıkış alt programları doğrudan donanım ile üst grup yazılımların bağlantısını sağlar. Alt programlar klavyeden karakter okur, göstergede karakter görüntüler, hard diskten bir grup program parçasını RAM belleğe getirmek gibi işletim sisteminin gereksinim duyduğu altprogramlardır. Bu alt programlar donanımı tasarlayan kişiler tarafından yazılır ve kalıcı tür bellekler içerisine yazılır. IBM PC'lerde bu belleklere BIOS (Basic Input/Output System) adı verilir. BIOS belleği programcı tarafından değiştirilemez. Fakat saat, tarih, şifre gibi sonradan değiştirilmesi gerekli değişkenlerin saklandığı bir pilli RAM veya EEPROM bellek anakart üzerine yerleştirilir.



Şekil-1.4 Yazılımın katmanları.

Programlayıcının donanım ile daha yakın ilişki kurabilmesi MİB'in yazaçlarının başlangıç değerleri giriş/çıkış alt programları tarafından sistem RAM'ine kaydedilir.

## Mikrodenetleyiciler 8051 Uygulamaları

Programcı bu RAM'in içeriğini değiştirerek kendine uygun şekilde donanımına ulaşabilir. BIOS giriş/çıkış alt programlarının yanı sıra sistemi başlatma programını da içerir. ROM'da kayıtlı olduğu için başlatma programı bağlı sistemlerin test edilmesi ve belleğin başlangıç koşullarına ayarlanması gibi sabit işlemleri içerebilir. Bunun dışında bootstrap loader alt programı diskin birinci izini okur ve program parçasını RAM'e kaydeder. Bu program işletme sisteminin temelini oluşturur.

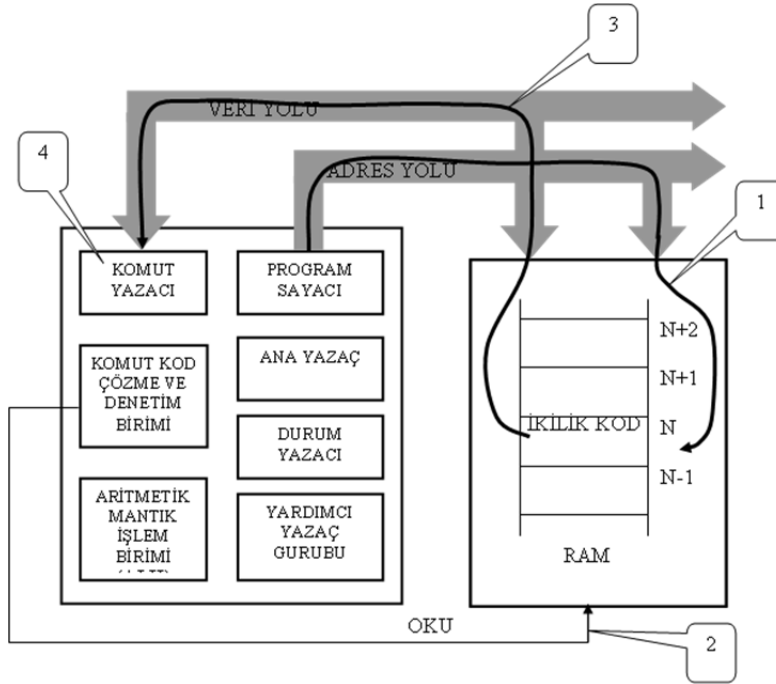
İşletim sistemi birçok programın toplamından oluşur ve bilgisayar ile birlikte gelir. Uygulama programlarının kullanımı için komut dili ve yardımcı programlar içerir. İşletim sistemi giriş/çıkış altprogramlarının bir veya bir kaçını birleştirerek kendine komut oluşturur. Dizin yaratma, kopyalama, silme, yeniden adlandırma gibi komutlar buna birer örnektir. Bu komutlar uygulama programları tarafından komut olarak kullanılır. Kullanıcı programı yazan bir programcı bilgisayarın mikroişlemcisinin assembler dilini bilmeden giriş/çıkış altprogramlarını rahatlıkla kullanabilir. Uygulama yazılımları bilgisayara iş yaptırmak için yazılmış programlardır. Kelime işlemcisi, programlama dilleri çizim programları, ses ve müzik programları gibi.

## Bilgisayarın İşleyişi

Bilgisayara güç uygulandığında veya reset tuşuna basıldığında mikroişlemci adres yoluna reset vektörü adı verilen adres bilgisini yazar. Daha sonra okuma işaretini üretmek bellekten bu adreste yer alan komutu komut yazacına aktarır. Reset vektörü üretici firma tarafından BIOS ROM'un ilk veya son adresi olarak belirlenir. Birinci komut okunduktan sonra program sayacı bir artırılır ve bir sonraki komut veya veri okunacak ilk bilgidir. Programın akışı işletilen komutlara göre değişecektir. Mikroişlemcinin bellekten komut okumasına komut getirme saykılı adı verilir. Şekil1.5'te komut getirme saykılına aşamaları gösterilmiştir. Komut getirme saykılındaki olaylar sırası ile şöyle gelişir;

1. Program sayacını içeriği adres yoluna aktarılır,
2. Denetim yolu hatlarından okuma işareti etkin yapılır,
3. İkilik bilgi RAM veya ROM'dan veri yoluna aktarılır,
4. Veri yolundaki ikilik bilgi MİB içerisindeki komut yazacına alınır,
5. Program sayacı bir sonraki komut getirme işlemi için artırılır.

Getirilen ikilik bilginin komut kodu olup olmadığı kod çözme ve denetim birimi tarafından belirlenir. Eğer bir komutun kodu ise bu kod geldiğinde yapılması gereken işlemler mikroişlemci içerisinde yazılıdır. Bu bilgiler mikroişlemci üretilirken yazılır ve sonradan değiştirilemez. Komutun kodunun çözülüp gerekli denetim işaretlerinin üretilmesi aşamasına komut yürütme saykılı adı verilir. Komut yürütme saykılında komut yazacına alınan komutun kodu çözülerek üretilmesi gereken iç denetim işaretleri üretilir ve zamanında ilgili iç birime gönderilir. Örneğin eğer işlem kodu bir toplama işlemi olduğunu söylüyorsa işlem için sayılar yazaçlardan aritmetik işlem birimine alınır. Toplama en düşük değerlikli bittten başlar ve en yüksek değerlikli bite doğru yapılır. Zamanlama işlemleri komut kod çözme ve denetim birimi tarafından yapılır. Komut yürütme işlemleri komut getirme işlemleri gibi sabit süreli değildirler. Yapılan işlemin karmaşıklığına göre süre uzayabilir. Anlamlı bir işlem yaptırmak için sıralanmış komut kümesine program ya da yazılım adı verilir. Amaçlanan işlemin hatasız bir şekilde gerçekleşmesi yazılan programın güçlüğünü gösterir. Güçlü programlar girilecek tüm girdileri göz önüne alınarak yazılır. Çoğunlukla programı işlettiğimizde bazı verilerde hatalı sonuçlar elde ettiğimizde bilgisayarı suçlar ve bilgisayar hata yaptı deriz. Aslında bilgisayar değil yazdığımız program hatalı veya eksiktir. Fakat donanım hatası var diyebiliriz. Bunu söylemek içinde yazılımımızın doğruluğundan emin olmalıyız.

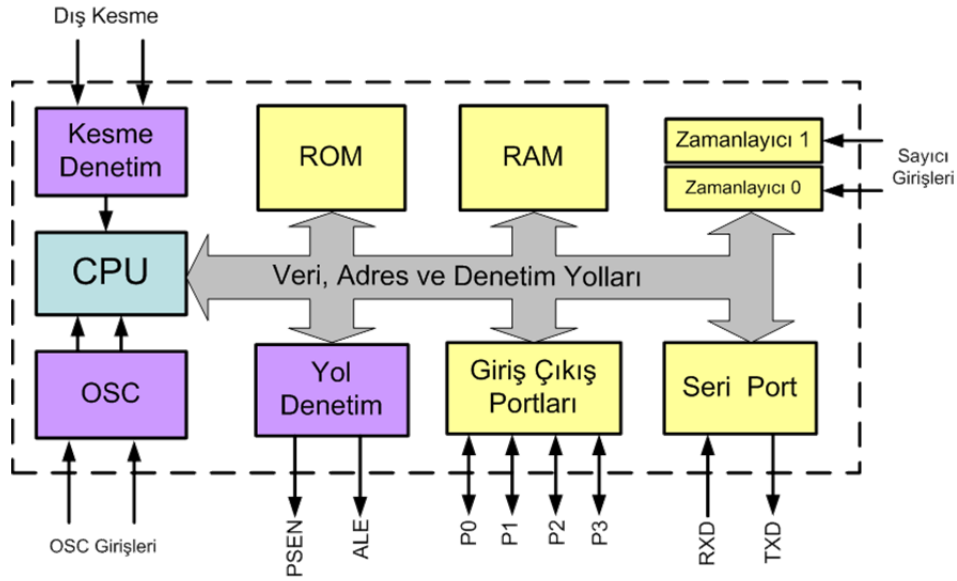


Şekil-1.5 Komut getirme saykılında işlem sırası.

## Mikrodenetleyiciler 8051 Uygulamaları

## Mikrodenetleyiciler

İlk yıllarda endüstride ve bilgisayarda aynı mikroişlemciler kullanıldı. Daha sonraki yıllarda bilgisayarın daha hızlı ve daha çok işlem yapan işlemcilere gereksinim duyması, endüstrinin ise yavaş fakat içerisinde sıkça kullanılan yardımcı birimleri içeren mikroişlemciler istemesi nedeniyle endüstrinin gereksinimi için yeni arayışlara girildi. İlk denemeyi Intel şirketi 1976 yılında 8748 mikroişlemcisi ile yaptı. Bu tümdevre içerisinde mikroişlemciye ek olarak 1Kbayt EPROM bellek, 64 Bayt RAM, 27 I/O bacağı ve 8 bit zamanlayıcı yer almaktaydı. Bu işlemciyi elde etmek için yaklaşık 17,000 transistör kullanılmıştır. 8748 kontrol uygulamalarının değişmez elemanı oldu ve birçok sistemin daha basit ve akıllı şekilde tasarlanmasını olanaklı kıldı. Özellikle otomatik çamaşır makineleri, trafik ışıkları, otomobil ateşleme sistemleri gibi endüstriyel cihazlarda yoğun olarak kullanıldı. Pazardan memnun kalan Intel geliştirilmiş endüstri mikroişlemcisini 1980 yılında MCS-51 ailesini olarak dünyaya tanıttı. Bu ailenin ilk elemanı 8051 olarak adlandırıldı. Bu tümdevre 60000 transistörden oluşuyordu ve içerisinde mikroişlemci, 32 giriş/çıkış hattı, 2 adet 16 bit zamanlayıcı, seri port, 4 Kbayt ROM bellek, 128 bayt RAM bellek barındırıyordu. Şekil-1.6'da ilk üretilen 8051 Mikrodenetleyicisini blok şeması gösterilmiştir.



Şekil-1.6 Mikrodenetleyicinin blok şeması.

Intel'den üretim izni alan yaklaşık 20'den fazla firma MCS-51 ailesini geliştirdiler, aynı yonga içerisinde kendi çalıştıkları alana uygun birimler eklediler. Günümüzün en

çok kullanılan mikrodenetleyicisi MCS-51 ailesidir. Motorola ise 68HC11 serisi mikrodenetleyicileri üretti, özellikleri 8051'e benzeyen bu mikrodenetleyici daha önce 6802 işlemcisini kullananlar tarafından tercih edildi. Microchip firması daha küçük uygulamalara dönük olarak daha az bacağı olan ve dış bellek bağlanmasına izin vermeyen mimaride 12XX, 14XX ve 16XX serisinde PIC (Peripheral Interface Controller) mikrodenetleyicilerini üretti. Zilog firması da PIC'e benzer mimaride Z8 serisi mikrodenetleyiciler üretti. Thomson firması ise tek yongada ADC ve DAC içeren ST62XX serisi mikrodenetleyiciler üreterek daha başka bir boyut getirdi. Günümüzde birçok firma mikrodenetleyiciye program belleği, veri belleği, MIB, seri kanal, değişik sayıda I/O portuna ek olarak denetim uygulamalarında sıkça kullanılan ADC, DAC, sayıcı/zamanlayıcı, SPI, I2C gibi ek birimler eklemektedirler. Bu devrelerin tek bir tümdevrede (chip) birleştirilmesi tüketilen enerjiyi azaltırken devrelerin uyum probleminden doğacak hataları ortadan kaldırmaktadır.

Son yıllarda tasarımcılar Analog Devices, Atmel, AMD, Maxim Dallas, Hynix, Infineon, Intel, ISSI, Micronas, Oki, Philips, SST, Winbond, Silicon Laboratories, Hyundai, ST Microelectronics, Samsung gibi 20'den fazla firmanın ürettiği 8051 çekirdekli ve bilgisayardan doğrudan programlanabilen flash belleği olan mikrodenetleyicileri kullanmayı tercih etmektedirler.

Mikroişlemci ile mikrodenetleyicileri komut kümeleri bakımından karşılaştırsak, kullanım amaçları farklı olduğu için komut kümeleri de farklıdır. Mikroişlemci daha güçlü uygulamalarda kullanılacağı için hız ve kelime uzunluğu büyük olarak tasarlanmıştır. Bunun sonucu olarak büyük veri gruplarını işleyebilecek şekilde bayt, çift bayt uzunluğundaki verileri bir defada işleyebilecek komutlara sahiptir. Fakat mikrodenetleyicilerde bit işlem yapan komutlar daha önemlidir. Birçok komutu bit adresleme kipinde kullanılır.

---

## Mikrodenetleyiciler 8051 Uygulamaları

## Sorular

1. Yaygın olarak kullanılan ilk mikroişlemci hangisidir? Hangi firma tarafından üretilmiştir?
2. Yarıiletken bellek türlerini yazın hangisinde veri kalıcı değildir?
3. Komut getirme evresinde adres ve veri yolunun içeriği nedir?
4. 14 bit program sayacı olan mikroişlemcinin belleme kapasitesi ne kadardır?
5. Sürekli hatta veri saklama cihazı ile arşiv türü veri saklama cihazları arasında ne gibi farklılıklar vardır açıklayın?
6. “Nonvolatile RAM” belleğin özelliği nedir?
7. Port adresli giriş/çıkış yöntemi ile bellek adresli giriş/çıkış yöntemi arasındaki farklar nelerdir?
8. En yaygın kullanılan çevre birimleri nelerdir?
9. Seri ve paralel portların farkları nelerdir?
10. Kesme ne demektir? Neden gereklidir?
11. Mikroişlemci ile mikrodenetleyicinin farkı nedir?
12. Mikrobilgisayarın blok şemasını çizip kısımlarının görevlerini açıklayınız.







# ***MCS-51 Ailesi Mikrodenetleyiciler***



## ***Giriş***

8051 mikrodenetleyicileri ilk olarak INTEL tarafından 1980 yılında üretilmiştir. MCS-51 ailesi mikrodenetleyiciler olarak ta adlandırılan 8051'in kullanım haklarını INTEL 30'dan fazla firmaya satmıştır. 1990'lı yıllardan başlayarak Intel 8051 üretimini kademeli olarak durdurmasına rağmen kullanım hakkını alan 30'dan fazla firma çekirdek yapısına ve komut setine sadık kalarak gününüz ihtiyaçlarına yanıt verecek şekilde geliştirerek günümüze kadar güncel kalmasını sağlamıştır. Ayrıca KEIL, IAR, NOHAU, TASKING, RAISONANCE gibi birçok firma ise geniş bir donanım ve yazılım geliştirme araçları desteği sunmaktadır. Bunun sonucu olarak 8051 ailesi, 1980'lerden bugüne bir endüstri standardı olmuştur.

Pek çok üretici firma, orijinal 8051'e çeşitli ek özellikler katarak türev ürünler geliştirmiştir. Çok değişik 8051 türev ürünler bulunmasına rağmen komut seti ve çekirdek yapı olarak bütün ürünler uyumludur. 8051 çekirdek mimarisi en basitten en karmaşığına kadar her türlü endüstriyel otomasyon uygulamalarında kullanıma uygundur. Piyasaya ilk sunuldukları tarihte 12 MHz'lik modelleri bir saniyede 1

---

## **Mikrodenetleyiciler 8051 Uygulamaları**

milyon (1 MIPS, Mega Instruction Per Second) komut yürütüyorken yeni türevlerde 24 MIPS, 50 MIPS ve 100 MIPS'lik hızlara ulaşılmıştır.

8051 Ailesi ürünlerin teknik ve ticari avantajlarının yanı sıra eğitimde de kullanmanın birçok avantajı vardır. 8051 hakkında birçok ders kitabı, teknik doküman, yazılım ve donanım gereçleri, pek çok İnternet Web Sayfası mevcuttur. Kitabın sonunda kaynaklar kısmında bu dokümanlar verilmiştir. Diğer bir kolaylığı ise 8051 mikrodeneleyicisini birçok elektronik parça satan firmadan ucuz olarak temin edilebilmesidir. Mikroşlemci-mikrodeneleyici derslerinde her hangi bir mimari yapının öğretilmesi halinde bir başka mimari yapıya sahip aileye uyum sağlamak çok fazla zaman almayacaktır. Ancak olanaklar elverdiğince piyasada yaygın kullanılan bir mikrodeneleyici ailesinin seçilmiş olması öğrencilerin iş hayatına uyumunu hızlandıracaktır.

### **MCS-51 Ailesi Mikrodeneleyiciler**

MCS-51 ailesi mikrodeneleyicilerin ilk üyesi 8051'dir. Üzerinde 4 Kbayt bir defa programlanabilir (OTP) tipi ROM program belleği yer alır. Bu ilk ürünün ismi çoğu zaman ailenin genel adı olarak söylenir. Birçok kaynak ya da kişi "MCS-51 ailesi mikrodeneleyiciler" yerine "8051 ailesi" kısaltmasını kullanmaktadır. Ailenin ilk üyesinin özelliklerini şöyle sıralayabiliriz.

- 8 bit Mikroşlemci,
- Mantıksal işlemler yapabilen işlemci,
- 4x8 biçiminde düzenlenmiş, 32 adet Giriş/Çıkış hattı,
- 128 bayt RAM bellek,
- ROM veya bazı modellerinde EPROM, FLASH bellek
- 128 adet bit adreslenebilen bellek hücresi,
- ACC ve B yazacına ek olarak 8 adet yazaç (R0, R1, ..., R7),
- Programlanabilir çift yönlü (full-duplex) seri port,
- Üzerinde var olan iç belleklere ek olarak dış bellekler ekleyebilme özelliği,
- İki adet 16 Bitlik zamanlayıcı/sayıcı,
- İki öncelik düzeyi olan beş adet kesme kaynağı,

### **Mikrodeneleyiciler 8051 Uygulamaları**

- Tümdevre üzerinde osilatör ve saat işaretleri oluşturma devresi.

8051'in program belleği ROM belleğin içeriğinin değiştirilememesi bazı kullanıcıların işini zorlaştırmıştır. Bu sorun program belleği EPROM olan 8751 üretilerek giderilmiştir. Bu ürünün içerisinde 8051'den farklı olarak 4 kbayt EPROM bellek kullanılmıştır. 8751'in iç program belleği özel programlayıcı ile programlanabilir, programı değiştirmek istediğimizde EPROM silici ultraviyole ışık ile 15 dakikada silinip tekrar programlanabilir. Programlama silme aygıtlarının pahalı olması nedeniyle birçok amatör kullanıcı 8751'i kullanamamıştır. Amatör kullanıcılar için fiyatı ucuz iç program belleği olmayan fakat diğer özellikleri 8051'ile aynı olan 8031 üretilmiştir. Daha sonraki yıllarda program ve veri belleklerinin gereksinimi karşılamaması nedeniyle bellek miktarları iki katına çıkarılmıştır. 8051'in ROM program belleği 8 Kbayt'a ve RAM veri belleği 256 Bayt çıkarılmış ve 8052, 8751 EPROM program belleği 8 Kbayt'a ve RAM veri belleği 256 Bayt çıkarılmış ve 8752, 8031'in ise veri belleği 256 bayta çıkarılmış ve 8032 olarak adlandırılmıştır. Çizelge–2.1'de 8051 ailesi mikrodenetleyicilerin en temel üyelerinin özellikleri gösterilmiştir.

	8051	8052	8751	8752	8031	8032
On-Chip ROM	4K	8K	4K	8 K	0 K	0 K
RAM (Bayt)	128	256	128	256	128	256
Zamanlayıcı	2	3	2	3	2	3
I/O Portları	4	4	4	4	2	2
Seri Port	1	1	1	1	1	1
Kesme Kaynağı	5	6	5	6	5	6

Çizelge–2,1 MCS–51 ailesinin en temel elemanları ve içerdikleri çevrebirimleri.

1990'lı yıllardan sonra tümdevre (on chip) üzerinde yer alan program belleği önce EEPROM bellekle değiştirilmiş böylece elektrik ile silinebilir hale gelmiştir. Daha sonraki yıllarda daha ucuz üretim teknolojisine sahip FLASH EEPROM üretilmiş ve program bellekleri bu tür bellek kullanılarak üretilmeye başlanmıştır. Intel ve bazı firmalar bu tür belleğe sahip MCS ailesi ürünleri 89FXX olarak adlandırmışlardır. FLASH EEPROM belleklerin yama silme adedi 1000 defadır, bu adet aşıldıktan sonra en son yapılan program bellekte kalır yeni yazma yapılamaz. Günümüzde üretilen 8051 türevlerini programlamak için özel programlayıcıya gereksinim yoktur. Üretici firmaya bağlı olarak bilgisayarın seri, paralel ve USB portunu kullanarak yapılacak basit bir devre ile hızlı bir şekilde programlama yapılabilmektedir. Hatta adım

modunda çalıştırılarak yazaçların içerikleri okunarak program hataları belirlenebilmektedir. 8051'in satış fiyatları içerisinde yer alan birime bağlı olarak bir € ile 50 € aralığında değişmektedir.

1985 yılından sonra Intel firması MCS-51 ailesi mikrodenetleyicilerin çekirdek yapısı ve komut setinin kullanım hakkını diğer firmalara satmaya başlamış ve günümüzde birçok firma tarafından 8051 çekirdeği ve komut kümesi kullanılmaktadır. Bu firmalar kendi kullanım alanlarına göre uygulamaya özel, her tüketiciye uygun 8051 çeşitleri üretmişlerdir. Bu özelliklerin bazılarını şöyle yazabiliriz.

- Programlanabilir sayıcı dizisi (PCA),
- Analog sayısal ve Sayısal analog dönüştürücüler,
- SPI, USB, I2C ve Microwire gibi seri haberleşme kanalları,
- CAN denetleyici,
- PWM,
- LCD sürücü birimi,
- RAM, EEPROM veri bellekleri.

Özellik	8051	PENTIUM	Açıklama
Hızı	12- 60 Mhz	1– 4 GHZ.	
Adres Yolu	16 bit	32 bit	8051 $2^{16}$ , veya 64 Kbyte bellek. Pentium ise $2^{32}$ , veya 4 GBayt belleği adresleyebilir.
Veri Yolu	8 bit	64 bit	Pentium bir defada daha fazla veriyi aktarır.
ALU kelime genişliği	8 bit	32 bit	Pentium bir defa da daha büyük sayıları toplayabilir.
Uygulama Alanı	Ev gereçleri, Endüstri	Bilgisayarlar	
Güç Tüketimi	250 mW	30–60 W	Pentium çok ısınır soğutma ister.
Maliyeti	1–2 €	100–200 €	

Çizelge–2.2 8051 ile Pentium serisi işlemcilerin karşılaştırılması.

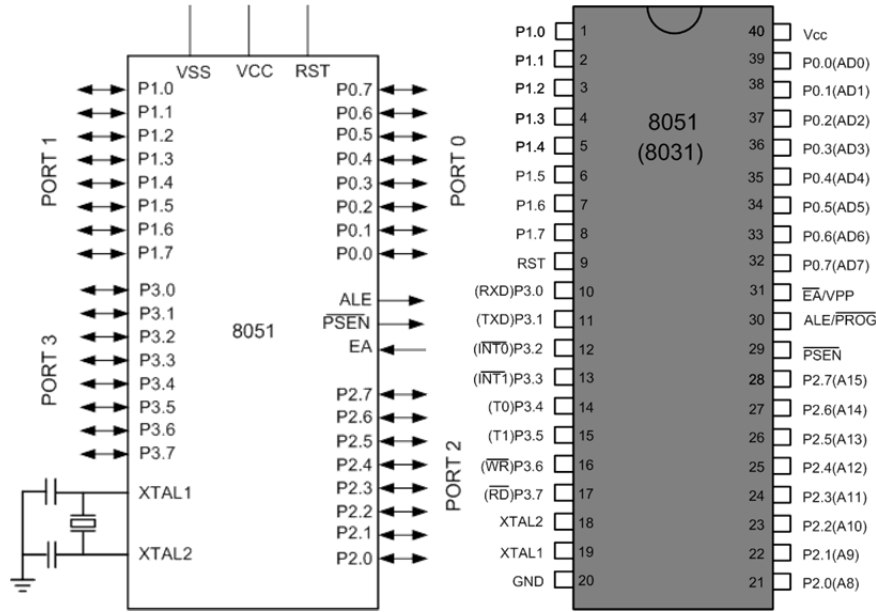
## Mikrodenetleyiciler 8051 Uygulamaları

<b>8051 Üreten firmaların Bazıları<sup>2</sup></b>	
<i>Acer Labs</i>	<i>Megawin</i>
<i>Actel</i>	<i>Mentor Graphics</i>
<i>Aeroflex UTM</i>	<i>Micronas</i>
<i>Altium</i>	<i>MXIC</i>
<i>Analog Devices</i>	<i>Myson Technology</i>
<i>ASIX</i>	<i>Nordic Semiconductor</i>
<i>Atmel</i>	<i>NXP</i>
<i>Cast</i>	<i>OKI</i>
<i>Chipcon</i>	<i>Oregano Systems RadioPulse</i>
<i>CML Microcircuits</i>	<i>Ramtron</i>
<i>Cybernetic</i>	<i>Sanyo</i>
<i>CybraTech</i>	<i>Sharp</i>
<i>Cypress</i>	<i>Silicon Laboratories</i>
<i>Daewoo</i>	<i>Siliconians</i>
<i>Digital Core Design</i>	<i>SMSC</i>
<i>Dolphin</i>	<i>SST</i>
<i>Domosys</i>	<i>STMicroelectronics</i>
<i>Goal Semiconductor</i>	<i>Teridian</i>
<i>Handshake Solutions</i>	<i>Texas Instruments</i>
<i>Honeywell</i>	<i>Tezzaron</i>
<i>Hynix</i>	<i>Triscend</i>
<i>ISSI</i>	<i>Vitesse</i>
<i>Infineon</i>	<i>Winbond</i>
<i>Intel</i>	<i>Zylogic</i>
<i>Maxim/Dallas</i>	

<sup>2</sup> Kaynak [www.keil.com](http://www.keil.com) ve [www.iar.com](http://www.iar.com)

8051 mikrodenetleyici olduğu için denetim uygulamalarında kullanılmak üzere geliştirilmiştir. Diğer yandan 8051 ile aynı yıllarda üretime başlayan 8088 mikroişlemcisi bilgisayarlarda kullanılmak üzere geliştirilmiş ve bugün Pentium ailesi olarak devam etmektedir. 8051 ile Pentium ailesi mikroişlemcilerin özelliklerinin karşılaştırılması çizelge-2.2’de gösterilmiştir.

Şekil-2.1’de 8051’in mantık simgesi ve bacak bağlantısı verilmiştir. 8051’in 40 bacağından 32 tanesi I/O hattıdır. Geriye kalan 8 bacak ise besleme ve denetim hatlarıdır. 32 I/O hattı 4 adet 8 hatlı port olarak adlandırılır. Bu portların I/O hattı dışında ikincil görevleri vardır. Eğer dış veri ve program belleği tasarlanan sistemde gerekli değilse bu hatlar serbest I/O hattı gibi kullanılabilirler.



Şekil-2.1 8051'in mantık simgesi ve bacak bağlantısı.

## 8051'in Çekirdek Yapısı

8051 MİB'inin içerisinde 8 bit ALU (Arithmetic Logic Unit), 8 bit A akümülatörü ve 8 bit B akümülatörü yer alır. Bunların dışında her MİB'de bulunması gereken komut yazacı, kod çözme birimi, port tutucuları ve sürücüler, zamanlama ve kesme öncelikleme birimleri yer alır. Bu yapının tanınması program yazımı sırasında mikrodenetleyicinin daha etkin kullanılmasını sağlar. Şekil-2.2'de 8051'in basitleştirilmiş yazaç yapısı gösterilmiştir.

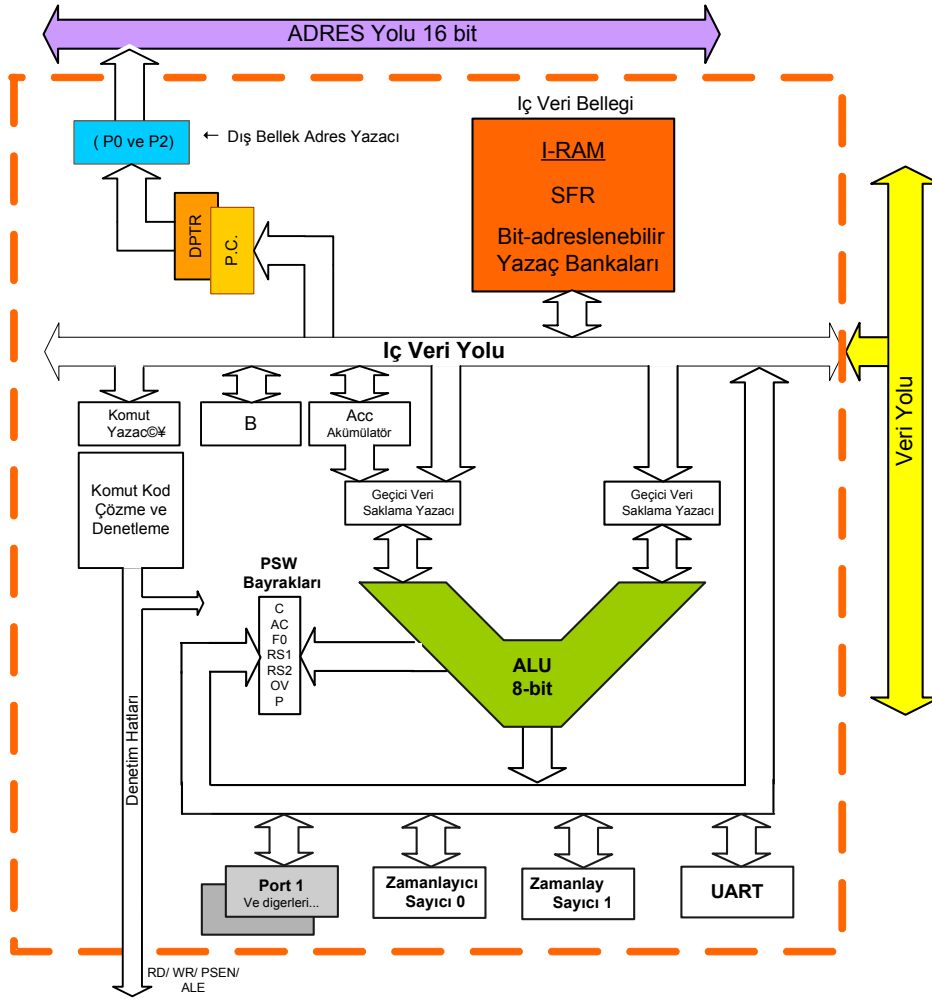
## Mikrodenetleyiciler 8051 Uygulamaları



8051'de mikroişlemciden farklı olarak iç RAM bellek birimi ile zamanlayıcı ve seri port denetim birimleride çekirdek yapı içerisine alınmıştır. Diğer birçok mikroişlemciden farklı olarak A ve B yazaçlarının da adresleri vardır.

## Denetim Hatları

8051'in denetim hatları osilatör girişleri, reset girişi ve dış bellek kullanımını sağlayan PSEN, ALE, EA hatlarından oluşmaktadır. Dış bellek kullanılmadığında bazı denetim hatları işlevsiz kalır, bu sebeple yeni türev 8051'lerde bu hatlar yer almayabilir. Bazılarında ise programlama sırasında yapılan seçimle etkin yapılabilir.



Şekil-2.2 8051'in iç yazaç yapısı

**PSEN**

Dış program belleğinden okuma yapan işaret çıkışıdır, program belleğinin (EPROM'un) OE girişine bağlanır. PSEN çıkışı dış bellekten program işletilirken komutun okunması sırasında düşük seviyeli vuru verir. Bu işaret etkin olduğunda adres yolu tarafından seçilmiş bellek satırının içeriği veri yoluna aktarılır. İç bellekten program işletildiğinde etkin olmaz.

**ALE**

Bu işaret PORT 0 adres ve veri yolu olarak kullanıldığında veri bilgisi ile adres bilgisinin demultiplex edilmesini sağlar. Dış bellekten işlem yapma saykılının ilk yarısında adres bilgisi PORT 0'a yazılır ve bu periyodun sonunda ALE işareti etkin yapılarak Port 0'ın içeriği D tutucu çıkışına aktarılır. Saykılın ikinci yarısında ALE işareti etkin olmaz ve Port 0 belleğin veri girişine bağlı kalır. Veri aktarımı bellekten işlem yapma saykılının ikinci yarısında gerçekleşir. ALE işareti on-chip osilatörün 1/6'sı hızında vuru üretir. 12 MHz ile çalışmada 2 MHz'lik bir vuru çıkışı verir. Bu kural, sadece MOVX komutu işletilirken geçerli değildir, bu komut için ALE işareti bir saykıl süresince sıfır düzeyinde kalır.

**EA (EXTERNAL ACCESS)**

EA (external access) girişi dış ya da iç program bellekleri arasında seçim yapmak için kullanılır. Bu girişe mantık 1 uygulandığında (3K3'lük bir direnç üzerinden +5V'a bağlanırsa) 8051 iç program belleğinden komut işletir. Mantık 0 uygulanırsa (GND hattına bağlanarak) ise dış program belleğinden komut işletir. 8031/8032'de iç program belleği bulunmadığından mutlaka şaseye bağlanmalıdır. İç program belleğinin programlama aşamasında bu girişe programlama gerilimi uygulanır. 8051'in EPROM'lu sürümü olan 8751'de programlama gerilimi 21 Volt, 87C52'de ise programlama gerilimi 12 voltur. Flash EPROM'lu sürümlerde ise 5 voltur.

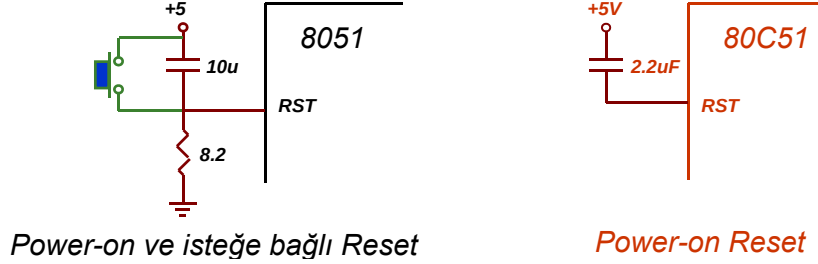
Açıklama: Son yıllarda üretilen 8051 türevlerinde yeteri kadar iç program belleği (64 Kbayt) ve iç veri belleği (4 Kbayt) bulunduğu için dış belleğe ihtiyaç duyulmamaktadır. Bu tür türevlerde (yaklaşık %90'ında) PSEN, ALE ve EA hatları bulunmaz. Bazı türevlerde ise programlama sırasında seçim yapılır. Eğer dış bellek bağlantısı seçilirse giriş/çıkış hatları bu iş için görevlendirilir.

**RESET**

8051 RST girişine en az 2 makine saykılı süresince yüksek seviye uygulandıktan sonra düşük seviye uygulanırsa başlangıç konumuna gelir (resetler). RST girişi bir basmalı anahtarla veya bir RC devre ile sadece güç verildiğinde (Power On Reset,

## Mikrodenetleyiciler 8051 Uygulamaları

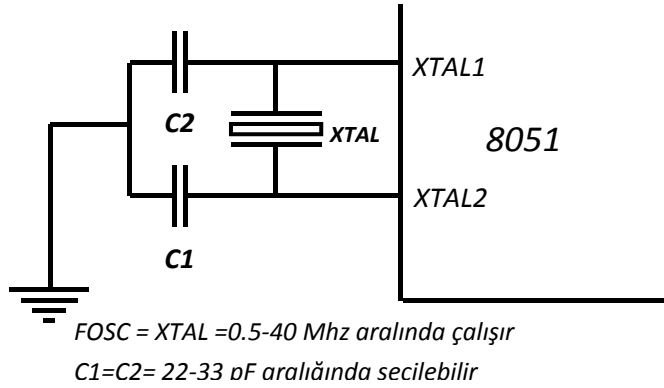
POR) etkin hale getirilir. Şekil-2.3'te her iki devre bağlantısı gösterilmiştir. 8051 resetlendiğinde SFR yazaçlarının içerikleri değişir ve başlangıç durumlarını alırlar. Bunlardan en önemlisi (PC) program sayacıdır. Reset sonrası program sayacının içeriği 0000H olur ve 8051 program işletmeye bu adresten başlar. Bu adrese reset vektörü denir. İç RAM belleğin içeriği reset işleminden etkilenmez. 8051'in çalışmaya başlayabilmesi için mutlaka POR'i alması gerekir, aksi halde çalışmaya başlamaz.



Şekil-2.3 8051 reset bağlantıları.

### Osilatör Girişleri

Şekil-2.4'te gösterildiği gibi 8051'in iki adet osilatör girişi vardır. Bu girişlere içerideki osilatöre kaynak olacak bir rezonans devresi bağlanır. Genellikle bir kristal bu görevi yerine getirir. MSC-51 ailesi mikrodenetleyicilerin yazılı kristal frekansları 12 MHz'dir, fakat 8051 her 12 osilatör saykılında 1 işlem gerçekler. Piyasaya ilk sunuldukları tarihte 12 MHz'lik modelleri bir saniyede 1 milyon (1 MIPS, Mega Instruction Per Second) komut yürütüyorken yeni türevlerde 24 MIPS, 50 MIPS ve 100 MIPS'lik hızlara ulaşılmıştır.



Şekil-2.4 8051'in osilatör girişleri ve kristal bağlantısı.

Mikroişlemcilerin bir komutu yürütme süresine makine saykılı denir. Her mikroişlemcinin makine saykılı kendine özgü formülü ile hesaplanır. Osilatör frekansı bilinen 8051'in makine saykılı aşağıda verilen formülle hesaplanabilir.

$$T_{Mak. Say.} = 12 / f_{OSC}$$

Bunun anlamı 8051 bir komutun işlenmesini 12 osilatör saykılında tamamlar. Bu durum birçok kullanıcı tarafından eleştirilmiştir. Motorola firmasının ürettiği 68HC12XX ailesi mikrodnetleyiciler 1 osilatör saykılında 1 komut işlerken PIC ailesi mikrodnetleyiciler 4 osilatör saykılında 1 komut işler. Yeni türev 8051'lerde istenirse programlama sırasında 6 osilatör, 4 osilatör veya 1 osilatör saykılarında bir komut işlemesi için seçim olanağı tanınmıştır. Tüm 8051 türevleri başlangıç olarak 12 osilatör saykılında 1 komut işleyecek şekilde programlanmıştır.

#### Örnek 1:

$f_{OSC} = 11.0592$  MHz olan 8051'in makina saykılını hesaplayın?

$$T_{MS.} = 12 / 11059200 \text{ Hz}$$

$$T_{MS.} = 1.085 \mu s$$

#### Örnek 2:

$f_{OSC} = 12$  MHz olan 8051'in makina saykılını hesaplayın?

$$T_{MS.} = 12 / 12000000 \text{ Hz}$$

$$T_{MS.} = 1 \mu s$$

#### Örnek 3:

$f_{OSC} = 16$  MHz olan 8051'in makina saykılını hesaplayın?

$$T_{MS.} = 12 / 16000000 \text{ Hz}$$

$$T_{MS.} = 0.75 \mu s$$

#### Örnek 3:

$f_{OSC} = 500$  KHz olan 8051'in makina saykılını hesaplayın?

## Mikrodnetleyiciler 8051 Uygulamaları

$$T_{MS} = 12 / 500000 \text{ Hz}$$

$$T_{MS} = 24 \mu\text{s}$$

### Gerilim Bağlantıları

8051 mikrodenetleyicisi tek bir 5 voltluk kaynaktan beslenir.  $V_{CC}$  40 numaralı bacak,  $V_{SS}$  ise 20 numaralı baktır. Kaynaktan çekilen akımın miktarı üretim teknolojisine ve güç denetim yazacı içerisinde yer alan kısık güç ve aylak kip bitlerinin değerlerine bağlıdır. Philips CMOS 8051 mikrodenetleyicilerin çektiği akım değerleri çizelge–2,3’de verilmiştir. Tasarım sırasında bu tabloda yer alan en büyük değeri dikkate alınız.

$I_{CC}$	$V_{CC}$	Min	Max	Birim
Aktif mod 16 MHZ	4.5 V-5.5 V	-	25	mA
Aktif mod 12 MHZ	4.5 V-5.5 V	-	20	mA
Aylak mod 16 MHZ	4.5 V-5.5 V	-	6.5	mA
Aylak mod 12 MHZ	4.5 V-5.5 V	-	5	mA
Kısık Güç 16 MHZ	4.5 V-5.5 V	-	75	$\mu\text{A}$
Kısık Güç 12 MHZ	4.5 V-5.5 V	-	50	$\mu\text{A}$

Çizelge–2.3 CMOS 8051’in güç tüketimi.<sup>3</sup>

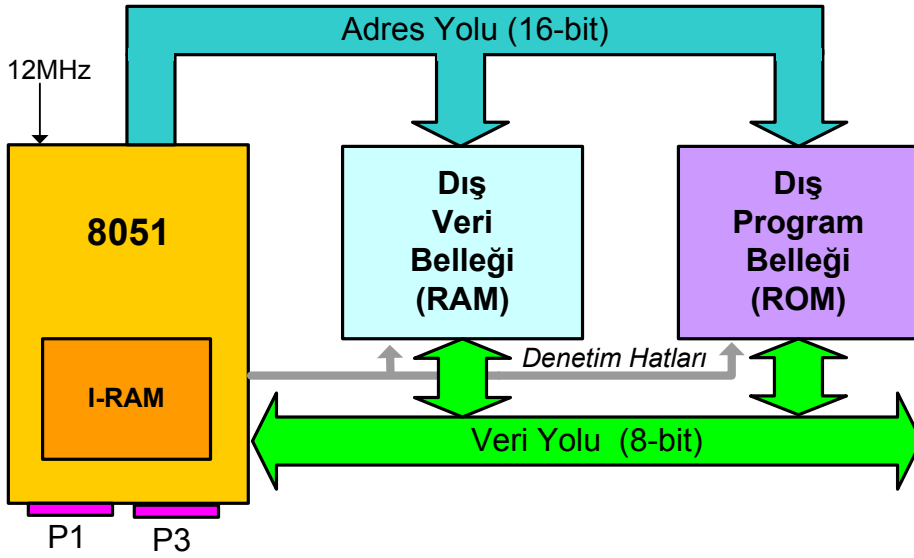
### Giriş Çıkış Hatları

8051’in 4 port şeklinde düzenlenmiş 32 adet giriş ve çıkış olarak kullanılabilen hattı vardır. Bu hatlara giriş çıkışın yanı sıra bazı ikincil görevler de verilmiştir. P1, P2, P0’ın hatlarının sürme kapasitesi 4 adet LS (low power Schottky) TTL yüküdür. P3’ün sürücü devresi güçlendirilmiş olduğu için 8 LS TTL yükü sürebilmektedir. Bu değerler yeni üretilen türev 8051’lerde farklıdır. Kullanılan 8051 türevinin veri yapraklarında en doğru değer elde edilebilir. Portlar SFR bölgesinde yazaç olarak yerleştirilmiştir. Portları 8’li olarak veya tek hat olarak okuma ve yazma yapabiliriz. Portların okunması ve yazması yapılarında küçük farklılıklar olmasına rağmen aynıdır. Reset sonrası portların çıkışları mantık “1” değerini alır.

<sup>3</sup> Kaynak: Philips 8051 data book.

## 8051'in Bellek Yapısı

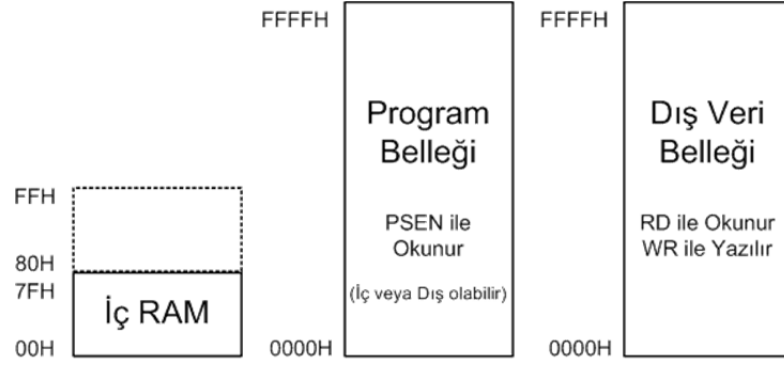
MCS-51 ailesinde iç RAM, dış RAM ve dış program belleği olmak üzere üç tür bellek yer alır. Bazı kaynaklarda RAM belleklere veri belleği, program belleğine ise kod belleği adı verilir. Program belleği 8051 ve 8751'de tümdevre üzerinde yer alır. İç program belleği yeterli olduğunda dış program belleği kullanmaya gerek yoktur. Dış bellek kullanıldığında giriş çıkış portlarının bir kısmı adres ve veri yolu olarak kullanılacağından giriş/çıkış hattı olarak kullanılamayacaktır. 8051'in bellek yapısı Şekil-2.5'te gösterilmiştir.



Şekil-2.5 8051'in bellek yapısı.

MCS-51 ailesi mikrodeneleyicilerde program belleği ile veri bellekleri ayrı adres alanlarına sahiptir. Her iki bellek türünden 64 Kbayt bellek bağlanabilir. Bu iki bellek satırı arasındaki ayırım kullanılan komut ve denetim hattı ile yapılır. 8051 komut okurken okuma işareti olarak PSEN hattını veri okurken RD hattını etkin yapılır. Veri yazarken ise WR işaretini etkin yapılır, program belleğine ise yazma yapılmaz. Devre tasarımı sırasında EPROM'un oku girişine PSEN hattı, RAM belleğin oku girişine RD hattı, yaz girişine ise WR hattı bağlanmalıdır. Bu ayırık bellek yapısına HARVARD mimarisi adı verilir. Bu mimaride MİB'e bağlanabilecek bellek kapasitesi iki katına çıkarken, denetim hattı sayısı ve komut sayısı artar. Şekil-2.6'da bellek alanları ve her bellek türünün kullandığı denetim hatları gösterilmiştir.

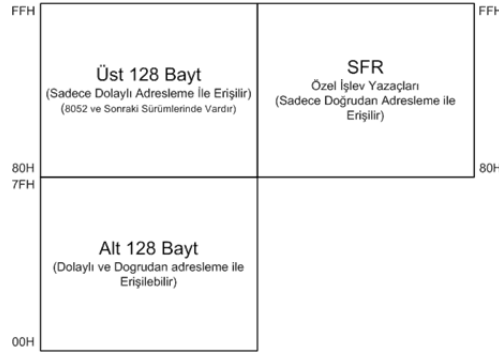
## Mikrodeneleyiciler 8051 Uygulamaları



Şekil - 2.6 8052'in adres alanları.

### İç RAM Belleğin Kullanımı

8051'de iç RAM olarak adlandırılan bölge MiB yazaçları, çevre birimini denetleyen yazaçlar, bit adreslenebilen bellek satırları ve işlem sonuçlarını geçici olarak saklamaya yarayan bellek satırlarından oluşur. İç RAM MiB yazaçları ve çevre birimlerini denetleyen yazaçların yer aldığı özel amaçlı yazaç bölgesi, yazaç bankaları, bit adreslenebilir ve genel kullanıma açık bellek satırlarının bulunduğu alt 128 bayt bölgesi ve sadece genel kullanım amaçlı bellek satırlarının bulunduğu üst 128 bayt bölgesi olmak üzere üç ana bölgeye ayrılmıştır. Şekil-2.7'da iç RAM belleğin kullanım bölgeleri gösterilmiştir.



Şekil-2.7 İç RAM belleğin kısımları.

### Alt 128 Bayt

Alt 128 baytın adres aralığı 00h-7Fh'dir ve bu kısma doğrudan ve dolaylı adresleme kiplerinin her ikisi ile de ulaşılabilir. Bu alanda yer alan 128 bayt farklı kullanım amaçlarına göre bölümlendirilmiştir. Bunlardan ilk 32 bayt yazaç bankaları olarak adlandırılır. Bunun hemen üzerinde yer alan 16 bayt bit adreslenebilen bölgedir. Geri kalan 80 bayt ise genel kullanıma açık bölgedir.



Bu bölgelerin kullanımı katı kurallara bağlı değildir, ayrılan görevde kullanmak programcıya kolaylık sağlar. Yazaç bankasını kullanmak veya kullanmamak programcıya bırakılmıştır. Reset sonrası bank 0 olarak adlandırılan yazaç gurubu etkin olur. Programcı seçmek istediği yazaç bankası numarasını yazarak etkin yazaç bankasını değiştirebilir. Kullanılmayan yazaç bankasını genel amaçlı RAM bellek olarak kullanılabilir. Bit adresli bölgenin bit adresleri 00h'den başlayarak 7Fh'ye kadar devam eder. Bu adresler bayt adreslere benzer fakat bit adreslemenin kullanıldığı komut ile bayt adreslemenin kullanıldığı komutlar farklı olduğu için herhangi bir kargaşaya sebep olmaz. Şekil-2.8'de alt 128 baytın kullanım alanları gösterilmiştir.

### Üst 128 Bayt

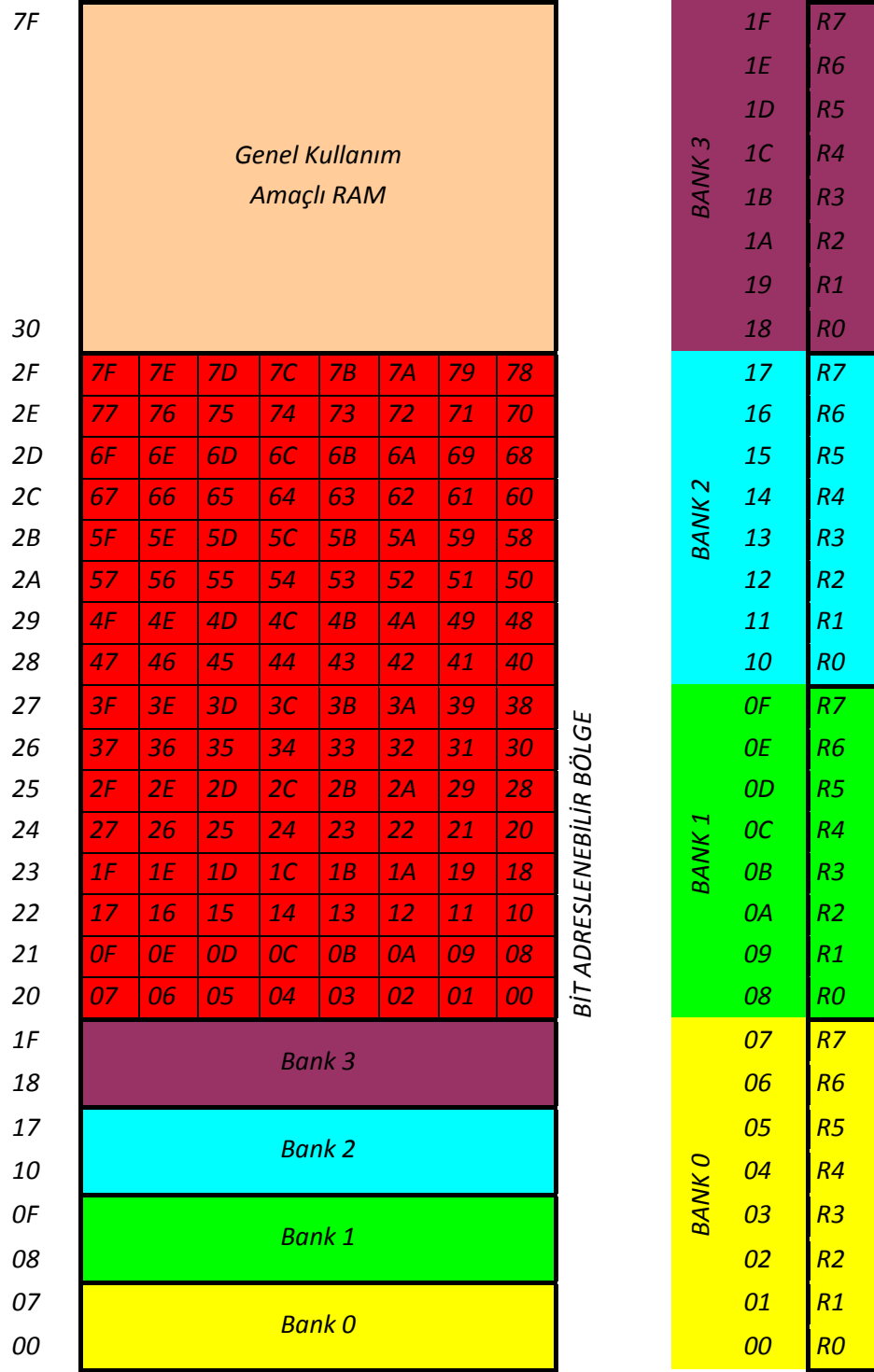
Üst 128 baytın adres aralığı 80h-FFh arasındır ve bu kısım sadece dolaylı adresleme kipi ile kullanılabilir. Bu kısım 8052 ve üst serilerde yer alır. Bu bellek alanı üst sürümlerde gerek duyulan daha büyük kapasiteli yığına bir çözüm olarak eklenmiştir. MCS-51 ailesi mikrodenetleyiciler yığının dış veri belleğinde oluşturulmasına izin vermez. Programcı başlangıçta yığının başlangıç adresini 80H ayarlayarak üst 128 baytı yığın olarak kullanabilir.

### Özel İşlev Yazaçları

Özel amaçlı yazaçlar (SFR) kısmının adres aralığı üst 128 bayta olduğu gibi 80H-FFH aralığıdır, aynı adresi iki birimin paylaşması sebebiyle karışıklığı engellemek için MCS-51 ailesi mikrodenetleyiciler bu bellek alanına sadece doğrudan adresleme kipi ile erişime izin vermiştir. Özel işlev yazaçları tümdevre üzerinde yer alan çevre birimleri ile merkezi işlem biriminde yer alması gereken ana, yardımcı ve denetleme yazaçlarından oluşmaktadır. Çevre birimlerine erişmek için bu yazaçlar kullanılır.

MCS-51 ailesi mikrodenetleyicilerde sonraki sürümleri düşünülerek bu bölge için 128 baytlık yer ayrılmıştır, fakat fiziksel olarak bunların çok azı ilk sürümlerinde kullanılmıştır. 8051'de bu yazaçların sayısı 21 adettir, 8052'de ise 26 adettir. 8051 içerisine çevre birimi eklendikçe Özel İşlev Yazaçlarının sayısı artacaktır. Şekil-2.10'da AT89S52'nin özel işlev yazaçlarının yerleşimi verilmiştir. Özel İşlev Yazaçlarının tümünün assemlelerde kullanılan simgeleri vardır, assembly programı yazarken simgeler kullanılabilir.

## Mikrodenetleyiciler 8051 Uygulamaları



Şekil-2.8 Alt 128 bayt RAM'in kullanım alanları.

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Şekil–2.9 SFR yazaçlarının bellek alanına yerleşimi.

Özel amaçlı yazaçların isimleri ve SFR bölgesine yerleşimleri şekil–2.9’da gösterilmiştir. Çizelge–2.4’te ise bu yazaçların reset sonrası aldıkları değerler belirtilmiştir. Akümülatör ve durum yazacı da bu bölgede yer alır ve birer adresleri vardır. Şimdi bu yazaçları daha yakından tanıyalım.

### Akümlatör

Kısaca A veya Acc olarak adlandırılan bu yazaç aritmetik ve mantık işlemlerde ana yazaç olarak kullanılır. Dış veri belleğinden veri alış verişi ve program belleğinden veri okuma işlemleri de sadece bu yazaçtan yapılabilir. SFR bölgesinde 0E0H adresinde yer alır, bit adreslenebilir. Kullanıldığı birçok komutta adresi op-kod içerisinde yer aldığından ayrıca adres yazılmaz. Eğer adresi op-kod içerisinde olmayan bir komut kullanılıyorsa assembler programına bildirmek için A yerine Acc kısaltması kullanılmalıdır.

## Mikrodenetleyiciler 8051 Uygulamaları

ADRES	SİMGE	YAZAÇ ADI	RESET
080H	P0	Paralel G/Ç Kapısı 0	FFH
081H	SP	Yığın Gösterici	07H
082H	DPH	Yüksek değerli Bayt	00H
083H	DPL	Düşük değerli Bayt	00H
087H	PCON	Güç Denetim (HMOS)	0XXXXXXXB
	PCON	Güç Denetim (CHMOS)	0XXX0000B
088H	TCON	Z/S Denetim	00H
089H	TMOD	Z/S Mod Denetim	00H
08AH	TL0	Zamanlayıcı/Sayıcı 0 DDB	00H
08BH	TL1	Zamanlayıcı/Sayıcı 1 DD	00H
08CH	TH0	Zamanlayıcı/Sayıcı 0 YDB	00H
08DH	TH1	Zamanlayıcı/Sayıcı 1 YDB	00H
090H	P1	G/Ç portu 1	FFH
098H	SCON	Seri Kapı Denetim	00H
099H	SBUF	Seri Kapı Tampon	00H
0A0H	P2	G/Ç portu 2	FFH
0A8H	IE	Kesme İzin	0X000000B
0B0H	P3	G/Ç portu 3	FFH
0B8H	IP	Kesme Öncelik	XXX00000B
0C8H	T2CON	Z/S 2 Denetim yazacı	00H
0CAH	RCAP2L	Z/S 2 Yakalama YDB	00H
0CBH	RCAP2H	Z/S 2 Yakalama DDB	00H
0CCH	TL2	Zamanlayıcı/Sayıcı 2 DDB	00H
0CDH	TH2	Zamanlayıcı/Sayıcı 2 YDB	00H
0D0H	PSW	Durum yazacı	00H
0E0H	ACC	Akümülatör	00H
0F0H	B	B Yazacı	00H

Çizelge–2.4 Özel amaçlı yazaçların isimleri, adresleri ve reset ile aldığı değerler.

### B Yazacı

B yazacı veya başka söyleyiş ile B akümülatörü 0F0H adresinde yer alır, çarpma ve bölme işlemlerinde yardımcı akümülatör olarak kullanılır. MUL AB komutu A akümülatörü ile B akümülatörünün içeriklerini çarparak 16 bitlik sonucun düşük değerli kısmını A akümülatörüne, yüksek değerli kısmını B akümülatörüne yazar. DIV AB komutu, A akümülatörünün içeriğini (bölünen) B akümülatörünün içeriğine (bölün) böler. Bölümü A akümülatörüne, kalanı ise B akümülatörüne yazar. Ayrıca B akümülatörü A akümülatörü gibi bit adreslenebilir, özel görevleri dışında genel amaçlı yardımcı yazaç olarak kullanılabilir.

### Durum Yazacı

Durum yazacı işlemcinin yaptığı en son işlem hakkında bilgiler içerir. Boş kalan bitleri denetim amacıyla kullanılmıştır. Bu yazaca hem durum göstermesi hem de denetim bitleri içermesi nedeniyle durum/denetim yazacı adı da verilir. D0h adresinde yer alan durum yazacının her bitinin görevi çizelge–2.5’te gösterildiği gibi farklıdır. Şimdi bu görevlerini yakından tanıyalım.

	7						0	
PSW	CY	AC	FO	RS1	RS0	OV	-	P

BİT	SEMBOL	BİT ADRES	AÇIKLAMA
PSW.7	CY	D7H	Elde bayrağı
PSW.6	AC	D6H	Yardımcı elde bayrağı
PSW.5	FO	D5H	Bayrak 0
PSW.4	RS1	D4H	Bank seçme biti
PSW.3	RS0	D3H	Bank seçme biti
PSW.2	OV	D2H	Taşma bayrağı
PSW.1	---	D1H	Rezerve
PSW.0	P	D0H	Çift eşlik bayrağı

Çizelge–2.5 Durum yazacının içeriği (PSW) içeriği.

#### Elde bayrağı (C veya CY)

Elde bayrağı aritmetik işlemlerde kullanılır. Toplama işleminde eğer toplanan sayıların en yüksek değerli bitlerinden bir elde oluşur ise elde bayrağı kurulur. Çıkarma işleminde ise çıkarılan daha büyük ise elde bayrağı kurulur. Elde bayrağı

ayrıca boolean akümülatörüdür. Bir bitlik işlemlere birinci işlenen elde bayrağında bulunmak zorunda ve ayrıca sonuçta elde bayrağına yazılır.

#### Yardımcı Elde Bayrağı (AC)

Yardımcı elde bayrağı İKO sayıların toplanmasında üçüncü bitlerin toplamından elde oluşması durumunda veya sonucun düşük değerli nibble 0Ah-0Fh aralığında ise kurulur.

#### Yazaç Bankı Seçme Bitleri

Bank seçme bitleri (RS0, RS1) etkin yazaç bankasını belirler. Reset işareti ile bu iki bit temizlenir ve böylece bank 0 başlangıç olarak seçilir. Diğer bankların etkin yapılabilmesi için yazılım ile bu bitlerin ayarlanması gerekir. Çizelge-2.6'da bankların seçimleri için RS0 ve RS1 bitlerinin değerleri verilmiştir.

RS1	RS0	SEÇİLEN BANK
0	0	BANK 0 (Reset ile seçilir)
0	1	BANK 1
1	0	BANK 2
1	1	BANK 3

Çizelge-2.6 Bank seçme bitleri ve seçilen yazaç bankası.

#### Bayrak 0

Bayrak 0 genel amaçlı bayraktır, herhangi bir özel görevi yoktur programcı kendi istediği görevi bu bite verebilir.

#### Taşma Bayrağı

Taşma bayrağı (OV) toplama veya çıkarma işlemi sonrası eğer aritmetik taşma varsa kurulur. Yazılım bu bite bakarak sonuç hakkında gerekli düzenlemeyi yapar. Bu bit işaretli sayılar ile aritmetik işlemler yapıldığında kullanılır. İşaretsiz sayılarla aritmetikte bu bayrak ihmal edilir. Toplama sonucu +127'den büyük veya -128'den küçük ise bu bit kurulur.

#### Eşlik Biti

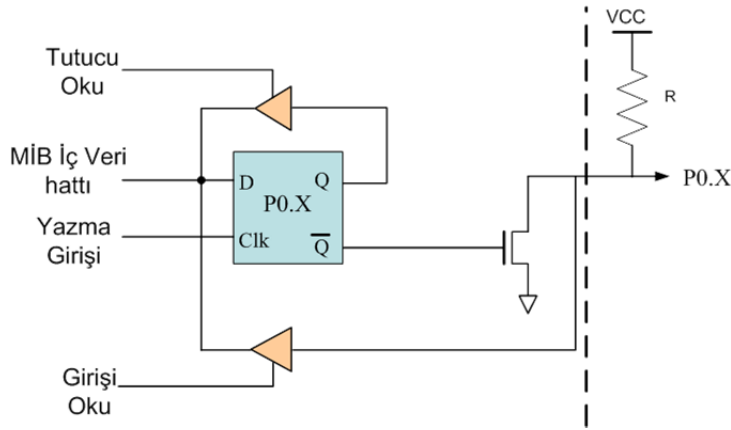
Eşlik biti her makine saykılında otomatik olarak akümülatördeki çift eşliği denetler, eğer birlerin sayısı tek ise bu bit kurulur. Akümülatör ile bu bitin durumu sürekli çift eşlik oluşturur. Başka bir söyleyişle akümülatörün içerisindeki birlerin sayısı tek olduğunda kurulur ve akümülatör ve bu bitin içeriğindeki bir sayısını çift hale getirir. Eşlik biti iletişim sistemlerinde hata denetlemek amacı ile kullanılır.

### Port Yazaçları

8051'in portlarına SFR bölgesinde yer alan yazaçları kullanılarak erişilebilir. portların yapılarında küçük farklılıklar vardır. Bu farklılıklar portları okuma ve yazma işleminde herhangi bir farklılık yaratmaz. Yapılarının farklı olmasının nedeni birçok port hattının giriş çıkış işleminin dışında ikincil görevleri vardır. Port yazaçları 80h, 90h, A0h, B0h adreslerinde yer alır. Eğer dış bellek kullanılıyor ise port 0, 2 ve 3 genel amaçlı giriş/çıkış hattı olarak kullanılamaz. Tüm portlar bit adreslenebilir, bu özellik denetim uygulamalarında programcıya kolaylık sağlayacaktır. Her hattı farklı bir motor veya röle bağlanarak çok sayıda değişken aç-kapa denetimi yapılabilir.

### PORT 0

Port 0'ın ikincil görevi dış bellek kullanıldığında zaman paylaşımli adres yolu düşük değerli baytı ve veri yolu olmasıdır. Port 0'ın iç yükseğe çekme dirençleri veri/adres yolu olarak kullanılması dışında yoktur. Bağlanılan cihaza göre bu porta dışarıdan yükseğe çekme dirençleri bağlanmalıdır.



Şekil- 2.10 Port 0'ın iç yapısı.

Şekil-2.10'da port 0'ın iç yapısı verilmiştir. Hatta bağlanacak R direncinin değeri kullanılan 8051'in veri yapraklarından elde edilecek port hatlarının sink akımına ve bu hatta bağlanacak yükün değerine göre seçilmelidir. Port 0'ın bir hattına veri yazmak için tutucu çıkışına istenilen veri yazılır. Tutucu çıkışta bulunan MOSFET'i sürerek yazmak istediğimiz veriyi hat çıkışına ulaştırır. Giriş olarak kullanmak istediğimizde önce çıkışa "1" yazılır ve çıkıştaki MOSFET'in yalıtıma geçmesini sağlarız ve daha sonra girişi okuyan komut ile girişin seviyesi 8051 yazaçına

## Mikrodenetleyiciler 8051 Uygulamaları



aktarılır. 8051 portlarını giriş ve çıkış olarak kullanılırken yönlendirme işlemi yapılmaz. Bir anlamda otomatik giriş çıkış seçimi yapılır diyebiliriz. Giriş işlemi öncesi 1 yazılmasının nedeni önceki işlemde 0 yazılı kalırsa port hattı girişine 1 uygulansada 0 uygulansa da her zaman MOSFET iletimde olacağı için sürekli 0 okunur. 8051'in giriş çıkış hatları yönlendirme yapılan mikrodenetleyici giriş çıkış hatları ile karşılaştırıldığında zayıf bir yapıya sahiptir aşırı akım ve gerilimlere karşı korumasız olduğu için kolay zarar görür. Güçlü çıkış akımın gerek olduğu uygulamalarda transistör veya mantık geçitleri kullanılarak akım olarak çıkış hattarı güçlendirilmelidir.

### PORT 1

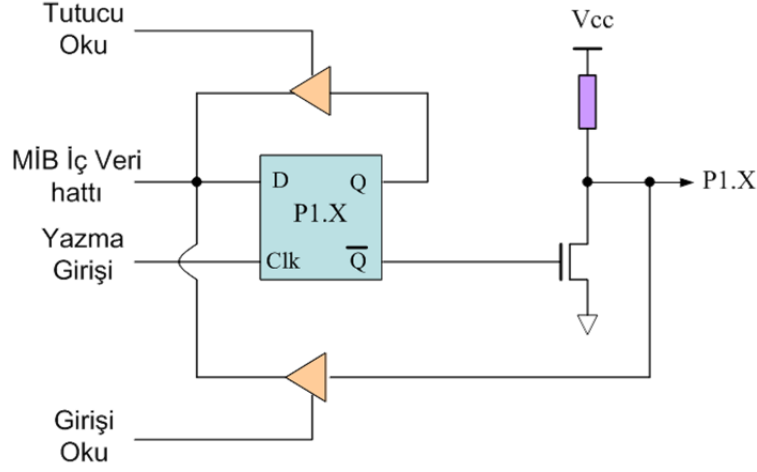
Port 1 iç yükseğe çekme dirençleri olan 8 adet çift yönlü çalışabilen Giriş/Çıkış hattından oluşur. Port 1'in her bir hattı 4 adet LS ailesi TTL geçidi sürebilecek kaynak/şase kapasiteye sahiptir. 8051'de port 1'in ikincil bir görevi yoktur, fakat 8052'de zamanlayıcı 2'nin girişleri için P1.0'a sayıcı tetikleme, P1.1'e ise yakalama/yeniden yükleme işlemi tetikleme girişi görevleri verilmiştir. Seri programlama yapılabilen 8052'nin yeni sürümlerinde P1.5'e MOSI, P1.6'ya MOSO ve P1.7'ye SCK görevleri verilmiştir. Port 1'in ikincil görevleri ve bit adresleri çizelge-2.7'de verilmiştir. Port 1, 2 ve 3'ün iç yapıları aynıdır ve şekil-2.11'de gösterilmiştir.

### PORT 2

Port 2 iç yükseğe çekme dirençleri olan 8 adet çift yönlü çalışabilen Giriş/Çıkış hattından oluşur. Port 2'nin her bir hattı 4 adet LS ailesi TTL geçidi sürebilecek kaynak/şase kapasiteye sahiptir. port 2'nin ikincil görevi dış bellek kullanımında adresin yüksek değerli 8 bitini taşımasıdır. Dış bellek kullanılmadığında veya 256 bayt kullanıldığında yüksek değerli adrese gereksinim olmayacaktır ve bu hatlar serbest I/O hattı olarak kullanılabilirler.

### PORT 3

Port 3 iç yükseğe çekme dirençleri olan 8 adet çift yönlü çalışabilen Giriş/Çıkış hattından oluşur. Port 3'in her bir hattı 4 adet LS ailesi TTL geçidi sürebilecek kaynak/şase kapasiteye sahiptir. Port 3'ün her hattının farklı bir ikincil görevi vardır, ikincil görevleri ve adresleri her hat için çizelge-2.7'de verilmiştir. Özel görevlendirildiği birim kullanılmadığı durumlarda bu hatlar serbest giriş/çıkış (I/O) hattı olarak kullanılabilir.



Şekil–2.11 Port 1,2 ve 3'ün içyapısı.

BİT	İSİM	BİT ADRES	İŞLEVİ
P3.0	RXD	B0H	Seri kanal veri girişi
P3.1	TXD	B1H	Seri kanal veri çıkışı
P3.2	INT0	B2H	Dış kesme 0 girişi
P3.3	INT1	B3H	Dış kesme 1 girişi
P3.4	TO	B4H	Zamanlayıcı/sayıcı 0 dış girişi
P3.5	T1	B5H	Zamanlayıcı/sayıcı 1 dış girişi
P3.6	WR	B6H	Dış belleğe yazma işaret çıkışı
P3.7	RD	B7H	Dış bellekten okuma işaret çıkışı
P1.0 <sup>4</sup>	T2	90H	Zamanlayıcı/sayıcı 2 dış girişi
P1.1	T2EX	91H	Zamanlayıcı 2 yakalama/yenidenyükleme
P1.5 <sup>5</sup>	MOSI	95H	Seri programlama giriş
P1.6	MISO	96H	Seri programlama çıkış
P1.7	SCK	97H	Seri programlama saat

Çizelge–2.7 PORT 3 ve PORT 1'in ikincil görevleri.

<sup>4</sup> 8052'de vardır.<sup>5</sup> AT89S52'de vardır.

Port 3 diğer portlara göre daha güçlü çıkış hatlarına sahip olduğu için ikinci görevleri olmadığı durumlarda tasarımda bu porta öncelik verilmelidir. Port 0'da olduğu gibi bu portta da giriş olarak kullanılmadan önce mutlaka çıkışa 1 yazılmalıdır. Özellikle kesme ve zamanlayıcı girişlerinde bu kurala kesinlikle uyulmalıdır.

### Veri Gösterici

Veri gösterici yazaç (DPTR) dış veri veya program belleğine ulaşmak için kullanılan 16 bitlik bir yazaçtır. 82H ve 83H adreslerinde yer alır, 8 bitine veya bir defada 16 bitine ulaşabilen komutlar vardır. Bu yazaca bir defada 16 bit sayı yüklenebilir veya içeriği arttırılabilir ya da azaltılabilir. Yeni türev 8051'lerde ikinci bir DPTR1 yazacı eklenmiştir çalışma şekli benzerdir.

### Yığın İşaretleyici

Yığın işaretleyici 81H adresinde yer alan 8 bit bir yazaçtır. Yığına en son atılan verinin adresini gösterir. Yığına veri atma işleminde önce yığın işaretleyicinin içeriği bir arttırılır ve bilgi yığına atılır. Yığından bilgi çekmede ise önce bilgi akümülatöre alınır ve sonra içerik bir azaltılır. 8051'de yığın alt 128 baytlık iç RAM ile sınırlıdır 8052'de ise iç RAM belleğin tamamı (256 bayt) yığın olarak kullanılabilir. Reset sonrası yığın işaretleyici 07H adresine ayarlanır. Bu demektir ki bank 0 dışındaki banklar ve bit adreslenen bölge yığın olarak kullanılacağı için özel görevlerinde kullanılamayacaklardır.

### Zamanlayıcı Yazaçları

8051 iki adet 16 bit zamanlayıcı/sayıcıya sahiptir. Zamanlayıcı 0 ve 1'in çalışma kipi seçme işlemleri için TMOD denetim ve durum gösterme işlemleri için TCON yazaçları görevlendirilmiştir. Bu iki yazaç 8 bitliktir ve her bitine ayrı görev verilmiştir. Zamanlayıcı/Sayıcı 0 ve 16 bit bir sayıcıdır. 8051'in SFR yazaç yapısı ise 8 bitlik olduğu için Zamanlayıcı/Sayıcı 0 TL0 ve TH0 olarak adlandırılmış iki 8 bit yazacın birleşiminden oluşur. Zamanlayıcı/Sayıcı 1 ise TL1 ve TH1 olarak adlandırılmış iki 8 bit yazacın birleşiminden oluşur.

8052'ye Zamanlayıcı/Sayıcı 2 olarak adlandırılan üçüncü 16 bitlik bir Zamanlayıcı/Sayıcı eklenmiştir. Bu zamanlayıcının çalışma kipi seçme bitleri, T2MOD yazacında denetim bitleri ve durum bitleri ise T2CON yazacında yer alır. Zamanlayıcı/Sayıcı 2 8'er bitlik T2H ve T2L yazaçlarından oluşur. Zamanlayıcı/Sayıcı 2 diğer zamanlayıcı/sayıcılardan farklı olarak 16 bit yeniden yükleme ve yakalama çalışma kiplerinde kullanmak üzere RCAP2L RCAP2H yazaçlarına sahiptir. Çizelge-

2.8’de 8052’nin zamanlayıcıların yazaçlarının isimleri, simgeleri, görevleri ve adresleri gösterilmiştir.

Adres	İSİM	İŞLEVİ
88H	TCON	Zamanlayıcı 0 ve 1 denetim
89H	TMOD	Zamanlayıcı 0 ve 1 kip seçme
8AH	TL0	Zamanlayıcı 0 DDB
8BH	TL1	Zamanlayıcı 1 DDB
8CH	TH0	Zamanlayıcı 0 YDB
8DH	TH1	Zamanlayıcı 1 YDB
0C8H	T2CON	Zamanlayıcı 2 denetim
0C9H	T2MOD	Zamanlayıcı 2 kip seçme
0CAH	RCAP2L	Zamanlayıcı 2 yenidenyükleme/yakalama DDB
0CBH	RCAP2H	Zamanlayıcı 2 yenidenyükleme/yakalama YDB
0CCH	TL2	Zamanlayıcı 0 DDB
0CDH	TH2	Zamanlayıcı 1 YDB

Çizelge–2.8 8052’nin zamanlayıcı yazaçları.

### Seri Port Yazaçları

8051 tümdevre üzerinde seri iletişim yapan cihazlarla iletişim kurmak amacı ile seri porta (UART) sahiptir. Seri port tampon yazacı (SBUF) 99h adresinde yer alır. Gönderilecek olan veriyi veya gelen veriyi içerir. Çalışma kipi ve işlem sonuçları ise seri port denetim yazacı (SCON) tarafından yapılır. Yeni türev 8051 üyelerinde ikinci bir UART1 birimi, SPI haberleşme birimi ve I<sup>2</sup>C seri haberleşme birimleri eklenmiştir ve bu birimleri kullanmak ve denetlemek için yeni yazaçlar eklenmiştir.

### Kesme Yazaçları

8051 iki önceliklime düzeyi olan 5 adet kesme kaynağına sahiptir. Sistem resetlendiğinde kesmeler etkisiz hale getirilir, yazılım ile kullanılacak kesmeler kesme izinleme (IE) yazacı ile genel ve bireysel olarak etkin hale getirilebilir. Kesme kaynaklarının algılanması sırasında oluşacak kargaşanın engellenmesi için kesme öncelik sıralama yazacı (IP) yerleştirilmiştir. 8052’de zamanlayıcı 2 kesmesi en düşük önceliği sahip olarak kesme kaynaklarına eklenmiştir.

## Mikrodenetleyiciler 8051 Uygulamaları

### Güç Denetim Yazacı

Güç denetim yazacının görevi mikroişlemcinin güç tüketimini azaltmaktır. Mikroişlemcinin normal çalışması dışında güç azaltılması mümkün değildir. Fakat işlemini bitiren ve yeni bir iş bekleyen mikroişlemci uyku moduna geçerek güç tüketimini azaltabilir. Güç denetim yazacı (PCON) içerisine bu amaca hizmet etmeyen başka birimlerin denetim bitleri de yerleştirilmiştir, isimleri ve görevleri çizelge–2.9’da gösterilmiştir. SMOD biti seri kanalın iletim hızını ikiye katlar. 4, 5, 6 nolu bitler kullanılmamıştır. Güç denetim bitleri IDL ve PD sadece CMOS tümdevrelerde kullanılmaktadır.

BIT	SİMGE	AÇIKLAMA
7	SMOD	Seri port mod 1, 2 veya 3 modlarında çalışırken iletişim hızını iki katına çıkarır.
6	---	Kullanılmamış
5	---	Kullanılmamış
4	---	Kullanılmamış
3	GF1	Genel kullanıma açık bayrak
2	GF0	Genel kullanıma açık bayrak
1*	PD	MİB’ni Kısık güçte çalışma kipine sokar sadece reset ile çıkar.
0*	IDL	MİB’i aylak çalışma kipine sokar reset veya kesme ile çıkar.

\*sadece CMOS sürümlerinde vardır.

Çizelge–2.9 PCON yazacının içeriği.

### Aylak Çalışma Kipi

IDL bitini kuran komut işletilen son komut olacaktır. Çünkü bu bit kurulduğunda MİB’nin saat işareti kapatılır, MİB en son halini korur. Seri portu ve zamanlayıcıların çalışmasını etkilemez. Portların mantık seviyeleri son durumlarını korur. ALE ve PSEN işaretleri yüksek seviyede kalırlar. IDL moddan ancak kesme veya sistem reseti ile çıkılabilir.

### Kısık Güçte Çalışma Kipi

PD bitini kuran komut en son işletilen komut olur. PD biti kurulduktan sonra işlemcinin saati kapatılır. Reset ile ancak kısık güç çalışmadan çıkılabilir. Tüm

işlemler sayıcılar ve seri port da olmak üzere durdurulur, portlar ve yazaçlar en son halini korurlar. ALE ve PSEN düşük seviyede kalırlar.

### **Program Belleği**

MCS-51 ailesi mikrodentleyiciler program belleği olarak tümdevre üzerinde yer alan ROM, EPROM, FLASH bellek türleri kullanılır. Üzerinde bellek bulunmayan üyelerine EA bacağı şaseye bağlanarak dışarıdan kalıcı tür program belleği bağlanabilir. Günümüzde üretilen MCS-51 üyelerinin hepsi yeteri kadar iç program belleğine sahiptir, yeterli olmadığında dış bellek bağlamak yerine daha yüksek belleği olan bir ürün seçilmektedir. 8051 iç ve dış olmak üzere 65536 satır program belleğini adresleyebilir. Bazı üreticiler kendilerine özgü yöntemle program belleğini kapasitesini 512 Kbayta kadar arttırmışlardır. Bazı üreticiler ise program belleğinin bir kısmını veri saklamak üzere çalışırken programlanmasına olanak sağlamışlardır. MCS-51 ailesi mikrodentleyiciler program belleğinden okuma yaparken PSEN hattını etkin yapar. Adres yolu olarak P0 ve P2'yi veri yolu olarakta yine P0'ı kullanır. P0'ın adres ve veri yolu olarak zaman paylaşımli kullanımını ALE işareti sağlar.

### **Dış Bellek Kullanımı**

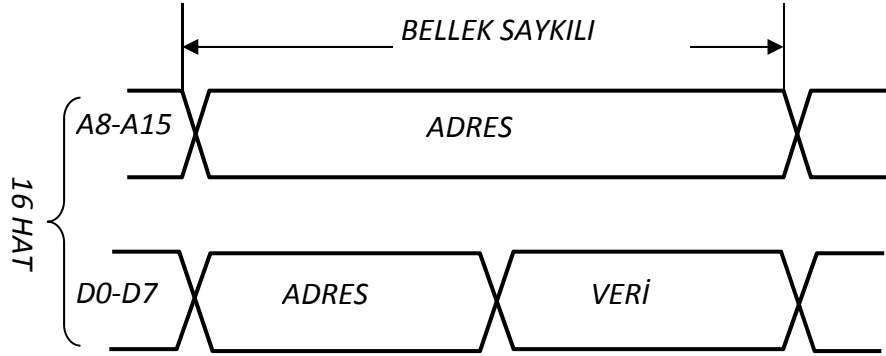
Mikrodenetleyicilerin sadece içerisinde bulunan birimleri ile devre tasarlamak kullanım alanlarını kısıtlar. Özellikle iç belleğin kullanımı bazı uygulamalarda yetersiz kalabilir. Mikrodenetleyici üreticileri devre tasarımcılarını daha fazla olanak sunmak için iç belleğin yanı sıra dışarıdan hem program belleği hem veri belleği bağlantısının yapılması için mikrodenetleyici üzerinde gerekli düzenlemeleri yapmışlardır. MCS-51 ailesi mikrodenetleyiciler de 64 K dış program ve veri belleği adres alanına sahiptir. İç belleğin yetersiz kaldığı uygulamalarda dışarıdan 64 Kbayta kadar program belleği, 64 Kbayt veri belleği eklenebilir. Ayrıca çevre arabirimi (Peripheral interface) dış bellek alanına eklenebilir. Dış bellek kullanıldığında Port 0 giriş/çıkış işlemleri için kullanılamaz. Bu port zaman paylaşımli adresin düşük değerli baytı (A0-A7) ve veri yolu (D0-D7) olarak kullanılır. Dış bellek işlem saykılının birinci yarısında ALE işareti ile adresin düşük değerli baytı tutucu çıkışlarında tutulur. İkinci yarıda veri yolu olarak kullanılır. Port 2 ise kullanıldığında adresin yüksek değerli baytı olarak görevlendirilir. Kullanılan dış bellek 256 baytın altında ise adres için 8 bit yeterli olduğundan port 2 giriş çıkış işlemleri için kullanılabilir. Zaman paylaşımli aynı portun adres ve veri yolu için kullanılması

### **Mikrodenetleyiciler 8051 Uygulamaları**

adres ve veri yolu için kullanılacak hat sayısını azaltır. Zaman paylaşımı kullanılmayan işlemcilerde 24 bacak bu işlem için kullanılırken zaman paylaşımı kullanan sistemlerde 16 bacak kullanılır. Diğer bacaklar farklı işlevler için kullanılabilir. Bu tür işlemcilerin DIP 40 soketinde üretildiğini düşünürsek 8 hat oldukça değerlidir. Şekil-2.12’de zaman paylaşım ve paylaşım sistemlerin zamanlama diyagramları verilmiştir. Zaman paylaşım çalışmada işlem saykılının birinci yarısında port 0 çıkış olarak yönlendirilir ve adresin düşük değerli baytı yazılır. ALE işareti etkin yapılarak 74HHC373 veya eşdeğer tutucuda tutulur. Port 2’ye ise adresin yüksek değerli baytı yazılır. İkinci yarıda port 2’nin içeriği değişmez fakat port 0, okuma işlemi yapılıyor ise giriş yapılır ve veri okunur. Yazma işlemi ise çıkış yapılır ve veri yazılır. İkinci yarıda ALE işareti etkin olmaz fakat program belleğinden işlem yapılıyorsa PSEN, veri belleğinden işlem yapılıyor ise WR veya RD işaretleri etkin yapılır.



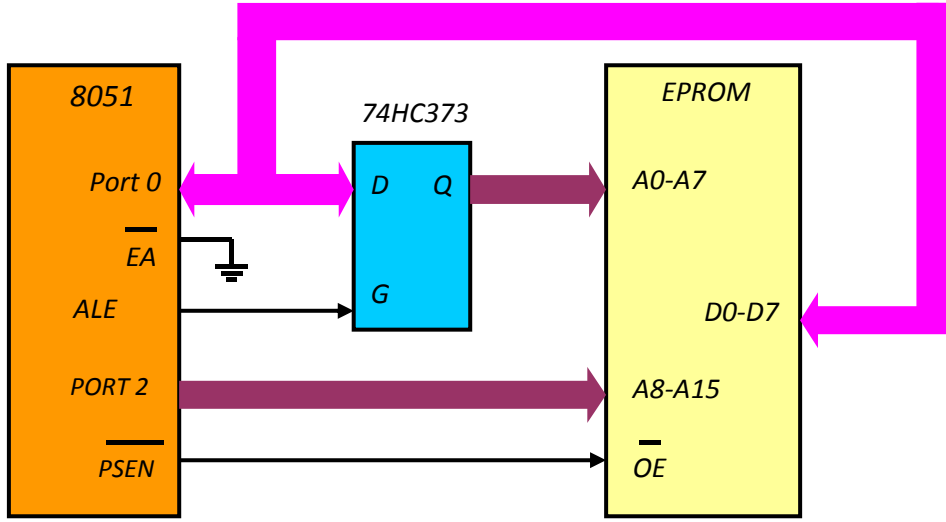
Şekil-2.12 Zaman paylaşım adres ve veri yolunun zamanlama diyagramı.



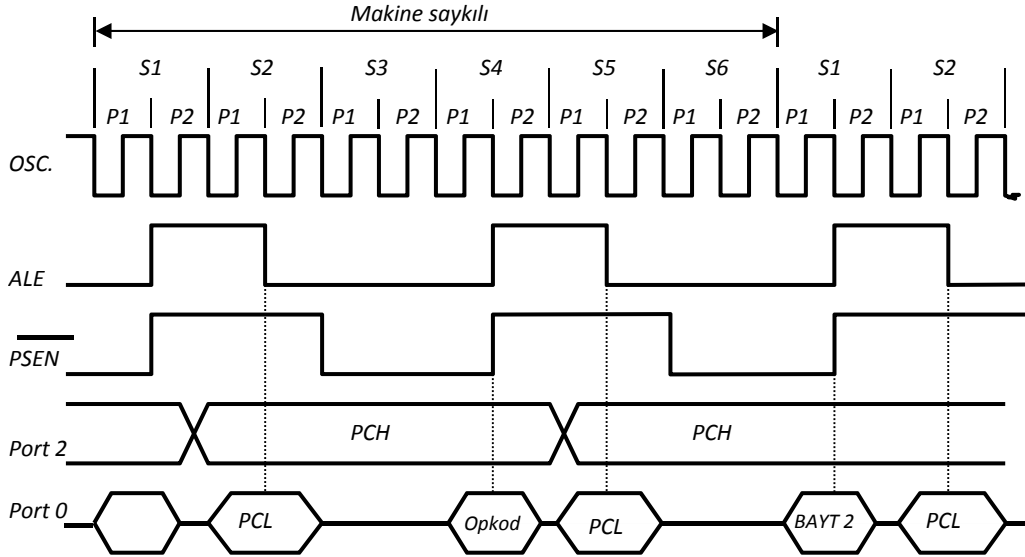
Şekil-2.13 Zaman paylaşım adres ve veri yolunun zamanlama diyagramı.

## Dış Program Belleği

Dış program belleği PSEN işareti ile etkin olan sadece okunabilir türü bellektir. Dış EPROM kullanıldığında port 0 ve port 2 genel amaçlı Giriş/Çıkış hattı olarak kullanılamaz. Bu tür belleğin 8051'e bağlantısı şekil-2.14'te gösterilmiştir. 8051'in bir makine saykılı 12 osilatör periyodundan oluşur. 12 MHz'lik kristal kullanıldığında bir makine saykılı 1  $\mu$ S'dir. Bir makine saykılı süresince ALE işareti iki defa etkin olur ve dış program belleğinden iki bayt okunur. Eğer okunan komut tek baytlık ise ikinci bayt atılır. Bu işlemin zamanlama diyagramı şekil-2.15'de gösterilmiştir.



Şekil-2.14 Dış program belleği bağlantısı.



Şekil-2.15 Dış program belleğinden okuma.

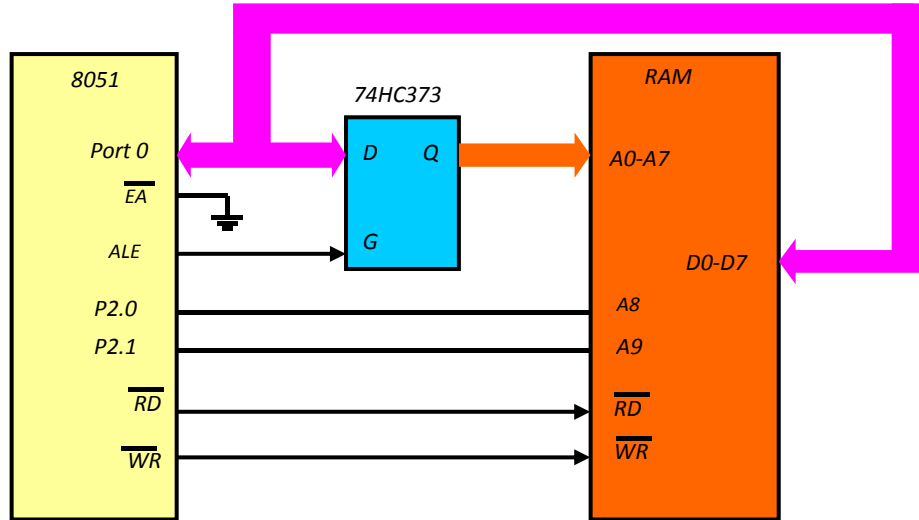
## Mikrodenetleyiciler 8051 Uygulamaları



## Dış Veri Belleği

Dış veri belleği RD ve WR işaretleri ile etkinleştirilen oku/yaz türü belleklerdir. RD ve WR işaretleri P3.7 ve P3.6 hatlarından MOVX komutu ile üretilirler. MOVX komutunda adres gösterici olarak DPTR kullanıldığında dış veri belleğinin adres hattı 16 olur, R0 veya R1 yazaçlarından biri kullanıldığında adres hattı 8 bittir. Dış veri belleği olarak 256 baytlık bellek kapasitesi yeterli ise ikinci seçenek kullanılır.

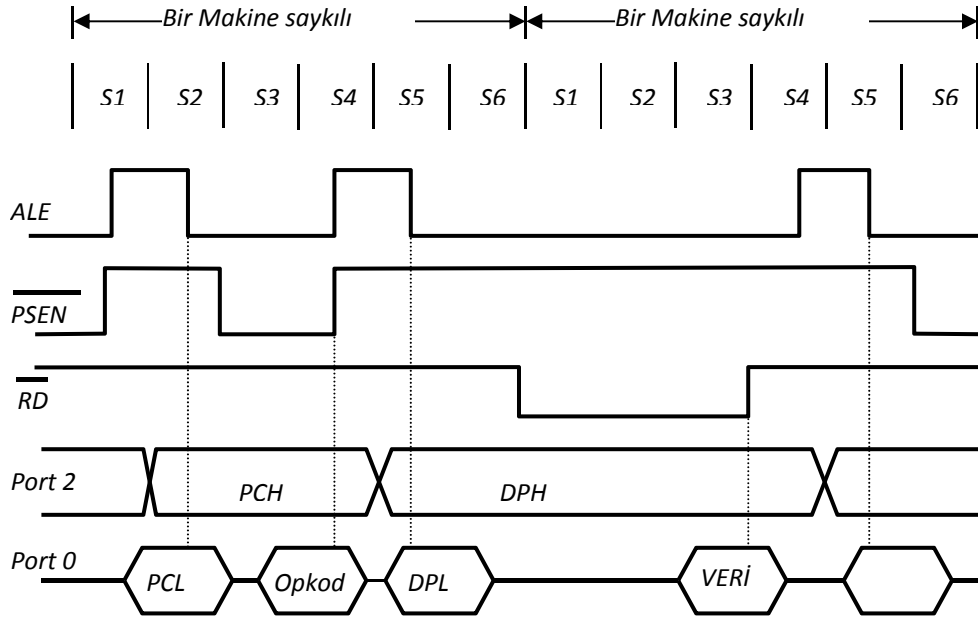
Şekil-2.16'da gösterildiği gibi veri belleği bağlantısı adresin düşük değerli kısmı ile veri yolu bağlantısı EPROM'da olduğu gibidir. Farklı kısmı ise RD işaretinin RAM belleğin OE girişine ve WR işaretinin de W girişine bağlanmasıdır. PSEN işareti dış veri belleği bağlantısında kullanılmaz. Port 0 ve Port 2 kullanılarak 64 Kbayta kadar dış veri belleği bağlanabilir. Örnek şekilde ise 1Kbayt'lık RAM bağlantısı gösterilmiştir. P2 portunun boşta kalan hatları bilinçli bir şekilde serbest G/Ç hattı olarak kullanılabilir. 8051 bu hatlar bir yere bağlanmasa da MOVX A,@DPTR veya MOVX @DPTR, A komutları işletildiğinde mantık seviyelerini değiştirebilir.



Sekil - 2.16 Dış veri belleği bağlantısı.

MOVX A,@R0 veya MOVX @R0, A işletildiğinde ise bu komutlarda sadece 8 bit adres kullanılması nedeniyle P2 portu etkilenmeyecektir. Bu yöntemle erişilebilecek bellek kapasitesi 256 bayttır. Daha büyük kapasite gerekiyorsa P2 ya da diğer boş olan portların hatları gerektiği kadar RAM belleğin A8, A9, A10 adres girişlerine bağlanır ve bu hatların mantık seviyeleri programcı tarafından okuma ve yazma komutları işletilmeden önce yapılır. Bu yöntem 2 Kbayta kadar bellek bağlamak için

verimli olabilir, daha büyük bellek bağlamak için veri gösterici olarak DPTR kullanılır. İşlem yapılacak veri belleğinin adresi DPTR yazacına yüklenir ve sonra okuma veya yazma işlemi yapılır. Şekil-2.17’de MOVX A, @DPTR komutunun işleme zamanlaması verilmiştir. Birinci makine saykılında opkod program belleğinden okunacağı için ALE ve PSEN sırasıyla etkin olmuştur. İkinci makine saykılında işlem dış veri belleğinden yapıldığı için PSEN etkin değildir RD işareti etkindir. Dış veri belleğine yazma işlemi de okuma ile aynı olacaktır, tek fark işlemci tarafından ikinci makine saykılında RD işareti yerine WR işareti etkin yapılacaktır.



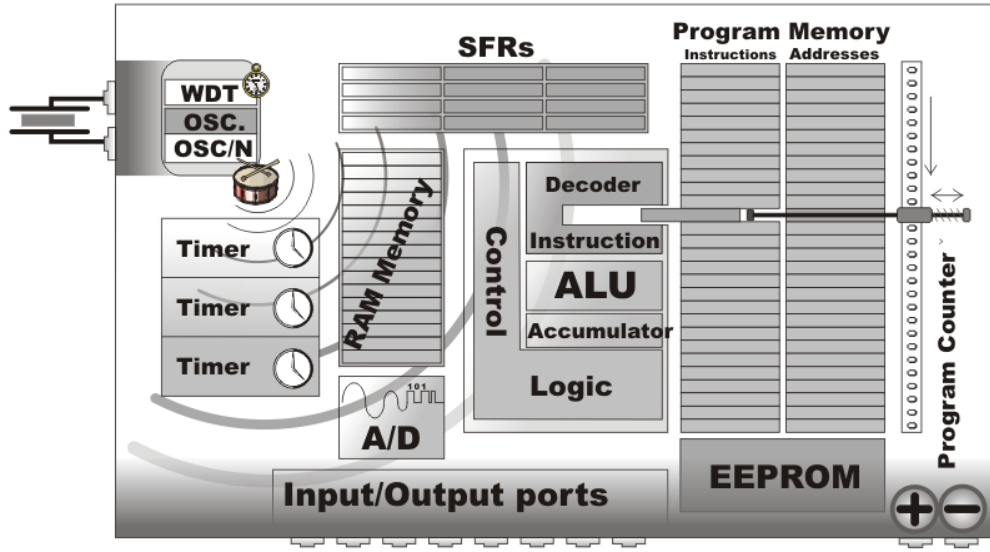
Sekil-2.17 MOVX komutunun zamanlama diyagramı.

## Sorular

1. MCS-51 ailesi mikrodenetleyicilerin bellek yapısını açıklayın?
2. İç RAM'in kısımlarını ve her kısmın görevlerini açıklayın?
3. SFR bölgesinde yer alan yazaç türlerini yazın?
4. MCS-51 ailesi mikrodenetleyicilerde dış program belleği ne zaman kullanılır?
5. Kısa ve uzun adres kullanan dış veri bellek bağlantıları arasında ne gibi farklılıklar vardır açıklayın?
6. Osilatör frekansı 24 Mhz olan 8051'in makine saykılını hesaplayınız.
7. 8051'in P0'ın yapısını ve özel görevlerini kısaca açıklayınız.
8. 8051'in P3'ün yapısını ve özel görevlerini açıklayınız.



# MCS-51 Ailesi Mikrodenetleyicilerin Komutları



## Giriş

Mikroişlemciyi yaptırmak istediğimiz işlemleri onun anlayacağı dilden kurallarına göre bildirmemiz gerekir. Her mikroişlemcinin temel işlemleri gerçekleyen komut kümesi vardır. Bu komutları kurallarına göre birleştirdiğimizde program elde ederiz. MCS-51 ailesi işlemcilerin komut kümesi denetim uygulamalarına yönelik olarak düzenlenmiştir. Bu tür uygulamalarda 8 bit kelime uzunluğu olan komutların yanı sıra bir bitlik mantık işlemlerini yapan komutlarda sıkça kullanılmaktadır. MIB içerisine yerleştirilen boolean işlemcisi sayesinde 8051 bir bitlik mantık işlemlerini yapabilmektedir. 8 bitlik komutların çoğu iç veri belleği ve özel amaçlı yazaçların bulunduğu alanda işlem yapar. Bunun yanı sıra program belleğinden veri okuyan ve

dış veri belleğinden veri okuyan ve yazan komutlarda yer almaktadır. Bu tür komutların op-code (opkod)'ları 8 bittir. 8 bit uzunluğunda  $2^8=256$  adet komut yazılabilir. Bunlardan 255 tanesi kullanılmış bir tanesi ise tanımlanmamıştır. Bazı komutlar veri veya adres için ikinci, üçüncü bayta sahip olabilir. MCS-51 ailesi işlemcilerin komutlarından 139 tanesi bir bayt uzunluğunda iken 92 tanesi 2 bayt uzunluğunda ve 24 tanesi 3 bayt uzunluğundadır. Ek-A'da tüm komutlar ve opkodları verilmiştir.

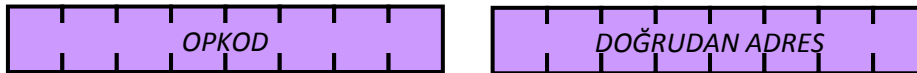
## Adresleme Kipleri

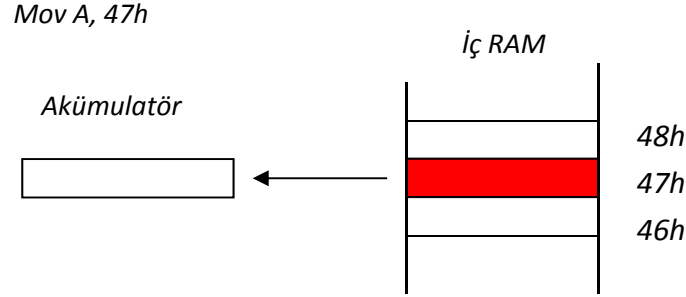
Mikroişlemciye istediğimiz işlemi belirtmek için kullandığımız komuta ek olarak işlemde kullanacağı verinin adresini belirtmemiz gerekir. Verinin bulunduğu belleğe göre adresin belirtme şekli farklıdır. Adres belirleme şekillerine komut adresleme kipleri adı verilir. Adresleme kiplerini mikroişlemci dilinin dilbilgisi kuralları olarak tanımlayabiliriz. MCS-51 ailesinin adresleme kiplerini 8 grupta özetleyebiliriz.

- Doğrudan adresleme
- İvedi adresleme
- Yazaç adresleme
- Bağlı adresleme
- Dolaylı adresleme
- Sıralı adresleme
- Mutlak adresleme
- Uzun adresleme

### Doğrudan Adresleme

Doğrudan adreslemede işlem yapılacak belleğin adresi komutla beraber yazılır. Bu adresleme kipi ile iç RAM bölgesinde alt 128 ve SFR yazaçları kısımları adreslenebilir. Opkodu izleyen ikinci bayt işlem yapılacak iç RAM satırının adresini içerir. Bu adresleme kipinde tüm komutlar iki bayt uzunluğundadır. Birinci baytı işlem kodu ikinci baytı adres oluşturur. Doğrudan adreslenen bölgenin 0-7FH aralığında alt 128, 80H-FFH aralığında ise özel amaçlı yazaçlar yer alır.





Örnek:

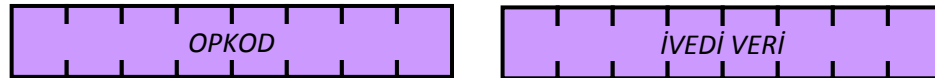
MOV A,25H ;25H adresli iç veri belleğini akümülatöre okur.  
MOV P1, A ;Akümülatörün içeriğini P1 portuna yazar.

Bu komutu makine diline dönüştürürsek;

10001001 – birinci bayt (opkod)  
10010000 – ikinci bayt (P1'in adresi).

### İvedi Adresleme

Programda gerekli sabitleri girmek için ya da yazaçların içeriklerini istenilen değere ayarlamak için kullanılır. Komutların birinci baytları opkod, ikinci baytları veriyi içerir. “#” işareti ivedi adreslemenin simgesidir.



MOV A, #12

Bu komut akümülatöre 0CH sayısını yükler. İki bayt uzunluğundadır, birinci baytı opkod ikinci baytı sabit değerdir. Bir komut dışında ivedi olarak girilebilecek değerlerin uzunluğu 8 bittir. Fakat bu kural DPTR yazacı için çiğnenmiştir. DPTR 16 bitlik olduğu için veri yüklemeyi kolaylaştırmak amacıyla 16 bit veri yükleyen komut 8051 komut kümesinde yer almıştır.

MOV DPTR, #8000H

Bu komut 8000h sayısını DPTR'ye yükler. 3 bayt uzunluğundadır. Birinci bayt opkoddan ikinci ve üçüncü baytlar ise girilecek değerden oluşur.

MOV P1, #55H

Bu komutta birden fazla adresleme kipi kullanılmıştır. Hedef adres doğrudan, kaynak adres ise ivedidir.

Makine diline dönüştürsek 75h, 90h, 55h

### Yazaç Adresleme

Yazaç bankalarında yer alan R0,.....,R7 yazaçları bazı komutlar tarafından kullanılır. Bu tür komutlar bir bayt uzunluğundadır ve 3 biti yazaç adresini tanımlar. Bu adresleme kipinde adres baytı olmadığından program içerisinde yazıldıklarında daha az yer işgal ederler. Opkod için ayrılan bit sayısı azaldığı için işlem sayısı azalır. Bu komutlar yazaç bankalarındaki tüm yazaçlarda kullanılır. Bildiğiniz gibi banka numarası PSW içerisindeki RS0 ve RS1 bayrakları ile belirlenir.



MOV A, R7 ;R7'nin içeriğini akümülatöre yükle.

Bu komut etkin olan bankanın 7 nolu yazaçının içeriğini akümülatöre yükler. Bir bayt uzunluğunda bir komuttur. Opcodu ise EFH'dir. Bazı komutlar sadece bir yazaçta geçerlidir. Bu tür komutlara yazaca özel komutlar adı verilir. İşlemin yapılacağı yer opkod içerisinde yer aldığından adres bilgisi içermezler. Genellikle bu buyruklar akümülatörü ilgilendiren buyruklar olmakla beraber diğer yazaca özel buyruklar da vardır.

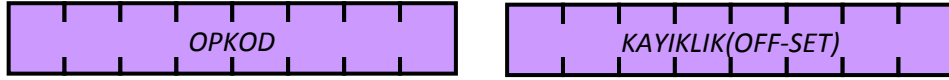
CLR A ;Akümülatörün içeriğini sıfırlar.

Bu komut sadece akümülatörde geçerlidir, bu nedenle bazı kaynaklarda akümülatöre özel adresleme modu kullanır şeklinde belirtilir.

### Bağıl Adresleme

Bağıl adresleme dallanma komutları ile kullanılır. Bağıl adres değeri veya başka deyişle kayıklık (off-set) değeri 8 bit işaretli sayıdır. -128 ile +127 arasında yer alır, dallanma komutunun işlenmesi tamamlandığında program sayacı bir sonraki komutun adresini içermektedir. İşlem yapılacak adres ki bu adrese hedef adres adı verilir, verilen kayıklık değerinin program sayacı ile toplanmasından elde edilir. Eğer negatif kayıklık verildi ise bu işlem bir çıkarma olacaktır ve programın akışı geri yönde olacaktır.





Kayıklık değerini hesaplamak zor olduğu için birçok Assembler programı etiket kullanıldığında kayıklık değerini bizim için hesaplar. Bu adresleme kipinin artısı adresin komutun bulunduğu yere bağlı olmamasıdır. Eksisi ise etki alanının +127 satır ile -128 satır ile sınırlı olmasıdır. Aşağıdaki formüller kullanarak hedef adres belli ise kayıklık değeri, kayıklık değeri belli ise hedef adres hesaplanabilir.

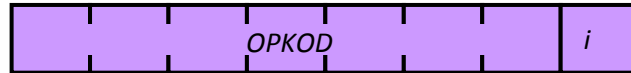
$$\text{Bulunduğu adres} + \text{Kayıklık} = \text{Hedef adres}$$

Veya;

$$\text{Kayıklık} = \text{Hedef adres} - \text{Bulunduğu adres}$$

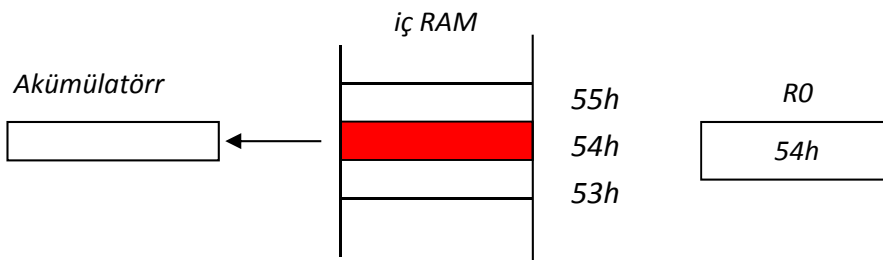
### **Dolaylı Adresleme**

Bu adresleme kipinde komut veri gösterici olarak bir yazacı kullanır. İşlem yapılacak adres bilgisi veri gösterici olarak seçilen yazaca daha önceden yüklenir. Veri gösterici olarak R0, R1, DPTR yazaçları kullanılır.



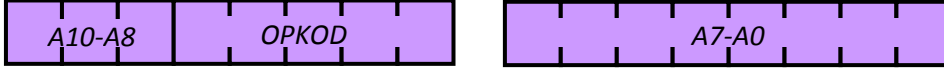
8051 dış veri belleğine sadece dolaylı adresleme kipi ile ulaşabilir. Dış RAM'in boyutuna göre veri gösterici olarak seçilen yazaç değişir. Belleğin boyutu büyük ise (256 bayttan daha büyük) veri gösterici olarak DPTR yazacı, 256 bayttan küçük ise R0 yazacı seçilir. İç bellekte üst 128 baytta da sadece dolaylı adresleme kipi ile ulaşılabilir, veri gösterici olarak R0 ve R1 Yazaçları kullanılır. Alt 128 baytlık kısma doğrudan ve dolaylı adreslemenin her ikisi ile ulaşılabilir. Bu kısımda da R0 ve R1 yazaçlarının her ikisi veri gösterici olarak kullanılabilir. Dolaylı adreslemenin belirtici "@" simgesidir. Bu simge veri gösterici yazacın önüne eklenir.

`Mov A,@R0`



### Mutlak Adresleme

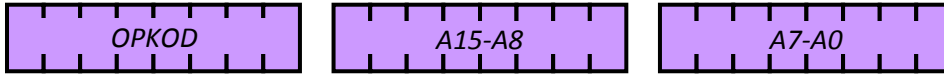
Mutlak adresleme kipi sadece AJMP ve ACALL komutları tarafından kullanılır. Bu adresleme kipinde hedef adresin 11 biti komut içerisinde belirtilir. 5 biti ise program sayacını o anki içeriğinden alınarak hedef adres elde edilir. Bu beş bitin belirlediği 2 Kbaytlık bir bölgeye bağlanılabilir. Komutlar iki bayt uzunluğundadır, birinci baytın ilk beş biti opkodu, son üç biti A10-A8 nolu adres bitleri oluşturur. İkinci baytı ise adresin düşük değerli baytı (A0-A7) oluşturur.

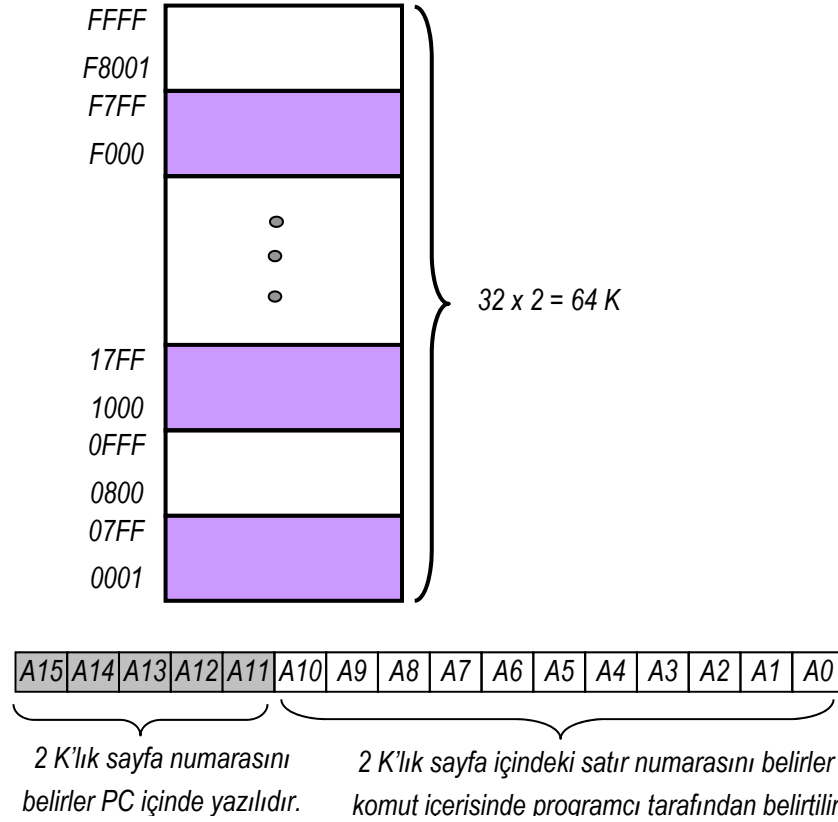


Bağıl adreslemeden daha geniş bir alana dallanabilmesi artısıdır, fakat bu alan adrese bağımlıdır. Eksisi ise 2 Kbaytlık bir bölgeye hem de adrese bağlı şekilde dallanabilmesidir. Bu adresleme kipinde 5 bit sayfa adedini belirlediğine göre  $2^5=32$  adet sayfa seçilebilir. 8051'in 64 Kbayt program belleği olduğuna göre her sayfanın boyutu 64/32'den 2 Kbayt elde edilir. Hedef adresle kaynak adresin ilk beş biti aynı olduğundan 2 Kbaytlık sınırın dışına dallanamaz. Sayfa sınırı sabittir, program sayacının içeriğine göre değişmez. Şekil-3.1'de sayfaların bölünmesi gösterilmiştir.

### Uzun Adresleme

Uzun adresleme LJMP ve LCALL komutlarında kullanılır. Komut 3 bayt uzunluğunda olur, birinci baytı opkod ikinci ve üçüncü baytları adres oluşturur. Artısı tüm 64 Kbaytlık adres aralığına dallanabilir. Eksisi ise dallanma noktasına bağlı olmasıdır. Program başka bir yere taşındığında dallanma noktası değişmediğinden eski yerine dallanma yapacaktır. Taşımalarda mutlaka adres değiştirilmelidir.

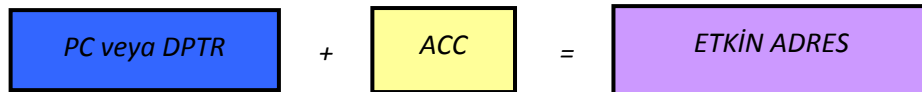




Şekil–3.1 Mutlak adreslemede sayfa sınırları ve komut içerisindeki adres bitlerinin görevleri.

### Sıralı Adresleme

Sıralı adresleme ile sadece program belleğinden okuma yapılabilir. Bu adres kipinin kullanılma amacı program belleğinde yazılı başvuru tablolarından bilgi okumaktır. 16 bitlik temel adresi içeren yazaç (DPTR veya program sayıcı olabilir) tablonun ilk adresini gösterir, akümülatör ise tablonun işlem yapılacak satırının numarasını içerir. Akümülatörün içeriği temel adrese eklenerek işlem yapılacak adres elde edilir. Sıralı adreslemenin diğer tipi de “durum jump” komutlarında kullanılır.



## 8051 Komut Kümesi

8051 komutları yaptıkları işlemlere göre 5 ana grupta incelenebilir. Bu kısımda ortak özellikleri dikkate alınarak komutlar incelenecektir. EK-A'da tüm komutlar alfabetik sıra ile ayrıntılı olarak incelenmiştir. Ek-b'de ise komutlar alfabetik sıraya göre kısaca açıklanmıştır.

- Veri aktaran komutlar.
- Aritmetik işlem yapan komutlar.
- Mantık işlem yapan komutlar.
- İkilik işlem yapan komutlar.
- Dallanma ve bağlanma komutları.

## Veri Aktarma Komutları

Veri aktarma komutları kaynakta yer alan veriyi hedefe kopyalarlar. Veri 8051'de bellek iç RAM, Dış RAM ve program belleği olmak üzere üç farklı kısma bölünmüştür. Bu üç belleğin veri, adres ve denetim yollarını kullanım şekilleri farklı olduğundan aynı işi yapan üç değişik veri aktarma komutu vardır. MOV komutu iç RAM'de, MOVX dış RAM'de ve MOVC program belleğinde veri aktarmak amacı ile kullanılır.



## İç Veri Belleğinde Veri Aktaran Komutlar

Tablo–3.1'de iç RAM'de kullanılan veri aktarma buyrukları kullanıldıkları adres modları ve işleme saykalları verilmiştir. Kaynak ve hedef iç RAM'deki herhangi bir bellek satırı olabilir. Birçok mikroişlemcide olan hedef veya kaynaktan bir tanesinin akümülatör olması şartı 8051'de yoktur.

Komut	Yaptığı İşlem	Doğr.	Dolaylı	Yazaç	İvedi
MOV A, <kaynak>	$A \leftarrow (\text{Kaynak})$	X	X	X	X
MOV <hedef>, A	$(\text{Hedef}) \leftarrow A$	X	X	X	
MOV <hedef>, <kaynak>	$(\text{Hedef}) \leftarrow (\text{Kaynak})$	X	X	X	X
MOV DPTR, #VERİ 16	DPTR 'ye 16 bit sabit yükle				X
PUSH <kaynak>	$(SP)+1 : ((SP)) \leftarrow \text{kaynak}$	X			
POP <hedef>	$(\text{Hedef}) \leftarrow ((SP)) : (SP) - 1$	X			
XCH A, <bayt>	$A \leftrightarrow (\text{bayt})$	X	X	X	
XCHD A, @Ri	A ve @Ri DDN'lerini yer değiştirir.		X		

Çizelge-3.1 İç RAM' de kullanılan veri aktarma buyrukları

8051'de yığın iç RAM'de yer alır ve dolaylı adreslenen yüksek değerli RAM bölgesine doğru genişletilebilir. PUSH komutu önce yığın gösterge yazacının içeriğini bir arttırır ve sonra belleğin içeriğini yığına kopyalar. POP komutu ise önce yığının en üstündeki veriyi alır ve doğrudan adresle belirtilen belleğe yazar, sonra yığın gösterge yazacını bir azaltır. Yüksek değerli 128 baytlık RAM bölgesi 8051'de yoktur, Eğer yığın gösterge yazacı bu bölgede bir adresi gösteriyorken yığına veri atılırsa bu veri kaybolur, veri çekilirse rasgele veridir. DPTR yazacının içeriği bir defada 16 bit ivedi adresleme ile istenilen değere kurulabilir. XCH A, <bayt> komutu akümülatör ile bellek satırının içeriğini karşılıklı yer değiştirecektir. XCHD A, @ Ri komutu da yer değiştirme yapar fakat yalnızca düşük değerli nibılların (dörtlü) yerlerini değiştirir.

Aktarma işleminin yapılacağı işlenenlerden birisi (hedef veya kaynak) akümülatör (A) ya da yazaç (Rn) ise komut iki bayt sürer. İkinci bayt diğer işlenenin doğrudan adresidir.

**Örnek 1:**

MOV R2, 25H ;25H nolu belleğin içeriğini R2 yazacına aktar.

MOV 32H, R2 ;R2 yazacının içeriği 32H nolu belleğe aktar.

**Örnek 2:**

MOV A, 32H ;32H nolu belleğin içeriğini Akümülatöre yükle.

MOV 65H, A ;Akümülatörün içeriğini 65H nolu belleğe yaz.

*Doğrudan adreslenen bir bellek gözüne doğrudan veri değeri aktarabilmek için gerekli komut üç bayt sürmektedir. Bellekten belleğe aktarma yapan komutlar da üç bayt uzunluktadır. Çünkü hem hedef, hem de kaynak doğrudan adreslenmektedir.*

#### Örnek 3:

*MOV 25H, #55H ;25H adresli belleğe 55H sayısı yüklenir.*

#### Örnek 4:

*MOV 25H, 55H ;55H adresli belleğin içeriği 25H adresli belleğe kopyalanır.*

*Paralel G/Ç kapıları (P0,P1,P2,P3) tümdevre üzerindeki Özel işlev yazaçları haritasında yer almaktadırlar ve dolayısıyla adreslenebilirler. Bir çok komut gibi veri aktarma komutları da paralel G/Ç kapıları üzerinde doğrudan işlem yapabilmektedir.*

#### Örnek 5:

*P2'den 3 değer okuyunuz ve önce bellekte saklayıp sonra P1'den giriş sırasının tersi sırayla yazın.*

*BAŞLA:*

*MOV 30H, P2 ;birinci değeri oku, bellekte sakla.  
MOV 31H, P2 ;birinci değeri oku, bellekte sakla.  
MOV 32H, P2 ;birinci değeri oku, bellekte sakla.  
MOV P1, 32H ;ters sırada P1'e yaz.  
MOV P1, 31H ;ters sırada P1'e yaz.  
MOV P1, 30H ;ters sırada P1'e yaz.*

#### Örnek 6:

*P1 portundan okunan veriyi 3 makine saykılı geciktirip P3 portuna yazan programı yazın.*

*MOV A, P1 ;P1'i oku  
NOP ;1 makine saykılı bekle  
NOP ;1 makine saykılı bekle  
NOP ;1 makine saykılı bekle  
MOV P3 ;P3'e yaz.*

*Komut yürütüldüğünde P1 portunun içeriği akümülatöre alınır. 3 makine saykılı bekletmek için üç adet NOP komutu kullanılır. Bu sürenin sonunda akümülatörün içeriği P3 portuna yazılır.*

**Örnek 7:**

Aşağıdaki komut ne yapar?

MOV DPTR, #4565h ;DPTR=4565H

Komut işletildiğinde 4565h sayısı DPTR'ye yüklenir. DPH=45h ve DPL=65h olur.

**Örnek 8:**

Yığın işaretleyici 45h' göstermektedir. İç RAM'de 045h adresli bellek satırı 34h, 044h adresli satırı bellek 77h, 43h adresli satırı bellek ise 25h içermektedir. Aşağıdaki komutlar işletildiğinde DPTR'nin içeriği ne olur?

POP DPH ;Yığının en üst satırını DPH'a aktar

POP DPL ;Yığının en üst satırını DPH'a aktar

POP SP ;Yığının en üst satırını DPH'a aktar

Komutlar yürütüldükten sonra DPTR=3477h ayarlanacaktır. Yığın işaretleyicinin içeriği ise 25h olacaktır.

**Dış Veri Belleğinde Veri Aktaran Komutlar**

Çizelge-3.2 de dış veri belleğinde kullanılan komutların listesi verilmiştir. Tüm komutlar dolaylı adresleme kipinde çalışır. Dış veri belleğinden sadece okuma ve yazma işlemleri yapılabilir. Dolaylı adreslemede adres belirteci olarak R0 ve R1 yazaçları ile DPTR yazacı kullanılır. 256 baytlık RAM belleğin yeterli olduğu uygulamalarda sadece P0 portunun adres yolu olarak kullanılmasını olanak sağlamak amacıyla 8 bit adres kullanan MOVX komutuna 8051 komut kümesinde yer verilmiştir.

Adres	Komut	Yaptığı işlem
8 bit	MOVX A, @Ri	Ri' nin gösterdiği satırı okur.
8 bit	MOVX @Ri, A	Ri'nin gösterdiği satıra yazar.
16 bit	MOVX A, @DPTR	DPTR'nin gösterdiği satırı okur.
16 bit	MOVX @DPTR, A	DPTR'nin gösterdiği satıra yazar.

Çizelge - 3.2 Dış RAM' de kullanılan aktarma buyrukları

MOVX A, @Ri veya MOVX @Ri, A komutları kullanıldığında P2 portu serbest giriş/çıkış hattı olarak kullanılabilir. 256 baytlık dış RAM' in yeterli olmadığı uygulamalarda 16 bitlik veri göstericisi DPTR yazacı kullanılır. MOVX A, @DPTR veya MOVX @DPTR, A komutları kullanıldığında P2 portu adresin yüksek değerlikli baytını taşıdığı için başka amaçla kullanılamaz.

#### Örnek 9:

R1=78H olduğuna göre aşağıdaki komut yürütüldüğünde akümülatöre yüklenecek sayı ne olur?

MOVX A, @R1 ;R1'in gösterdiği bellek satırını oku.

Dış veri belleğinin 78h numaralı satırın içeriği akümülatöre yüklenecektir.

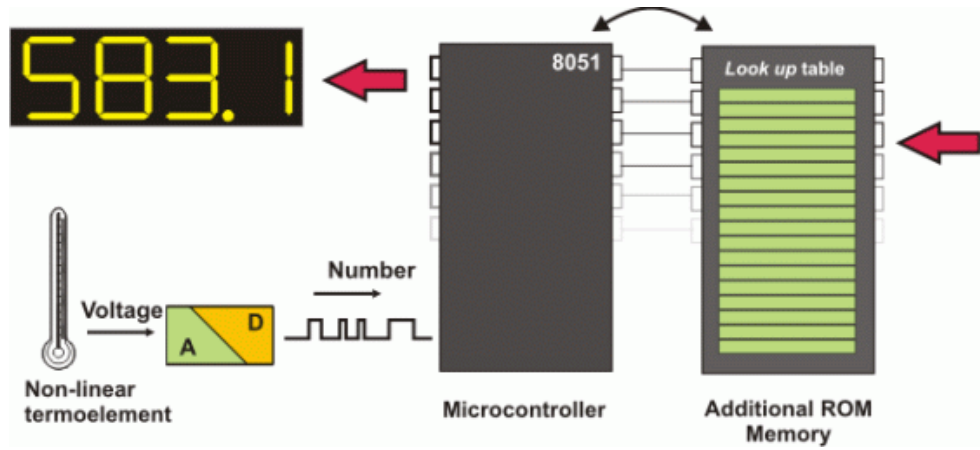
#### Program Belleğinden Veri Okuyan Komutlar

Program belleğine alt programlar tarafından kullanılan sabit değerler tablosu veya kod dönüşüm tabloları kalıcı bir şekilde yazılabilir. Belirli bir sıraya göre yazılmış bu sabit değerlerin oluşturduğu tabloya başvuru tablosu adı verilir. Bu tablolar birkaç alt program tarafından sınırsız olarak kullanılabilir. Program belleğine yazma yapılamadığından sadece okuma komutu vardır. Çizelge-3.3 de program belleğinden okuma yapan komutların listesi verilmiştir. Tablonun başlangıç adresini belirtmek için DPTR veya PC kullanılır. Tablo satır numarası ise sadece akümülatör içerisine yazılabilir. Tablodan yüklenecek verinin adresi DPTR veya PC ile Akümülatörün içeriği toplanarak elde edilir. Tablo değeri Akümülatöre yüklenir ve tablo satır numarası kaybolur. Bu durum genellikle sorun çıkarmaz, çünkü veri yüklendikten sonra bir daha satır numarasına gereksinim duyulmaz. Program belleğinden bilgi okumada dış RAM' de olduğu gibi hedef sadece Akümülatör olabilir.

Komut	Yaptığı işlem
MOVC A, @A+DPTR	A+DPTR adresli program belleğinin içeriğini okur.
MOVC A, @A+PC	A+PC adresli program belleğinin içeriğini okur.

Çizelge - 3.3 Program belleğinden look-up tablolarını okuyan buyruklar.





Şekil-3.1 Başvuru tablosunun kullanımı

### ***DPTR İle Başvuru Tablosu Yapmak***

*MOVC A, @A+DPTR komutu bir bayt uzunluğundadır. Bu komut işletildiğinde 16-bitlik DPTR yazacına 8 bitlik akümülatörün içeriği toplanır ve elde edilen 16 bitlik sayı program belleğinden okunacak verinin adresidir. Bu adresteki veri akümülatöre okunur. İşlem sonunda akümülatörün içeriği değişirken DPTR'nin içeriği korunur. Bu komut program belleğine kaydedilmiş dizi veya tablo değerlerinin okunması amacıyla kullanılır. Bunun için tablo veya dizinin başlangıç adresi DPTR yazacına okunacak verinin başlangıçtan uzaklığı (sıra numarası) akümülatöre yazılır.*

### **PC İle Başvuru Tablosu Yapmak**

*MOVC A, @A+PC komutu bir bayt uzunluktadır. Bu komut işletildiğinde 16-bitlik PC ile 8 bitlik akümülatörün içeriği toplanır ve elde edilen 16 bitlik sayı program belleğinden okunacak verinin adresidir. Bu adresteki veri akümülatöre okunur. İşlem sonunda akümülatörün içeriği değişirken PC'nin içeriği korunur. MOVC komutu PC veya DPTR ile özdeş işlemleri gerçekleştiriyor gibi görünse de PC ile yapıldığında işler bir parça karmaşık hal almaktadır. Bunun nedeni PC'ye istenilen bir tablo tabanı değerini yerleştirmenin olanaksız olmasıdır. Bu nedenle ileri düzey bir programcı değilseniz tabloya DPTR yazacını kullanarak ulaşmayı deneyin.*

### ***Örnek 10:***

*P1 portundan okuduğu (0-9 arası) sayının karesini alıp P2'ye yazan programı yazın.*

```
ORG 0           ;Program buradan başlar
MOV DPTR, #TAB1 ;Tablonun başlangıç adresini belirle.
```

```

MOV P1, #0FFH      ;P1'i giriş yapmak için tüm hatlarına 1 yaz.
L01: MOV A,P1        ;P1'i oku
      MOVC A,@A+DPTR  ;Okunan sayının karesini al
      MOV P2,A        ;Karesini P2'ye yaz.
      SJMP L01        ;Sürekli yap.
ORG 300H            ; Tablo buradan başlıyor.
TAB1: DB 0,1,4,9,16,25,36,49,64,81
END

```

## Aritmetik İşlem Komutları

Aritmetik işlem buyrukları ve kullanıldıkları adres modları çizelge-3.4'te verilmiştir. Toplama komutu 8051'de eldeli ve eldesiz olmak üzere iki türdür. Eldesiz toplamada akümülatör ve belirtilen bayt toplanır sonuç akümülatöre yazılır. İşlem sonucunda oluşan elde bayrağı kurar, elde kurulu ise sonuç 9 bit demektir. Eldeli toplamada akümülatör belirtilen baytı ve eldenin içeriğini toplar sonucu akümülatöre yazar. Sonuç 8 bitten büyükse elde bayrağı kurulur aksi halde temizlenir. ADD A,<bayt> komutu aşağıdakiler gibi yazılabilir.

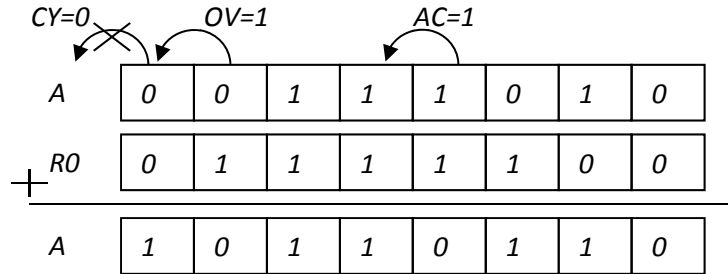
Kısa ad	İşlem	Doğr.	Dolaylı	Yazaç	İvedi
ADD A,<bayt>	A=A+<bayt>	X	X	X	X
ADDC A,<bayt>	A=A+<bayt>+C	X	X	X	X
SUBB A,<bayt>	A=A-<bayt>-C	X	X	X	X
INC A	A=A+1	Akümülatöre özel			
INC <bayt>	<bayt>=<bayt>+1	X	X	X	
INC DPTR	DPTR=DPTR+1	Veri göstericisine özel			
DEC A	A=A-1	Akümülatöre özel			
DEC <bayt>	<bayt>=<bayt>-1	X	X	X	
MUL AB	A.B → A <sub>7-0</sub> , B <sub>8-15</sub>	Yazaca Özel			
DIV AB	A/B A=bölüm, B=kalan	Yazaca Özel			
DA A	Onlu sonuca ayarla	Akümülatöre özel			

Çizelge-3.4 8051 Aritmetik işlem buyrukları.

**Örnek 11:**

Akümülatörün içeriği 3Ah, R0'ın içeriği 7Ch olduğuna göre aşağıdaki komut işletildikten sonra akümülatör, elde, taşma ve yardımcı elde bayraklarının içeriklerini belirleyin.

ADD A, R0



Akümülatör ile R0 toplandığında 3 bitlerin toplamından bir elde oluşur ve Yardımcı El de bayrağı kurulur. 7 bitlerin toplamından elde oluşmadığı için elde bayrağı temizlenir. Altıncı bitlerin toplamından elde oluştuğu için Taşma Bayrağı (OV) kurulur. Bu sayılar işaretli sayı olarak kabul edilirse her ikisinin işaret bitleri (yedinci bitleri) 0 olduğu için pozitif sayıdır. İki pozitif sayının toplamından negatif sayı elde edilmiştir. Sonucun işaret biti doğru değildir, aslında sonuç negatif değildir, toplamın büyüklüğü 7 bite sığmadığı için taşma olmuştur. Bu taşmada Taşma Bayrağını (OV) kurmuştur. Sonucun doğruluğu Elde Bayrağı (CY) ve Taşma Bayrağı (OV) birlikte değerlendirilerek belirlenir. Bu örnekte büyüklük doğrudur, fakat işaret biti hatalıdır.

İKO sayıların toplanmasında kullanılan farklı bir komut yoktur, aynı toplama komutları kullanılır. Toplama komutu 3. bitlerin toplanmasında elde oluşur ise AC bayrağını kurar. Eğer İKO toplama yapılmak isteniyor ise toplanacak 8 bit değişkenlerin İKO formunda yazılı olması gerekir. Toplama yapıldıktan sonra DA A komutu işletilerek akümülatörün içeriği İKO'ya dönüştürülür. DA A komutu AC bayrağı kurulu veya düşük değerli 4 bit 9'dan büyük ise bu kısma 6 toplar. İşlemin sonunda elde oluşur ise yüksek değerli kısma yansıtılır. Elde bayrağı kurulu veya yüksek değerli kısım 9'dan büyük ise 6 toplanır, elde oluşur ise elde bayrağı kurulur, oluşmaz ise temizleme yapılmaz. DA A komutu diğer komutlar sonrası dönüşümü doğru olarak yapmaz.

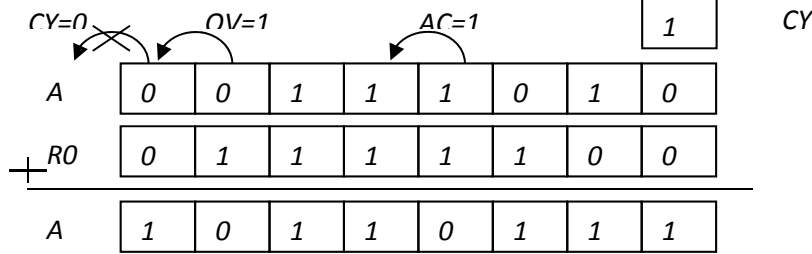
**Örnek 12:**

Akümülatörün içeriği 56h, R3 yazacının içeriği 67h bu sayılar İKO 56 ve 67 sayılarını göstermektedirler. Bu iki sayıyı doğru olarak toplayan programı yazın.

Toplanacak sayılar İKO olarak kodlu oldukları için önce toplama işlemi yapılır sonra DA A komutu yürütülerek sonuç düzenlenebilir.

ADD A, R3

DA A



Önce bu iki sayı toplanır, toplama sonucu 10111101B olur. Elde Bayrağı (CY) ile Yardımcı Elde Bayrağı (AC) temizlenir. İKO'ya uyarla komutu yürütüldüğünde düşük değerli nibil 92dan büyük olduğu için 6 eklenir. Akümülatörün içeriği 10110011B olur. Elde Bayrağı (CY) kurulur. İkinci aşamada hem Elde Bayrağı (CY) kuruludur hem de yüksek değerli nibil geçersiz İKO sayıdır, yüksek değerli nibilıda 6 eklenerek İKO'ya uyarlanır. Akümülatörün yeni içeriği 00100011B (23h) olur. İşlem sonunda Elde Bayrağı (CY) kurulur, çünkü sonuç 123'tür.

İKO sayılar toplama yöntemi ile 01h ve 99h sayıları toplanarak bir artırılabilir veya azaltılabilir. 30H sayısını 99h sayısını toplar ve İKO'ya uyarlama komutunu yürütürsek bir azaltmış oluruz.

ADD A, #99

DA A

İKO'ya uyarlama sonrası oluşan elde alır. 30+99=129 elde atıldığında 29 elde edilir.

**Örnek 13:**

Akümülatörün içeriği 3Ah, R0'ın içeriği 7Ch, Elde Bayrağı (CY) içeriği kurulu olduğuna göre aşağıdaki komut işletildikten sonra akümülatör, elde, taşıma ve yardımcı elde bayraklarının içeriklerini belirleyin.

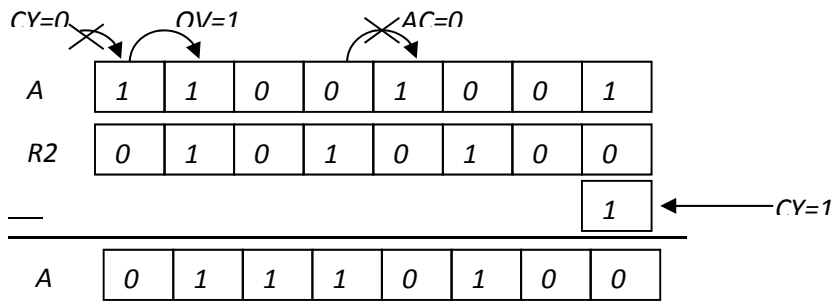
ADD A, R0

**Örnek 14:**

Akümülatörün içeriği 0C9h, R2'in içeriği 54h ve Elde Bayrağı (CY) kurulu olduğuna göre aşağıdaki komut işletildiğinde akümülatörün ve Elde Bayrağı (CY) durumunu belirleyin.

**SUBB A, R2**

Komut işletildiğinde R2'nin içeriği ve elde bayrağı kurulu olduğu için akümülatörün içeriğinden çıkarılır. Bit 3'lerin çıkarılması sırasında bit 4'ten borç alma gereksini olmadığı için AC bayrağı temizlenmiştir. Bit 6'ların çıkarılması sırasında bit 7'den borç alınmıştır ve bunun sonucunda Taşma Bayrağı (OV) kurulmuştur. Bit 7'lerin çıkarılmasında borç gerekli olmamıştır ve Elde Bayrağı (CY) temizlenmiştir. İşlem sonucu 74H olarak akümülatörde elde edilmiştir.



İç RAM'de her bellek satırının içeriği INC ve DEC komutları ile akümülatör kullanılmadan arttırılabilir veya azaltılabilir. Bu komutlar bayrakları etkilemez. INC DPTR komutu ise 16 bit DPTR yazacının içeriğini bir artırır. 8051'de çarpma ve bölme komutları ikişer bayt girdileri yine ikişer bayt sonuçlara dönüştürmektedirler. Her iki komutta sadece A ve B akümülatörleri üzerinde tanımlanmıştır. 8051 komut kümesinde tüm diğer komutlar 1-2 makine saykılı sürerken çarpma ve bölme komutları 4 makine saykılı (48 osilatör periyodu) sürmektedir. A\*B işleminin sonucunun düşük değerli Baytı A'da, yüksek değerli baytı B'de yer alır. Sonuç her zaman iki bayta yerleştirilir (A ve B). Çarpımın bir bayttan büyük olduğu (B≠0) taşma bayrağı kurularak belirtilir (OV=1). Değilse (B=0) taşma bayrağı temizlenir (OV=0). Çarpma işlemi sonucu da C her zaman sıfırlanır, AC etkilemez, P ise alışılmış ödevini yapar.

**Örnek 15:**

Akümülatör içeriği 75h, B yazacının içeriği 60h olduğunda aşağıdaki komut yürütülürse akümülatör ve B yazacının yeni içeriklerini belirleyin. OV bayrağının durumunu belirleyin.

MUL AB

Komut yürütüldüğünde akümülatörün içeriği ile B yazacının içeriği çarpılır YDB B yazacına yazılır. İşlem sonucu 255'ten büyük olduğu için OV bayrağı kurulur.

A=E0,      B=2B,      OV=1

**Örnek 16:**

Akümülatör içeriği 255, B yazacının içeriği 30 olduğuna göre aşağıdaki komut yürütüldükten sonra bu yazaçların içeriklerini belirleyin.

DIV AB

Komut yürütüldüğünde akümülatörün içeriği 08h (bölüm), B yazacının içeriği 0Fh (Kalan) olacaktır.

**Örnek 17:**

0FAH ile 05H'yi çarpan programı yazın.

MOV A, #0FAH      ;birinci sayıyı A'ya al.

MOV B, #05H      ;ikinci sayıyı B'ye al.

MUL AB      ;çarp.

8051' de bölme komutu tek bir biçimde gerçekleştirilebilmektedir. Bölünecek sayı A'ya bölen sayı B'ye yazılmalıdır. A/B işleminin sonucunda bölüm A'ya, kalan ise B'ye yerleştirilir. Sayıların işaretli sayılar olduğu varsayılmaktadır. Bölüm, sıfıra bölme dışında, bir bayt büyüklüktedir. Sıfıra bölme durumu OV=1 ile belirtilmektedir. Bölme işlemi sonucunda C her zaman sıfırlanır, AC etkilenmez, P ise alışılmış ödevini yapar.

## Mikrodenetleyiciler 8051 Uygulamaları

**Örnek 18:**

8CH değerini 6'ya bölen programı yazın.

```
MOV A, #8CH      ;Birinci sayıyı A'ya al.
MOV B, #6        ;İkinci sayıyı A'ya al.
DIV AB           ;Böl, bölüm A'ya kalan B'ye yazılır
```

**Örnek 19:**

R1 yazacı 65h, R6 yazacı 77h, 65h ve 64h adresli RAM belleklerin içerikleri 00h'dir. Aşağıdaki komutlar işletildikten sonra belleklerin ve yazaçların içeriklerini belirleyin.

```
DEC @R1
DEC R1
DEC @R1
```

DEC R6 Birinci komut yürütüldüğünde R1'in gösterdiği belleğin içeriği bir azaltılacaktır. 65h nolu bellek satırının içeriği 00h idi azaltma sonrası FFh olacaktır. İkinci komut yürütüldüğünde R1 yazacının içeriği 65h'den 64h'ye azaltılacaktır. Üçüncü komut birinci komutta olduğu gibi R1'in gösterdiği bellek satırının içeriği bir azaltılacaktır. R1 yazacı bu defa 64h nolu bellek satırını göstermektedir, içeriği 00h'dir azaltıldığında bu belleğin içeriği FFh olacaktır. Dördüncü komut yürütüldüğünde R6'nın içeriği bir azaltılarak 76h olur.

## Mantık İşlem Buyrukları

Mantık işlem buyrukları ve kullanıldıkları adres modları çizelge-3.5'te verilmiştir. 8051 4 temel boolean işlemi gerçekler, birinci değişkenin aynı numaralı biti ile ikinci değişkenin aynı numaralı biti boolean işlemine tabi tutulur ve birinci değişkenin aynı numaralı bitine yazılır. İşlemin yapılabilmesi için değişkenlerden birinin akümülatörde olma zorunluluğu yoktur. Şekil-3.2'de AND işleminin yapılışı gösterilmiştir. ANL A, <bayt> komutu aşağıdakiler gibi yazılabilir.

Kısa ad	Yaptığı işlem	Doğr.	Dolaylı	Yazaç	İvedi
ANL A, <bayt>	$A = A \wedge \text{<bayt>}$	X	X	X	X
ANL <bayt>, A	$\text{<bayt>} = \text{<bayt>} \wedge A$	X			
ANL <bayt>, #VERİ	$\text{<bayt>} = \text{<bayt>} \wedge \text{VERİ}$	X			
ORL A, <bayt>	$A = A \vee \text{<bayt>}$	X	X	X	X
ORL <bayt>, A	$\text{<bayt>} = \text{<bayt>} \vee A$	X			
ORL <bayt>, #VERİ	$\text{<bayt>} = \text{<bayt>} \vee \text{VERİ}$	X			
XRL A, <bayt>	$A = A \oplus \text{<bayt>}$	X	X	X	X
XRL <bayt>, A	$\text{<bayt>} = \text{<bayt>} \oplus A$	X			
XRL <bayt>, #VERİ	$\text{<bayt>} = \text{<bayt>} \oplus \text{VERİ}$	X			
CLR A	$A = 00H$	A'ya özel			
CPL A	$A = A'$	A'ya özel			
RL A	Bir bit sola döndür.	A'ya özel			
RLC A	C üzerinden sola döndür.	A'ya özel			
RR A	Bir bit sağa döndür.	A'ya özel			
RRC A	C üzerinden sağa döndür.	A'ya özel			
SWAP A	Nibbleları yer değiştir.	A'ya özel			

Çizelge - 3.5 8051 mantık işlem buyrukları.

ANL A, 7FH (doğrudan adresleme )

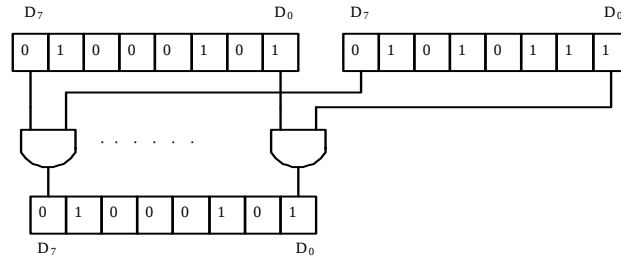
ANL A, @R0 (dolaylı adresleme )

ANL A, R7 (yazaç adresleme)

ANL A, #127 ( ivedi adresleme )

Akümülatörde yapılan tüm mantık işlem buyrukları 1  $\mu S$  sürerken diğer buyruklar 2  $\mu S$  sürer. Boolean işlemleri akümülatör kullanılmadan da veri belleğinin herhangi bir satırında gerçekleştirilebilir. Örneğin XRL P1 #veri buyruğu kısa yoldan portun giriş iken çıkışa dönüştürülmesini yada çıkış iken girişe dönüştürülmesine olanak sağlar. Elde üzerinden döndürme komutları dışındaki komutlar bayrakları etkilemez.





Şekil-3.2 mantık VE işleminin gerçekleştirilmesi.

### Örnek 20:

Akümülatörün içeriği 56h R1'in içeriği 9Ah olduğuna göre aşağıdaki komut işletildiğinde akümülatörün içeriğini belirleyin.

ANL A, R1

Aşağıda gösterildiği gibi bitler birbirinden bağımsız olarak VElenmiştir. Her hangi bir bayrak işleminden etkilenmemiştir.

A	0	1	0	1	0	1	1	0
R1	1	0	0	1	1	0	1	0
A	0	0	0	1	0	0	1	0

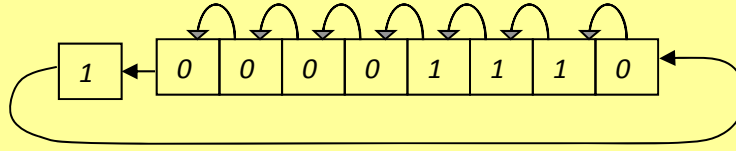
### Örnek21:

Akümülatörün içeriği 87h ve Elde Bayrağı (CY) temiz ise aşağıdaki komut işletildikten sonraki içeriğini belirleyin. RLC A

Akümülatörün içeriği

1	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

İşlem sonrası akümülatörün içeriği.



Komut işletildiğinde akümülatörün içeriği bir bir sola döndürülecektir.  $D_7$  ise  $D_0$ 'a yerleşecektir ve akümülatörün yeni içeriği 0Eh olacaktır ve Elde Bayrağı (CY) kurulacaktır.

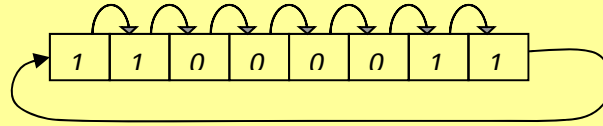
Örnek:

Akümülatörün içeriği 87h ise aşağıdaki komut işletildikten sonraki içeriğini belirleyin. RRA

Akümülatörün içeriği.

1	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

İşlem sonrası akümülatörün içeriği.



Komut işletildiğinde akümülatörün içeriği bir bir sola döndürülecektir.  $D_0$  ise  $D_7$ 'ye yerleşecektir ve akümülatörün yeni içeriği C3h olacaktır.

Örnek 22:

Akümülatörün içeriği 87h ve Elde Bayrağı (CY) temiz ise aşağıdaki komut işletildikten sonraki içeriğini belirleyin.

RLC A

Komut işletildiğinde akümülatörün içeriği bir bir sola döndürülecektir.  $D_7$  ise  $D_0$ 'a yerleşecektir ve akümülatörün yeni içeriği 0Eh olacaktır ve Elde Bayrağı (CY) kurulacaktır.

**Örnek 23:**

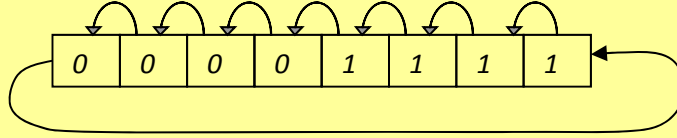
Akümülatörün içeriği 87h ise aşağıdaki komut işletildikten sonraki içeriğini belirleyin.

**RL A**

Akümülatörün içeriği.

1	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

İşlem sonrası akümülatörün içeriği.



RLC A komut işletildiğinde akümülatörün içeriği bir bir sola döndürülecektir. D7 ise D0'a yerleşecektir ve akümülatörün yeni içeriği 0Fh olacaktır.

**Örnek 24:**

Akümülatörün içeriği 55h R1 yazacının içeriği AAh olduğunda aşağıdaki komut işletildiğinde Akümülatörün içeriğini belirleyin.

**XRL A, R1**

A	0	1	0	1	0	1	0	1
R2	1	0	1	0	1	0	1	0
A	1	1	1	1	1	1	1	1

SWAP A komutu akümülatörün düşük değerli 4 bitini ile yüksek değerli 4 bitini sıralı bir şekilde yer değiştirir. Kod dönüşümü için kullanışlı bir komuttur.

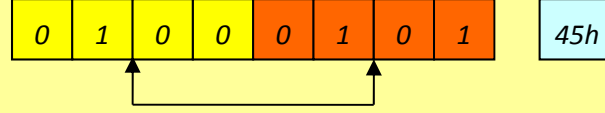


**Örnek 25:**

Aşağıdaki komutlar işletildikten sonra Acc'nin içeriğini belirleyin?

Mov A, #45h; Acc'ye 45h sayısını yükle  
Swap A ;Nibbılları yer değiştir.

Birinci komut işletildiğinde Acc'nin içeriği



İkinci komut işletildiğinde Acc'nin içeriği



## Boolean İşlem Yapan Komutlar

8051 bit işlem yapan boolean işlemcisi içerir. Bit adresleme alt 128 baytlık bölgede 128 adet ve SFR bölgesinde 128 adet olmak üzere toplam 256 adet bellek hücresinde doğrudan adresleme modunda kullanılır. Bit işlem yapan komutlarda akümülatörün görevi elde bayrağına verilmiştir. Bit işlem yapan komutlar bitin içeriğini değiştiren, mantık işlem yapan ve test edip karar veren komutlar olmak üzere üç ana gruba ayrılır. Çizelge-3.6 da bit adreslemede kullanılan buyrukların listesi verilmiştir.

Bitin içeriğini değiştiren komutlar Mov, Setb, Cpl ve Clr komutlarıdır. Bu komutlardan Mov (Muv diye okunur) kaynak bitin içeriğini hedef bite kopyalar, setb komutu adresi belirtilen bitin içeriğini kurar (mantık 1 yapar). Clr komutu adresi belirtilen biti temizler (mantık 0 yapar), Cpl ise adresi belirtilen bitin 1'e tümleyenini (değilini) alır.

MOV C, bit ;Adresi belirtilen bitin içeriğini C'ye aktar.

MOV Bit, C ;C'nin içeriğini adresi belirtilen bite aktar.

Bit işlem yapan mantık komutları VE ile VEYA işlemleri olmak üzere iki çeşittir fakat iki adet VE işlemi vardır. Bunlardan birincisi elde bayrağının içeriği ile adresi

## Mikrodenetleyiciler 8051 Uygulamaları

belirtilen bitin içeriğini VE işlemi yapar ve sonucu elde bayrağına yazar. Bit işlem yapan mantık komutlarında birinci işlenen ve işlem sonucu elde bayrağında yer almak zorundadır. İkinci VE işlemi yapan komut ise elde bayrağı ile adresi belirtilen bitin içeriğinin değilini VE işlemi yapar ve sonucunu elde bayrağına yazar.

Kısa ad	Yaptığı işlem	Süresi
ANL C, bit	$C = C \wedge \text{bit}$	2
ANL C, /bit	$C = C \wedge / \text{bit}$	2
ORL C, bit	$C = C \vee \text{bit}$	2
ORL C, /bit	$C = C \vee / \text{bit}$	2
MOV C, bit	$C = \text{bit}$	1
MOV bit, C	$\text{Bit} = C$	2
CLR C	$C = 0$	1
CLR bit	$\text{Bit} = 0$	1
SETB C	$C = 1$	1
SETB bit	$\text{Bit} = 1$	1
CPL C	$C = /C$	1
CPL bit	$\text{Bit} = / \text{bit}$	1
JC bağıl adres	$C = 1$ ise dallan	2
JNC bağıl adres	$C = 0$ ise dallan	2
JB bit, bağıl adres	$\text{Bit} = 1$ ise dallan	2
JNB bit, bağıl adres	$\text{Bit} = 0$ ise dallan	2
JBC bit, bağıl adres	$\text{Bit} = 1$ ise biti temizle ve dallan	2

Tablo–3.6 Boolean buyrukları.

ANL C, bit ;  $C = C \wedge \text{bit}$   
 ANL C, /bit ;  $C = C \wedge / \text{bit}$

VEYA işlemi yapan ORL komutunu kullanımı da ANL komutu ile aynıdır. Birinci ORL komutu elde bayrağının içeriği ile adresi belirtilen bitin içeriğini VEYA işlemi yapar ve sonucu elde bayrağına yazar. İkinci ORL komutu ise elde bayrağı ile adresi belirtilen bitin içeriğinin değilini VEYA işlemi yapar ve sonucunu elde bayrağına yazar.

ORL C, bit ; C=C V bit  
 ORL C, /bit ; C=C V /bit

Bitin içeriğini test edip karar veren komutlar beş adettir. Komutların hepsi adresi belirtilen birtin içeriğini komutun belirttiği değere karşılaştırır, eğer sonuç doğru ise belirtilen adrese dallanma gerçekleşir. Eğer yanlış ise dallanma gerçekleşmez, 8051 bir sonraki komutu yürütmeye devam eder. Komut yazımında dallanılacak adres doğrudan adres olarak değil bağıl adres olarak verilmelidir. Birçok assmbler programı bu bağıl adresi kendisi hesaplar bizim etiket ile dallanılacak adresi belirtmemiz yeterlidir. Eğer belirtilen etiketin komuta olan uzaklığı +127 adres veya -128 adresden fazla ise program işletilirken hata oluşur.

JC son;Eğer elde bayrağı "1" ise dallan.

JC komutu elde bayrağının 1 olup olmadığını test eder. Eğer sonuç doğru ise son etiketinin bulunduğu adrese dallanır. Yanlış ise bir sonraki komuttan devam edilir.

JNC son ;eğer elde bayrağı temiz ise dallan.

JNC son komutu, elde bayrağının "0" olup olmadığını test eder. Eğer yanıt doğru ise son olarak belirtilen satıra dallanır. Yanlış ise bir sonraki komuttan devam edilir.

JB bit, son ;Adresi belirtilen bitin içeriği "1" ise dallan.

JNB bit, geri ;Adresi belirtilen bitin içeriği "0" ise dallan.

JB komutu adresi belirtilen bitin "1" olup olmadığını test eder. Eğer yanıt doğru ise "son" olarak belirtilen satıra dallanır. Yanlış ise bir sonraki komuttan devam edilir. JNB komutu adresi belirtilen bitin "0" olup olmadığını test eder. Eğer yanıt doğru ise "geri" olarak belirtilen satıra dallanır. Yanlış ise bir sonraki komuttan devam edilir.

JBC bit, ileri ; Adresi belirtilen bitin içeriği "1" ise önce bitin içeriğini temizler sonra "ileri" olarak belirtilen adrese dallan.

JBC bit, ileri komutu iki komutun birleşik halidir. Önce adresi belirtilen bitin içeriğini test eder. Eğer yanıt doğru ise bitin içeriğini temizler ve "ileri" olarak adlandırılan satıra dallanır. Yanlış ise bitin içeriği değiştirilmeden bir sonraki satırdan devam eder. AYRICALIKLI VEYA işlemi yapan komut yoktur diğer komutlardan türetilmelidir.

#### Örnek 26:

Aşağıdaki komutlar işletildiğinde PC sayacının içeriği ne olur? Belirleyin.

JNB P3.2, ileri

## Mikrodenetleyiciler 8051 Uygulamaları

JNB ACC.3, geri

Birinci komut işletildiğinde eğer Port 3'ün 2 numaralı hattı 0 ise program sayacının içeriği "ileri" olarak adlandırılan adresi içerir. 0 değilse ikinci komut işletilir. Bu komut işletildiğinde akümülatörün 3 numaralı biti eğer 0 ise PC'nin içeriği "geri" olarak adlandırılan satırın adresini içerir.

#### Örnek 27:

Akümülatörün içeriği 10111011B olduğuna göre aşağıdaki komutlar işletildiğinde program sayacının içeriği ne olur.

JB ACC.5, İLERİ

JB ACC.2, GERİ

Birinci komut işletildiğinde 5 bitin durumuna göre dallanma olur veya olmaz. Beşinci bit 1 olduğuna göre dallanma gerçekleşir. Program sayacının içeri İLERİ olarak adlandırılmış satırın adresi olur. Beşinci bit 0 olsaydı, dallanma olmayacak bir alt satırdaki komut işletilecekti. Bir alt satırda ise ikinci bite bakan bir dallanma komutu vardır ikinci bit 0 olduğu için dallanma gerçekleşmeyecek ve bir alt satırdaki komut işletilmeye başlanacaktır.

#### Örnek 28:

P1.3 =1, ACC.5=0 ve OV=1 bu koşullardan bir tanesi yerine gelirse elde bayrağını kuran programı yazın.

MOV C, P1.3

ORL C, /ACC.5

ORL C, OV

Komut yürütüldüğünde P1.3 hattının değeri elde bayrağına alınır. İkinci komutta akümülatör 5 numaralı bitinin değili ile VEYAlanır. Üçüncü komutta ise OV bayrağı ile elde bayrağı VEYAlanır.

#### Örnek 29:

Akümülatörün içeriği 10111011B olduğuna göre aşağıdaki komutlar işletildiğinde program sayacının içeriği ne olur.

JB ACC.5, İLERİ

JB ACC.2, GERİ

Birinci komut işletildiğinde 5 bitin durumuna göre dallanma olur veya olmaz. Beşinci bit 1 olduğuna göre önce bu bit 0 yapılır ve sonra dallanma gerçekleşir. Program sayacının içeriği İLERİ olarak adlandırılmış satırın adresi olur. Dallanma sonrası akümülatörün içeriği 10011011B olacaktır. Beşinci bit 0 olsaydı, dallanma olmayacak bir alt satırdaki komut işletilecekti. Bir alt satırda ise ikinci bite bakan bir dallanma komutu vardır ikinci bit 0 olduğu için dallanma gerçekleşmeyecek ve bir alt satırdaki komut işletilmeye başlanacaktır.

#### Örnek 30:

Elde bayrağı kurulu iken aşağıdaki komutlar işletilirse program sayacının içeriği ne olur?

JC İLERİ

CPL

JC GERİ

#### Örnek 31:

Aşağıdaki komutlar işletildiğinde PC sayacının içeriği ne olur? Belirleyin.

JNC ileri

CPL

JNC, geri

Birinci komut işletildiğinde eğer elde bayrağı 0 ise program sayacının içeriği “ileri” olarak adlandırılan adresi içerir. 0 değilse ikinci komut işletilir, bu komut elde bayrağının içeriğinin 1’e tümleyenini alır”. Bu komut işletildiğinde akümülatörün 3 numaralı biti eğer 0 ise PC’nin içeriği “geri” olarak adlandırılan satırın adresini içerir bu programda ya birinci komutta dallanma olur veya üçüncü komutta başka seçenek yoktur.

#### Örnek 32:

Bit adres 07h’nin içeriği 1’dir aşağıdaki komut işletildiğinde yeni içeriği ne olur.

CLR 00h

## Mikrodenetleyiciler 8051 Uygulamaları



Bitin içeriği 0 olur.

Örnek 33:

Aşağıdaki birinci komut elde bayrağını ikinci komut Akümülatörün 5 numaralı bitini kurar.

SETB C  
SETB ACC.5

## Bağlanma Komutları

Program akışını koşul gerektirmeden değiştirmek için kullanılan bağlanma komutu üç farklı adresleme kipinde kullanılır. Bağlanma komutlarının listesi çizelge-3.7 verilmiştir.

Kısa ad	Yaptığı işlem
L JMP 16 adres	Belirtilen uzun adrese bağlanır.
AJMP 11 adres	Belirtilen mutlak adrese bağlanır.
SJMP Bağlı adres	Belirtilen bağlı adrese bağlanır.
JMP @A+DPTR	DPTR + A adresine bağlanır.
ACALL 11 bit adres	Belirtilen adresteki alt programı çağırır.
LCALL 16 bit adres	Belirtilen adresteki alt programı çağırır.
RET	Alt programdan geri döner.
RETI	Kesme alt programından geri döner.
NOP	İşlem yapmadan 1 makine saykılı zaman geçirir.

Çizelge-3.7 Bağlanma komutları.

SJMP komutunun adresi 8 bitlik bağlı kayıklıdır. —128 ile +127 arasında değerler olabilir. Negatif kayıklık geri doğru bağlanmaya sebep olur. Temel adres SJMP komutundan sonraki komutun başlangıç adresidir. Bu komut 2 bayttır. Birinci baytı opkod ikinci baytı da kayıklık değerinden oluşur. LJMP komutu 16 bit uzunluğunda doğrudan adres kullanır. Buyruk 3 bayt uzunluğundadır, birinci baytı opkod, ikinci ve üçüncü baytı da adres bilgisini içerir. Buraya yazılacak olan adres program

belleğindeki 64 Kbaytlık alandan herhangi bir satır adresi olabilir. AJMP komutu 11 bit adres içerir. Buyruk 2 bayttan oluşur, birinci baytı opkod ve adresin yüksek değerli 3 bitini, ikinci baytı ise adresin düşük değerli kısmını gösterir. Komut yürütüldüğünde PC'nin düşük değerli 11 bitine komutun adresi yazılır yüksek değerli 5 bit değişmez. Bu durumda hedef adres eğer program belleğini 2 Kbaytlık bloklara ayırırsak o anda bulunulan blok içerisinde olabilir. Birçok assembler programları JMP ismini tanır ve uygun adresleme modunu kendisi belirler.

JMP @A+DPTR komutu koşullu dallanma benzeri bir bağlanma komutudur. DPTR bağlanma yapılacak yerin sıfır nolu satırının adresini içerir. Akümülatör ise dallanılacak olan satırın bu noktaya olan uzaklığını belirler. Komut yürütüldüğünde Akümülatör ile DPTR'nin içeriği toplanır ve etkin adres bulunur. Bu komut sadece yüklenen sayıya göre bir adrese dallanılacak ise kullanılır.

#### Örnek 34:

```
MOV DPTR, #TABLO
MOV A, SIRA NUMARASI
JMP @A+DPTR
.....
.....
TABLO:
AJMP DURUM 0
AJMP DURUM 1
AJMP DURUM 2
.....
.....
AJMP DURUM 255
```

Bu program yürütüldüğünde DPTR'ye bağlanma tablosunun adresi Akümülatöre ise sıra numarası yüklenecektir. Sıralı bağlanma gerçekleştiğinde program o noktadan işletilmeye devam edecektir.

CALL komutu altprogram çağırma komutudur. Altprogram sıkça yapılan işlemlerin küçük program haline getirilmiş halidir. CALL komutu mutlak ve uzun adreslemelerde kullanılır. Uzun adreslemede 3 bayt uzunluğundadır ve belleğin her adresinden altprogram çağırır. Mutlak adreslemede ise bulunduğu 2Kbaytlık blok bellek içerisinde yer alan altprogramları çağırabilir. CALL komutu uzunadreslemede

LCALL, mutlak adreslemede ACALL adını alır. Birçok assembler programında CALL yazmanız yeterlidir kendisi uygun olan adresleme modunu derleme sırasında belirler. Ücretsiz dağıtılan assembler programların geçerlilik değeri 2048 bayt olduğu için LCALL komutunu derleyemez

Alt program işletildikten sonra ana programa geri dönüş RET komutu ile sağlanır. CALL komutu işlendikten sonra o andaki PC içeriği saklanarak alt programa gidilir. Alt program işletildikten sonra alt program sonuna yerleştirilen RET komutu işlenir ve önceden saklanan geri dönüş adresi PC içerisine alınır ve kalınan yerden ana program işletilmeye devam eder.

RETI komutu ise RET komutu ile benzer işlem yapar. RETI komutu kesme servis alt programından ana programa dönüşü sağlar.

#### Örnek 35:

Yığın işaretleyici 38h içerdiği durumda 38H ve 37h adresli iç RAM bellek satırları sırası ile 45h, 89h sayılarını içermektedir. Aşağıdaki komut işletildiğinde program yürütülmeye hangi adresten devam edilir?

RET

38h ve 37h satırlarının içerikleri PC'ye alınacak ve program işletilmeye 8945H adresinden devam edecektir.

#### Örnek 36:

Yığın işaretleyici 38h içerdiği durumda 38H ve 37h adresli iç RAM bellek satırları sırası ile 45h, 89h sayılarını içermektedir. Aşağıdaki komut işletildiğinde program yürütülmeye hangi adresten devam edilir?

RETI

38h ve 37h satırlarının içerikleri PC'ye alınacak ve program işletilmeye 8945H adresinden devam edecektir.

#### Koşullu Dallanma Komutları

Çizelge-3.8 de koşullu dallanma komutlarının listesi verilmiştir. Koşullu dallanma komutlarını tümü bağıl adreslemede kullanılır. Bağıl adreslemede kullanılan kayıklık değeri -128 ile +127 arası olabilir JZ ve JNZ komutları PSW içerisinde sıfır bayrağı yeralmadığı için Akümülatörün içeriğini denetleyerek dallanıp dallanmayacağını karar verir. JZ komutu Acc'nin tüm bitleri sıfır ise belirtilen adrese dalanır, değilse bir

sonraki satırdan devam eder. JNZ komutu ise ACC'nin içeriği sıfırdan farklı ise belirtilen adrese dallanın aksi halde bir sonraki kumuttan devam eder.

DJNZ komutu ise önce belirtilen belleği ya da yazacın içeriğini bir azaltır, eğer sıfır değilse belirtilen satıra dallanır, aksi halde bir sonraki kumuttan devam eder. DJNZ komutu, olay saymak için kullanılır. CJNE, karşılaştır eşit değilse dallan, komutu da döngü kontrolünde kullanılan diğer bir koşullu dallanma komutudur. Bu komutun diğer kullanım alanı ise iki sayıdan büyük olan ile küçük olanı ayırma işlemidir. Eğer birinci sayı, ikinciden küçük ise elde bayrağı kurulur, tersi ise elde bayrağı temizlenir. Koşullu dallanma komutları yalnızca koşul gerçekleşmişse yürütülmektedir. Eğer koşul gerçekleşmemişse bir sonraki komutun yürütülmesine geçilmektedir.

CJNE A, #Sayı, ileri

$A \geq \text{Sayı} \rightarrow C=0$

$A < \text{Sayı} \rightarrow C=1$

Kısa ad	Yaptığı işlem	Doğr.	Dolaylı	Yazaç
JZ bağıl adres	$A=0$ ise dallan	A'ya özel		
JNZ bağıl adres	$A \neq 0$ ise dallan	A'ya özel		
DJNZ <bayt>, bağıl adres	Azalt sıfır değilse dallan	X		X
CJNE A,<bayt>, bağıl adres	$A \neq \text{<bayt>}$ ise dallan	X		X
CJNE<bayt>, #veri, bağıl adres	$\text{Veri} \neq \text{<bayt>}$ ise dallan		X	X

Çizelge–3,8 Koşulu dallanma komutları.

#### Örnek 37:

Aşağıdaki programda birinci komut işletildiğinde akümülatöre 01h sayısı yüklenecektir. Sıfırdan farklı olduğu için ikinci komutta dallanma gerçekleşmez. Üçüncü komutta akümülatörün içeriği bir azaltıldığında içerik sıfır olacak ve dördüncü komuttaki dallanma gerçekleşecektir.

MOV A, #01H

JZ İLERİ

DEC A

JZ GERİ

**Örnek 38:**

Aşağıdaki program, P2'den okunan değer sıfır olunca biter.

```
TKR:  MOV A, P2    ;P2'den oku.  
      MOV P1, A    ;P1'e yaz.  
      JNZ TKR      ;Çıkış 0 olunca işlemi bırak.  
      NOP
```

**Örnek 39:**

Akümülatörün içeriği 65h, R7'nin içeriği 89h olduğuna göre aşağıdaki komut işletildiğinde program hangi adresten yürütülmeye devam eder.

```
CJNE R7, #A8H, ESİTD
```

```
.....  
ESİTD: .....
```

R7 ile A8h sayısı karşılaştırılacaktır, R7'nin içeriği 89h olduğuna göre eşit değildir ve dallanma gerçekleşir program yürütülmeye EŞİTD olarak adlandırılmış satırdan devam eder.

## Sorular

- Aşağıdaki komutlardan hangilerinin geçerli olduğunu belirleyin?  
 MOV R3, 30H  
 MOV R3, #30H  
 MOV A, R7  
 MOV A, A  
 MOV 40, 30H  
 MOV 20H, A  
 MOV #20H, A  
 MOV A, #25H  
 MOV A, 25H
- P1'i okuyup bank 3'ün R7'sinde saklayan programı yazın.
- P1'den okunan veriyi P3'e yazan programı yazın.
- A yazacı ile B yazacının içeriklerini toplayıp sonucu bank 2'nin R0 ve R1'inde saklayan programı yazın.
- A yazacının içeriğinin karesini alan ve sonucu bank 2'nin R2 ve R3'ünde saklayan programı yazın.
- Aşağıdaki program adım modunda işletildiğinde belirtilen yazaç ve belleklerin içeriklerini her adım için belirleyin.

	A	B	R0	R7	(20H)	(0FH)
Cpl A						
Div AB						
Mov R7,B						
Mov R0,A						
Setb 07H						

- Reset sonrası aşağıdaki program adım modunda işletildiğinde belirtilen yazaç ve belleklerin içeriklerini her adım için belirleyin. Her komut satırında yapılan işlemi açıklayın

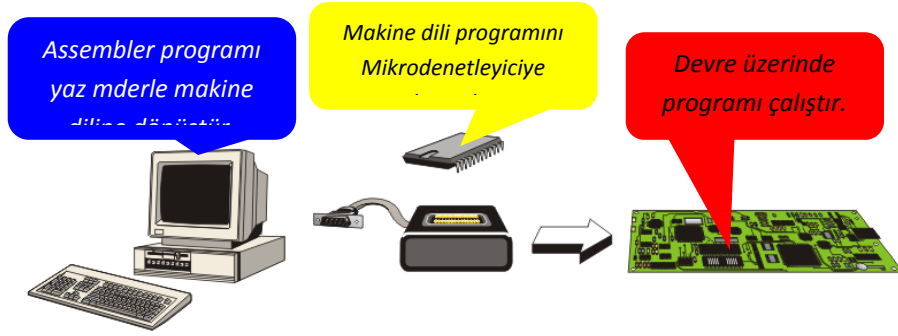
<i>org 0</i>	<i>A</i>	<i>R2</i>	<i>2FH</i>	<i>1AH</i>	<i>Açıklama</i>
<i>Mov A,#255</i>					
<i>Inc A</i>					
<i>ORL A,#0F0H</i>					
<i>CPL A</i>					
<i>Orl PSW,#18H</i>					
<i>RRC A</i>					
<i>MOV 7FH,C</i>					
<i>Mov R2,A</i>					
<i>End</i>					

8. *A ve B yazaçlarının 5 ve 3 nolu bitlerini temizleyen programı yazın.*
9. *A ve B yazaçlarının 7 nolu bitlerini kuran programı yazın.*
10. *A ve B yazaçlarının 5 nolu bitini kuran ve 3 nolu bitini temizleyen programı yazın?*
11. *R0 yazacının 3 ve 4 nolu bitlerini diğer bitleri değiştirmeden 1'e tümleyen programı yazın.*
12. *P0 / R2 işlemini yapan ve bölümü R3 yazacına ve kalanı R4 yazacına yazan programı yazın.*





# Assembly Programlama



## Giriş

Assembly dili makine dili ile yüksek seviyeli dillerin arasında kalan bir programlama dilidir. Mikroişlemci veya mikrodenetleyici programlamak için ilk yıllarda makine dili olarak adlandırdığımız onaltılı sayı sisteminde komutlar oluşturulmuş komutlar kullanılmıştır. Bilgisayarın kullanımının artması ve mikroişlemci programlayıcılarının zorlanması nedeniyle assembly dili ortaya çıkmış ve kullanımı hızla yaygınlaşmıştır. Hatta mikroişlemci kullanımını yagınlaştırmak için bazı üretici firmalar BASIC dilinde programlanabilen mikroişlemciler üretmişlerdir. Günümüzde ise mikroişlemci ve mikrodenetleyici programlamak için yeni öğrenenler assembly dilini profosyoneller ise C dilini kullanmaktadırlar.

Makine dilinde komutlar ikili sayılar ile ifade edilir. Assembly dili ise ikili kodları hatırlaması kolay yapılan işlemi temsil eden iki, üç veya dört harfden oluşan kısaltmalar ile değiştirir. Örneğin toplama komutu makine dilinde "10110011" olarak yazılırken assembly dilinde "ADD" olarak yazılır. Assembly dilinde yazılmış

programlar doğrudan mikroişlemciler tarafından işletilemez. Program yazımı bittikten sonra makine diline dönüştürülmelidir. ADD kısaltması "10110011" haline getirilmelidir.

### Editör

Assembly dilinde program yazmak için kullanılam programa verilen isimdir. Standart ASCII kod üreten her türlü kelime işlem programı editör olarak kullanılabilir.

### Assembler

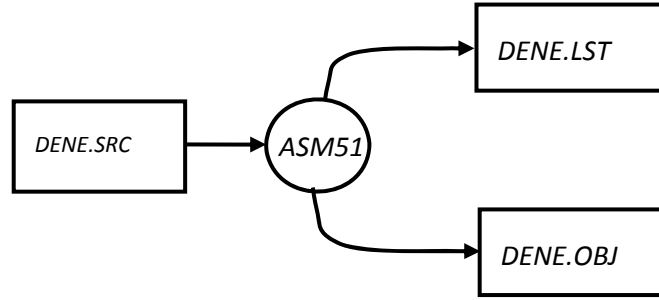
Editörde yazılmış Assembly programını makine diline dönüştüren programdır.

### Linker

Birden fazla dosyada yazılmış ve assembly edilmiş makine kodlarını düzenleyen, birleştiren programdır. Bu işlemi yapan programlara linker/relocator adı verilir. Öğrenme aşamasında genellikle tek bir editör dosyasından oluşan programlar yazıldığı için linker/relocator programı kullanılmaz.

## Assembly İşlemi

8051 mikrodnetleyiciler için program geliştirmeye yarayan birçok assembler programı ve diğer geliştirme programları ticari olarak satılmaktadır. Intel'in önerdiği ASM-51 şu anda satılmamaktadır. Fakat birçok assembler için standart oluşturmuştur.



Şekil-4,1 Kaynak programın assembly edilmesi.

8051 kaynak program herhangi bir bilgisayarda bir text editörde (standart ASCII karakter üreten) yazılır. ASM-51 kullanılarak (bazı kaynaklarda cross assembler olarak adlandırılır) object kod ve list dosyası elde edilir. Assembler programı

## Mikrodnetleyiciler 8051 Uygulamaları

çalıştırıldığında sizden uzantısı ASM olan kaynak dosyanın adını ister. Assembly işlemi tamamlandıktan sonra program tarafından iki adet yeni dosya oluşturulur. Bu iki dosyanın isimleri kaynak dosya ile aynı olup uzantıları ise OBJ ve LST'dir. Örnek olarak dene.asm adlı kaynak dosyayı assembly edelim işlem sonunda dene.obj ve dene.lst adında iki yeni dosya program tarafından kaynak dosyanın bulunduğu dizine kaydedilir. Bazı assembler programları OBJ ve LST dosyaları için isim girme seçeneğini kullanıcıya verir. Bazıları ise bu seçeneği tanımadan kaynak dosyanın adını alır.

Birçok assembler programı dönüşüm işlemini iki aşamada gerçekleştirir, bu tür assembler programlarına iki aşamalı onaylayan (two-pass assembler) assembler adı verilir.

### **Birinci Aşama Onayı**

Birinci aşama onayı süresince sembol tablosu oluşturulur. Yer göstergesi (location counter) O'a veya ORG yönergesi ile belirtilen yere ayarlanır. Komutların uzunluklarına göre her satırda arttırılır. Rastlanılan etiket sembol tablosuna kaydedilerek adresi belirlenir. Komutların bitiminde DB, DW veya DS yönergeleri için gerekli adres atamaları yapılır. Bu değerler kaydedildikten sonra ikinci aşama onayına geçilir.

### **İkinci Aşama Onayı**

İkinci aşamada object ve liste dosyaları elde edilir. Kısa adlar (Mnemonics) ile yazılmış komutların ikilik karşılıkları yerleştirilir. İşlenenler sembol tabloları kullanılarak gerçek adres ve veriye dönüştürülür. İkinci aşamada elde edilen liste dosyaları ileride bu dosyayı başka dosyalar ile birleştirmek istediğimizde referans olarak kullanılacaktır.

LST dosyayı MS-DOS ortamında TYPE komutu ile görüntülenebilir. Birinci sütunda satır numarası, ikincide adres, üçüncüde işlenenler, dördüncüde kaynak programın etiket kısmı, beşincide komutların kısa adları, altıncıda işlenenler, yedincide açıklama kısmı yer alır. OBJ dosya ise adres ve on altılı sayılar içerir.

## **Assembly Dilinde Program Düzeni**

Assembly dilinde yazılan program şu bileşenleri içerir.

- Makine komutları

- Assembler yönergeleri.
- Assembler kontrol eden komutlar.
- Açıklamalar.

Makine komutları kısa adlarla yazılmış komutlardır, örneğin MOV gibi. Assembler yönergeleri makine komutu değildir. Fakat assembler programına program yapısını, sembollerini, verileri, sabitleri oluşturmada yardımcı olan assembler komutları olarak adlandırabileceğimiz komutlardır. Örneğin programın başlangıç adresini belirleyen ORG yönergesi veya bayt veri tanımlayan DB yönergesi gibi. Assemblere kontrol eden komutlar ise assemblerın akışını kontrol eden komutlardır. Açıklamalar assembler programı tarafından değerlendirilmeyen programı yazan veya okuyan kişiye yardımcı olmak için yazılmış yazılardır. Program satırları her biri birbirinden tab veya boşluk ile ayrılan aşağıda belirtilmiş alanlarda yazılmalıdır.

**[etiket]      Kısa ad [işlenen], [işlenen] [.....] [:açıklamalar]**

Etiket ile kısa ad aynı satırda olmayabilir, aynı şekilde kısa ad ile açıklama kısmı da aynı satırda olmayabilir. Fakat kısa ad ile işlenenler aynı satırda olmak zorundadır aralarında sadece bir boşluk veya tab karakterleri olmalıdır.

#### Etiket Alanı

Etiket bulunduğu satırdaki komutun adresini belirtir, özellikle dallanma ve bağlanma komutları için adres yazmak kaynak programı yazarken çok zordur. Etiket verilerek dallanma komutun işlenen kısmına bu etiket yazılır. Assembler programı etiketlerden yola çıkarak doğru adresi hesaplar.

Örneğin SJMP ATLA gibi.

Semboller veya etiketler mutlaka harfler ile veya ?, \_ işaretleri ile başlamalıdır. 31 karaktere kadar uzunlukta olabilir iki kelimeden oluşamaz. Büyük veya küçük harf kullanılabilir, fakat her iki yerde aynı olmalıdır. Komut yönerge ve operatör olarak görevlendirilmiş kelimeler ve Türkçe karakter içeren kelimeler etiket olarak kullanılamazlar.

#### Kısa Ad (Mnemonic)

Etiket alanını izleyen alanda yer alan kısa ad ve assembler yönerge alanı önceden belirlenmiş komutlardır. Komutlara örnek olarak MUL, DIV, MOV, INC gibi. Yönergelere örnek olarak ORG, EQU, veya DB verebiliriz.

## Mikrodenetleyiciler 8051 Uygulamaları

### İşlenen (Operand) alanı

Kısa adı izleyen alanda yer alan işlenen alanı adres veya veri içerir. Bu alanda etikette adresi temsilen kullanılabilir. Sembolde sabit değeri temsilen bu alanda yer alabilir. Bazı komutların işlenen kısmı yoktur, bazılarının bir, iki veya daha işlenen yer alabilir.

### Açıklama (Comment) Alanı

Bu alan noktalı virgül ile başlamak zorundadır. Bu alan assembler tarafından değerlendirilmez, sadece programcı ve programı okuyacak diğer kişilere yardımcı olmak için kullanılır. Birkaç satırdan oluşan açıklamalar genellikle programın genel amacını açıklar.

### Özel Assembler Simgeleri

Özel assembler sembolleri yazaca özel adresleme kipinde kullanılan ve genellikle yazacın adresini temsil eden sembollerdir. Bunlar A, R0, R1, R2, R3, R4, R5, R6, R7, DPTR, PC, C, AB'dir. Bunlara ek olarak \$ işareti adres sayacının mevcut değerini belirtmek için kullanılır.

## Assembler Yönergeleri

Yönergeleri aşağıdaki gibi gruplanabilir.

- Assembler durum kontrolü yapanlar. (ORG, END)
- Sembol belirleyenler. (SEGMENT, EQU, SET, DATA, IDATA, XDATA, BIT, CODE)
- Saklama yeri belirleme veya ayırma yapanlar. (DS, DBIT, DW, DB)
- Program birleştirenler. (PUCLIC, EXTRN, NAME)
- Segment seçenler. (RSEG, CSEG, DSEG, ISEG, BSEG, XSEG)

### Durum Denetimi Yapan Assembler Yönergeleri

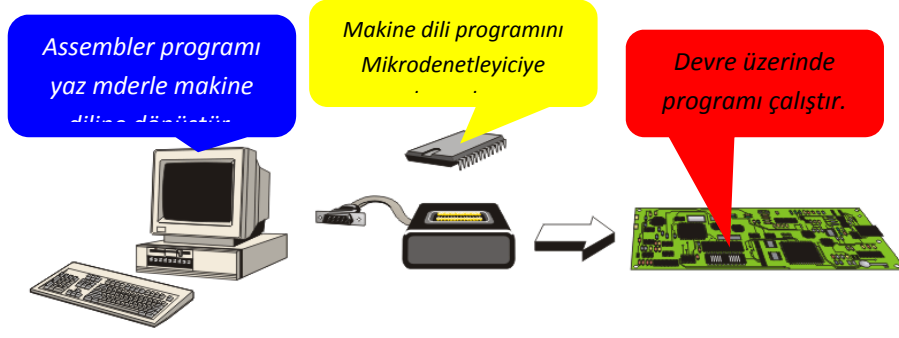
ORG yönergesi yer (adres ) göstericinin içeriğini değiştirir. Bu yönerge öncesi etiket kullanılamaz. Aşağıda kullanılan iki örneği verilmiştir. END yönergesi kaynak dosyanın son komutudur. Etiket kullanımına izin verilmez. Bu komuttan sonra yazılan komutlar assembler tarafından işletilmez. EQU yönergesi, yazaçların

içeriklerini sabit sayıya kurmak ve iç veya program belleğinin, bit adreslenebilir bölgenin her bitine yeni bir isim vermek için kullanılır. DB yönergesi, program belleğinde başvuru tablosu oluşturmak için kullanılır.

## Program Geliştirme

Program geliştirme işlemine öncelikle kullandığımız mikrodenetleyiciye uygun editör, assembler, linker/relocator ve simülâtör programların seçimli yapılır. Son yıllarda bu işlemlerin hepsini ve simülâtör programlarını içeren birleşik program geliştirme ortamları (IDE, Integrated Development Enviroment) yaygın şekilde kullanılmaktadır. 8051 ailesi için Keil, Raisonance, Tasking, IAR, Franklin Software ve Hitex gibi firmalar IDE üretmişlerdir. Bu firmaların değerlendirme sürümlerini internetten ücretsiz olarak edinebilirsiniz. Birçoğunun değerlendirme sürümleri tam sürümünün özelliklerine sahiptir ancak süre veya dosya boyutu kısıtlaması içerirler. Öğrenciler için dosya boyutu kısıtlaması 2048 bayt olan IDE'ler yeterlidir. Profesyoneller için fiyat, performans ve kullanım kolaylığı gibi özellikleri karşılaştırılarak seçim yapılmalıdır.

Seçilen IDE programında yeni proje dosyası açılır. Proje açmak her IDE'de farklılık gösterir, proje dosyası kullanıcıdan işlemcinin adını, programı C dilindemi yoksa assembler dilinde mi olacağını öğrenmek ister. Proje açıldıktan sonra editör dosyası açılır ve kurallara uygun olarak dosyanın isimlendirilip kaydedilmesi gerekir. Kaydetme işleminden sonra editör dosyası proje dosyası ile ilişkilendirilmedir. Kaynak program yazıldıktan sonra assembly edilir. Yazım hatası var ise bu hatalar çeşidi ve satır numaraları ile birlikte kullanıcıya bildirilir. Hatalar düzeltildikten sonra tekrar assembly edilir ve simülâtör çalıştırılır. Simülâtörde program adım adım çalıştırılarak ilgili birim ve yazaçlar belirlenen amaca yönelik olarak gözetlenir. Simülâtörde yazılan programın mantığının doğru olup olmadığı programcı tarafından belirlenir. Yazılan assembly programının mantık hatalarını bulan bir IDE yoktur.



Şekil-4.2 Program geliştirme aşamaları.

Simulator üzerinde doğruluğu belirlenen programın makine dili dosyası hedeflenen mikrodenetleyicinin program belleğine yazılarak gerçek ortamda çalıştırılmalıdır. Aksi halde program geliştirme işlemi tamamlanmamış olur. Şekil-4.2’te program geliştirmenin aşamaları grafiksel olarak gösterilmiştir. Yazma işlemi mikrodenetleyiciye göre bir programlayıcı devre veya dungle kullanarak yapılır. Programlayıcı devre ile bilgisayar arası bağlantı seri port, paralel port veya USB port üzerinden yapılır. Kullanılan devre ve porta göre uygun arayüz programı ilgili firmanın internet sayfasından edinilebilir. Arayüz programları assembly işleminden sonra elde edilen bin, hex veya obj uzantılı dosyalardan bir tanesi kullanarak yazma işlemini yapar. Son yıllarda firmalar mikrodenetleyicilere ISP (In Circuit Programmer) özelliğini (Mikrodenetleyiciyi devreden sökmeden programlama) eklemişlerdir. Hatta bazı üreticiler programı adım adım işletme ve yazaçların içeriklerini okuyabilme olanağı sağlamışlardır. Deneme devresini kendiniz tasarlayabileceğiniz gibi hazır üretici firmalar veya deneme kartı üreten firmalardan satın alabilirsiniz. Geliştirilen programları çalıştırmak için kullanılan tasarlanan sisteme emulator adı verilir. Üretici firmalardan emulator devre şeması ve yazılımı edinebilirsiniz.

### Örnek Kaynak Program Sayfası

Kaynak programın ilk satırında programın adı ve varsa sürüm numarası yazılmalıdır. İkinci satırda yazıldığı tarih ve yer adı yazılmalıdır. Üçüncü satırda programı yazan kişi veya kuruluşun adı yazılmalıdır. Dördüncü satırda birden fazla satır olabilir. Programın amacı, çalışma şekli, kullandığı alt programla, kullandığı bellekler, kullandığı yazaçların isimleri ve kullanım amaçları yazılabilir.

```

;programın adı varsa sürüm numarası
;yazıldığı tarih
;Yazan kişinin adı
;programın işlevini tanımlayan açıklama
        ORG 0H
BASLA:   MOV A, #00H      ;Komutun yaptığı işlemin açıklaması
        .....           ;.....
        .....           ;.....

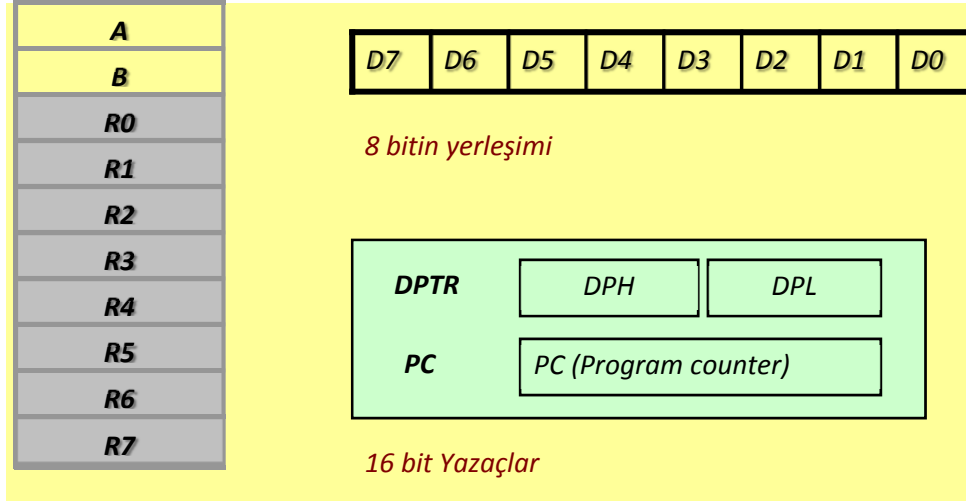
        .....           ;.....
        .....           ;.....
        .....           ;.....
        .....           ;.....
End

```

Şekil-4.2 Örnek kaynak program görüntüsü.

Şekil-4.4 program yazarken ve simulatörde çalıştırırken sıkça kullanılan 8051'in 8 ve 16 bit yazaçları gösterilmiştir.





Şekil-4.4 8051'in sıkça kullanılan 8 ve 16 bit yazaçları.

## Altprogramlar ve Yığının Kullanımı

Altprogramlar sıkça yapılan işlemleri gerçekleyen küçük program parçalarıdır. Genellikle tek amaca yöneliktir, tuş takımının okunması, göstergeye bir karakter yazılması gibi, 16 bitlik iki sayının toplanması gibi. Her altprogramın bir ismi ve en sonunda da altprogramdan ana programa geri dönüşü sağlayan RET komutu vardır. Program akışını ana programdan altprograma geçiren komut CALL komutudur. CALL komutu uzun adreslemede LCALL ve mutlak adreslemede ACALL adını alır. Kullanılmak istenen altprogramın programa belleğinin neresinde olduğuna bağlı olarak uygun adresleme kipi seçilmelidir. Eğer kabul ediyorsa seçim işlemini assembler programına bırakarak sadece CALL olarak yazabilirsiniz.

CALL komutu İngilizcede çağrı anlamına geldiği için programcılar altprograma geçme, bağlanma gibi deyimler yerine "altprogram çağırma" deyimini kullanılır. Altprogram çağrıldığında dönüş adresini bilir ve işlemi tamamladıktan sonra ana programda kaldığı adrese geri döner. Ana programın içerisinde birden fazla yerden çağrılabilir.

Altprogram kullanmak programın bellekte kapladığı alanı azaltırken, programın yazan kişiler dışında başka kişiler tarafından da kolay anlaşılmasını sağlar. Altprogram çağrıldığında geri dönüş adresini yığında saklar. Altprogram içerisinde yığın bu adresi bozmayacak şekilde kullanılmalıdır. Program içerisinde CALL komutu yürütüldüğünde ana programa dönüş adresi, düşük değerli bayt önce olmak üzere,

iki bayt olarak, yığında saklanır. Program sayacına (PC) alt programın başlama adresi yüklenir ve yürütme bu noktadan sürdürülür. Alt programın sonunda RET komutu işlendiğinde, yığının en üstünde yer alan dönüş adresi PC'ye yüklenir ve Call komutundan sonraki satırdan program işletilmeye devam eder.

### Yığının Yapısı Ve Kullanımı

Yığın, tümdevre üzerindeki RAM bellekte verilerin geçici olarak adres kullanılmadan hızlı bir şekilde saklanması için ayrılmış bölgedir. Yığın için ayrılan bölge 8051'de 0–127 8052'de ise 0–255 aralığındadır. Yığın gösterici (SP) yığının o an için en üstünde bulunan değişkenin saklandığı adresi içerir. SP bir özel işlev yazacıdır ve doğrudan adresi 81H'dir. Yığında bir değer saklanmak istendiğinde, bu işleme “yığına veri atma” adı verilir, önce SP'nin değeri donanım tarafından bir arttırılır ve bu adrese saklanacak değer yazılır. Yığından bir baytın okunması, bu işleme “yığından veri çekme” adı verilir, en üstteki bilgi okunur ve sonra yığın göstericinin içeriği donanımca bir azaltılır. Şekil-4.5'te veri atma ve çekme işlemleri grafiksel olarak gösterilmiştir. Yığın alanı SP'ye verilen başlangıç değeri ve yığına depolanan değerlerin miktarı ile belirlenmektedir. SP'nin reset sonrası değeri 07H' dir, yazaç bankaları kullanılacaksa yığın gösterici en azından 31H adresine ayarlanmalıdır. 8052'de ise yığın alanı olarak üst 128 baytlık iç RAM bölgesi kullanılır.

MOV SP,#yığının başlangıç adresi

Yığın işaretleyiciye ivedi adreslemede yığının başlangıç adresi yüklenerek 8051'de yığın iç RAM bölgesinde istenen değere ayarlanabilir.

Yığını programcı “PUSH bayt” ve “POP bayt” komutları ile kullanılırken 8051 donanımı altprogram çağırıldığında veya kesme altprogramına bağlanıldığında geri dönüş adresini saklamak amacıyla kullanır. Yürütülmekte olan ana programın herhangi bir yerinde alt program çağırıldığında ana programın kullandığı veri ve özel işlev yazaçlarının içerikleri altprogram dönüşü ana program tarafından kullanılıyor ise, bu yazaçların içerikleri altprogramın başında yedeklenmeli ve sonunda tekrar yedekten alınmalıdır. Eğer çağırılan alt program ana programın kullandığı bazı yazaçlar ile işlem yapıyorsa, bu yazaçların içerikleri ana veya altprogram tarafından korunmalıdır.

“PUSH doğrudan adres” komutu yürütüldüğünde aşağıda belirtilen işlemler sırasıyla gerçekleştirilir.

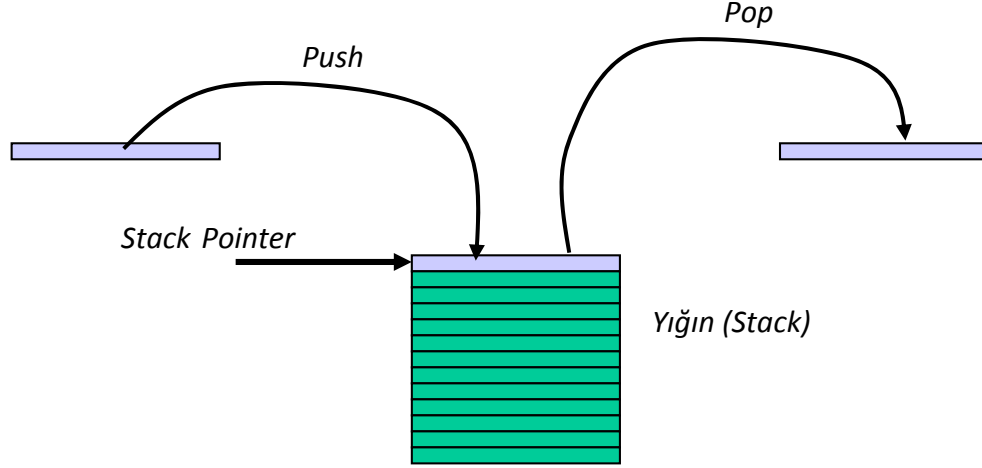
- $SP = SP + 1$

### Mikrodenetleyiciler 8051 Uygulamaları

- $(\text{bayt}) \rightarrow (SP)$  POP

“POP doğrudan adres” komutu yürütüldüğünde aşağıda belirtilen işlemler sırasıyla gerçekleştirilir.

- $(SP) \rightarrow (\text{bayt})$
- $SP = SP - 1$



Şekil-4.5 Yığına veri atma ve yığından veri alma işlemleri.

#### Örnek 4.1

```
Mov SP, #0x40 ; SP'yi ayarla
Push 0x55      ; SP ← SP+1, M[SP] ← M[55], M[41] ← M[55]
Pop B          ; B ← M[55]
```

## Yazılım Tabanlı Zaman Geciktirme Döngüleri

Mikrodenetleyicinin yavaş çalışan çevre birimlerini beklemek veya bir olayın gerçekleşmesini beklemek amacıyla yazılım tabanlı zaman geciktirme döngüleri yazılır. Bu döngülerde bir yazaca veya belleğe istenilen zamana uygun bir değer atanır ve DJNZ komutu ile bu yazacın içeriği 0 olana kadar geriye doğru bir döngü oluşturulur. Elde edilen gecikme süresi döngü içerisinde yer alan komutların makine saykıllarına bağlıdır.

Döngü için eğer uygun ise R0-R7 yazaçları kullanılır. Yazaçlar 8 bitlik olduğu için içerisine yazılabilecek değer 1-255 aralığındadır. Yazaç içerisine 0 yazılmamalıdır. En

basit zaman geciktirme döngüsünü (time delay loop) R7 yazacına 1 sayısını yükleyerek ve DJNZ komutu ile R7 yazcının içeriğini sıfır olana kadar azaltarak yazabiliriz. MOV komutu R7 yazacına 1 sayısını yükler ve 1 makine saykılında işlenir. DJNZ komutu R7 içeriğini bir azaltır, tekrar R7 yazacına yazar ve içeriğinin sıfır olup olmadığını test eder eğer sıfır değilse belirtilen adrese dallanır. Sıfırsa dallanma gerçekleşmez sıradaki komut işlenir. DJNZ komutunun 1 defa işlenmesi 2 saykılında gerçekleşir.

MOV R7,#1 1 makine saykılında işlenir.

DJNZ R7,\$ 2 makine saykılında işlenir.

Döngünün işlenme adedi R7 içerisine yüklenen değer 1 olduğu için toplam makine saykılı 3'tür. Elde edilecek gecikmenin süresini hesaplamak için 1 makine saykılının süresini belirlemek gerekir.  $f_{osc} = 12 \text{ MHz}$  olduğunda;

$$T = 12 / f_{osc} = 12 / 12 \cdot 10^6 = 1 \cdot 10^{-6} \text{ s} = 1 \mu\text{s} \text{ olarak hesaplanır.}$$

$$t = T \cdot \text{Toplam makine saykılı} = 1 \mu\text{s} \cdot 3 = 3 \mu\text{s}$$

Eğer R7 yazacına olası en büyük 8 bit sayı yüklersek elde edilecek süreyi hesaplayalım.

MOV R7,#255 1 makine saykılında işlenir.

DJNZ R7,\$ 2 makine saykılında işlenir.

Döngü 255 defa tekrarlanacağı için DJNZ komutunun işlenmesi için  $2 \times 255$  makine saykılı gereklidir. Mov komutu döngünün dışında kaldığı için yine 1 makine saykılında işletecektir. Elde edilen toplam makine saykılı 511, süre ise  $511 \mu\text{s}$ 'dir.

Zaman geciktirme döngüleri genellikle altprogram halinde program içerisinde kullanılırlar. Döngü sayısı artırılarak zaman geciktirme döngüsünün süresi istenildiği kadar artırılabilir. Kullanım yerine göre altprogram çağırma ve altprogram dönme komutları hatta yedekleme gerekiyorsa yedekleme ve geri alma komutları da hesaplama dâhil edilmelidir.

### **Tek Döngülü Zaman Geciktirme Altprogramları**

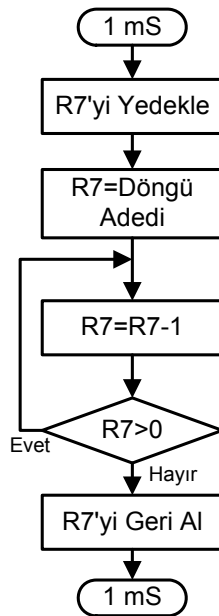
Hedeflenen süreye bağlı olarak zaman geciktirme altprogramlarında kullanılan döngü sayısı belirlenir. Tek döngülü zaman geciktirme altprogramlarında sadece bir adet DJNZ komutu yer alır. Döngünün tekrarlanma sayısı yazaca yüklenen sayı ile belirlenir. Süreyi arttırmak için döngü içerisinde yer alan komut sayısı artırılabilir.

## **Mikrodenetleyiciler 8051 Uygulamaları**

Tek döngülü zaman geciktirme altprogramlarının akış diyagramı şekil-4.6'te gösterilmiştir. Tek döngülü zaman geciktirme altprogramlarında elde edilecek süre örnek altprogramda gösterildiği gibi hesaplanır.

	Komut	Saykıl	Tekrarlanma	Gecikme
Zgd_1:	PUSH 07h	2	1	2 saykıl
	MOV R7,#X	1	1	1 saykıl
	DJNZ R7,\$	2	X	2X saykıl
	POP 07h	2	1	2 saykıl
	RET	2	1	2 saykıl

$$T_{MS} = 2X + 7 \text{ Saykıl}$$



Şekil-4.6 Tek döngülü zaman geciktirme altprogramlarının akış diyagramı.

Elde edilen formül kullanılarak  $X=1$  ile  $X=255$  aralığında istenilen gecikmeler hesaplanır.

En düşük gecikme;

$$T_{MS} = 2.1 + 7 = 9 \text{ makine saykılı.}$$

En yüksek gecikme;

$$T_{MS} = 2.255 + 7 = 517 \text{ makine saykılı.}$$

Bu gecikmeler yeterli olmadığında döngü içerisine DJNZ komutunun yanı sıra işletildiğinde program sayacı dışındaki yazaçların içeriğini değiştirmeyen NOP komutu kullanılarak gecikme artırılabilir. Bu yöntemle 1 mS'lik zaman geciktirme döngüsü yazabiliriz. Döngü içerisine iki adet NOP komutu eklenir ve formül yeniden yazılır.

	Komut	Saykıl	Tekrarlanma	Gecikme
Zgd_1:	PUSH 07h	2	1	2 saykıl
	MOV R7, #X	1	1	1 saykıl
Z1:	NOP	1	X	X
	NOP	1	X	X
	DJNZ R7, Z1	2	X	2X saykıl
	POP 07h	2	1	2 saykıl
	RET	2	1	2 saykıl

$$T_{MS} = 4X + 7 \text{ Saykıl}$$

Elde edilmek istenen süre  $t = 1 \text{ mS}$  olduğuna göre, 1 makine saykılıının süresi ve bu süreyi elde etmek için gerekli makine saykılı hesaplanır.  $F_{OSC} = 12 \text{ MHz}$  kabul edilirse;

$$T = 12 / 12 \cdot 10^6 = 1 \mu\text{S} \text{ olarak hesaplanır.}$$

$$T_{MS} = t / T = 1000 \mu\text{S} / 1 \mu\text{S} = 1000 \text{ makine saykılı olarak hesaplanır.}$$

$$1000 = 4 \cdot X + 7$$

$$X = (1000 - 7) / 4 = 248.25 \text{ olarak hesaplanır.}$$

Tam sayı olmaması nedeniyle alta veya üste tamamlanmalıdır. Alta tamamlandığında  $999 \mu\text{S}$ , üste tamamlandığında  $1003 \mu\text{S}$  gecikme elde edilir. Kullanıldığı uygulama hangisini kabul ediyorsa o değer seçilir. Tam değer elde etmek isteniyorsa alta tamamlanır ve eksik kalan  $1 \mu\text{S}$  için döngü dışına NOP komutu yazılır.

Zgd\_1:

```

PUSH 07h           ;R7'yi yedekle.
MOV R7, #248       ;1 mS için gerekli sayıyı yükle

```

## Mikrodenetleyiciler 8051 Uygulamaları

```

Zgd_11:
    NOP                ;Süreyi ayarla.
    NOP
    DJNZ R7,Zgd_11     ;R7 sıfır olana kadar tekrarla.
    POP 07h            ;R7'yi yedekten al.
    NOP                ;1 µS eksik tamamlanır.
    RET                ;ana programa dön.

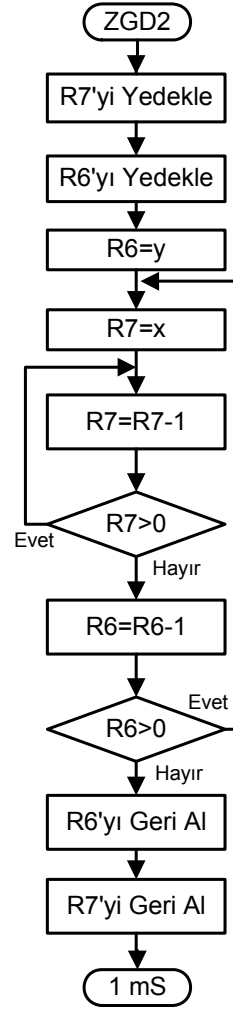
```

### Çift Döngülü Zaman Gecktirme Altprogramları

Tek döngülü zaman gecktirme altprogramları ile elde edilemeyecek süreleri elde etmek ikinci bir döngü yazarak mümkündür. Yazılan ikinci döngünün her tekrarında birinci döngü yeniden başlayacak ve elde edilen süre uzayacaktır. Benzer kurala uyularak üç veya daha fazla döngülü zaman geciktirme altprogramları yazılabilir. Şekil-4.7'de çift döngülü zaman geciktirme altprogramının akış diyagramı gösterilmiştir. İkinci döngü sayacı olarak R6 yazacı kullanılmıştır. İçeride yer alan birinci döngü sayacına 8 bitlik x sayısı, ikinci döngü sayacına 8 bitlik y sayısı yüklenmiştir.

	Komut	Saykıl	Tekrarlanma	Gecikme
Zgd_2:	PUSH 07h	2	1	2 saykıl
	PUSH 06h	2	1	2 saykıl
	MOV R6,#y	1	1	1 saykıl
Z2:	MOV R7,#x	1	y	y saykıl
	DJNZ R7,\$	2	x.y	2xy saykıl
	DJNZ R6,Z1	2	y	2y saykıl
	POP 06h	2	1	2 saykıl
	POP 07h	2	1	2 saykıl
	RET	2	1	2 saykıl

$$T_{MS} = 2xy + 3y + 11$$



Şekil-4.7 Çift döngülü zaman geciktirme altprogramlarının akış diyagramı.

#### Örnek:

$f_{osc}=8$  MHz olan 8051 için 100 mS'li zaman geciktirme altprogramını yazın.

$$T = 12/8.10^6 = 1.5 \mu S$$

$$T_{MS} = t / T = 100000 \mu S / 1.5 \mu S = 66666.66 \text{ saykıl}$$

$T_{MS} = 66667$  saykıl alınabilir.

$$66667 = 2xy + 3y + 11$$



İki bilinmeyenli bir denklemin çözülebilmesi için iki denkleme gereksinim vardır. Fakat eldeki veriler ile ikinci bir denklem yazmak mümkün değildir. Bu denklem iterasyon (sıralı yaklaşım ) yöntemi ile çözülebilir. Bu yöntemde x veya y'ye tahmini değer verilir diğeri hesaplanır eğer elde edilen çözüm geçerli ise yaklaşım doğrudur. Değilse artı veya eksi yönde yaklaşıma çözüme ulaşana kadar devam edilir. Bu denklemde içerdeki döngünün adedini belirleyen x değeri en büyük değer olan 255 yaklaşımı yapılarak çözüme başlanır.

$$66667 = 2.255.y + 3y + 11$$

$$66667-11 = 513y$$

$$y = 129.93$$

Üste tamamlayarak 130 kabul edilebilir. Kabul edilen ve bulunan değerler denklemde yerine konarak tekrar hesaplama yapılır.

$$66667 = 2.255.130 + 3.130 + 11$$

$$66667 = 66701$$

Yapılan hata 34 makine saykılıdır. 1 makine saykılı 1.5  $\mu S$  olduğuna göre 51  $\mu S$ 'lik bir sapma gerçekleşir. Bu değer kullanıldığı uygulamaya bağlı olarak kabul edilebilir. Bu hata yaklaşıma devam edilerek küçültülebilir.

$$x = 254 \text{ edilirse.}$$

$$66667 = 2.254.y + 3.y + 11$$

$y = 130.44$  alta tamamlanır 130 alınır ise sapma -220 üste tamamlanırsa +285 olur. Sapma artmıştır, çözüme devam edilirse hata küçülecektir. Adımları birer birer azaltmak yerine daha yüksek değerle yaklaşım yapılabilir.

## Örnek Altprogramlar

### ÖRNEK 4.2 16 Bit Değişkene 8 Bit Değişken Toplama.

```

/*Bu alt program A'yı DPTR'ye toplar, taşma var ise elde
bayrağı kurulur. İşletilmeden önce DPTR'ye ve A'ya bir
değer girilmelidir.
İşlem sonunda;
    DPTR=DPTR+A
Etkilenen yazaçlar;
    PSW.CY, DPTR */

topla16_8:
    push acc        ; A'yı yedekle.
    add a,dpl       ; A'ya DPL topla.
    mov dpl,a       ; Sonucu DPL'ye yaz.
    mov a,dph       ; DPH'yi al.
    addc a,#00h     ;elde var ise YDB'tı bir artır.
    mov dph,a       ; DPH yaz.
    pop acc         ; A'yı yedekten al.
    ret             ; Ana programa geri dön.

```

### ÖRNEK 4.3: 16 Bit Değişkenden 8 Bit Değişken Çıkarma.

```

/*Bu alt program DPTR yazacından Acc'yi çıkarır. Sonuç
negatif ise elde bayrağını kurar. İşletilmeden önce
DPTR'ye ve Acc'ye bir değer girilmelidir. İşlem sonunda;
    DPTR=DPTR-Acc
Etkilenen yazaçlar;
    PSW.CY, DPTR */

cikar16_8:
    push acc        ; A'yı yedekle.
    clr c           ; Eldeyi temizle
    xch a,dpl       ; A ile DPL'yi yer değiştir.
    subb a,dpl      ; Çıkar.
    mov dpl,a       ; DPL'yi yaz
    mov a,dph       ; DPH'yi al.
    subb a,#00h     ;Borç var ise YD kısmı bir azalt.
    mov dph,a       ; DPH'yi yaz.
    pop acc         ; A'yı yedekten al

```

```
ret          ; Ana programa geri dön
```

#### ÖRNEK 4.4 DPTR Yazacını Bir Bit Sağa Öteleme.

*/\* Bu alt program DPTR yazacının içeriğini aritmetik bir bit sağa öteler. İşletilmeden önce Dptr içerisine bir değer girilmeli. \*/*

*Dptrsag:*

```
    Push acc      ; A'yı yedekle.
    Clr c         ; Eldeyi temizle.
    Mov a,dph     ; DPTR'nin yüksek kısmını al.
    Rrc a         ; Sağa ötele.
    Mov dph,a     ; DPH'ye yaz.
    Mov a,dpl     ; DPTR'in düşük kısmını al.
    Rrc a         ; Sağa ötele.
    Mov dpl,a     ; DPL'yi yaz.
    Pop acc       ; Acc'yi tedekten al.
    Ret          ; Ana programa dön.
```

#### ÖRNEK 4.5 DPTR Yazacını Bir Bit Sola Öteleme.

*/\* Bu alt program DPTR yazacının içeriğini bir bit sola öteler. İşletilmeden önce Dptr içerisine bir değer girilmeli. \*/*

*Dptrsol:*

```
    Push acc      ; A'yı yedekle.
    Clr c         ; Eldeyi temizle.
    Mov a,dpl     ; DPTR'nin DDB'ını al.
    Rlc a         ; Sola ötele.
    Mov dpl,a     ; DPL'ye yaz.
    Mov a,dph     ; DPTR'in YDB'ını al.
    Rlc a         ; Sola ötele.
    Mov dph,a     ; DPH'ye yaz.
    Pop acc       ; Acc'yi tedekten al.
    Ret          ; Ana programa dön.
```

**ÖRNEK 4.6 DPTR'nin 10 Katını Alma.**

```

/* Bu alt program DPTR'ye 10 çarpar. DPTR'ye işlem
öncesi bir değer yazılmalıdır. İşlem sonunda;
    DPTR = DPTR * 10
Etkilenen yazaçlar;
    PSW.CY, DPTR */
DPTRX10:
    pushb        ;B'yi yedekle.
    pushacc       ;A'yı yedekle
    mov a,dpl     ;Düşük kısmı al
    mov b,#10     ;Çarpanı B'ye yaz
    mul ab        ;BA = DPL * 10
    mov dpl,a     ;Sonucun düşük kısmını DPL yaz
    pushb        ;Sonucun yüksek kısmını yedekle
    mov a,dph     ;Yüksek kısmını al
    mov b,#10     ;Çarpanı B'ye yaz
    mul ab        ;BA = DPH * 10
    pop b         ;B yedekten al
    add a,b       ;Düşük kısımdan oluşan ile topla
    mov dph,a     ;Sonucun yüksek kısmını DPH yaz
    pop acc       ;A'yı yedekten al
    pop b         ;B'yi yedekten al
    ret          ;Ana programa dön

```

**ÖRNEK 4.7 DPTR'yi 100 İle Çarpma.**

```

elde bayrağı kurulur. DPTR'ye işlem öncesi bir değer
yazılmalıdır. İşlem sonunda;
    DPTR = DPTR * 100
Etkilenen yazaçlar;
    PSW.CY, DPTR          */

DPTRX100:
    call DPTRX10 ; 10 ile çarp
    call DPTRX10 ; 100 ile çarp
    ret          ; ana programa geri dön.

```

**ÖRNEK 4.8 DPTR'nin 1000 Katını Alama.**

```
/* Bu alt program DPTR'yi 1000 ile çarpar. Taşma olursa
elde bayrağı kurulur. DPTR'ye işlem öncesi bir değer
yazılmalıdır. İşlem sonunda;
```

```
    DPTR = DPTR * 1000
```

```
Etkilenen yazaçlar;
```

```
    PSW.CY, DPTR          */
```

```
DPTRX1000:
```

```
    call DPTRX10 ; 10 ile çarp
```

```
    call DPTRX10 ; 100 ile çarp
```

```
    call DPTRX10 ; 1000 ile çarp
```

```
    ret          ; ana programa geri dön.
```

**Örnek 4.9 DPTR'yi 4 Bit Sağa Ötele**

```
/* Bu Alt Program Dptr'yi 4 Bit Sağa ötele.
```

```
    Etkilenen Yazaçlar: Psw.Cy, Dptr          */
```

```
Dptrrot4:
```

```
    PushAcc      ; A'yı Yedekle
```

```
    Push0        ; R0'ı Yedekle
```

```
    Mov R0,#4    ; Sayacı Ayarla
```

```
L1:
```

```
    Clr c        ;Eldeyi temizle
```

```
    Mov A,Dph    ;YDB'ı Al
```

```
    Rrc A        ;Sağa ötele
```

```
    Mov DPL,A    ;YDB'ı yerine yaz
```

```
    Mov A,Dpl    ;DDB'ı Al
```

```
    Rrc A        ;Sağa Döndür
```

```
    Mov Dpl,A    ;DDB'ı yerine yaz
```

```
    DjnzR0,L1    ;Sayaç Sıfırlana kadar Yap
```

```
    Pop 0        ;Yedekten Al
```

```
    Pop Acc      ;Yedekten Al
```

```
    Ret          ;Ana Programa Geri Dön
```

**Örnek 4.10 Dptr 4 Bit Sola Öteleme**

*/\* Bu Alt Program Dptr Yazacını 4 Bit Sola Öteler, Düşük Değerli Bitlere sıfır Yerleştirir.*

*Etkilenen Yazaçlar;*

*Psw.Cy, Dptr \*/*

*Dptrrotele\_4:*

```

    PushAcc      ; A'yı Yedekle
    Push 0       ; R0'ı Yedekle
    Mov R0,#4     ; Öteleme Sayacını Kur
L1:  Clr C        ; Öteleme İçin Sıfırla
    Mov A,Dpl     ; Dptr'nin Düşük Kısmını Al
    Rlc A        ; Sola Ötele
    Mov Dpl,A     ; Dpl'yi Yaz
    Mov A,Dph     ; Dptr'nin Yüksek Kısmını Al
    Rlc A        ; Sola Ötele
    Mov Dph,A     ; Dph'yi Yaz
    Djnz R0,L1    ; Sayaç Sıfır Olana Kadar Devam Et
    Pop 0        ; Yedekten Al
    Pop Acc      ; Yedekten Al
    Ret          ; Ana Programa Geri Dön

```

**ÖRNEK 4.11 16 Bitlik Toplama**

*/\*Bu program 16 bit iki sayıyı toplar.*

*R7R6+R5R4=R3R2R1 \*/*

*toplal6bit:*

```

    MOV A,R6      ;Birinci değişkenin DDB'ını A'ya al
    ADD A,R4      ;İkinci değişkenin DDB'ı ile topla
    MOV R1,A      ;Sonucun düşük baytını yaz
    MOV A,R7      ;Birinci değişkenin YDB'ını A'ya al
    ADDC A,R5     ;İkinci değişkenin YDB'ını ile topla.
    MOV R2,A      ;Sonucun düşük baytını yaz
    MOV A,#00h    ;16 bitten taşma varsa sonucu 3. baytı
    ADDC A,#00h   ;olarak yaz
    MOV R3,A      ;sonucun 3. baytını yaz
    RET          ;ana programa geri dön

```

**ÖRNEK 13: 16 Bitlik Çıkarma**

```

/* Bu program 16 bit sayıdan 16 bit sayıyı çıkarır.

      R7R6-R5R4=R3R2
*/
Cıkar16bit:
      MOV A,R6      ;1. değişkenin DDB'ını A'ya al
      CLR C          ;Eldeyi temizle
      SUBB A,R4      ;2.değişkenin DDB'ını A'dan çıkar
      MOV R2,A       ;Sonucun düşük kısmını yaz
      MOV A,R7       ;1. değişkenin YDB'ını A'ya al
      SUBB A,R5      ;2. değişkenin YDB'ını A'dan çıkar
      MOV R3,A       ;Sonucun yüksek kısmını yaz
      RET            ;sonucu R2 ve R3'te ana programa götür.

```

**ÖRNEK 14: 16 Bitlik Çarpma**

/\* 16 bit çarpma 8 bit çarpma kullanılarak yapılır. Çarpım sırası ve ara toplamalar dikkatli yapıldığında işlem basittir. Aşağıdaki tabloda örnek çarpma işlemi gösterilmiştir.

	Bayt 4	Bayt 3	Bayt 2	Bayt 1
1. Değişken	.	.	62	30
2. değişken	.	.	43	2E
1. Ara Sonuç	.	.	08	A0
2. Ara Sonuç	.	11	9C	.
3. Ara Sonuç	.	0C	0A	.
4. Ara Sonuç	19	A6	.	.
Sonuç	19	C4	34	A0

Bu örnekte verilerin yazılı olduğu ve sonuçların saklanacağı yazaç isimleri aşağıdaki tabloda verilmiştir.

	Bayt 4	Bayt 3	Bayt 2	Bayt 1
1. Değişken			R6	R7
2. değişken			R4	R5
Sonuç	R0	R1	R2	R3

R5 ile R7'yi çarp, 16 bit sonucu R2 ve R3'e yaz.

R5 ile R6'yı çarp, 16 bit çarpımı R1 ve R2'ye topla.

R4 ve R7'yi çarp, 16 bit çarpımı R1 ve R2'ye topla.

R4 ve R6'yı çarp, 16 bit çarpımı R0 ve R1'e topla. \*/

Carp16bit:

; R5 ile R7'yi çarp, 16 bit sonucu R2 ve R3'e yaz

MOV A,R5 ;R5'i akümülatöre al

MOV B,R7 ;R7'yi B al

MUL AB ;iki değeri çarp

MOV R2,B ;B'de yer alan çarpımın YB R2'ye yaz

MOV R3,A ;A'da yer alan çarpımın DB R1'e yaz

;R5 ile R6'yı çarp, 16 bit çarpımı R1 ve R2'ye topla.

MOV A,R5 ;R5'i akümülatöre tekrar al

MOV B,R6 ;R6'yı B'ye al

MUL AB ;İki değeri çarp

ADD A,R2 ;Çarpımın DB R2'ye ekle

MOV R2,A ;Toplamı R2'ye yaz

MOV A,B ;Çarpımın YB toplama için A'ya aktar

ADDC A,#0h ;DB elde var ise yüksek kısma topla

MOV R1,A ;Sonucu R1'e yaz

MOV A,#0h ;A'yı sıfırla

ADDC A,#0h ;Elde oluştu ise bir üst kısma aktar

MOV R0,A ;Toplamı R0'a yaz.

;R4 ve R7'yi çarp, 16 bit çarpımı R1 ve R2'ye topla

MOV A,R4 ;R4'ü çarpma için A'ya al

MOV B,R7 ;R7'yi B'ye al

MUL AB ;İki değeri çarp

ADD A,R2 ;Çarpımın düşük kısmını R2'ye ekle

MOV R2,A ;Toplamı R2'ye yaz

MOV A,B ;Çarpımın YB toplama için A'ya aktar

ADDC A,R1 ;Elde ile birlikte R1'e ekle

MOV R1,A ;Sonucu R1'e yaz


MOV A,#00h ;A'yı sıfırla

ADDC A,R0 ;Elde ile birlikte R0'a ekle

MOV R0,A ;Sonucu R0'a yaz.



```
;R4 ve R6'yı çarp, 16 bit çarpımı R0 ve R1'e topla.  
MOV A,R4      ;R4'ü tekrar akümülatöre al  
MOV B,R6      ;R6'yı B'ye al  
MUL AB        ;İki değeri çarp  
ADD A,R1      ;Çarpımın düşük kısmını R1'e ekle  
MOV R1,A      ;sonucu R1'e yaz  
MOV A,B       ;Yüksek değerli kısmı A'ya aktar  
ADDC A,R0     ;YDB'a topla  
MOV R0,A      ;Sonucu R0'a yaz  
RET           ;son
```



### Sorular

1. MOV DPTR,#02FF ve INC DPTR komutları sırasıyla işletildikten DPTR'nin içeriği nedir?
2. Akümülatörün yedici bitini diğer bitleri değiştirmeden hangi komut kurar?
3. MOV PSW,#0 ve SETB PSW.5 komutları sırasıyla işletildikten sonra etkin yazaç bankası hangisidir?
4. MOV A,#039H ve INC A komutları sırasıyla işletildikten akümülatörün içeriği nedir?
5. MOV A,#0 ve DEC A komutları sırasıyla işletildikten akümülatörün içeriği ve elde bayrağının durumu nedir?
6. Akümülatörün içeriğın 2 ile çarpan komutlar hangisidir?
7. Reset sonrası aşağıdaki program adım modunda işletildiğinde belirtilen yazaç ve belleklerin içeriklerini her adım için belirleyin.

org 0	A	B	R0	R7	20H	1FH
Setb 02H						
Mov B,#235						
Mov A,B						
Cpl A						
ORL A,#F0H						
Addc A,B						
Div AB						
Mov R0,A						
Mov 20H,B						
Orl PSW,#24						
Mov R0,#40						
Mov R7,#246						
Sjmp \$						

8. İç veri belleğinin 30H ile 5FH aralığında yazılı sayıları iç veri belleğinin üst 128 baytlık kısmının A0H adresinden başlayarak kopyalayan programı yazın.

9. 030h-040h aralığında büyükten küçüğe sıralı sayıları aynı yerde küçükten büyüğe sıralayan programı yazın.
10. Y,S,Z,V,T bir bitlik değişkenler olduğuna göre,  $Y=ZS+VS+ZT+VT$  mantık işlemini yapan programı 8051 mikrodenetleyicisini kullanarak yazın.;
- Bağlantı şemasını çizin.
  - Kaynak programını açıklamaları ile birlikte yazın.
11. Y,S,Z,V,T bir bitlik değişkenler olduğuna göre,  $Y=(Z+V)(S+T)$  mantık işlemini yapan programı 8051 mikrodenetleyicisini kullanarak yazın.
- Bağlantı şemasını çizin.
  - Kaynak programını açıklamaları ile birlikte yazın.
12. P0'ı giriş portu olarak ayarlayarak P0'dan okuduğunuz 50 adet veriyi okuma sırasına uygun olarak iç veri belleğinin 90H adresinden başlatarak saklayan programı yazın
13. Reset sonrası aşağıdaki program adım modunda işletildiğinde belirtilen yazaç ve belleklerin içeriklerini her adım için belirleyin

org 0	A	R4	0CH	Açıklama
MOV R4,#0AAH				
SETB RS0				
ANL A,#0				
ORL A,04H				
XRL A,#0FFH				
SETB PSW.7				
RLC A				
Mov 0CH,A				
End				



# *MCS-51*

## *Sayıcıları*

### *Zamanlayıcıları*

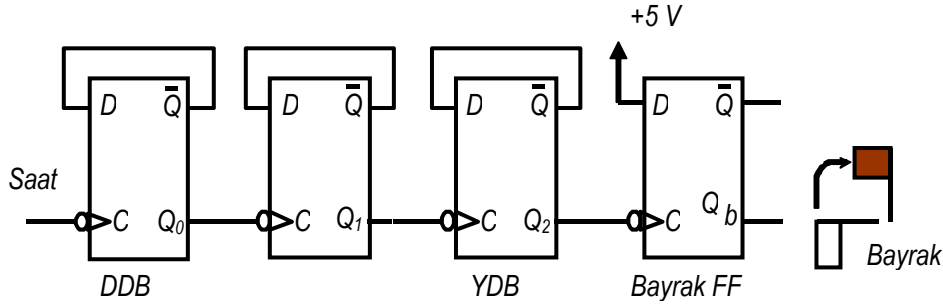
#### *Giriş*

Endüstriyel denetim uygulamalarında sıkça kullanılan sayıcılar mikrodenetleyicilere ilk eklenen birimdir. 8051'in yeni türevlerinde sayıları 8'e kadar çıkmıştır. Zamanlayıcı asenkron sayıcıdan elde edilir. Asenkron sayıcılar bir dizi flip-flopun ardışık olarak bağlanmasıyla oluşur. İlk flip-flop girişine sayıcının hızını belirleyen, saat işareti uygulanır. Her flip-flop çıkışı bir sonrakinin saat girişine bağlanır. Flip-flop sayısı  $n$  ise sayıcının sayma aralığı  $2^n$ 'dir. En sona yerleştirilen flip-flop veya bayrak zamanlayıcının taşma durumunu test eder. Sayılan en büyük değerden sonra zamanlayıcı veya asenkron sayıcı tekrar en düşük değere ulaştığında bu bayrak veya flip-flop kurulur. Bu bayrağın durumu yazılım veya kesmeye izin verildiğinde donanım tarafından denetlenebilir. Örnek olarak 16 bit zamanlayıcıyı ele alacak olursak, zamanlayıcı FFFFH değerinde 0000H değerine geçişte bayrak kurulur.

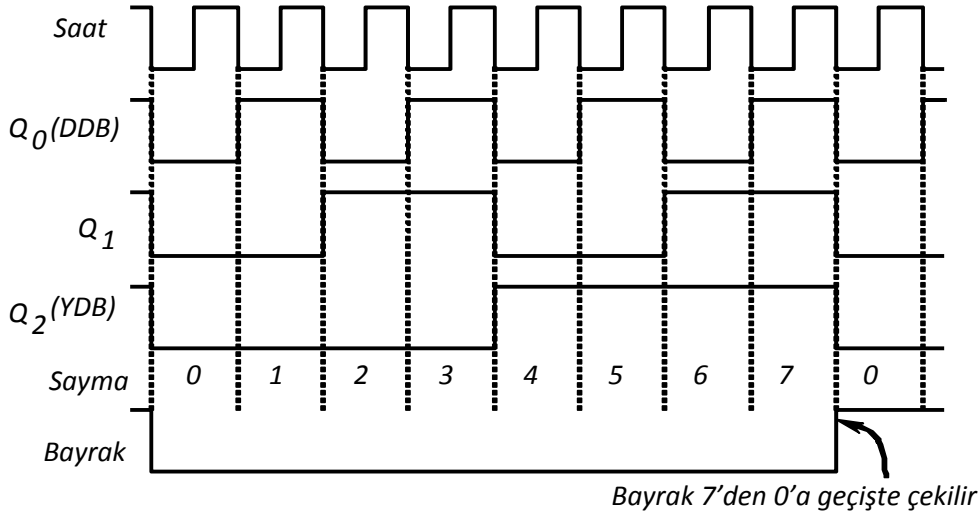
Şekil - 5,1'de 3 bitlik basit bir zamanlayıcının devresi gösterilmiştir. Bayrak olarak D tutucusu kullanılmıştır. Sayıcı flip-flopları düşen kenar tetikli D flip-floplardır. Her flip-flop Q' çıkışları kendi D girişine bağlanmıştır. Q çıkışları ise bir sonrakinin saat

girişlerine bağlanmıştır. Şekil-5.1b’de bu zamanlayıcıya ait zamanlama diyagramı gösterilmiştir. Birinci flip-flop çıkışını her saat vurusunun düşen kenarında tümler böylece girişine uygulanan saat frekansını ikiye böler. İkinci flip-flop ise  $Q_0$ ’ın düşen kenarlarında çıkışını tümler, birinci flip-flopun çıkış frekansını ikiye bölmüş olur. Diğerleri de aynı yöntemle çalışır.

Zamanlayıcılar sabit frekanslı osilatör işareti ile tetiklendiğinde zaman ölçmek amacıyla, dışarıdan herhangi bir sensörden elde edilen vuru ile tetiklenirse olay sayma amacıyla kullanılır. Bu nedenle 8051 üzerindeki zamanlayıcılara zamanlayıcı/sayıcı adı verilir.



(a). 3 bit zamanlayıcı devresi



(b). 3 bit zamanlayıcı zamanlama diyagramı.

Şekil - 5.1 3 bit zamanlayıcı devresi ve zamanlama diyagramı.

## Mikrodenetleyiciler 8051 Uygulamaları

## Zamanlayıcı Yazaçları

8051 zamanlayıcılarını programlamak için 6 adet özel amaçlı yazaç kullanılır. Bu yazaçlardan dört tanesi zamanlayıcının değerinin tutulduğu yazaçlardır, diğer ikisi ise zamanlayıcı çalışma kipini belirleme ve sonuçlarını değerlendirmenin yapıldığı yazaçlardır. 8052’de zamanlayıcı 2 için bunlara ek olarak beş adet yazaç yer almıştır. Bu yazaçların hepsi Çizelge–5.1’de adresleri ile birlikte verilmiştir. Zamanlayıcıları denetleyen yazaçlar bit adreslenebilir yazaçlardır, diğer yazaçlar ise bayt adreslenebilir.

SFR	Kullanım Amacı	Adres	Bit Adresleme
TCON	Denetim	88H	Var
TMOD	Çalışma kipi belirleme	89H	Yok
TL0	Zamanlayıcı 0 DDB	8AH	Yok
TL1	Zamanlayıcı 1 DDB	8BH	Yok
TH0	Zamanlayıcı 0 YDB	8CH	Yok
TH1	Zamanlayıcı 1 YDB	8DH	Yok
T2CON*	Zamanlayıcı 2 denetim	C8H	Var
RCAP2L*	Zamanlayıcı DDB yakalama	CAH	Yok
RCAP2H*	Zamanlayıcı YDB yakalama	CBH	Yok
TL2*	Zamanlayıcı 2 DDB	CCH	Yok
TH2*	Zamanlayıcı 2 YDB	CDH	Yok

\* 8051, 8751, 8031’ de yoktur.

Çizelge - 5.1 Zamanlayıcı Yazaçları

### Zamanlayıcı Mod Seçme Yazacı

TMOD yazacı zamanlayıcı 0 ve zamanlayıcı 1’in çalışma kipini belirleyen iki adet dört bitlik gruptan oluşur. Çizelge–5.2’de bu bitlerin görevleri gösterilmiştir.

Gate biti zamanlayıcının çalışmasını dışarıdan denetlemek istediğimizde kullanılır. Bu bit kurulduğunda Zamanlayıcı 1’de INT1, Zamanlayıcı 0’da INT0, yüksek seviye yapılmadığında zamanlayıcı diğer tüm koşullar yerine gelse bile çalışmaz. Bu biti kısaca dış izin biti olarak adlandırabiliriz. C/T biti zamanlayıcının iç saat

kaynağından mı? Yoksa TX girişine uygulanan dış kaynaktan mı tetikleme işaretini alacağını seçer. İç kaynaktan tetiklendiğinde zaman dilimi ölçtüğü için zamanlayıcı adı verilirken, dış kaynaktan tetiklendiğinde dışarıda gelişen olayı saydığı için sayıcı adı verilir. M1 ve M0 bitleri ise zamanlayıcıların çalışma kiplerini (modlarını) seçer. Çizelge-5.3'te çalışma kiplerinin seçimi gösterilmiştir. Kip 0 8048'den kalan bugün kullanımı fazla olmayan 13 bit zamanlayıcı olarak çalışmasıdır. Kip1 16 bit zamanlayıcı çalışması, kip 2 8 bit yeniden yüklemeli zamanlayıcı çalışması ve kip 3 8 bit ayrık zamanlayıcı çalışması olarak adlandırılmıştır. Çalışma kipleri zamanlayıcı ve sayıcı çalışma bitinden bağımsız çalışır. Bu çalışma kipleri  $C/\bar{T}$  biti 1 olsa da 0 olsa da aynı şekilde seçilir.

BİT	ADI	Z/S	Açıklama
7	GATE	1	Geçit biti kurulduğunda T1 sadece INT1 yüksek seviyede ise çalışır.
6	C/T	1	Sayıcı/zamanlayıcı seçme biti, 1 sayıcı, 0 zamanlayıcı.
5	M1	1	Kip bit 1 (Bakınız Çizelge-5.3)
4	M0	1	Kip bit 0 (Bakınız Çizelge-5.3)
3	GATE	0	T0 geçit biti.
2	C/T	0	Sayıcı zamanlayıcı seçme biti, 1 sayıcı, 0 zamanlayıcı.
1	M1	0	Kip bit 1 (Bakınız Çizelge-5.3)
0	M0	0	Kip bit 0 (Bakınız Çizelge-5.3)

Çizelge - 5.2 TMOD yazacının içeriği ve görevleri.

M1	M0	KİP	AÇIKLAMA
0	0	0	13 bit zamanlayıcı kipi
0	1	1	16 bit zamanlayıcı kipi.
1	0	2	8 bit yeniden yüklemeli zamanlayıcı kipi.
1	1	3	Ayrık zamanlayıcı kipi.

Çizelge-5.3 Zamanlayıcı çalışma kipleri.

## Zamanlayıcı Denetim Yazacı

### Mikrodenetleyiciler 8051 Uygulamaları



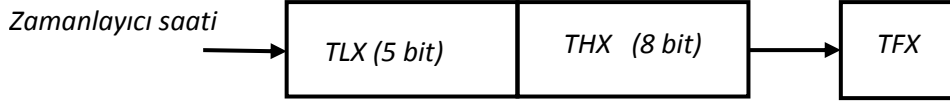
TCON yazacı zamanlayıcı 0 ve zamanlayıcı 1'in durum ve denetim bitlerini içerir. Çizelge-5,4'de içeriği, bitlerin isimleri ve bit adresleri gösterilmiştir. Yüksek değerli dört bit zamanlayıcıları çalıştırma ve kapama işlemlerini yapar TR1, TR0. Diğer iki bit ise zamanlayıcı çalıştırıldı ise taşma oluştuğunda kesme istenip istenmeyeceğini belirler, TF0, TF1. Düşük değerli dört bitin görevi ise zamanlayıcılar ile ilgili değildir. Dış kesme kaynaklarını çalıştırma ve denetleme görevlerini üstlenmişlerdir. Bu bitlerin çalışması kesmeler kısmında açıklanacaktır.

## Zamanlayıcı Çalışma Kipleri

Aşağıda her iki zamanlayıcının çalışması açıklanmıştır. İki zamanlayıcının birçok işlemi benzerdir. Açıklama yapılırken açıklanan işlem her iki zamanlayıcıda da geçerli ise zamanlayıcı numarası yerine "X" işareti yerleştirilmiştir. Şekil-5.2'de her çalışma kipinde TLx, THx ve TFx' aldığı görevler açıklanmıştır.

### 13 Bit Zamanlayıcı Çalışma Kipi

Kip 0 8051'in önceki sürümü olan 8048'e uyumlu olabilmesi için kullanılmıştır. Yeni tasarımlarda bu çalışma kipi kullanılmaz. TLx'in 5 biti ile THx'in 8 biti 13 biti oluşturur. TLx'in düşük değerli 3 biti kullanılmaz. Şekil-5.2'de kip 0'ın çalışması gösterilmiştir. Bu çalışma kipinde zamanlayıcı 0 ile  $2^{13}$  aralığında sayar makine saykılı 1 mikro saniye olduğunda en fazla 8192 mikrosaniyelik bir zaman gecikmesi sağlayabilir.



Şekil-5.2 Kip 0 13 bit zamanlayıcı olarak çalışma.

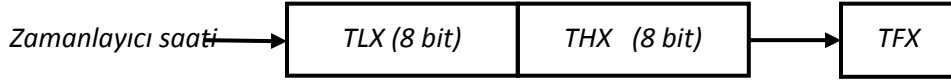
BİT	ADI	ADRES	AÇIKLAMA
TCON.7	TF1	8FH	T1 taşma bayrağı. T1'de taşma olduğunda kurulur ve kesme ister.
TCON.6	TR1	8EH	T1 çalıştırma/durdurma biti.
TCON.5	TF0	8DH	T0 taşma bayrağı. T0'da taşma olduğunda kurulur ve kesme ister.

TCON.4	TR0	8CH	T0 çalıştırma/durdurma biti.
TCON.3	IE1	8BH	IT1 1 ise; INT1 girişinde oluşan düşen kenarda kurulur ve kesme ister.
TCON.2	IT1	8AH	1 düşen kenarda kesme algılanır. 0 düşük seviye de kesme algılanır.
TCON.1	IE0	89H	IT0 1 ise; INT0 girişinde oluşan düşen kenarda kurulur ve kesme ister.
TCON.0	IT0	88H	1 düşen kenarda kesme algılanır. 0 düşük seviye de kesme algılanır.

Çizelge-5.4 TCON yazacının bitleri ve görevleri.

### 16 Bit Zamanlayıcı Kipi

Kip 1 16 bit zamanlayıcı çalışma kipidir. Çalışması kip 0 ile aynıdır fakat bu kipte 16 bitin tamamı kullanılır. Saat vurusu TLx'in DDB'ine uygulanır, TLx'in 7 numaralı çıkışı THx'in DDB'inin tetikleme girişine uygulanır. Sayma 0000H'den başlar 0001H, 0002H gibi devam eder. Taşma FFFFH'den 0000H'ye geçişte gerçekleşir. Bu anda bayrak kurulur sayma devam eder. TCON yazacındaki TFX bayrağı yazılım ile okunup yazılabilir. Şekil-5.3'de bu kipin çalışması gösterilmiştir. Sayıcının en yüksek değerli biti THx yazacının 7 numaralı bitidir, en düşük değerli biti ise TLx'in 0 numaralı bitidir. En düşük değerli bitte giriş saat frekansı ikiye, en yüksek değerli bit çıkışında ise 65 536'ya bölünmüş olarak elde edilir.

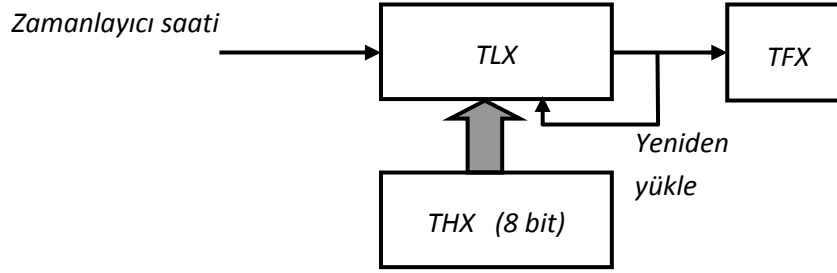


Şekil-5.3 Kip 1 16 bit zamanlayıcı olarak çalışma

### 8 Bit Yeniden Yüklemeli Kip

Bu çalışma kipinde TLx 8 bit sayıcı olarak çalışırken THx yeniden yükleme değerini içerir. Sayma işlemi THx içine yazılan sayıdan başlar FFH'den bir sonraki sayıya geçtiğinde taşma bayrağı kurulur ve THx'de saklanan değer tekrar TLx yüklenir ve saymaya buradan başlanır. Örnek olarak THx'e 5DH yüklenmiş olsun sayıcı 5DH ile FFH arası sayar. Her FFH'den 5DH'ye geçişte taşma bayrağı kurulur. Şekil-5.4'de zamanlayıcının kip 2'de çalışması gösterilmiştir. Bu çalışma kipinde elde edilebilen en yüksek sayma değeri 256'dır. 1 Mhz'lik tetikleme işareti kullanıldığında elde edilebilecek en uzun zaman dilimi 256 makine saykılı olacaktır.

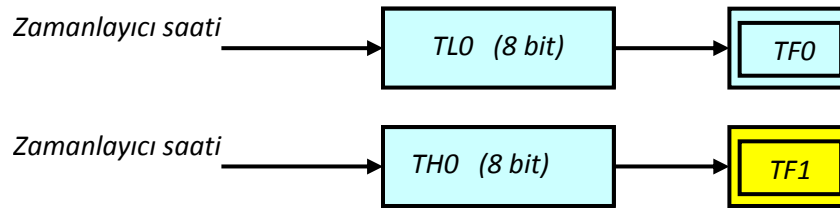
## Mikrodenetleyiciler 8051 Uygulamaları



Şekil-5.4 Kip 2 Yeniden yüklemeli 8 bit zamanlayıcı olarak çalışma

### Ayrık Zamanlayıcı Kipi

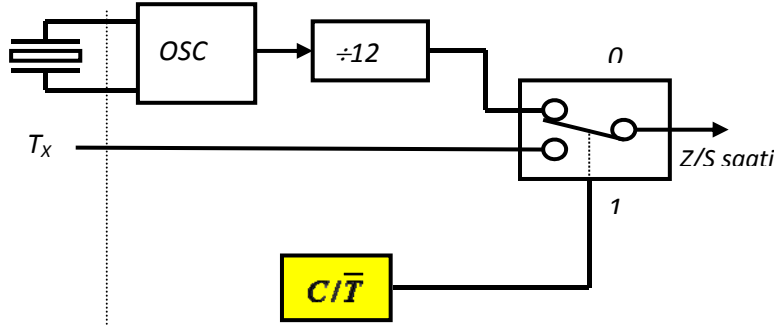
Kip 3 ayrık zamanlayıcı kipidir. Diğer zamanlayıcılardan farklıdır. Zamanlayıcı 1 bu kipte çalışmaz. Zamanlayıcı 0 ise TL0 ve TH0 8'er bitlik iki ayrı zamanlayıcı olarak çalışırlar. TL0 taşma anında TF0'ı kurarken, TH0 taşma anında TF1'i kurar. Bu çalışma kipinde 8051 üçüncü bir zamanlayıcıya sahip olur. İki tanesi 8 bitlik, bir tanesi 16 bitliktir. Şekil-5.3'te zamanlayıcının kip 3'te çalışması gösterilmiştir. Zamanlayıcı 0 kip 3'te çalışırken zamanlayıcı 1 taşma bayrağını ihtiyaç duymayan çalışma kipinde özellikle seri kanalın "baud rate"ini üretmek için kullanılabilir. Hata ile bayrak kullanılacak kip seçilirse TF1 T1'deki taşmadan etkilenmez TH0'ın taşmasını gösterir.



Şekil-5.5 Kip 3 ayrık iki adet 8 bit zamanlayıcı olarak çalışma

### Tetikleme Kaynakları

8051 zamanlayıcıları içeriden makine saykılı hızında dışarıdan ise T0 ve T1 girişlerine uygulanan tetikleme işaretinin frekansında sayar. Eğer uygulanan tetikleme işareti periyodik ise zaman dilimi ölçülebilir değişken ise olay sayması yapılabilir. Zamanlayıcı ayarlanırken TMOD yazacı içerisindeki C/T biti değiştirilerek saat işaretinin kaynağı belirlenebilir. Bu bit 0 olduğunda iç osilatörden 1 olduğunda ise dış tetikleme girişinden tetiklenir.



Şekil-5.6 Zamanlayıcıyı tetikleme kaynakları

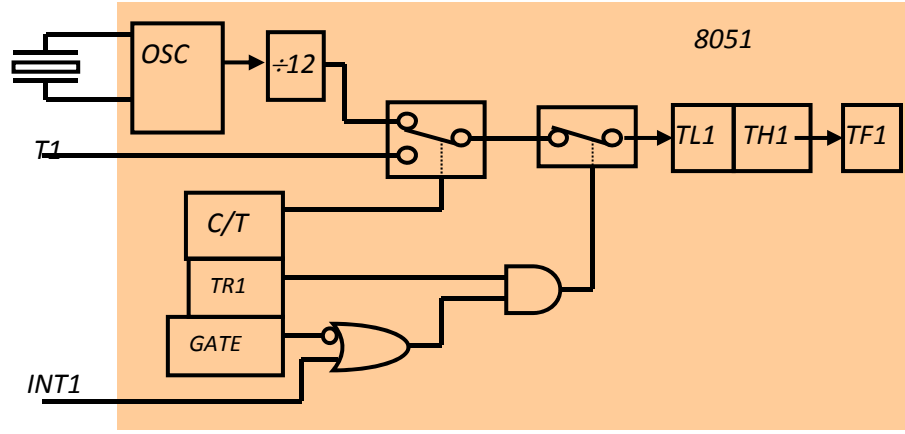
Eğer  $C/\bar{T}$  biti "1" yapılarak dış tetikleme seçildiyse zamanlayıcı sayaçlarını dış tetikleme girişine uygulanan vuruların düşen kenarlarında bir arttırılır. Dışarıda oluşan bir olay saydığı için bu çalışma şekline sayıcı adı verilir. Dış tetikleme işareti zamanlayıcı 0 için P3.4'ten zamanlayıcı 1 için P3.5'ten yapılır. Bu hatların alternatif adları da T0 ve T1 olarak bilinir. Şekil-5.6'da iç ve dış tetikleme kaynaklarının bağlantısı gösterilmiştir. Dış tetikleme girişi her makine saykılının S5 P2 fazı boyunca örneklenir. Vuru bir saykıl boyunca yüksekte kalır bir sonraki saykıl da düşüğe geçerse sayıcı arttırılır. Artırma işlemi takip eden saykılın S3 P1 fazında yapılır. 1'den 0'a düşüş 2 makine saykılında denetlenebildiği için girişe uygulanan vurunun frekansı 12 Mhz kristal kullanıldığında 500 KHz'den fazla olamaz.

## Zamanlayıcıların Kullanımı

Zamanlayıcıları kullanmadan önce çalışma kipini, tetikleme kaynağını ve başlangıç değerini belirlememiz gerekir. Bu ayarlamalar yapıldıktan sonra zamanlayıcıyı başlatma bitini kurarak işlemi başlatabiliriz. Zamanlayıcıların sayma yazaçları çalışırken okunabilir fakat yazılamaz. Yazma işlemi için öncelikle zamanlayıcının durdurulması gerekir.

Zamanlayıcıların ayarlama sırası.

- Tetikleme kaynağı seçilir.
- Çalışma kipi seçilir.
- Başlangıç değeri belirlenir.
- Zamanlayıcı başlatılır.



Şekil-5.7 Zamanlayıcı 1'in kip 1'de çalışması

Zamanlayıcı 1'i çalıştırmak için;

SETB TR1

Komutunu, durdurmak için;

CLR TR1

Komutunu yazma yeterlidir. Zamanlayıcının çalışması için eğer dışarıdan bir donanım onayı gerekiyorsa bu işlem TMOD yazacı içerisindeki GATE kurularak yapılır. Bu bit kurulduğunda dış kesme girişi INTx'i kurulmadıkça diğer koşullar grine gelse dahi zamanlayıcı çalışmayacaktır.

ORL TMOD,#00001000B ;Zamanlayıcı 0'ın GATE bitinin kurulması.

ORL TMOD,#10000000B ;Zamanlayıcı 1'in GATE bitinin kurulması.

ANL TMOD,#11110111B ;Zamanlayıcı 0'ın GATE bitinin temizlenmesi.

ANL TMOD,#01111111B ;Zamanlayıcı 1'in GATE bitinin temizlenmesi.

Diğer bitleri değiştirmeden GATE bitinin kurulması ve temizlenmesi ORL ve ANL komutları ile yapılır.

## Zamanlayıcı 2

8052 ve türevlerinde yer alan zamanlayıcı 2 diğer zamanlayıcılar T0 ve T1'e göre gelişmiş çalışma kiplerine sahiptir. Zamanlayıcı 2'nin kip seçme bitleri, denetim biti ve zaman aşım bayrağı T2CON yazacında yer alır. Zamanlayıcı 2 yazaçları ise TL2 ve

TH2 olarak adlandırılmıştır. 16 bit yeniden yükleme kipinde çalışabildiği için yeniden yükleme değerinin tutulması için iki adet 8 bit yazaç ayrılmıştır. RCAP2L ve RCAP2H olarak adlandırılan bu yazaçlar, yakalama kipinde yakalama yazaçları olarakta kullanılır. Zamanlayıcı 2 denetim yazacı, T2CON bit adreslenebilir, bu yazacın bitlerinin adları ve görevleri çizelge-5.6'de gösterilmiştir. Zamanlayıcı 2 tetiklemesi sistem saati ile veya dış kaynaktan yapılabilir, seçim T2CON yazacında yer alan C/T2 biti ile yapılır. Dış kaynak seçildiğinde T2 girişinden (P1.0) uygulanan vurunun frekansına göre sayma yapılır. Bu tetikleme dışarıda gelişen olayları saymak veya sistem saati dışında bir referansa göre zaman ölçmek için kullanılır.

Zamanlayıcı 1 ve 0'dan farklı olarak Zamanlayıcı2'de dış bağlantıda bazı eklentiler vardır. C/T2 biti ile dış tetikleme kaynağı seçildiğinde işaret diğer zamanlayıcılarda olduğu gibi, T2 girişine (P1.0) uygulanır. Bunun dışında Zamanlayıcı 2'nin T2EX (P1.1) olarak adlandırılan bir dış girişi daha vardır. Bu hattın izinlemesi EXEN2 biti ile yapılır. Bu bit kurulduğunda T2EX girişinde oluşan 1'den 0'a geçişte zamanlayıcının seçilen çalışma kipine göre yakalama veya yeniden yükleme yapılırken, aynı zamanda dış hat gösterge bayrağı olan EXF2 kurulur. EXF2 kesme isteği ile TF2 kesme istekleri VEYA'lanak tek bir kesme isteği olarak MİB'e iletilir.

Zamanlayıcı 2'nin çalışma kipi TCLK, RCLK ve  $CP/\overline{RL2}$  bitleri tarafından belirlenir. Öncelik TCLK ve RCLK bitlerindedir, bu bitlerden herhangi birinin kurulması durumunda diğer bite bakılmaksızın Zamanlayıcı 2 baud rate üretici olarak kullanılır, bu çalışma kipinde TF2 bayrağı taşma anında bile kurulmaz. EXF2 bayrağının herhangi bir değeri yoktur. TCLK ve RCLK bitleri kurulu olmadığında çalışma kipi  $CP/\overline{RL2}$  biti tarafından belirlenir. Bu bit kurulu ise Zamanlayıcı 2 yakalama, temizlendiğinde ise yeniden yükleme kipinde kullanılır. Çizelge-5.7'de çalışma kipleri gösterilmiştir.

Bit	Simge	Adres	Açıklama
T2CON.7	TF2	CFH	Zamanlayıcı 2 taşma bayrağı, zamanlayıcı taşıdığıda kurulur.
T2CON.6	EXF2	CEH	Zamanlayıcı 2 dış gösterge bayrağı. EXEN2 biti kurulduğunda T2EX etkin olursa kurulur ve kesme siteğinde bulunur.
T2CON.5	RCLK	CDH	Bu bit kurulduğunda Zamanlayıcı 2 seri portun alıcı kısmının iletişim hızını belirler. Verici kısmının iletişim

## Mikrodenetleyiciler 8051 Uygulamaları

T2CON.4	TCLK	CCH	<i>hızı zamanlayıcı 1 tarafından belirlenir.</i>
T2CON.3	EXEN2	CBH	<i>Bu bit kurulduğunda Zamanlayıcı 2 seri portun verici kısmının iletişim hızını belirler. Alıcı kısmının iletişim hızı zamanlayıcı 1 tarafından belirlenir.</i>
T2CON.2	TR2	CAH	<i>Zamanlayıcı 2 dış giriş izinleme biti. Bu bit kurulduğunda T2EX girişinde (P1.1) oluşan 1'den 0'a geçişte EXF2 bayrağı kurulur ve yeniden yükleme veya yakalama bu işarete göre yapılır.</i>
T2CON.1	C/T2	C9H	<i>Zamanlayıcı 2 zamanlayıcı/sayıcı seçme biti. Kurulduğunda zamanlayıcı olay sayar. Temizlendiğinde sistem saatini kullanarak zaman aralığı ölçer.</i>
T2CON.0	CP/RL2	C8H	<i>Zamanlayıcı 2 yakalama/yeniden yükleme biti. Kurulduğunda T2EX girişinde oluşan 1'den 0'a geçişte, eğer EXEN2 biti kurulu ise yakalama gerçekleşir. Temizlendiğinde yeniden yükleme kipi seçilmiş olur ve EXEN2 kurulu ise T2EX girişinde oluşan 1'den 0'a geçişte yeniden yükleme yapılır. Aksi durumda TF2 bayrağı kurulduğunda yeniden yükleme yapılır.</i>

Çizelge–5.6 T2CON yazacının bitleri ve görevleri.

TCLK+RCLK	CP/RL2	TR2	KİP
0	0	1	16 Bit yeniden yüklemeli
0	1	1	16 Bit yakalama
1	X	1	Baud rate üretici
X	X	0	Kapalı

Çizelge -5.7 Zamanlayıcı 2 çalışma kipleri.

### Yeniden Yükleme Kipi

T2CON yazacında yer alan CP/RL2 biti temizlenerek bu kip seçilir. TL2 ve TH2 16 bit zamanlayıcı yazaçları olarak kullanılırken, RCAP2L ve RCAP2H yazaçları 16 bit yeniden yüklenecek değeri saklamak için kullanılır. Zamanlayıcı 1 ve 0'da olduğu

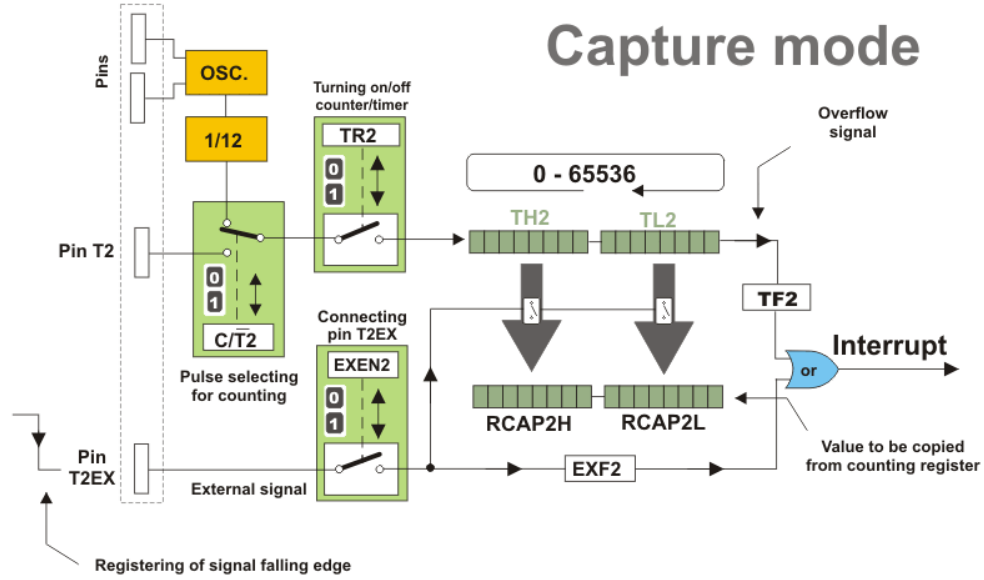




denetlendiğinde ise temizleme işlemi de yazılım ile yapılmalıdır. EXEN2 biti kurulu ise yeniden yükleme işlemi T2EX girişinde oluşan 1'den 0'a geçişte gerçekleşir. Aynı anda EXF2 bayrağı da kurulur, bu bayrak yazılım ile temizlenmelidir. Zamanlayıcı 2'nin 16 bit yeniden yüklemeli kipte çalışması şekil-5.8'te gösterilmiştir.

### Yakalama Kipi

CP/RL2 biti kurularak yakalama kipi seçilir. Zamanlayıcı 2 16 bit zamanlayıcı olarak çalışır ve TH2 ve TL2'in içeriği FFFFH değerinden 0000h değerine artırılması sırasında TF2 bayrağı kurulur. Yakalama kipinin izinlenmesi için T2CON'daki EXEN2 bitinin kurulması gerekir. Bu bit kurulu iken T2EX hattında 1'den 0'a geçiş gerçekleştiğinde TH2 ve TL2'in o andaki değerleri RCAP2L ve RCAP2H yazaçlarına kopyalanır. Aynı anda EXF2 bayrağı kurulur ve kesme isteği olarak MİB'e iletilir. Zamanlayıcı 2'nin yakalama kipinde çalışması şekil-5.9'de gösterilmiştir.



Şekil-5.9 Zamanlayıcı 2'nin yakalama kipinde çalışması<sup>7</sup>.

### Baud Rate Üretici Olarak Kullanılması

8051'de zamanlayıcı 1'in gerçekleştirdiği iletişim hızını belirleme görevi 8052'de kısmi olarak veya tamamen zamanlayıcı 2'ye verilebilir. Özellikle alıcı ile vericinin

<sup>7</sup> Kaynak: Atmel 8052 datasheets.

iletişim hızlarının farklı olması gereken uygulamalarda Zamanlayıcı 2 ve zamanlayıcı 1 birlikte kullanılır.

Zamanlayıcı 2 RCLK ve/veya TCLK bitleri kurularak baud rate üretici olarak kullanılabilir. Şekil-5.10'da Zamanlayıcı 2'nin bu çalışma kipinde kullanılması gösterilmiştir. Bu kipte Zamanlayıcı 2'nin kullanımı yeniden yüklemeli kipteki ile aynıdır. T2H yazacının en büyük değerine ulaşıldıktan sonraki adımda Zamanlayıcı 2 yazaçları yeniden yüklemeli yazaçlarda önceden yazılımla ayarlanmış değerlerle yeniden yüklenecektir. Yeniden yükleme sırasında taşma bayrağı kurulmaz ve kesme isteme olasılığı yoktur.

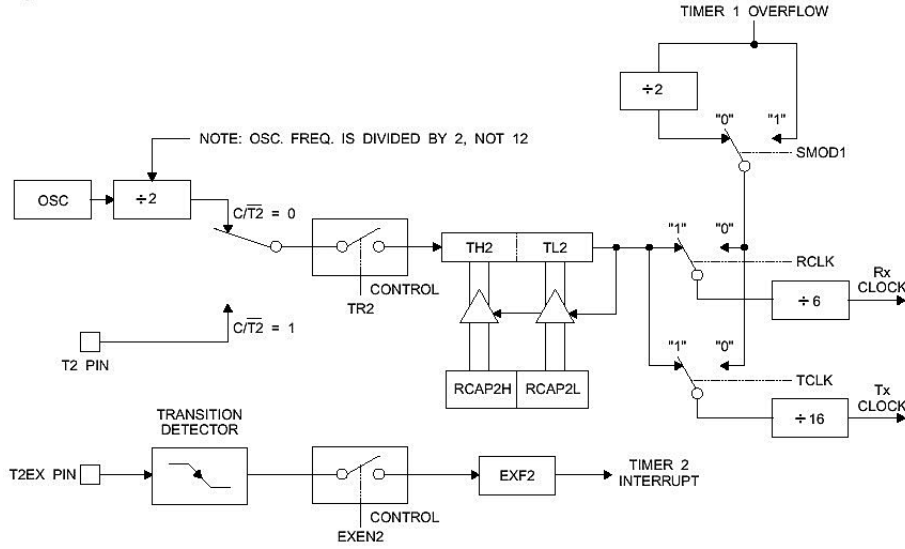
Seri portun kip 1 ve 3'te çalışması durumunda baud rate üretici olarak Zamanlayıcı 2 kullanıldığında taşma değeri aşağıdaki formülden elde edilir.

$$\text{Kip 1 ve 3'te baud rate} = \frac{\text{Zamanlayıcı 2 taşma değeri}}{16}$$

Eğer  $C/\overline{T2}$  biti kurulu ise sistem saati yerine T2 girişinden uygulan işaretin düşen kenarında sayıcını içeriği bir artacaktır. Böylece T2 girişine gelen işaret 2'ye bölünmüş olur. Bu durumda baud rate aşağıda verilen formülle bulmamız gerekir.

$$\text{Kip 1 ve 3'te baud rate} = \frac{\text{Dış osilatör frekansı}}{32 \times (65536 - (RCAP2H, RCAP2H))}$$

RCAP2H, RCAP2H yazaçlarının içeikleri 16 bit olarak onlu sayıya dönüştürülmelidir. Baud rate üretici olarak kullanıldığında Zamanlayıcı 2 yazaçlarının en büyük değerden en küçük değere geçerken TF2 bayrağını kurmadığını belirtmiştik. Eğer EXEN2 kurulursa bu çalışma kipinde T2EX girişinde oluşacak düşen kenar EXF2 bayrağını kuracaktır, eğer kesme izinlendi ise MİB'den kesme isteğinde bulunur. Zamanlayıcı 2 bu kipte çalışırken T2EX girişi üçüncü dış kesme girişi gibi kullanılabilir.



Şekil-5.10 Zamanlayıcı 2'nin baud rate üretici olarak kullanılması<sup>8</sup>.

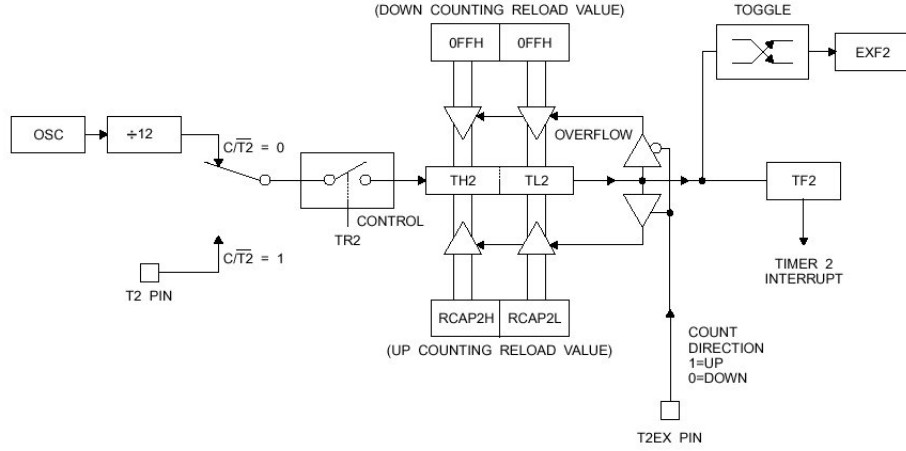
## AT89S52'nin Artıları

AT89C52'nin Zamanlayıcı 2'sinde T2MOD yazacı yer alır, bu yazaç SFR bölgesinde C9H adresinde yer alır sadece iki biti kullanılmıştır. Bu iki bit sayesinde yukarı aşağı sayıcı elde edebilir veya dışarıya işaret gönderebiliriz. DCEN biti kurulduğunda zamanlayıcı 2 yeniden yüklemeli kipte yukarı/aşağı sayıcı gibi çalışır. Sayma yönü T2EX girişinin seviyesine bağlıdır eğer T2EX girişi yüksek seviyede ise Zamanlayıcı 2 yukarı doğru sayma yapar ve taşma en büyük değerden sonra gerçekleşir. Bu hattın seviyesi düşük ise sayma aşağı doğru olur ve alt taşma T2H ve T2L yazaçlarının içerikleri RCAP2H ve RCAP2L yazaçlarına eşit olduklarında gerçekleşir. Zamanlayıcı 2 yazaçlarına taşma sonrası FFFFH sayısı yüklenir ve azaltmaya bu sayıdan devam edilir. Şekil-5.8'de Zamanlayıcı 2'nin aşağı/yukarı sayıcı olarak çalışması gösterilmiştir. Diğer çalışma kiplerinde bu bitin yaptırımı yoktur.

T2OE biti kurulduğunda Zamanlayıcı 2 programlanabilir saat üretici olarak kullanılır. %50 çalışma oranlı saat işareti P1.0'dan dışarıya verilir. Bu çalışma şekli yeniden yüklemeli zamanlayıcı ve baud rate üretici kiplerinde geçerlidir. Bu çalışma şeklinde Zamanlayıcı 2 kesme üretmez. Aynı anda baud rate üretici ve programlanabilir saat üretici olarak kullanmak mümkündür. Fakat frekansları aynı

<sup>8</sup> Kaynak: Atmel 8052 datasheets.

olacaktır. Elde edilen saat işaretinin frekansı osilatör frekansına ve yeniden yükleme yazaçlarının içeriklerine bağlıdır. 16 Mhz'lik kristal kullanıldığında 61 Hz – 4 Mhz arası saat işareti üretilebilir. Elde edilecek işaretin frekansı aşağıdaki formülden elde edilebilir.

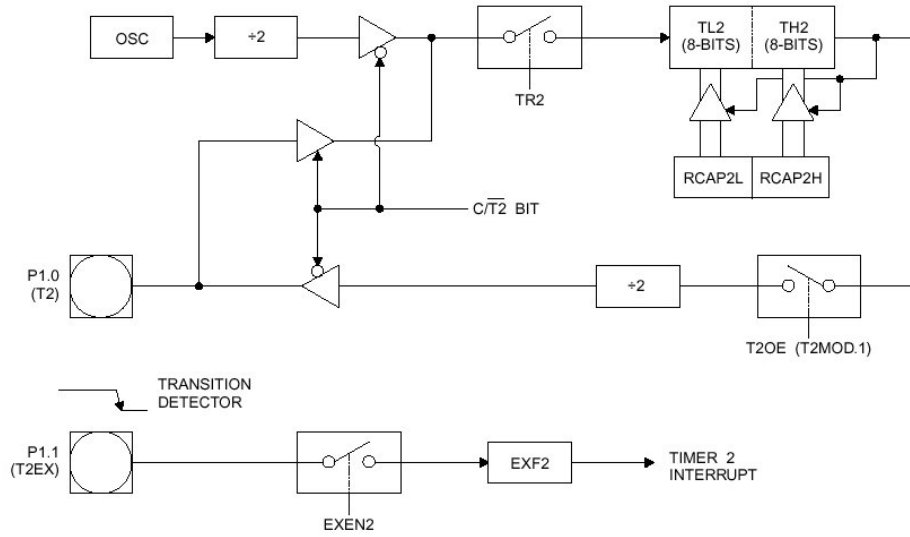


Şekil–5.11 Zamanlayıcı 2'nin aşağı/yukarı sayıcı olarak çalışması<sup>9</sup>.

$$\text{Saat çıkışı frekansı} = \frac{\text{Osilatör frekansı}}{4 \times (65536 - (\text{RCAP2H}, \text{RCAP2H}))}$$

Şekil–5,9'da Zamanlayıcı 2'nin programlanabilir saat üretici olarak kullanılması gösterilmiştir. EXEN2 biti kurulu ise EXF2 bayrağı T2EX girişinde oluşan negatif geçişte kurulur ve kesme izinlendi ise MİB'den kesme isteğinde bulunur. Bu çalışma şeklinde dış keme gibi kullanılabilir.

<sup>9</sup> Kaynak: Atmel 8052 datasheets.



Şekil-5.12 Zamanlayıcı 2'nin programlanabilir saat üretici olarak kullanılması<sup>10</sup>.

<sup>10</sup> Kaynak: Atmel 8052 datasheets.

**ÖRNEK 5.1**

8051'in üretebileceği en yüksek frekanslı işareti üretme

P1.0'dan mümkün olan en yüksek frekanslı vuruyu üreten programı yazın. Frekansını ve duty saykılını 12 Mhz kristalin kullanıldığını varsayarak hesaplayın.

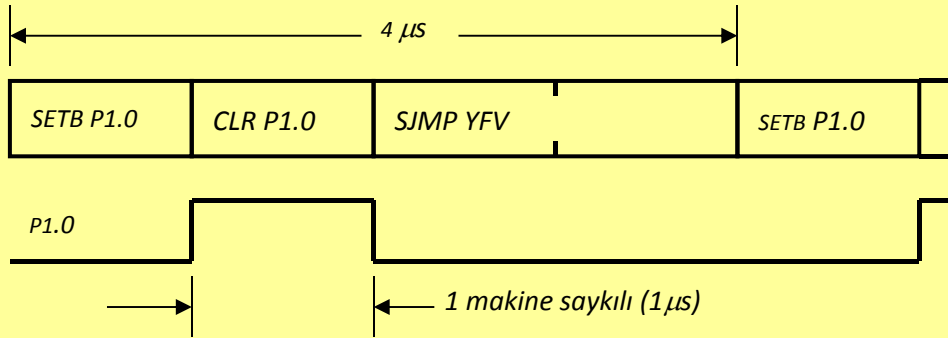
**Çözüm:**

YFV:

```
Setb P1.0    ;1 makine saykılı
Clr P1.0     ;1 makine saykılı
Sjmp YFV     ;2 makine saykılı
```

12 Mhz'lik kristal kullanıldığında 1 makine saykılı 1  $\mu$ saniyedir. Komutları dalga şekli ile beraber zaman ekseninde çizersek aşağıdaki şekil elde edilir. Bir periyot 4  $\mu$ s olduğuna göre frekans 250 Khz olur. Bu elde edilebilecek en yüksek vuru frekansıdır. Duty saykıl ise;

$\%D=1/4=\%25$  olarak elde edilir.



Şekil - 5.13 örnek 1'in dalga şekli.

Bu dalganın periyodu NOP komutu kullanılarak genişletilebilir her NOP komutu periyodu 1  $\mu$ s artırır. Arttırmalar yapılarak  $D=\%50$  yapılabilir bu çözüm 10  $\mu$ s kadar pratiktir, daha uzun süreler için zamanlayıcıları kullanılmalıdır. Periyodu 256  $\mu$ saniyeye kadar olan vurular için 8 bit yeniden yüklemeli kipte çalışan zamanlayıcı kullanılmalıdır. 256 ile 65536  $\mu$ saniyeye arası periyot süreleri için 16 bit zamanlayıcı kullanılmalıdır.

**ÖRNEK 5.2**

*T0'ı kullanarak P1.5'den sürekli D=%50 olan kare dalga üreten programı (Frekans yüklenen sayı ile değişir.) yazın.*

**Çözüm:**

```
TOD50:    mov TMOD,#01      ;T0, kip 1
tekrar:   mov TL0, #XXH
          mov TH0, #XXH      ;Başlangıç = XXXXH
          cpl P1.5
          acall ZGD
          sjmp tekrar
ZGD:      setb TR0            ;T0'ı çalıştır.
          jnb TF0, $          ;TF0 zaman aşımı için tara.
          clr TR0             ;T0'ı durdur.
          clr TF0             ;TF0 bayrağını temizle.
          RET
```

**ÖRNEK 5.3**

*P3.5'teki vuruların sayısını P2'de görüntüle.*

**Çözüm:**

**Basla:**

```
mov TMOD, #01100000B ;kip 2, C/T=1
mov TH1, #0           ;0x00 - 0xFF say.
setb P3.5             ;P3.5 giriř
```

**tekrar:**

```
setb TR1              ;Çalıştır
```

**geri:**

```
mov A, TL1            ;TL1'i oku.
mov P2, A             ;P2'de göster
jnb TF1, geri         ;TF1 kuruldu mu?
clr TR1               ;Durdur.
clr TF1               ;TF1 bayrağını temizle.
sjmp tekrar           ;sürekli yap.
```

**ÖRNEK 5.4**

*P1. 6'ya bir buton, P1.7'ye bir buzzer bağlanmıştır. Butona her basıldığında buzzeri 1 saniye öttüren programı yazın.*

**Çözüm:**

```

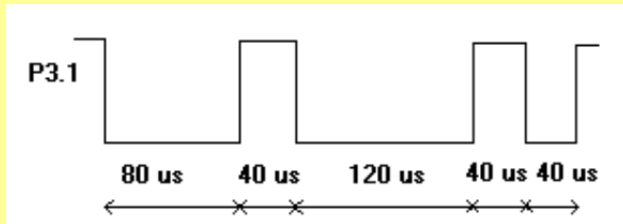
yüzler EQU 100      ;yüzler çarpanı
sayma EQU -10000    ,Zamanlayıcı değeri
ORG oH

        MOV TMOD, #01H      ;T0 16 bit zamanlayıcı
dolan:  JNB P1.6, dolan      ;tuşa basıldı mı?
bekle:  JB P1.6, bekle       ;bırakıldı mı?
        SETB P1.7           ;buzzerı aç.
        CALL tdl            ;bekle.
        CLR P1.7           ;buzzerı kapat.
        SJMP dolan          ;sürekli yap
tdl:     MOV R7,#yüzler
tekrar: MOV TH0,#ydb_sayma   ;YDB'ı yükle.
        MOV TL0,#ddb_sayma   ;DDB'ı yükle.
        SETB TR0            ;zamanlayıcıyı çalıştı
bekle2: JNB TF0,bekle2       ;zaman aşıldı mı?
        CLR TF0             ;evet
        CLR TR0            ;T0'ı durdur.
        DJNZ R7,tekrar
        RET
        END

```

### ÖRNEK 5.5

P3. 1'den aşağıdaki dalga şeklini üreten programı yazın.



P3.1 hattından verilen özelliklerde kare dalga elde edebilmek için öncelikle istenilen sürelerde zaman geciktirme döngüleri yazmak gerekir. Burada en düşük süre 40 mikro saniyedir. İlk olarak 40 $\mu$ s'lik gecikme sağlayan bir alt program yazılır. Bu alt program, ana program içinde üst üste 2 kez çağırılırsa 80  $\mu$ s, üst üste 3 kez çağırılırsa da 120  $\mu$ s elde edilebilir. Zaman geciktirme döngüsü (ZGD) zamanlayıcılar veya yazılım döngüleri ile yapılabilir.

Yazılım döngüsü ile temel olarak Djnz komutu kullanılır, yazaç veya belleğin içerisine yüklenen sayı ile dönme adedi belirlenir ve istenilen gecikme elde edilir. Tek Djnz

## Mikrodenetleyiciler 8051 Uygulamaları



komuttu içeren ZGD'leri ile 0–512 makine saykılı arasında gecikme elde etmek mümkündür. Djnz komutunun işletilmesi 2 makine saykılı sürer. (Komutların işletilme sürelerini Ek-A'da bulabilirsiniz.) Döngüde yer alan diğer komutların işletilme süreleri 1 makine saykılıdır.

gecikme40us:

Mov R0,#X	; 1 makine saykılı, 1 defa işletilir.
Djnz R0,\$	; 2 makine saykılı, X defa işletilir.
Ret	; 1 makine saykılı, 1 defa işletilir.

Toplam Makine Saykılı =  $1 + 2X + 2 = 2X + 3$

X'in değeri elde edilecek süreye bağlı olarak hesaplanan toplam makine saykılı değerinden belirlenecektir.

$F_{OSC} = 12 \text{ Mhz}$  olduğuna göre

$T_{mak. Saykılı} = 12 / f_{OSC}$  formülünden hesaplanabilir.

$T_{mak. Saykılı} = 12 / (12 \times 10^6) = 1 \mu\text{Saniyedir.}$

$t = (\text{Toplam makine saykılı}) \times (T_{mak. Saykılı})$

$40 \mu\text{S} = (\text{Toplam makine saykılı}) \times (1 \mu\text{S})$

Toplam makine saykılı = 40 adettir.

$40 = 2X + 3,$

$X = (40 - 3) / 2$

$X = 18.5$  tam sayı olmadığı için R0 yazacına yazılamaz. Kullanılan uygulamaya bağlı olarak alta yada üste tam sayıya tamamlanmalıdır. 18 olarak alta tamamlayabiliriz.

$X = 18$  aldığımızda elde edeceğimiz gecikme,

$t = (2X + 3) \times (T_{mak. Saykılı}) = (2 \times 18 + 3) \times (1 \mu\text{S}) = 39 \mu\text{S olur.}$

Yapılan hata  $1 \mu\text{S}$ 'dir. Tam değeri elde etmek istiyorsak programda Ret komutu ile Djnz komutu arasına 1 makine saykılı süren ve hiçbir yazacın içeriğini değiştirmeyen NOP komutunun eklemesi yeterlidir.

gecikme40us:

Mov R0,#X	; 1 makine saykılı, 1 defa işletilir.
Djnz R0,\$	; 2 makine saykılı, X defa işletilir.
Nop	; 1 makine saykılı, 1 defa işletilir.

```

Ret                                ; 1 makine saykılı, 1 defa işletilir.
;*****
;Bu program P3.1 ucunda istenilen özelliklere sahip kare
dalga oluşturur.
;*****
ORG 00H                            ;programın başlangıç adresi
karedalga:

    Clr P3.1                        ;p3.1 temizle
    Acall gecikme40us               ;40 µs bekle.
    Acall gecikme40us               ;40 µs bekle.
    Setb p3.1                       ;p3.1 set et.(1 yap)
    Acall gecikme40us               ;40 µs bekle.
    Clr p3.1                        ;p3.1 temizle(0 yap)
    Acall gecikme40us               ;40 µs bekle.
    Acall gecikme40us               ;40 µs bekle.
    Acall gecikme40us               ;40 µs bekle.
    Setb p3.1                       ;p3.1 set et.(1 yap)
    Acall gecikme40us               ;40 µs bekle.
    Clr p3.1                        ;p3.1 temizle(0 yap)
    Acall gecikme40us               ;40 µs bekle.
    Setb p3.1                       ;p3.1 kur.(1 yap)
    Sjmp karedalga                  ;sürekli yap

;*****
;40µs gecikme sağlayan alt program
;*****
gecikme40us:
    Mov R0,#18                      ;R0= 18
    Djnz R0,$                       ;R0=0 olana kadar devam et.
    Nop                             ;işlem yapmadan 1 makine saykılı
    Ret
    end

Gecikme altprogramı zamanlayıcı ile de yapılabilir. 40
µs lik gecikme için zamanlayıcının 40 maine saykılı
sayma yapması gerekir.

Kip 1 kullanılırsa

65536- 40= 65496= FFD8 bulunan bu değer zamanlayıcıya
yüklenecek olan başlangıç değeridir.

```

```

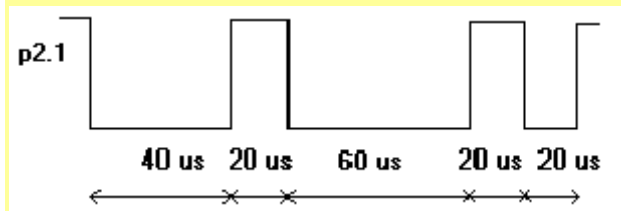
;*****
;40µs gecikme sağlayan alt program (zamanlayıcı 1, kip 1
kullanıldı)
;*****
gecikme40us:
    MOV TMOD, #10H    ;Zamanlayıcı kip 1
    MOV TL1, #0D8H    ;TL1=D8H
    MOV TH1, #0FFH    ;TH1=FFH
    SETB TR1          ;T1'i başlat.
bekle:
    JNB TF1, bekle     ;hala süre dolmadıysa (40 µs) bekle.
    CLR TR1            ;süre doldu, zamanlayıcı durdur.
    CLR TF1            ; taşma bayrağını temizle.
    RET                ;ana programa geri dön.

```

Ana programda herhangi bir değişiklik yok sadece alt program değişecektir. Bu alt programda gecikme altprogramı tam olarak 40 µs sağlamayabilir çünkü arada süreyi uzatacak başka komutlarda vardır. Mov, clr, setb vb. Gerçekte uygulama yapılmak istenirse bu komutlar da göz önünde bulundurulmalıdır.

### ÖRNEK 5.6

P2.1'den aşağıdaki dalga şeklini üreten programı yazın.  $f_{osc} = 16 \text{ Mhz}$ 'dir



P2.1 hattından verilen özelliklerde kare dalga elde edebilmek için öncelikle istenilen sürelerde zaman geciktirme döngüleri yazmak gerekir. Burada en düşük süre 20 µs'dir. İlk olarak 20 µs'lik gecikme sağlayan bir alt program yazılır. Bu alt program, ana program içinde üst üste 2 kez çağırılırsa 40 µs, üst üste 3 kez çağırılırsa da 60 µs elde edilebilir.

Zaman geciktirme döngüsü ile 20 µs'lik zaman geciktirme altprogramını yazalım.

Gecikme20us:

```

    Mov R0, #X          ; 1 makine saykılı, 1 defa işletilir.
    Djnz R0, $           ; 2 makine saykılı, X defa işletilir.

```

*Ret ; 1 makine saykılı, 1 defa işletilir.*

*Toplam Makine Saykılı = 1 + 2 X + 2 = 2X + 3*

*X'in değeri elde edilecek süreye bağlı olarak hesaplanan toplam makine saykılı değerinden belirlenecektir.*

*F<sub>OSC</sub> = 16 Mhz olduğuna göre*

*T<sub>mak. Saykılı</sub> = 12 / f<sub>OSC</sub> formülünden hesaplanabilir.*

*T<sub>mak. Saykılı</sub> = 12 / (16 x 10<sup>6</sup>) = 0.75 μSaniyedir.*

*t = (Toplam makine saykılı) x (T<sub>mak. Saykılı</sub>)*

*20 μS = (Toplam makine saykılı) x (0.75 μS)*

*Toplam makine saykılı = 26.67 adettir. Makine saykılı tam olmak zorundadır. Üste tamamlarsak 27 olur.*

*27 = 2X + 3,*

*X = (27 - 3) / 2*

*X = 12, olduğunda elde edeceğimiz gecikme,*

*t = (2X + 3) x (T<sub>mak. Saykılı</sub>) = (2 x 12 + 3) x (0.75 μS) = 20.25 μS olur.*

*Yapılan hata 0.25 μS'dir. Tam değeri elde etmek mümkün değildir.*

*\*\*\*\*\*  
;Bu program **P3.1** ucunda istenilen özelliklere sahip kare dalga oluşturur.*

*\*\*\*\*\*  
ORG 00H ;programın başlangıç adresi*

*karedalga:*

```

Clr P2.1          ;p2.1 temizle
Acall gecikme20us ;20 μs bekle.
Acall gecikme20us ;20 μs bekle.
Setb p3.1         ;p3.1 set et.(1 yap)
Acall gecikme20us ;20 μs bekle.
Clr p3.1          ;p3.1 temizle(0 yap)
Acall gecikme20us ;20 μs bekle.
Acall gecikme20us ;20 μs bekle.
Acall gecikme20us ;20 μs bekle.
Setb p3.1         ;p3.1 set et.(1 yap)
Acall gecikme20us ;20 μs bekle.

```

```

Clr p3.1          ;p3.1 temizle(0 yap)
Acall gecikme20us ;20 µs bekle.
Setb p3.1         ; p3.1 kur.(1 yap)
Sjmp karedalga    ;sürekli yap

;*****
;20µs gecikme sağlayan alt program
;*****
Gecikme20us:
    Mov R0,#12      ;R0= 18
    Djnz R0,$        ;R0 =0 olana kadar devam et.
    Ret
end

```

Gecikme altprogramı zamanlayıcı ile de yapılabilir. 20 µs lik gecikme için zamanlayıcının 20 maine saykılı sayma yapması gerekir.

65536- 20= 65516= FFEC bulunan bu değer zamanlayıcıya yüklenecek olan başlangıç değeridir.

```

;*****
;20µs gecikme sağlayan alt program (zamanlayıcı1,model
kullanıldı)
;*****
gecikme20us:
    MOV TMOD,#10H    ;Zamanlayıcı1,kipl
    MOV TL1,#0ECH    ;TL1=ECH
    MOV TH1,#0FFH    ;TH1=FFH
    SETB TR1         ;Zamanlayıcıyı başlat.
bekle:
    JNB TF1,bekle    ;hala süre dolmadıysa (20µs) bekle.
    CLR TR1          ;süre doldu, timer durdur.
    CLR TF1          ;taşma bayrağını temizle.
    RET              ;ana programa geri dön.

```

## Sorular

1. 5, 20, 50 mS'lik zaman geciktirme altprogramlarını yazın.
2. 1 saniyelik zaman geciktirme altprogramlarını yazın.

3. Zamanlayıcı 0'ı 13 bit sayıcı olarak kullanmak için gerekli ayarlamaları yapan programı yazın.
4. 1. Zamanlayıcı 1'ı 16 bit sayıcı olarak kullanmak için gerekli ayarlamaları yapan programı yazın..
5. ZGD3 altprogramı bir defa çağrıldığında ne kadar geçikme sağlar? Hesaplayınız.

ZGD3:

```
MOV A,#25
TKR: SUBB A,#5
      NOP
      JNZ TKR
      RET
```

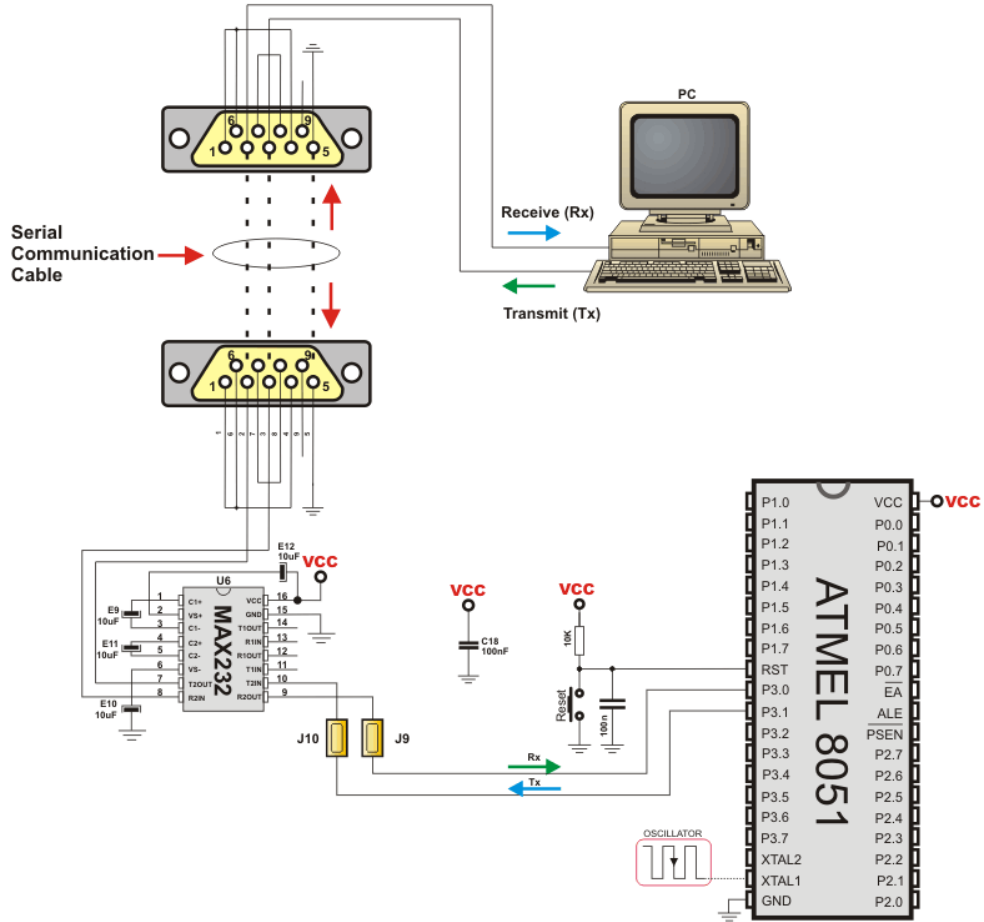
6.

7. ZGD4 altprogramı bir defa çağrıldığında ne kadar geçikme sağlar? Hesaplayınız.

ZGD4:

```
MOV A,#6
TKR: SUBB A,#50
      NOP
      JNC TKR
      RET
```

# Seri Haberleşme ve 8051 Seri Portu



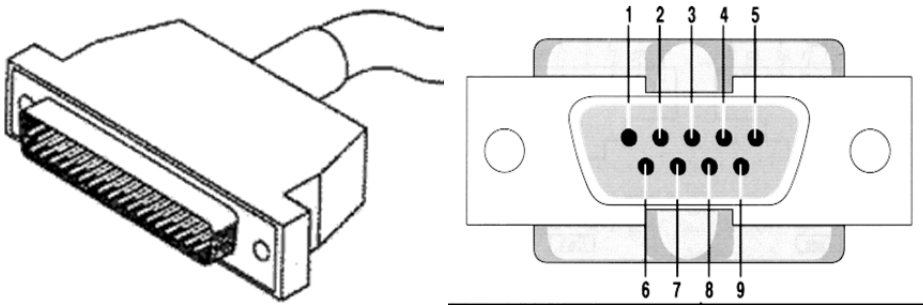
## Giriş

Seri veri iletimi iki bilgisayar arasında veri iletiminin daha az sayıda iletken kullanılarak gerçekleştirmek amacıyla geliştirilmiştir. Alıcı ve verici bilgisayarlar arası eş güdümü sağlamak amacıyla veri hattından ayrı hat kullanılırsa senkron kullanılmaz ise asenkron seri veri iletimi adı verilir. En yaygın kullanılan seri veri

haberleşme standardı RS-232'dir. Her bilgisayar üzerinde en az bir adet RS-232 ( COM port ) yer alır. Son yıllarda benzer bir haberleşme tekniğini kullanan USB standardı daha yaygın bir şekilde kullanılmaya başlamıştır. USB RS-232 gibi asenkron seri veri iletimi yapar fakat veri iletim hızı çok daha yüksektir. USB haberleşme portu son yıllarda üretilen mikrodenetleyicilerin üzerinde de standart olarak yer almaya başlamıştır.

## Asenkron Seri Veri İletimi

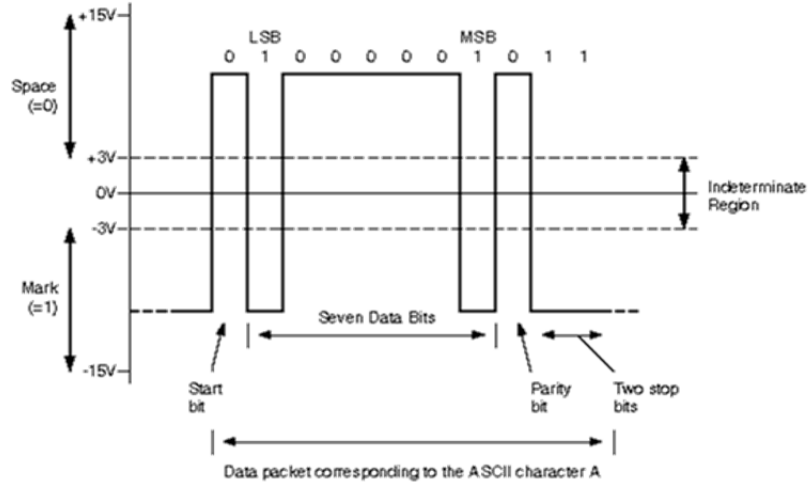
RS-232 Standardı ilk olarak 1962 yılında kullanılmaya başlanmış çeşitli değişimlere uğradıktan sonra 1969 yılında RS-232C olarak adını almıştır. RS-232D standardı ise RS-232 C üzerinde genişletme yapmak için 1987 yılında ortaya çıkmıştır. RS232 D standardı aynı zamanda EIA-232-D olarak da bilinir. RS-232 C Electronic Industries Assosiation (EIA) tarafından daha bilgisayarın başlangıç evresinde tasarlanmıştır. Gelecekte RS-232 C standardının her türlü gereksinime yanıt verebilmesi için çok sayıda farklı sinyal hatları sisteme eklenmiştir. Fakat günümüzde bu sinyal hatlarının çoğu kullanılmaktadır. RS-232 C ile kullanılan en yaygın konektör şekil-6.1'de gösterilen DB-25 tipi konektördür. Amerika Birleşik Devletleri dışında DB-25 konektörü V.24 ve V.28 olarak da adlandırılır. V.24 ve V.28 standartları, Consultative Committee On International Telegraph and Telephone (CCIT) olarak bilinen Uluslararası standartlar grubu tarafından kabul edilmiştir. 1984 yılında IBM'in firmasının AT bilgisayar kasalarının üzerinde çok yer kaplayan DB-25 konektörü yerine DB-9'u kullanmasıyla o yıldan sonra RS-232C için ikinci bir konektör tipi ortaya çıkmıştır. RS-232C hatlarının sinyal gerilim seviyeleri +12 V ve -12V' aralığındadır. Eski sistemlerde gerilim seviyeleri, +25 V ile -25 V değerlerine kadar ulaşabilir. Şekil-6.2'de RS-232C standardı ile bir karakterin iletimi için gerekli işaretin şekli gerilim seviyeleri ile birlikte gösterilmiştir.



Şekil-6.1 RS-232C'de kullanılan DB-25 ve DB-9 konektörleri.

## Mikrodenetleyiciler 8051 Uygulamaları

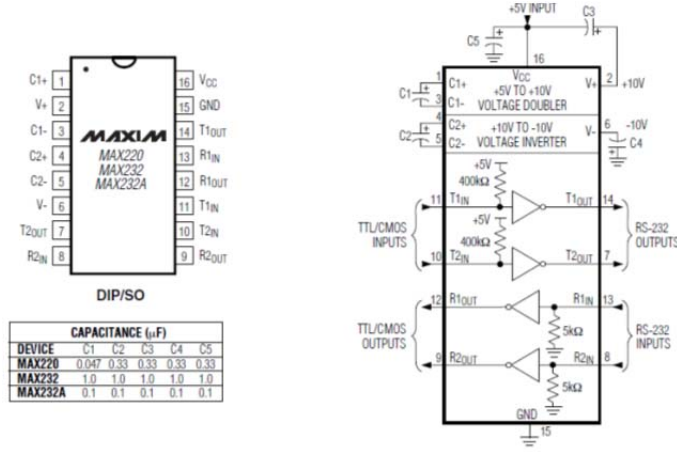




Şekil-6.2 RS-232C veri çerçevesi ve sinyal seviyeleri.

RS-232C haberleşme sisteminde veriyi Göderen birime DTE (Data Terminal Equipment) ve iletişimi sağlayan birime ise DCE (Data Communication Equipment) adı verilir. DTE çoğunlukla bilgisayardır, DCE ise modemdir. RS-232C işareti yükseltici kullanılmadan 15 metre mesafeye taşınabilir daha uzak mesafeler için sayısal işaretler analoğa dönüştürülerek her yerde rahatlıkla erişilebilen telefon hatları kullanılır.

Bilgisayar içerisinde paralel olarak yer alan sayısal (TTL seviyesi) veriler devreler vasıtasıyla seriye dönüştürülür ve daha sonra DC-DC dönüştürücü kullanarak işaretin seviyesi RS-232C seviyesine dönüştürülür. Alıcı tarafta ise RS-232C standırnda gelen veri DC-DC dönüştürücü kullanılarak TTL seviyesine dönüştürüldükten sonra seriden paralele dönüştüren devreler kullanılarak paralele dönüştürülür ve bilgisayara iletilir. Verici durumunda paralelden seriye alıcı durumunda seriden paralele dönüştüren devreler tümleşik olarak üretilir ve UART (Universal Asenkrenous Recevier Transmitter) adı verilir. Bu birim bir çok mikrogenetleyicide tümdevre ürerinde üretilir. TTL seviyesini RS-232C'ye ve RS-232C seviyesini TTL seviyesine dönüştüren birimler birleşik olarak tek bir tümdevre içerisinde üretilirler. Bu tümdevrelerin en yaygın kullanılanları MAX232, ILC232, MAX233'dür. Şekil-6.3'te MAX232 tümdevresinin mantık simgesi gösterilmiştir. Tümdevrenin düzgün çalışabilmesi için 5 V beslemeye ve 5 adet kondansatör bağlanması gerekir.



Şekil-6.3 MAX232 Tümdevresinin mantık simgesi ve bacak bağlantısı<sup>11</sup>.

## 8051 UART

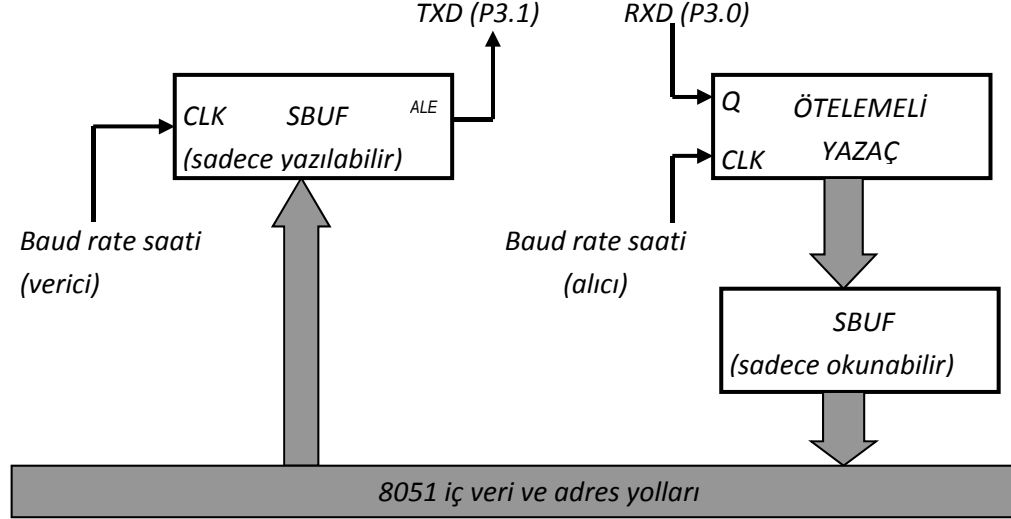
MCS-51 ailesi mikrodeneleyiciler üzerinde senkron ve asenkron olarak kullanılabilen bir adet seri port yer almaktadır. Bazı gelişmiş sürümlerinde seri port sayısı 2 veya daha fazla olabilmektedir. 8051 seri portunun bağlantısı P3.1 (TXD) ve P3.0 (RXD) hatlarından yapılır. İki özel amaçlı yazaç seri portun yazılım denetimini yapar. SBUF yazacı gönderilecek verinin yazıldığı ve gelen verinin okunduğu yazaçtır. Aslında aynı adresi ve ismi paylaşan iki farklı yazaçtır. Aynı adresi kullanmaları işlevlerinden dolayı sakınca yaratmamaktadır. Verici tampona sadece yazılabilir, alıcı tampon sadece okunabilir. Seri port denetim yazacı (SCON) 98H adresinde yer alır ve bit adreslenebilir. İçerisinde yer alan bitlerin bazıları durum ve bazıları ise denetim bitleridir. Kontrol bitleri seri portun çalışma kipini belirlerken, durum bitleri alma veya gönderme işleminin tamamlandığını gösterir. Durum bitleri yazılım veya programlı kesme ile denetlenebilir. Şekil-6.4'de 8051 seri portunun yapısı gösterilmektedir.

8051'in seri portu karakter tabanlı senkron veya asenkron veri iletimi yapar. Aynı anda veri alıp gönderebilir (full duplex çalışma). İletişim hızı zamanlayıcı kullanılarak veya sistem osilatörü kullanılarak ayarlanabilir. Asenkron veri iletiminde 8051'in seri portu 8 bit karakter verisinin başına ve sonuna alıcı birimi ile uyumlu çalışabilmesi için başla ve dur bitleri ekleyerek 10 bitlik bir çerçeve oluşturur. Veri iletiminde hata

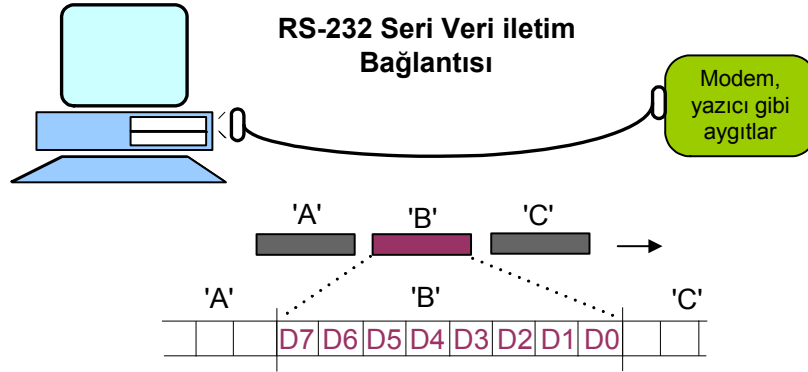
<sup>11</sup> Kaynak <http://www.maxim-ic.com>

## Mikrodeneleyiciler 8051 Uygulamaları

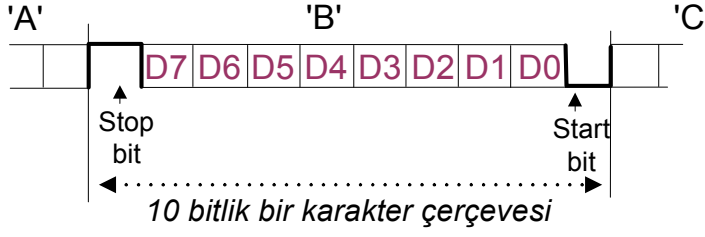
denetimi yapılmak istendiğinde 10 bit çerçeveye bir bit eşlik biti eklenerek 11 bitlik çerçeve elde edilir. Şekil-6.5'te çerçevesiz, 10 bit çerçevesiz ve 11 bit çerçevesiz karakter paketlerinin oluşturulması gösterilmiştir. Şekil-6.5.a'da yer alan karakter paketi 8051'in senkron seri veri iletiminde kullanılır. Şekil-6.5.b ve c'deki 10 ve 11 bitlik veri paketleri ise asenkron veri iletiminde kullanılır.



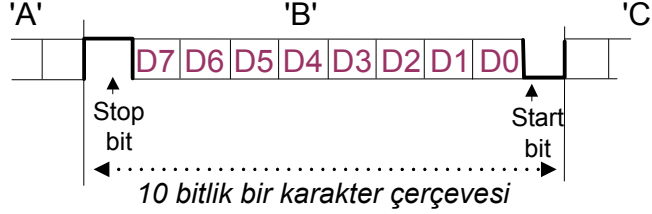
Şekil-6.4 8051'in seri portunun yapısı.



a. Seri port kullanarak bir karakterin çerçevesiz iletimi.



b. Başla ve dur bitlerinin eklenmesi ile oluşturulan 10 bitlik bir karakter çerçevesi.



c. Başla ve dur bitlerine eşlik bitinin eklenmesi ile oluşturulan 11 bitlik bir karakter çerçevesi.

Şekil-6.5 bir karakterin çerçevesiz, 10 bit ve 11 bit çerçeveli paketlenmesi.

## Seri Port Çalışma kipleri

8051'in seri portu asenkron ve senkron çalışabilen 4 adet çalışma kipine sahiptir. Çalışma kipleri SCON yazıcının SM0 ve SM1 bitleri kullanılarak belirlenir. Çizelge-6.1'de SCON yazıcında yer alan denetim ve durum bitlerinin görevleri belirtilmiştir. SCON yazıcında yer alan SM2 biti ise çoklu ortam veri iletimi yapılmak istendiğinde kurulur diğer durumlarda ise temizlenir. REN bitine alıcıyı çalıştırma durdurma görevi verilmiştir kurulduğunda alıcı ilk algıladığı START biti ile ötelemeli yazaca veri bitlerini yerleştirmeye başlar. Bu bit temizlendiğinde alıcı veri almayı durdurur. TB8 biti kip 2 ve kip 3'te vericinin dokuzuncu biti olarak çalışır. RB8 biti ise aynı görevi alıcı için yapar. Bu bitlerin kip 0 ve kip 1'de herhangi bir görevi yoktur. TI biti verici kesme bayrağıdır. Verici bufferına yazılan verinin son biti gönderildikten sonra bu bit kurularak kesme isteğinde bulunulur. RI biti alıcı kesme bayrağı olarak görevlendirilmiştir. Alıcı son biti aldıktan sonra RI bitini kurarak kesme isteğinde bulunur. Kesme algılandıktan sonra TI ve RI bayrakları yazılım ile temizlenmelidir.

## Mikrodenetleyiciler 8051 Uygulamaları

BİT	SİMGE	ADRES	TANIM
SCON.7	SM0	9FH	Seri port kip seçme biti 0
SCON.6	SM1	9EH	Seri port kip seçme biti 1
SCON.5	SM2	9DH	Seri port kip seçme biti 2
SCON.4	REN	9CH	Alıcı izin verme, veri almak için kurulmalıdır.
SCON.3	TB8	9BH	Kip1 ve 3'te vericinin dokuzuncu biti olarak kullanılır.
SCON.2	RB8	9AH	Kip1 ve 3'te alıcının dokuzuncu biti olarak kullanılır.
SCON.1	TI	99H	Verici kesme bayrağı, gönderilen karakterin tüm bitleri gönderildikten sonra donanım tarafından kurulur. Yazılım ile temizlenir.
SCON.0	RI	98H	Alıcı kesme bayrağı, karakterin alımı bittikten sonra donanım tarafından kurulur. Yazılım ile temizlenir.

Çizelge-6.1 SCON yazacının bitlerinin isimleri, adresleri ve görevleri.

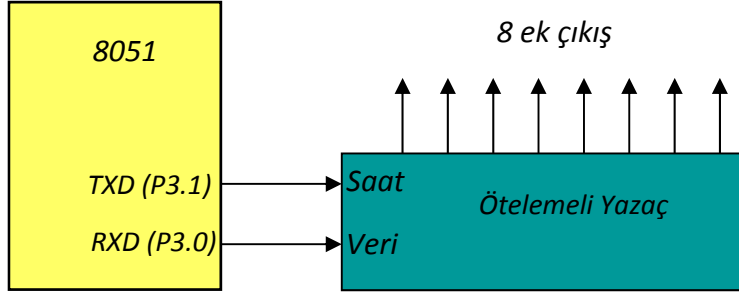
SM1	SM0	KİP	TANIM	BAUD RATE
0	0	0	KAYAR YAZAÇ	Sabit ( $f_{osc}/12$ )
0	1	1	8 BİT UART	Değişken (T1 tarafından ayarlanır)
1	0	2	9 BİT UART	Sabit ( $f_{osc}/32$ veya $/64$ )
1	1	3	9 BİT UART	Değişken (T1 tarafından ayarlanır)

Çizelge-6.2 Seri port çalışma kipi seçme bitleri.

Çizelge-6.2'de SM0 ve SM1 bitlerinin kullanımı ve seri port çalışma kipleri gösterilmiştir. Kip 0'da 8051 senkron 8 bit veri iletimi yapar. Kip 1'de 8 bit UART, 2 ve 3'te ise 9 bit UART olarak çalışır.

### Kip 0 8 Bit Kayar Yazaç Kipi

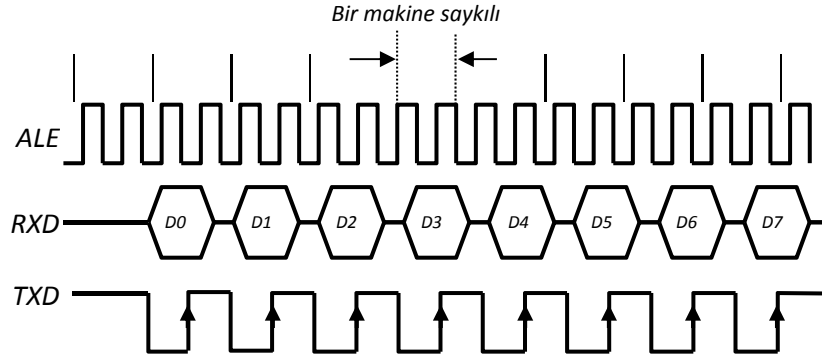
Kip seçme bitleri SM0 ve SM1 bitleri sıfır yapılarak kayar yazaç çalışma kipi seçilir. Kip 0, 8051'in diğer birimler ile senkron 8 bit çerçevesiz seri veri iletimi yaptığı tek çalışma kipidir. Bu çalışma kipinde REN biti sıfır yapılarak verici, 1 yapılarak alıcı olarak çalışması sağlanır, seri veri RXD hattından gönderilir veya alınırken, TXD hattı saat çıkışı olarak kullanılır. 8 bitin önce en düşük değerli biti gönderilir ve önce en düşük değerli biti alınır. Baud rate makine saykılı hızındadır. Kayar yazaç kipi 8051'in giriş/çıkış hattının sayısı yeterli olmadığı durumlarda hat sayısını arttırmak için kullanılır. Şekil-6.6'da örnek bağlantı verilmiştir.



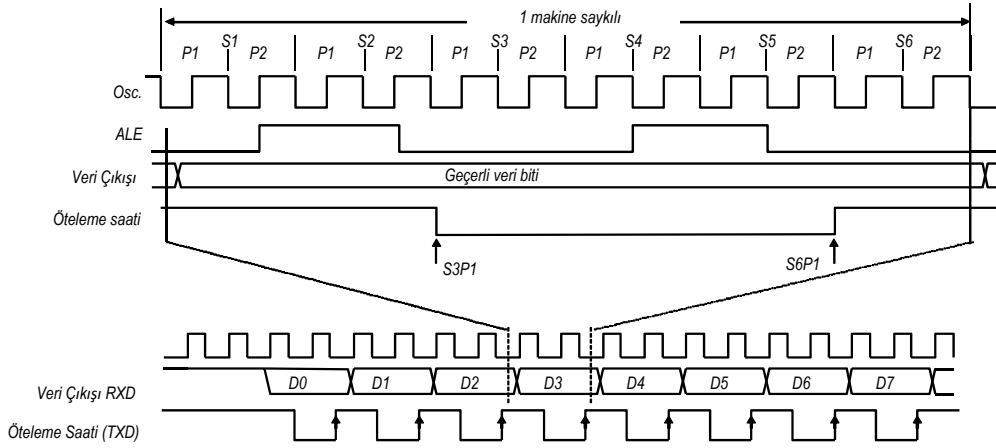
Şekil-6.6 Kayar yazaç kipinde 8051'in çıkış port hatlarının sayısının arttırılması.

Şekil-6.7'de kip 0'da verici olarak çalışmada zamanlama diyagramı gösterilmiştir. Kip 0'da gönderme işlemi SBUF'a yazma yapan her komut ile başlatılabilir. Veri RXD, P3.0'dan dışarıya doğru ötelenirken saat işareti de TXD P3.1 hattından gönderilir. Gönderilen her bit RXD hattında bir makine saykılı geçerli olarak kalır. Her makine saykılında TXD'dan gönderilen saat saykılı S3 P1'de düşük seviyeye geçer, S6 P1'de tekrar yüksek seviyeye geçer. Şekil-6.8'de D3 bitinin gönderilme zamanlaması ayrıntılı olarak gösterilmiştir.

Kip 0'da alma işlemi REN biti kurulu iken RI biti temizlendiğinde başlatılmış olur. Genel kural olarak önce seri port ayarlamaları sırasında REN biti kurulur, sonra RI biti temizlenerek alma işlemi başlatılır. RI temizlendiğinde TXD'dan saat çıkışı gönderilir ve bir sonraki bit bu saat işareti ile RXD hattına gelir. Böylece gelen veri TXD hattından gönderilen saat işareti ile senkronize edilmiş olur. Veri RXD hattına saatin yükselen kenarı ile aktarılır.



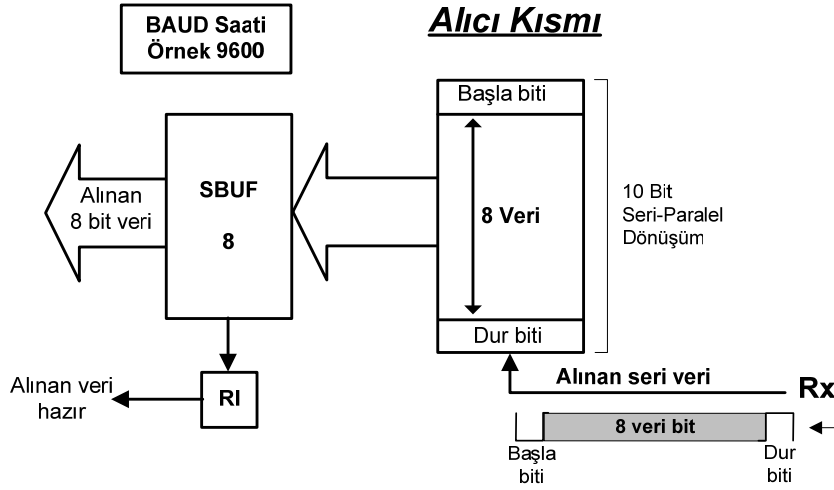
Şekil-6.7 Kip 0 kayar yazaç kipinde verici olarak çalışmada zamanlama.



Şekil-6.8 Kip 0 çalışmada seri port verici zamanlaması.

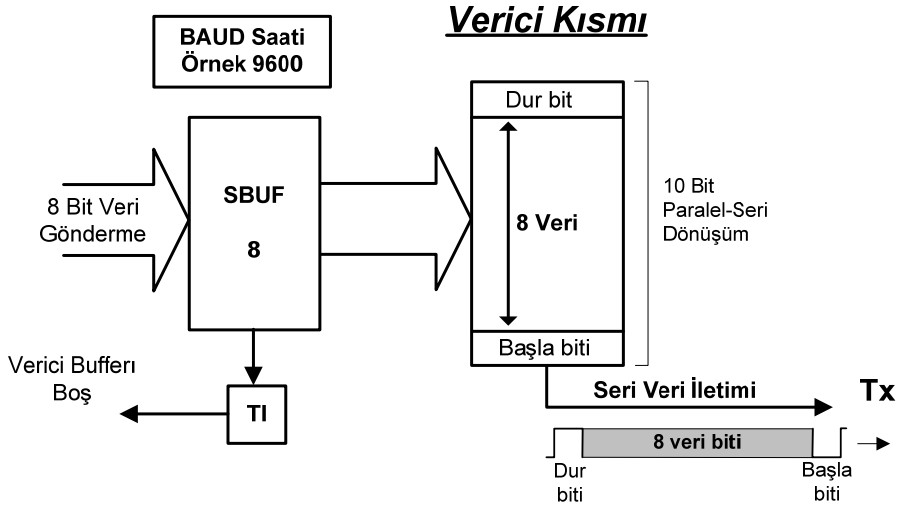
### Kip 1 Değişken Hızlı 8 Bit UART

Kip 1'de seri port 8 bit değişken baud rate UART olarak çalışır. Bu çalışma kipinde 8 veri biti bir adet başla ve 1 adet dur biti olmak üzere 10 bit çerçevelenmiş veri iletir. Başla ve dur bitleri alıcı ile verici birim arasındaki eş güdümü sağlamak amacıyla eklenir. Bu bitler seri port tarafından gönderilen veriye eklenir, gelen veriden ise ayıklanarak SBUF'a sade 8 adet veri biti iletilir. 8 bit UART çalışma kipinde iletim hızı (baud rate) 8051'de sadece zamanlayıcı 1 taşıma oranına göre belirlenir, 8052'de ise zamanlayıcı 2 veya 1 tarafından veya her ikisinin birlikte kullanılmasıyla belirlenir. Biri gönderme hızını diğeri ise alma hızını belirler.



Şekil-6.9 Verinin gönderilmesi.

Şekil-6.9'da gelen seri 10 bit çerçeve verinin dur ve başla bitlerinden ayıklanarak paralele dönüştürölüp SBUF'a aktarılması gösterilmiştir. Seri port gelen veriyi SBUF'a aktardıktan sonra RI bayrağını 1 yaparak kesme isteğinde bulunur. SBUF okunduktan sonra RI bayrağı mutlaka temizlenmelidir. Şekil-6.10'da SBUF'a yazılan paralel 8 bit verinin başına başla, sonuna dur bitlerinin eklenmesi ve seriye dönüştürölerek TXD hattından gönderilmesi gösterilmiştir.



Şekil-6.10 8bit verinin kip 1'de gönderilmesi.

Seri prottan gönderme işlemi SBUF'a yazma işlemi takip eden makine saykılında kendiliğinden başlar. Veri alma işleminin başlayabilmesi için REN bitinin 1 olması ve



RXD hattında oluşacak düşen kenarın algılanması ile başlar. Alıcı yanlış başla biti algılama devresine sahiptir, ilk negatif geçiş algılandıktan sonra sekiz sayma değeri kadar sıfır durumunda kalarak ikinci bir geçiş bekler eğer geçiş gelmezse sayıcının değeri sıfırlanır ve aptal konumunda beklemeye devam eder. Bu durumda alıcının gürültü tarafından tetiklendiği varsayılır. Doğru tetikleme alındığında başla biti atlanarak veri bitleri seri port yazacına sırayla ötelenir.

### **Kip 2 Sabit Hızlı 9-Bit UART**

SM1=1 ve SM0=0 ise seri port kip 2'de çalışır. Bu çalışma kipinde on bir bit iletilir ve alınır, bir başla sekiz veri bir adet programlanabilir dokuzuncu veri biti veya eşlik biti de olabilir, bir adet dur biti. Verici durumunda dokuzuncu veri biti SCON içerisindeki TB8'e yazılır. Alıcı durumunda bu bit RB8'e yazılır. Bu çalışma kipinde Baud rate sabittir, SMOD biti 0 yapılırsa osilatör frekansı 32 1 yapılırsa 64'e bölünerek elde edilir.

### **Kip 3 Değişken Hızlı 9-Bit UART**

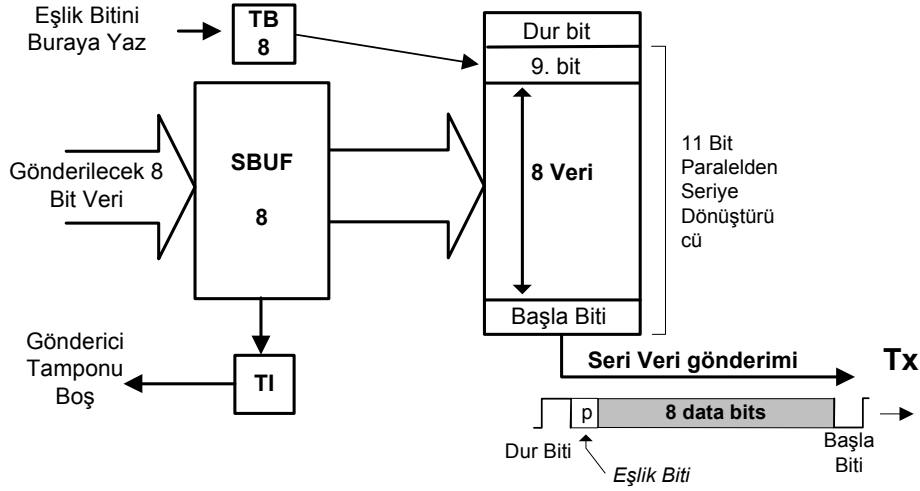
Kip 3'ün kip 2'den tek farkı iletim hızının zamanlayıcı 1 taşması ile ayarlanabilir olmasıdır. Kip 2 ve 3'te gönderilen ve alınan çerçeve toplamda 11 bittir. Bunlar başla biti, dur biti 9 veri bitidir. Bu çalışma kiplerinde gönderilmek istenen verinin 8 biti SBUF yazacına yazılır, dokuzuncu bit ise SCON'da yer alan TB8 bitine yazılır. 8051'in seri portu şekil-6.11'de gösterildiği gibi bu iki veriyi 9 bit olarak kayar yazaçta birleştirir başla ve dur bitlerini ekleyerek en başla bitinden başlayarak sıra ile karşı tarafa gönderir. TB8'in içeriği gönderildikten sonra TI bayrağı kurularak 8051'den kesme isteğinde bulunulur.

Kip 2 ve 3'te alıcının çalışması vericinininkine benzerdir. RXD hattından seri olarak gelen veri paketi şekil-6.12'de gösterildiği gibi sıra ile 11 bit seriden paralele dönüşüm yapan kayar yazaca alınır. Başla ve dur bitleri ayrıldıktan sonraki 8 bit SBUF'a yüklenirken dokuzuncu bit RB8 bitine yazılır ve RI bayrağı kurularak 8051'den kesme isteğinde bulunur. Kip 2 ve 3 8 bit veri iletimi ve eşlik biti denetimi yapılmak istenen haberleşme sistemlerinde kullanılır.

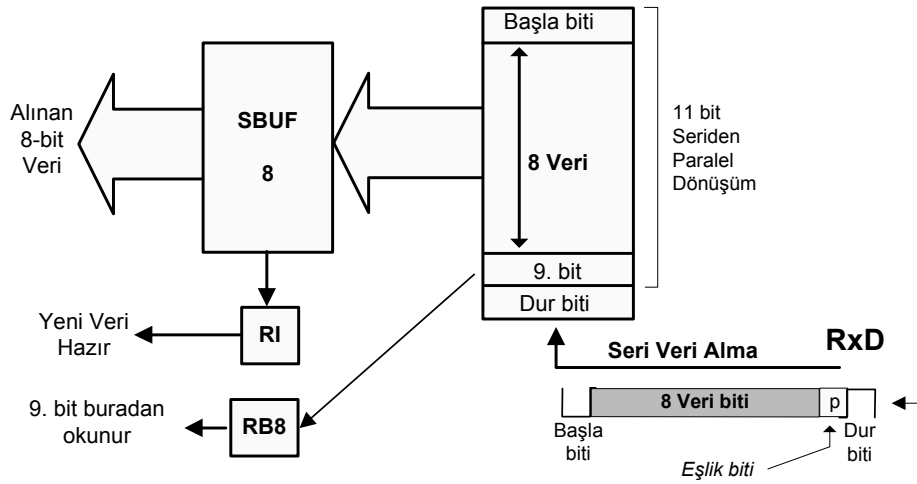
## **Yazaçların Ayarlanması**

8051 seri portunu istenilen şekilde kullanmak için öncelikle ayarlama işlemlerini yapılması gerekir. Çalışma kipine göre ayarlanacak yazaçlar faklılık gösterebilir fakat tüm çalışma kiplerinde ayarlama işlemine SCON yazacından başlanmalıdır.

Çalışma kipi belirlendikten sonra değişken veri hızı kullanılacaksa zamanlayıcı 1'in yeniden yükleme değeri ve eğer kullanılacaksa PCON içerisinde yer alan SMOD biti ayarlanmalıdır.



Şekil-6.11 9 bit verinin kip 2 ve 3'de gönderilmesi.



Şekil-6.12 9 bit verinin kip 2 ve 3'de alınması.

### *Alıcının Etkinleştirilmesi*

Alıcı etkinleştirme biti (REN) SCON içerisinde yer alır. Karakter alımının başlayabilmesi için yazılım ile bu bitin kurulması gerekir. Bu işlem genellikle

program başında tüm yazaçlar ayarlanırken yapılır. İki farklı komut kullanılarak bu bit kurulabilir.

SETB REN

Veya

ORL SCON, #00010000B

### Eşlik Bitinin Eklenmesi

Dokuzuncu veri biti genellikle eşlik biti olarak kullanılır. PSW içerisindeki P bayrağı akümülatör içerisinde yer alan karakterin içerisinde yer alan 1'lerin sayısı çift (even) ise kurulur, diğer durumda sıfırlanır. İletişim 8 veri ve bir eşlik biti içeriyorsa 8 bit veri akümülatöre alınır bu P bayrağını kuracak veya sıfırlayacaktır önce P bayrağı TB8'e yazılır ve sonra akümülatör SBUF'a yazılır.

MOV C, P ;Çift eşlik bitini TB8'e yerleştir.

MOV TB8, C ;Eşlik biti dokuzuncu veri biti oldu.

MOV SBUF, A ;8 veri bitini SBUF'a yaz.

Tek eşlik biti kullanılırsa;

MOV C, P ;Çift eşlik bitini eldeye al.

CPL C ;Tek eşliğe dönüştür.

MOV TB8, C ;Eşlik biti dokuzuncu veri biti oldu.

MOV SBUF, A ;8 veri bitini SBUF'a yaz.

Eşlik biti kullanımı sadece kip 2 ve 3 ile sınırlı değildir kip 1'de de eşlik biti kullanılabilir. 8 veri bitinden bir tanesi eşlik biti için ayrılabilir. Böylece 7 veri biti ve bir eşlik biti iletilir.

CLR ACC.7 ;YDB'ti temizle, çift eşlik P bayrağındadır.

MOV C, P ;Çift eşlik bitini C'ye kopyala.

MOV ACC.7, C ;Çift eşlik bitini YDB'e yerleştir.

MOV SBUF, A ;Karakteri gönder, 7 veri ve bir eşlik biti.

### Kesme Bayrakları

SCON içerisindeki RI ve TI kesme bayrakları 8051'in seri iletiminde önemli görevler üslenmişlerdir. Bu iki bayrak donanım tarafından kurulur fakat yazılım tarafından temizlenir. RI bir karakterin alındıktan sonra kurulur ve SBUF'ın dolduğunu gösterir. Bu durum program tarafından denetlenir veya kesmeye izin verilir. Eğer 8051 seri

kanaldan veri bekliyor ise bu bayrağı okur ve test eder kurulu ise temizler ve SBUF'ı okur. Sıfır ise kurulana kadar okuma ve test işlemine devam eder.

BEKLE: JNB RI, BEKLE ;RI kurulana kadar test et.  
CLR RI ;RI'yi temizle.  
MOV A, SBUF ;karakteri oku.

TI bayrağı gönderme işleminin bitiminde kurulur, gönderme tamponunun boş olduğunu gösterir. Yazılımın yeni bilgi göndermeden önce bu bayrağı test ederek tamponun boş olup olmadığını test etmesi gerekir. Aşağıdaki komutlar akümülatör içerisindeki veriyi gönderir.

BEKLE: JNB TI, BEKLE ;TI kurulana kadar test et.  
CLR TI ;TI'yi temizle.  
MOV SBUF, A ;karakteri YAZ.

### Seri Port İletişim Hızının Belirlenmesi

Seri port haberleşmesin veri iletim hızı baud rate olarak tanımlanır. Birim zamanda gönderilen bit sayısı baud rate BPS veya iletim hızı olarak adlandırılır. 8051'in seri portu her çalışma kipinde farklı iletim hızlarında haberleşme sağlayabilir. Kip 0'da veri iletim hızı üretim sırasında şekil-6.13'te gösterildiği gibi osilatör frekansının 12'ye bölünmesi ile elde edilir. Kip 0'da 8051'in osilatör frekansı 12 Mhz ise veri iletim hızı 1 Mhz'dir. Bu kipte veri iletim hızını değiştirmek için osilatör frekansının değiştirilmesi gerekir. Kip 2'de iletim hızı kip 0'dakine benzerdir fakat bu çalışma kipinde şekil-6.14'te gösterildiği gibi osilatör frekansı 64'e bölünerek iletim hızı belirlenir. Bu çalışma kipinde daha hızlı veri iletimi yapılmak isteniyorsa SMOD biti kurularak osilatör frekansının 32 bölünmesi sağlanır ve iletim iki kat hızlanır. Şekil-6.15'te gösterildiği gibi kip 1 ve 3'te iletim hızları aynı şekilde zamanlayıcı taşmasının 32'ye bölünmesi ile elde edilir. İletim hızı arttırılmak isteniyorsa SMOD biti 1 yapılarak iki katına çıkarılabilir. Ayrıca bu çalışma kiplerinde T1 taşmasını ayarlanarak iletim hızı arttırılabilir.

Kip 0'da iletim hızı;

$$\text{Baud Rate} = \frac{f_{osc}}{12}$$

Kip 2'de iletim hızı;

### Mikrodenetleyiciler 8051 Uygulamaları

Eğer  $SMOD=0$  ise

$$\text{Baud Rate} = \frac{f_{osc}}{64}$$

Eğer  $SMOD=1$  ise

$$\text{Baud Rate} = \frac{f_{osc}}{32}$$

Kip 1 ve 3'te iletim hızı;

Eğer  $SMOD=0$  ise

$$\text{Baud rate} = f_{osc} / (384 \times (256 - TH1))$$

Eğer  $SMOD=1$  ise

$$\text{Baud rate} = f_{osc} / (192 \times (256 - TH1))$$

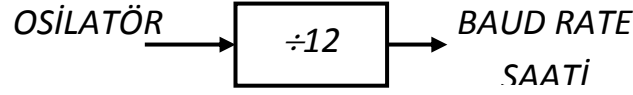
PCON bit adreslenebilir olmadığı için akümülatöre okunur istenen bit değiştirildikten sonra tekrar yazılır veya aşağıda gösterilen VEYA ve VE .

SMOD bitinin kurulması;

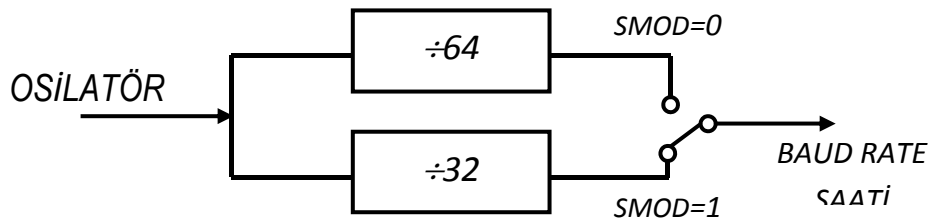
ORL PCON, #1000000B ;SMOD=1

SMOD bitinin temizlenmesi;

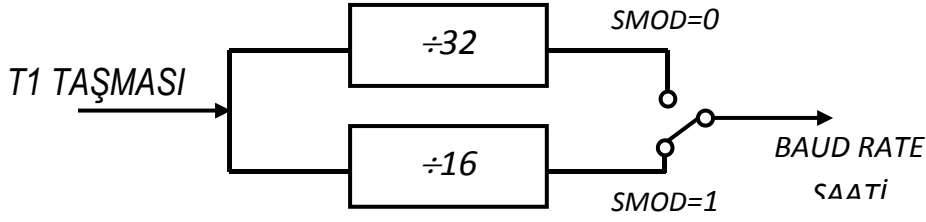
ANL PCON, #01111111B ;SMOD=0



Şekil-6.13 Kip 0'da baud rate saatinin üretilmesi.



Şekil-6.14 Kip 2'de baud rate saatinin üretilmesi.



Şekil-6.15 Kip 1 ve 3'te baud rate saatinin üretilmesi.

### Baud Rate Saati Olarak T1'in Kullanılması

8051'de en kolay baud rate saati zamanlayıcı 1 ile üretilebilir. TH1'e gerekli yeniden yüklenecek sayı yerleştirilir, her taşmada bu değer TL1'e yeniden yüklenerek istenilen değerde saat işareti elde edilebilir. Bu işlem için öncelikle TMOD yazacının T1'i kip 2'de çalıştıracak şekilde ayarlanması gerekir. Ayarlama aşağıdaki gibi yapılabilir.

```
MOV TMOD, #0010XXXXB
```

Baud rate ayarlamasının tek yolu bu değildir, zamanlayıcı 1'de 16 bit çalışma kipinde bu tür ayarlamalar yapılabilir. Taşma sonrası kesmeye izin verilir ve kesme alt programı her defasında belirlenen değeri yeniden TH1 ve TL1 yazaçlarına yükler. Başka bir yöntemi ise zamanlayıcı 1'e dışarıdan tetikleme işareti uygulayarak saymasını sağlamaktır. Baud rate saati elde etmek için T1'in taşması SMOD=1 ise 16'ya SMOD=0 ise 32'ye bölünür.

Baud rate göre sayıcıya yüklenecek değer aşağıdaki yöntemle bulunur.

Kip 1 ve 3'te iletim hızı;

Eğer SMOD=0 ise

$$TH1 = 256 - ((f_{osc} / 384) / \text{Baud rate})$$

Eğer SMOD=1 ise

$$TH1 = 256 - ((f_{osc} / 192) / \text{Baud rate})$$

Fosc = 11.059Mhz ise 19,200 baud rate için zamanlayıcı değeri hesaplarsak;

$$TH1 = 256 - ((f_{osc} / 384) / \text{Baud rate})$$

$$TH1 = 256 - ((11059000 / 384) / 19200)$$

$$TH1 = 256 - ((28,799) / 19200)$$

$$TH1 = 256 - 1.5 = 254.5$$

Alt değere tamamlandığında 254 olur ve elde edilen iletişim hızı 14,400 olur. Eğer 255'e tamamlarsak 28,800 hızına ulaşırız. Her iki değerde istenilenden değerden çok uzak ise SMOD biti 1 yapılarak yeni değer hesaplanmalıdır.

$$\begin{aligned} \text{SMOD} &= 1 \text{ yapıldığında;} \\ \text{TH1} &= 256 - ((f_{\text{osc}} / 192) / \text{Baud rate}) \\ \text{TH1} &= 256 - ((11059000 / 192) / 19200) \\ \text{TH1} &= 256 - ((57699) / 19200) \\ \text{TH1} &= 256 - 3 = 253 \end{aligned}$$

Eğer 12MHz kristal kullanılırsa zamanlayıcının frekansı 1000KHz'dir. Zamanlayıcının 38.4 kHz'de taşma yapması isteniyorsa  $1000 \div 38.4 = 26.04$  saat vurusu yuvarlama yapıldığında 26 olarak alınabilir. Taşma FFh'den 00h'ye geçişte gerçekleştiğine göre zamanlayıcıya 256-26 yüklenmelidir. Kısaca -26 yazılır assembler programı onu onaltılık tabana dönüştürür. (E6h) bu değer TH1'e yüklenir.

MOV TH1, #-26 Veya

MOV TH1, #0E6H

Yazılabilir.

BAUD RATE	$f_{\text{osc}}$	SMOD	TH1	GERÇEK RATE	BAUD HATA
9600	12.000MHz	1	-3 (F9H)	8923	%7
2400	12.000MHz	0	-13 (F3H)	2404	%0.16
1200	12.000MHz	0	-26 (E6H)	1202	%0.16
19200	11.059MHz	1	-3 (FDH)	19200	0
9600	11.059MHz	0	-3 (FDH)	9600	0
2400	11.059MHz	0	-12 (F4H)	2400	0
1200	11.059MHz	0	-24 (E8H)	1200	0

Çizelge-6.3 Baud rate ve hata oranları

Yuvarlamalardan dolayı çok az bir hata oluşacaktır. %5'in altındaki hata oranları asenkron veri iletiminde kabul edilebilir. Hatasız baud rate'ler için 11.059MHz'lik kristaller kullanılmalıdır. Çizelge-6,3'de 12 ve 11.059MHz'lik kristaller

kullanıldığında kullanılması gereken yeniden yükleme değerleri ve hata payları verilmiştir.

Baud Rate	Fosc. (MHz)					Bit SMOD
	11.0592	12	14.7456	16	20	
150	40 h	30 h	00 h			0
300	A0 h	98 h	80 h	75 h	52 h	0
600	D0 h	CC h	C0 h	BB h	A9 h	0
1200	E8 h	E6 h	E0 h	DE h	D5 h	0
2400	F4 h	F3 h	F0 h	EF h	EA h	0
4800		F3 h	EF h	EF h		1
4800	FA h		F8 h		F5 h	0
9600	FD h		FC h			0
9600					F5 h	1
19200	FD h		FC h			1
38400			FE h			1
76800			FF h			1

### Örnek 6,1

Seri portu 2400 baud rate hızında 8 bit UART olarak ayarlayın. Baud rate saatini zamanlayıcı 1'i kullanarak elde edin.

#### ÇÖZÜM:

Bu örnekte SMOD, TCON, TMOD ve TH1 yazaçlarının içerikleri aşağıdaki gibi düzenlenmelidir.

	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
<b>SCON:</b>	0	1	0	1	0	0	1	0
	GTE	C/T	M1	M0	GTE	C/T	M1	M0
<b>TMOD:</b>	0	0	1	0	0	0	0	0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
<b>TCON:</b>	0	1	0	0	0	0	0	0
<b>TH1:</b>	1	1	1	1	0	0	1	1

8 bit UART için SM1=0, SM0=1 yapılmalıdır. REN=1 yaparak veri almasına izin verilir, ilk karakterin gönderilebilmesi için TI=1 yapılmalıdır. Zamanlayıcı 1 için TMOD

## Mikrodenetleyiciler 8051 Uygulamaları



içerisinde 8 bit yeniden yüklemeli zamanlayıcı olarak çalışma kipinin seçilebilmesi için  $M1=1$ ,  $M0=0$  yapılır. TCON içerisinde ise  $TR1=1$  yapılarak zamanlayıcı 1çalıştırılır. Kullanılmayan bitler 0 olarak yazılmıştır.

2400 baud rate için TH1'e yüklenmesi gereken değer hesaplanır

$$\begin{aligned} TH1 &= 256 - ((f_{osc} / 384) / \text{Baud Rate}) \\ &= 256 - ((12\,000\,000 / 384) / 2400) \\ &= 242.979 = 243 \text{ olarak üste tamamlanır.} \end{aligned}$$

AYAR:

```
MOV SCON, #52H    ;Seri port kip 1 seçildi.
MOV TMOD, #20H    ;Zamanlayıcı 1 kip 2 seçildi.
MOV TH1, #243     ;2400 baud rate için sayıcı değeri.
SETB TR1          ;Zamanlayıcıyı başlat.
```

END

### Örnek 6.2

Akümülatör içerisinde bulunan 7 bit ASCII karakteri seri porttan sekizinci biti tek eşlik biti olarak gönderen alt programı yazın. Akümülatör içeriğini alt program sonunda korusun.

ÇÖZÜM:

GONDER:

```
MOV C, P          ;Eşlik bitini elde bayrağına al.
CPL C             ;Çift eşlik bitini tek eşliğe dönüştür.
MOV ACC.7, C      ;Eşlik bitini karaktere ekle.
```

TKR:

```
JNB TI, TKR      ;Bir önceki karakter gitti mi? hayır
bekle
CLR TI           ;Evet, bayrağı temizle.
MOV SBUF, A      ;Karakter gönder.
CLR ACC.7        ;Akümülatörü başlangıç durumuna getir.
RET
```

END

Programın başında PSW içerisinde yer alan eşlik biti elde bayrağına alınmıştır, fakat 8051 çift eşlik biti kullandığı için tümlenmiştir. Veri iletimi 8 bit yapılacağı için akümülatörün 7 numaralı bitine yerleştirilmiştir. Sonraki adımda daha önceki karakterin gönderilme işinin tamamlanıp tamamlanmadığı test edilmiş ve

tamamlanmadıysa beklenmiş, tamamlandıysa bayrak temizlenmiş ve karakter seri port tamponuna yazılmıştır. Son adımda ise akümülatöre yerleştirilen eşlik biti atılmış, alt program çıkışında akümülatörün değeri başlangıç değerine eşitlenmiştir.

### Örnek 6.3

Seri porttan 7 bit ASCII karakter alan, aldığı bitin sekizinci bitini tek eşlik biti olarak kabul eden, eğer hatalı eşlik biti varsa elde bayrağı kurulu olarak alt programdan dönen alt programı yazın.

ÇÖZÜM:

GEL:

```
JNB RI, GEL ;Karakter geldi mi? Gelmediyse bekle.
CLR RI      ;Geldiyse bayrağı temizle.
MOV A, SBUF ;Gelen karakteri tampondan oku.
MOV C, P    ;Tek eşlik için P=1 olması gerekir.
CPL C       ;Hata varsa bildirmek için tümle.
CLR ACC.7   ;Eşlik bitini at.
RET
```

END

### Örnek 6.4

Osilatör frekansı 6 Mhz olan 8051'in seri portunu 900 baud rate hızında 8 bit veri iletimi yapacak şekilde ayarlayan altprogramı yazın.

$$TH1 = 256 - ((f_{osc} / 384) / \text{Baud Rate})$$

$$= 256 - ((6\,000\,000 / 384) / 900)$$

$$= 238,6 = 239 \text{ alınır}$$

SERP:

```
MOV SCON, #52H ;Seri port kip 1 seçildi.
MOV TMOD, #20H ;Zamanlayıcı 1 kip 1 seçildi.
MOV TH1, #239   ;900 baud rate için sayıcı değeri.
SETB TR1       ;Zamanlayıcıyı başlat.
```

End

### Örnek 6.5 Bcd-İkili Dönüşüm.

```
/* Bu alt program akümülatörün içindeki BCD değeri
ikiliye (hex) dönüştürür.
Etkilenen yazaçlar;
```

PSW.CY, PSW.Z, PSW.P, Acc \*/

BCD\_ikili:

```
pushb      ; B'yi yedekle
pushacc    ; A'yı yedekle
anl a,#0f0h ; Yüksek değerli 4 biti koru
swapa      ; Düşük kısma al
mov b,#10  ; 10'lar basamağı
mul ab     ; Çarp
mov b,a    ; B'de sakla
pop acc    ; BCD değeri yedekten al
anl a,#0fh ; düşük değerli dört biti koru
add a,b    ; yüksek kısma ekle
pop b      ; B'yi yedekten al
ret        ; Ana programa geri dön
```

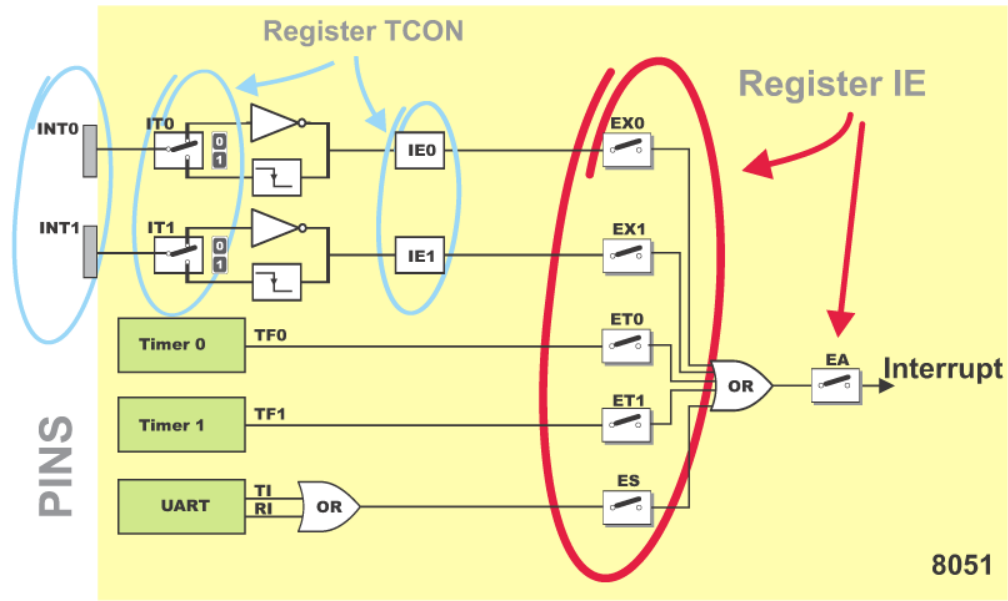
## Sorular

1. Osilatör frekansı 24 Mhz olan 8051'in seri portunu 1200 baud rate hızında 9 bit veri iletimi yapacak şekilde ayarlayan altprogramı yazın.
2. Sıfırla sonlandırılmış 8 bit sayıların oluşturduğu diziye 2400 baud rate hızında 8051'in seri portundan gönderen programın;
  - a. Algoritmasını yazın ve akış diyagramını çizin.
  - b. Kaynak programı açıklamalarıyla birlikte yazınız.



# Kesme ve MCS-51

## Kesmeleri



### Giriş

Kesme, program akışının donanım tarafından üretilen veya dışarıdan uygulanan işaretlerle değiştirilmesidir. Kesme girişleri her komut işleme saykılının başında denetlenen asenkron girişleridir. Kesme girişlerinin sayısı, algılanmaları ve çalışma şekilleri mikroişlemcilerde göre farklılık gösterir. Kesme zamana bağlı denetleme işlemlerinde, mikroişlemci dışında rasgele gelişen olayları denetlemek amacıyla kullanılır. Kesme kullanarak tasarlanan sistemde birden fazla denetim işlemi koşut olarak gerçekleştirilebilirken aynı zamanda sistem performansı da yükseltilebilir. 8051'de 3 adet iç ve 2 adet dış kesme kaynağı bulunur. İç kesme kaynakları seri iletişim arabirimi, zamanlayıcı 0 ve zamanlayıcı 1'e aittir. Dış kesme 0 ve dış kesme 1

kaynaklarına 8051'e bağlanan aygıtların kesme çıkışları bağlanabilir. Reset işlemi de kesmenin tanımına uyar fakat tekrar kaldığı noktaya kesme sonrası dönmemesi reseti, diğer kemelerden ayırır. Kesme kaynaklarının izinlenmesi ve sıralanması için iki adet yazac vardır. Kesme kaynaklarından MİB'e gelen kesmelerin algılanmasını denetleyen kesme izinleme (interrupt enable) yazacı ile istenen kesme kaynağının izinlenmesi yapılırken kesme önceliklime yazacı izinlenen kesme kaynaklarının aynı anda kesme istemeleri durumunda algılanma sıralarını belirler. Algılama işlemi önceliği yüksek olan kaynaktan başlanarak yapılır.

İzinlenen kesme kaynağı girişine kesme isteği geldiğinde MİB isteğin geldiği anda işletilmekte olan komutu tamamladıktan sonra kesme girişlerini denetler, istek hala devam ediyorsa kesme algılanır. Algılama sonrası MİB öncelikle bir sonraki komutun adresini yığında yedekler daha sonra kesme kaynağının numarasını belirler ve numaraya bağlı olan adresi program sayacına yükler. Bu adrese kesme vektörü adı verilir. 8051 kesme kaynaklarının kesme vektörleri Çizelge-7.1'de gösterilmiştir. Kesme vektörleri birbirine yakın olduklarından eğer kesme altprogramı bu alana sığmayacak kadar uzun olduğunda bağlanma komutu ile başka alana bağlanılır. Kesme alt programının sonuna ana programa dönüş için RETI komutu yerleştirilir. Bu komut işletildiğinde MİB yığının en üstünde yer alan dönüş adresini PC'ye yerleştirir ve ana program kaldığı komuttan başlanarak işletilmeye devam eder. Şekil-7.1'de bir sistemin program işleyişi gösterilmiştir. İşlemci (base-level) ana düzey de ve kesme düzeyinde (interrupt-level) olmak üzere iki düzeyde program işletir, zaman her iki program işletilirken ilerleyecektir.

Kesme kaynağı	Kesme vektörü
Zamanlayıcı 2	002B
Seri kanal	0023
Zamanlayıcı 1	001B
Dış kesme 1	0013
Zamanlayıcı 0	000B
Dış kesme 0	0003
Reset	0000

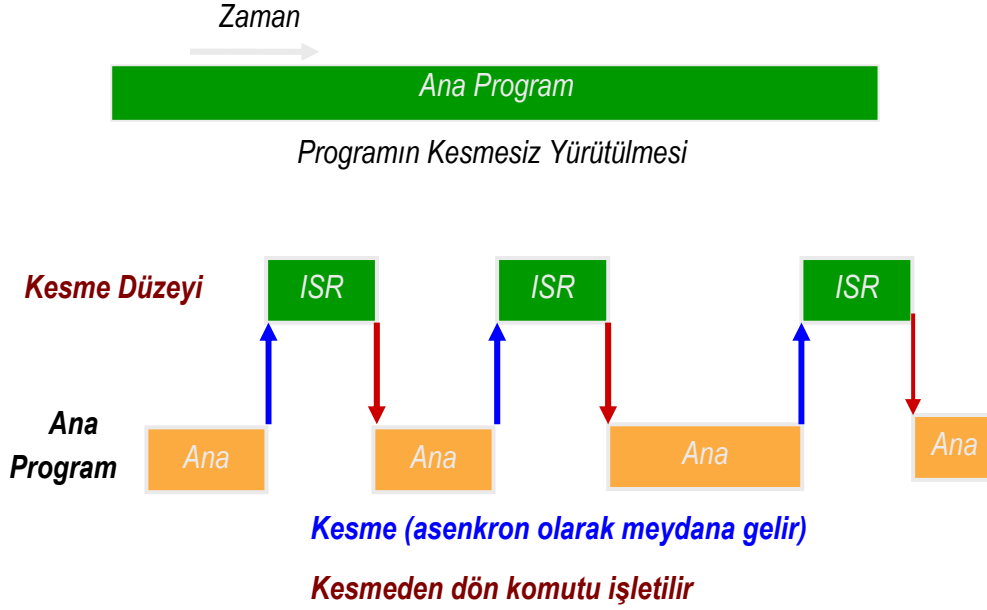
Çizelge - 7.1 Kesme kaynakları ve kesme vektörleri.

Örnek olarak otomatik kumandalı klimayı ele alalım mikrodenetleyicinin asıl görevi klimanın istenilen değerde oda sıcaklığını sabit tutmasıdır. Yeni bir değer girmek istediğimizde kumanda üzerinde bir tuşa basıldığında kesme isteyecektir. Kesme kabul edildiğinde işlemci kumanda işini kısa süreli bırakacak bir anlamda ikincil görevi olan kumandadan veri okuyacaktır. Okuma işinden sonra asıl görevi olan denetleme işlemine devam eder.

Yazıcı, modem, keyboard ve benzeri birçok çevre aygıtları bir mikroişlemcinin etkin çalışma zamanının çok küçük bir bölümüne gerek duymaktadır. Örneğin; bir yazıcının bir tek karakter yazması için gereken sürede, MİB binlerce komut yürütülebilir. Eğer 8051 bu yazıcıyı doğrudan denetlerse, yazılacak her karakter için bir öncekinin yazılmasını bekleyecek ve zamanı boşa geçirecektir. Eğer, yazıcı 8051'e bir dış kesme ucu üzerinde bağlanırsa, yazıcı bir sonraki karakter için hazır oluncaya dek, diğer işleri için zaman ayırabilecektir. 8051'e kesme geldiğinde ilgili kesme programını yürütmeye geçecek ve bir karakter daha gönderdikten sonra diğer işlerini yürütmeyi sürdürmek için geldiği yere geri dönecektir. Böylece yazıcı, işlemci tümüyle kendisine ayrılmışçasına hizmet görürken, eşzamanlı diğer işlerde yapılacaktır.

8051 tümdevre üzerinde yer alan seri kanal ve zamanlayıcılar işlemlerini tamamladıklarını MİB'e kesme bayraklarını kurarak haber verirler. Seri kanal alıcı tamponu dolduğunda veya verici tamponu boşaldığında kesme bayrağını kurar. Eğer kesme kaynağı izinlenmiş ise bu istek algılanır. Seri kanal veri alırken veya gönderirken aynı zamanda MİB diğer işlemleri yapabilir. Aynı durum zamanlayıcılar için de geçerlidir.

Şekil-7.1'de kesme kullanılmayan ve kesme kullanılan programların zaman düzleminde nasıl yürütüldükleri gösterilmiştir. Kesme kullanmayan programlarda sadece ana program vardır ve kesintisiz program yürütülür. Kesme kullanılan programlarda işlemci ana programı yürütürken gelen kesme isteği ile kesme servis altprogramına (ISR) bağlanır ve bu altprogram RETI komutu yürütülene kadar devam eder. RETI komutu yürütüldüğünde tekrar kaldığı yerden ana program yürütülür. Kesme istekleri asenkron gelişen olay olduğu için zamanı ve adedi belirli değildir.



Şekil–7.1 Kesme olduğunda ve olmadı ğında programın işleyişi.

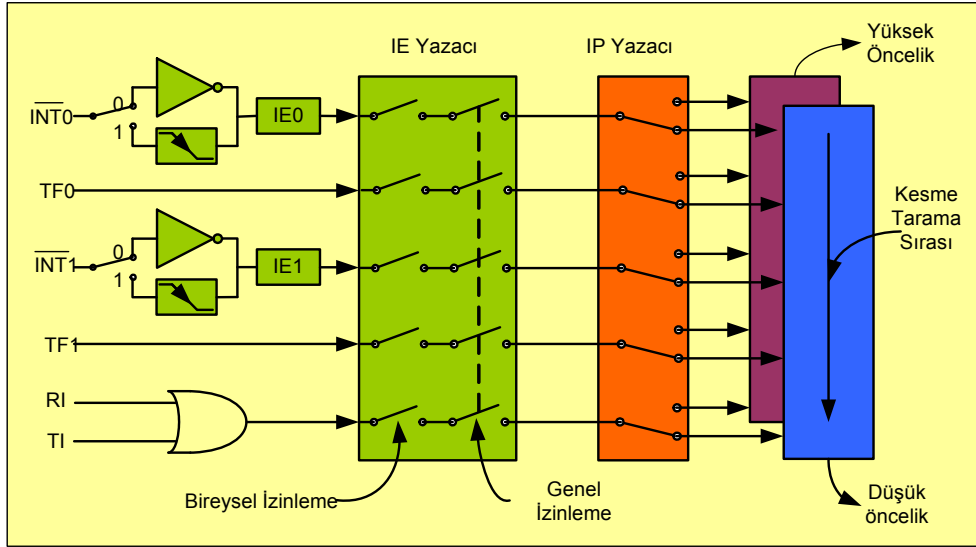
## 8051'in Kesme Yapısı

İki dış, bir seri port, iki zamanlayıcı olmak üzere beş adet kesme kaynağı olan 8051'de, sistem reseti ile tüm kesmeler algılanmaz durumdadır. Programcı kullanacağı kesme kaynaklarını kesme izinleme yazacını kullanarak yazılım ile izinlenmelidir. Beş kaynaktan herhangi iki veya daha fazlası aynı anda kesme isteğinde bulunabilir, işlemci ise sadece birine yanıt verebilir. Bu durumda kesmenin kaynağının belirlenebilmesi için tarama (polling) ve öncelik sıralaması gerekir. Tarama işlemi sabittir ve değiştirilemez, fakat öncelik düzeyi program ile değiştirilebilir.

Her kesme kaynağı bit adreslenebilir IEN yazacının ilgili biti kurularak izinlenebilir veya temizlenerek engellenebilir. Bu yazaçta yer alan IEN biti temizlenerek tüm kesmeler birlikte engellenebilir. Şekil–7.3'te IEN yazacının içeriği gösterilmiştir. Kesme kaynaklarının öncelik sırası IP (interrupt priority) yazacı kullanılarak belirlenebilir. Kesme kaynaklarını iki öncelik kümesine ayırabiliriz. Eğer tüm kesme kaynakları aynı kümeye ayrılırsa doğal sıralamaya göre öncelenir. Aynı kümede yer alan kesme kaynaklarının öncelenmesi doğal sıralamaya göre yapılır. Doğal sıralamada IP yazacının en düşük değerli biti yani dış kesme 0 birinci önceliğe sahiptir, ikinci ve diğer sıralama ise sırasıyla yüksek değerli bite doğru devam eder.

## Mikrodenetleyiciler 8051 Uygulamaları





Şekil-7.2 8051'in kesme yapısı.

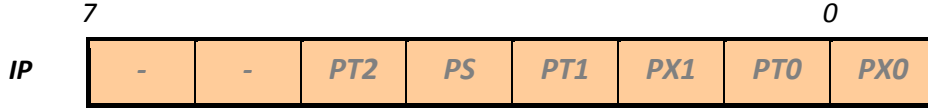


BİT	SİMGE	TANIM
IEN.7	EA	Genel kesme izin biti.
IEN.6	-	Kullanılmamış.
IEN.5	ET2	Zamanlayıcı 2 kesmesi izinleme biti.
IEN.4	ES	Seri port kesme izinleme biti.
IEN.3	ET1	Zamanlayıcı 1 kesmesi izinleme biti.
IEN.2	EX1	Dış kesme 1 izinleme biti.
IEN.1	ET0	Zamanlayıcı 0 kesmesi izinleme biti.
IEN.0	EX0	Dış kesme 0 izinleme biti.

Şekil-7.3 Kesme izinleme yazacı IEN'in içeriği.

Son sırada 8051'de seri kanal kesmesi, 8052'de ise zamanlayıcı 2 kesmesi yer alır. İzinlenmeyen kesmeler doğal sıralamanın dışında yer alır. Kesme isteği kabul edilen kesme kaynağının vektörüne bağlanmadan önce kendinden daha az önceliğe sahip kesme kaynaklarının kesme isteklerini IEN yazacındaki ilgili bitlerini sıfırlayarak

engeller. Kesme alt programından dönüşte RETI komutu işletildikten sonra dönüş adresini yığından PC'ye yüklemeyen önce IEN yazacını eski değerine getirir.



BİT	SİMGE	TANIM
IP.7	-	Kullanılmamış.
IP.6	-	Kullanılmamış.
IP.5	PT2	Zamanlayıcı 2 kesmesi sıralama biti.
IP.4	PS	Seri port kesme sıralama biti.
IP.3	PT1	Zamanlayıcı 1 kesmesi sıralama biti.
IP.2	PX1	Dış kesme 1 sıralama biti.
IP.1	PT0	Zamanlayıcı 0 kesmesi sıralama biti.
IP.0	PX0	Dış kesme 0 sıralama biti.

Şekil-7.4 IP yazacının içeriği.

Şekil-7.4'te 8051'in kesme devresinin mantık şeması gösterilmiştir. Kaynaktan gelen kesmenin MİB'e iletilebilmesi için öncelikle bu kesme kaynağının kendi izin anahtarının kapalı olması ve sonra genel izin anahtarlarının kapalı olması gerekir. Bu iki izin aşamasını geçen kesme IP yazacında belirlenen seçime göre yüksek öncelik veya düşük öncelik kodlayıcı girişine iletilir. Eğer kendinden daha büyük bir kesme isteği yok ise kesme isteği MİB'e iletilir. 8051'deki kesme düzeneği, gelen bir kesme isteğine eğer kesme izni olan bir kaynaktan geliyorsa, kesmeyi algılar. Program sayacının içeriğini yığında korur ve yerine ilgili kesme vektörünü yerleştirerek bu vektöre bir LCALL oluşturur. Bu işlem bazı özel durumlarda hemen gerçekleştirilmez bu özel durumları aşağıdaki gibi özetleyebiliriz.

- Kesme isteği geldiğinde eş yada yüksek öncelikli bir başka kesmeye ilişkin istek karşılanıyorsa yeni gelen kesme bekletilir.
- Kesme isteğinin sezildiği makine çevrimi yürütülen komutun en son çevrimi değilse son çevrime kadar kesme isteği bekletilir.

- 
- The diagram illustrates the timing of the 8086 microprocessor during an interrupt sequence. It shows the relationship between the ALE (Address Latch Enable) signal, the INT (Interrupt) signal, and the execution of specific instructions.
- Top Section (KESME ALTPROGRAMI):**
- ALE:** A series of pulses indicating address latching.
  - INT:** An active-low signal that transitions from high to low, initiating the interrupt sequence.
  - Execution:** The KESME instruction is executed, consisting of a 12-cycle period (SONRAKİ KOMUT) and a 24-cycle period (DANANIMSAL "CALL").
  - Timing:** A 38-hour period (38 SAAT PERYODU) follows the execution of the KESME instruction.
- Bottom Section (MUL VEYA DIV KOMUTU):**
- Execution:** The MUL VEYA DIV instruction is executed, consisting of a 12-cycle period (SONRAKİ KOMUT) and a 48-cycle period (MUL VEYA DIV KOMUTU).
  - Timing:** A 86-hour period (KÖTÜMSER DURUM 86 SAAT PERYODU) follows the execution of the MUL VEYA DIV instruction.

*Kesme düzeneği, her makine saykılının bitmesinden iki osilatör saykılı önce, kesme kaynaklarını denetler. Eğer bir kesme isteği belirlenmişse, bunun işleme konması, en erken bir sonraki makine saykılı sonunda gerçekleşecektir. Bunun için kesme isteğinin algılandığı andan sonra gelen makine saykılı için 12 ve en sonunda uygulanacak LCALL için 24 osilatör periyodu harcanır. En iyimser olarak 38 osilatör salınımı sonra (12MHZ'de  $3.17\mu s$ ) kesme işleme konmuş olur. Şekil-7.5'te bu duruma örnek zamanlama diyagramı gösterilmiştir. Kesme isteğinin algılanması gereken an geçilmiş ise sonraki komutta algılama gerçekleşir. Algılandıktan sonraki komut eğer bir çarpma veya bölme gibi işlenmesi için 4 makine saykılı gerektiren bir komut ise kesmeye yanıt verilmesi için geçen süre 86 saat saykılı olacaktır. (12Mhz'de  $7.17\mu s$ )*

## Dış Kesme Girişlerinin Denetimi

Dış kesme girişlerinin yapısını denetleyen bitler TCON yazacında yer almaktadır. Çizelge-7.2'de TCON yazacının içeriği ve görevleri belirtilmiştir. IT0 ve IT1 bitleri kurulduğunda INT0 ve INT1 girişlerinde oluşan düşen kenarları kesme olarak algılayarak IE0, IE1 bayraklarını kurar. Bu bitler temizlenirse dış kesme girişlerine örnekleme süresince düşük seviye uygulandığında kesme olarak algılanır.

BİT	ADI	ADRES	AÇIKLAMA
TCON.7	TF1	8FH	T1 taşma bayrağı, taşma olduğunda kurulur, yazılımla veya kesme algılandığında temizlenir.
TCON.6	TR1	8EH	T1 çalıştırma/durdurma biti yazılım ile içeriği değişir.
TCON.5	TF0	8DH	T0 taşma bayrağı. taşma olduğunda kurulur, yazılımla veya kesme algılandığında temizlenir.
TCON.4	TR0	8CH	T0 çalıştırma/durdurma biti. yazılım ile içeriği değişir.
TCON.3	IE1	8BH	Dış kesme 1 kenar bayrağı. INT1 girişinde düşen kenar geldiğinde kurulur; yazılım ile veya MİB kesme vektörüne bağlandığında donanım tarafından temizlenir.
TCON.2	IT1	8AH	Dış kesme 1 kenar bayrağı. Yazılım ile kurulup temizlenir. Kurulduğunda INT1'de düşen kenarda IE1 bayrağı kurulur; temizlendiğinde düşük seviyede kesme algılanır ve kesme bayrağı kurulmaz.
TCON.1	IE0	89H	Dış kesme 0 kenar bayrağı. INT0 girişinde düşen kenar geldiğinde kurulur; yazılım ile veya MİB kesme vektörüne bağlandığında donanım tarafından temizlenir.
TCON.0	IT0	88H	Dış kesme 0 tip seçme bayrağı. Yazılım ile kurulup temizlenir. Kurulduğunda INT0'da düşen kenarda IE0 bayrağı kurulur; temizlendiğinde düşük seviyede kesme algılanır ve kesme bayrağı kurulmaz.

Çizelge-7.2 TCON yazacının bitleri ve görevleri.

## Kesme Servis Altprogramı

Kesme algılandıktan sonra dönüş adresi yığında donanımcı saklanır ve program sayacına gelen kesme isteğinin numarasına göre vektör adresi yüklenir. Kesme

## Mikrodenetleyiciler 8051 Uygulamaları

vektöründe altprogram için ayrılan yer 8 bayttır. Bu alan bir çok kesme altprogramı için yeterli değildir. Bu gibi durumlarda bağlanma komutu ile belleğin başka alanına bağlanılır. Kesme servis altprogramının çalışma şekli bazı farklılıkları olsa da altprograma benzerdir. Özellikle altprograma girmek ile kesme altprogramına girmek arasında farklılık vardır. Altprograma girmek için CALL komutu adresi ile birlikte işletilir, komutun ne zaman işletileceği önceden belirlidir. Kesme alt programına girmek için ise komut işletilmez ve çağrılacağı zaman belirli değildir. Donanım işareti kesme girişlerinin birine gelir, algılanan kesme girişine bağlı olan adres (kesme vektörü) donanım tarafından program sayacına yüklenerek kesme servis altprogramı çağrılır. Her iki altprogram çağrılmadan önce dönüş adresi donanımcı yığında yedeklenir. Kesme altprogramında ayrıca kendisinden daha düşük önceliğe sahip kesme isteklerinin kesme altprogramın kesmemesi için bu kaynaklara ait izin bitleri donanım tarafından sıfırlanır. Dış bir olaydan gelen bir kesme isteğinin, yürütülmekte olan programın neresinde geleceği önceden kestirilmediğinden, 'asenكرون' olarak nitelenmektedir. Alt program ise programın bilinen bir yerinde çağırıldığından 'senكرون' olarak nitelenir.

Kesme hizmet programından ayrılmak alt programından ayrılmaya benzemektedir. Her ikisinde de tek Baytlık bir komutla geri dönüş adresi yığından geri alınıp program sayacına yüklenmekte ve önceki işlemler kaldıkları yerden sürdürülmektedir. Kesme hizmet programından ayrılmanın farkı geri dönüş komutu RETI komutunun RET komutunun yaptıklarına ek olarak kendisinden daha düşük önceliğe sahip kesmelerin tekrar kesme geldiği andaki durumuna getirmesidir.

### AT89S52'nin Kesme Kaynakları

AT89C52 ve AT89S52 mikrodnetleyicisinde 8051'den farklı olarak sadece Zamanlayıcı 2 kesmesi kesme kaynaklarına eklenmiştir. Zamanlayıcı 2 kesmesi en düşük önceliğe sahiptir, istendiğinde diğerleri gibi IP yazacından önceliği düzenlenebilir. Zamanlayıcı 2 kesmesi IEN yazacının 5 nolu biti kurularak izinlenebilir.

### Aylak Ve Kısık Güçte Çalışma

PCON yazacı içerisinde yer alan PD ve IDL bitleri kullanılarak mikrodnetleyici işi bittikten sonra kendi kendine kapattırılabilir veya daha sonra uyandırılmak üzere

uyutulabilir. Eğer mikroişlemci kendini kapatırsa bu çalışma şekline kısık güçte çalışma, uyutulursa aylak çalışma kipleri olarak adlandırılırlar.

### Aylak Çalışma Kipi

Yazılım ile IDL bitinin kurulması ile mikrodnetleyici uyutulur uyku sırasında tüm çevre birimleri çalışmalarını sürdürürler. SFR yazaçlarının içerikleri ve iç RAM belleğin içerikleri korunur. Portlara yazılmış olan veriler korunur. Uykudan ancak izinlenmiş bir kesme veya reset ile çıkılabilir. Reset ile çıkıldığında mikrodnetleyici kendisinin reset gelmeden önceki 2 makine saykılı geride varsayarak program işletmeye çalışır. Bu durumda iç RAM bölgesine ulaşma yasaklanmıştır. Fakat portlara ulaşmak mümkündür. Fakat hatalı değişimlere sebebiyet vermemek için portlara ve dış belleğe yazma engellenmiştir.

### Kısık Güçte Çalışma Kipi

PD biti temizlendiğinde kısık güçte çalışma başlamış olur. Osilatör durdurulur, kısık güce geçme komutu işletilen son komut olur. Kısık güçte çalışma yapılırken iç RAM ve portların içerikleri korunur. Bu çalışma kipinden çıkmanın tek yolu RESET'tir. Reset sonrası SFR'lerin içerikleri yeniden belirlenir, fakat iç RAM belleğin içeriği değişmez. Çizelge–7.3'de aylak ve kısık güçte denetim işaretlerinin seviyeleri ve portların durumları gösterilmiştir.

Kip	Prg. Bel.	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
Aylak	İç	1	1	Veri	Veri	Veri	Veri
Aylak	Dış	1	1	Yüzer halde	Veri	Adres	Veri
Kısık güç	İç	0	0	Veri	Veri	Veri	Veri
Kısık güç	Dış	0	0	Yüzer halde	Veri	Veri	Veri

Çizelge–7.3 Aylak ve kısık güçte denetim işaretlerinin seviyeleri ve portların durumları.

**Örnek 7.1**

Zamanlayıcı 0'ı kullanarak P1'e bağlı LED göstergedeki sayıyı saniyede bir arttıran programı yazın.

**Çözüm:**

Zamanlayıcı 0 kip 1'de çalıştırıldığında 65536 makine saykılı bir zaman dilimi gecikme elde edilir. 1 saniyelik gecikmenin elde edilebilmesi için yazılım ile kesme adedi sayılmalıdır. Bu program aynı zamanda A ve PSW yazaçlarının içeriklerini korur. Program çalıştırıldığında P1'in içeriği saniyede bir artacaktır.

Org 0h

Ajmp anaprg

Org 0bh

Push Acc ;A'yı yedekle

Push PSW ;Durum yazacını yedekle

Djnz R0,son ;1 sn olmadı ise p1'i değiştirmeden çık

Inc P1 ;arttır

Mov R0, #16 ;bir saniye sayacını tekrar kur

son:

Pop PSW ;durum yazacının içeriği yedekten al

Pop Acc ;A'nın içeriği yedekten al

Reti

anaprg:

Mov R0,#16 ;her 16 kesme bir saniye eder

Mov TMOD,#01h ;zamanlayıcı 0'ı kip 1'e ayarla

Setb TR0 ;zamanlayıcı 0'ı çalıştır

Mov IE,#10000010b;Zamanlayıcı 0'ın kesmesini izinle

Mov P1,#0 ;P1'i sıfırla

Sjmp \$

**Örnek 7.2**

*Kesme ve zamanlayıcı kullanarak buton zıplamasından kaynaklanan gürültülü işaretin önlenyen programı yazın.*

**Çözüm:**

*Mikroişlemcinin işlem hızı insanın hareket hızına göre çok hızlı olduğu için butona bir adet basılması mikroişlemci tarafından birden fazla basılmış gibi algılanır. Bu sakınca kesme ve zamanlayıcı birlikte kullanılarak ortadan kaldırılabilir. P3.2'ye bir buton ve P1.0'a bir LED bağlanmıştır, butona birinci defa basıldığında LED yanacak ikincide ise sönecektir.*

```

Org 0h
    Ajmp ana_prg
Org 03h
    Mov TH0,#0C0h      ;20 mili saniye beklet
    Mov TL0,#00h
    Setb IE.1          ;zamanlayıcı kesmesini izinle
    Setb TR0
    Reti
Org 0bh
    Clr TR0
    Clr TF0            ;Zaman aşımını sıfırla
    Clr IE.1           ;sadece buton kesmesini algıla
    Cpl P1.0           ;LED yanık ise söndür, sönük ise yak
    Reti
Org 020h
ana_prg:
    Mov TMOD,#01h      ;zamanlayıcı 0'ı kip 1'e ayarla
    Mov TCON,#00000001b ;Kesmeyi düşen kenarda algıla
    Mov IE,#10000001b   ;Dış kesme 0'ı izinle
    Sjmp $

```



### Örnek 7.3

P1.7'den 10 KHz kare dalga üreten programı yazın.

#### Çözüm:

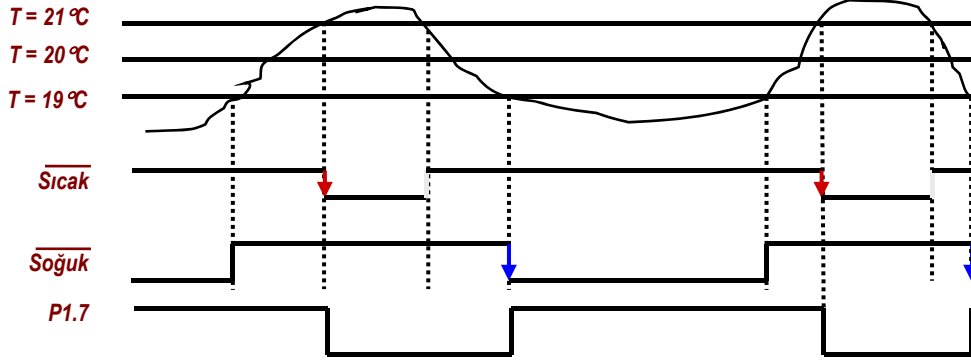
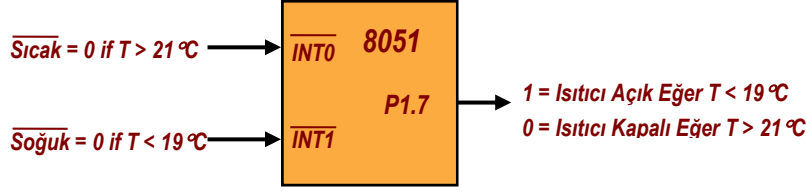
12 Mhz'lik kiristal kullanıldığı kabul edilirse, makine saykılı 1 µsaniye

KHZ\_10:

```
org 0
    Ajmp 10khz_kare    ;Ana programa bağlan
org 0bh
    clr tf0            ;Zaman aşımını temizle
    cpl p1.7           ;hattı tersle
    reti
org 020h
khz_kare:
    mov tmod,#00000010b    ;T0'ı kip 2'ye ayarla
    mov th0,#-50          ;50 makine saykılı beklet
    setb tr0             ;Zamanlayıcıyı çalıştır
    mov ie,#10000010b;Zamanlayıcı kesmesini izinle
    sjmp $               ;Zaman aşımını bekle
```

## Örnek 7.4

Oda sıcaklığını 19 ile 21°C aralığında sabit tutan programı yazın.

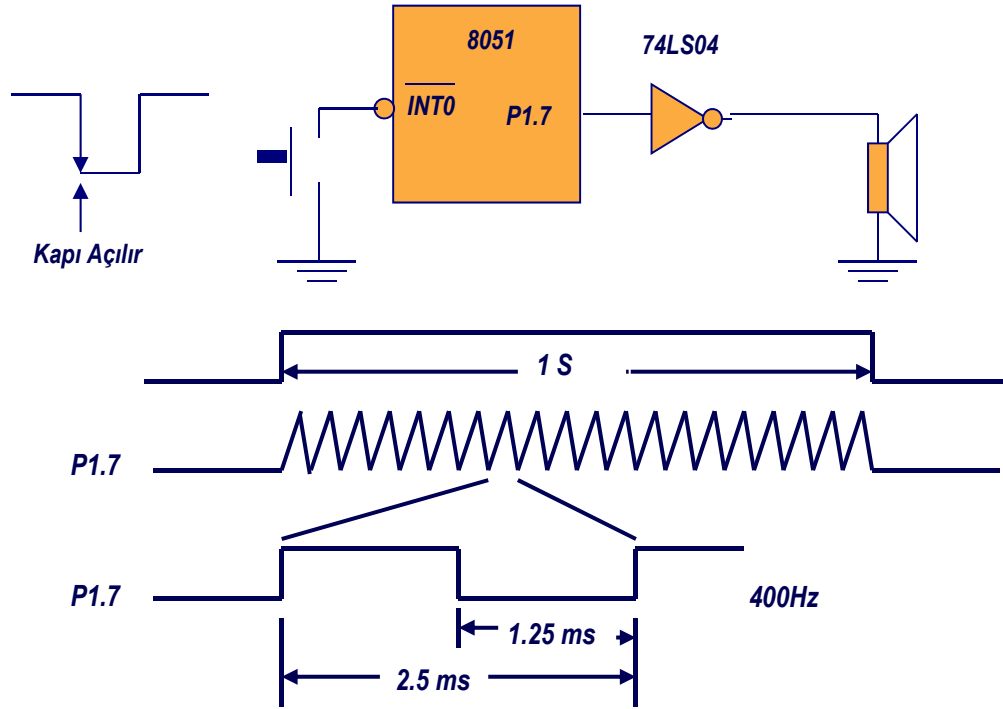


Çözüm:

```
ORG 0
    LJMP ana
EX0ISR:
    CLR P1.7          ;ısıtıcıyı kapa
    RETI
ORG 13H
EX1ISR:
    SETB P1.7         ;Isıtıcıyı aç
    RETI
ORG 0030H
ana: MOV IE, #85H     ;kesmeyi izinle
    SETBIT0           ;düşen kenarı kesme algıla
    SETBIT1
    SETB P1.7         ;ısıtıcıyı aç
    JB P3.2, son2     ;eğer T>21?
    CLR P1.7          ;Evet, ısıtıcıyı kapat
son2:
    SJMP $            ;bekle
```

## Örnek 7.5

Kapı açıldığında 1 saniye sıra ile alarm çıkışı veren programı yazın.



```

ORG 0
    LJMP ANA
    LJMP EX0ISR

ORG 0BH
    LJMP T0ISR           ;T0 ve R7=20 1 S için kullan.

ORG 1BH
    LJMP T1ISR           ;T1'i ses alarmı için kullan.

ANA:    SETB IT0          ;Düşen kenarı kesme algıla
        MOV TMOD,#11H    ;16-bit zamanlayıcı
        MOV IE,#81H      ;EX0 izinle

L1:     SJMP $            ;kesme gelene kadar bekle

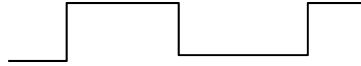
EX0ISR: MOV R7,#20        ;20x50000 ms = 1 s
        SETB TF0         ;T0 ISR
        SETB TF1         ;T1 ISR
        SETB ET0         ;T0 kesmesini izinle
        SETB ET1         ;T1 kesmesini izinle

```

```
                RETI
T0ISR:  CLR  TR0                ;T0'ı durdur
        DJNZ R7,L2            ;20 olmadı ise zamanlayıcı çalışsın
        CLR  ET0                ;oldu ise zamanlayıcı dursun
        CLR  ET1                ;ses dursun
        LJMP EXIT
L2:      MOV  TH0,#0ECh          ;0.05s gecikme için değer
        MOV  TL0,#77h
        SETB TR0                ;T0 tekrar çalıştır
EXIT:    RETI
T1ISR:  CLR  TR1                ;T1'i durdur
        MOV  TH1,#0FBh          ;400 Hz için devam et
        MOV  TL1,#1Dh           ;ses
        CPL  P1.7
        SETB TR1                ;zamanlayıcıyı başlat
        RETI
```

## Sorular

1. 8052'in kesme yapısını ve kesme ile ilgili yazaçların görevlerini açıklayın.
2. 32. 8051'i kullanarak aşağıdaki TTL işaretin yüksekte kaldığı süreyi ölçen ve sonucu DPTR yazacına yazan programı yazın.
  - a. Bağlantı şemasını çizin.
  - b. Algoritmasını yazın.
  - c. Kaynak programı yazın



3. 33. 8051 kullanarak, dış kesme 0 geldiğinde P1.0 den 7 kHz ve dış kesme1 geldiğinde P1.1 den 500 Hz frekanslarında kare dalgaları zamanlayıcı kesmelerini kullanarak üreten programı yazın.
  - a. Bağlantı şemasını çizin.
  - b. Algoritmasını yazın.
  - c. Kaynak programı yazın
4. 8051 kullanarak, dış kesme 0 geldiğinde P2.0 dan 5 kHz ve dış kesme1 geldiğinde P2.1 den 400 Hz frekanslarında kare dalgaları zamanlayıcı kesmelerini kullanarak üreten programı yazın.
  - a. Bağlantı şemasını çizin.
  - b. Algoritmasını yazın.
  - c. Kaynak programı yazın



# MCS-51 Komut Kümesi

## KISALTMALAR

( )	.... içeriği.
(( ))	.... tarafından işaret edilen veri.
rrr	sekiz yazaçtan biri 000=R0, 001=R1, gibi.
dddddddd	veri bitleri.
aaaaaaaa	adres bitleri.
bbbbbbbb	bit adres.
i	dolaylı adreslemede yazaç numarasını belirtir i=1 veya i=0 olabilir.
eeeeeeee	8 bit bağıl adres (OFF-SET).

### ACALL adr 11

**İşlev:** Mutlak altprogram çağırma komutu.

**Açıklama:** ACALL komutu koşul aramaksızın belirtilen adresteki alt programı çağırır. Komut işletildiğinde PC içeriği bir sonraki komutun adresini belirlemek için iki artırır ve sonra 16 bit adres önce DDB olmak üzere yığına atılır. Hedef adres artırılan PC'nin yüksek değerli 5 biti opkodun 7-5 nolu bitleri ve komutun ikinci baytında yer alan 8 bit adres birleştirilerek elde edilir. Hedef adresteki komutun bulunduğu 2 K'lık blok içerisinde yer alan alt program çağırma işi gerçekleşmiş olur. Alt program program belleğinde olmalıdır. Bu komut işletildiğinde bayrakları etkilemez.

Bayt 2

İşlem süresi: 2

Kodu: aaa10001 aaaaaaaa

Not:aaa=A10-A8 ve aaaaaaaa=A7-A0.

İşlem: (PC)=(PC)+2

(SP)←(SP)+1

((SP)) ←(PC7-PC0)

(SP) ←(SP)+1

((SP)) ←(PC15-PC8)

(PC10-PC0) ←sayfa adresi

### ADD A <kaynak-bayt>

İşlev: Toplama.

Açıklama: ADD komutu belirtilen adresteki bayt ile akümülatörü toplar, sonucu akümülatöre yazar. Elde bayrağı bit 7'den elde olursa kurulur, oluşmadığında temizlenir. Yardımcı elde bayrağı bit 3'ün toplamından elde olursa kurulur, oluşmadığında temizlenir. İşaretsiz sayıların toplamında elde bayrağı taşmayı gösterir. Taşma bayrağı OV bit 6'dan elde olursa ve bit 7'den oluşmaz ise veya bit 7'den elde oluşur fakat bit 6'dan elde oluşmaz ise kurulur. Aksi hallerde temizlenir. Negatif sayıların toplamında OV bayrağı iki negatif sayının toplamından pozitif sayı elde edildiğini veya iki pozitif sayının toplamından negatif sayının elde edildiğini gösterir. Kaynak bayt için dört adresleme kipi kullanılır.

### ADD A, Rn

Bayt: 1

İşlem süresi: 1

Opkodu: 00101rrr

İşlem: (A)←(A)+(Rn)

### ADD A, doğrudan adres

Bayt: 2

İşlem süresi: 1

Opkodu: 00100101 aaaaaaaa

İşlem: (A)←(A)+(doğrudan adres)

### ADD A, @Ri

Bayt: 1



*İşlem süresi: 1*

*Opkodu: 0010011i*

*İşlem:  $(A) \leftarrow (A) + ((Ri))$*

#### **ADD A, #Veri**

*Bayt: 2*

*İşlem süresi: 1*

*Opkodu: 00100100 dddddddd*

*İşlem:  $(A) \leftarrow (A) + \#Veri$*

#### **ADDC A, <kaynak bayt>**

*İşlev: Elde ile birlikte topla.*

*Açıklama: Belirtilen bayt ile akümülatör ve Elde Bayrağının (CY) içeriklerini toplar, sonucu akümülatöre yazar. Elde bayrağı bit 7'den elde oluşursa kurulur, oluşmadığında temizlenir. Yardımcı elde bayrağı bit 3'ün toplamından elde olursa kurulur, oluşmadığında temizlenir. İşaretsiz sayıların toplamında elde bayrağı taşmayı gösterir. Taşma bayrağı OV bit 6'dan elde oluşursa ve bit 7'den oluşmaz ise veya bit 7'den elde oluşur fakat bit 6'dan elde oluşmaz ise kurulur. Aksi hallerde temizlenir. Negatif sayıların toplamında OV bayrağı iki negatif sayının toplamından pozitif sayı elde edildiğini, veya iki pozitif sayının toplamından negatif sayının elde edildiğini gösterir. Kaynak bayt için dört adresleme kipi kullanılır.*

#### **ADDC A, Rn**

*Bayt: 1*

*İşlem süresi: 1*

*Opkodu: 00110rrr*

*İşlem:  $(A) \leftarrow (A) + (C) + (Rn)$*

#### **ADDC A, doğrudan adres**

*Bayt: 2*

*İşlem süresi: 1*

*Opkodu: 00110101 aaaaaaaaa*

*İşlem:  $(A) \leftarrow (A) + (C) + (\text{doğrudan adres})$*

#### **ADDC A, @Ri**

*Bayt: 1*

*İşlem süresi: 1*

Opkodu: 0011011i  
İşlem:  $(A) \leftarrow (A) + (C) + ((Ri))$

### ADDC A, #Veri

Bayt: 2  
İşlem süresi: 1  
Opkodu: 00110100 dddddddd  
İşlem:  $(A) \leftarrow (A) + (C) + \#Veri$

### AJMP adr 11

İşlev: Mutlak bağlanma komutu.

Açıklama: AJMP komutu koşul aramaksızın belirtilen adrese bağlanır. Komut işletildiğinde PC içeriği bir sonraki komutun adresini belirlemek için iki artırılır. Hedef adres artırılan PC'nin yüksek değerli 5 biti opkodun 7-5 nolu bitleri ve komutun ikinci baytında yer alan 8 bit adres birleştirilerek elde edilir. Hedef adresteki komutun bulunduğu 2 K'lık blok içerisindeki sayfa numarası belirtilmiş adrese bağlanmış olur. Bağlanılacak adres program belleğinde olmalıdır. Bu komut işletildiğinde bayrakları etkilemez.

Bayt 2  
İşlem süresi: 2  
Kodu: aaa00001 aaaaaaaaa  
Not:aaa=A10-A8 ve aaaaaaaaa=A7-A0.  
İşlem:  $(PC) = (PC) + 2$   
 $(PC_{10} - PC_0) \leftarrow \text{sayfa adresi}$

### ANL <HEDEF ADRES>, <KAYNAK ADRES>

İşlev: Bayt uzunluğundaki iki değişkene mantık VEler.

Açıklama: ANL komutu belirtilen kaynak adres ile belirtilen hedef adresi bit bit VE'leyerek sonucu hedef adrese yazar. İşlemden bayraklar etkilenmez. Komutta yer alan iki işlenen altı adres kipinde kullanılır. Hedef akümülatör ise kaynak yazaç, doğrudan, dolaylı veya ivedi adresli olabilir. Bu komut giriş/çıkış portlarının içeriğini değiştirmek için kullanıldığında çıkış pinlerinin içeriği okunmaz sadece çıkış tutucularının içeriği okunur.

### ANL A, Rn

Bayt: 1

İşlem süresi: 1

Opkodu: 01011rrr

İşlem:  $(A) \leftarrow (A) \wedge (Rn)$

#### **ANL A, doğrudan adres**

Bayt: 2

İşlem süresi: 1

Opkodu: 01010101 aaaaaaaa

İşlem:  $(A) \leftarrow (A) \wedge (\text{doğrudan adres})$

#### **ANL A, @Ri**

Bayt: 1

İşlem süresi: 1

Opkodu: 0101011i

İşlem:  $(A) \leftarrow (A) \wedge ((Ri))$

#### **ANL A, #Veri**

Bayt: 2

İşlem süresi: 1

Opkodu: 01010100 dddddddd

İşlem:  $(A) \leftarrow (A) \wedge \#Veri$

#### **ANL doğrudan adres, A**

Bayt: 2

İşlem süresi: 1

Opkodu: 01010010 aaaaaaaa

İşlem:  $(\text{doğrudan adres}) \leftarrow (\text{doğrudan adres}) \wedge (A)$

#### **ANL doğrudan adres, #Veri**

Bayt: 3

İşlem süresi: 2

Opkodu: 01010011i

İşlem:  $(\text{doğrudan adres}) \leftarrow (\text{doğrudan adres}) \wedge \#Veri$

#### **ANL C, <Kaynak Bit>**

İşlev: Elde Bayrağı (CY) ile bit değişkenleri VEler.

**Açıklama:** Kaynak biti 0 ise elde bayrağını temizler, 1 ise Elde Bayrağının (CY) içeriğini değiştirmez. Kaynak bitin önündeki / (slash) işaretinin assembly dilindeki anlamı işlem yapılırken kaynak bitin değili alınarak işlem yapılacak fakat kaynak bit bundan etkilenmeyecektir. Kaynak bit için sadece doğrudan adresleme kullanılır.

#### **ANL C, Bit**

Bayt: 2  
İşlem süresi: 2  
Opkodu: 10000010 bbbbbbbb  
İşlem:  $(C) \leftarrow (C) \wedge (\text{Bit})$

#### **ANL C, /Bit**

Bayt: 2  
İşlem süresi: 2  
Opkodu: 10110000 bbbbbbbb  
İşlem:  $(C) \leftarrow (C) \wedge \text{NOT}(\text{Bit})$

**CALL** (ACALL veya LCALL komutlarına bakınız.)

#### **CJNE <Hedef Bayt>, <kaynak bayt>, kayıklık**

**İşlev:** Kaynak ile hedefi karşılaştırır ve eşit değilse dallanır.

**Açıklama:** Kaynak ile hedefin büyüklüklerini karşılaştırır ve eşit değilse belirtilen bağıl adrese dallanır. Hedef dallanma adresi komut işletildikten sonraki PC'nin içeriğine belirtilen bağıl adres toplanarak elde edilir. Eğer hedef adresin işaretsiz büyüklüğü kaynağın işaretsiz büyüklüğünden daha küçük ise Elde Bayrağı (CY) kurulur. Aksi hallerde temizlenir. İşlemden her iki işlenenin içerikleri etkilenmez.

#### **CJNE A, Doğrudan adres, Bağıl adres**

Bayt: 3  
İşlem süresi: 2  
Opkodu: 10110101 aaaaaaaaa eeeeeeee  
İşlem:  $(PC) \leftarrow (PC) + 3$   
EĞER (A) <> (Doğrudan adres)  
THEN  
 $(PC) \leftarrow (PC) + \text{Bağıl adres}$   
EĞER (A) <> (Doğrudan adres)  
THEN  
 $(C) \leftarrow 1$

ELSE  
(C)←0

### **CJNE A, #Veri, Bağıl adres**

Bayt: 3  
İşlem süresi: 2  
Opkodu: 10110100 dddddddd eeeeeeee  
İşlem: (PC)←(PC)+3  
EĞER (A) <> Veri  
THEN  
(PC)←(PC)+Bağıl adres  
EĞER (A) <> Veri  
THEN  
(C)←1  
ELSE  
(C)←0

### **CJNE Rn, #Veri, Bağıl adres**

Bayt: 3  
İşlem süresi: 2  
Opkodu: 10111rrr dddddddd eeeeeeee  
İşlem: (PC)←(PC)+3  
EĞER (Rn) <> Veri  
THEN  
(PC)←(PC)+Bağıl adres  
EĞER (Rn) <> Veri  
THEN  
(C)←1  
ELSE  
(C)←0

### **CJNE @Ri, #Veri, Bağıl adres**

Bayt: 3  
İşlem süresi: 2  
Opkodu: 1011011i dddddddd eeeeeeee  
İşlem: (PC)←(PC)+3

```

EĞER (Ri) <>Veri
THEN
(PC)←(PC)+Bağıl adres
EĞER (Ri) <> Veri
THEN
(C)←1
ELSE
(C)←0

```

### CLR A

**İşlev:** Akümülatörün içeriğini temizler.

**Açıklama:** Akümülatörün tüm bitleri 0 yapılır. Hiçbir bayrak bu işlemde etkilenmez. Akümülatörün içeriği 67H iken;

CLR A

Komutu işletilir ise akümülatörün içeriği 00h olur.

```

Bayt:          1
İşlem süresi: 1
Opkodu:        11100100
İşlem:         (A)←0

```

### CLR Bit

**İşlev:** Belirtilen biti 0 yapar.

**Açıklama:** Belirtilen bitin içeriği 0 yapılır. İşlemden bayraklar etkilenmez.

### CLR C

```

Bayt:          1
İşlem süresi: 1
Opkodu:        11000011
İşlem:         (C)←0

```

### CLR Bit

```

Bayt:          2
İşlem süresi: 1
Opkodu:        11000010 BBBB BBBB
İşlem:         (Bit)←0

```

### CPL A

**İşlev:** Akümülatörün içeriğini 1'e tümler.

**Açıklama:** Her bit 1'e tümlenir. İşlemden bayraklar etkilenmez.

**Bayt:** 1

**İşlem süresi:** 1

**Opkodu:** 11110100

**İşlem:**  $(A) \leftarrow \text{NOT}(A)$

### CPL Bit

**İşlev:** Belirtilen bitin değerini alır.

**Açıklama:** Belirtilen bit değişkenin içeri mantık DEĞİL işlemine tabi tutulur. Eğer bitin içeriği 1 ise işlem sonrası 0, 0 ise işlem sonrası 1 olur. CPL komutu Elde Bayrağını (CY) değıllemek için kullanılır. Bunun dışındaki bayraklar işlemden etkilenmez. Portlarda kullanıldığında giriş/çıkış bacaklarındaki veriler değıl tutuculardaki veriler esas alınır.

### CPL C

**Bayt:** 1

**İşlem süresi:** 1

**Opkodu:** 10110011

**İşlem:**  $(C) \leftarrow \text{NOT}(C)$

### CPL Bit

**Bayt:** 2

**İşlem süresi:** 1

**Opkodu:** 10110010 bbbbbbbb

**İşlem:**  $(\text{Bit}) \leftarrow \text{NOT}(\text{Bit})$

### DA A

**İşlev:** Toplama sonrası akümülatörün içeriğini İKO'ya ayarlar.

**Açıklama:** İKO olarak kodlanmış iki sayı toplandıktan sonra akümülatörün içeriğini İKO'ya dönüştürür. 8051'de İKO toplama komutu olmaması nedeniyle bu komut kullanılır. ADD ve ADDC komutlarından sonra kullanılabilir. Toplama sonrası akümülatörün düşük değerli nibılı 9'dan büyük ise geçersiz İKO sayı olacaktır. Toplama sonrası DA A komutu işletildiğinde düşük değerli nibıl 9'dan büyük veya Yardımcı Elde Bayrağı (AC) kurulu ise düşük değerli nibıla 6 eklenir. Bu toplama sonunu elde oluşur ise Elde Bayrağı (CY) kurulur, aksi halde Elde Bayrağının (CY) içeriği değışmez. Bu elde bayrağı yüksek değerli tüm bitlere etkiletilir. Eğer düşük

değerli niblın işlemi sonrası Elde Bayrağı (CY) kuruldu ise veya yüksek değerli nibl geçersiz İKO sayı ise 6 eklenir ve geçerli İKO sayı elde edilir toplama sonrası eğer elde oluşur ise Elde Bayrağı (CY) kurulur aksi halde ise Elde Bayrağı (CY) içeriği değiştirilmez. Böylece toplama komutu sonrası eğer elde oluştu ise bu elde birden fazla baytdan oluşan sayıların toplanmasında kullanılır. Bu işlemlerin hepsi bir işlem saykılında gerçekleşir. Aslında yapılan işlem karmaşık değildir. Yapılan işlem İKO dönüşümdür, dönüşüm yapılırken önce akümülatör ve PROGRAM DURUM GÖSTERGE YAZACI içeriği test edilir. Test sonuçlarına göre akümülatöre 00h, 06h, 60h, 66h sayılarından biri toplanır.

Not:: DA A komutu basit olarak akümülatörün içeriğini İKO'ya dönüştürme yapmaz, mutlaka toplama sonrası kullanılmalıdır ve toplanan iki sayıda geçerli İKO olmalıdır. Ayrıca çıkarma işlemi sonrası kullanılamaz.

Bayt: 1  
İşlem süresi: 1  
Opkodu: 11010100  
İşlem: IF [(A3-A0)>9] VEYA [(AC)=1]  
THEN (A3-A0)← (A3-A0)+6  
VE  
IF [(A7-A4)>9] VEYA [(C)=1]  
THEN (A7-A4)← (A7-A4)+6

### DEC BAYT

İşlev: Baytın içeriğini bir azaltır.

Açıklama: Belirtilen baytın içeriğini bir azaltır. Eğer içerik 00h ise azaltıldığında yeni içerik FFh olur. İşlemden bayraklar etkilenmez.

Not: Bu komut çıkış portunun içeriğini azaltmak için kullanıldığında tutucu çıkışlarını okuyarak azaltır. Port bacaklarında oluşan değerlere bakmaz.

### DEC A

Bayt: 1  
İşlem süresi: 1  
Opkodu: 00010100  
İşlem: (A)← (A)-1

### DEC Rn

Bayt: 1



İşlem süresi: 1

Opkodu: 00011rrr

İşlem:  $(Rn) \leftarrow (Rn) - 1$

### DEC doğrudan Adres

Bayt: 1

İşlem süresi: 1

Opkodu: 00010101

İşlem:  $(\text{Doğrudan Adres}) \leftarrow (\text{Doğrudan Adres}) - 1$

### DEC Rn

Bayt: 1

İşlem süresi: 1

Opkodu: 0001011i

İşlem:  $((Rn)) \leftarrow ((Ri)) - 1$

### DIV AB

İşlev: Bölme

Açıklama: Akümülatörün içeriğini B yazacının içeriğine böler. İşlem sonunda bölüm akümülatöre kalan B yazacına yazılırken Elde Bayrağı (CY) ve Taşma Bayrağı (OV) temizlenir. Eğer bölen 00h ise akümülatör ve B yazacının içerikleri değişmez Elde Bayrağı (CY) temizlenirken Taşma Bayrağı (OV) kurulur.

Bayt: 1

İşlem süresi: 4

Opkodu: 10000100

İşlem:  $(A) \leftarrow (A) / (B)$  bölüm  
 $(B) \leftarrow (A) / (B)$  Kalan

### DJNZ <Bayt>, <bağıl adres>

İşlev: İçeriğini bir azalt ve sıfır değilse dallan.

Açıklama: Birinci işlenenin içeriği bir azaltılır eğer sıfır değilse ikinci işlenende belirtilen bağıl adrese dallanır. Eğer azaltma öncesi birinci işlenen baytın içeriği 00h ise azaltma sonrası FFh olur. İşlemden bayraklar etkilenmez. Dallanma adresi ikinci işlenende belirtilen bağıl adres ile sonraki komutun başlangıç adresine arttırıldıktan sonra toplanarak elde edilir. Bağıl adres işaretli sayıdır, eğer 80h-FFh arası sayılar varsa dallanma geriye doğru gerçekleşir. 00h-7Fh arası sayılarda ise dallanma ileri doğru gerçekleşir.

*Not: Bu komut çıkış portunun içeriğini azaltmak için kullanıldığında utucu çıkışlarını okuyarak azaltır. Port bacaklarında oluşan değerlere bakmaz. Bu komut basit bir döngü ile 2 işlem saykılı ile 512 işlem saykılı arasında gecikme yaratır. Başka bir kullanım alanı da sayılı işlerin yapılmasında işlem saymadır.*

### **DJNZ Rn, Bağlı Adres**

Bayt: 2  
İşlem süresi: 2  
Opkodu: 11011rrr eeeeeeee  
İşlem:  $(PC) \leftarrow (PC) + 2$   
 $(Rn) \leftarrow (Rn) - 1$   
IF  $(Rn) = 0$   
THEN  
 $(PC) \leftarrow (PC) + \text{Bayt } 2$

### **DJNZ Doğrudan adres, Bağlı Adres**

Bayt: 2  
İşlem süresi: 2  
Opkodu: 11011rrr eeeeeeee  
İşlem:  $(PC) \leftarrow (PC) + 2$   
 $(\text{Doğrudan Adres}) \leftarrow (\text{Doğrudan Adres}) - 1$   
IF  $(\text{Doğrudan Adres}) = 0$   
THEN  
 $(PC) \leftarrow (PC) + \text{Bayt } 2$

### **INC <Bayt>**

*İşlev: Arttırma.*

*Açıklama: Belirtilen değişkenin içeriğini bir arttırır. Eğer önceki değer OFFh ise arttırma sonucu 00h olur, işlemde bayraklar etkilenmez.*

*Not: Bu komut çıkış portunun içeriğini arttırmak için kullanıldığında tutucu çıkışlarını okuyarak arttırır. Port bacaklarında oluşan değerlere bakmaz.*

### **INC A**

Bayt: 1  
İşlem süresi: 1  
Opkodu: 00000100  
İşlem:  $(A) \leftarrow (A) + 1$

**INC Rn**

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 00001rrr  
 İşlem:  $(Rn) \leftarrow (Rn) + 1$

**INC Doğrudan Adres**

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 00000101 aaaaaaaaa  
 İşlem:  $(\text{Doğrudan Adres}) \leftarrow (\text{Doğrudan Adres}) + 1$

**INC @Rn**

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 0000011i  
 İşlem:  $(Ri) \leftarrow (Ri) + 1$

**JB <Bayt>, bağıl adres**

**İşlev:** Eğer bit 1 ise dallanır.

**Açıklama:** Belirtilen adresteki bit 1 ise dallanma gerçekleşir. Dallanılacak adres komutun üçüncü baytında bağıl olarak verilir. Bağıl adres PC 3 arttırıldıktan sonra PC'ye toplanarak hedef adres elde edilir. İşlemden bayraklar etkilenmez.

Bayt: 3  
 İşlem süresi: 2  
 Opkodu: 00010000 bbbbbbbb eeeeeeee  
 İşlem:  $(PC) \leftarrow (PC) + 3$   
           EĞER (bit)=1 ise  
           Sonra  
            $(PC) \leftarrow (PC) + \text{Bayt\_2}$

**JBC <Bayt>, bağıl adres**

**İşlev:** Eğer bit 1 ise temizler ve dallanır.

**Açıklama:** Belirtilen adresteki bit 1 ise önce bit temizlenir sonra dallanma gerçekleşir. Dallanılacak adres komutun üçüncü baytında bağıl olarak verilir. Bağıl

adres PC 3 arttırıldıktan sonra PC'ye toplanarak hedef adres elde edilir. Bit sıfır ise dallanma olmaz bir alt satırdaki komut işletilir. İşlemden bayraklar etkilenmez.

Bayt: 3  
 İşlem süresi: 2  
 Opkodu: 00010000 bbbbbbbb eeeeeeee  
 İşlem:  $(PC) \leftarrow (PC) + 3$   
           EĞER (bit)=1 ise  
           Sonra  
            $(bit) \leftarrow 0$   
            $(PC) \leftarrow (PC) + \text{Bayt\_2}$

### JC bağıl adres

İşlev: Elde varsa dallanır.  
 Açıklama: Elde bayrağının içeriği 1 ise dallanma gerçekleşir. Dallanılacak adres komutun üçüncü baytında bağıl olarak verilir. Bağıl adres PC 3 arttırıldıktan sonra PC'ye toplanarak hedef adres elde edilir. Bit sıfır ise dallanma olmaz bir alt satırdaki komut işletilir. İşlemden bayraklar etkilenmez.

Bayt: 2  
 İşlem süresi: 2  
 Opkodu: 00010000 eeeeeeee  
 İşlem:  $(PC) \leftarrow (PC) + 2$   
           EĞER (C)=1 ise  
           THEN  
            $(PC) \leftarrow (PC) + \text{Bayt\_2}$

*JMP <Hedef> (SJMP, AJMP, LJMP Komutlarına bakınız).*

### JMP @A+DPTR

İşlev: Dolaylı bağlan.  
 Açıklama: Akümülatörün içeriğini 16 bit DPTR'ye ekler ve elde edilen sayı program sayacına aktararak bu adrese bağlanır. Akümülatörün içeriği işaretli 8 bit sayı olabilir. 16 bit ile 8 bit 16 bit toplama şeklinde yapılır. 7 numaralı bitlerin toplamından oluşan elde 8 numaralı bite aktarılır. Akümülatör PSW ve DPTR yazaçlarının içerikleri işlemden etkilenmez.

Bayt: 1  
 İşlem süresi: 2  
 Opkodu: 01110011  
 İşlem:  $(PC) \leftarrow A + DPTR$

### JNB bit, rel

**İşlev:** Bit Kurulu değilse dallan.

**Açıklama:** Belirtilen bit 0 ise komutun üçüncü baytında verilen işaretli kayıklık bir sonraki komutun adresine arttırılmış PC'ye toplanarak elde edilen adrese dallanır. Test edilen bit ve bayraklar işleminden etkilenmez.

Bayt: 3  
 İşlem süresi: 2  
 Opkodu: 00110000 bbbbbbbb eeeeeeee  
 İşlem:  $(PC) \leftarrow (PC) + 3$   
           EĞER (bit)=0  
           İSE  
            $(PC) \leftarrow (PC) + BAYT\_3$

### JNC rel

**İşlev:** Elde yoksa dallan.

**Açıklama:** Elde bayrağının içeriği 0 ise komutun ikinci baytında verilen işaretli kayıklık bir sonraki komutun adresine arttırılmış PC'ye toplanarak elde edilen adrese dallanır. Test edilen bayrak ve diğer bayraklar işleminden etkilenmez.

Bayt: 2  
 İşlem süresi: 2  
 Opkodu: 01010000 eeeeeeee  
 İşlem:  $(PC) \leftarrow (PC) + 2$   
           EĞER C=0  
           İSE  
            $(PC) \leftarrow (PC) + BAYT\_2$

### JNZ rel

**İşlev:** Akümülatör 00 değilse dallan.

*Açıklama: Akümülatörün içeriği sıfır değilse komutun ikinci baytında verilen işaretli kayıklık bir sonraki komutun adresine arttırılmış PC'ye toplanarak elde edilen adrese dallanır. Test edilen akümülatör ve bayraklar işleminden etkilenmez.*

Bayt: 2  
 İşlem süresi: 2  
 Opkodu: 01110000 eeeeeeee  
 İşlem:  $(PC) \leftarrow (PC) + 2$   
           EĞER (A) < > 0  
           İSE  
            $(PC) \leftarrow (PC) + \text{BAYT\_2}$

### JZ rel

*İşlev: Akümülatör 00 ise dallan.*

*Açıklama: Akümülatörün içeriği sıfır ise komutun ikinci baytında verilen işaretli kayıklık bir sonraki komutun adresine arttırılmış PC'ye toplanarak elde edilen adrese dallanır. Test edilen akümülatör ve bayraklar işleminden etkilenmez.*

Bayt: 2  
 İşlem süresi: 2  
 Opkodu: 01100000 eeeeeeee  
 İşlem:  $(PC) \leftarrow (PC) + 2$   
           EĞER (A) = 0  
           İSE  
            $(PC) \leftarrow (PC) + \text{BAYT\_2}$

### LCALL 16 Bit Adres

*İşlev: 16 bit adresle alt program çağırır.*

*Açıklama: bu komut ile YDB'ı ikinci baytta, DDB'ı üçüncü baytta yer alan alt program çağırılır. Komut işlenirken önce PC üç arttırılarak bir sonraki komutun adresi bulunur. Bu adres yığına atılır, yığın işaretleyici iki arttırılır ve komutun ikinci ve üçüncü baytlarında yer alan 16 bit adres PC'ye yüklenerek bu adreste yer alan alt program çağırılır. ACALL komutunda olduğu gibi 2K kısıtlaması yoktur.*

Bayt: 3  
 İşlem süresi: 2

Opkodu: 00010001 aaaaaaaaa, aaaaaaaaa  
 İşlem:  $(PC) \leftarrow (PC) + 3$   
 $(SP) \leftarrow (SP) + 1$   
 $((SP)) \leftarrow (PC_{7-0}) + 3$   
 $(SP) \leftarrow (SP) + 1$   
 $((SP)) \leftarrow (PC_{15-8}) + 3$

### LJMP 16 Bit Adres

İşlev: 16 bit adresle koşulsuz bağlanır.  
 Açıklama: Bu komut ile YDB'ı ikinci baytta, DDB'ı üçüncü baytta yer alan adrese bağlanır. Komut işlenirken her hangi bir koşul aramaksızın istenilen adrese bağlanır. SJMP ve AJMP komutlarında olduğu gibi kısıtlama yoktur 64 K'lık bellek alanının tamamına dallanma gerçekleşir.

Bayt: 3  
 İşlem süresi: 2  
 Opkodu: 00010001 aaaaaaaaa, aaaaaaaaa  
 İşlem:  $(PC) \leftarrow (PC_{15-0}) + 3$

### MUL AB

İşlev: Bayt değişkenleri çarpar.  
 Açıklama: Bu komut akümülatör ve B yazacında yer alan iki işaretli 8 bitlik sayıyı çarpar sonucu yine akümülatöre (DDB) ve B yazacına (YDB) yazar. Çarpım 255'ten büyük ise OV bayrağı kurulur, aksi halde temizlenir.

Bayt: 1  
 İşlem süresi: 4  
 Opkodu: 10100100  
 İşlem:  $A.B \rightarrow (B)_{15-8}, (A)_{7-0}$

### MOV <Hedef Bayt>, <Kaynak bayt>

İşlev: Bayt değişkeni aktarır.  
 Açıklama: İkinci bayta belirtilen bayt değişken birinci bayt değişkenin üzerine kopyalanır. Kaynak bayt işleminden etkilenmez. Başka yazaç veya bayraklar da işleminden etkilenmez. 8051 deki tüm veri aktarma komutları MOV olarak adlandırılmıştır. Bu yüzden 15 farklı yazımı vardır.

### MOV A, Rn

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 11101rrr  
 İşlem:  $(A) \leftarrow (Rn)$

#### **MOV A, Doğrudan Adres**

Bayt: 2  
 İşlem süresi: 1  
 Opkodu: 11100101 aaaaaaaaa  
 İşlem:  $(A) \leftarrow (\text{Doğrudan Adres})$

#### **MOV A, @Ri**

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 1110011i  
 İşlem:  $(A) \leftarrow ((Ri))$

#### **MOV A, İvedi**

Bayt: 2  
 İşlem süresi: 1  
 Opkodu: 01110100 dddddddd  
 İşlem:  $(A) \leftarrow \#Veri$

#### **MOV Rn, A**

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 11111rrr  
 İşlem:  $(Rn) \leftarrow (A)$

#### **MOV Rn, Doğrudan adres**

Bayt: 2  
 İşlem süresi: 2  
 Opkodu: 10101rrr ddddddd  
 İşlem:  $(Rn) \leftarrow (\text{Doğrudan Adres})$

#### **MOV Rn, #Veri**

Bayt: 2



İşlem süresi: 1

Opkodu: 01111rrr dddddddd

İşlem: (Rn) ← #Veri

### **MOV Doğrudan adres, A**

Bayt: 2

İşlem süresi: 1

Opkodu: 11110101 aaaaaaaa

İşlem: (Doğrudan Adres) ← (A)

### **MOV Doğrudan adres, Rn**

Bayt: 2

İşlem süresi: 2

Opkodu: 10001rrr aaaaaaaa

İşlem: (Doğrudan Adres) ← (Rn)

### **MOV Doğrudan adres, Doğrudan adres**

Bayt: 3

İşlem süresi: 2

Opkodu: 10000101 aaaaaaaa aaaaaaaa

İşlem: (Doğrudan Adres) ← (Doğrudan Adres)

Not: İkinci bayt hedefin adresi, üçüncü bayt ise kaynağın adresini belirtir.

### **MOV Doğrudan adres, @Ri**

Bayt: 2

İşlem süresi: 2

Opkodu: 1000011i aaaaaaaa

İşlem: (Doğrudan Adres) ← ((Ri))

### **MOV Doğrudan adres, #Veri**

Bayt: 3

İşlem süresi: 2

Opkodu: 10000101 aaaaaaaa dddddddd

İşlem: (Doğrudan Adres) ← #Veri

### **MOV @Ri, A**

Bayt: 1

İşlem süresi: 1

Opkodu: 1111011i

İşlem:  $((Ri)) \leftarrow (A)$

### **MOV @Ri, Doğrudan Adres**

Bayt: 2

İşlem süresi: 2

Opkodu: 1010011i aaaaaaaaa

İşlem:  $((Ri)) \leftarrow (\text{Doğrudan Adres})$

### **MOV @Ri, #Veri**

Bayt: 2

İşlem süresi: 1

Opkodu: 0111011i dddddddd

İşlem:  $((Ri)) \leftarrow \#Veri$

### **MOV <Hedef bit>, <Kaynak bit>**

İşlev: Bit değişkeni hedef adrese aktarır.

Açıklama: Üçüncü baytta belirtilen bit değişkeni ikinci baytta belirtilen bit adrese aktarılır. İşlenenlerden birinin mutlaka elde bayrağı olmalıdır. İşlemden hedef bitin dışında bitin içeriği veya bayraklar etkilenmez.

### **MOV C, Bit**

Bayt: 2

İşlem süresi: 1

Opkodu: 10100010 bbbbbbbb

İşlem:  $(C) \leftarrow (bit)$

### **MOV Bit, C**

Bayt: 2

İşlem süresi: 2

Opkodu: 10010010 bbbbbbbb

İşlem:  $(bit) \leftarrow (C)$

### **MOV DPTR, #Veri 16**

İşlev: DPTR'ye 16 bit sabit yükler.

Açıklama: İkinci ve üçüncü baytta verilen 16 bit sayı DPTR'ye yüklenir. İkinci bayttaki sayı DPH'ye üçüncü bayttaki sayı ise DPL'ye yüklenir.

Bayt: 3  
 İşlem süresi: 2  
 Opkodu: 10010000 dddddddd dddddddd  
 İşlem: (DPTR) ← #16 Veri

### **MOVC A, @A+<Temel yazaç>**

**İşlev:** Program belleğinden sabit baytı veya komut baytını akümülatöre yükler.

**Açıklama:** Akümülatörün içerisindeki 8 bit işaretli sayı temel 16 bitlik yazaca toplanarak kaynak adres elde edilir. Kaynak adresteki opkod veya sabit değer akümülatöre yüklenir. Temel yazaç olarak PC veya DPTR kullanılabilir. PC kullanıldığında önce içeriği bir sonraki komutun adresine artırılır ve sonra akümülatörün içeriği ile PC toplanarak kaynak adres elde edilir. DPTR temel yazaç olarak kullanıldığında ise işlemden içeriği etkilenmez. Bayrakların içerikleri işlemden etkilenmez.

### **MOVC A, @A+DPTR**

Bayt: 1  
 İşlem süresi: 2  
 Opkodu: 10010011  
 İşlem: (A) ← ((A)+(DPTR))

### **MOVC A, @A+PC**

Bayt: 1  
 İşlem süresi: 2  
 Opkodu: 10000011  
 İşlem: (PC) ← (PC)+1  
 (A) ← ((A)+(PC))

### **MOVX <hedef baytı>, <kaynak baytı>**

**İşlev:** Dış veri belleğinden veri baytını akümülatöre yükler.

**Açıklama:** Bu komut dış veri belleğinden akümülatöre veya akümülatörden dış veri belleğine veri aktarır. İki tür MOVX komutu vardır, bunlardan biri 8 bit adres kullanır ve sadece 256 satırlık dış veri belleğinden işlem yapabilir. Diğeri ise 16 bit adres kullanır ve 64K'lık bellek alanından işlem yapabilir. Birinci tip MOVX komutlarında adres gösterici olarak seçili bulunan R0 ve R1 yazaçları kullanılır.

Adres 8 bit olduğu için P0 portu adres veri yolu multiplex olarak kullanılır. P2 portu ise genel amaçlı giriş/çıkış portu olarak kullanılabilir. Adreslenebilecek bellek kapasitesi çok düşük olduğunda başka hatlar kullanılarak bu kapasite artırılabilir. Yalnız bu hatların seviyeleri MOVX komutu öncesi programcı tarafından belirlenmelidir. İkinci tip MOVX komutlarında 16 bit adres DPTR yazacı tarafından belirlenir. P2 portundan adresin YDB'ı çıkılır (DPH), P0 portu ise adresin DDB (DPL) ile veri yolu tarafından zaman paylaşımli olarak kullanılır. Bu tip movx komutu ile 64 K'lık bellek alanından işlem yapılabilir.

### **MOVX A, @Ri**

Bayt: 1  
İşlem süresi: 2  
Opkodu: 1110001i  
İşlem:  $(A) \leftarrow ((Ri))$

### **MOVX A, @DPTR**

Bayt: 1  
İşlem süresi: 2  
Opkodu: 11100000  
İşlem:  $(A) \leftarrow ((DPTR))$

### **MOVX @Ri, A**

Bayt: 1  
İşlem süresi: 2  
Opkodu: 11110011  
İşlem:  $((Ri)) \leftarrow (A)$

### **MOVX @DPTR, A**

Bayt: 1  
İşlem süresi: 2  
Opkodu: 11110000  
İşlem:  $(DPTR) \leftarrow (A)$

### **NOP**

İşlev: İşlem yapma.

**Açıklama:** PC bir arttırılarak bir sonraki komutu yürütmeye devam eder. İşlenmesi 1 makine saykılı gecikme yaratır. PC dışında hiçbir yazaç veya bayrak işleminden etkilenmez.

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 00000000  
 İşlem:  $(PC) \leftarrow (PC) + 1$

### **ORL <hedef baytı>,<kaynak baytı>**

**İşlev:** Bayt değişkenleri mantık VEYAlar.  
**Açıklama:** Belirtilen bayt değişkenleri aynı numaralı bitlerini mantık VEYAlar. Sonucu hedef adreste saklar, işlemde bayraklar etkilenmez. Her iki işlenende toplam 6 değişik adresleme kipinde kullanılabilir.  
**Not:** Bu komut çıkış portunun içeriğini başka sayı ile VEYAladığında tutucu çıkışlarını okuyarak VEYAlar. Port bacaklarında oluşan değerlere bakmaz

### **ORL A,Rn**

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 01001rrr  
 İşlem:  $(A) \leftarrow (A) \vee (Rn)$

### **ORL A,doğrudan adres**

Bayt: 2  
 İşlem süresi: 1  
 Opkodu: 01000101 aaaaaaaaa  
 İşlem:  $(A) \leftarrow (A) \vee (\text{doğrudan adres})$

### **ORL A,@Ri**

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 0100011i  
 İşlem:  $(A) \leftarrow (A) \vee (Ri)$

### **ORL A,#veri**

Bayt: 2  
 İşlem süresi: 1  
 Opkodu: 01000100 dddddddd  
 İşlem:  $(A) \leftarrow (A) \vee \#veri$

#### ORL doğrudan adres,A

Bayt: 2  
 İşlem süresi: 1  
 Opkodu: 01000010 aaaaaaaa  
 İşlem:  $(doğrudan\ adres) \leftarrow (doğrudan\ adres) \vee (A)00$

#### ORL doğrudan adres, #Veri

Bayt: 3  
 İşlem süresi: 2  
 Opkodu: 01000011 aaaaaaaa dddddddd  
 İşlem:  $(doğrudan\ adres) \leftarrow (doğrudan\ adres) \vee Veri$

#### ORL C, <Kaynak Bit>

İşlev: Bit değişkeni VEYA'la.

Açıklama: kaynak biti 1 ise  $C=1$  yapar, kaynak biti 0 ise  $C$ 'yi olduğu gibi bırakır. Kaynak biti önüne / işareti konursa kaynak bitinin değili ile  $C$  VEYA'lanır. Fakat bu durumda kaynak bit değişmez.

#### ORL C, Bit

Bayt: 2  
 İşlem süresi: 2  
 Opkodu: 01110010 bbbbbbbb  
 İşlem:  $C \leftarrow C \vee (Bit)$

#### ORL C, /Bit

Bayt: 2  
 İşlem süresi: 2  
 Opkodu: 10100000 bbbbbbbb  
 İşlem:  $C \leftarrow C \vee / (Bit)$

#### POP Doğrudan Adres

İşlev: Yığından bayt çeker hedefe yazar.

**Açıklama:** Yığın işaretleyicinin gösterdiği yığın satırı okunarak belirtilen hedefe yazılır. İşlem sonunda yığın işaretleyici bir azaltılır. İşlemden bayraklar etkilenmez.

Bayt: 2  
 İşlem süresi: 2  
 Opkodu: 11010000 aaaaaaaa  
 İşlem: (doğrudan adres)  $\leftarrow$  ((SP))  
 (SP)  $\leftarrow$  (SP) - 1

### **PUSH Doğrudan Adres**

**İşlev:** Yığına bayt atar.

**Açıklama:** Yığın işaretleyicinin içeriği bir arttırılır ve belirtilen belleğin içeriği yığına atılır. İşlemden bayraklar etkilenmez.

Bayt: 2  
 İşlem süresi: 2  
 Opkodu: 11000000 aaaaaaaa  
 İşlem: (SP)  $\leftarrow$  (SP) + 1  
 ((SP))  $\leftarrow$  (doğrudan adres)

### **RET**

**İşlev:** Alt programdan geri dön.

**Açıklama:** RET komutu yığından PC'nin yüksek değerli baytı ve düşük değerli baytını çekerek program yürütülmesini bu adresten devam etmesini sağlar. Genellikle bu adres ACALL veya LCALL komutundan bir sonraki komutun adresidir. İşlemden bayraklar etkilenmez.

Bayt: 1  
 İşlem süresi: 2  
 Opkodu: 00100010  
 İşlem:  
 $(PC_{15} - PC_8) \leftarrow ((SP))$   
 (SP)  $\leftarrow$  (SP) - 1  
 $(PC_7 - PC_0) \leftarrow ((SP))$   
 (SP)  $\leftarrow$  (SP) - 1

### **RETI**

**İşlev:** Kesmeden geri dön.

**Açıklama:** RETI komutu yığından PC'nin yüksek değerli baytı ve düşük değerli baytını çekerek program yürütülmesini bu adresten devam etmesini sağlar.

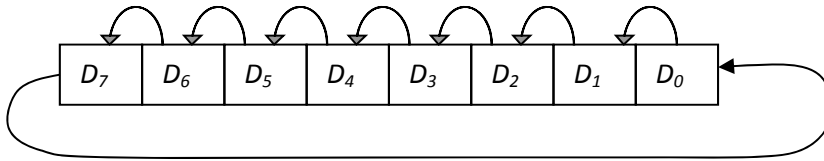
PROGRAM DURUM GÖSTERGE YAZACI kesme ile otomatik olarak saklanmaz, programcının kesme servis alt programı başlangıcında saklamalıdır ve RETI komutu öncesi tekrar geri almalıdır. İşlemden bayraklar etkilenmez.

Bayt: 1  
 İşlem süresi: 2  
 Opkodu: 00110010  
 İşlem:  $(PC_{15} - PC_8) \leftarrow (SP)$   
 $(SP) \leftarrow (SP) - 1$   
 $(PC_7 - PC_0) \leftarrow (SP)$   
 $(SP) \leftarrow (SP) - 1$

### RLA

İşlev: Akümülatörü sola döndür.  
 Açıklama: Akümülatör içerisindeki 8 bit bir bit sola döndürülür. Bit 7 bit sıfırın bulunduğu yere gelir. Dışarıdan bir bit girişi yoktur. İşlemden bayraklar etkilenmez.

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 00100011  
 İşlem:  $(D_{n+1}) \leftarrow (D_n), n=0 - 6$   
 $(D_0) \leftarrow (D_7)$



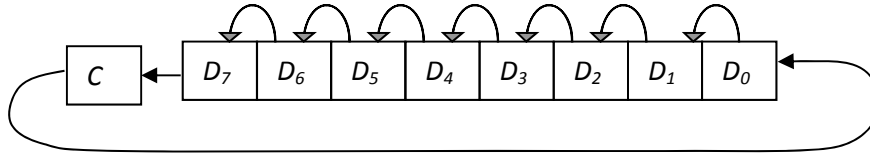
### RLCA

İşlev: Akümülatörü elde üzerinden sola döndür.  
 Açıklama: Akümülatör içerisindeki 8 bit bir bit sola döndürülür. Bit 7 Elde Bayrağına (CY) Elde Bayrağının (CY) içeriği ise bit sıfırın bulunduğu yere gelir. Elde Bayrağından (CY) başka bayrak işlemden etkilenmez.

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 00110011  
 İşlem:  $(A_n + 1) \leftarrow (A_n), n=0 - 6$



$$(A_0) \leftarrow (C)$$

$$(C) \leftarrow (A_7)$$


### RRA

**İşlev:** Akümülatörü sağa döndür.

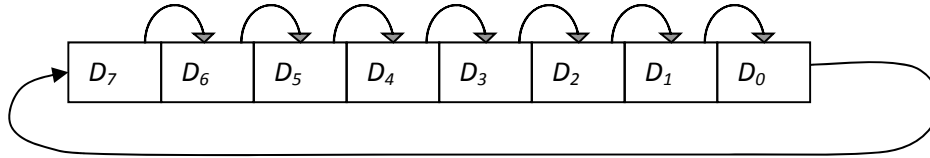
**Açıklama:** Akümülatör içerisindeki 8 bit bir bit sağa döndürülür. Bit 0 bit 7'nin bulunduğu yere gelir. Dışarıdan bir bit girişi yoktur. İşlemden bayraklar etkilenmez.

Bayt: 1

İşlem süresi: 1

Opkodu: 00000011

İşlem:  $(A_n) \leftarrow (A_n + 1), n=0 - 6$

$$(A_7) \leftarrow (A_0)$$


### RRC A

**İşlev:** Akümülatörü elde üzerinden sağa döndür.

**Açıklama:** Akümülatör içerisindeki 8 bit bir bit sağa döndürülür. Bit 0 Elde Bayrağına (CY) Elde Bayrağının (CY) içeriği ise bit 7'nin bulunduğu yere gelir. Elde Bayrağından (CY) başka bayrak işlemden etkilenmez.

Bayt: 1

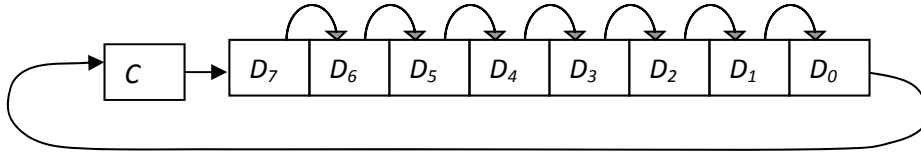
İşlem süresi: 1

Opkodu: 00110011

İşlem:  $(A_n) \leftarrow (A_n + 1), n=0 - 6$

$$(A_7) \leftarrow (C)$$

$$(C) \leftarrow (A_0)$$



### SETB <bit>

**İşlev:** Bit değişkeni kurar.

**Açıklama:** belirtilen bit adresin içeriğini 1 yapar. Elde bayrağında ve doğrudan adreslenebilen bitlerin içeriğini kurar. Başka bayrak işleminden etkilenmez.

### SETB C

**Bayt:** 1

**İşlem süresi:** 1

**Opkodu:** 11010011

**İşlem:**  $C \leftarrow 1$

### SETB bit

**Bayt:** 1

**İşlem süresi:** 1

**Opkodu:** 11010010 bbbbbbbb

**İşlem:**  $(\text{Bit}) \leftarrow 1$

### SJMP Bağlı Adres

**İşlev:** Kısa bağlanma.

**Açıklama:** Koşulsuz olarak belirtilen adrese bağlanır. Bağlanma adresi 2 arttırılmış PC'nin içeriğine opkod sonrası belirtilen işaretli kayıklık toplanarak elde edilir. Bağlanma uzaklığı 127 adım ileri ve 128 adım geriye doğrudur. Bağlanılacak uzaklık az olduğu için kısa bağlanma komutu olarak adlandırılır.

**Bayt:** 2

**İşlem süresi:** 2

**Opkodu:** 10000000 eeeeeeee

**İşlem:**  $(PC) \leftarrow (PC) + 2$

$(PC) \leftarrow (PC) + \text{Byt\_2}$

### SUBB A, <Kaynak Bayt>

**İşlev:** Elde ile birlikte çıkar.

*Açıklama: Bu komut belirtilen bayt adresin içeriği ile elde bayrağının içeriğini akümülatörden çıkarır. SUBB komutu eğer bit 7'lerin çıkarılmasında borç gerekirse elde bayrağını kurar aksi halde elde bayrağını temizler. Çıkarma işlemi öncesi elde bayrağı kurulu olursa bunun anlamı bundan önceki çıkarma işleminde borç alınmıştır, bu borç bu çıkarma işleminde akümülatörden alınacaktır. Bu işlem birden fazla uzunluktaki sayıların çıkarma işleminde kullanılacaktır. AC bayrağı ise bit 3'lerin işleminde borç gerekirse kurulur diğer durumda temizlenir. Taşma Bayrağı (OV) bit 6'ların işleminde borç gerekirse kurulur eğer gerekmez ise bit 7'lerin işleminde borç gerekirse kurulur. İşaretli sayılar ile işlem yaparken Taşma Bayrağının (OV) kurulu olması pozitif sayıdan negatif sayı çıkarıldı ve negatif sonuç elde edildiğini veya negatif sayıdan pozitif çıkarıldığında pozitif sayının elde edildiğini gösterir.*

#### **SUBB A, Rn**

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 10011rrr  
 İşlem:  $(A) \leftarrow (A) - (C) - (Rn)$

#### **SUBB A, Doğrudan Adres**

Bayt: 2  
 İşlem süresi: 1  
 Opkodu: 10010101 aaaaaaaa  
 İşlem:  $(A) \leftarrow (A) - (C) - (\text{Doğrudan Adres})$

#### **SUBB A, @Ri**

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 1001011i  
 İşlem:  $(A) \leftarrow (A) - (C) - ((Ri))$

#### **SUBB A, #Veri**

Bayt: 2  
 İşlem süresi: 1  
 Opkodu: 10010100 dddddddd  
 İşlem:  $(A) \leftarrow (A) - (C) - \#Veri$

**SWAP A**

**İşlev:** Akümülatörün içeriğinin (nible) dörtlü bit gruplarını yer değiştirir.

**Açıklama:** Düşük değerli dört bit ile yüksek değerli dört biti karşılıklı yer değiştirir.

Sadece akümülatörde çalışan bir komuttur. İşlemden bayraklar etkilenmez.

Bayt: 1

İşlem süresi: 1

Opkodu: 11000100

İşlem:  $(A_3 - A_0) \leftrightarrow (A_7 - A_4)$

**XCH A, <Bayt>**

**İşlev:** Akümülatör ile bayt değişkenin içeriğini takas eder.

**Açıklama:** Akümülatör ile belirtilen adresteki baytları takas eder. İşlemden bayraklar etkilenmez.

**XCH A, Rn**

Bayt: 1

İşlem süresi: 1

Opkodu: 11001rrr

İşlem:  $(A) \leftrightarrow (Rn)$

**XCH A, Doğrudan Adres**

Bayt: 2

İşlem süresi: 1

Opkodu: 11000101 aaaaaaaaa

İşlem:  $(A) \leftrightarrow (\text{Doğrudan Adres})$

**XCH A, @Ri**

Bayt: 1

İşlem süresi: 1

Opkodu: 1100011i

İşlem:  $(A) \leftrightarrow ((Ri))$

**XCHD, @Ri**

**İşlev:** Akümülatör ile bayt değişkenin düşük değerli 4 bitlerini takas eder.

*Açıklama: Akümülatör ile belirtilen adresin içeriklerinin düşük değerli 4 bitleri takas edilir. İKO sayıların işlemlerinde bu komut kullanılabilir. İşlemden bayraklar etkilenmez.*

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 1101011i  
 İşlem:  $(A3 - A0) \leftrightarrow ((Ri3 - Ri0))$

### ***XRL <hedef baytı>, <kaynak baytı>***

*İşlev: Bayt değişkenleri ÖZEL VEYAlar (EXOR)*

*Açıklama: XRL komutu belirtilen kaynak ile hedef adresin içeriklerini aynı numaralı bitlerini ÖZEL VEYAlar ve hedefe yazar. İşlemden bayraklar etkilenmez.*

*Not: Bu komut çıkış portunun içeriğini başka sayı ile ÖZEL VEYAladığında tutucu çıkışlarını okuyarak ÖZEL VEYAlar. Port bacaklarında oluşan değerlere bakmaz*

### ***XRL A, Rn***

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 01101rrr  
 İşlem:  $(A) \leftarrow (A) \oplus (Rn)$

### ***XRL A, Doğrudan Adres***

Bayt: 2  
 İşlem süresi: 1  
 Opkodu: 01100101 aaaaaaaaa  
 İşlem:  $(A) \leftarrow (A) \oplus (\text{Doğrudan Adres})$

### ***XRL A, @Ri***

Bayt: 1  
 İşlem süresi: 1  
 Opkodu: 0110011i  
 İşlem:  $(A) \leftarrow (A) \oplus ((Ri))$

### ***XRL A, #veri***

Bayt: 2  
 İşlem süresi: 1

Opkodu: 01100101 dddddddd

İşlem:  $(A) \leftarrow (A) \oplus \#veri$

### ***XRL Doğrudan Adres, A***

Bayt: 2

İşlem süresi: 1

Opkodu: 01100010 aaaaaaaaa

İşlem:  $(\text{Doğrudan Adres}) \leftarrow (\text{Doğrudan Adres}) \oplus (A)$

### ***XRL Doğrudan adres, #veri***

Bayt: 3

İşlem süresi: 2

Opkodu: 01100011 aaaaaaaaa dddddddd

İşlem:  $(\text{DoğrudanAdres}) \leftarrow (\text{Doğrudan Adres}) \oplus \#veri$

## Kaynakça

1. MacKenzie Scott. *The 8051 Microcontroller*, Prentice Hall. 3rd. Ed. 1999
2. Yeralan and Ahluwalia. *Programming and Interfacing the 8051 Microcontroller*. Addison-Wesley. 1995.
3. Haluk Gümüşkaya, *Mikroişlemciler ve 8051 ailesi*, Alfa yayıncılık, 1998,2002.
4. Jan Axelson, *Her yönüyle 8051/52*, Bilişim yayınları, 2000 (Türkçe çevirisi).
5. Vault Information Services firmasının 8051 hakkında hazırladığı bir sitedir. <http://www.8052.com>
6. Intel 8051 mikrodnetleyici veri yaprakları. <http://developer.intel.com/design/mcs51/>
7. Philips 8051 mikrodnetleyici veri yaprakları.
8. <http://www-us.semiconductors.philips.com/microcontrol/>
9. Infineon (eski Siemens) 8051 mikrodnetleyici veri yaprakları. <http://www.infineon.com/products/micro/micro.htm>
10. Keil 8051 program geliştirme yazılımı hakkında bilgi ve örnek programlar içerir. <http://www.keil.com/home.htm>
11. Raisonance 8051 program geliştirme yazılımı hakkında bilgi ve örnek programlar içerir. <http://www.raisonance.com>
12. Analog Devices ADuC812 (8051/8052 uyumlu) işlemcilerinin veri yapraklarını içerir. [www.analog.com/microconverter](http://www.analog.com/microconverter)





## Dictionary

### A

<i>Absolute addressing</i>	<i>Mutlak adresleme</i>
<i>Absolute Jump</i>	<i>Mutlak bağlanma</i>
<i>Access time</i>	<i>Erişim Zamanı</i>
<i>Active High</i>	<i>Aktif Yüksek</i>
<i>Active Low</i>	<i>Aktif Düşük</i>
<i>Addition</i>	<i>Toplama</i>
<i>Address</i>	<i>Adres</i>
<i>Address Bus</i>	<i>Adres Yolu</i>
<i>Address Decoding</i>	<i>Adres Kod Çözümü</i>
<i>Address Latch Enable</i>	<i>Adres tutucusu izinleme</i>
<i>Addressing Modes</i>	<i>Adresleme kipleri</i>
<i>ALE</i>	<i>Adres tutucusu izinleme</i>
<i>Architecture</i>	<i>Mimari</i>
<i>Arithmetic Instruction</i>	<i>Aritmetik işlem komutu</i>
<i>Assembler</i>	<i>Birleştirici</i>
<i>Assembly</i>	<i>Birleştirme</i>
<i>Assembly language</i>	<i>Birleştirici dili, Assembly dili</i>

### B

<i>Baud Rate</i>	<i>iletişim hızı</i>
<i>BCD, Binary Coded Decimal</i>	<i>İKO, ikilik kodlanmış onlu</i>
<i>Bidirectional</i>	<i>İki Yönlü</i>
<i>Bill Of Material, BOM</i>	<i>Malzeme Listesi</i>
<i>Binary</i>	<i>İkili</i>
<i>Binary Coded Decimal, BCD</i>	<i>İkilik Kodlanmış onlu, İKO</i>
<i>Boolean Processing</i>	<i>Lojik Operasyonlar</i>
<i>Branch Prediction</i>	<i>Dallanma Tahmini</i>
<i>Branching Instructions</i>	<i>Dallanma komutları</i>
<i>Buffer</i>	<i>Tampon</i>

<i>Bus</i>	<i>Yol</i>
<i>Bus cycle</i>	<i>Yol Çevrimi</i>
<i>Cache</i>	<i>Önbellek</i>
<i>Carry flag</i>	<i>Elde bayrağı</i>
<i>Central Processing Unit, CPU</i>	<i>Merkezi İşlem Birimi, MİB</i>
<i>Check Sum</i>	<i>Toplam ile denetleme</i>
<i>Chip</i>	<i>Tümdevre</i>
<i>CISC, Complex Instruction Set Code</i>	<i>Karmaşık komut kümesi kodu</i>
<i>Clock</i>	<i>Saat</i>
<i>Code memory</i>	<i>Program belleği</i>
<i>Column Decoder</i>	<i>Sütün Kod Çözücü</i>
<i>Common ANOT</i>	<i>Ortak Anotlu</i>
<i>Common Bus</i>	<i>Ortak Yol</i>
<i>Common I/O</i>	<i>Ortak Giriş/Çıkış</i>
<i>Common KATOT</i>	<i>Ortak Katotlu</i>
<i>Compile</i>	<i>Derleme</i>
<i>Complement</i>	<i>Tümleyen</i>
<i>Complex Instruction Set Code, CISC</i>	<i>Karmaşık komut kümesi kodu</i>
<i>Complier</i>	<i>Derleyici</i>
<i>Control Bus</i>	<i>Kontrol Yolu</i>
<i>Control Word</i>	<i>Kontrol Kelimesi</i>
<i>Core</i>	<i>Çekirdek</i>
<i>Count</i>	<i>Sayma</i>
<i>Counter</i>	<i>Sayıcı</i>

**D**

<i>Data</i>	<i>Veri</i>
<i>Data Acquisition</i>	<i>Veri Toplama</i>
<i>Data Bus</i>	<i>Veri Yolu</i>
<i>Data Bus</i>	<i>Veri Yolu</i>
<i>Data Flow</i>	<i>Veri Akışı</i>
<i>Data memory</i>	<i>Veri belleği</i>

<i>Data Processing</i>	<i>Veri İşleme</i>
<i>Data Transfer Instructions</i>	<i>Veri aktarma komutları</i>
<i>Debouching</i>	<i>Zıplama</i>
<i>Decoder</i>	<i>Kod Çözücü</i>
<i>Destination</i>	<i>Hedef</i>
<i>Development Kit</i>	<i>Geliştirme kiti</i>
<i>Digit</i>	<i>Hane</i>
<i>Direct addressing</i>	<i>Doğrudan adresleme</i>
<i>Direct Memory Access</i>	<i>Doğrudan bellek erişimi</i>
<i>Disabled</i>	<i>Etkin olmayan duruma alma</i>
<i>DMA</i>	<i>Doğrudan bellek erişimi</i>
<i>Duplex</i>	<i>Alıcı ve Verici birimleri olan</i>

**E**

<i>Edge</i>	<i>Kenar</i>
<i>Edge triggered</i>	<i>Kenar Tetiklemeli</i>
<i>EEPROM</i>	<i>Elektrik ile programlanabilir elektrik ile silinebilir sadece okunabilir bellek</i>
<i>Electrical Programmable ROM</i>	<i>Elektrik ile programlanabilir ışıkla silinebilir sadece okunabilir bellek</i>
<i>Embedded</i>	<i>İçinde</i>
<i>Enable</i>	<i>İzinleme</i>
<i>Enabled</i>	<i>izinlendiğinde</i>
<i>EPROM</i>	<i>Elektrik ile programlanabilir ışıkla silinebilir sadece okunabilir bellek</i>
<i>Evolution Kit</i>	<i>Değerlendirme kiti</i>
<i>Execute</i>	<i>Komut Yürütme</i>
<i>Execution</i>	<i>Yürütme</i>
<i>External code memory</i>	<i>Dış program belleği</i>
<i>External data memory</i>	<i>Dış veri belleği</i>

**F**

<i>Factory-masked</i>	<i>Fabrika Maskeli</i>
-----------------------	------------------------

<i>Falling Edge</i>	<i>Düşen Kenar</i>
<i>Fetch</i>	<i>Komut Okuma</i>
<i>Flag</i>	<i>Bayrak</i>
<i>FLASH EEPROM</i>	<i>Elektrik ile programlanabilir elektrik ile silinebilir sadece okunabilir bellek</i>
<i>Floating-Point Unit, FPU</i>	<i>Kayan-Nokta Birimi</i>
<i>Flow chart</i>	<i>Akış diyagramı</i>
<i>Full Decoding</i>	<i>Tam Kod Çözücü</i>
<i>Full-Duplex</i>	<i>Aynı anda veriyi alıp vermek</i>
<b>G</b>	
<i>Gate</i>	<i>Kapı</i>
<b>H</b>	
<i>Half-Duplex</i>	<i>Aynı anda alma veya verme</i>
<i>Hand Shake</i>	<i>El sıkışma</i>
<i>Immediate addressing</i>	<i>İvedi adresleme</i>
<i>In-circuit Emulator</i>	<i>Devre üstünde deneme</i>
<i>Indexed addressing</i>	<i>Sıralı adresleme</i>
<i>Indirect addressing</i>	<i>Dolaylı adresleme</i>
<i>Input</i>	<i>Giriş</i>
<i>Instruction</i>	<i>Komut</i>
<i>Instruction Queue</i>	<i>Komut Sırası</i>
<i>Instruction Register, IE</i>	<i>Komut Saklayıcısı</i>
<i>Instruction set</i>	<i>Komut Kümesi</i>
<i>Interface</i>	<i>Arabirim</i>
<i>Internal RAM</i>	<i>İç RAM</i>
<i>Interrupt</i>	<i>Kesme</i>
<i>Interrupt Request</i>	<i>Kesme İsteği</i>
<i>Interrupt Service Routine</i>	<i>Kesme Hizmet Programı</i>
<b>İ</b>	
<i>Jump backward</i>	<i>Geri dallanma</i>
<i>Jump forward</i>	<i>İleri dallanma</i>

**K****L**

<i>Level</i>	<i>Seviye</i>
<i>Level Triggered</i>	<i>Seviye Tetiklemeli</i>
<i>Linear Address</i>	<i>Doğrusal Adres</i>
<i>Linear Selection</i>	<i>Doğrusal Seçimli</i>
<i>Location</i>	<i>Yer</i>
<i>Logical Word</i>	<i>Lojik Kelime</i>
<i>Long addressing</i>	<i>Uzun adresleme</i>
<i>Look-up Table</i>	<i>Başvuru Tablosu</i>

**M**

<i>Machine Cycle</i>	<i>Makine Çevrimi</i>
<i>Mask-ROM</i>	<i>Üretilen firma tarafından programlanmış sadece okunabilir bellek</i>
<i>Memory</i>	<i>Bellek</i>
<i>Memory</i>	<i>Hafıza</i>
<i>Memory Address Space</i>	<i>Bellek Adres Alanı</i>
<i>Memory Address Space</i>	<i>Hafıza Adres Alanı</i>
<i>Memory Array</i>	<i>Bellek Dizisi</i>
<i>Memory Array</i>	<i>Hafıza Dizisi</i>
<i>Memory Cell</i>	<i>Bellek Hücresi</i>
<i>Memory Cell</i>	<i>Hafıza Hücresi</i>
<i>Memory Decoder</i>	<i>Bellek Kod Çözücü</i>
<i>Memory Decoder</i>	<i>Hafıza Kod Çözücü</i>
<i>Memory Management</i>	<i>Bellek Yönetimi</i>
<i>Memory Management</i>	<i>Hafıza Yönetimi</i>
<i>Memory Map</i>	<i>Bellek Haritası</i>
<i>Memory Map</i>	<i>Hafıza Haritası</i>
<i>Microcontroller</i>	<i>Mikrodenetleyici</i>
<i>Microprocessor</i>	<i>Mikroişlemci</i>
<i>Multiplexed</i>	<i>Çoklu seçenekten Seçme</i>

<i>Multi-tasking</i>	<i>Çok amaçlı, çok görevli</i>
<i>Multi-user</i>	<i>Çok kullanıcı</i>
<b>N</b>	
<i>Negative Clock Pulse</i>	<i>Negatif Saat Darbesi</i>
<i>Negative Transition</i>	<i>Negatif Geçiş</i>
<i>Nibble</i>	<i>Dört bit</i>
<b>O</b>	
<i>On-Chip</i>	<i>Tümdevre Üzeri</i>
<i>One Time Programmable</i>	<i>Bir defa programlanabilir</i>
<i>Op-code</i>	<i>İşlem kodu</i>
<i>Operand</i>	<i>İşlem Verisi</i>
<i>Operation Code</i>	<i>İşlem Kodu</i>
<i>OTP</i>	<i>Bir defa programlanabilir</i>
<i>Output</i>	<i>Çıkış</i>
<i>Output data bus</i>	<i>Çıkış veri yolu</i>
<i>Output device</i>	<i>Çıkış cihazı</i>
<i>Overflow</i>	<i>Taşma</i>
<b>P</b>	
<i>Paging</i>	<i>Sayfalama</i>
<i>Parity Bit</i>	<i>Eşlik biti</i>
<i>Partial Decoding</i>	<i>Kısmi Kod Çözümü</i>
<i>Pin</i>	<i>Uç</i>
<i>Pipelined</i>	<i>İş hatlı</i>
<i>Pointer</i>	<i>İşaretçi, gösterici</i>
<i>Positive Clock Pulse</i>	<i>Pozitif Saat Darbesi</i>
<i>Positive Transition</i>	<i>Pozitif Geçiş</i>
<i>PPM, Pulse Per Minute</i>	<i>Dakikadaki vuru sayısı</i>
<i>Priority</i>	<i>Öncelik</i>
<i>Processor</i>	<i>İşlemci</i>
<i>Program Control Instructions</i>	<i>Program denetim komutları</i>
<i>Program Development Environment</i>	<i>Program Geliştirme Ortamı</i>

<i>Program Status Word, PSW</i>	<i>Durum yazacı</i>
<i>Programmable Read Only Memory, PROM</i>	<i>Programlanabilir sadece okunabilen bellek</i>
<i>PROM</i>	<i>Programlanabilir sadece okunabilen bellek</i>
<i>Protected Mode</i>	<i>Korumalı Mod</i>

**R**

<i>Random Access</i>	<i>Rasgele Erişimli</i>
<i>Read</i>	<i>Okuma</i>
<i>Read Only Memory, ROM</i>	<i>Salt Okunabilir Hafıza</i>
<i>Real-Mode</i>	<i>Gerçek mod</i>
<i>Real-Time</i>	<i>Gerçek Zaman</i>
<i>Reduced Instruction Set Code</i>	<i>İndirgenmiş komut kümesi kodu</i>
<i>Register</i>	<i>Saklayıcı</i>
<i>Register</i>	<i>Yazaç</i>
<i>Register addressing</i>	<i>Yazaç adresleme</i>
<i>Register Set Bank</i>	<i>Saklayıcı Kümesi</i>
<i>Relative addressing</i>	<i>Bağıl adresleme</i>
<i>Relocatable</i>	<i>Tekrar Yerleştirilebilir</i>
<i>Reset</i>	<i>Sıfırlamak, başlangıç konumuna getirme</i>
<i>RISC</i>	<i>İndirgenmiş komut kümesi kodu</i>
<i>Rising Edge</i>	<i>Yükselen Kenar</i>
<i>RMS, Root Mean Square</i>	<i>Etkin değer, AC voltmetrenin ölçtüğü değer</i>
<i>Rotate</i>	<i>Döndürme</i>
<i>Row Decoder</i>	<i>Satır Kod Çözücü</i>
<i>RPM</i>	<i>Dakikadaki vuru sayısı</i>

**S**

<i>Set</i>	<i>Kurmak, 1 yapmak</i>
<i>Shift</i>	<i>Öteleme</i>
<i>Simulation</i>	<i>Benzetim</i>

<i>Slot</i>	<i>Yuva</i>
<i>Source</i>	<i>Kaynak</i>
<i>Source File</i>	<i>Kaynak Dosya</i>
<i>Stack</i>	<i>Yığın</i>
<i>Stack Pointer, SP</i>	<i>Yığın İşaretçisi, Göstericisi</i>
<i>State</i>	<i>Durum</i>
<i>Subroutine</i>	<i>Altprogram</i>
<i>Subtract with Borrow</i>	<i>Borç ile birlikte çıkar</i>
<i>Subtraction</i>	<i>Çıkarma</i>
<i>Successive Approximation</i>	<i>Ardışıl Yaklaşım</i>
<i>Synchronization</i>	<i>Eşzamanlama</i>
<i>System Bus</i>	<i>Sistem Yolu</i>

**T**

<i>Target address</i>	<i>Hedef Adres</i>
<i>Time delay loop</i>	<i>Zaman geciktirme döngüsü</i>
<i>Timer</i>	<i>Zamanlayıcı</i>
<i>Timing</i>	<i>Zamanlama</i>
<i>Transmit</i>	<i>Vermek, iletmet</i>
<i>Transmitter</i>	<i>Verici</i>
<i>Transparent</i>	<i>Saydam</i>
<i>Trigger</i>	<i>Tetikleme</i>
<i>Two-level Decoding</i>	<i>İki Seviyeli Kod Çözücü</i>

**U**

<i>UART</i>	<i>Universal Asenkron Reciver Transmitter</i>
<i>Unconditional jump instructions</i>	<i>Koşulsuz dallanma komutları</i>
<i>Unidirectional</i>	<i>Tek Yönlü</i>
<i>USART</i>	<i>Universal Senkron Asenkron Reciver Transmitter</i>

**V**

<i>Virtual</i>	<i>Sanal</i>
<i>Virtual Memory</i>	<i>Sanal bellek</i>



<i>Wait State</i>	<i>Bekleme Durumu</i>
<i>Word</i>	<i>Kelime</i>
<i>Word Length</i>	<i>Kelime Uzunluğu</i>
<i>Write</i>	<i>Yazma</i>
<i>Z</i>	
<i>Zone</i>	<i>Bölge</i>

## Sözlük

### A

<i>Adres</i>	<i>Address</i>
<i>Adres Kod Çözümü</i>	<i>Address Decoding</i>
<i>Adres tutucusu izinleme</i>	<i>Address Latch Enable</i>
<i>Adres tutucusu izinleme</i>	<i>ALE</i>
<i>Adres Yolu</i>	<i>Address Bus</i>
<i>Adresleme kipleri</i>	<i>Addressing Modes</i>
<i>Akış diyagramı</i>	<i>Flow chart</i>
<i>Aktif Düşük</i>	<i>Active Low</i>
<i>Aktif Yüksek</i>	<i>Active High</i>
<i>Alıcı ve Verici birimleri olan</i>	<i>Duplex</i>
<i>Altprogram</i>	<i>Subroutine</i>
<i>Arabirim</i>	<i>Interface</i>
<i>Ardışıl Yaklaşım</i>	<i>Successive Approximation</i>
<i>Aritmetik işlem komutu</i>	<i>Arithmetic Instruction</i>
<i>Aynı anda alma veya verme</i>	<i>Half-Duplex</i>
<i>Aynı anda veriyi alıp vermek</i>	<i>Full-Duplex</i>

### B

<i>Bağıl adresleme</i>	<i>Relative addressing</i>
<i>Başvuru Tablosu</i>	<i>Look-up Table</i>
<i>Bayrak</i>	<i>Flag</i>
<i>Bekleme Durumu</i>	<i>Wait State</i>
<i>Bellek</i>	<i>Memory</i>
<i>Bellek Haritası</i>	<i>Memory Map</i>
<i>Bellek Adres Alanı</i>	<i>Memory Address Space</i>
<i>Bellek Dizisi</i>	<i>Memory Array</i>
<i>Bellek Hücresi</i>	<i>Memory Cell</i>
<i>Bellek Kod Çözücü</i>	<i>Memory Decoder</i>
<i>Bellek Yönetimi</i>	<i>Memory Management</i>

<i>Benzetim</i>	<i>Simulation</i>
<i>Bir defa programlanabilir</i>	<i>One Time Programmable</i>
<i>Bir defa programlanabilir</i>	<i>OTP</i>
<i>Birleştirici</i>	<i>Assembler</i>
<i>Birleştirici dili, Assembly dili</i>	<i>Assembly language</i>
<i>Birleştirme</i>	<i>Assembly</i>
<i>Borç ile birlikte çıkar</i>	<i>Subtract with Borrow</i>
<i>Bölge</i>	<i>Zone</i>

## Ç

<i>Çekirdek</i>	<i>Core</i>
<i>Çıkarma</i>	<i>Subtraction</i>
<i>Çıkış</i>	<i>Output</i>
<i>Çıkış cihazı</i>	<i>Output device</i>
<i>Çıkış veri yolu</i>	<i>Output data bus</i>
<i>Çok amaçlı, çok görevli</i>	<i>Multi-tasking</i>
<i>Çok kullanıcı</i>	<i>Multi-user</i>
<i>Çoklu seçenekten Seçme</i>	<i>Multiplexed</i>

## D

<i>Dakikadaki ripıl sayısı</i>	<i>RPM</i>
<i>Dakikadaki vuru sayısı</i>	<i>PPM, Pulse Per Minute</i>
<i>Dallanma komutları</i>	<i>Branching Instructions</i>
<i>Dallanma Tahmini</i>	<i>Branch Prediction</i>
<i>Değerlendirme kiti</i>	<i>Evolution Kit</i>
<i>Derleme</i>	<i>Compile</i>
<i>Derleyici</i>	<i>Complier</i>
<i>Devre üstünde deneme</i>	<i>In-circuit Emulator</i>
<i>Dış program belleği</i>	<i>External code memory</i>
<i>Dış veri belleği</i>	<i>External data memory</i>
<i>Doğrudan adresleme</i>	<i>Direct addressing</i>
<i>Doğrudan bellek erişimi</i>	<i>Direct Memory Access</i>
<i>Doğrudan bellek erişimi</i>	<i>DMA</i>

<i>Doğrusal Adres</i>	<i>Linear Address</i>
<i>Doğrusal Seçimli</i>	<i>Linear Selection</i>
<i>Dolaylı adresleme</i>	<i>Indirect addressing</i>
<i>Döndürme</i>	<i>Rotate</i>
<i>Dört bit</i>	<i>Nibble</i>
<i>Durum</i>	<i>State</i>
<i>Durum yazacı</i>	<i>Program Status Word, PSW</i>
<i>Düşen Kenar</i>	<i>Falling Edge</i>

**E**

<i>El sıkışma</i>	<i>Hand Shake</i>
<i>Elde bayrağı</i>	<i>Carry flag</i>
<i>Elektrik ile programlanabilir elektrik ile silinebilir sadece okunabilir bellek</i>	<i>EEPROM</i>
<i>Elektrik ile programlanabilir elektrik ile silinebilir sadece okunabilir bellek</i>	<i>FLASH EEPROM</i>
<i>Elektrik ile programlanabilir ışıkla silinebilir sadece okunabilir bellek</i>	<i>EPROM</i>
<i>Elektrik ile programlanabilir ışıkla silinebilir sadece okunabilir bellek</i>	<i>Electrical Programmable ROM</i>
<i>Erişim Zamanı</i>	<i>Access time</i>
<i>Eşlik biti</i>	<i>Parity Bit</i>
<i>Eşzamanlama</i>	<i>Synchronization</i>
<i>Etkin değer, AC voltmetrenin ölçtüğü değer</i>	<i>RMS, Root Mean Square</i>
<i>Etkin olmayan duruma alma</i>	<i>Disabled</i>

**F**

<i>Fabrika Maskeli</i>	<i>Factory-masked</i>
------------------------	-----------------------

**G**

<i>Geliştirme kiti</i>	<i>Development Kit</i>
<i>Gerçek mod</i>	<i>Real-Mode</i>
<i>Gerçek Zaman</i>	<i>Real-Time</i>
<i>Geri dallanma</i>	<i>Jump backward</i>

<i>Giriş</i>	<i>Input</i>
<b><i>H</i></b>	
<i>Hafıza</i>	<i>Memory</i>
<i>Hafıza Adres Alanı</i>	<i>Memory Address Space</i>
<i>Hafıza Dizisi</i>	<i>Memory Array</i>
<i>Hafıza Haritası</i>	<i>Memory Map</i>
<i>Hafıza Hücresi</i>	<i>Memory Cell</i>
<i>Hafıza Kod Çözücü</i>	<i>Memory Decoder</i>
<i>Hafıza Yönetimi</i>	<i>Memory Management</i>
<i>Hane</i>	<i>Digit</i>
<i>Hedef</i>	<i>Destination</i>
<i>Hedef Adres</i>	<i>Target address</i>
<b><i>İ</i></b>	
<i>İç RAM</i>	<i>Internal RAM</i>
<i>İçinde, Gömülü</i>	<i>Embedded</i>
<i>İki Seviyeli Kod Çözücü</i>	<i>Two-level Decoding</i>
<i>İki Yönlü</i>	<i>Bidirectional</i>
<i>İkili</i>	<i>Binary</i>
<i>İkilik Kodlanmış onlu, İKO</i>	<i>Binary Coded Decimal, BCD</i>
<i>İKO, ikilik kodlanmış onlu</i>	<i>BCD, Binary Coded Decimal</i>
<i>İleri dallanma</i>	<i>Jump forward</i>
<i>iletişim hızı</i>	<i>Baud Rate</i>
<i>İndirgenmiş komut kümesi kodu</i>	<i>Reduced Instruction Set Code</i>
<i>İndirgenmiş komut kümesi kodu</i>	<i>RISC</i>
<i>İş hatlı</i>	<i>Pipelined</i>
<i>İşaretçi, gösterici</i>	<i>Pointer</i>
<i>İşlem Kodu</i>	<i>Operation Code</i>
<i>İşlem kodu</i>	<i>Op-code</i>
<i>İşlem Verisi</i>	<i>Operand</i>
<i>İşlemci</i>	<i>Processor</i>
<i>İvedi adresleme</i>	<i>Immediate addressing</i>

*İzinleme**Enable**izinlendiğinde**Enabled***K***Kapı**Gate**Karmaşık komut kümesi kodu**Complex Instruction Set Code, CISC**Karmaşık komut kümesi kodu**CISC, Complex Instruction Set Code**Kayan-Nokta Birimi**Floating-Point Unit, FPU**Kaynak**Source**Kaynak Dosya**Source File**Kelime**Word**Kelime Uzunluğu**Word Length**Kenar**Edge**Kenar Tetiklemeli**Edge triggered**Kesme**Interrupt**Kesme Hizmet Programı**Interrupt Service Routine**Kesme İsteği**Interrupt Request**Kısmi Kod Çözümü**Partial Decoding**Kod Çözücü**Decoder**Komut**Instruction**Komut Kümesi**Instruction set**Komut Okuma**Fetch**Komut Saklayıcısı**Instruction Register, IE**Komut Sırası**Instruction Queue**Komut Yürütme**Execute**Kontrol Kelimesi**Control Word**Kontrol Yolu**Control Bus**Korumalı Mod**Protected Mode**Koşulsuz dallanma komutları**Unconditional jump instructions**Kurmak , 1 yapmak**Set***L***Lojik Kelime**Logical Word*

*Lojik işlemler**Logical Operations**M**Makine Çevrimi**Machine Cycle**Malzeme Listesi**Bill Of Material, BOM**Merkezi İşlem Birimi, MİB**Central Processing Unit, CPU**Mikrodenetleyici**Microcontroller**Mikroişlemci**Microprocessor**Mimari**Architecture**Mutlak adresleme**Absolute addressing**Mutlak bağlanma**Absolute Jump**N**Negatif Geçiş**Negative Transition**Negatif Saat Darbesi**Negative Clock Pulse**O**Okuma**Read**Ortak Anotlu**Common ANOT**Ortak Giriş/Çıkış**Common I/O**Ortak Katotlu**Common KATOT**Ortak Yol**Common Bus**Ö**Önbellek**Cache**Öncelik**Priority**Öteleme**Shift**P**Pozitif Geçiş**Positive Transition**Pozitif Saat Darbesi**Positive Clock Pulse**Program belleği**Code memory**Program denetim komutları**Program Control Instructions**Program Geliştirme Ortamı**Program Development Environment**Programlanabilir sadece okunabilen bellek**Programmable Read Only Memory, PROM*

*Programlanabilir sadece okunabilen bellek PROM*

## R

*Rasgele Erişimli Random Access*

## S

*Saat Clock*  
*Saklayıcı Register*  
*Saklayıcı Kümesi Register Set Bank*  
*Salt Okunabilir Hafıza Read Only Memory, ROM*  
*Sanal Virtual*  
*Sanal bellek Virtual Memory*  
*Satır Kod Çözücü Row Decoder*  
*Saydam Transparent*  
*Sayfalama Paging*  
*Sayıcı Counter*  
*Sayma Count*  
*Seviye Level*  
*Seviye Tetiklemeli Level Triggered*  
*Sıfırlamak, başlangıç konumuna getirme Reset*  
*Sıralı adresleme Indexed addressing*  
*Sistem Yolu System Bus*  
*Sütün Kod Çözücü Column Decoder*

## T

*Tam Kod Çözücü Full Decoding*  
*Tampon Buffer*  
*Taşma Overflow*  
*Tek Yönlü Unidirectional*  
*Tekrar Yerleştirilebilir Relocatable*  
*Tetikleme Trigger*  
*Toplam ile denetleme Check Sum*  
*Toplama Addition*



<i>Tümdevre</i>	<i>Chip</i>
<i>Tümdevre Üzeri</i>	<i>On-Chip</i>
<i>Tümleyen</i>	<i>Complement</i>

**U**

<i>Uç</i>	<i>Pin</i>
<i>Universal Asenkron Reciver</i>	<i>UART</i>
<i>Transmitter</i>	
<i>Universal Senkron Asenkron Reciver</i>	<i>USART</i>
<i>Transmitter</i>	
<i>Uzun adresleme</i>	<i>Long addressing</i>

**Ü**

<i>Üretilen firma tarafından programlanmış sadece okunabilir bellek</i>	<i>Mask-ROM</i>
---	-----------------

**V**

<i>Veri</i>	<i>Data</i>
<i>Veri Akışı</i>	<i>Data Flow</i>
<i>Veri aktarma komutları</i>	<i>Data Transfer Instructions</i>
<i>Veri belleği</i>	<i>Data memory</i>
<i>Veri İşleme</i>	<i>Data Processing</i>
<i>Veri Toplama</i>	<i>Data Acquisition</i>
<i>Veri Yolu</i>	<i>Data Bus</i>
<i>Veri Yolu</i>	<i>Data Bus</i>
<i>Verici</i>	<i>Transmitter</i>
<i>Vermek, iletmet</i>	<i>Transmit</i>

**Y**

<i>Yazaç</i>	<i>Register</i>
<i>Yazaç adresleme</i>	<i>Register addressing</i>
<i>Yazma</i>	<i>Write</i>
<i>Yer</i>	<i>Location</i>
<i>Yığın</i>	<i>Stack</i>
<i>Yığın İşaretçisi, Göstericisi</i>	<i>Stack Pointer, SP</i>

<i>Yol</i>	<i>Bus</i>
<i>Yol Çevrimi</i>	<i>Bus cycle</i>
<i>Yuva</i>	<i>Slot</i>
<i>Yükselen Kenar</i>	<i>Rising Edge</i>
<i>Yürütme</i>	<i>Execution</i>

**Z**

<i>Zaman geciktirme döngüsü</i>	<i>Time delay loop</i>
<i>Zamanlama</i>	<i>Timing</i>
<i>Zamanlayıcı</i>	<i>Timer</i>
<i>Zıplama</i>	<i>Debouncing</i>