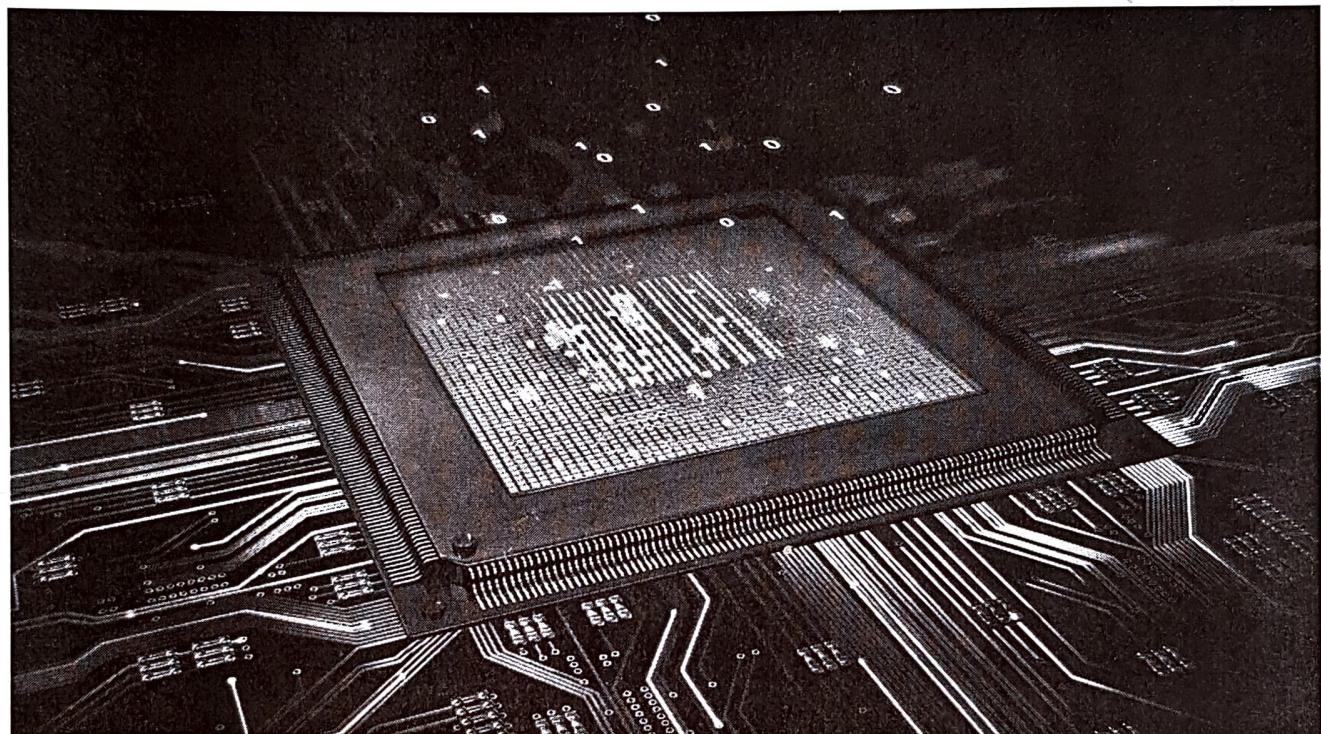


T.C.  
SAKARYA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ  
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

**EEM 437 MİKROİŞLEMÇİLER DERS NOTLARI**



SAY

(2015-2016)

## **Dersin amacı:**

Temel mikroişlemci ve mikrodenetleyici mimari yapısının, sayısal ölçme/kontrol/kumanda işlemlerinin gerçekleştirilmesi için gerekli olan çevre birimlerine (ADC, DAC, PWM, EEPROM, SPI, vb.) sahip gelişmiş bir mikrodenetleyici mimarisinin öğrenilmesi. Mimariye dayalı adresleme yöntemlerini kullanarak makine dilinde programlama yeteneğinin geliştirilmesi, kesme kaynaklarının kullanılabilmesi, problem çözüm algoritmalarının geliştirilmesi yeteneklerinin kazandırılması, disiplinler arası ilişki içeren ilgili senaryoya göre karmaşık mühendislik problemlerin çözümünde donanımsal ve yazılımsal tasarımların/çözümlerin üretilmesidir.

## **Ders sonunda hedeflenen öğrenme çıktıları:**

1. Mikroişlemci/Mikrodenetleyici (intel-8051) mimarisinin öğrenilmesi
2. Dahili/Harici Hafıza ve RAM yapılarının öğrenilmesi
3. Adresleme modları ve makine dilinin öğrenilmesi
4. Gelişmiş mikrodenetleyici mimarisi ve çevre birimlerinin öğrenilmesi(ADUC841)
5. Kesme kaynaklarının öğrenilmesi ve kullanabilme yeteneğinin geliştirilmesi
6. Disiplinler arası ilişki içeren karmaşık mühendislik problemleri için çözüm algoritmalarını geliştirme ve makine dilinde programlama yeteneğinin elde edilmesi
7. Asenkron/Senkron seri ve SPI haberleşme yeteneklerinin geliştirilmesi
8. Yazılım ve Donanım güvenliği sağlayabilme yeteneğinin geliştirilmesi



# İÇİNDEKİLER

BÖLÜM 1 .....	6
MİKROİŞLEMCI VE MİKRODENETLEYİCİ TARİHÇESİ.....	6
1. Genel Tarihçe .....	6
1.2. Mikroişlemci ve Mikrodenetleyici karşılaştırması.....	9
1.3 Mikrodenetleyicilerin Kullanım alanları.....	10
1.4. Sayı Gösterimleri (Number Representation).....	14
1.4.1 İşaretli Sayı Kavramı .....	15
BÖLÜM 2 .....	17
MİKROİŞLEMÇİLER VE MERKEZİ İŞLEM BİRİM (CPU) MİMARİSİ.....	17
2.1 Ortak Yol (Bus).....	17
2.2 CPU Birimleri ve Özellikleri .....	18
2.3 Mikroişlemci Programlama.....	21
BÖLÜM 3. 8051 MİKROİŞLEMÇİSİ VE ADUC841 MİKRODENETLEYİCİSİ.....	27
3.1. Standart 8051 .....	27
3.2. Aduc841 Mikrodenetleyicisi.....	30
3.3 8051 Mikrodenetleyici Mimarisi ve Çevre Birimleri: .....	32
3.3.1. Yonga (Chip) Üzerindeki Bazı Çevre Birimleri .....	32
3.3.2. Hafıza Yapısı.....	34
BÖLÜM 4. ADRESLEME MODLARI.....	41
4.1. İvedi Adresleme (Immediate Adressing) .....	41
4.2. Doğrudan Adresleme (Direct Adressing).....	43
4.3. Saklayıcı Adresleme (Register Adressing) .....	45
4.4. Saklayıcı - Dolaylı Adresleme (Register- Indirect Adressing) .....	46
4.5. Saklayıcı - İndisli Adresleme (Register Indexed Adressing) .....	47
BÖLÜM 4 8051 KOMUT KÜMESİ .....	51
4.1 Veri Transfer Komutları.....	51
4.1.1 Dahili Veri Hafıza Transfer Komutları .....	51
4.1.2 Harici Veri Hafıza Transfer Komutları .....	52
4.1.3 Bakma (Look-up) Tabloları .....	53
4.2. Veri İşleme Komutları.....	54
4.2.1. Lojik Komutlar .....	54
4.2.2 Aritmetik Komutlar Tablosu .....	56
4.2. Bit Transfer Komutları.....	65

4.3. Program Akışı Kontrol Komutları .....	67
4.3.1 Durumdan Bağımsız Dallanma Komutları.....	67
4.3.2 Duruma bağımlı dallanma komutları .....	71
BÖLÜM 5 ALT PROGRAM.....	76
5.1. Duruma Bağlı Dallanma ve Alt Program.....	76
5.2. Alt Programdan Ana Programa Dönüş Adresine Donanım Tarafından Hesaplanması .....	77
5.3. İç -İçe Alt Program Yapısı .....	78
5.4. Program Çökмелерinin Olası Nedenleri.....	79
5.5. Yığın İşaretçisi (Stack Pointer, SP).....	81
BÖLÜM 6 Hafıza Yapısı .....	83
6.1. Üst RAM Bölgesi.....	83
BÖLÜM 6 8051 TÜMDEVRE UÇ FONKSİYONLARI .....	96
8051 Tümdevre Uç Fonksiyonları .....	96
6.1. Besleme Uçları .....	96
6.2. RST (RESET)- Power on Reset .....	96
6.3. Osilatör Girişleri (XTAL1 ,XTAL2).....	97
6.4. Harici Hafıza Erişim Pinleri.....	98
6.5. Giriş/Çıkış Pinleri .....	98
6.5.1 Port 1 (P1) .....	98
6.5.2 Port 0 (P0) .....	101
6.5.3. Port 2 (P2) .....	102
6.5.4. Port 3 (P3) .....	103
6.6. Aduc841 Tümdevre Uç Fonksiyonları.....	104
BÖLÜM 7 ZAMANLAYICI/SAYICI YAPISI:.....	106
7.1. Sistem Saat üreteci ve Makine Çevrimi .....	106
7.2 Aduc841 Makine Çevrimi.....	107
7.3. Zamanlayıcı/Sayıci.....	108
7.3.1 Mod-0: 13-bit zamanlayıcı/sayıci .....	112
7.3.2 Mod-1: 16-bit zamanlayıcı/sayıci .....	113
7.3.4. Mod-2: 8-bit otomatik yüklemeli zamanlayıcı/sayıci .....	113
7.3.4. Mod-3: Bağımsız çalışma modu .....	113
7.4. Timer 2 .....	114
7.4.1 T2CON Saklayıcısı .....	114
BÖLÜM 8 KESMELER (INTERRUPTS) .....	116

8.1. Kesme Kaynakları ve Öncelik Seviyesi .....	117
8.2. Reset Kesmesi .....	119
8.3. IE Saklayıcısı.....	120
8.4. IP Saklayıcısı.....	120
8.5. TCON Saklayıcısı .....	121
8.6. ADUC841 Kesme Yapısı .....	125
8.6.1. IIEIP2 Saklayıcısı.....	126
BÖLÜM 9 ADUC841 ÇEVRE BİRİMLERİ .....	128
9.1. Seri Kanal Haberleşmesi .....	128
9.1.1. SCON Saklayıcısı .....	129
9.2. CFG841 Saklayıcısı .....	130
9.3. Analog-Sayısal Dönüştürücü Birimi (Analog-Digital Converter, ADC).....	131
9.3.1. ADCCON1 Saklayıcısı .....	133
9.3.2. ADCCON2 Saklayıcısı .....	134
9.3.3. ADCCON3 Saklayıcısı .....	134
9.4. Sayısal - Analog Dönüştürücü Birimi (Digital - Analog Converter, DAC).....	135
9.4.1. DACCON Saklayıcısı .....	138
9.5. Darbe-Genişlik Modülasyon (Pulse-Width Modulator, PWM) Birimi.....	139
9.5.1. PWMCON Saklayıcısı .....	140
9.5.2. PWM Çalışma Modları .....	141
9.6. Kısır-Döngü Zamanlayıcısı (Watchdog Timer) .....	144
9.7. Power-On Reset (POR) .....	145
9.8. Besleme Gerilimi İzleyicisi (Power Supply Monitor, PSM) .....	146
9.9. Seri Çevresel Arayüz Haberleşmesi (SPI): .....	146
Kaynaklar .....	154

# BÖLÜM 1

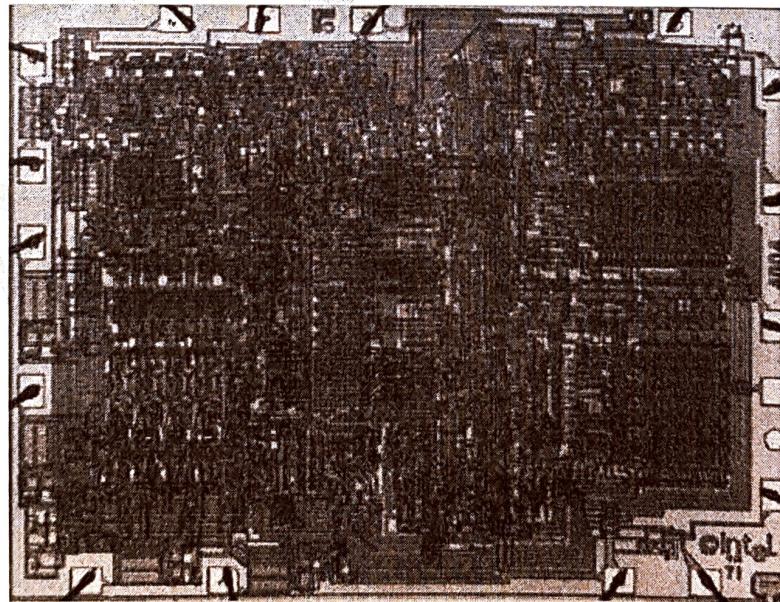
## MİKROİŞLEMCI VE MİKRODENETLEYİCİ TARİHÇESİ

### 1. Genel Tarihçe:

1940'lı yıllarda ilk transistor'ün keşfi ile elektronik endüstrisinde yeni bir çağ başlamıştır. Bu keşiften birkaç yıl sonra transistor'ün ticarileşmesi ile elektronik devreler yarı iletkenler ile üretilmeye başlanmış ve devre üzerinden geçen akımın düşük miktardaki başka bir akım geçisi ile kontrol edilebildiği tümdevreler ortaya çıkmıştır. Buna ek olarak, bilgisayar mimarisinde paralel gelişimlerin doğması ve tümdevrelerin seri üretime geçmesini sağlayacak olan "mikroişlemci" (microprocessor) veya "chip işlemci (Computer- on- a Chip)" teknolojisi ilk defa 1971 yılında bilinmeyen küçük bir firma olan "Intel" tarafından 4-bitlik "4004" adlı mikroişlemcisi ile hayat bulmuştur.



Şekil 1 İlk defa patent alınan tüm devre



Şekil 2 Intel'in piyasaya sunduğu ilk işlemci "4004"

Aynı anda yürütebildiği bit sayısına bakılarak güçlü olup olmadığı anlaşılan işlemcilere daha sonra kısa bir süreç için sınırlı sayıda işlem yapabilen 8 bitlik 8008 işlemcisi eklenmiştir. 1974 yılında Intel 8080 adlı işlemcisini, hemen ardından da önceki işlemci ile pek farkı olamayan Motorola 6800 adlı işlemcisini piyasaya sürmüştürlerdir. Birbirleri arasında küçük farklılıklar olan iki işlemci daha piyasaya sürülmüştür. Bunlar, MOS Technology firması tarafından üretilen 6502 ve Zilog firması tarafından üretilen Z-80 işlemcileridir.

Mikroişlemciler	Üretim Yılı	Kaydedici Sayısı	Kaydedici Büyüklüğü	Veri Yolu Genişliği	Adres Yolu Genişliği	Adresleme Kapasitesi
<b>Intel 8085</b>	<b>1974</b>	<b>8 2</b>	<b>8 16</b>	<b>8</b>	<b>16</b>	<b>64K</b>
<b>Motorola 6800</b>	<b>1975</b>	<b>3 3</b>	<b>8 16</b>	<b>8</b>	<b>16</b>	<b>64K</b>
<b>Zilog Z-80</b>	<b>1975</b>	<b>17 1 4</b>	<b>8 7 16</b>	<b>8</b>	<b>16</b>	<b>64K</b>
<b>Mostek 6502</b>	<b>1976</b>	<b>1 16</b>	<b>8 16</b>	<b>8</b>	<b>16</b>	<b>64K</b>

Şekil 3 8 bitlik popüler mikroişlemcilerin teknik özellikleri

8 bitlik 8080 ve Z 80 mikroişlemcilerinde hesaplama yapmak maksadıyla bol miktarda kaydedici vardır. Bundan dolayı bu işlemcilere kaydediciye dayalı işlemciler denilmektedir.

Diger 8 bitlik işlemciler 6800 ve 6502, anlaşılır komutlar ve daha fazla adresleme modu kullanmaları, kaydedicilerinin fazla olmamasından dolayı veri manevrasında sık sık belleği kullanmalarından dolayı belleğe dayalı işlemciler olarak anılırlar. Bu gruplar birbirlerinin bellek ve G/C (Giriş/Çıkış) yongalarını kullanabilmektedirler.

Bulunduğu sistemin belirli bir düzen içerisinde çalışmasını sağlayan, aritmetik ve lojik işlem yapabilme yeteneğine sahip olan **mikroişlemciler** geçen zaman içerisinde sahip oldukları özellikler açısından olağanüstü bir geliştirme göstererek modern yaşamın vazgeçilmez bir parçası haline gelmiştir [1].

Başlangıçta endüstriyel uygulamalarda da kullanılan mikroişlemciler, en basit gerçek zaman uygulamalarında bile ihtiyaç duyulan hafıza, giriş/çıkış birimleri, vb. yapıları mimarilerinde dahili olarak sahip değildirler. Bu gibi çevre birimleri harici olarak ilave edilerek mikroişlemci tabanlı uygulamalar gerçekleştirilebilir. Ancak gerçek zaman endüstriyel uygulamalarda gerekli olan çevre birimlerinin karmaşıklılığı ve sayısı zamanla artmıştır. Sonuç olarak mikroişlemci tabanlı sistemlerin tasarımını ve gerçekleştirilmesi oldukça karmaşık, zor ve maliyetli olduğundan günümüzde endüstriyel uygulamalarda “Mikrodenetleyici”ler kullanılır .

Mikrodenetleyiciler; mimarisinde bir mikroişlemciye ilave olarak bir çok çevre birimini de dahili olarak bulunduran, tek bir tümdevre üzerinde üretilen, endüstriyel uygulamalar için geliştirilmiş özel bir mikro bilgisayar olarak değerlendirilebilir. Ölçme, kumanda, kontrol, vb. endüstriyel uygulamalarda ihtiyaç duyulan bir çok çevre birimini tek bir tümdevre üzerinde bulunduran mikrodenetleyiciler mikroişlemcilere oranla çok daha basit, ucuz ve kolay çözümler sunar. Günümüzde farklı mimari özelliklere sahip mikrodenetleyiciler çok sayıda firma tarafından üretilmektedir.

8051 ailesinin yeni nesil üyeleri endüstriyel uygulamalarda karşılaşılan ihtiyaçlara cevap verebilmek amacıyla temel mimari özelliklere ilave olarak, analog-sayısal dönüştürücü (ADC), sayısal-analog dönüştürücü (DAC), darbe genişlik modülasyonu (PWM), ilave hafıza birimleri, yüksek çalışma frekansları, kısır-döngü zamanlayıcısı (Watchdog Timer) gibi çok çeşitli özelliklere sahip olarak üretilmektedir.

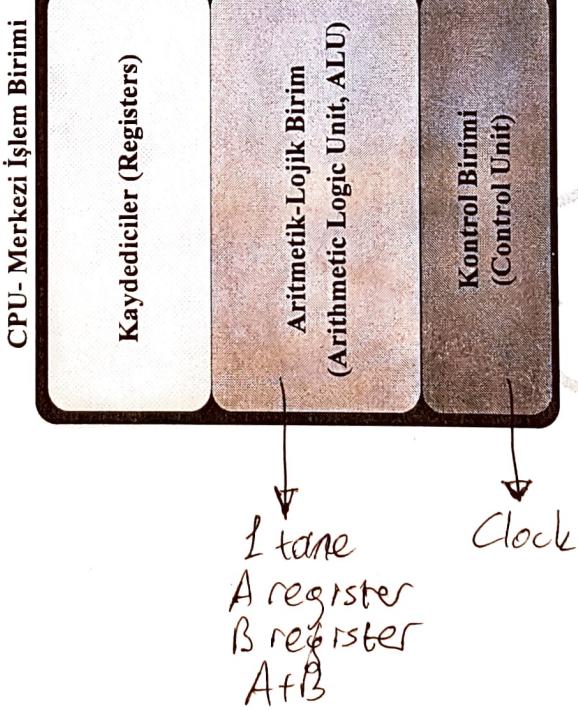
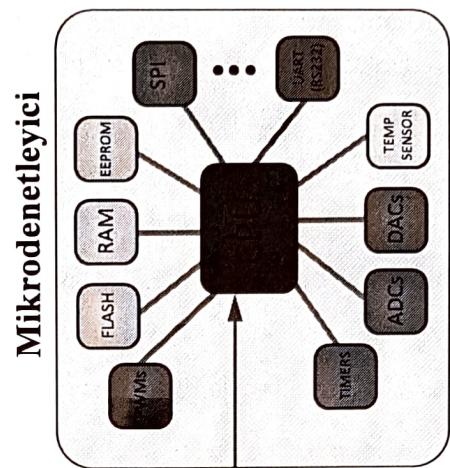
Aduc841 mikrodenetleyicisi Analog Device firması tarafından üretilen ve günümüzde bir çok endüstriyel uygulamada tercih edilen, ADC, DAC, PWM, I2C ve SPI haberleşme birimleri, besleme gerilimi izleme, vb. bir çok gelişmiş özelliğe sahip, 8051 tabanlı, 8-bit'lik bir mikrodenetleyicidir.

Günümüzde 8051 ailesi dışında çeşitli firmalar tarafından farklı mimari yapıya sahip mikrodenetleyici aileleri (PIC, Motorola, ..) mevcuttur. Ancak bu mikrodenetleyiciler tek bir firma tarafından üretildiginden, ürün çeşitliliği ve teknik destek açısından dezavantaj oluşturabilmektedir. Diğer yandan başta Intel olmak üzere çok sayıda firma 8051 tabanlı mikrodenetleyiciler üretmektedir. Bunun neticesinde ürün çeşitliliği çok olduğundan yapılacak olan uygulamaya yönelik 8051 tabanlı mikrodenetleyici temini daha kolay olacaktır. Ayrıca, yine bir çok firma 8051 ailesi için teknik doküman, yazılım ve donanım desteği sunmaktadır.

Bu bölümün amacı mikroişlemci ve mikrodenetleyici kavramını açıklamak ve bir mikrodenetleyicinin karakteristik özelliklerini incelemektir. Ders notlarının geri kalan 8051 işlemcisi ve ADUC841 mikrodenetleyici mimarileri ve çevre birimleri hakkında detaylı bilgiler verilecektir.

## 1.2. Mikroişlemci ve Mikrodenetleyici karşılaştırması:

Mikroişlemci (Microprocessor)	Mikrodenetleyici (Microcontroller)
<p><b>Tanım:</b> Mikroişlemci, (Microprocessor, bazen kısaltma olarak <math>\mu</math>P kullanılır) ana işlem biriminin (CPU) fonksiyonlarını tek bir yarı iletken tümleşik devrede (IC) veya CPU-on-a-single-chip (CPU-tek-çip) birleştiren programlanabilir bir sayısal elektronik bileşendir.</p>	<p><b>Tanım:</b> Mikrodenetleyiciler (Microcontrollers), mimarisinde bir CPU'ya ilave olarak bir çok çevre birimini (ROM, RAM, giriş-çıkış birimleri (pin/port), vb.) de dahili olarak bulunduran, tek bir tümdeve üzerinde üretilen, özel bir mikro-bilgisayar olarak değerlendirilebilir.</p>



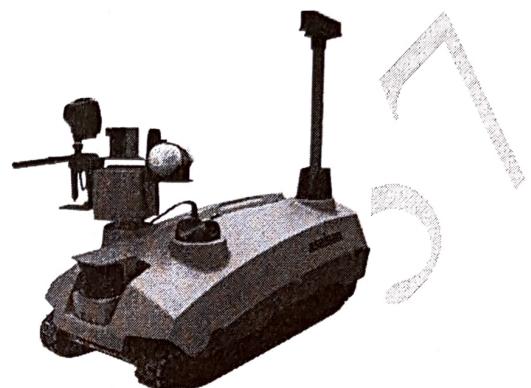
### **1.3 Mikrodenetleyicilerin Kullanım alanları:**

Mikrodenetleyiciler otomotiv, tıp, endüstri, askeri ve uzay teknolojileri gibi birçok alanda yoğun bir şekilde kullanılmaktadır. Aşağıda farklı endüstriyel alanlar için mikrodenetleyici tabanlı bazı örnek ürünler-cihazlar gösterilmiştir.

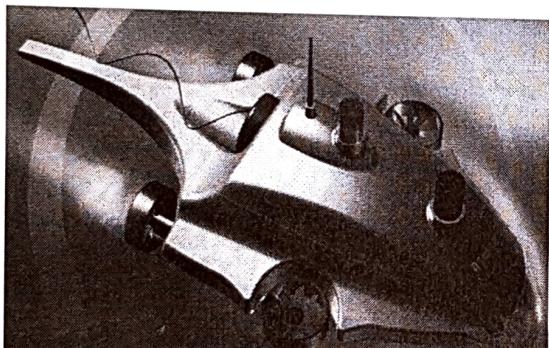
✓ Savunma sanayi,



Uzaktan kontrol edilebilen atış sistemi



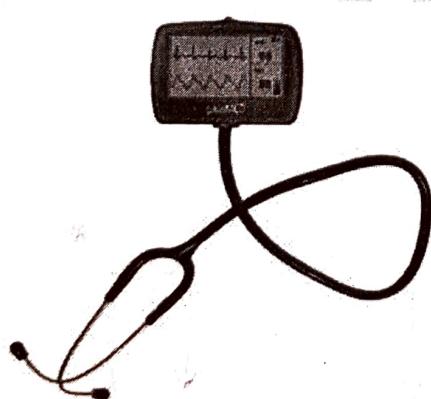
İnsansız kara aracı(Aselsan)



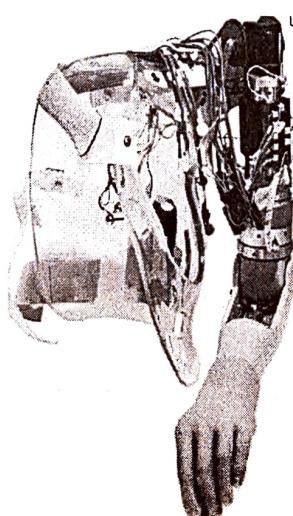
İnsansız deniz aracı (Aselsan)  
Medikal ürünler,



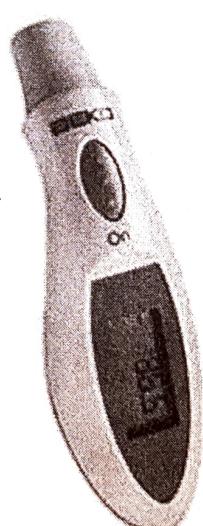
İnsansız hava aracı, ANKA (TAİ)



Sayısal tansiyon ölçme aleti

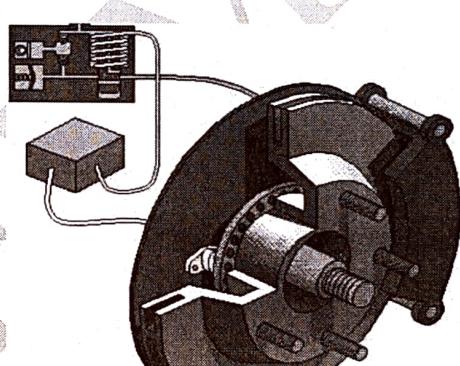
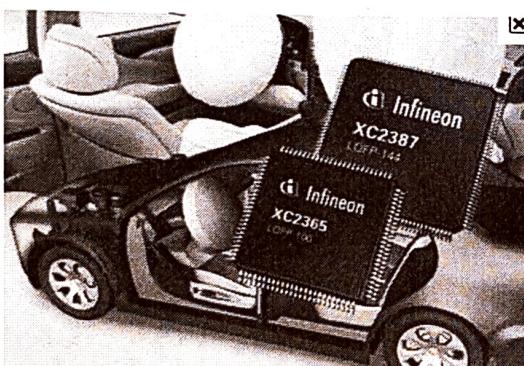
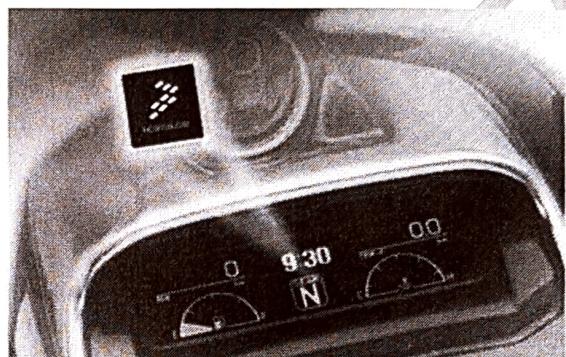
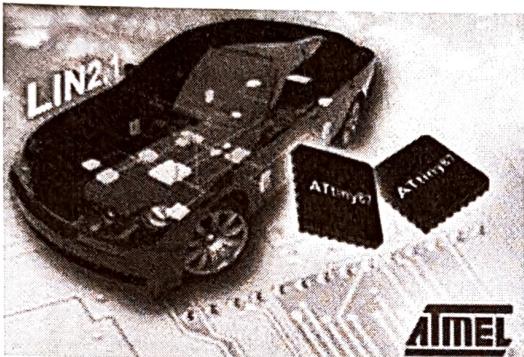
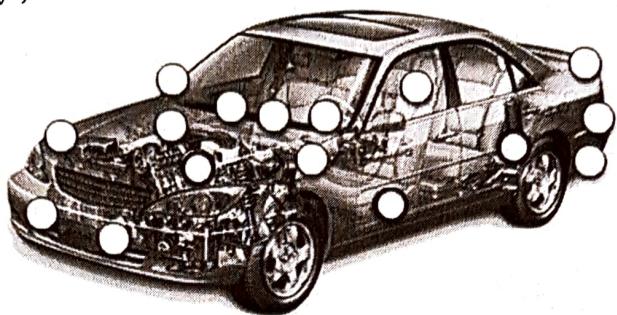


Protez  
*Kol Gögüsler*

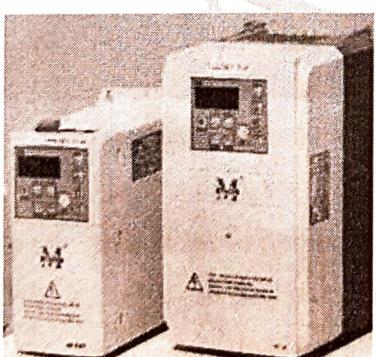


Sayısal ateş ölçer

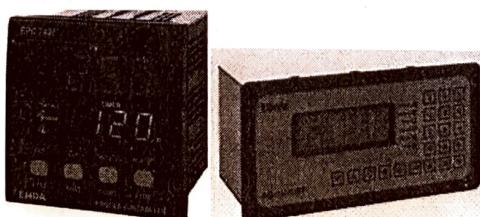
✓ Otomotiv sanayi,



✓ Endüstriyel uygulamalar,



AC-DC Motor sürücülerini  
kontrol rölesi

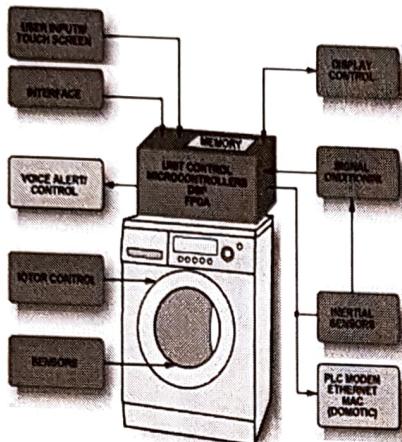


Endüstriyel kontrolörler

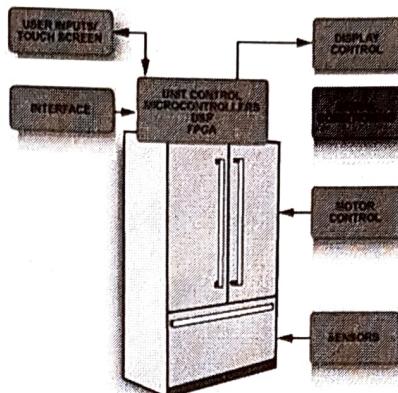


Reaktif güç

✓ Beyaz eşya,



Çamaşır makinesi

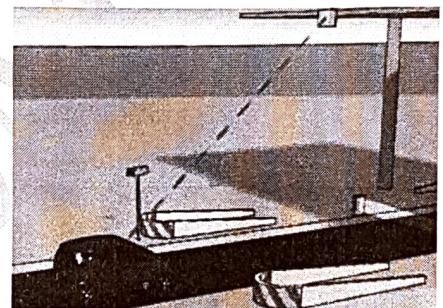


Buzdolabı

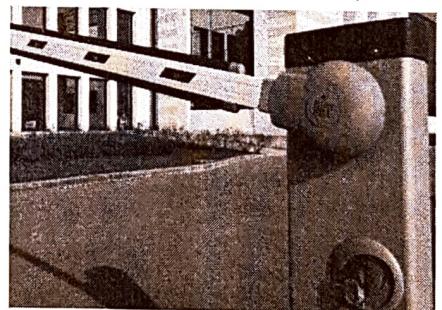
✓ Mikrodenetleyici tabanlı diğer uygulamalar,



Otomat



OGS (Otomatik Geçiş Sistemi)



Otomatik kapı-bariyer sistemleri

Şekil 4 Mikrodenetleyicilerin günümüzdeki uygulama alınlardan örnekler

### Mikrodenetleyicilerin günümüzde kullanılan uygulama alanları aşağıda sıralanmıştır:

- Buzdolabı, çamaşır makinesi, bulaşık makinesi vb. beyaz eşyalarda değişik parametrelerin ölçülmesi ve süreç denetiminde,
- Otomobillerde ve diğer araçlarda algılama, denetim ve göstergelerde,
- Elektrik motoru içeren sistemlerde, motor hızı, açı kontrol uygulamalarda,
- Fare, klavye, oyun denetçileri vb. Bilgisayar çevre birimlerinde, monitörlerde...
- Elektronik ajanda, cep bilgisayarları (PDA)
- Endüstriyel ağlarda düşük maliyetli denetleyici bölgesi ağı (CAN),
- Genel amaçlı seri yol (USB) algılayıcı-güncelleyici arabirimini,
- Güvenlik Çevre Birimleri: Kızılıötesi uzaktan (IR) algılama ve iletişimini,
- Elektronik Lamba Balastı
- Anahtarsız uzaktan Kapı açılması: RKE
- Televizyonlarda uzaktan kumanda, ekran üzerinde ayar ve gösterge birimlerinde
- Elektronik saatlerde,
- Cep telefonları içinde ses, görüntü giriş birimlerinde

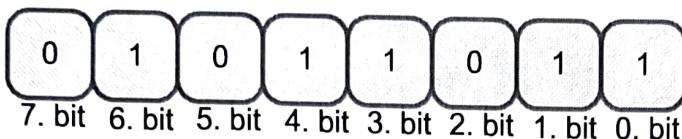
### Elektronik Sistemlerde Kullanılan Mikrodenetleyici Uygulamaları:

- Paralel veri giriş/çıkış birimlerinde
- Seri veri giriş/çıkış birimlerinde
- Sayıcı/ zamanlayıcı denetim ve ölçümünde
- Endüstriyel ölçme ve kontrol birimlerinde gösterge, çıkış birimi veya klavye, giriş birimi
- Algılayıcıların çıkış işaretinin dönüştürülmesinde
- Akım, gerilim, direnç ölçü aletlerinde
- Sıcaklık ölçümü ve denetiminde
- Sayısal PID denetleyici uygulamalarında
- Bir internet denetleyicisine bağlantı (TCP/IP)
- Yerleşik internet uygulamaları
- Genel amaçlı seri yol (USB)
- Denetleyici bölgesi ağı (CAN)
- Darbe genişliği modülasyonu (PWM)
- Bölgesel iç bağlantı ağı (LIN)
- Radyo frekans, telli ve telsiz çalışma (RKE)

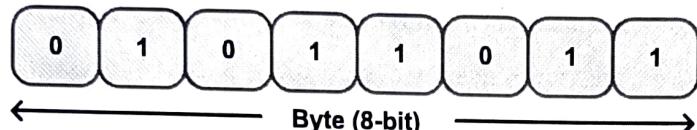
#### 1.4. Sayı Gösterimleri (Number Representation)

Sayıları gösterirken onların boyutlarını (hane sayısı bakımından) göz önünde bulundururuz.

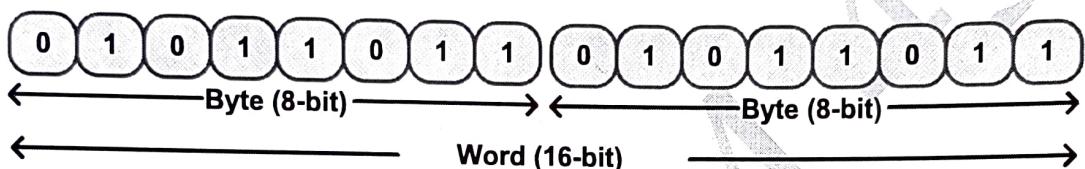
- ✓ **Bit:** Tek bir binary digit (hane)



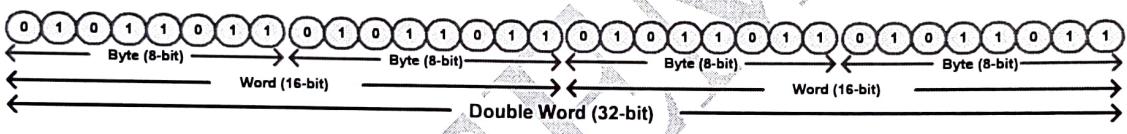
- ✓ **Byte:** 8 tane binary digit



- ✓ **Word:** 16 binary digit



- ✓ **Double Word:** 32 binary digit



- ✓ **Heksadesimal (Hexadecimal):** Heksadesimal , 16 tabanlı sayı sistemidir.

“Hex” bilgisayar belleğindeki 8-bit'lik byte'ları göstermek için kullanılan bir kestirme yoldur. Bu sayı sistemine “16 tabanlı sayı sistemi” denilmesinin nedeni, 16 tane sembolden oluşmasıdır.

Sembollerden 10 tanesi rakamlarla (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), geri kalan 6 tanesi harflerle (A, B, C, D, E, F) temsil edilir.

$0_{hex} = 0_{dec} = 0_{oct}$	0 0 0 0
$1_{hex} = 1_{dec} = 1_{oct}$	0 0 0 1
$2_{hex} = 2_{dec} = 2_{oct}$	0 0 1 0
$3_{hex} = 3_{dec} = 3_{oct}$	0 0 1 1
$4_{hex} = 4_{dec} = 4_{oct}$	0 1 0 0
$5_{hex} = 5_{dec} = 5_{oct}$	0 1 0 1
$6_{hex} = 6_{dec} = 6_{oct}$	0 1 1 0
$7_{hex} = 7_{dec} = 7_{oct}$	0 1 1 1
$8_{hex} = 8_{dec} = 10_{oct}$	1 0 0 0
$9_{hex} = 9_{dec} = 11_{oct}$	1 0 0 1
$A_{hex} = 10_{dec} = 12_{oct}$	1 0 1 0
$B_{hex} = 11_{dec} = 13_{oct}$	1 0 1 1
$C_{hex} = 12_{dec} = 14_{oct}$	1 1 0 0
$D_{hex} = 13_{dec} = 15_{oct}$	1 1 0 1
$E_{hex} = 14_{dec} = 16_{oct}$	1 1 1 0
$F_{hex} = 15_{dec} = 17_{oct}$	1 1 1 1

Heksadesimal Dönüşüm Tablosu

- Sayıları gösterirken boyutları ihmali etmemeliyiz.
- 8 bit yani 1 byte ile  $0-255_{10}$  e kadar olan sayıları gösterebiliriz. ( $1111\ 1111_2$  ya da  $FF_{16}$ )
- 16 bit yani word gösterimi ile  $0-65535_{10}$  e kadar olan sayıları gösterebiliriz. (veya  $1111\ 1111-1111\ 1111_2$  ya da  $FFFF_{16}$ )
- Bu sınırlar kullanabileceğimiz sayıları belirler.

#### 1.4.1 İşaretli Sayı Kavramı:

Mikrodenetleyicilerde dolayısıyla mikroişlemcilerde işaretli sayıların da kullanılması gerekmektedir. Mesela tasarlamak istediğimiz pratik bir devre sırasında ( $-5^{\circ}C$ ) gibi bir sıcaklık değeri karşımıza çıkabilir. Bu nedenle mikroişlemcide negatif sayıların da temsil edilmesi ve işlenebilmesi gereklidir. Bu da aslında sadece “0” ve “1” lerle işlem yapan mikroişlemcilerde negatifliğin bir şekilde temsil edilmesi ihtiyacı olduğu anlamına gelir.

Mikroişlemcilerde programlama yapılrken öncelikle işaretli sayılar (negatif işaretli) veya işaretsiz sayılar ile çalışılacağı belirlenmelidir ve program bu mantık kabul edilerek yazılmalıdır. Mikroişlemci hesap yaparken bunu bilmez. Çünkü bu sayıya bir bakış açısından bakıldığında 0 olmasına karşılık -1 de olmasına karşılık gelir. Yoksa mikroişlemci için işaretli sayılarla yapılan işlemin işaretsiz sayılarla yapılan işlemlerden farkı yoktur. Bazı mikroişlemcilerde işaret bitinin değeri bir başka özel bitle belirtilmektedir.

İşaretli sayılarda 7. Bit işaret biti olarak kullanılır.

- 7. Bit  $\rightarrow$  0 ise sayı pozitif
- 7. Bit  $\rightarrow$  1, 1 ise negatiftir.

**Örnek:**

$$83 = 01010011$$

$$-71 = 10111001$$

Şeklinde gösterilir. Negatif sayılar kullanılırken genelde ikinin tümleyenini alma yöntemi kullanılır.

- ✓ İlk bit işaret bitini gösterir,
- ✓  $-n$  sayısını göstermek istediğimizi farz edelim,  $+n$  sayısının bitlerinin tersini alıp 1 eklediğimizde  $-n$  sayısını elde ederiz. Bu yöntem “ikiye tümleme” olarak adlandırılır.

**Örnek:** (-) 109 sayısı için;

$$109_{10} = 01101101_2 \text{ ikiye tümleyeni alınırsa;}$$

$$(-)109 = 10010011 \text{ elde edilir.}$$

- ❖ Neden ikinin tümleyeni alınır?
- ✓ Bu sayede pozitif ve negatif sayıları üzerinde işlem yapmak daha kolaydır. Negatif sayı ile yapılan işlem aslında mikroişlemci için toplama işlemine dönüştürülmüş olur. İşlemci mimarisi toplama işlemi için uygundur.

Örneğin; A-B işlemi için; A+(-B) işlemi yapılmış olur. Bunu bir örnekle açıklayalım;

**Örnek:**

$$83 = 01010011$$

(-) 71 = 10111001 bu iki sayıyı toplarsak  $83 + (-)71$  işlemini yapmış oluruz.

Ve sonuç = (1) 00001100<sub>2</sub> = 12<sub>10</sub> elde edilir.

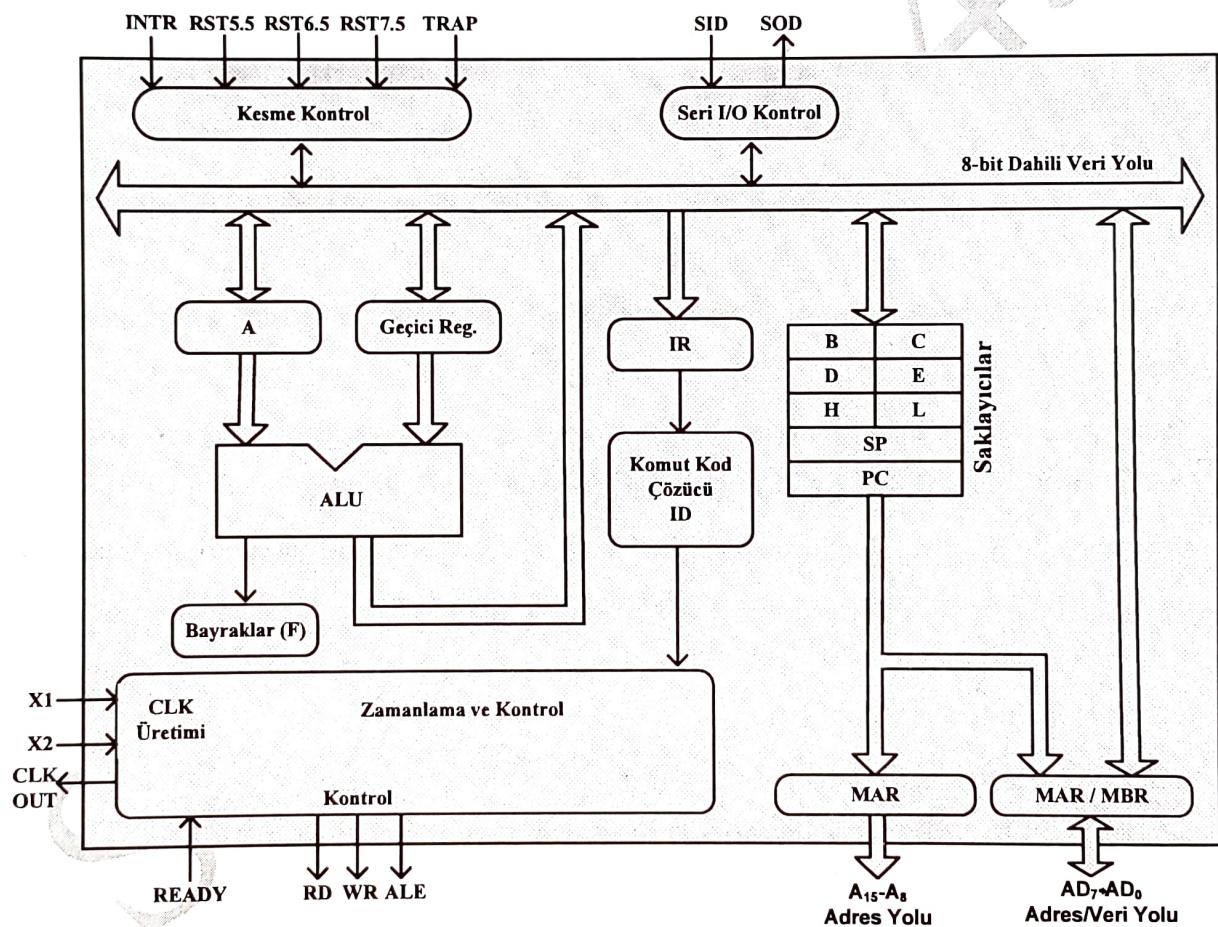
## BÖLÜM 2 MİKROİŞLEMCİLER VE MERKEZİ İŞLEM BİRİM (CPU) MİMARİSİ

Bir işlemcinin donanımsal özelliklerini kavrayabilmek için onun mimari yapısına hâkim olmak gereklidir. İyi bir programcı programladığı işlemcinin mimarisini ve özelliklerini bildiği takdirde karşılaştığı hataları yorumlayabilir ve çözüme kavuşturabilir.

Bu bölümde ilk olarak CPU-(Central Processing Unit) veya Merkezi İşlem birimi (MİB) mimarisi ve özellikleri hakkında genel bilgiler verilecektir.

Merkezi işlem birimine ait genel mimari aşağıda belirtilmiştir.

- Verilen mimari herhangi bir işlemciye ait olmayıp geneldir.



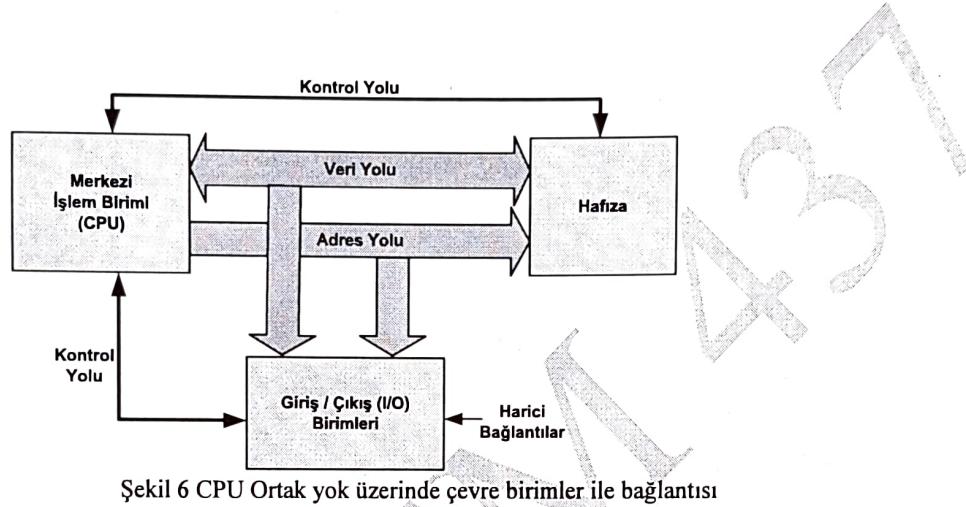
Şekil 5 CPU- Merkezi İşlem Birimi Genel Mimarisi

CPU'nun iç yapısı; aritmetik lojik birim (ALU), program sayacı (program counter-PC), yığın işaretleyicisi (stack pointer-SP ) gibi birimlere ait blok diyagramı Şekil 5'de gösterilmiştir.

### 2.1 Ortak Yol (Bus)

Ortak yol (Bus) aynı amaç için veri taşıyan paralel bağlantıları tanımlar. Bağlantıların sayısı ortak

yolun(hat) genişliğini (4-bit, 8-bit, vb.) başka bir ifade ile aynı anda iletilebilecek bit sayısını belirler. örneğin 8-bit'lik bir CPU'da veri yolundan bir seferde 8-bit uzunlığında veriler taşınabilir. Bu durum aynı zamanda işlemcinin hızını da işaret eder, günümüzde 126-bit işlemciler mevcuttur bunun anlamı ortak yoldan aynı anda taşınan bit uzunluğunun 128-bit olduğunu gösterir. CPU mimarisinde; Adres yolu (Address bus), veri yolu (Data bus) ve lojik kontrol işaretlerinin iletildiği kontrol yolu (control bus) olmak üzere üç ayrı ortak yol (hat) vardır.



**Veri yolu (Data bus):** İşlemci, hafıza ve çevre birimleri arasında veri iletişimini sağlamak için kullanılır.

**Adres yolu (Adress bus):** İşlemcinin program komutlarına ve veri saklama alanlarına erişimi sağlayan hafıza adreslerini, ROM ve RAM hafızalarına göndermek için kullanılır.

**Kontrol yolu (Control Bus):** Kontrol biriminin ürettiği sinyalleri ilgili çevre birimlerine ve ilgili çevre biriminden kontrol birimine gelecek sinyalleri iletmek için kullanılır.

## 2.2 CPU Birimleri ve Özellikleri:

### KAYDEDİCİ VE SAYICI :

Kaydediciler ve sayıcılar gerek mikroişlemci içerisindeki gerekse de mikroişlemci ile diğer devreler arasındaki işlemleri destekleyen devrelerdir. Saklayıcılar CPU' nun ufak birer bilgi depolama birimleridir ve diğer bellek birimleri gibi ikili (binary) hücrelerden (filp, floplardan) oluşturulmuşlardır.

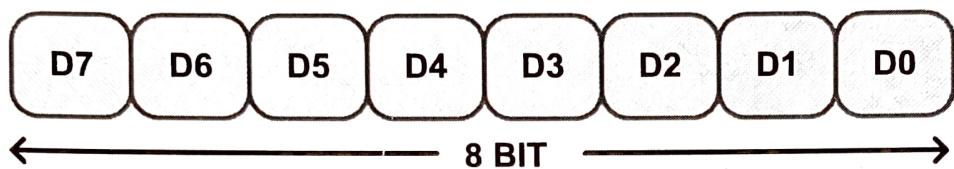
Bazları yazılım kontrolü altındadır ve CPU' nu her bir bilgi alış – verisi için bellek bölümüne başvurmama olanağı sağlarlar. Böylece işlem süresi çok kısa olur. Bazları ise denetim için gerekli bilgileri saklarlar. Temel yapısı, D ve J – K Flip – Flop'lardan oluşur. Sayıcıların görevi ise işlemi yapılacak olan komut verilerin adresini belirlemektir. Temel yapısı, J-K Flip – Flop gruplarıdır.

Flash Bellek → Dogrular lorsular o anda tutturup sonra gelecekte

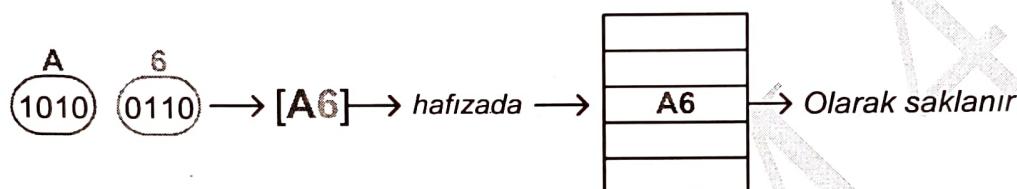
**ACCUMULATOR:** (ACC, A -Akümülatör) CPU'nun ana saklayıcıdır. Özel tanımlanmış kaydedici olup, ALU (Arithmetic-Logic Unit) ile yapılan herhangi bir aritmetik veya lojik işlem sonucunu saklar.

Veri Yolu (Data bus) ile aynı bit uzunluğuna sahiptir. Öyle ki; 8 bitlik mikroişlemci deyiminden o mikroişlemcinin 8 bitlik data bus ve akümülatör özelliklerine sahip olduğu anlaşılır.

Veri Yolu; 4- 8-16-32-64 bitlik olabilir. Mikroişlemcilerde genellikle 8 bitlik veri alınır, verilir.



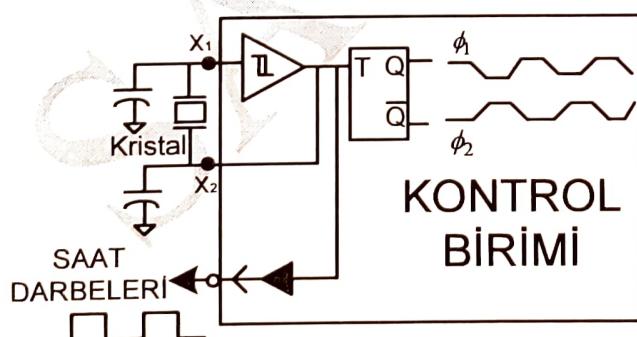
Örnek Saklama: "10100110" verisi heksadesimal yani 16 bitlik tabana çevrilir ise,



**ALU:** (Arithmetic Logic Unit- Aritmetik- Lojik İşlem Ünitesi) Matematiksel (toplama, çıkarma vb.) ve/veya lojik işlemler gerçekleştirilir. ALU'ya doğrudan ulaşılamaz; sadece komutlar ile işlem yaptırılabılır. ALU mikroişlemcinin "beyni" gibi düşünülebilir.

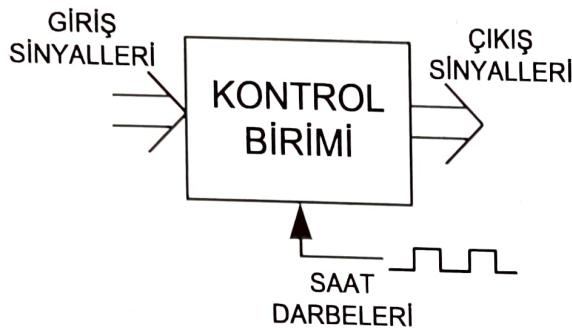
**BUFFER REGISTER:** (Tampon/Önbellek kaydedicisi) Bilginin ALU'ya girmeden önce saklandığı birimdir. Mikroişlemci ile eşit bit uzunluğuna sahiptir.

**CLOCK:** (Saat Darbesi) Kontrol biriminin (control unit) zamanlama darbeleri olarak kullandığı belirli frekansta saat darbeleri üretir. Kontrol birimi  $X_1$  ve  $X_2$  girişlerine bağlanan kristalin frekansı üretilecek saat darbelerinin frekansını belirler.



Saat Darbeleri çok önemlidir çünkü bu darbeler olmazsa mikroişlemci lojik mimari gereği çalışmaz. Yani bu darbeler Mikroişlemcinin "kalp atışları" gibidir.

**CONTROL UNIT:** (Kontrol Birimi) CPU'nun dış ortamla ilişki kurabilmesi için gerekli sinyalleri üretir ve ilgili birimleri kontrol eder. Kontrol birimi mikroişlemcinin "kalbi" gibi düşünülebilir. Kalp insanda nasıl organları canlı tutmak için atıyorsa, kontrol biriminden çıkan sinyallerde mikroişlemci içerisindeki birimleri canlı-aktif olarak çalışmasını sağlar.

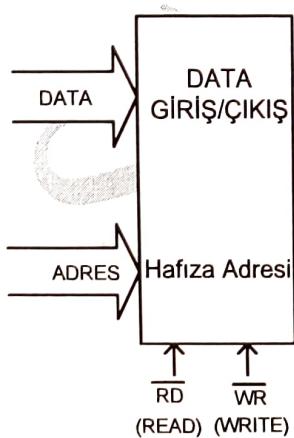


- i. MUX (Multiplexer) ve ALU için kontrol hatalarını düzenler.
- ii. Kaydediciler, hafızalar ve giriş-çıkış cihazları için okuma-yazma darbeleri üretir.
- iii. Program sayacısını (PC) artırır veya eksiltir.
- iv. Tüm işlemleri bu birim kontrol eder ve müdahale yapılamaz!

**IR:** (Instruction Register/Komut Kaydedicisi) CPU'nun koşturacağı kodu binary kodunda tutan kaydedicidir.

**ID:** (Instruction Decoder/Komut Kod Çözücü) Komut kaydedicisinde tutulan bilgiyi tanımlayarak kontrol biriminin doğru zamanlama ve kontrol darbelerini üretmelerini sağlar.

**MAR:** (Memory Adress Register/Hafıza Adres Kaydedici ) Hafızada okuma-yazma işlemi yapılacaksa; ulaşmak istenilen bölgenin adresinin tutulduğu yerdir.

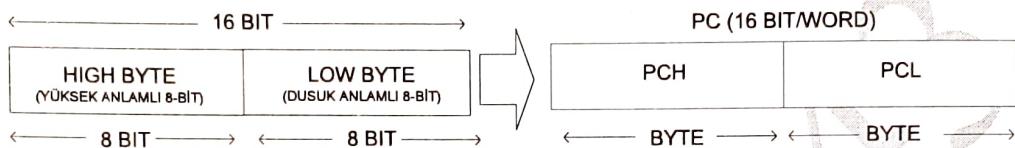


Read/Write (RD, WR) sinyalleri Kontrol birimi tarafından üretilir, adres bilgisi MAR'dan alınır. Hem veri hem adres yoluna bağlıdır.

**MUX:** (Multiplexer/Çoğullayıcı) Kaydediciler, adres yolu ve veri yolu arasında bilgi transferi için yol seçimini yapan elektronik anahtarlama elemanıdır. Denetimi kontrol birimi (control unit) tarafından yapılır.

**PC:** (Program Counter/Program Sayacı) Genellikle 16-bit olan PC; programdaki bir sona gelecek komutun adresini saklar. Yani çalıştırılacak komut buradan yola çıkar. PC'da adres değişikleri yapılır, satır atlanabilir. PC'nin gösterdiği adres, son komut kullanıldıktan sonra otomatik olarak artar veya program sayıcısına yüklenen 16 bitlik yeni bir komut adresi ile değiştirilebilir.

- ✓ 16 bitlik adres bilgisi 8 bitlik bir işlemcide aşağıda gösterildiği gibi taşınır;



Veri yolu (Data bus) 8-bit (byte) olduğu için 16-bit (word) bir bilgi iki adet 8-bit (byte) veri olarak iki kerede RAM'den PC'ye taşınır. Bu akış kontrol birimi tarafından yapılır. Örneğin; A6 C1 verisi A6 → PCH ve C1 → PCL olmak üzere verinin ilk parçası olan "A6" PCH'a 8 bitlik data bus'tan gelir ve yerlesir. Diğer parça da hemen ardından aynen taşınır ve PCL'a yerlesir.

Adres yolu (Address bus) 16 bitlik olduğundan parçalanma olmaz aynen iletim olur.

**WR:** (Working register/ Çalışma Kaydedicisi) Bu kaydediciler çeşitli fonksiyonları gerçekleştirebilirler. Bir kısmı çeşitli hafıza, bir kısmı da özel fonksiyon kaydedicileri olarak kullanılır. Sayıları çok fazla olabılır (50-54 civarı) 8 bit veya 2x8 bit olabilir. Örnek olarak bir zamanlayıcı (timer) eklemek istersek bunu WR ile gerçekleştirebiliriz.

### 2.3 Mikroişlemci Programlama

**Programlama Dili ve Seviyeleri:**

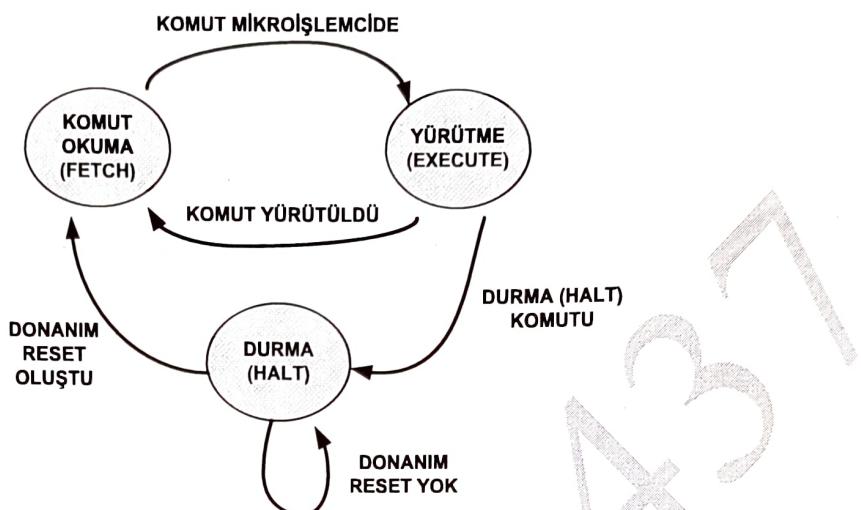


Şekil 7 Programlama Dili seviyeleri

Mikroişlemci programlamada program dili seviyesi makine dili seviyesine ne kadar yakınsa program dili o derece karmaşıktır fakat bunun yanında mimariye o derece esnek bir şekilde müdahale edilebilir. Bu ders boyunca makine diline en yakın olan Assembly programlama dili hakkında bilgi verilecektir.

## Mikro Program Koşutulması:

Bir mikro program fetch (gidip-getirme periyodu) ve execute (yürütme) olmak üzere iki parçadan oluşur.



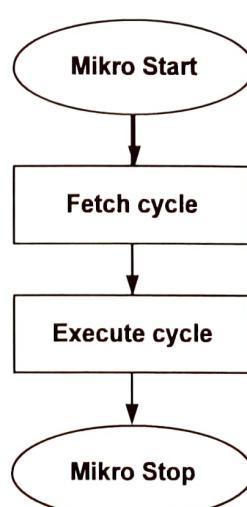
Şekil 8 Bir mikro programın koşutulması:

**Fetch Cycle:** [Gidip-getirme Periyodu] Mikro programın ilk kısmı (işlem kodu/opcode) komutun bulunduğu hafızadan alınması ve CPU'da uygun kaydediciye konmasıdır.

Komut saklayıcısına gelen komut hangi birime gideceği kod çözücü tarafından belirlenir. Bu ise kontrol biriminin diğer devrelerine uygun ve doğru sinyaller göndermesini sağlar. 2 veya 3 byte'lık komutlarda işlem kodu byte 1 alınır alınmaz komutun byte 2 ve byte 3 kısmını hafızadan alınmaktadır. Bu işlemlerin tümü "fetch cycle" olarak adlandırılır.

**Execute Cycle:** [Yürütme periyodu] Mikroprogramın ikinci kısmı ise CPU'ya getirilen komutun koşutulmasıdır.

## Genel mikro program akış diyagramı



## Örnek uygulama:

CPU'nun fetch ve execute periyodu belirlenen bir hafıza bölgesindeki bilginin akümülatöre yüklenmesini inceleyelim;

- Detay:

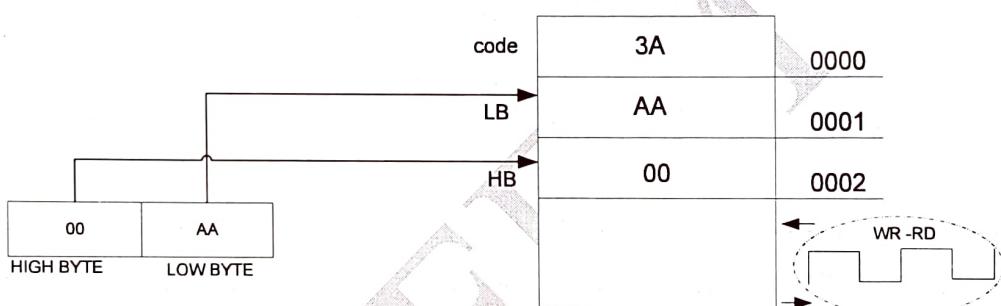
- 3 Byte komut, sırasıyla 0000,0001,0002 (hex) adresleridir.
- Akümülatöre yüklenecek sayı: 38 (hex)
- Akümülatöre yüklenecek sayının adresi: 00AA (hex)

- Kod hafızası:

İşlemi yapacak Kod → “LDA addr”

LDA komutunun karşığı → “3A”dır.

“LDA 00AA hex” Önce low byte yüklenir ardından High byte yüklenir.



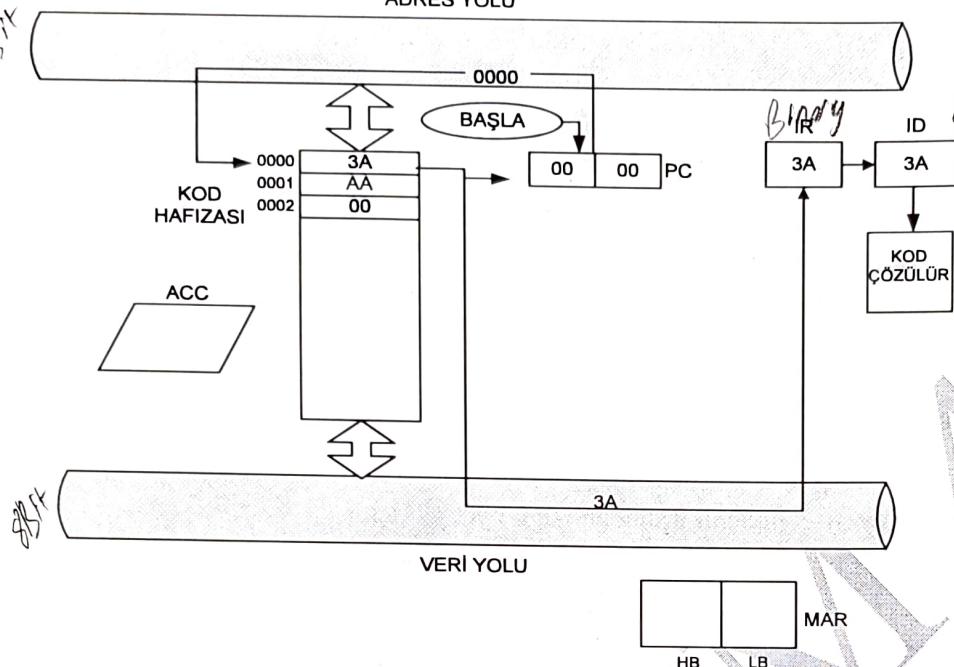
Adres okunacağı zaman RD kısmına oku sinyali ( RD ) gelir okunacak adresdeki veri bu sinyal ile veri yoluna çıkartılır.

- Uygulama:

LDA 00AA KOMUTUNUN MİMARİ ÜZERİNDE FETCH CYCLE  
AŞAMALARININ GÖSTERİMİ

I. Adım

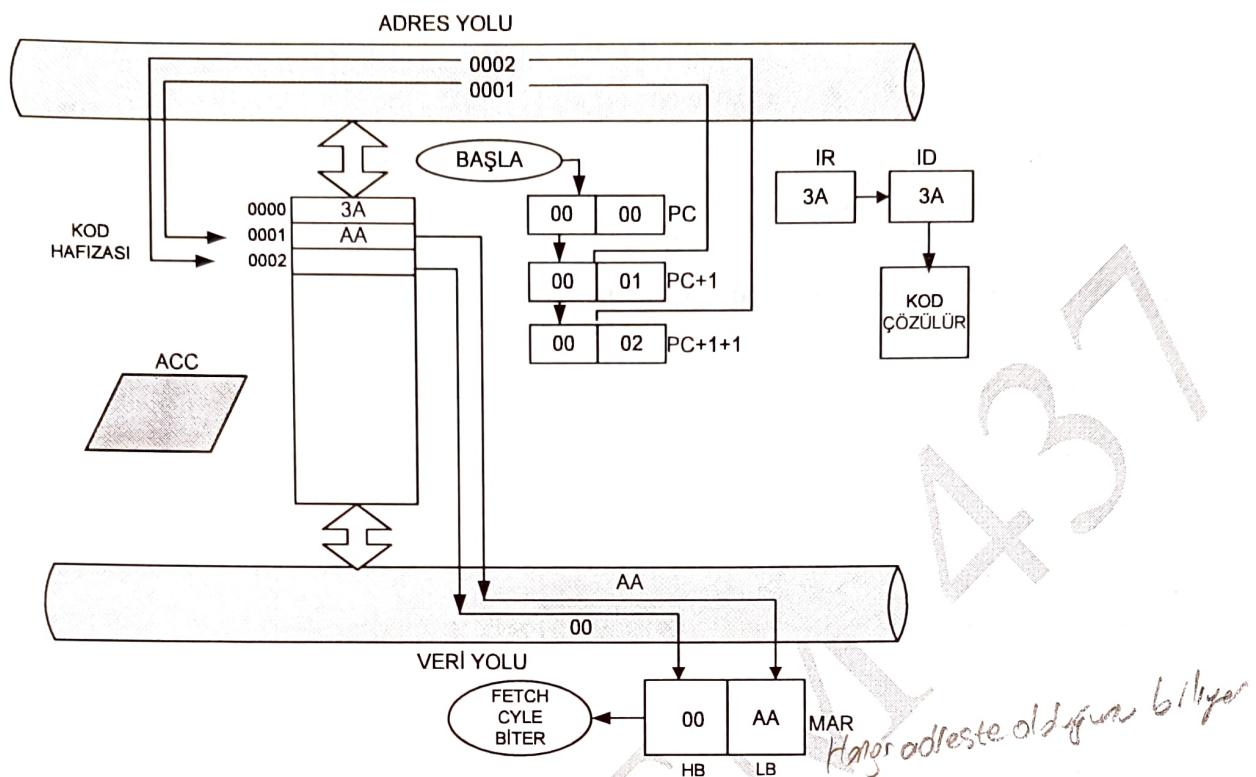
ADRES YOLU



ADIM	İŞLEM	SONUÇ
1	Reset- Program Counter (PC) Donanımsal	PC=0000
2	PC'den adres yoluna transfer	0000 → Adres Yolu'na
3	0000'daki bilgi veri yoluna	3A → Veri Yoluna
4	Veri yolundaki bilgi IR'ye yüklenir	3A → IR
5	IR'deki bilgi ID'ye yüklenir	3A → ID

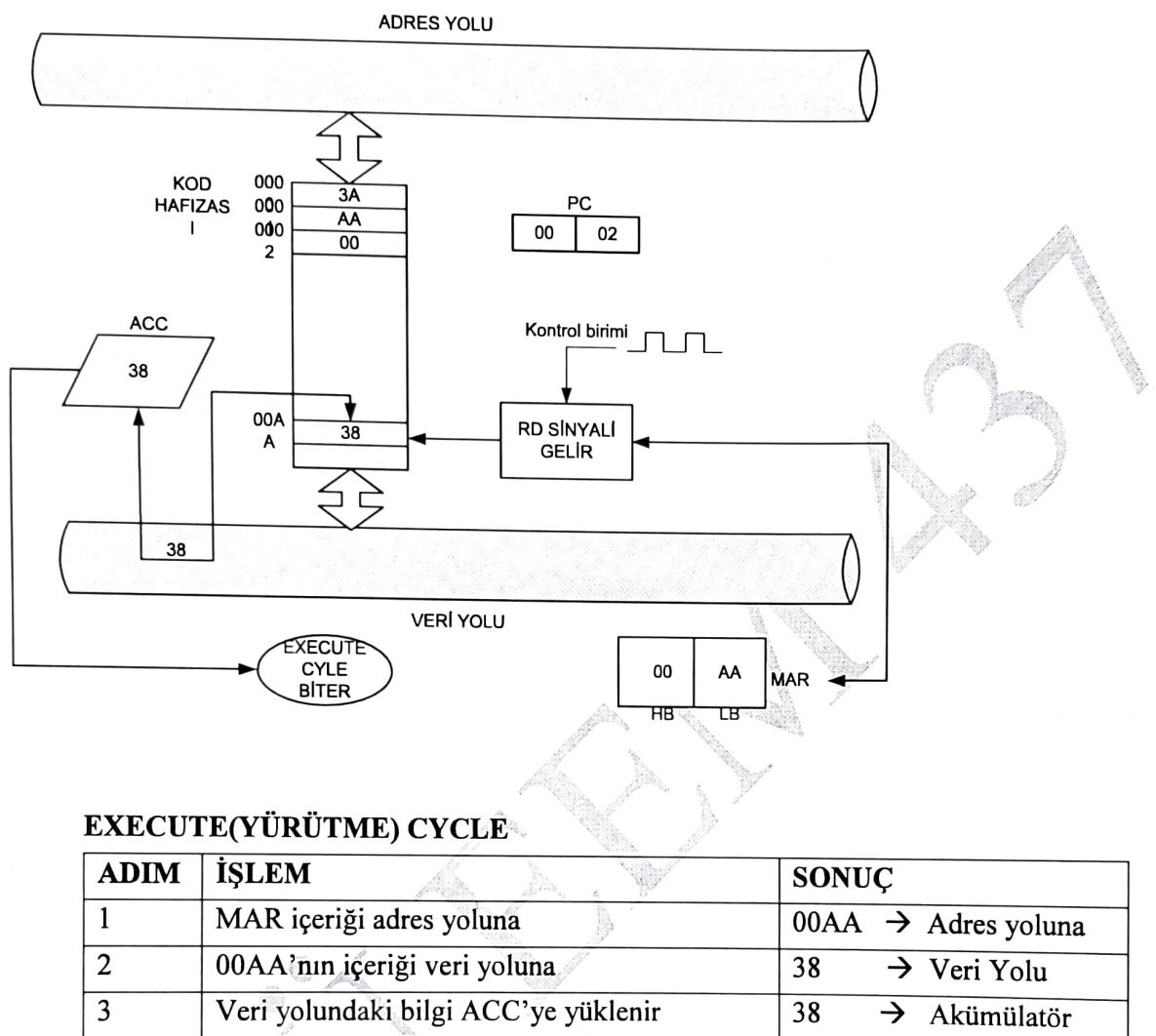
I. ADIM

## II. Adım



ADIM	İŞLEM	SONUÇ	II. ADIM
6	Kontrol birimi komut zamanlama sinyallerini üretir		
7	$PC+1 \rightarrow$ Bir sonraki	$0001 \rightarrow PC$	
8	$PC$ 'den adres yoluna transfer	$0001 \rightarrow$ Adres Yolu'na	
9	$0001$ 'deki bilgi veri yoluna aktarılır	$AA \rightarrow$ Veri yolu	
10	Veri yolundaki bilgiyi MAR alır	$AA \rightarrow MAR$	
11	$PC+1 \rightarrow$ Bir sonraki	$0002 \rightarrow PC$	
12	$PC$ 'den adres yoluna transfer	$0002 \rightarrow$ Adres Yolu'na	
13	$0001$ 'daki bilgi veri yoluna	$00 \rightarrow$ Veri Yolu	
14	Veri yolundaki bilgi MAR'ye yüklenir	$00 \rightarrow MAR$	
15	$PC+1 \rightarrow$ Bir sonraki	$0003 \rightarrow PC$	
<b>FETCH CYCLE BITER</b>			

## LDA 00AA KOMUTUNUN MİMARİ ÜZERİNDE EXECUTE CYCLE AŞAMALARININ GÖSTERİMİ



- ✓ **Önemli Özellik:** "PC" mikroişlemci ilk çalıştırılması için enerji verildiği anda kendini resetler ve daima "0000" adresinden okumaya başlar.

## BÖLÜM 3. 8051 MİKROİŞLEMCİSİ VE ADUC841 MİKRODENETLEYİCİSİ

**3.1. Standart 8051:** 8051 Intel firması tarafından, 1980'lerin başında piyasaya sunulan, dünyanın en popüler 8-bit mikroişlemcisidir. Bu mikroişlemci için, başta Intel olmak üzere, pek çok üretici firma (Philips, Dallas, Siemens, Oki ve Matra/Harris gibi) geniş bir donanım ve yazılım desteği sunmuş ve bunun neticesi, 8051, 1980'lerden bugüne, bir endüstri standarı olmuştur.

8051 ailesi bazen MCS-51 (Microcomputer System - MCS) ailesi olarak da belirtilir. "MCS" harfleri, geleneksel olarak, Intel firmasının üretmiş olduğu, bir sistemi veya CPU'lar, hafızalar, saat üreteçleri, giriş/ çıkış birimleri ve benzeri birimler içeren, mikrobilgisayar ve uyumlu parçaları (components) belirtmede kullanılmıştır. 8051 ve MCS-51 ifadeleri, eş anlamlı olarak aynı aileyi belirtmek için kullanılmıştır ve genelde, bir işlemciyi belirtmeyip ailenin ismi olarak kullanılmaktadır. Bununla beraber, 8051, MCS-51 ailesinin ilk üyelerinden olan, 8051, 8751 ve 8031 mikroişlemcilerinden birinin de adıdır. Bugün için değişik mikroişlemci aileleri arasında, 8051 ailesi, gelişmiş ürünleriyle beraber yaklaşık % 40 gibi bir piyasa payına sahiptir.

### Bu Derste 8051 ailesinin anlatılmasının sebepleri:

**Çok kaynaklılık:** Günümüzde çok değişik 8051 işlemcisi üreten 70 üreticisinin üzerinde firma bulunmakta ve sayısız yazılım ve bilgi kaynakları bulunmaktadır.

### **Bazı 8051 Ailesi üreticilerinden bazı örnekler:**

- Intel (original)
- Atmel
- Philips/Signetics
- AMD
- Infineon (formerly Siemens)
- Matra
- Dallas Semiconductor/Maxim

**Popülerlik:** Kolay bir şekilde bulunmakta ve desteklenmektedir. 8051 geliştiricileri için birçok Internet Web sayfası, kitaplar, teknik dökümanlar, yazılım ve donanım gereçleri bulunmaktadır.

**Uygun, Hızlı ve güçlü:** 8051 çekirdek mimarisi hedef kontrol uygulamalar için çok uygun olup hızlı ve güçlündür.

**Geniş yelpaze ve uyumluluk:** Çok değişik 8051 ürünleri olmasına karşın, ikili kod düzeyinde bütün ürünler uyumludur (compatible). Diğer mikroişlemci aileleri, 8051'in sunmuş olduğunu, farklı ve

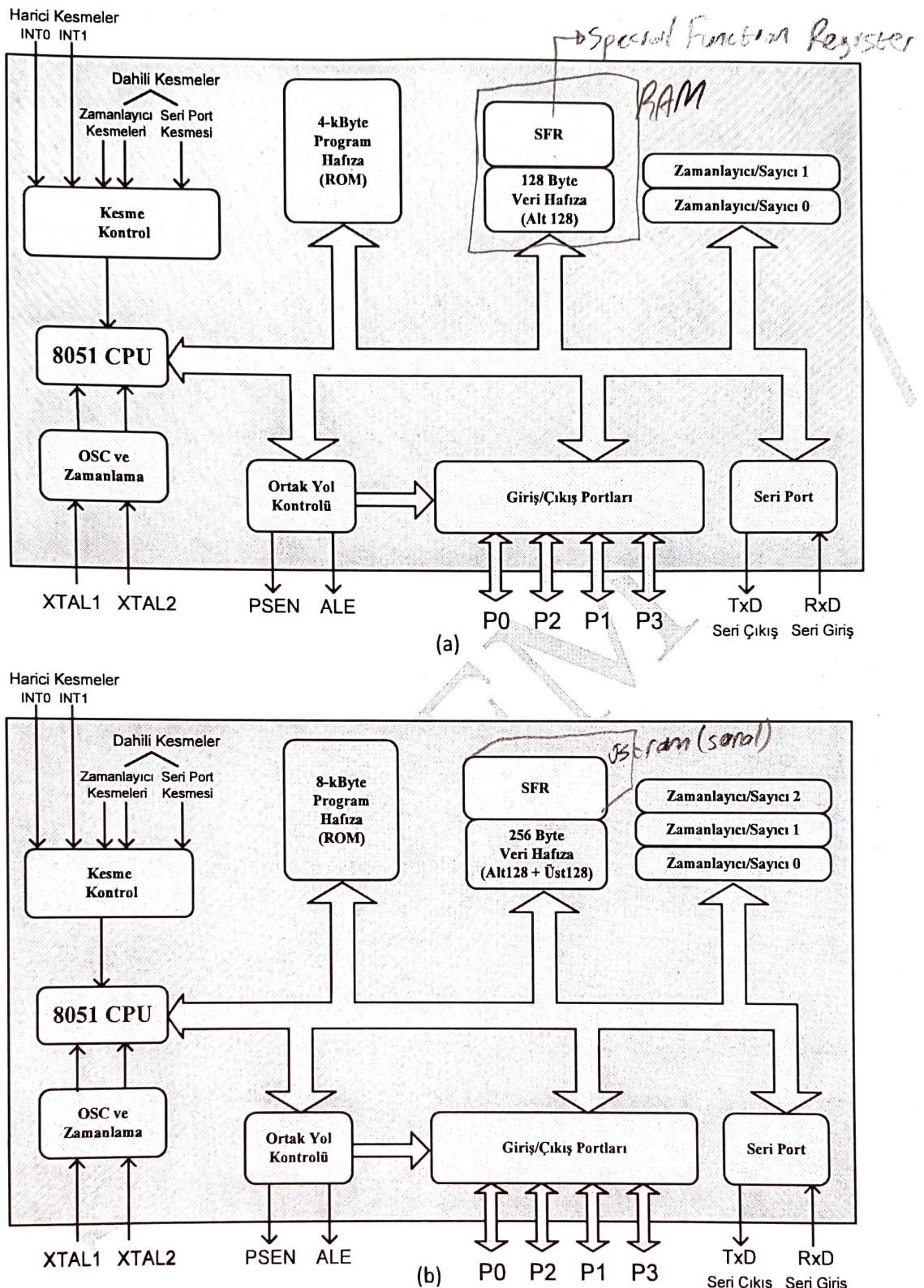
uyumsuz işlemcilerle (genellikle tek üretici firma kaynaklı olarak) ancak sağlayabilmektedir. Bu uyumluluk, kolaylık ve esneklik, program geliştirme araçlarında, eğitiminde ve yazılım desteğinde de bulunmaktadır.

**Sürekli iyileştirme:** 1980'lerden bugüne silikon ve tasarım olarak sürekli geliştirilen 8051'ler hızını ve gücünü arttırmıştır. Günümüzde 16-bit modellerinin de bulunduğu 8051'ler değişik kapasite, boyutlarda ve fiyatlarda kolay bir şekilde temin edilebilmektedir.

8051 Intel firması tarafından üretilen MCS-51 ailesinin ilk mikroişlemci olup MCS-51 ürünlerinin temel çekirdeğidir. 8051 mikrodenetleyicisinin temel özellikleri şunlardır:

- Kontrol uygulamalarına yönelik 8-bit CPU
- Yoğun Boolean işlemleri yapabilme (tek-bit lojik işlemler) özelliği
- 64K Program Hafıza (Program Memory) adres alanı
- 64K Veri Hafıza (Data Memory) adres alanı
- 4K tümdevre-üzeri (on-chip) Program Hafıza
- 128 byte tümdevre-üzeri veri RAM
- 32 tane iki yönlü adreslenebilir I/O hatları
- 2 tane 16-bit Zamanlayıcı/Sayıçı (Timer/Counter)
- Full duplex UART (Universal Asynchronous Receiver Transmitter)
- İki öncelik seviyesine sahip 6-kaynak / 5-vektörlü kesme donanım yapısı
- Uzun Boolean işlemleri yapılabilir.
- On-chip saat osilatörü.

Şekil 9'da gösterildiği gibi temel 8051 mimarisinde tüm devre üzerinde 4-kByte program hafıza ve 128-Byte veri hafıza mevcuttur. Ailenin sonraki üyelerinde hafıza değerleri zamanla artmıştır. Örneğin 8052 mimarisinde 8-kByte program hafıza ve 256-Byte veri hafıza bulunur. 8052 mikrodenetleyici mimarisinde aşağıda blok diyagramında gösterildiği gibi standart 8051'e ilave olarak 128-Byte dahili veri hafıza (Toplam 256-Byte dahili RAM), Zamanlayıcı/Sayıçı-2 (Toplam 3 adet 16-bit Zamanlayıcı/Sayıçı) bulunur.



Şekil 9 8051 (a) ve 8052 (b) mikrodenetleyicilerine ait basitleştirilmiş blok diyagramı

### 3.2. Aduc841 Mikrodenetleyicisi

Mikrodenetleyici mimarilerinde dahili olarak bulunan çevre birimleri endüstriyel uygulamalarda duyulan ihtiyaçlara ve teknolojik gelişmelere paralel olarak sürekli olarak gelişmekte ve yeni çevre birimleri mikrodenetleyici mimarilerine ilave edilmektedir.

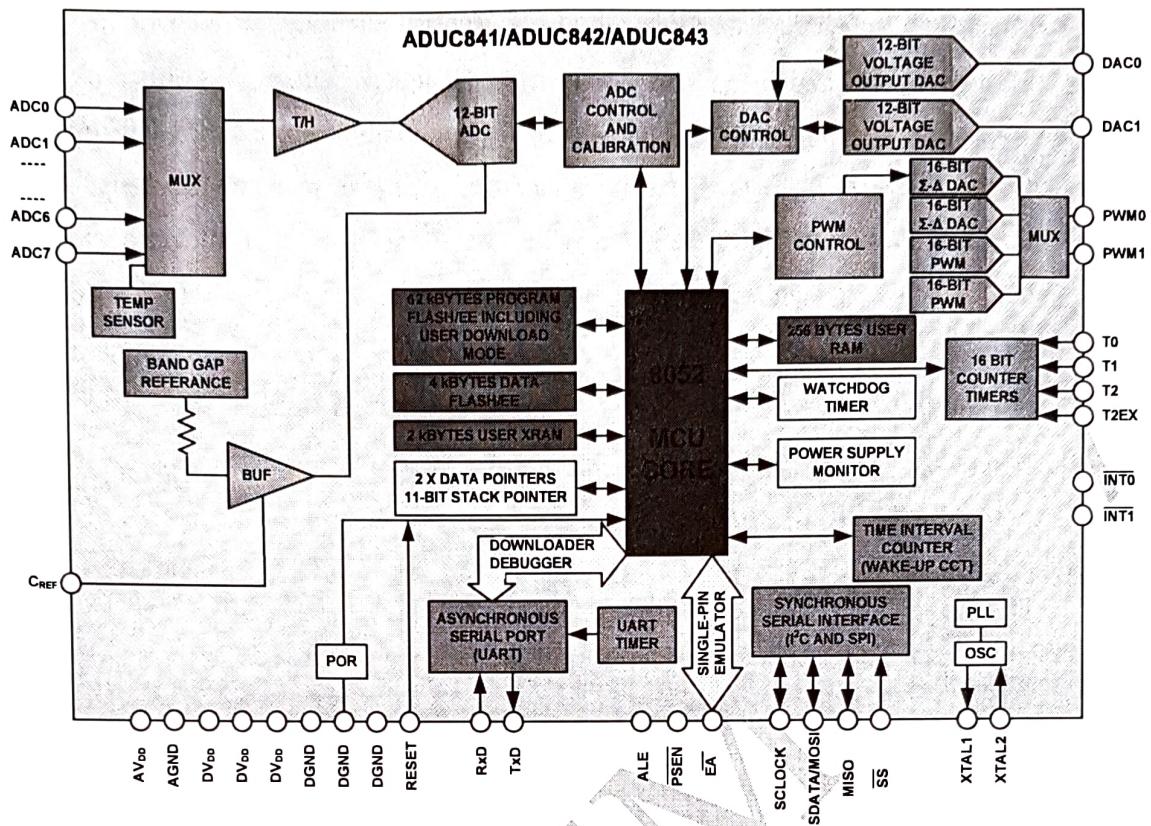
İlk olarak Intel firması tarafından 1980 yılında üretilmeye başlanan 8051 ailesi de sürekli olarak gelişmekte ve standart 8051 mimarisinin temel özellikleri korunarak endüstriyel uygulamalardaki ihtiyaçlara cevap verebilecek 8051 ailesinin yeni nesil üyeleri birçok firma tarafından üretilmektedir.

Bu doğrultuda ADUC841 mimarisinde,

- yüksek hızlı analog giriş (ADC)
- analog çıkış (DAC)
- darbe genişlik modülasyon (Pulse Width Modulation, PWM)
- ilave hafıza (dahili XRAM, flash veri hafıza) external → internal
- Uzun ömürlü (100 yıl) ve büyük kapasiteli Flash Program hafıza
- ilave kesme kaynakları
- yüksek hızlı haberleşme birimleri ( $I^2C$ , SPI, CAN)
- kısır-döngü zamanlayıcısı (Watchdog Timer)
- besleme gerilimi izleme (Power Supply Monitor, PSM)
- uyku modu (Power-down)
- devre üzerinde programlanabilme (In System Programming, ISP)

vb. bir çok özelliği barındıran 8051 tabanlı mikrodenetleyiciler günümüzde çeşitli firmalar tarafından üretilmektedir.

Aşağıda Analog Devices firması tarafından üretilen 8052 tabanlı gelişmiş bir mikrodenetleyiciye (Aduc841) ait blok diyagram verilmiştir.



Şekil 10. Aduc841 blok diyagramı

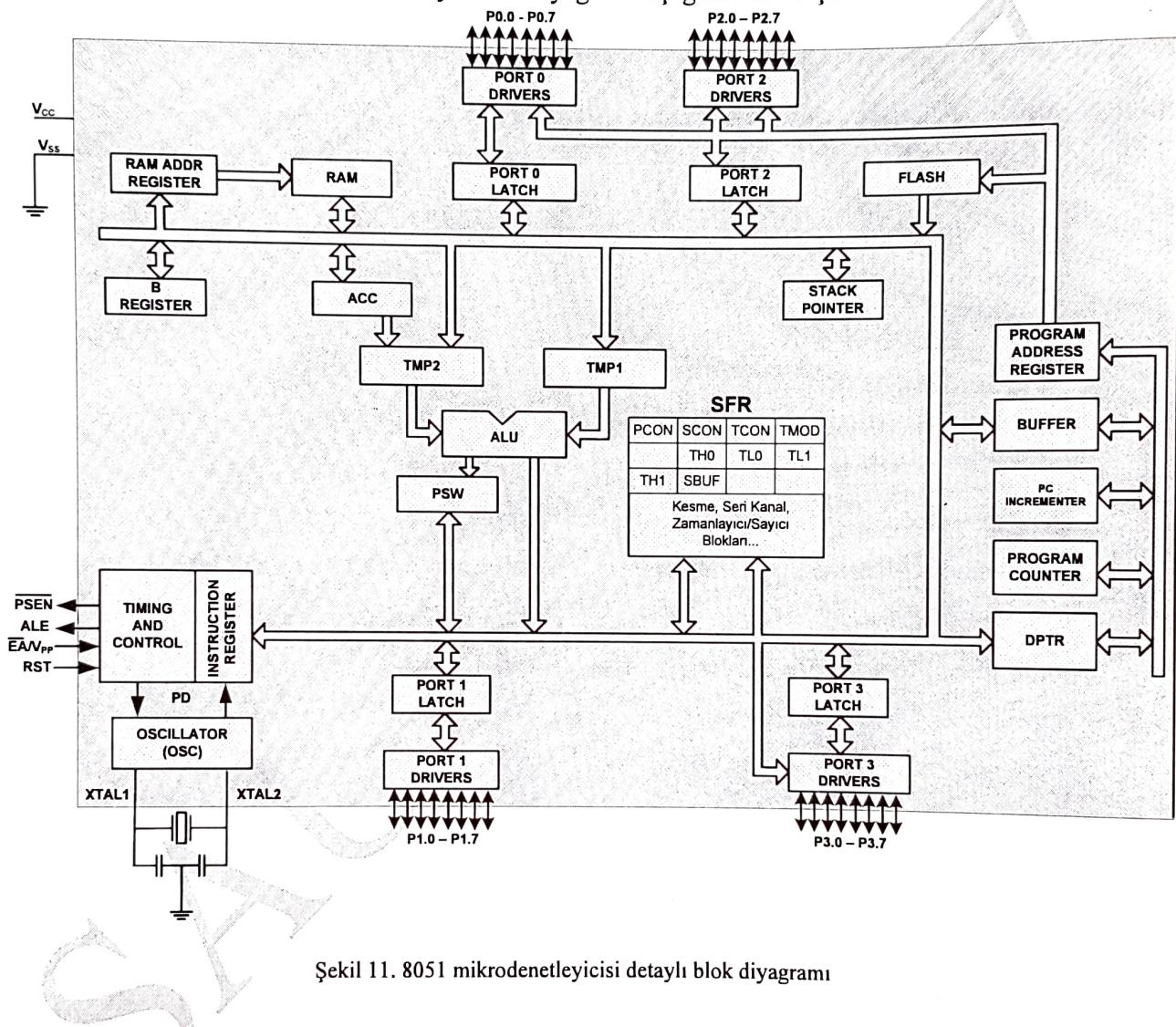
Ders süresince öncelikle temel 8051 mikrodenetleyici çekirdeği, ardından Aduc841 mikrodenetleyicisi mimarisinde bulunan çevre birimleri detaylı olarak incelenecaktır. Aduc841 mikrodenetleyicisi temel bazı özellikleri aşağıda belirtilmiştir.

- Endüstri standartı 8051 mikrokontrolör mimari yapısı
- Tek-çevrim, 20 MIPS işlem hızı
- 62 kB'a varan program belleği,
- 4 kB'a varan Flash/EEPROM Data belleği
- Program ve veri hafızalarının devre üzerinde (in-circuit) programlanabilme özelliği
- 2304 Byte dahili XRAM
- Yüksek doğrulukta – yüksek hızda (420 kSPS), 12-bit, 8- kanal ADC
- ADC için yüksek kararlılık (15 ppm/ $^{\circ}$ C) dahili gerilim referansı
- 2 adet 12-bit DAC
- Çift çıkışlı PWM
- Tümdevre üzeri sıcaklık izleme
- 2 öncelik seviyeli 12 kesme kaynağı
- Çift DPTR ve genişletilmiş (11-bit) yiğin (SP)
- I<sup>2</sup>C, SPI haberleşme birimleri

- Kısır-döngü zamanlayıcısı (Watcdog Timer)
- Besleme gerilimi izleme (Power Supply Monitor, PSM)
- Enerji tasarrufu için uykdu modu (Power-down)

### 3.3 8051 Mikrodenetleyici Mimarisi ve Çevre Birimleri:

8051 mikrodenetleyici mimarisinin detaylı blok diyagramı aşağıda verilmiştir.



Şekil 11. 8051 mikrodenetleyicisi detaylı blok diyagramı

#### 3.3.1. Yonga (Chip) Üzerindeki Bazı Çevre Birimleri:

##### 1. 4 Adet 8 Bitlik Paralel Giriş - Çıkış:

- ✓ 8051 mimarisinde 4 adet 8-bitlik giriş-çıkış portu (Port0-P0, Port1-P1, Port2-P2, Port3-P3) bulunur. P0, P1, P2, P3; portların özel isimleridir.
- ✓ Portlar-pinler fiziksel dünya ile mikrodenetleyici arasında lojik veri transferinin (1/0, On/Off) gerçekleşmesini mümkün kılar.
- ✓ 4x8=32 adet giriş-çıkış olup her biri kendinden bağımsız çalışabilir.

- ✓ P0.3,P0.4,P3.3, vb; bu sembollerle istenilen **her bir bit** kontrol edilebilir. (Bit tabanlı veya byte tabanlı giriş-çıkış yapılabılır.)
- ✓ Intel 8051 mikroişlemcisinin en önemli özelliklerinden biri bit-bit işlem yapabilmesidir. Bu olay; kumanda gibi bit işlem gerektiren durumlarda oldukça fazla düzeyde kolaylık sağlar.
- ✓ **Reset** sonrasında P0, P1, P2 ve P3 port saklayıcıları donanım tarafından **0FFh** değeri ile yüklenir.

## 2. 1 Adet Seri-Giriş-Çıkış:

- ✓ Paralel haberleşmede her bir bit için ayrı bir iletişim hattı gereklidir.
- ✓ Seri haberleşmede ise iletişim tek bir veri hattı ile sağlanır. Mevcut paralel veriler, ara işlemle paralel bilgilere çevrilir ve tek bir hat üzerinden iletişim sağlanmış olur.

## 3. 2 Adet 16 bitlik Zamanlayıcı Sayacı (Timer/ Counter ):

Bağımsız olarak programlanabilirler. Timer ve Counter opsyonel olup, ayrı ayrı kullanılabilirler.

## 4. Stack Pointer: (Yığın İşaretleyicisi/ 8 Bit)

Alt programlara ve kesme serisi programlarına dikkat ederken, mikroişlemci biriminin durumunu saklaması ve program esnasında parametre aktarımı, geçici değişken saklanması (RAM' de) için RAM' deki yığın adresini gösterir. İleride Stack Pointer ile ilgili daha detaylı bilgi verilecektir.

- ✓ 8051 entegresinin ilk çalışmasıyla, yığın işaretleyicisi 07h değerini alır. Bunun anlamı yığının başlangıç adresinin 08h olduğunu söylemektedir. Yığın 08h adresinden yukarı doğru genişler. Bu sebeple eğer programınızda register bankları 1, 2 ve 3 ü kullanacaksanız dikkatli olmalısınız. İleride bu konuya ilgili detaylı bilgi verilecektir.

## 5. Data Pointer:

- ✓ Data Pointer (DPTR- Veri işaretçisi) 16-bit (2-Byte) uzunluğunda kaydedicidir.
- ✓ DPH (yüksek Byte) ve DPL (düşük Byte) saklayıcıları aşağıda gösterildiği gibi DPTR saklayıcısının sırasıyla yüksek ve düşük değerli 8-bitini oluşturmaktadır.



- ✓ DPH ve DPL saklayıcılarına ayrı ayrı erişilebildiği (okunup-yazılabilir) gibi bazı özel komutlarla ikisinede aynı anda erişilebilir.

- ✓ Program dallanmalarında ve yonga dışındaki veri bölgelerine erişimlerde kullanılacak adres tabanının saklandığı yerdir.
- ✓ 64 KB'lık harici veri alanından akümülatöre veya tam tersi yönde veri aktarımı sağlar.
- ✓ Kısaca MAR → DPTR olmuştur.

## 6. PSW (Program Status Word/ Program Durum Göstergesi):

- ✓ Komutların yürütülmesi esnasında bazı özel durumların oluşup-olmuşadığını gösteren bayrakların (flags) göstergesidir.

## 7. B Register:

Yardımcı bir kaydedicidir. Aritmetik işlem esnasında RAM giriş çıkış olarak kullanılabilir.

### 3.3.2. Hafıza Yapısı:

Genel olarak mikrodenetleyiciler iki tür hafızaya sahiptirler.

- i) Veri Hafıza
- ii) Program Hafıza

Veri hafıza ise mikrodenetleyicinin çalışması esnasında oluşan verilerin, geçici değerlerin tutulduğu kalıcı olmayan (volatile) türde hafızalardır. Bu tür hafızalarda ise besleme gerilimleri kesildiğinde içerdikleri veri korunamaz. Program hafıza mikrodenetleyicinin koşturacağı programa ait makine kodlarının saklandığı kalıcı (non-volatile) türde hafızadır. Kalıcı hafızaların isminden de anlaşılacağı gibi besleme gerilimleri kesildiğinde içlerinde depolanan veri kaybolmaz. Mikroişlemcili/Mikrodenetleyicili sistemlerde kullanılan hafıza birimleri genel olarak 5 grupta incelenebilir.

### RAM (Random Access Memory)

Hem okunabilir hem de yazılabilir hafıza olan RAM mikrodenetleyicilerde veri hafıza olarak kullanılır. RAM'lerin beslemeleri kesildiğinde içerdikleri veriler kaybolur, silinir (volatile memory).

- ✓ Intel 8051 mikroişlemcisinin iç yapısında 128 Byte büyüklüğünde dahili RAM bulunmaktadır.

### ROM (Read Only Memory)

Sadece okunabilir hafıza olan ROM mikrodenetleyicilerde program hafıza olarak kullanılır. ROM hafıza üretim esnasında programlanır (Factory Mask ROM) ve kullanıcı ROM'un içeriği veriyi sadece okuyabilir, değiştiremez. ROM'lar kalıcı hafızalardır (non-volatile memory).

## **EPROM (Erasable Programmable Read Only Memory)**

ROM'lardan farklı olarak EPROM'lar uygun programlama cihazları ile yeniden programlanabilir. EPROM'lar yeniden programlanmadan önce üzerlerinde bulunan küçük cam yüzey ultra-viole ışığı tutularak içeriklerinin silinmesi gereklidir.

## **EEPROM (Electrically Erasable Programmable Read Only Memory)**

EPROM'lardan farklı olarak EEPROM'larda silme işlemi de elektrikli olarak yapılır ve ardından uygun programlama cihazları ile yeniden programlanabilir. Ancak EEPROM'ların silme ve yazma işlemi yavaştır.

## **Flash (Flash EEPROM)**

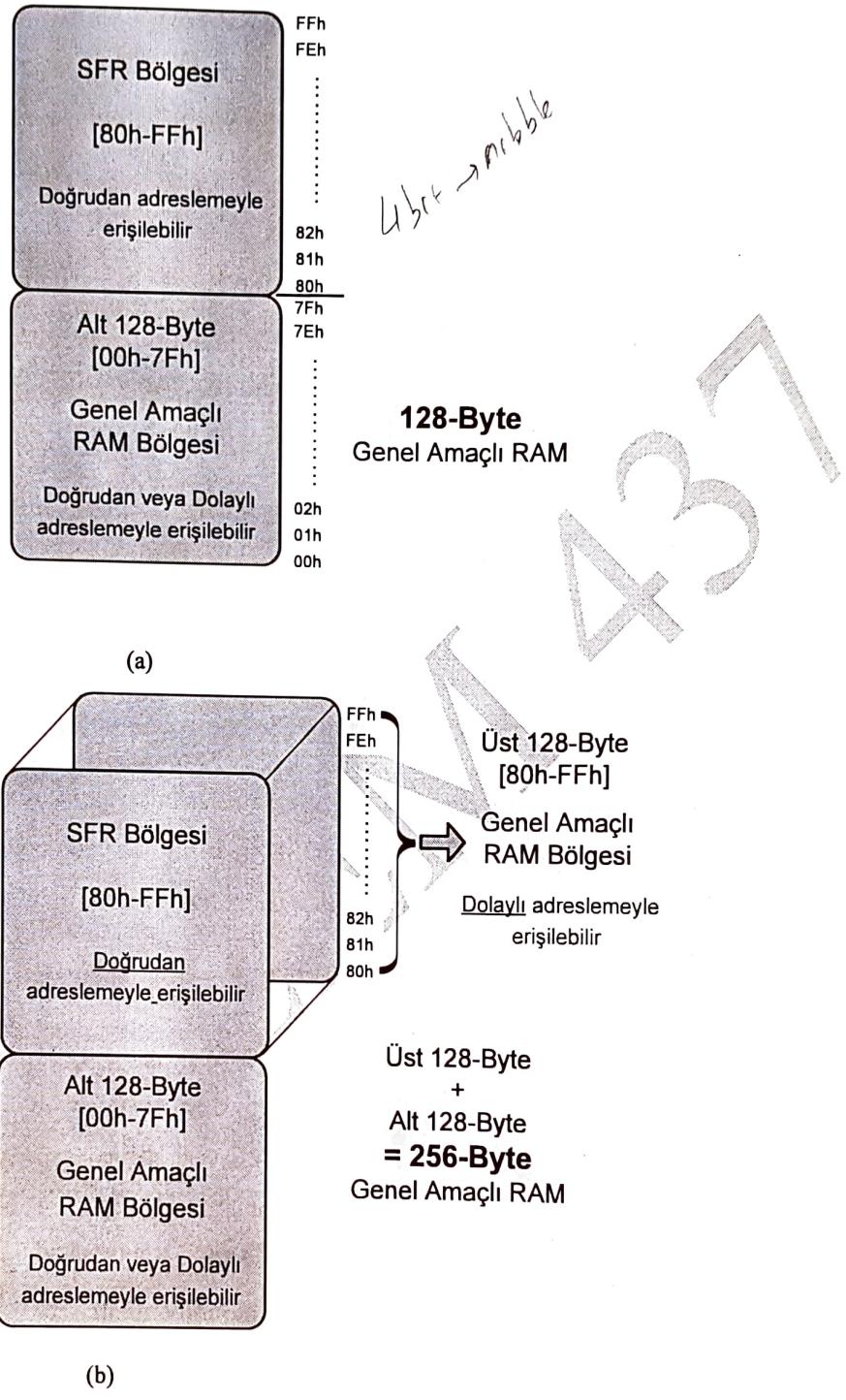
EEPROM'ların silme ve yazma işlem hızının artırıldığı özel bir türüdür. Günümüzde tüm mikrodenetleyiciler program hafıza olarak Flash hafızaya sahiptirler.

## **Veri Hafıza**

Mikrodenetleyicinin çalışması esnasında gerçekleştirilen çeşitli işlemlerin sonuçlarına ait verilerin geçici olarak saklandığı genel amaçlı hafızadır. Veri hafızada tutulan veri değerleri çalışma anında değiştirilebilir (okunup-yazılabilir). Mikrodenetleyicilerde veri hafıza (RAM) dahili (internal, on-chip) veya harici (XRAM) olarak kullanılabilir.

## **Dahili Veri Hafıza**

8051 ailesinin bazı üyeleri (8051, 8751) 128-Byte dahili veri hafızaya bazı üyeleri (8052) ise 256-Byte dahili veri hafızaya sahiptir. 8051 ailesinde dahili veri hafıza aşağıda gösterildiği gibi alt-128 (Lower RAM), üst-128 (Upper RAM) ve Özel Fonksiyon Saklayıcıları (Special Function Register, SFR) bölgesi olmak üzere 3 kısma ayrıılır. Alt 128-Byte ve SFR bölgesi 8051 ailesinin tüm üyelerinde mevcut iken üst 128-Byte ailenin bazı üyelerinde (örneğin 8052) bulunmaktadır.



Şekil 12 8051(a) ve 8052(b) veri hafıza yapısı

### Alt RAM Bölgesi

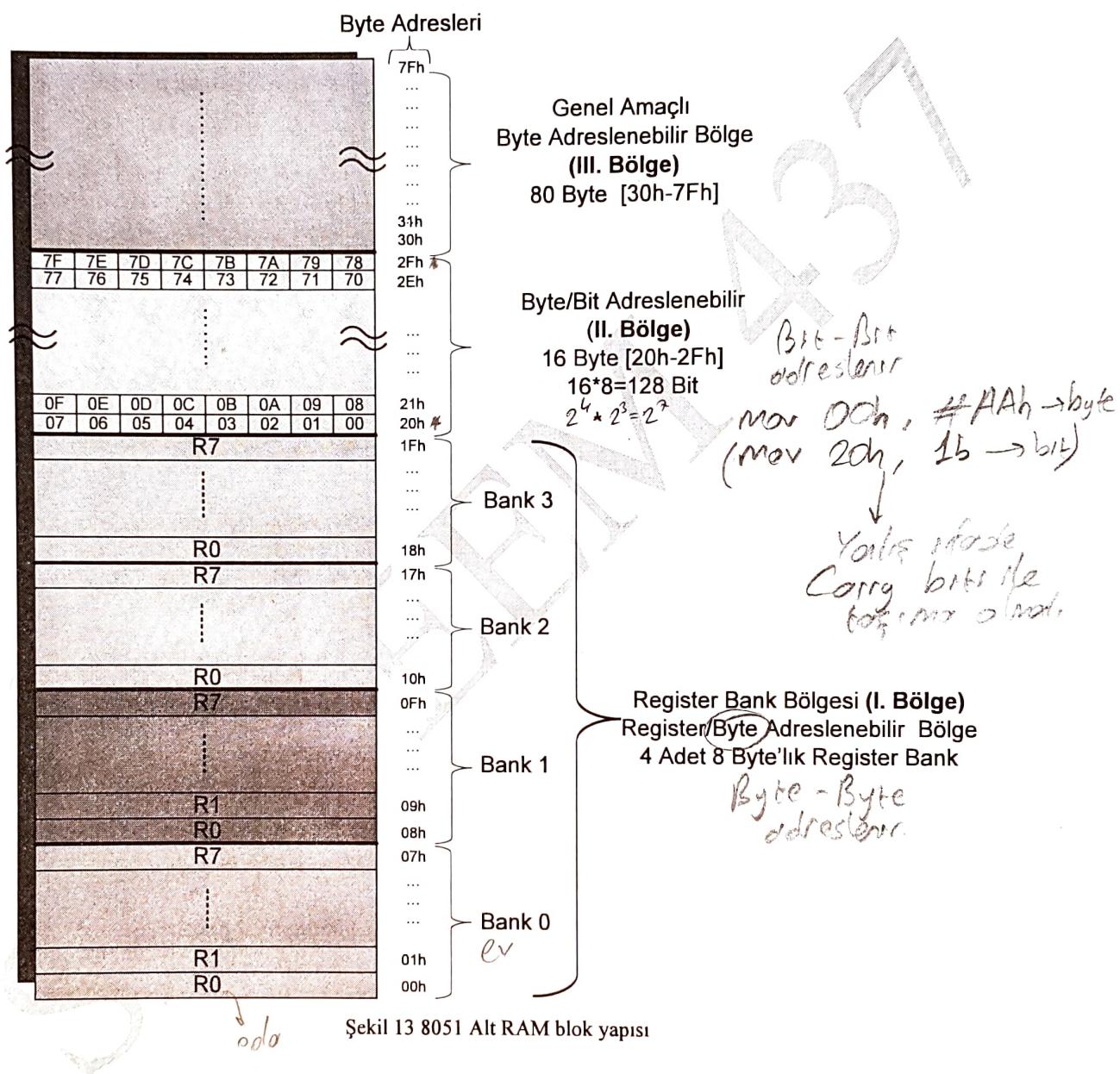
8051 ailesinin tüm üyelerinde bulunan alt RAM bölgesi hem doğrudan hem de dolaylı adreslemeyle erişilebilinen genel amaçlı, 128-Byte [00h – 7Fh] büyüklüğünde veri alanını kapsar. Alt RAM bölgesi aşağıda gösterildiği gibi kendi içerisinde 3 ayrı bölgede incelenebilir.

- 8051 dâhili RAM'ı genel amaçlı RAM olarak kullanılabılır.

i) Saklayıcı (Register) Bölgesi (I. Bölge)

ii) Bit Adreslenebilir Bölge (II. Bölge)

iii) Genel Amaçlı Bölge (III. Bölge)



### PSW Saklayıcısı (Program Status Word, Program Durum Saklayıcısı)

PSW, komutların yürütülmesi esnasında bazı özel durumların oluşup-olşmadığını gösteren bayrakların yer aldığı, saklayıcı banklarının (Bank 0, Bank 1, Bank 2 ve Bank 3) seçimin yapıldığı önemli bir SFR'dır. PSW saklayıcısı **bit adreslenebilir**.

$$PSO = PSW.3$$

$$PS1 = PSW.4$$

CY

PSW:	CY		RS1	RS0			
------	----	--	-----	-----	--	--	--



RS1 ve RS0 bitleri aşağıdaki tabloya göre aktif olan saklayıcı deposunu belirler.

BANK 0

RS1	RS0	Aktif Saklayıcı Deposu
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

**CY:** CY (veya C) biti aritmetik işlemlerde elde fonksiyonunu gerçekleştirir. Çeşitli komutlar yürütülürken elde bayrağının değeri değişimdir. Elde bayrağı komutlarda çoğunlukla "C" olarak kullanılır.

**i) Saklayıcı (Register) Bölgesi:** Alt RAM bölgesinin ilk 32-Byte'lık [00h-1Fh] kısmı saklayıcı bölgesi olarak adlandırılır. Bu alanda her biri 8 saklayıcıdan oluşan 4 adet saklayıcı bankı (deposu) bulunur. Her bir bank'a ait saklayıcılar R0, R1, ... R6, R7 etiketleri ile adlandırılır. Herhangi bir anda bu banklardan sadece bir tanesi aktif olarak kullanılabilir ancak çalışma anında gerektiği durumlarda ilgili kontrol bitleri (PSW SFR'sinin RS0 ve RS1 bitleri) ile aktif saklayıcı bankı istenildiği gibi değiştirilebilir. Aşağıda Alt Ram bölgesi ile ilgili önemli özellikler sıralanmıştır;

- Mikroişlemcide ilk enerji verildiği anda PSW'deki 4 ve 3 no'lu bitler "00" numarasını gösterir. Eğer PSW üzerinde bir oynama yapılmazsa; mikroişlemci Bank 0'dan okumaya başlar.
- PSW'deki 4 ve 3 nolu bitlerdeki diğer kombinasyonlar elde edilerek, RAM'de istenilen **Bank** üzerine işlemler yapılabilir. Örneğin; **Bank 2** için PSW (3 ve 4 no'lu bitleri) = "1-0" olur.
- RAM üzerinde istenilen yere; **hem byte adresleri** (00h, 01h gibi) ile hem de kaydedici adı ile (R0, R1, ..., R7) ulaşılabilir. Ancak kaydedici kullanılırken bank seçiminde kullanım karışıklığı neden olmaması için dikkatli olunmalıdır.
- **II. Bölgede;** hem byte olarak hem de bit düzeyinde işlemler yapılabilir. Bu olay sadece bahsedilen II. Bölgede gerçekleşebilir.
- Aynı isme sahip **bit** ve **byte** adresleri; kullanılacak komutlar vasıtasyyla birbirlerine karıştırılmazlar. (Bit ve byte adres komutları farklıdır.)
- II. Bölgede bulunan 128 bitlik bölgede; **bit bit işlem** yapılabildiğinden dolayı; aynı işlemci ile **128 adet farklı on-off uygulama gerçekleştirilebilir** ve bu özellik 8051'in kumanda olaylarında yaygın olarak kullanılmasının en önemli sebebidir.
- Intel 8051 mikroişlemcisi ile yapılacak kişiler için RAM'ın mimari yapısının çok iyi bilinmesi oldukça önemlidir. Çünkü RAM'ın kişiye sağladığı tüm özellikler bilinerek program yazılırsa; profesyonellik derecesi artar ve yerine göre 128 satırlık işlemler iki-üç satır ile

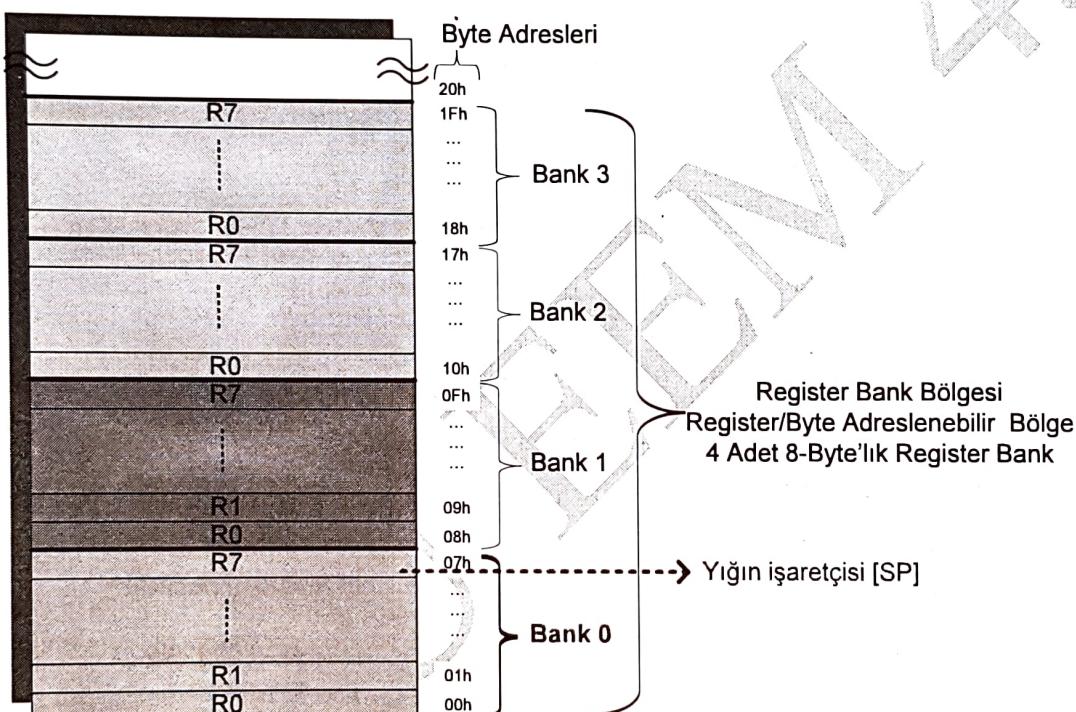
gerçekleştirilebilir. Ayrıca "bit" kullanımı gerektiren yerlerde "byte" kullanılması gibi fazlalık işlemlerden kaçınılmış olunur.

- 8051'e ilk enerji verildiğinde "SP (stack pointer)" RAM' deki "**07**" nolu adresi gösterir.
- PSW'deki CY(Carry): Dış dünya ile ilgili bit bilgi transferi buradan gereklesir.

P1.0 → (Lojik "0" bilgisi) → CY → [00] RAM (Kaydetme)

[05] → (RAM'deki bilgi) → CY → P3.4 Çıkış (Bilgi Okuma)

- Portlardan ya **bit-bit** okuma yapılır ya da **8 bitlik yani 1 Byte'lık** okuma yapılır. Çünkü RAM' de kaydetme ve okuma bilgi saklama birimleri ya 1 Byte'lık ya da 1 Bitlik'tir. Ayri ayrı 2-3 bitlik işlemler yapılamaz.



Şekil 14. Saklayıcı bölgesi

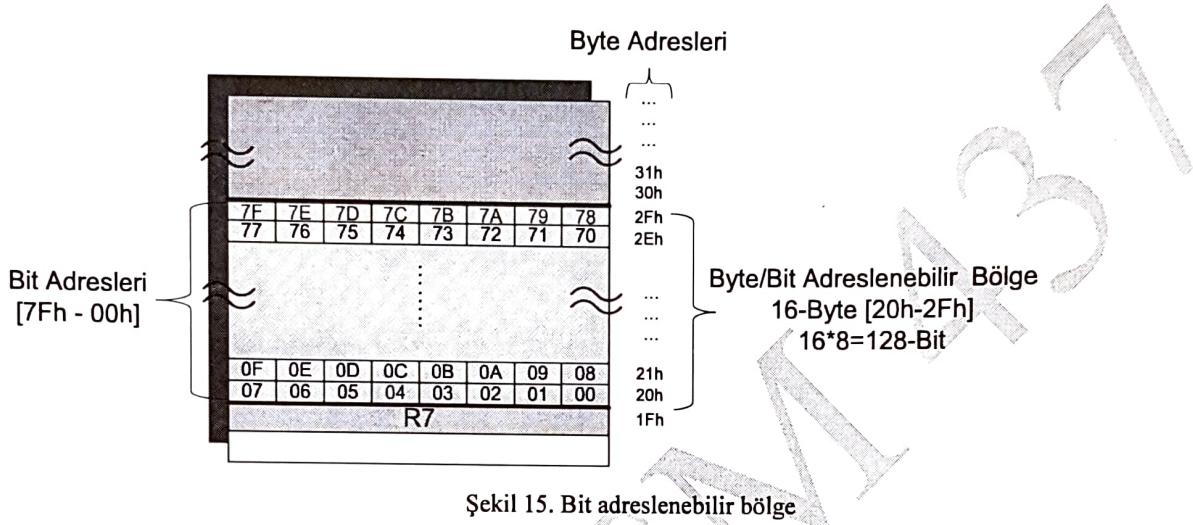
Şekilde gösterildiği gibi tüm depolardaki saklayıcılara aynı isimle  $R_i \ i=0\dots 7$  erişilir. Örneğin  
`mov R0, #45h`

komutu R0 saklayıcısına 45h sayısını yüklemektedir. Fakat komutta hangi depoya ait R0 saklayıcısına (00h, 08h, 10h, 18h-Byte adreslerinden hangisi?) erişilmek istediği bilgisi bulunmaz. Belirtilen R0 saklayıcısı komut icra edilirken aktif olan saklayıcı deposuna aittir. Dolayısıyla saklayıcılar kullanılırken aktif olan saklayıcı deposuna dikkat edilmelidir.

Ayrıca, saklayıcı bank bölgesi kullanılırken yığın işaretçisinin (Stack Pointer, SP) reset sonrasında değerinin 07h olduğu unutulmamalıdır. SP'nin çalışma mantığı alt programlar kısmında detaylı olarak

açıklanmıştır.

**ii) Bit Adreslenebilir Bölge:** Mikrodenetleyicilerin önemli özelliklerinden biri de Byte tabanlı işlemlere ilave olarak Bit tabanlı işlem de yapabilmeleridir. Alt RAM'da [20h-2Fh] arasındaki 16-Byte'luk kısım bit adreslenebilen hafıza bölgesidir. 128-bit içeren bu alandaki her bir bite bit adresleri [00h – 7Fh] ile tek tek erişilebilir. Bit adreslenebilir bölgeye byte adresleme ile de erişmek mümkündür.



**iii) Genel Amaçlı Bölge:** 30h-7Fh arasındaki 80-Byte'luk hafıza alanı byte adresleme ile erişilebilinen genel amaçlı RAM bölgesidir.

### Üst RAM Bölgesi

8051 ailesinin her üyesinde bulunmayan üst RAM bölgesi genel amaçlı saklayıcı olarak kullanılır. Alt RAM bölgesi gibi fonksiyonlara (register bölgesi, bit adreslenebilir bölge) sahip değildir.

Üst 128-Byte RAM bölgesi Şekil 12 (b)'de gösterildiği gibi SFR'ler ile aynı hafıza adres alanını [80h – FFh] kullanır. Dolayısıyla bu iki bölgeden (SFR ve genel amaçlı üst 128-Byte) hangisine erişileceği tamamen kullanılan komutun yapısına bağlıdır. Genel amaçlı üst 128-Byte RAM bölgesine sadece dolaylı adresleme ile erişilirken SFR alanına ise sadece doğrudan adresleme ile erişilebilir. Adresleme modları ileride detaylı olarak incelenmiştir.

## BÖLÜM 4. ADRESLEME MODLARI

Assembly, her bir komutun yerine getirdiği işlevin İngilizce karşılığının kısaltılması (mnemonik) veya akronim (acronym) ile ifade edildiği, makine dili ile yüksek-seviyeli diller arasında kalan alt-düzey bir programlama dilidir. Örneğin,

**mov ACC, #53h ;** (mov → move: hareket etmek, taşınmak, nakil)

komutu ile ACC saklayıcısına sabit 53h değeri yüklenir, atanır. Benzer şekilde,

**djnz ACC, x1 ;** (djnz: Decrement and Jump Not Zero)

komutu ACC saklayıcısının değerini 1 azaltır ve ACC değeri sıfır olmaz ise program akışı “x1” etiketine dallanır.

Assembly dilinde program yazılabilmesi için mikrodenetleyici donanımının detaylı olarak bilinmesi gereklidir. Bu başlangıçta bir dezavantaj olarak görülsede tasarımcıya daha hızlı koşturulan ve hafızada daha az yer tutan kodlar üretilebilmesi, donanım üzerinde daha fazla kontrol sağlanması gibi önemli avantajlar sunar.

Assembly dili makine diline oranla programlayıcıya büyük kolaylık sağlamasına rağmen özellikle aritmetik işlemlerin assembly'de gerçekleştirilmesi zordur. Bu nedenle günümüzde çeşitli firmalar (Keil, IAR) 8051-ailesi için C derleyicisi geliştirmektedir.

### 8051 Adresleme Modları:

Adresleme modu herhangi bir komutta kaynak ve hedef'in belirtilme şeklini ifade eder. Aşağıda “move-mov” komutu kullanılarak 8051 ailesi için tanımlanmış olan adresleme modları detaylı olarak açıklanmıştır.

- **mov <hedef>, <kaynak>** Adresleme bu şekilde yapılır.
  - ✓ BYTE düzeyinde veri transferi yapılır,
  - ✓ Genel transfer komutudur,
  - ✓ Veri, kaynaktan hedefe doğru transfer edilir.

8051'de 5 çeşit adresleme modları vardır. Adresleme modları kaynağın ve hedefin belirtilme tarzına göre adlandırılır.

#### **4.1. İvedi Adresleme (Immediate Addressing)**

Bu yöntemle istediğimiz veriyi belirlenen adrese yazabilirim yani kaynak bölümü her zaman '#data' şeklinde olmalıdır.

❖ **mov <hedef>, #veri**

“#” işaretini ivedi adresleme olduğunu gösterir. # işaretinin sağındaki sabit değer (Kaynak) <hedef> 'e

atanır. Genel kullanım türleri aşağıda belirtilmiştir.

```
mov a, #data      ; a=ACC=Akümülatör  
mov Rn, #data    ; n=0,1,...,7 = R0,R1,...,R7  
mov @Ri, #data   ; i=0,1 = R0,R1  
mov direct, #data  
mov DPTR, #data16 ; data16, 16-bit veri
```

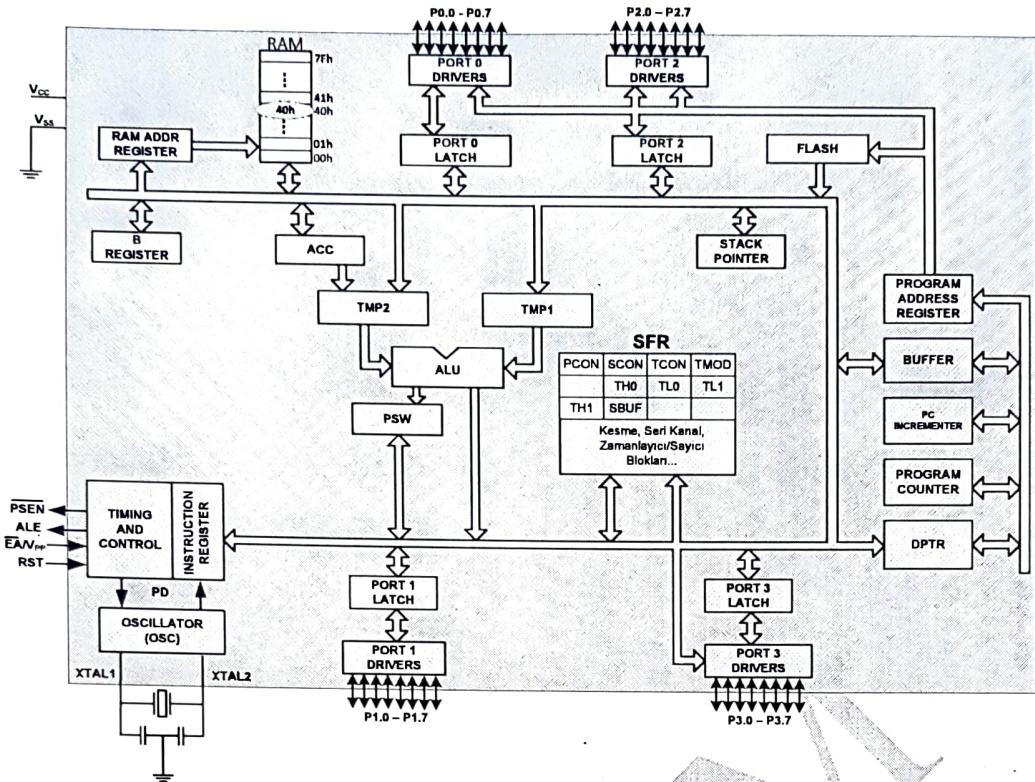
**direct:** RAM' deki doğrudan adreslenebilen bölgeyi temsil eder.

### Örnek verirsek;

- **Mov a,#02h** → 02h bilgisi Akümülatöre yazılmıştır, 02 bilgisinin önündeki “#” işaretinin anlamı kendisinden sonra gelen verinin belirlenen adrese yazılacağını belirler. Bu işlemden sonra Akümülatörün değeri → 00000010<sub>2</sub> olur.
- **Mov 20h,#20h** → RAM' deki 20h adresine 20h bilgisini yazar.
- **Mov r0,#0AAh** → PSW' de belirtilmiş olan BANK'a ait R0 kısmına “aa” yazar.
- **Mov DPTR,#01FE7h** → Data Pointer'a DPTRH (DPTR high) kısmına “1F” ve DPTRL (DPTR low) kısmına ise “E7” bilgisini yazar.
- **Mov p0, #0CCh** → Port 0'a 'CC' bilgisini yazar. Bu durumda

P0.0	P0.1	P0.2	P0.3	P0.4	P0.5	P0.6	P0.7
1	1	0	0	1	1	0	0

\* Böylece port 0 çıkışının bazı pinleri on (Lojik-1) bazı pinleri off (Lojik-0) yapılır.



Şekil 16. "mov 40, #40h" komutu sonrası RAM'in 40h adresi sabit 40h değerini içerir

#### 4.2. Doğrudan Adresleme (Direct Addressing)

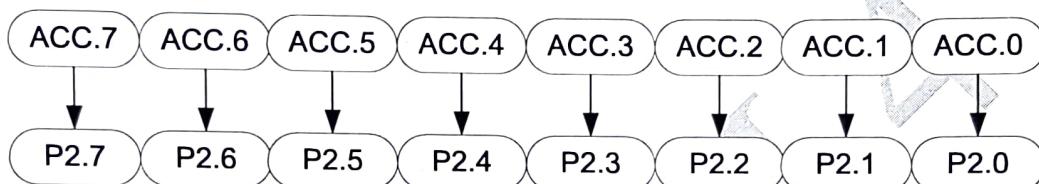
Komuttaki 8 bit adres ile dahili RAM'de veya özel fonksiyon kaydedicileri belirleme işlemidir.

Bu yöntemle bir adresteki veriyi belirlenen diğer bir adrese yazabiliriz. Genel olarak komut tablosu aşağıdaki gibidir:

- **mov a,Rn**
- **mov a,direct**
- **mov Rn,a** ;n=0,1,2..7
- **mov direct,a**
- **mov Rn,direct** ;n=0,1,2..7
- **mov direct,Rn** ;n=0,1,2..7
- **mov direct,direct**
- **mov a,Pn** ;n=0,1,2,3
- **mov Pn ,a** ;n=0,1,2,3

Örnek verirsek;

- **mov a,r7** → PSW' de belirtilmiş olan BANK'a ait R7 kısmına kaydedicisindeki veriyi akümülatöre yazar.
- **mov a,02h** = RAM' de bulunan "02" adresindeki veriyi akümülatöre transfer eder.
- **Mov a, #02h** = Akümülatöre "02" verisini transfer eder.  
!! Sonuç olarak "#" işaretini var ise veri; yoksa adres oluşturur.
- **mov 30h,40h** → RAM' de 40h adresinde bulunan veriyi 30h adresine yazar.
- **mov a,22h** → RAM' de 22h adresinde yer alan bilgiyi akümülatöre yaz diğer bir ifadeyle 22h adresindeki bilgiyi okuma anlamına gelir.
- **mov r5,30h** → RAM' de 30h adresinde yar alan bilgiyi r5 kaydedicisine yaz anlamına gelir.
- **mov P2,a** → Akümülatördeki bilgi doğrudan Port 2 çıkışına yollanır.



- Yukarıdaki komutlara ek olarak bit adresleme için de aşağıdaki komutlar kullanılır:

**mov c,bit**

**mov bit,c**

- Hedef bit ile kaynak bitin herhangi birisi carry flag (elde bayrağı) olmak zorundadır, diğeri doğrudan adreslenebilir bit olabilir.

Örnek verirsek;

**mov c,acc.6** ;Bu komutla elde bayrağına (c) "1" verisi yazılmıştır.

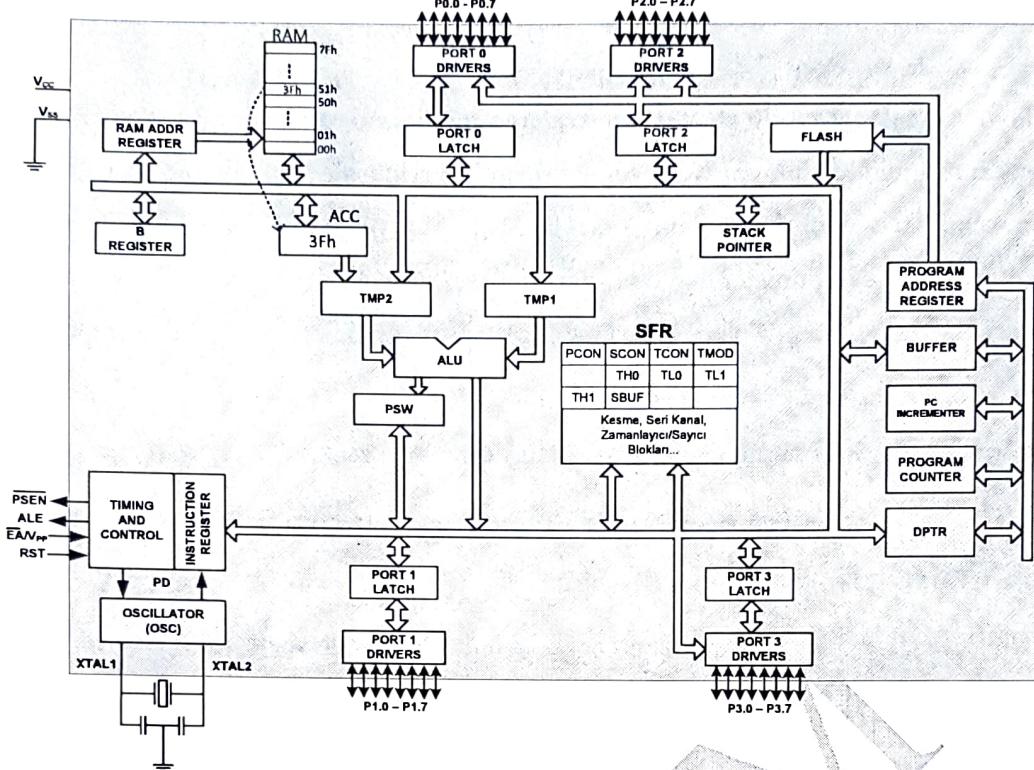
**mov c,p1.2** ;p1.2'nin içeriği veriyi c'ye kopyala.

#### UYARI:

- \* RAM' de bit kaydedilen bölgede byte transferi olabilir ancak byte kaydedilen bölgeden bit transferi olmaz. 'NULL değer'

**ÖNEMLİ:** RAM'ın içindeki bilgiler; enerji kesildiği anda kaybolur.

Şekil 17'de örnek "**mov a, 51h**" komutunun koşturulması sonrasında saklayıcının aldığı değer gösterilmiştir.



Şekil 17. “mov a, 51h ” komutu sonrası ACC sabit 3Fh değerini içerir

#### 4.3.Saklayıcı Adresleme (Register Addressing)

**mov Rn, <hedef> / mov <kaynak>,Rn**

Saklayıcı adresleme modunda kaynak veya hedef byte program çalışması sırasında seçili olan saklayıcı kümelerindeki 8 saklayıcıdan (r0, r1, r2, r3, r4, r5, r6, r7) biridir. Dataları tutmak için kaydedicilerin kullanımını içerir.

**Not:** <hedef> ve <kaynak> aynı anda saklayıcı olamaz. Başka bir ifade ile; “**mov Rn, Rn**” şeklinde bir komut yoktur.

Örnek verirsek;

- **mov r0,a** → Accumulator (Akümülatör) ‘deki bilgiyi r0 kaydedicisine yaz anlamına gelir.
- **mov B,r2** → RAM’ deki r2 kaydedicisindeki veriyi B kaydedicisine yaz anlamına gelir.
- **mov a,r5** → RAM’ deki r2 kaydedicisindeki veriyi a’ya yaz anlamına gelir.
- **mov r4,r7** → Böyle bir kullanım yoktur, kullanıldığı takdirde Assembly hata verecektir.

#### 4.4.Saklayıcı - Dolaylı Adresleme (Register- Indirect Addressing)

Saklayıcı – Dolaylı Adresleme modu en güçlü adresleme modlarından biridir. Bu modda, kaynak veya hedefin adresi komutun içinde açık olarak verilmez. Bunun yerine bir saklayıcının içeriği adreslemede kullanılır. O an seçili saklayıcı kümесinin 2 saklayıcısı, R0 ve R1, dolaylı adresleri belirlemede kullanılabilir. Saklayıcı-dolaylı adresleme modu, veri transferleri çevrimleri kurmada faydalıdır. Ayrıca, Acc ve harici veri hafızası arasında veri aktarmada kullanılır. Genel olarak komut tablosu aşağıdaki gibidir:

mov a,@Ri	;Ri saklayıcısının işaret ettiği RAM adresindeki veri Acc'ye yüklenir.
mov @Ri,a	;A saklayıcısındaki veri Ri saklayıcısının işaret ettiği RAM adresine yüklenir.
mov @Ri,#data	
movx a,@Ri	; Ri saklayıcısının işaret ettiği harici RAM adresindeki veri Acc'ye yüklenir.
movx a,@DPTR	;Harici RAM'deki DPTR'nin göstermiş olduğu adresteki veri akümülatöre yüklenir
movx @Ri,a	
movx @DPTR,a	

#### Örnek:

- **mov a,@r1** ;r1 kaydedicisinin göstermiş olduğu adresteki veriyi a'ya yaz anlamına gelir.
- **movx a,@r0** ;Harici Ram'deki r0'ın göstermiş olduğu adresteki veriyi oku anlamına gelir.
- **movx @DPTR,a** ;Harici Ram'deki DPTR'ın göstermiş olduğu adresteki veri a'ya yaz anlamına gelir.
- **Kod parçası olarak örnek;**

mov 20h,#40h

mov r0,20h

mov a,@r0 ;r0 kaydedicisindeki veri 40h'tir, dolaylı adresleme modunda kullanılan komutla bu bilgi a'ya yazılır.

- **Kod parçası olarak örnek;**

mov psw,#00h

mov r1,#30h

`mov 30h,#20h`

`mov 40h,@r1` ; r1 kaydedicisindeki veri 30h'tır, bu modda kullanılan komutla 40h adresine 20h bilgisi yazılır.

### Örnek Uygulama:

Tüm RAM'deki veriler sıfırlamak istenirse ya 128 satırlık bir program yazılması gereklidir ya da dolaylı adresleme modu kullanılarak aşağıdaki gibi yazılır;

**Mov r0, #07h** → R0 kaydedicisine "07" verisini yaz,

**Mov a, #00h** → Akümülatöre "00" verisini yaz,

X1: **mov@r0, a** → R0'ın gösterdiği adrese akümülatörün içeriğini yaz

**DjnZ r0, X1** → R0 sıfır değerini alana kadar R0'ı bir azalt ve X1'e git.

Böylece sadece 4 satır ile RAM'ın içeriği sıfırlanmış olur.

### **4.5. Saklayıcı - İndisli Adresleme (Register Indexed Addressing)**

Saklayıcı – İndisli Adresleme (Register Indexed Addressing / Base- Plus Index Register Addressing) modunda, hedef veya kaynak adres, bir taban adresi, Acc'nin eklenmesiyle elde edilir. Taban adres DPTR veya PC'nin içeriği olur. Bu adresleme modu, örneğin hafızada tek boyutlu bakma tablosu (Look-up table) olarak adlandırılan özel dizilere erişimlerde kullanılır. Genel olarak komut tablosu aşağıdaki gibidir:

**movc a,@a+DPTR** ; Program Hafıza @(A+DPTR) Hücresin Oku

**movc a,@a+PC** ; Program Hafıza @ (A+PC ) Hücresin Oku

- ✓ **Look-up Table:** RAM'in enerjisi kesildiği zaman içindeki veriler kaybolduğundan dolayı sabit ve daimi olan bilgiler Code bölgesindeki harici belleklere yazılırlar. Bu bilgiler "look-up table" oluşturur.

### Örnek :

- 7-Segment'e istediğimiz sayıyı (örneğin 6) aşağıdaki programla yazdıralım.

`mov a,#06h`

`mov DPTR,#sayigoster`

`movc a,@a+DPTR`

`mov p1,a ; 7-Segment'in p1'e bağlanarak 6 sayısını gösterir.`

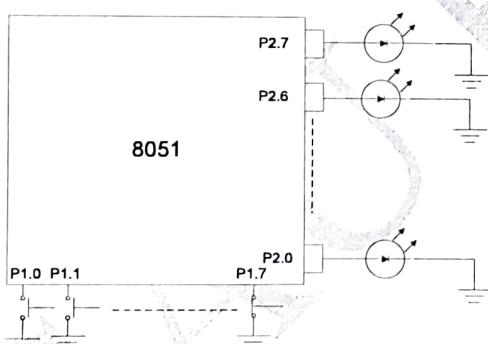
`sayigoster:db 3f,0b,05,04,6b,6d,7d,07,7f,6f,77,7c,39,5e,79,71`

- Vereceğimiz örnekte P1'den okuyacağımız değerin karesini alarak P3'e göndereceğiz.

```
org 00h  
sjmp start  
  
start: mov DPTR,#Tablo  
       mov A,#OFFh  
       mov P1,A  
  
dongu: mov A,P1  
       movc A,@A+DPTR  
       mov P3,A  
       sjmp dongu
```

Tablo: DB 0, 1, 4, 9, 16, 25, 36, 49, 64, 81

### Uygulama:



P1 portuna bağlı 8 adet buton vardır. Butonların durumu, RAM'den okunacak verinin adresini göstermektedir. P1portunun gösterdiği adresin P2 portuna yazan programı oluşturunuz

### ÇÖZÜM:

```
mov ro, p1  
mov a, @ro  
     mov p2, a
```

} Portlar arası bilgi transferi dolaylı adreslemede önce akümülatöre ardından ilgili porta olur.

## Örnek:

```
Org 00h  
sjmp basla  
. . .  
Basla;  
. . .  
Start  
. . .  
end
```

### Yandaki program satırlarını açıklayalım;

**Org:** Bu direktif program yazarken tüm satırların 0000h ‘den itibaren yazılmasını sağlar.

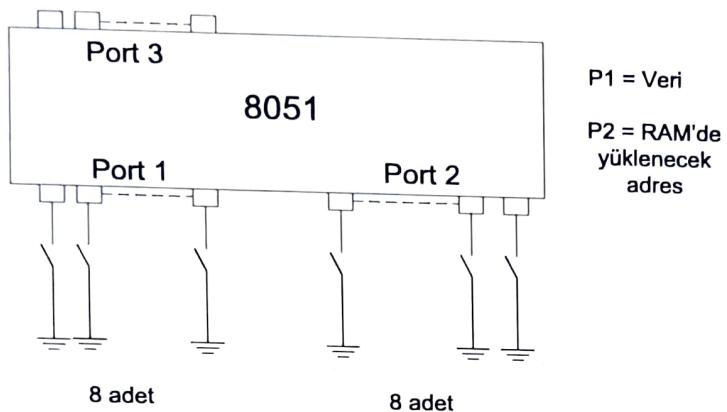
- ✓ Org ve end satırları komut değil derleyici için gerekli direktiflerdir.
- \* Bu programda program her reset aldığında sjmp basla komutu ile basla olan satır gidecektir. İşlemci reset aldığı zaman PC 0000h adresinden başlayacağını unutmayalım.

- ❖ Verilen ekte her bir komutun Code ROM’da kapladığı alan belirtilmiştir. Örneğin sjmp komutu ROM bellekte 2 byte yer tutar. Bu nedenle PC sjmp ve vb. komutlarda birden fazla artış gösterir.

```
→ 0Γj1  
Org 00h  
sjmp basla  
org 00A0h  
. . .  
Basla;  
. . .  
Start  
. . .  
end
```

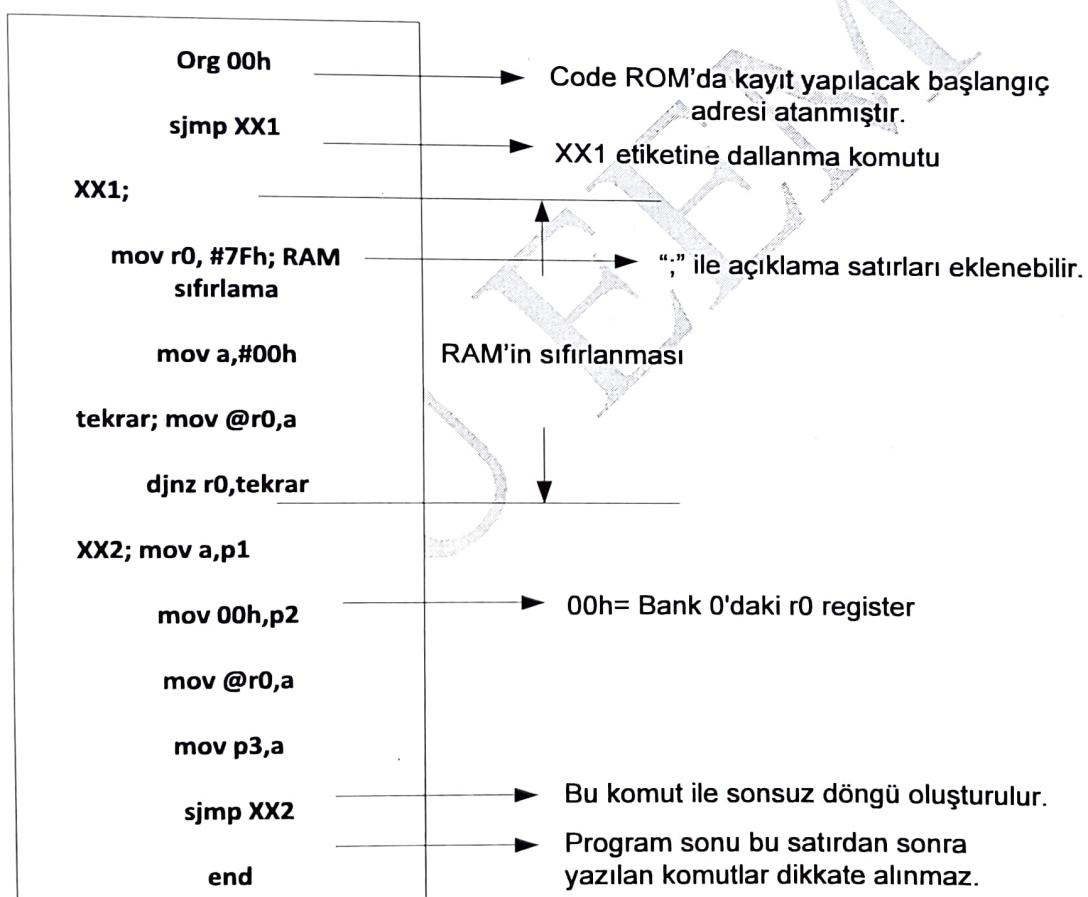
- ➔ Yandaki program parçasında ‘org’ ile ‘sjmp basla’ komutu 0000h’den itibaren Code ROM’a kaydedilir.
- ➔ İşlemci resetlenince program sjmp ile ‘basla’ satırına dallanır.
- ➔ “org 00A0h” satırı ile ondan sonra yazılan program satırları ROM’da 00A0h adresinden itibaren kaydedilir.

## UYGULAMA:



P1 portuna 8 adet anahtar elemanı ile, 8 bitlik veri girişi ve P2'ye 7 adet anahtar ile adres girişi yapılmıştır. P1'den okunan veriyi; P2'de gösterilen RAM adresine yerlestiren ve bu adressteki veriyi tekrar P3'e yazan programı gerçekleştiriniz.

## ÇÖZÜM:



## BÖLÜM 4 8051 KOMUT KÜMESİ

Farklı uzunlukta (1, 2 ve 3 Byte) ve farklı koşturma sürelerine (1,2,3 ve 4 makine çevrimi) sahip 255 komuttan oluşan ve Ek-1'de detaylı açıklamaları ile birlikte verilen 8051 komut kümesi;

- i. Veri transfer komutları,
- ii. Veri işleme komutları,
- iii. Program akışı kontrol komutlarıdır.

### 4.1 Veri Transfer Komutları

8051 mikrodenetleyicisinde dahili veri hafıza (alt-RAM, üst-RAM ve SFR bölgesi) içerisinde veri okuma-yazma için kullanılan komutlardır. Veri transfer komutları, **dahili, harici hafıza ve program hafızadan** olan erişimlerine göre üçe ayrılmaktadır. Byte veri transfer komutları bayrakları etkilemez. Veri transfer komutları (“**mov**”, “**movx**” ve “**move**”) kaynak(taki) veriyi hedefe aktarır (yükler). İşlem sonrası kaynak ve hedefte aynı değer bulunur. Veri transfer komutları verinin taşındığı hafızaya bağlı olarak;

1. Dahili veri hafıza transfer komutları
2. Harici veri hafıza transfer komutları
3. Program hafızadan veri transfer komutları

#### 4.1.1 Dahili Veri Hafıza Transfer Komutları

Dahili veri hafıza transfer komutlarının genel yapısı, “**mov <hedef>, <kaynak>**” şeklidindedir

Tablo 1, dahili Veri Hafıza adres alanı içinde byte boyutundaki ve tablo 2 ise bit boyutundaki veri transferi için kullanılan komutları ve her biri için mümkün olan adresleme modlarını göstermektedir.

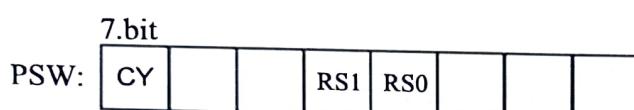
(**Dir:** Doğrudan adresleme, **Ind:** Dolaylı adresleme, **Reg:** Saklayıcı adresleme, **Imm:** İvedi adresleme)

Tablo 1: Dâhili Veri Hafıza Alanına Byte Veri Transfer Komutları

Gösterim	İşlem	Adresleme modları			
		Dir	Ind	Reg	Imm
MOV A ,<kaynak>	A = <kaynak>	*	*	*	*
MOV <hedef>, A	<hedef> =A	*	*	*	
MOV <hedef> ,<kaynak>	<hedef>=<kaynak>	*	*	*	*
MOV DPTR, #data16	DPTR =16-bit ivedi sabit				*

**Tablo 2: Dâhili Veri Hafıza Alanına Bit tabanlı Veri Transfer Komutları**

Gösterim	Açıklama	Bayrak	Byte
<b>MOV C, Bit</b>	Bir bitlik veriyi elde bayrağına yükle	C	2
<b>MOV Bit, C</b>	Elde bayrağındaki veriyi bir bitlik adrese yükle	C	2
➤ <b>Mov &lt;hedef bit&gt;, &lt;kaynak bit&gt;</b> , şeklindeki bit düzeyinde veri transfer komutlarıdır ve bit düzeyindeki bütün transferler “Carry flag-C” elde bayrağı (PSW.7) üzerinden gerçekleşir.			



**CY-C:** CY (veya C) biti aritmetik işlemlerde elde fonksiyonunu gerçekleştirir. Çeşitli komutlar yürütülürken elde bayrağının değeri değişimdir. Elde bayrağı komutlarda çoğunlukla “C” olarak kullanılır.

**Örnek:** P3.7 portunu okuyup 00h adresine yazmak istiyorum;

**Mov c,p3.7** } Bit düzeyinde transferlerde veri mutlaka önce “C” (Carry Biti)’ye gider.  
**Mov 00h,c** }

**MOV <hedef>, <kaynak>** komutu, herhangi iki dahili RAM veya SFR hücreleri arasında, ACC üzerinden geçmeden, veri transferini gerçekleştirir.

- **Örnek:** Mov 07h, A0h; mov P0,P3
- **MOV DPTR, #data16** komutu ile DPTR saklayıcısı bir 16-bitlik ilk değer ile yüklenir. Bu komut, Programa Hafızada yer alan tablolara erişimlerde veya 16-bit harici Veri Hafıza erişimleri için kullanılır.

#### 4.1.2 Harici Veri Hafıza Transfer Komutları

Tablo 3 harici Veri Hafızaya erişen veri transfer komutlarının listesini göstermektedir. Harici Veri Hafızaya erişimde, sadece dolaylı adresleme kullanılabilir Bir-byte veya iki-byte adres seçimi mümkündür. @R<sub>i</sub> ile belirtilen bir-byte adres, seçili Saklayıcı Kümesindeki R<sub>0</sub> veya R<sub>1</sub> saklayıcılarının içeriğidir. İki-byte adres @DPTR ile ifade edilir.

**Tablo 3: Harici Veri Hafıza Alanına Erişen Veri Transfer Komutları**

Adres genişliği	Gösterim	İşlem
8- bit	MOVX A, @R <sub>i</sub>	Harici RAM @R <sub>i</sub> ‘ yi oku
8- bit	MOVX @R <sub>i</sub> , A	Harici RAM @R <sub>i</sub> ‘ yi yaz
16- bit	MOVX A, @DPTR	Harici RAM @DPTR‘ yi oku
16- bit	MOVX @DPTR ,A	Harici RAM @DPTR‘ a yaz

- Tabloda belirtildiği gibi Harici Veri Hafızaya bütün erişimlerde, ACC her zaman veri için ya hedef veya kaynak olmaktadır.
- Harici RAM hafıza okuma ve yazma sinyalleri RD ve WR, sadece bir MOVX komutu yürütülürken aktif olur. Normalde bu sinyaller aktif değildir ve eğer okuma/yazma amaçlı kullanılmayacaklarsa I/O hatları olarak kullanılması mümkündür.

#### 4.1.3 Bakma (Look-up) Tabloları

Mikrodenetleyicinin besleme gerilimi kesildiğinde RAM hafızada bulunan veriler kaybolur. Bazı programlarda program akışı esnasında çeşitli amaçla sabit bazı verilerin (katsayı, vb.) kullanılması gerekebilir. Bu katsayılar off-line olarak hesaplanır ve bir tablo halinde program hafızaya yerleştirilir ve ihtiyaç anında yerleştirildikleri adresden okunur. Tablo 4 Program Hafızada bulunan, bakma (look-up) tablolarından okuma için kullanılan, iki veri transfer komutunu göstermektedir. Bu iki komut, sadece Program Hafızaya erişebildiği için, bakma tabloları sadece okunabilir olup üzerinde değişiklik yapılamaz. **MOVC** komutu sabit kopyala (**MOVE Constant**) anlamındadır.

“**MOVC**” komutu tablo taban adresi olarak **PC** veya **DPTR**’yi kullanır. Taban adrese ACC değere ilave edilerek program hafızadan okunacak adres değeri elde edilir.

**Tablo 4: Look-up table Okuma Komutları**

Gösterim	İşlem
<b>MOVC A, @A+DPTR</b>	Prog. Hafıza @(A+DPTR)Hücreni Oku
<b>MOVC A, @A+ PC</b>	Prog. Hafıza @ (A+PC ) Hücreni Oku

Tablonun kendisi Program Hafızada RET komutundan hemen sonra gelmelidir. Bu çeşit bir tablo, 1'den 255'e kadar numaralandırılmış, 255 taneye kadar veri içerebilir. Sıfır sıra numarası kullanılamaz. Çünkü MOVC komutu yürütülmesi sırasında PC, RET komutunun adresini içermektedir. Sıfır sıra numaralı veri, RET komutunun işlem kodu olacaktır.

İkinci MOVC komutunda, tablo taban adresi PC'de bulunur ve tablo bir altprogram yoluyla erişilir. Bu komut kullanılmadan önce, aşağıdaki kod örneğindeki gibi, tablodan okunacak verinin sıra numarası (indis) ACC' ye yüklenir ve sonra alt program çağrıılır. İleride daha detaylı örnek verilecektir.

**MOV D PTR, #TABLO ;** Tablo adresi datapointer (dptr)'e yüklenir.

**MOVC A, @A+DPTR ;** Bu komut ile dptr'deki 16 bitlik adres ile a'daki veri toplanır. Yeni oluşan veri a'ya yazılır.

**TABLO: DB 00h,01h,02h,....**

## 4.2. Veri İşleme Komutları

Veri işleme komutları Lojik komutlar ve Aritmetik komutlar olmak üzere 2 grupta incelenebilir. Daha önce de belirtildiği gibi Mikroişlemci mimarisindeki ALU (Arithmetci ve Logic Unit) biriminde bu işlemler gerçekleştirilebilir.

Bu komutlar bayrakları etkiler. Aşağıda bu komutlar örnekleriyle anlatılmaktadır.

### 4.2.1. Lojik Komutlar

<i>Giriş</i>	<i>anl</i>	; and işlemini gerçekleştir ve sonucu yükle
<i>Kelime</i>	<i>orl</i>	; or işlemini gerçekleştir ve sonucu yükle
<i>L</i>	<i>xrl</i>	; ex-or işlemini gerçekleştir ve sonucu yükle

“**anl** <hedef>, <kaynak>”, <hedef> ve <kaynak> değerleri “and” lanır, sonuç <hedef>’e yüklenir.

anl	a, Rn	; $(A) \leftarrow (A) \wedge (R_n)$
anl	a, @Ri	; $(A) \leftarrow (A) \wedge ((R_i))$
anl	a, #data	; $(A) \leftarrow (A) \wedge \#data$
anl	a, direct	; $(A) \leftarrow (A) \wedge (\text{direct})$
anl	direct, a	; $(\text{direct}) \leftarrow (\text{direct}) \wedge (A)$
anl	direct, #data	; $(\text{direct}) \leftarrow (\text{direct}) \wedge \#data$
mov	a, #76h	; $(A) \leftarrow 01110110b$
mov	40h, #45h	; $(40h) \leftarrow 01000101b$
anl	40h, a	; $(40h) \leftarrow 01000100b$

“**orl** <hedef>, <kaynak>”, <hedef> ve <kaynak> değerleri “or” lanır, sonuç <hedef>’e yüklenir.

orl	a, Rn	; $(A) \leftarrow (A) \vee (R_n)$
orl	a, @Ri	; $(A) \leftarrow (A) \vee ((R_i))$
orl	a, #data	; $(A) \leftarrow (A) \vee \#data$
orl	a, direct	; $(A) \leftarrow (A) \vee (\text{direct})$
orl	direct, a	; $(\text{direct}) \leftarrow (\text{direct}) \vee (A)$
orl	direct, #data	; $(\text{direct}) \leftarrow (\text{direct}) \vee \#data$

“**xrl** <hedef>, <kaynak>”, <hedef> ve <kaynak> değerleri “ex-or” lanır, sonuç <hedef>’e yüklenir.

xrl	a, Rn	; $(A) \leftarrow (A) \vee \neg (R_n) \vee$
xrl	a, @Ri	; $(A) \leftarrow (A) \vee \neg ((R_i)) \vee$
xrl	a, #data	; $(A) \leftarrow (A) \vee \neg \#data \vee$
xrl	a, direct	; $(A) \leftarrow (A) \vee \neg (\text{direct}) \vee$
xrl	direct, a	; $(\text{direct}) \leftarrow (\text{direct}) \vee \neg (A) \vee$
xrl	direct, #data	; $(\text{direct}) \leftarrow (\text{direct}) \vee \neg \#data \vee$

Lojik komutlar çeşitli amaçlar için kullanılabilir. **anl** ve **orl** komutları, bir kontrol saklayıcısındaki belirli bit'leri **maskeleme** (masking) işleminde faydalıdır. Örneğin aşağıdaki örneklerde PSW saklayıcısının

3. ve 4. bitlerinin (RS0 ve RS1, Saklayıcı bank seçme bitleri) değeri değiştirilmek istensin. Fakat bu bitlerin değeri değiştirilirken saklayıcılardaki diğer bitlerin değeri değişmemelidir. Lojik komutlar kullanılarak bu tip uygulamalar aşağıda gösterildiği gibi rahatlıkla gerçekleştirilebilir.

**Örnek:** PSW' deki diğer bitlerin değeri değiştirilmeden RS0 = 0, RS1 = 0 yapınız.

...

; öncesinde PSW değeri bilinmiyor.

;  $(PSW) = x \ x \ x \ x \ x \ x \ x \ b$  ( $x=0$  veya  $1$ )

anl PSW, #11100111b ;  $(PSW) \leftarrow (PSW) \wedge \#11100111b$

PSW	x	x	x	x	x	x	x	x	x
and									
	1	1	1	0	0	1	1	1	1
PSW	x	x	x	0	0	x	x	x	x

Gördüğü gibi “anl PSW, #data” komutu ile PSW'nin sadece RS0-RS1 bitleri istenilen doğrultuda değiştirilirken diğer bitlerin değeri korunmuş oldu.

**Örnek:** PSW' deki diğer bitlerin değeri değiştirilmeden RS0 = 1, RS1 = 1 yapınız.

...

; öncesinde PSW değeri bilinmiyor.

;  $(PSW) = x \ x \ x \ x \ x \ x \ x \ b$  ( $x=0$  veya  $1$ )

orl PSW, #00011000b ;  $(PSW) \leftarrow (PSW) \vee \#00011000b$

PSW	x	x	x	x	x	x	x	x	x
or									
	0	0	0	1	1	0	0	0	0
PSW	x	x	x	1	1	x	x	x	x

### Temizle (Clear) ve Tersle (Complement) Komutları

Bu komutlar ACC üzerinde işlem yapan saklayıcı-özel komutlardır.

**CLR A** ;komutu, ACC' nin bütün bit'lerini temizler.

**CPL A** ;komutu ise ACC 'nin her bir bit'inin tersini alır.

#### 4.2.2 Aritmetik Komutlar Tablosu

Tablo 5, 8051 ailesinin aritmetik komutlarını özetlemektedir, Bu tablodaki birçok komut, Saklayıcı-Özel adresleme modunu kullanmaktadır.

toplama: **add, addc, inc**

çıkarma: **subb, dec**

çarpma: **mul**

bölme: **div**

Komutların detaylı kullanımı için Tablo-5'i inceleyiniz.

Tablo 5: Aritmetik Komutlar

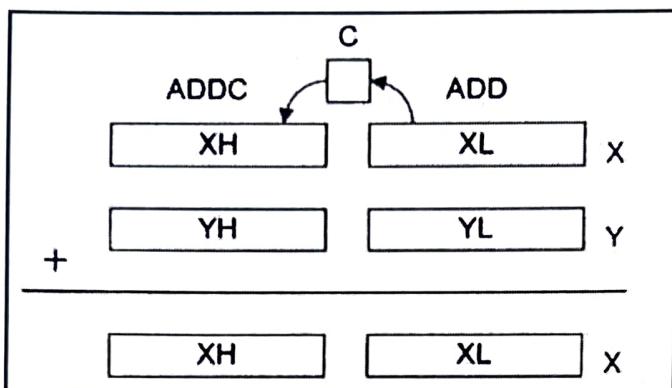
Gösterim	İşlem	Adresleme modları			
		Dir	Ind	Reg	Imm
<b>ADD A ,&lt;byte&gt;</b>	$A = A + <byte>$	*	*	*	*
<b>ADDC A ,&lt;byte&gt;</b>	$A = A + <byte> + C$	*	*	*	*
<b>SUBB A ,&lt;byte&gt;</b>	$A = A - <byte> - C$	*	*	*	*
<b>INC A</b>	$A = A + 1$	Sak. Özel (sadece ACC)			
<b>INC &lt;byte&gt;</b>	$<byte> = <byte> + 1$	*	*	*	
<b>INC DPTR</b>	$DPTR = DPTR + 1$	Sak. Özel (sadece DPTR)			
<b>DEC A</b>	$A = A - 1$	Sak. Özel (sadece ACC)			
<b>DEC</b>	$<byte> = <byte> - 1$	*	*	*	
<b>MUL AB</b>	$B:A = B \times A$	Sak. Özel (sadece ACC ve B)			
<b>DIV AB</b>	$A = \text{Int}(A/B)$ $B = \text{Mod}(A/B)$	Sak. Özel (sadece ACC ve B)			
<b>DA A</b>	Ondalık Ayar	Sak. Özel (sadece ACC)			

#### Toplama ve Çıkarma Komutları (ADD, ADDC, SUBB)

İki tane toplama komutu vardır: **ADD** ve **ADDC** (**C**—elde (carry) ile topla). Her iki komut 2 byte değişkenin toplama işlemini gerçekleştirmektedir. Birinci operand her zaman ACC' dendir. İkinci operand, doğrudan, dolaylı, saklayıcı veya ivedi adresleme modları ile belirtilir. Bir ADD ve ADDC işlemlerinden sonra, bayraklardan üçü (C, AC ve OV) sonuca göre 1 'lenir veya 0 'lanır. ADDC komutu, C bayrağının sonuca eklenmesi farkının dışında, ADD komutu gibidir. ADDC komutu, özellikle uzun tamsayı toplamalarında kullanılır.

4 işlem yaparak sona C'yi sıfırla

Şekil 1' de gösterilen ve aşağıda verilen kod örneğinde, iki 16-bit X ve Y tamsayılarının toplama işleminde ( $X = X + Y$ ), ADD ve ADDC komutlarının kullanımı gösterilmektedir. Her iki sayı, 8-bit düşük (XL ve YL) ve yüksek değerli (XH ve YH) byte'lar olarak işleme alınmaktadır.



Şekil 18 İki 16-bit X ve Y Tamsayılarının Toplama İşlemi ( $X = X + Y$ )

Bu byte'ların saklayıcılarında aşağıda verildiği gibi saklandığını düşünülürse;

Byte	Saklayıcı
XL	78H
XH	79H
YL	7AH
YH	7BH

$X = 1234h$  ve  $Y = 12EFh$  tamsayılarını bu hafıza hücrelerine yükleyen program parçası:

```

MOV 78h, #34h ; XL
MOV 79h, #12h ; XH
MOV 7Ah, #0EFh ; YL
MOV 7Bh, #12h ; YH

```

$XL = XL + YL$  toplamını gerçekleyen program kodu:

```

MOV A, 78h ; A = XL
ADD A, 7Ah ; A = XL + YL
MOV 78H, A ; XL = A

```

Yapılan işlemin bu noktasında, 78h adresli hücre 23h değerini içermektedir ve C bayrağı l'lenmiştir. Şimdi  $XH = XH + YH + C$  işlemini gerçekleştiren program parçasına bakalım:

"Subb" diye bir komut yok <sup>57</sup> puanı kazanmış olsaydı çok iyi

subb a, #23h a = a - 23h - c

```

MOV A, 79h ; A = XH
ADDC A, 7BH ; A = XH + YH + C
MOV 79h, A ; XH = A

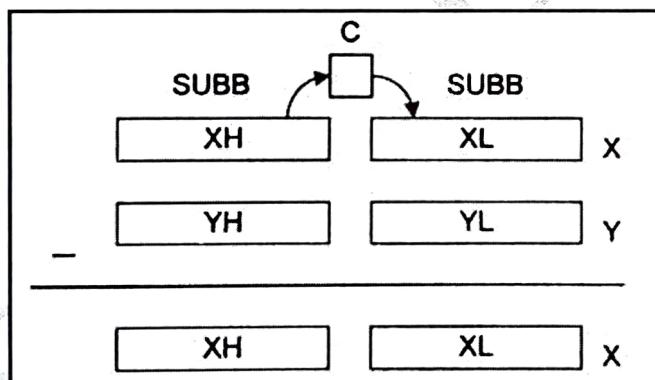
```

Bu programın çalıştırılması sonucu, dahili saklayıcı hücreleri 78h ve 79h'ta 2523h değeri doğru olarak saklanır.

Çıkarma işlemi **SUBB** de, ACC' yi ilk operand olarak kullanır. Doğrudan, dolaylı, saklayıcı veya ivedi adresleme modları ile belirtilen ikinci operand, ACC'den çıkarılır ve sonuç yine ACC' ye yerleştirilir.

C bayrağı, uzun tamsayı çıkarma işlemlerinde, toplama işlemine benzer şekilde kullanılır. Çıkarma işlemi, bir ödünç almaya ihtiyaç duyar ise, C bayrağı 1'lenir. Eğer bir çıkışma işleminden önce, C 1'lenmiş ise, yapılan çıkışma sonucundan bir eksiltilir. Bu sayede, birden fazla byte uzunluğunda olan tamsayıların çıkışma işlemi, en düşük değerli byte'lardan başlayıp yüksek değerlikli byte'lara doğru, peş peşe byte çıkartma işlemi şeklinde yapılır. Bu durumda, ilk çıkışma işleminden önce C bayrağının sıfırlanması gerekmektedir.

Şekil 2' de gösterilen ve aşağıda verilen program örneği, iki 16-bit X ve Y tamsayılarının çıkışma işlemini ( $X = X - Y$ ) göstermektedir. Her iki sayı, 8-bit düşük (XL ve YL) ve yüksek değerlikli (XH ve YH) byte'lar olarak işleme alınmaktadır.



Şekil 19. İki 16-bit X ve Y Tamsayılarının Çıkarma İşlemi ( $X = X - Y$ )

$X = 1234h$  ve  $Y = 1135h$  tamsayılarını hafızaya yükleyen program parçası:

```

MOV 78h, #34h ; XL
MOV 79h, #12h ; XH
MOV 7Ah, #35h ; YL

```

$$\begin{array}{r}
 5d \\
 - 6d \\
 \hline
 -13d
 \end{array}
 \quad
 \begin{array}{r}
 0000\ 0101 \\
 0000\ 0110 \\
 \hline
 1111\ 1111
 \end{array}$$

**MOV 7Bh, #11h ; YH**

**CLR C ; C bayrağını temizle**

XL = XL - YL çıkarma işlemini gerçekleyen program parçası:

**MOV A, 78h ; A = XL**

**SUBB A, 7Ah ; A = XL - YL - 0**

**MOV 78H, A ; XL = A**

Bu noktada 78h adresli hücre FFh değerini içermektedir ve C bayrağı 1'lenir. Şimdi XH = XH - YH - C işlemini gerçekleştiren program parçasına bakarsak:

**MOV A, 79h ; A = XH**

**SUBB A, 7BH ; A = XH - YH - C**

**MOV 79h, A ; XH = A**

Bu programın çalıştırılması sonucu, dahili saklayıcı hücreleri 78h ve 79h'ta 00FFh değeri doğru olarak saklanır.

### Artırma ve Azaltma Komutları (INC,DEC)

Bu komutlar, çevirim sayılarını veya veri işaretçilerini artırma veya azaltma işlemlerinde faydalıdır.

Aşağıda verilen bu komutların kullanıldığıörnekte, dahili Veri Hafızada bulunan X ve Y vektörleri (veri blokları) toplanmaktadır ( $X = X + Y$ ). Bu program parçasında, X ve Y vektörlerinin uzunlukları R3 saklayıcısında olduğu ve ayrıca, X vektörüne R0 ve Y vektörüne R1 saklayıcılarının işaret ettiğini varsayılmaktadır.

**MOV A, @R0 ; X'ten bir byte oku.**

inc @r0

**ADD A, @R1 ; Y byte'ı ile topla.**

00h  
r0

**MOV @R0, A ; Sonucu X'e yerleştir.**

ed

**INC R0 ; Bir sonraki X byte'ına işaret et.**

dec d  
r0  
00h  
@r0

**INC R1 ; Bir sonraki Y byte'ına işaret et.**

## Çarpma ve Bölme Komutları (MUL,DIV)

8051 ailesinin üyeleri donanım çarpma ve bölme birimlerine sahiptir. Bu komutlar, 4 makine çevrimi ile en uzun zamanı alır. Çarpma ve bölme komutları, saklayıcı-özel komutlar olup ACC ve B saklayıcılarını kullanır.

**MUL AB**

;  $(A)_{7-0} \leftarrow (A) \times (B)$  ve  $(B)_{15-8}$



komutu, ACC ve B saklayıcılarındaki iki işaretsiz tamsayıyı çarpar. 16-bit çarpma sonucunun düşük byte'ı, ACC' de ve yüksek byte'ı B saklayıcısında bulunur. Sonuç FFFFh'tan büyük olamaz. Yani C bayrağı hiçbir zaman l'lenmez (etkilenmez). Eğer sonuç FFh'tan büyük ise, taşıma bayrağı OV 1 'lenir. OV bayrağının sıfırlanması, B saklayıcısının 0 olduğu anlamına gelir.

**DIV AB**

;  $(A)_{15-8} \leftarrow (A)/(B)$  ve  $(B)_{7-0}$

komutu, ACC'deki 8-bit işaretsiz tamsayıyı B saklayıcısındaki 8-bit işaretsiz tamsayıya böler. Sonucun tamsayı kısmı ACC' de, kalan kısmı ise B saklayıcısında tutulur. C bayrağı her zaman temizlenir. Taşma bayrağı OV, sıfır bölme durumunu belirtir. Eğer B saklayıcısında bu komuttan önce sıfır bulunursa, sonuç belirsiz olur ve taşıma bayrağı OV 1'lenir.

## Ondalık Ayarlama (Decimal Adjust) Komutu

DA komutu, ikili kodlanmış ondalık (BCD—Binary Coded Decimal) sayıların toplama işleminden sonra kullanılır. Bir BCD sayının her 4-bit'i (nibble) bir ondalık sayıyı belirtir. Yani, her bir 4-bit'in değeri 0 (0000) ile 9 (1001) sayıları arasında olabilir. DA komutu ile yapılan işlem, ACC ve PSW içeriklerine göre, ACC' ye 0, 6, 60h veya 66h ekleme işlemi olarak görülebilir. Aşağıdaki örneklerde, DA komutunun bu özelliği gösterilmektedir.

**DA** ; IF  $[(A_{3-0}) > 9]$   $[(AC) = 1]$  THEN  $(A_{3-0}) \leftarrow (A_{3-0}) + 6$  AND IF  $[(A_{7-4}) > 9]$   $[(C) = 1]$  THEN  $(A_{7-4}) \leftarrow (A_{7-4}) + 6$

**MOV A, #12h** ; A = 12h (BCD)

**MOV B, #29h** ; B = 29h (BCD)

**ADD A, B** ;  $12h + 29h = 3Bh$ , BCD değil!

**DA A** ; ondalık ayarlama, burada sonuca 6 eklenir,  $3Bh + 6 = 41h$  doğru BCD ; sonuç  $(12 + 29)$  bulunur.

**MOV A, #55h** ; A = 55h (BCD)

```

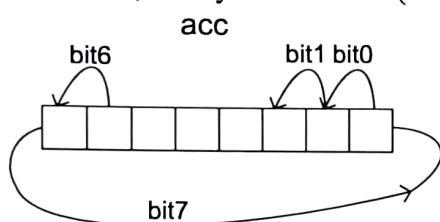
MOV B, #66h ; B = 66h (BCD)
ADD A, B      ; 55h + 66h = BBh, BCD değil!
DA A          ; ondalık ayarlama, burada sonuca 66h eklenir, BBh + 66h = 121h
              ; doğru BCD sonuç (55+66) bulunur ACC, 21h (BCD) içermektedir
              ; ve C bayrağı da 1 'lenmiştir.

```

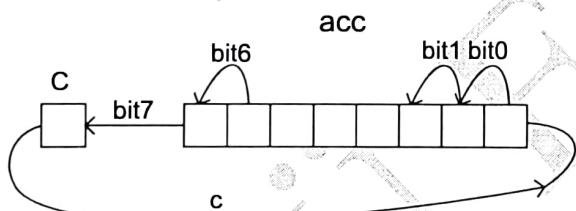
### Döndürme (Rotate) Komutları

Bu komutlar ACC üzerinde işlem yapan saklayıcı-özel komutlardır. 4 tane döndürme komutu vardır. Bu komutlarla, ACC' deki 8-bit sayı veya ACC ve C bayrağındaki 9-bit sayı, sağa veya sola birer bit döndürülür.

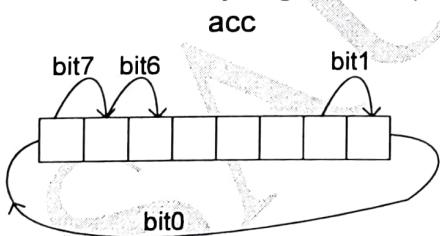
**RL A** ;ACC'yi sola döndür (rotate left, RL)



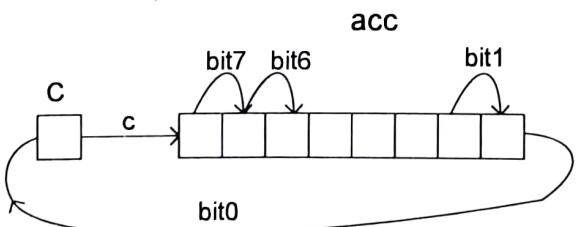
**RLC A** ;ACC'yi elde biti üzerinden sola döndür (rotate left with carry, RLC)



**RR A** ;ACC'yi sağa döndür (rotate right, RR)



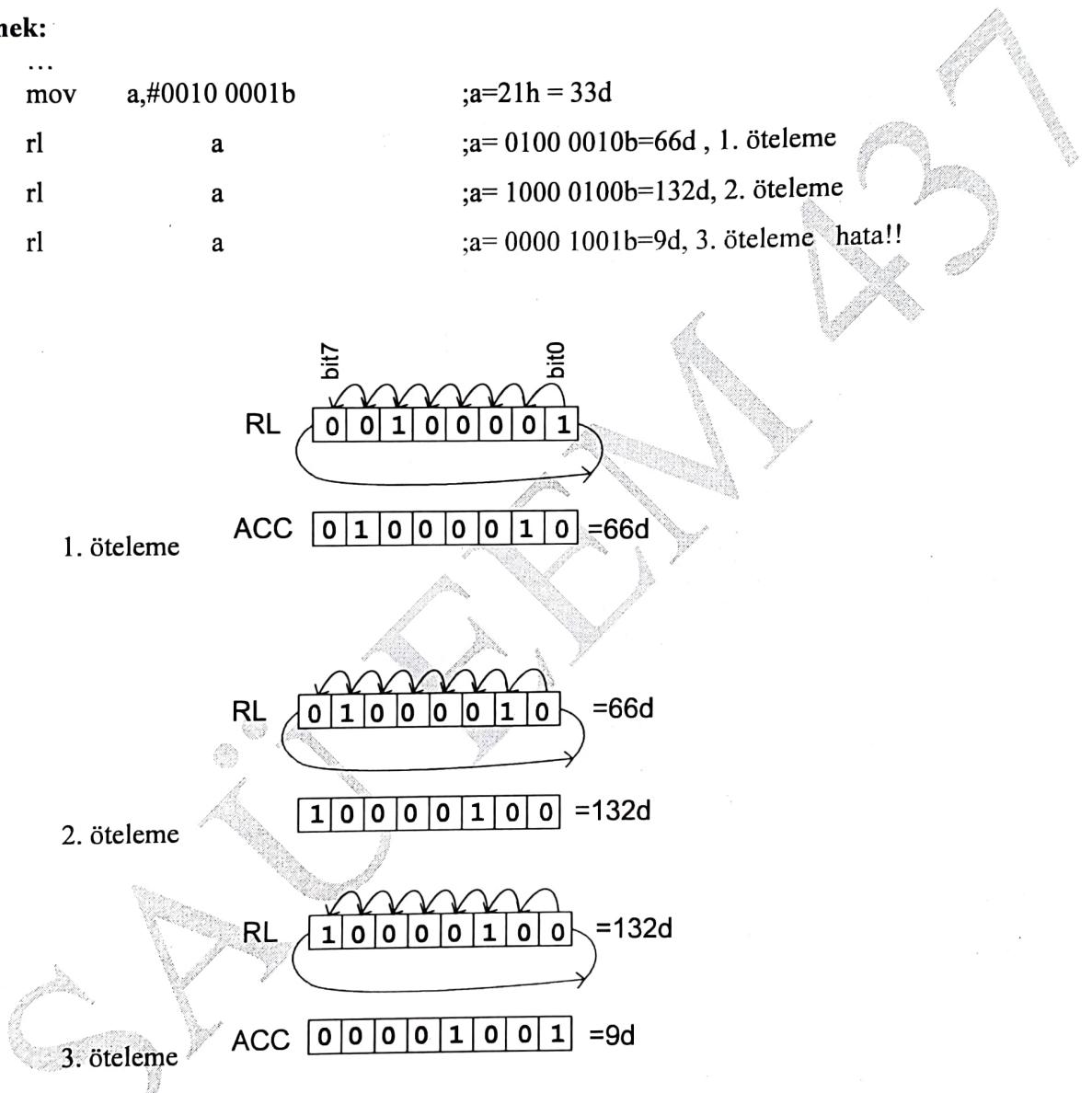
**RRC A** ;ACC'yi elde biti üzerinden sağa döndür (rotate right with carry, RRC)



Döndürme komutları, maskeleme byte'ları oluşturmada, 2'nin katları ile çarpma veya bölme işlemlerinde kullanılır. Bit 0'nın sıfır olması sağlanarak, ACC' yi bir bit sola kaydırma işlemi, ACC' yi 2 ile çarpma işlemine eşittir. Benzeri şekilde, peş peşe iki kere sola kaydırma, ACC' yi 4 ile çarpma demektir. Sağa döndürme ise, 2 ile bölme işlemine eşit olur. İkinin katları ile çarpma ve bölme işlemlerinde, MUL ve DIV komutları yerine, döndürme komutlarının kullanılması işlem hızı açısından önemli kazanç sağlar.

### Örnek:

```
...
mov    a,#0010 0001b      ;a=21h = 33d
rl     a                  ;a= 0100 0010b=66d , 1. öteleme
rl     a                  ;a= 1000 0100b=132d, 2. öteleme
rl     a                  ;a= 0000 1001b=9d, 3. öteleme hata!!
```



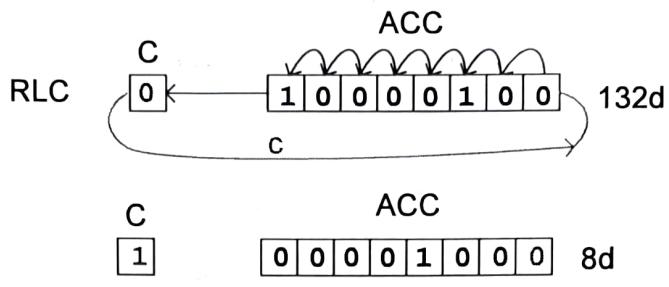
Yukardaki örnekte 3. öteleme sonrasında Bit-0 "1" değerini aldığından çarpma sonucu hatalı olmuştur. Bu durumda aşağıda gösterildiği gibi elde biti üzerinden döndürme yapılarak çarpma işlemi gerçekleştirilebilir. Elde bitinin işlem sonrasındaki değeri dikkate alınmalıdır.

```
mov    a,#0010 0001b      ;a=21h = 33d
rl     a                  ;a= 0100 0010b=66d , 1. öteleme
```

```

rl      a      ;a= 1000 0100b=132d, 2. öteleme
clr      c
rlc      a      ;c = 1, a= 0000 1000b=8d, 3. öteleme

```

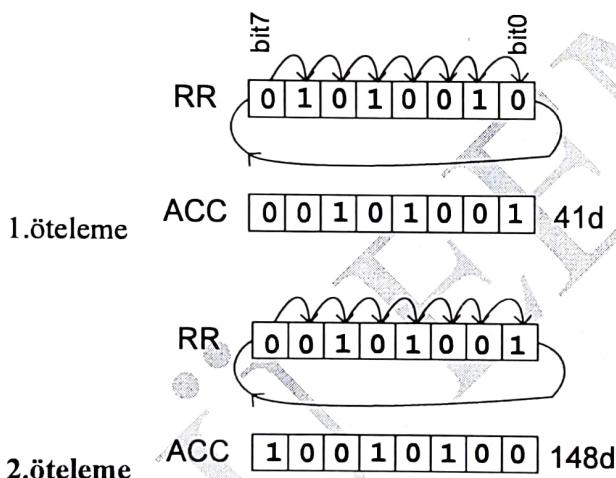


### Örnek:

```

mov    a,#0101 0010b   ;a=82d
rr     a                 ;a=0010 1001b = 41d (82/2=41) 1.öteleme
rr     a                 ;a=1001 0100b=148d hatalı! 2.öteleme
...

```

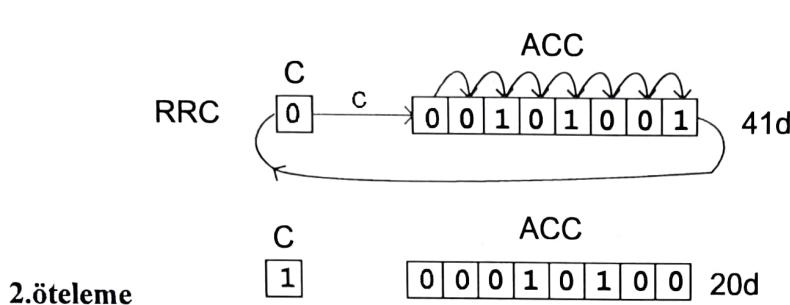
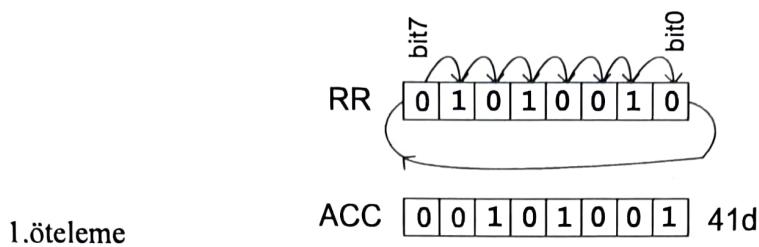


Yukardaki örnekte 2. Öteleme sonrasında Bit-7 “1” değerini aldığından bölümme sonucu hatalı olmuştur. Bu durumda aşağıda gösterildiği gibi elde biti üzerinden döndürme yapılarak bölümme işlemi gerçekleştiriliyor. Elde bitinin işlem sonrasında değeri dikkate alınmaz.

```

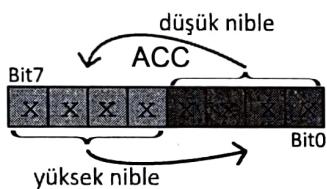
mov a,#0101 0010b      ;a=82d
rr a                   ;a=0010 1001b = 41d (82/2=41) 1.öteleme
clr c
rrc a                  ;a=0001 0100b=20d      2.öteleme

```



### Karşılıklı Değiştirme Komutları

**SWAP A ;** Akümülatorün yüksek nibble'si ile düşük nibble'si karşılıklı yer değiştirir.



Örnek:

```
mov a,#7Bh ;a=7Bh
swap a ;a=B7h
...
```

**XCH A, <byte> ;** ACC ve adreslenen byte verilerini karşılıklı değiştirir.

Akümülatorün içeriği ile byte adreslenen değer karşılıklı olarak yer değiştirir.

```
xch a,direct
xch a,@Ri
xch a,Rn
```

Örnek:

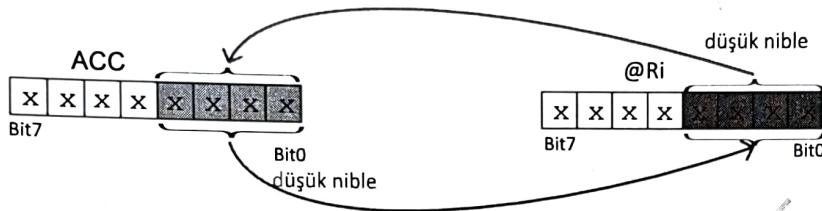
```
...  
mov r0,#25h ;R0'a 25h sabit değerini yükle
```

```

mov 25h,#78h ;25h RAM adresine 78h sabit değerini yükle.
mov a,#0Fh ;ACC'ye 0Fh sabit değerini yükle.
xch a,@R0 ; A←#78h, 25h←#0Fh
xch a,R0 ; A←#25h, R0←#78h
xch a,25h ; A←#0fh, 25h←#25h
...

```

**XCHD A, @Ri ;ACC'nin ve Ri' nin göstermiş olduğu adresdeki verinin düşük nibbleleri karşılıklı yer değiştirir.**



**Örnek:** ...

```

mov r1, #15h ;R1'e 15h sabit değerini yükle.
mov 15h, #79h ;15h RAM adresine, 79h sabit değerini yükle.
mov a, #2Bh ;ACC'ye 2Bh sabit değerini yükle;
xchd a, @R1 ;İşlem öncesi ACC=#2Bh, 15h=#79h, işlem sonrası ACC=#29h,
               15h=#7Bh olur

```

#### 4.2. Bit Transfer Komutları

8051 ailesinin ürünlerini, yoğun bit-tabanlı özelliklere sahiptir. "c", PSW saklayıcısının 7.biti (PSW.7) olan elde (carry, CY) bayrağını temsil eder. Bit transferinde kaynak veya hedeften biri daima "c" dir.

```

mov c, bit
mov bit, c

```

Bit-tabanlı işlemlerde elde bayrağı C, 1 -bit ACC olarak kullanılır. Birçok bit-tabanlı mantık işlemlerinde, C bayrağı, operand'lardan biri ve sonuç için de hedef 1 -bit saklayıcı olarak kullanılır.

**Örnek:**

```

mov c,07h ; bit adreslenebilir RAM bölgesindeki 07h biti c'ye yüklenir
mov c,acc.6 ;acc'nin 6. Biti carry bayrağına taşınır.
mov 0Fh,c ;c değeri RAM'in bit adreslenebilir bölgesinin =Fh bitine taşındı

```

**Örnek:** P3 portunu aşağıda verilen biçimde düzenleyen assembly programını yazınız.

P3:	MSB	1	1	1	1	p1.0	p1.5	p1.2	LSB	p1.6

```

org 00h
sjmp basla

basla:
    mov 20h,#0F0h ;20h = 1 1 1 1 0 0 0 0
    mov c,p1.6
    mov 00h,c ;20h = 1 1 1 1 0 0 0 P1.6
    mov c,p1.2
    mov 01h,c ;20h = 1 1 1 1 0 0 P1.2 P1.6
    mov c,p1.5
    mov 02h,c ;20h = 1 1 1 1 0 P1.5 P1.2 P1.6
    mov c,p1.0
    mov 03h,c ;20h = 1 1 1 1 P1.0 P1.5 P1.2 P1.6
    mov p3,20h ;P3 = 1 1 1 1 P1.0 P1.5 P1.2 P1.6
x: sjmpx
end

```

### Bit Tabanlı Lojik Komutlar

**anl**    c, bit ;  $(c) \leftarrow (c) \wedge (\text{bit})$   
**anl**    c, /bit ;  $(c) \leftarrow (c) \wedge \neg(\text{bit})$     "/bit" ifadesi ilgili bitin değilini belirtir.  
  
**orl**    c, bit ;  $(c) \leftarrow (c) \wedge (\text{bit})$   
**orl**    c, /bit ;  $(c) \leftarrow (c) \wedge \neg(\text{bit})$

#### Örnek:

**anl**    c, 12h ; C ile bit adreslenebilir alanın 12h bitini and'lenip sonuç C' ye yazılır.  
**orl**    c, P2.0 ; C ile P2 portunun 0.biti or'lanıp sonuç C'ye yazılır.  
**anl**    c,/acc.3 ; C ile Acc'nin 3.bitinin tersi and'lenip sonuç C' ye yazılır.

### Temizleme – Setleme Komutları

**clr**    c ;  $c = 0$   
**clr**    bit ;  $(\text{bit}) = 0$   
  
**setb**    c ;  $c = 1$   
**setb**    bit ;  $(\text{bit}) = 1$   
  
**cpl**    c ;  $c = \neg c$

cpl bit ; (bit) =  $\neg$ (bit)

#### 4.3. Program Akışı Kontrol Komutları

Dallanma komutları, mikrodenetleyicinin yürütme sırasında farklı işlemler yapmasını yönlendiren komutlardır. Örneğin, bir tuşun durumuna veya bir kontrol sinyaline göre, bir motorun durdurulması veya çalışmasının devamına karar verme gibi işlemlerde dallanma komutları kullanılır.

Bir dallanma komutu ile PC'nin içeriği değişip program akışı da değişir. Her komut okuma çevriminde, PC, bir sonraki komuta işaret edecek şekilde yenilenmektedir. Normalde bir sonra yürütülecek komut, hafızada o anki komuttan sonra gelen komut olmaktadır. PC'nin değişmesi durumunda ise PC'deki adresinden itibaren program akışı devam eder. Yani hafızada başka bir program alanına dallanma gerçekleştirilmiş olur.

Dallanma komutları, **JMP** (Jump), **CALL** ve **RET** (RETurn) komutlarıdır. Bir JMP komutu sadece PC'yi değiştirmektedir. Dallanma komutları, **durumdan bağımsız (unconditional)** ve **duruma bağımlı (conditional)** olmak üzere ikiye ayrılır. 8051 ailesinin mikrodenetleyicileri, 3 tane durumdan bağımsız dallanma komutuna sahiptir: **SJMP**, **AJMP** ve **LJMP**.

##### 4.3.1 Durumdan Bağımsız Dallanma Komutları

Aşağıda **durumdan bağımsız (unconditional)** dallanma komutlarının bir özeti verilmektedir. Bu komutların her biri ile gerçekleşen program akışındaki dallanma, PC içeriğinin değişmesiyle olur. Aşağıda verilen ilk 3 dallanma komutu birbirine benzemektedir. Verilen en son indisli dallanma komutu ise, ACC ile DPTR'in içeriklerini toplayıp bir sonraki okunacak ve yürütülecek komutun adresini hesaplayan güçlü bir komuttur.

**sjmp** <adres>  
**ajmp** <adres>  
**ljmp** <adres>  
**ljmp** @a+dptra

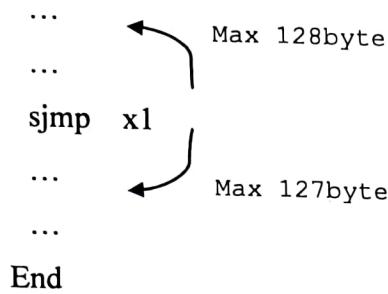
##### Kısa ve Uzun Dallanmalar (SJMP,LJMP)

Kısa dallanma komutları (**sjmp**), genelde bir altprogramın içinde kullanılır. Bu komutlarda dalledilecek alan, dallanma komutunu takip eden 128 byte ilerisi ile 128 byte gerisi arasında sınırlıdır.

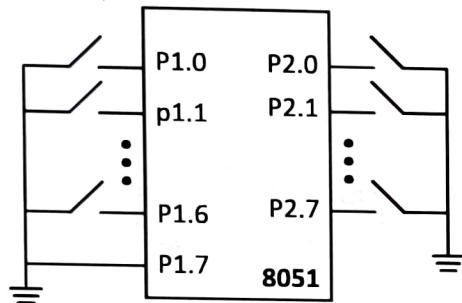
##### Örnek: SJMP

```
org 00h  
sjmp basla
```

basla:



**Örnek:** P2 portundan okunan veriyi acc'ye taşıyan program kodunu yazınız.



```
org 00h  
sjmp x2  
x2: mov a, p2  
x1: sjmp x1 ; sonsuz döngü  
end
```

“x1: sjmp x1” komut satırı program akışını sürekli olarak x1 adresine yani kendi üzerine yönlendirir. Bu tip döngüler, normal çalışma durumunda program akışı hiçbir zaman buradan çıkamayacağından sonsuz döngü olarak adlandırılır. Sonsuz döngülerden sadece daha sonra bahsedilecek olan “kesme (interrupt)” olması durumunda çıkışılabilir. Gelen kesmeye cevap verildikten sonra program akışı tekrar sonsuz döngüye döner.

Dallanma adresi, bir sonraki komuta göre göreceli (relative) ofset tanımlı olduğu için, program kodu başka bir adrese tekrar taşınabilir durumdadır (relocatable code). Bir program veya bir blok kod, program hafızada yerleştirildiği yerden bağımsız olarak doğru çalışıyorsa, tekrar yerleştirilebilir (relocatable) diye adlandırılır. Kodun tekrar yerleştirilebilir olma özelliği, dallanma komutları kullanıldığında önemli olur. Eğer bir dallanma komutu, program akışını Program Hafızada belli bir adrese yönlendirirse, dallanma adresinde geçerli bir kodun olması, programcının sorumluluğundadır. Kısa dallanma komutları kullanan bir kod bloğunun, hafızada bir başka adrese taşınması durumunda, program,

yerleştirildiği yerden bağımsız olarak, düzgün çalışacaktır. Çünkü, bir dallanma adresi, sonraki komuta göre, göreceli ofset olarak hesaplanır. Aşağıda verilen program parçasına bakılırsa:

```
ORG 8000h  
MOV C, P0.0  
MOV P1.0, C  
LJMP 8000h ; tekrar
```

Bu program 8000h adresinden itibaren yerleştirilmelidir. Çünkü, son komut 8000h adresine bir dallanma yapar. Eğer program bir başka adrese yerleştirilseydi doğru çalışmazacaktı. Şimdi de aşağıdaki değişiklik ile programa bakalım:

```
ORG 8000h  
START:  
MOV C, P0.0  
MOV P1.0, C  
SJMP START ; tekrar
```

Bu durumda başlangıç (ORG-origin) adresi değiştirilse de program doğru olarak çalışacaktır. Yukarıda verilen üç komutun her biri 2 byte uzunluğundadır. Program başlarken PC 8000h değerindedir ve sırasıyla, 8002h ve son komut yürütülmeden 8004h değerlerini alır. Son komut da 2 byte uzunluğunda olduğu için, PC artırılarak 8006h olur. Bununla beraber, son komut bir dallanma komutu olduğu için, PC değerinin göreceli 6 byte gerisindeki adres olan 8000h değeri PC'nin yeni değeri olur. Böylece, işlemci sonsuz bir çevirim içinde çalışmasına devam eder. Eğer program başlangıç adresi değiştirilse, örneğin, 9000h yapılsa; program yine doğru çalışacaktır. Komutlar 9000h, 9002h ve 9004h adreslerinden başlar ve son komut yürütüldüğünde, PC 9000h adresinden başlayarak çevirim tekrar eder. Programın kısa dallanmalı ikinci şekli, Program Hafızada tekrar yerleştirilebilir olup birinci verilen örnek hafıza-özeldir.

Yukarıdaki örnekler genellestirecek olunursa, uzun dallanmalar kullanan programlar hafıza-özeldir. Buna karşın, kısa dallanmalar kullananlar ise tekrar yerleştirilebilir programlardır. Uzun dallanmalara, bilhassa, kod boyunun kısa dallanmaların alanı dışına taşması durumlarında ihtiyaç vardır. LJMP komutu ile 64KByte'lık program hafızanın tamamında dallanmalar yapılabilir.

## Mutlak (absolute) Dallanma

Mutlak (absolute) dallanma komutu (AJMP), 2KByte'luk alan içerisinde derleme ve bağlama işlemlerini ayırmaya bir alternatif sunarak, kod için bir çeşit tekrar yerleştirilebilirlik sunmaktadır. Mutlak dallanmalar, dallanma adresinin en düşük değerli U-bit'ini tanımlar. Böylece, dallanma alanı, Program Hafıza'da aynı 2K'luk bir blok içinde olur. Mutlak dallanmalar ve kısa dallanmalar kullanan herhangi bir program segment'i, 0, 800h, 1000h, 1800h, ...., 7800h adresli başlangıçlardan herhangi bir 2K'luk bir bloğa yerleştirilebilir. Kontrol yazılımlarının büyük bir kısmı, 2K'luk bir hafıza içine sığabileceğinden, mutlak dallanmalar, 32 farklı adrese yerleştirilebilen ve en az çaba ile, küçük program bloğu yazmak için pratik bir yol sağlar.

### Örnek:

```
org      00h
sjmp    basla
basla:
...
ajmp  x1
...
X1: ...
End
```

**Max 2Kbyte**  
Burada sjmp kullanılırsa program hata verir.

İndisli Dallanma *Belirli bir sona*

İndisli dallanma komutu, LJMP @A+DPTR, bir dallanma tablosu kurmak için kullanılabilir. Bu komutun kullanıldığı aşağıdaki programı incelenirse:

```
ORG 8000H
MOV A, 0           ; ACC` ye bir işlem değeri yükle ( 0-3 arasında )
ANL A, 3          ; 0,1,2 veya 3 olduğundan emin ol.
RL   A             ; 2 ile çarp.
RL   A             ; 2 ile tekrar çarp.
MOV DPTR, SECENEKLER ; Tabloya işaret et.
JMP  @A+DPTR      ; Tabloda bir girişe dallan.
```

## SECENEKLER:

MOV B, P0 ; işlem 0

SJMP DEVAM

MOV B, P1 ; işlem 1

SJMP DEVAM

MOV B, P2 ; işlem 2

SJMP DEVAM

MOV B, P3 ; işlem 3

## DEVAM:

MOV A, P1 ; Port 1'i oku ve A' ya yerleştir.

MUL AB ; Port 1 ve Port n'i çarp, n program başlangıcındaki  
;ACC'nin içeriğidir.

Bu program, P1'in içeriği ile, program başında belirlenen, ACC'nin değerine bağlı olarak P0, P1, P2 veya P3'ün içeriklerini çarpmaktadır. Seçenekler tablosunun, her birinde MOV ve SJMP komutlarını içeren, 4 farklı giriş noktası vardır. Bu iki komut 4 byte uzunluğunda olduğundan, ACC, 4 ile çarpılır. Bu durumda en fazla 64 seçenek mümkündür. DPTR, seçenekler tablosunun taban adresi ile yüklenir. İndisli dallanma komutu, program akışını istenen MOV komutuna yönlendirir. Program daha sonra, DEVAM etiketli adresten itibaren devam eder.

### 4.3.2 Duruma bağımlı dallanma komutları

Koşullu dallanma komutlarının hepsinde test edilen <koşul> sağlanıyor ise program akışı belirtilen adrese dallanır. Koşul sağlanmıyor ise program akışı bir alt satırda başka bir ifadeyle bir sonraki komuttan devam eder.

~~ZERO~~  
JZ <adres> ; (jump if acc == 00h, akümülator=00h ise <adres>' e dallan)  
JNZ <adres> ; (jump if acc ≠ 00h, akümülator=00h değil ise <adres>' e dallan)  
JC <adres> ; (jump if c == 1, taşıma bayrağı lojik '1' ise <adres>' e dallan)  
JNC <adres> ; (jump if c ≠ 1, taşıma bayrağı lojik '0' ise <adres>' e dallan)  
JB <bit>, <adres> ; (jump if bit = 1, belirtilen bit lojik '1' ise <adres>' e dallan)  
JNB <bit>, <adres> ; (jump if bit ≠ 1, belirtilen bit lojik '0' ise <adres>' e dallan)  
JBC <bit>, <adres> ;(jump and clear bit if bit=1, belirtilen bit '1'ise biti sıfırla ve <adres>'e dallan)

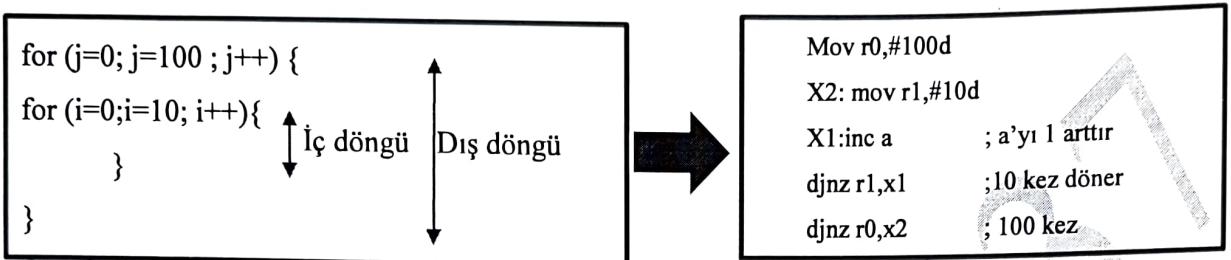
\*CJNE <byte1>, <byte2>, <adres> ; <byte1> ve <byte2> değerlerini karşılaştır ve eğer bu iki

değer  
*compare jump not equal* ; eşit değil ise <adres>'e dallan.

"CJNE" komutunun aşağıda belirtilen 4 farklı kullanım biçimini vardır.

cjne a,direct, adres  
 cjne a,#data, adres  
 cjne Rn,#data, adres  
 cjne @Ri,#data, adres

**DJNZ <byte>, <adres>** ; (DJNZ: decrement and jump if not zero) <byte> değerini 1 azalt, yeni değeri 0 ;değil ise <adres>'e dallan. Bu komut ile C'deki gibi iç içe döngüler oluşturulabilir.



Örnek:

```

...
mov a,#57h           ;ACC'ye 57h sabit değerini yükle
jz x1                ;ACC=57h, koşul sağlanmıyor.. sonraki komuttan devam et
mov r0, #7Bh          ;R0'a 7Bh sabit değerini yükle.
clr a                ;ACC'yi sil (ACC=00h).
jnz x2               ;ACC=00h, koşul sağlanmıyor.. sonraki komuttan devam et
mov r1, #03h          ;R1'e 03h sabit değerini yükle.
jz x3                ;ACC=00h, koşul sağlanıyor.. belirtilen adrese (x3)dallan
...
...
x1: ...
...
x2: ...
...
x3: ...
...
end.
    
```

; buraya kadar olan komutlar icra edilmedi..  
;program akışı buradan (x3 adresinden) devam eder..

Örnek: ...

```

...
mov a, #75h           ; A'ya 75h sabit değerini yükle
mov 40h, #2Ah          ;40h RAM adresine 2Ah sabit değerini yükle
mov R1, #40h            ;R1 reg. 40h sabit değerini yükle
cjne a, #75h, x1        ;ACC=75h, koşul sağlanmıyor.. sonraki komuttan devam et
clr a                  ;ACC=0h
cjne @R1, #1Ah, x2      ;((R1))=2Ah, koşul sağlanıyor.. belirtilen adrese (x2) dallan
cjne 40h, #1Bh, x3
...
...
x1: ...
...
    
```

```

; buraya kadar olan komutlar icra edilmedi..
x2: ...
    ...
    ...
;program akışı buradan (x2 adresinden) devam eder..
x3: ...
    ...
    ...
end.

```

### Örnek:

```

org      00h
sjmp    x1
x1:    mov     r0, #7Fh      ; R0 saklayıcısına 7Fh sabit değerini yükle.
x2:    mov     @R0, #0h      ; R0'ın işaret ettiği adresse (7Fh adresine) 0h sabit değerini yükle,
        djnz   r0, x2      ; R0'ın değerini 1 azalt, (R0=7Eh oldu!) R0≠0h olduğundan X2 adresine
                            ; dallanılır.. R0'ın işaret ettiği adresse (7Eh adresine) 0h sabit değerini yükle,
                            ; R0'ın değerini 1 azalt, (R0=7Dh oldu!) R0≠0h olduğundan X2 adresine
                            ; dallanılır..
...
...
x:     sjmp   x       ;sonsuz döngüde kal
end

```

Bu şekilde yukarıda verilen 3 satırlık komut dizisi ile ;128-Byte'luk RAM alanının tamamına 0h değeri yazılarak RAM temizlenmiştir. Görüldüğü gibi dolaylı adresleme ve koşullu dallanma komutları kullanılarak uzun program kodları son derece kısa bir şekilde yazılmaktadır. Böylece hem esnek bir yazılım yapılmış hem de program hafızadan çok ciddi alan tasarrufu yapılmış olur. Şöyleki, eğer yukarıda verilen RAM temizleme işlemi her bir adrese teker teker 0h değerinin yüklenmesi şeklinde yapılmak istense idi,

```

org      00h
sjmp    x1
x1:
    mov     7Fh, #0h
    mov     7Eh, #0h
    mov     7Dh, #0h

```

...

128-Byte'lık alan için her biri program hafızadan 3-Byte'lık alan işgal eden "mov direct, #data" komutunun 128 adet kullanılması gerekiydi. Program hafızadan toplamda  $128 \times 3 = 384$ -Byte sadece RAM temizleme işlemi için kullanılmış olacak. Oysa aynı işlem

```
x1: mov r0,#7Fh  
x2: mov @R0,#0h  
djnz r0,x2
```

satırları ile sadece 6-Byte'lık program hafıza alanı kullanılmış olur.

Not: Her bir komutun program hafızada kapsadığı Byte alanı farklıdır. Komut kümesinde her bir komut için gerekli açıklayıcı bilgiler verilmiştir.

#### 4.2. 8051 İçin Assembly Program Yapısı

Büyük-küçük harfe duyarlı olmayan (Case-Insensitive) assembly dilinde genel program yapısına ait şablon aşağıda verilmiş ve programın her bir bileşeni incelenmiştir. Komutlarda Türkçe karakter kullanılamaz.

```
EQU katsayi1      54h  
EQU katsayi2      65d  
org 00h  
sjmp basla  
  
org 03h  
sjmp kesme0  
basla: setb ea ;açıklama  
set ex0  
setb it0  
  
----- ;açıklama  
acall alt_1  
  
ajmp basla  
alt_1: ----- ;açıklama  
-----  
ret  
kesme0: ----- ;açıklama  
-----
```

reti

Tablo1: DB “0123456789ABCDEF”

Tablo2: DB 43h,54h,55h

Tablo3: DB “sakarya”

end

## Direktifler

Assembly dilinde değişken tanımlanması, program hafızanın kullanılmaya başlanacağı adresin belirtilmesi, program sonunu belirtilmesi vb. amaçlar için kullanılan, komut olarak değerlendirilmeyen çeşitli direktifler vardır. Bu direktiflerin yaygın olarak kullanılanları aşağıda kısaca açıklanmıştır.

“**equ**” (equal = eşittir) program içerisinde çok fazla tekrar edilen çeşitli değerlerin ve adreslerin anlaşılır olması ve esnekliğin artırılması amacıyla sayısal bir değerin herhangi bir simge adına atanmasını sağlar.

“**org**” (origin = orijin, başlangıç) program kodunun program (flash) hafızada yerleştirilmeye başlanacağı adresi belirtir. Program kodu çoğunlukla 0000h adresinden itibaren (org 00h) yerleştirilmeye başlanır. Tek bir program içerisinde birden fazla org kullanılabilir.

“**db**” (Define Byte) Program hafızada saklanarak gerektiğinde buradan okunabilecek sabit değerlerin tanımlanmasını sağlar.

“**end**” derleyiciye programın sonunu belirtir.

Assembly dilinde herhangi bir satırda “;” den sonraki ifadeler derlemeye tabi tutulmaz. Dolayısı ile program içerisinde yapılacak kısa açıklamalar “;” den sonra verilebilir.

Assembly dilinde yazılan bir programın belirli bir zaman sonra programı yazan kişi tarafından bile takip edilmesi ve anlaşılması çok zordur. Bu nedenle program yazılrken verilen kısa açıklamalar programın takibi ve anlaşılabilirliği açısından önemlidir.

## BÖLÜM 5 ALT PROGRAM

### 5.1. Duruma Bağlı Dallanma ve Alt Program

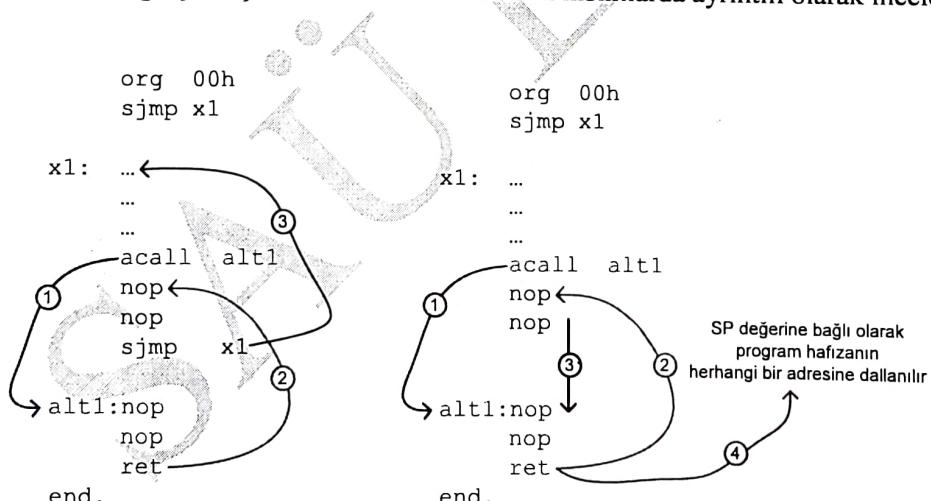
Bir programın yazımı esnasında, program içerisinde birden fazla koşturulacak olan kodlar olabilir. Bu kodların gerekli olduğu her yerde tekrar yazılması program hafızanın şişmesine ve programın uzayarak takip edilebilirliğinin azalmasına neden olur. Bu nedenle birden çok tekrar eden algoritma parçalarının alt programlar halinde yazılması uygundur. Benzer şekilde ana program içerisinde sadece 1 kere koşturulacak olan çalışma ayarları gibi kısımlarında alt programlar halinde yazılması ana programın daha sade ve takip edilebilir olmasını sağlar. Alt programlar ana program içerisinde gerekli hallerde çağrılarak koşturulur. Alt programdan çıktıktan sonra programın akışı alt programın çağrııldığı seviyeye geri döner. Alt program çağrıma ve alt programdan dönüş komutları:

```
acall <alt program ismi> ; 2-kByte alan içerisinde  
lcall <alt program ismi> ; 64-kByte alan içerisinde  
ret ; alt programdan çıkış
```

Alt programlar ilk “ret” komutunu görene kadar koşmaya devam eder. İlk “ret” komutu ile koşturulmakta olan alt programdan çıkarılır. Programın akışı alt programın çağrııldığı komuttan sonraki komutun koşturulması ile devam eder. Aşağıda alt program içeren örnek bir program üzerinde komut satırllarının işletilme sırası gösterilmiştir.

#### SP Saklayıcısı

Programın içası esnasında alt programlardan veya kesme hızmet alt programlarından çıkışılıp ana programa geri dönülürken geri dönüş adresinin tutulduğu, PUSH ve POP komutları ile veri yazılmış okunan saklayıcıdır. Yığın işaretçisinin (Stack Pointer, SP) hatalı kullanımı programın çökmesine istenmeyen durumların gerçekleşmesine neden olur. İleriki kısımlarda ayrıntılı olarak incelenecaktır.



Ana program akışının alt programa kontrollsüz bir şekilde (“acall” veya “lcal” komutları kullanılmadan) erişilmesine izin verilmemelidir. Aksi halde programın çökmesi söz konusudur. Yukarıdaki örnekte “sjmp x1” komutu ile ana programın sonsuz bir döngü içerisinde devam etmesi sağlanmıştır. Böyle bir döngü kurulmaması durumunda programın akışı kontrollsüz bir şekilde “alt1” alt programına erişecek ve “ret” komutunun koşturulması sonrasında programın kontrolü-akışı yığının o anki değerine bağlı olarak program hafıza içerisinde herhangi bir adrese dallanacaktır. Başka bir ifade ile yazılım çökecektir.

## 5.2. Alt Programdan Ana Programa Dönüş Adresine Donanım Tarafından Hesaplanması

### ACALL <alt program ismi>

“ACALL” komutu program hafızada 2-Byte yer işgal eder. Koşulsuz olarak isim ile belirtilen altprogramı çağırır. PC, yiğita atılır. Bir sonraki komuta işaret eden adres, önce düşük byte olmak üzere yiğita atılır. Altprogram program hafızasının 2K’lık bloğu içinde olmalıdır. Bu komut yürütülmesi sırasında donanım tarafından otomatik olarak aşağıdaki işlemler gerçekleştirilir.

(PC)  $\leftarrow$  (PC)+2 → acall komut 2byte lik olduğunu gösterir  
 (SP)  $\leftarrow$  (SP)+1  
 ((SP))  $\leftarrow$  (PC<sub>7..0</sub>) → dönüş adresine SP ye saklayacaktır  
 (SP)  $\leftarrow$  (SP)+1  
 ((SP))  $\leftarrow$  (PC<sub>15..8</sub>)  
 (PC)  $\leftarrow$  alt program adresi

1 (PC)  $\rightarrow$  (PC+2)  
 2 (SP)  $\rightarrow$  (SP)+1  
 3 @SP  $\leftarrow$  PCL  
 4 (SP)  $\rightarrow$  (SP)+1  
 5 @SP  $\leftarrow$  PCH  
 6 PC  $\rightarrow$  gideceği adresten

SP’nm  
gideceği  
adres

Yukarıda belirtildiği gibi bu komutun icrasında öncelikle PC değeri 2-Byte (acall komutu 2-Byte uzunluğundadır) artırılarak alt programdan dönüldüğünde program akışının devam edeceği adres değeri (yani “acall” komutundan sonraki komutun adresi) PC’ye yüklenir. Daha sonra bu adres değerinin düşük (PC<sub>7..0</sub>) ve yüksek (PC<sub>15..8</sub>) Byte’ı RAM hafızada yiğina tahsis edilen alana aktarılır. Dönüş adresi saklandıktan sonra PC’ye dallanılacak olan adres değeri (alt programın başlangıç adresi) yüklenerek alt programa dallanma gerçekleştirilir.

### LCALL <alt program ismi>

“LCALL” komutu program hafızada 3-Byte yer işgal eder. Alt program 64K’lık program belleğinin herhangi bir adresinden başlayabilir. Koşulsuz olarak isim ile belirtilen altprogramı çağırır. Komutun icrası “ACALL” komutu ile aynıdır. Sadece PC değeri 3-Byte artırılır.

(PC)  $\leftarrow$  (PC)+3  
 (SP)  $\leftarrow$  (SP)+1  
 ((SP))  $\leftarrow$  (PC<sub>7..0</sub>)  
 (SP)  $\leftarrow$  (SP)+1  
 ((SP))  $\leftarrow$  (PC<sub>15..8</sub>)  
 (PC)  $\leftarrow$  alt program adresi

### RET

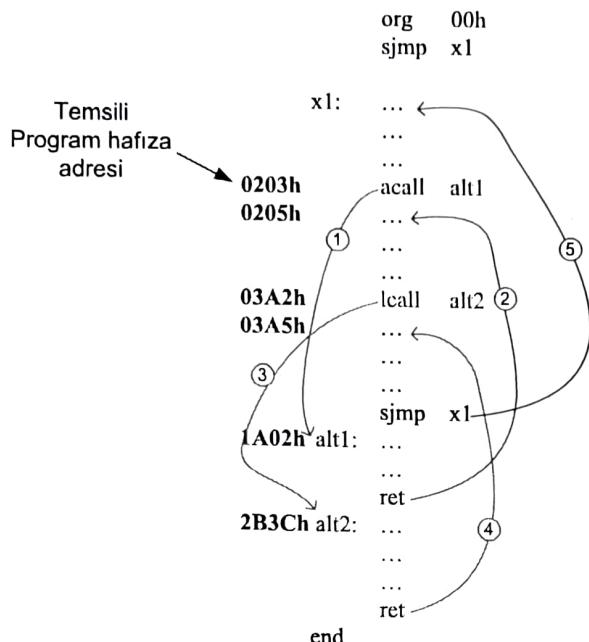
Komut satırlarının koşturulması esnasında “RET” komutu ile karşılaşınca program akışı içinde bulunduğu alt programdan çıkararak alt programın çağrıldığı komut satırının bir alt satırına döner. Dönüş adresinin PC’ye yüklenmesi donanım tarafından otomatik olarak aşağıda gösterildiği gibi gerçekleştirilir.

(PC<sub>15..8</sub>)  $\leftarrow$  ((SP))  
 (SP)  $\leftarrow$  (SP)-1  
 (PC<sub>7..0</sub>)  $\leftarrow$  ((SP))  
 (SP)  $\leftarrow$  (SP)-1

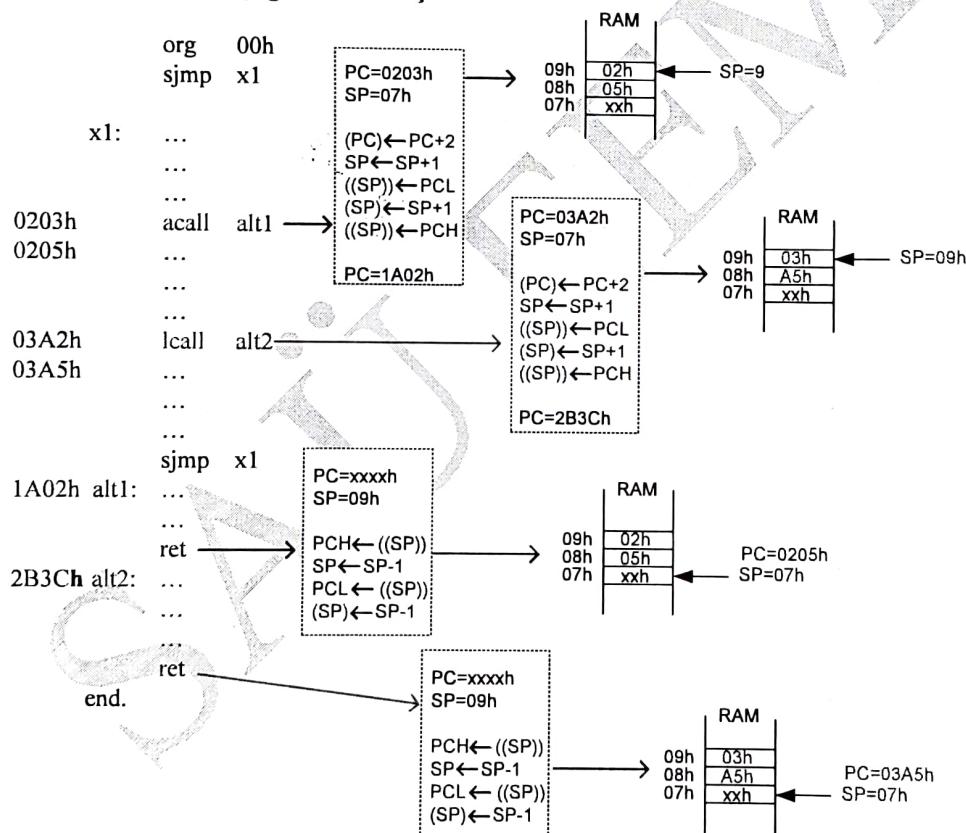
1 PCH  $\leftarrow$  @SP  
 2 (SP) = (SP)-1  
 3 PCL  $\leftarrow$  @SP  
 4 (SP) = (SP)-1  
 5 PC  $\leftarrow$  GİT

alt programın  
sonuna ulaşın  
SP’de oynamaz  
SP’ye gelir

## Örnek:



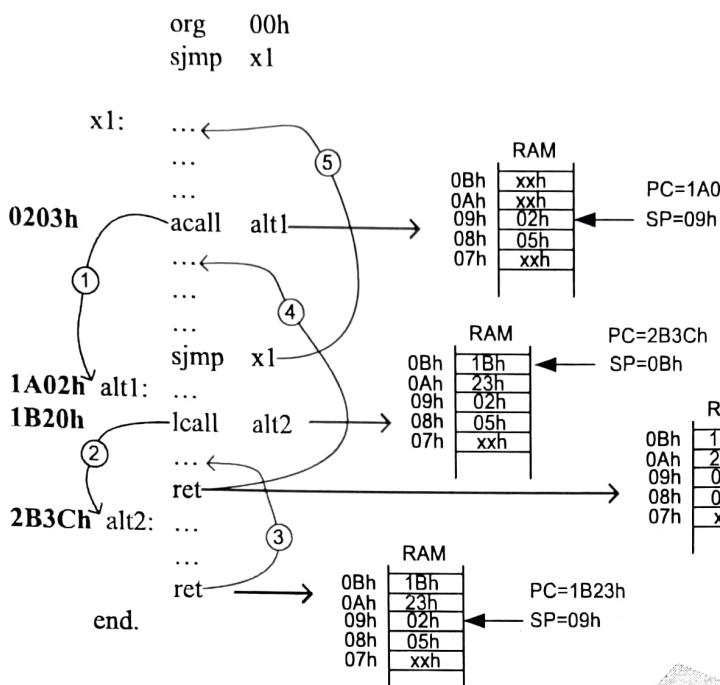
Yukarıda verilen program örneğinde “acall”, “lcall” ve “ret” komutlarının icraları sırasında SP, PC ve RAM değişimleri aşağıda verilmiştir.



### 5.3. İç -İçe Alt Program Yapısı

Aşağıda verilen program örneğinde gösterildiği gibi bazı durumlarda bir alt programın içerisinde bir başka alt programa, onun içerisinde de bir başka alt programa dallanmalar yapılabilir. Ancak bu şekilde

yapılan iç içe dallanmalar SP için gerekli olan hafıza alanının büyümeye neden olacağından gereksiz yere iç içe dallanmalardan kaçınılmalıdır.



## PUSH ve POP komutları

Bütün 8051 ürünlerinde yiğin (stack) hafıza, tümdevre-üstü RAM'da yer alıp yukarıya doğru büyür. **PUSH** komutu, önce yiğin işaretçisini (SP—Stack Pointer) bir artırır, sonra SP ile işaretli hücreye bir byte kopyalanır. **POP** komutu ile PUSH işleminin tersi gerçekleştirilir. SP ile işaretli hücreden bir byte okunur ve SP'nin değeri bir azaltılır.

PUSH ve POP komutları, saklanacak veya geri alınacak byte'ı belirlemek için, sadece doğrudan adresleme modunu kullanır. Fakat yiğinin kendisine, SP saklayıcısını kullanarak, dolaylı adresleme ile erişilir. Bu, yiğinin Yüksek 128'lük bölümde yer alabileceği, ancak SFR alanında olamayacağı anlamına gelir. Yüksek 128'i olmayan ürünlerde, eğer SP Yüksek 128'e işaret ederse, PUSH ile hafızaya yazılmak istenen byte'lar kaybolur ve POP'a okunmak istenen byte'lar ise belirsizdir. Çünkü bu bölgede fiziksel bir RAM hafıza yoktur.

Gösterim	İşlem	Adresleme modları			
		Dir	Ind	Reg	Imm
PUSH <kaynak>	INC SP : MOV @SP , <kaynak>	*			
POP <hedef>	MOV<hedef> , @SP : DEC SP	*			

## 5.4. Program Çökmenin Olası Nedenleri

First In Last Out

Bir alt programdan çıkışlıp alt programın çağrıldığı komut satırının bir alt satırına dönülürken gerekli olan dönüş adresi yukarıda gösterildiği gibi donanım tarafından SP içeriğinden okunurak PC'ye yüklenir. Dönüş adres değerinin hatalı hesaplanmasıına neden olacak işlemlerin gerçekleşmesi durumunda programın akışı bozulur. Bu durum program çökmesi olarak adlandırılır. Aşağıda dönüş adres değerinin hatalı hesaplanmasıına neden olabilecek durumlar belirtilmiş ve detaylı olarak incelenmiştir.

Program çökmesinin olası 3 nedeni olabilir;

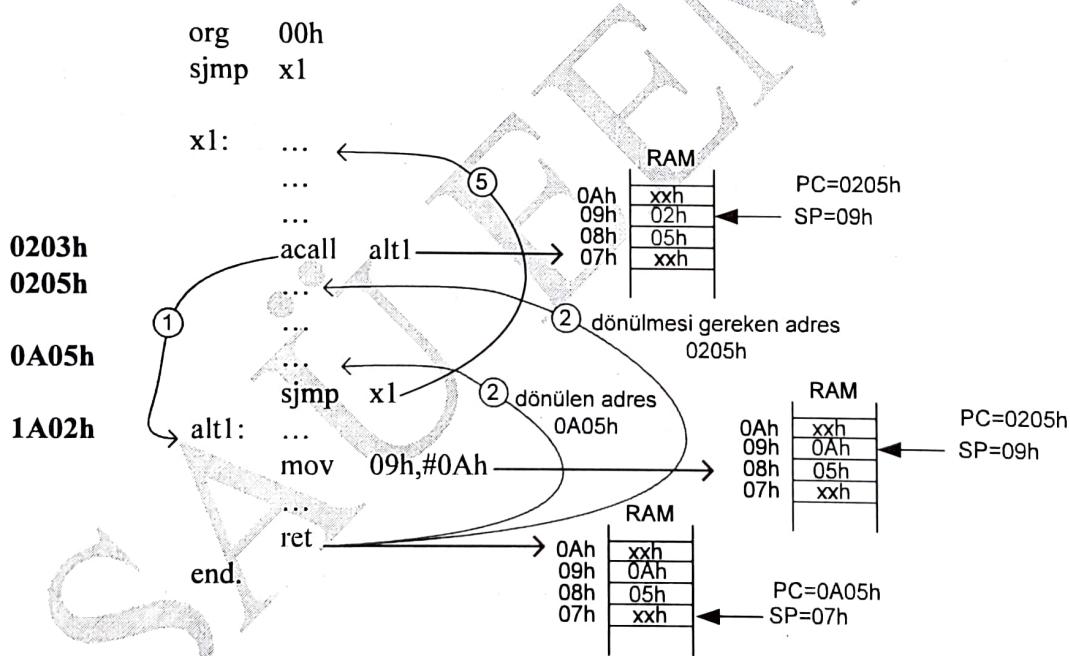
- 1- SP'ın kullanıldığı hafıza alanına kontrolsüz erişim.
- 2- "push" ve "pop" komutlarının hatalı kullanılması.

2.1. push <direct>  
 $(SP) \leftarrow (SP)+1$   
 $((SP)) \leftarrow (\text{direct})$

2.2. pop <direct>  
 $(\text{direct}) \leftarrow (SP)$   
 $(SP) \leftarrow (SP)-1$

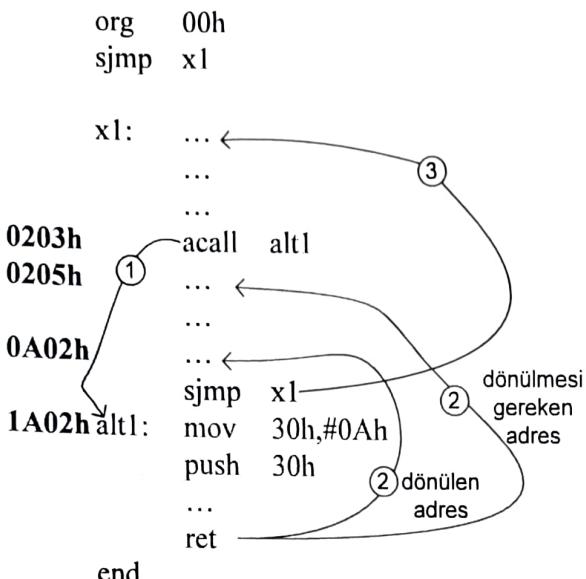
3- SP' nin reset sonrası başlangıç adresi 07h'dir. Bu adres değeri saklayıcı banklarının olduğu bölgeyi işaret etmektedir. Saklayıcılar özel adresleme modlarına sahip olduğundan genellikle SP programın başında veri hafızanın başka amaçlar için kullanılmayacak olan üst bölgesine "mov sp,#data" komutu ile yönlendirilir. Bundan sonra SP'ye yazmalar belirtilen adresten itibaren yapılır. Örneğin "mov sp,#7Ah" komutu ile SP'ın gösterdiği adres değeri 7Ah'e taşınmış olur. İşte bu gibi durumlarda stack (yığın) gerekliliği olan adres alanı yanlış hesaplanmış ise SP'de taşıma olacağını program çökmesi yaşanabilir.

**Örnek:** Aşağıda verilen program örneğinde, yığın alanı için boş bir RAM bölgesi belirlenmemesinden dolayı adresleme esnasında oluşan yanlış adres dönüşünü göstermektedir.



Alt program kodları arasında "mov 09h,#0Ah" bulunan komut satırı ile RAM'ın 09h adresine sabit bir değer yüklenmiştir. Bu alan SP tarafından PC'un dönüş adres değerinin saklandığı bölgeidir. Sabit değerinin yazılması ile dönüş adresinde değişmiş olacaktır. Bu durumda "ret" komutundan sonra PC'ye yüklenecek dönüş adresi 0205h yerine 0A05h olarak değiştirilmiş oldu. Böylece "ret" komutu yürütüldükten sonra ana programda yanlış kod satırına dönmüş olur ve yürütülmesi gereken kodlar yürütülmemiş olur. Böylece program yanlış bir çalışma içerisine girmiştir.

**Örnek:** Aşağıda verilen program örneğinde alt program komutları işlenmesi esnasında kullanılan "push" komutu sonrasında oluşan yanlış geri dönüş adresini ve ana programın işleyişindeki bozulmayı göstermektedir.

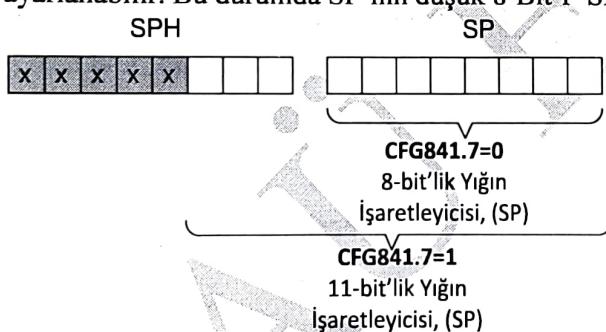


end.

Alt program içerisinde “push” komutu kullanılmış olup; “pop” komutu kullanılmadan “ret” komutu ile karşılaşılmıştır. Bu durumda “push” komutu ile yiğine taşınan veri (#0Ah) dönüş adresinin yüksek byte’ını; yüksek byte adres verisi (#02h) de düşük byte adres verisine dönüşmüştür. PC yüklenmesi gereken dönüş adresi 0205h olması gerekirken 0A02h’e dönüşmüştür.

## 5.5. Yiğin İşaretçisi (Stack Pointer, SP)

Standart INTEL-8051 ve INTEL-8052 mimarisinde 8-Bit’lik yiğin işaretleyici (Stack Pointer, SP)’ye sahiptir. Aduc841 mikrodenetleyicisinde ise başlangıçta (reset sonrasında) SP 8-Bit uzunluğundadır. Ancak, istendiğinde **CFG841** SFR’sinde bulunan “**EXSP**” (**CFG841.7**) biti setlenerek SP 11-Bit uzunluğa ayarlanabilir. Bu durumda SP’nin düşük 8-Bit’i ‘**SP**’, yüksek 3-Bit’i ise ‘**SPH**’ SFR’lerinde tutulur.



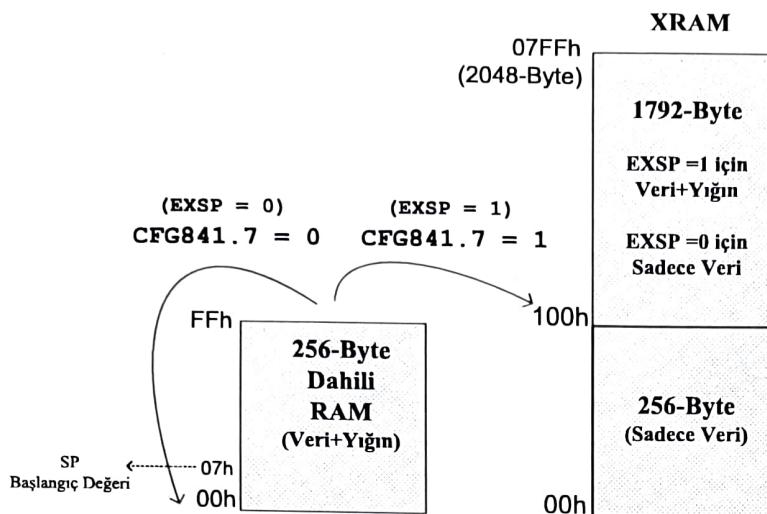
“**EXSP**” (**CFG841.7**) bitinin değerine bağlı olarak SP’nin işaret edebileceği adres değerleri aşağıda gösterilmiştir.

Şekilde gösterildiği gibi EXSP=0 için SP’nin kullanımı 8051’de olduğu gibidir. EXSP=1 yapıldığında ise SP 11-bit uzunluğa sahip olur. Böylece SP’nin adresleyebileceği alan 2048-Byte (**0000h – 07FFh**) olur. Bu durumda SP’nin **00h-FFh** arasındaki değerleri dahili veri hafızadan (RAM), **100h-7FFh** arasındaki değerleri ise Aduc841 yongası üzerinde bulunan XRAM’den gösterilir.

```

orl cfg841,#80h      ;SP 11bit adresleyebilmek için ayarlandı
mov sph,#01h
mov sp,#00h          ;SP=0100h

```



Şekil 20. Aduc841 SP yapısı

#### CFG841 Saklayıcısı

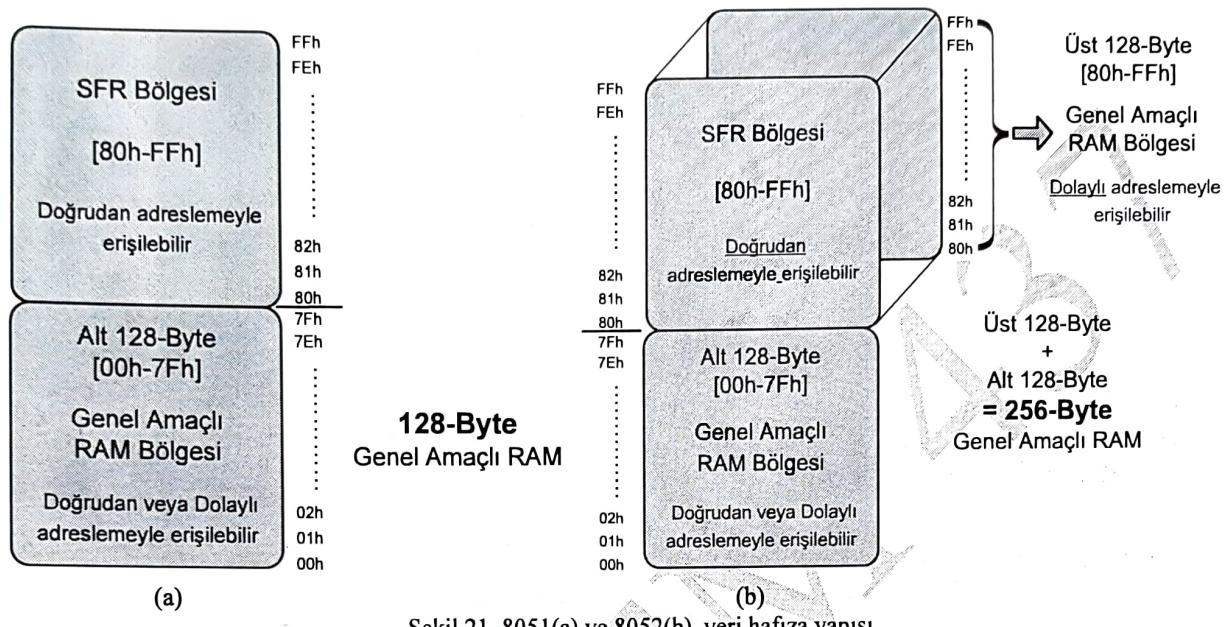
ADUC841 mimarisinde yer alan çeşitli çevre birimlerinin ayarının yapıldığı CFG841 SFR'si **bit adreslenemez**. Açık renkli işaretlenmiş yerler ileride detaylandırılacaktır.

CFG841:	EXSP	PWPO	DBUF	EPM2	EPM1	EPM0	MSPI	XRAMEN
---------	------	------	------	------	------	------	------	--------

Bit	İsim	Açıklama																												
7	EXSP	Setlendiğinde genişletilmiş SP yapısı (11-bit) aktif olur. Temizlendiğinde ise standart SP yapısı (8-bit) seçilmiş olur.																												
6	PWPO	Setlendiğinde P3.4 ve P3.3 pinleri PWM çıkışları olarak atanır. Temizlendiğinde ise P2.6 ve P2.7 PWM çıkışları olarak atanır.																												
5	DBUF	Setlendiğinde dahili DAC çıkış tامponları by-pass edilir. Temizlendiğinde ise DAC çıkış tامponları aktif hâluma geçer.																												
4	EPM2	EPM2, EPM1 ve EPM0 bitleri aşağıdaki tabloya göre PWM saat frekansı için bölmeye faktörünü belirler.																												
3	EPM1																													
2	EPM0	<table border="1"> <tr> <th>EPM2</th> <th>EPM1</th> <th>EPM0</th> <th>Bölmeye Faktörü</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>32</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>64</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>128</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>256</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>512</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>1024</td> </tr> </table>	EPM2	EPM1	EPM0	Bölmeye Faktörü	0	0	0	32	0	0	1	64	0	1	0	128	0	1	1	256	1	0	0	512	1	0	1	1024
EPM2	EPM1	EPM0	Bölmeye Faktörü																											
0	0	0	32																											
0	0	1	64																											
0	1	0	128																											
0	1	1	256																											
1	0	0	512																											
1	0	1	1024																											
1	MSPI	Setlendiğinde SPI haberleşme biriminin MISO, MOSI ve SCLOCK fonksiyonlarını sırasıyla P3.3, P3.4 ve P3.5 pinlerine atamır.																												
0	XRAMEN	Setlendiğinde dahili XRAM kullanıma açılır ve harici veri hafıza adres alanının alt 2-kByte 'lık kısmına yerleştirilir. Temizlendiğinde ise dahili XRAM kullanıma kapalıdır.																												

## BÖLÜM 6 Hafıza Yapısı

### 6.1. Üst RAM Bölgesi



8051 ailesinin her üyesinde bulunmayan üst RAM bölgesi genel amaçlı saklayıcı olarak kullanılır. Alt RAM bölgesi gibi fonksiyonlara (register bölgesi, bit adreslenebilir bölge) sahip değildir. Üst 128-Byte RAM bölgesi Şekil 2.b'de gösterildiği gibi SFR'ler ile aynı hafıza adres alanını [80h – FFh] kullanır. Dolayısıyla bu iki bölgeden (SFR ve genel amaçlı üst 128-Byte) hangisine erişileceği tamamen kullanılan komutun yapısına bağlıdır. Genel amaçlı üst 128-Byte RAM bölgesine sadece dolaylı adresleme ile erişilirken SFR alanına ise sadece doğrudan adresleme ile erişilebilir.

### SFR Bölgesi

Mikrodenetleyicilerin sahip oldukları her bir çevre birimi (I/O portları, kesme kaynakları, zamanlayıcı/sayıacı, vd.) için tanımlanmış çeşitli çalışma modları vardır. Örneğin 8051 ailesinde zamanlayıcı/sayıcılar 4 farklı çalışma modunda kullanılabilir. Benzer şekilde seri haberleşme birimi (UART) için birden fazla çalışma modu vardır. Dolayısı ile mikrodenetleyicinin sahip olduğu çevre birimlerinin amacına uygun şekilde kullanılabilmesi için çalışma ayarlarının önceden doğru bir şekilde yapılması gereklidir.

Mikrodenetleyicilerde çevre birimlerinin çalışma ayarlarının yapıldığı ve/veya ilgili çevre birimine ait anlık çalışma değerlerinin saklandığı, her bir çevre birimine özel olarak atanmış bir veya daha fazla saklayıcı vardır. Kullanım amaçlarından dolayı “**Özel Fonksiyon Saklayıcısı, (Special Function Register, SFR)**” olarak adlandırılan bu saklayıcılar 8051-ailesinde dahili veri hafızanın [80h-FFh] adreslerinde yer alır. SFR'ler için tahsis edilen RAM'ın bu alanının genel amaçlı veri saklamak amacıyla kullanılması tavsiye edilmez.

Şekil 2(b)'de belirtilen 8052 dahili veri hafıza yapısı kullanımı aşağıdakik gibi anlatılmıştır. Standart 8051 çekirdeğine ait SFR haritası aşağıda gösterilmiştir. Şekilden görüldüğü gibi SFR'lerin her birinin ilgili olduğu çevre birimine ve kullanım amacına uygun özel ismi vardır. Örneğin kesmelerin

yetkilendirme bitlerini içeren SFR IE: Interrupt Enable ismi ile, seri kanal kontrol ayarlarının yapıldığı SFR SCON:Serial Control ismi ile, vb. kullanılır. Aşağıda kısaca açıklanan SFR'lerin her biri ilerleyen kısımlarda detaylı olarak incelenecektir.

SFR Byte Adresleri									
FFh									
FEh									
FDh									
ECh									
EDh									
DFh									
D7h									
CP0									
C7h									
BFh									
B7h									
Afh									
A7h									
9Fh									
97h									
8Fh									
87h									

Bit adreslenebilir SFR'ler

Şekil 22. 8051 SFR haritası

### A Saklayıcısı (Akümülatör, ACC)

A saklayıcısı (akümülatör) çok amaçlı önemli bir saklayıcıdır. ALU tarafından yürütülen aritmetik ve lojik işlemlerin sonuçları A saklayıcısında saklanır. Veri transferlerinde, harici RAM erişimlerinde vb. bir çok komutta kullanılan akümülatörün komutlardaki kullanımı ACC veya A şeklindedir. (MOV A, #53h, MOV ACC, #53h, CLR A.0) A saklayıcısı bit adreslenebilirdir.

### B Saklayıcısı

Çarpma ve bölme işlemlerinde A saklayıcısı ile birlikte kullanılan bir kaydedicidir. B saklayıcısı işlem sonuçları vb. geçici verilerin saklanması içinde kullanılabilir.

### Zamanlayıcı/Sayıçı Saklayıcıları

TCON, TMOD, TLO, TL1, TH0 ve TH1 8051 mimarisinde bulunan iki adet zamanlayıcı/sayıçının (zamanlayıcı/sayıçı 0, T/C 0 ve zamanlayıcı/sayıçı 1, T/C 1) çalışma ayarlarının yapıldığı ve sayma değerlerinin tutulduğu saklayıcılardır. Zamanlayıcı/sayıcılar ileri kısımlarda ayrıntılı olarak incelenecektir.

### SCON-SBUF Saklayıcıları

Mimaride bulunan asenkron seri haberleşme biriminin (Universal Asynchronous Receiver Transmitter, UART) çalışma ayarlarının yapıldığı (SCON) ve alınan-iletilen verilerin okunup-yazıldığı (SBUF) saklayıcılardır. İleriki kısımlarda ayrıntılı olarak incelenecektir.

### P0, P1, P2 ve P3 Saklayıcıları

8051 mimarisinde 4 adet 8-bitlik giriş-çıkış portu (Port0-P0, Port1-P1, Port2-P2, Port3-P3) bulunur. 32 adet ( $4 \times 8 = 32$ ) giriş-çıkış pininin her biri bağımsız bir şekilde çalışabilir. Bit tabanlı veya byte tabanlı çıkış yapılabılır. Portlar-pinler fiziksel dünya ile mikrodenetleyici arasında lojik veri transferinin (1/0,

On/Off) gerçekleşmesini mümkün kılar. P0, P1, P2 ve P3 saklayıcıları portlara aktarılmak istenen verilerin yazıldığı ve portlardan okunan verilen tutulduğu saklayıcılardır. Reset sonrasında P0, P1, P2 ve P3 port saklayıcıları donanım tarafından OFFh değeri ile yüklenir.

### SP Saklayıcısı

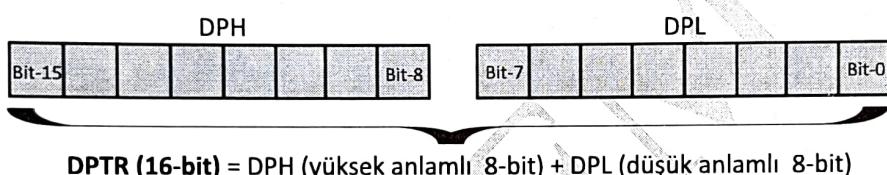
Programın icrası esnasında alt programlardan veya kesme hizmet alt programlarından çıkışılıp ana programa geri dönülürken geri dönüş adresinin tutulduğu, PUSH ve POP komutları ile veri yazılıp okunan saklayıcıdır. Yiğin işaretçisinin (Stack Pointer, SP) hatalı kullanımı programın çökmesine istenmeyen durumların gerçekleşmesine neden olur. İleriki kısımlarda ayrıntılı olarak incelenecaktır.

### IE – IP Saklayıcıları

Kesme kaynaklarının yetkilendirme (Interrupt Enable, IE) ve öncelik (Interrupt Priority, IP) ayarlarının yapıldığı saklayıcılardır. İleriki kısımlarda ayrıntılı olarak incelenecaktır.

### DPH-DPL Saklayıcıları

Veri işaretçi (Data Pointer, DPTR) 16-bit'lik (2-Byte) bir saklayıcıdır. DPH (yüksek Byte) ve DPL (düşük Byte) saklayıcıları aşağıda gösterildiği gibi DPTR saklayıcısının sırasıyla yüksek ve düşük değerli 8-bitini oluşturmaktadır. DPH ve DPL saklayıcılarına ayrı ayrı erişilebildiği (okunup-yazılabilir) gibi bazı özel komutlarla ikisinede aynı anda erişilebilir. DPTR saklayıcısı harici hafıza birimlerine erişimlerde kullanılmaktadır.



### ÇIFT DPTR- DUAL DPTR (DATAPointer)

ADUC841 mimarisinde **ana (main)** ve **gölge (shadow)** olarak adlandırılan çift DPTR (datapointer) desteklemektedir. İkinci data pointer **gölge DPTR**'dir ve DPTR ayar SFR'si **DPCON** üzerinden konfigüre edilir. MOVC/MOVX komutlarının kullanımından sonra;

- DPTR değerinin donanımsal olarak otomatik artırılması, azaltılması
- Ayrıca ana ve gölge DPTR'ler arasında donanımsal olarak otomatik toggle (mandallama) gibi kullanışlı özellikler aşağıdaki tabloda anlatılmıştır.

Resetten sonra ana DPTR etkindir.

**NOT:** Ana ve gölge DPTR'ler arasındaki seçim için elektronik olarak anahtarlanmasına kısaca **mandallama** denmektedir.

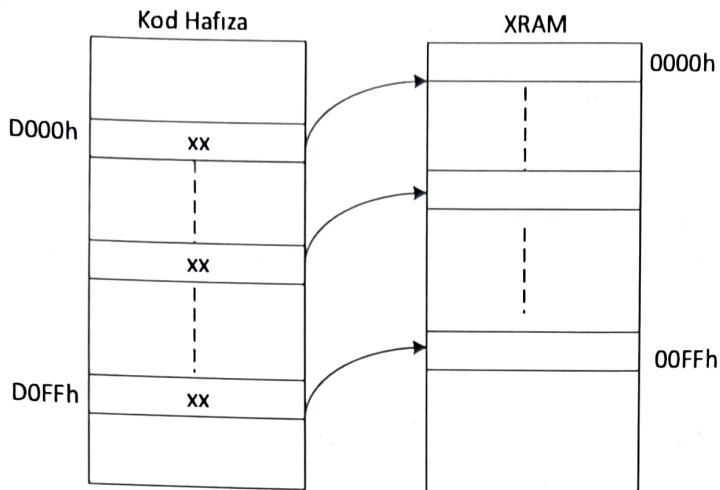
**DPCON:** Byte adreslenebilir. Resetten sonraki değeri 00h'tır.

DPCON:	--	DPT	DP1m1	DP1m0	DP0m1	DP0m0	-	DPSEL
--------	----	-----	-------	-------	-------	-------	---	-------

Bit	İsim	Açıklama															
7	--	Rezerve															
6	DPT	“1” ise otomatik mandallama yetkilendirilir. “0” ise yetkisizdir.															
5	DP1m1	Gölge DPTR modları aşağıdadır. <table border="1" style="margin-left: 20px;"> <tr> <td>DP1m1</td> <td>DP1m0</td> <td>Gölge DPTR Çalışma Modları</td> </tr> <tr> <td>0</td> <td>0</td> <td>Standart 8052 gibi davranışır</td> </tr> <tr> <td>0</td> <td>1</td> <td>Her MOVX ve MOVC komutlarının kullanımından sonra DPTR donanımsal olarak otomatik bir artırılır (inc kullanmadan)</td> </tr> <tr> <td>1</td> <td>0</td> <td>Her MOVX ve MOVC komutlarının kullanımından sonra DPTR donanımsal olarak bir azaltılır (dec kullanmadan)</td> </tr> <tr> <td>1</td> <td>1</td> <td>Her MOVX ve MOVC komutlarının kullanımından sonra DPTR’nin en anlamsız (LSB) 8 byte’ı mandallanır.</td> </tr> </table>	DP1m1	DP1m0	Gölge DPTR Çalışma Modları	0	0	Standart 8052 gibi davranışır	0	1	Her MOVX ve MOVC komutlarının kullanımından sonra DPTR donanımsal olarak otomatik bir artırılır (inc kullanmadan)	1	0	Her MOVX ve MOVC komutlarının kullanımından sonra DPTR donanımsal olarak bir azaltılır (dec kullanmadan)	1	1	Her MOVX ve MOVC komutlarının kullanımından sonra DPTR’nin en anlamsız (LSB) 8 byte’ı mandallanır.
DP1m1	DP1m0	Gölge DPTR Çalışma Modları															
0	0	Standart 8052 gibi davranışır															
0	1	Her MOVX ve MOVC komutlarının kullanımından sonra DPTR donanımsal olarak otomatik bir artırılır (inc kullanmadan)															
1	0	Her MOVX ve MOVC komutlarının kullanımından sonra DPTR donanımsal olarak bir azaltılır (dec kullanmadan)															
1	1	Her MOVX ve MOVC komutlarının kullanımından sonra DPTR’nin en anlamsız (LSB) 8 byte’ı mandallanır.															
4	DP1m0																
3	DP0m1	Ana DPTR modları aşağıdadır. <table border="1" style="margin-left: 20px;"> <tr> <td>DP0m1</td> <td>DP0m0</td> <td>Ana DPTR Çalışma Modları</td> </tr> <tr> <td>0</td> <td>0</td> <td>Standart 8052 gibi davranışır</td> </tr> <tr> <td>0</td> <td>1</td> <td>Her MOVX ve MOVC komutlarının kullanımından sonra DPTR donanımsal olarak otomatik bir artırılır (inc kullanmadan)</td> </tr> </table>	DP0m1	DP0m0	Ana DPTR Çalışma Modları	0	0	Standart 8052 gibi davranışır	0	1	Her MOVX ve MOVC komutlarının kullanımından sonra DPTR donanımsal olarak otomatik bir artırılır (inc kullanmadan)						
DP0m1	DP0m0	Ana DPTR Çalışma Modları															
0	0	Standart 8052 gibi davranışır															
0	1	Her MOVX ve MOVC komutlarının kullanımından sonra DPTR donanımsal olarak otomatik bir artırılır (inc kullanmadan)															
2	DP0m0																
1	--	Rezerve															
0	DPSEL	“0” Ana (main) DPTR seçilir. “1” Gölge (Shadow) DPTR seçilir.															

**ÖRNEK:**

Kod hafıza D000h ile D0FFh arasındaki verileri XRAM’ın 0000h ile 00FFh adresleri arasına taşınacaktır. Dual DPTR yapısını kullanarak ilgili programı yazınız.



Aşağıda çift DPTR kullanıldığından ve kullanılmadığındaki kodlar gösterilmiştir.

ÇİFT DPTR KULLANMADAN	ÇİFT DPTR KULLANDIĞIMIZDA
<pre> ORG 00H SJMP START  START: MOV 20H,#0D0H MOV 21H,#00H      ; kod hafıza                   ; adresleri MOV 22H,#00H MOV 23H,#00H      ; XRAM adresleri  MOV DPH, 20H MOV DPL,21H  XX: CLR A  MOVC A,@A+DPTR INC DPTR  MOV 20H,DPH MOV 21H,DPL  MOV DPH, 22H MOV DPL,23H MOVC A,@A+DPTR INC DPTR MOV 22H,DPH MOV 23H,DPL  JNZ XX END </pre>	<pre> MOV DPTR,#0          ; Ana DPTR = 0  MOV DPCON,#55H       ;Gölge DPTR seçildi, DPTR1 otomatik artırma                       ;modunda; DPTR0 otomatik artırma modunda;                       ;DPTR otomatik değiştirme (toggling) ON  MOV DPTR,#0D000H     ; Gölge DPTR = D000H MOVELOOP: CLR A MOVC A,@A+DPTR      ;Adresin gösterdiği Veriyi al ACC'ye yaz, DPTR'yi                       ;bir artır, Ana.DPTR'ye geç (toggle)                       ;; ACC'deki değeri DPTR'nin gösterdiği adresin                       ;içeriğine yaz, DPTR'yi bir ;artır                       ;; Gölge DPTR'ye geç (toggle)  MOV A, DPL JNZ MOVELOOP END </pre>

### PSW Saklayıcısı (Program Status Word, Program Durum Saklayıcısı)

PSW, komutların yürütülmesi esnasında bazı özel durumların oluşup-olmadığını gösteren bayrakların yer aldığı, saklayıcı banklarının (Bank 0, Bank 1, Bank 2 ve Bank 3) seçim yapıldığı önemli bir SFR'dır. PSW saklayıcısı bit adreslenebilir.

PSW:	CY	AC	F0	RS1	RS0	OV	F1	P
	*						*	

Bit	İsim	Açıklama																								
7	CY	CY (veya C) biti aritmetik işlemlerde elde fonksiyonunu gerçekleştirir. Çeşitli komutlar yürütürken elde bayrağının değeri değişimdir. Elde bayrağı komutlarda çoğunlukla "C" olarak kullanılır.																								
6	AC	Yardımcı elde bayrağı (auxiliary carry, AC) herhangi bir Byte'ın düşük 4-bitinden yüksek 4-bitine elde olma durumunda setlenir. BCD (Binary Coded Decimal) sayılar ile işlem yaparken kullanılabilir.																								
5	F0	Genel amaçlı kullanıma açık 1-bit'lik alan																								
4	RS1	RS1 ve RS0 bitleri aşağıdaki tabloya göre aktif olan saklayıcı deposunu belirler.																								
		<table border="1"> <thead> <tr> <th>RS</th> <th>RS</th> <th>Aktif</th> <th>Saklayıcı</th> </tr> <tr> <th>1</th> <th>0</th> <th>Depo</th> <th>Deposu</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Depo 0</td> <td></td> </tr> <tr> <td>0</td> <td>1</td> <td>Depo 1</td> <td></td> </tr> <tr> <td>1</td> <td>0</td> <td>Depo 2</td> <td></td> </tr> <tr> <td>1</td> <td>1</td> <td>Depo 3</td> <td></td> </tr> </tbody> </table>	RS	RS	Aktif	Saklayıcı	1	0	Depo	Deposu	0	0	Depo 0		0	1	Depo 1		1	0	Depo 2		1	1	Depo 3	
RS	RS	Aktif	Saklayıcı																							
1	0	Depo	Deposu																							
0	0	Depo 0																								
0	1	Depo 1																								
1	0	Depo 2																								
1	1	Depo 3																								
3	RS0																									
2	OV	Aritmetik işlemlerde taşıma (işlem sonucunun 255'i aşması) durumunda donanım tarafından setlenir.																								
1	F1	Genel amaçlı kullanıma açık 1-bit'lik alan																								
0	P	Eşlenik (parity) biti; ACC'deki "1"lerin sayısı tek ise P=1, çift ise P=0 olur.																								

### PCON Saklayıcısı

İçerdiği bitlerin açıklaması aşağıda verilmiştir. PCON SFR'si bit adreslenemez.

PCON:	SMOD	SERIPD	INT0PD	ALEOFF	GF1	GF0	PD	IDL
	*	*	*	*				

Bit	İsim	Açıklama
7	SMOD	Seri haberleşme veri iletişim hızının belirlenmesinde kullanılır. Detaylı bilgi için seri haberleşme kısmına bakınız.
6	SERIPD	Setlendiğinde düşük güç modunda I <sup>2</sup> C/SPI kesmesi aktif olur
5	INT0PD	Setlendiğinde düşük güç modunda harici kesme0 aktif olur
4	ALEOFF	Setlendiğinde ALE çıkışları kapatılır
3	GF1	Genel amaçlı kullanıma açık 1-bit'lik alan
2	GF0	Genel amaçlı kullanıma açık 1-bit'lik alan
1	PD	Setlendiğinde mikrodenetleyici düşük güç (Power-Down) moduna geçer, donanımsal reset, TIC kesmesi eğer ve yetkilendirilmiş ise I <sup>2</sup> C/SPI ve harici kesme0 kesmeleri ile normal çalışmaya geçer
0	IDL	Setlendiğinde mikrodenetleyici boşta çalışma (Idle) moduna geçer, donanımsal reset veya yetkilendirilmiş herhangi bir kesme ile normal çalışmaya geçer

\*: Bu bitler Aduc841 için verilmiştir. Standart 8051'de ise rezerve olarak bırakılmıştır.

8051 ailesinin bir üyesi olan 8052 çekirdeğine ait SFR haritası aşağıda gösterilmiştir. 8052'ye ait SFR haritasında üstte verilen 8051 SFR haritasından farklı olarak sadece 8052 mimarisinde bulunan 3. Zamanlayıcı/Sayıciya ait SFR'ler (T2CON, RCAP2L, RCAP2H, TL2 ve TH2) bulunmaktadır. Diğer tüm SFR'lerin ve bulundukları adreslerin aynı olduğuna dikkat ediniz.

SFR Byte Adresleri

F8h									FFh
F0h	B								F7h
E8h									EFh
E0h	ACC								E7h
D8h									DFh
D0h	PSW								D7h
C8h	T2CON		RCAP2L	RCAP2H	TL2	TH2			CFh
C0h									C7h
B8h	IP								BFh
B0h	P3								B7h
A8h	IE								AFh
A0h	P2								A7h
98h	SCON	SBUF							9Fh
90h	P1								97h
88h	TCON	TMOD	TL0	TL1	TH0	TH1			8Fh
80h	P0	SP	DPL	DPH				PCON	87h

Bit adreslenebilir SFR'ler

Şekil 23. 8052 SFR haritası

Fdh	SPICON	DAC0L	DAC0H	DAC1L	DAC1H	DACCON			FFh
F0h	B	ADCCOPSL	ADCCOPSH	ADCGAINL	ADCGAINH	ADCCON3		SPIDAT	F7h
E8h	I2CCON							ADCCON1	EFh
E0h	ACC								EAh
D8h	ADCCON2	ADCDATAL	ADCDATAH					PSMCON	DFh
D0h	PSW		DMAL	DMAH	DMAP			PLLCON	D7h
C8h	T2CON		RCAP2L	RCAP2H	TL2	TH2			CFh
C0h			CHIPID				EDARL	EDARH	C7h
B8h	IP	ECON			EDATA1	EDATA2	EDATA3	EDATA4	BFh
B0h	P3	PWM0L	PWM0H	PWM1L	PWM1H			SPH	B7h
A8h	IE	IEIP2					PWMCON	CFG841/ CFG842	AFh
A0h	P2	TIMECON	HTHSEC	SEC	MIN	HOUR	INTVAL	DPCON	A7h
98h	SCON	SBUF	I2CDAT	I2CADD		T3FD	T3CON		9Fh
90h	P1	I2CADD1	I2CADD2	I2CADD3					97h
88h	TCON	TMOD	TL0	TL1	TH0	TH1			8Fh
80h	P0	SP	DPL	DPH	DPP			PCON	87h

Bit adreslenebilir SFR'ler

Şekil 24 Aduc841 SFR haritası

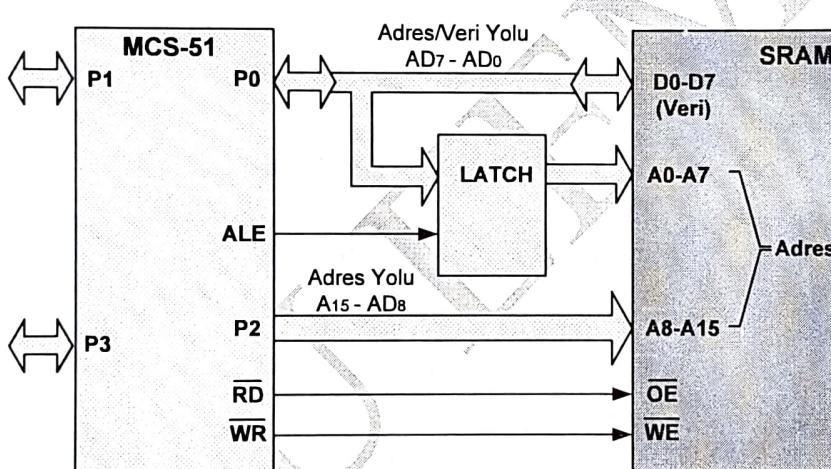
Bu durum 8051 (veya 8052) tabanlı tüm mikrodenetleyiciler için geçerlidir. Standart 8051 mimarisinde 21 adet 8052'de ise 26 adet SFR bulunmaktadır. Mikrodenetleyicilerin sahip oldukları çevre birimleri endüstriyel ihtiyaçlara ve teknolojideki gelişmelere paralel olarak sürekli olarak artmaktadır. Dolayısı ile ilave edilen her yeni çevre birimine (Analog Giriş/Çıkış birimleri, haberleşme birimleri, darbe genişlik modülasyon birimi, vb.) ait SFR'lerde SFR haritasının boş olan adreslerinde yerini almaktadır. Aşağıda Aduc-841 mikrodenetleyicisi için SFR haritası verilmiştir.

Aduc-841 SFR haritası incelendiğinde 8051'e ait tüm SFR'lerin mevcut adreslerde bulundukları görülmektedir. İlave çevre birimlerine ait SFR'ler ise üretici firmanın tasarımına bağlı olarak boş olan SFR adreslerine yerleştirilmiştir.

### **Harici Veri Hafıza**

Mikrodenetleyici tabanlı uygulamalarda dahili veri hafızanın yetersiz kaldığı, ilave veri hafızaya ihtiyaç duyulduğu durumlar olabilir. Bu gibi durumlarda harici veri hafıza (External RAM = XRAM) kullanılabilir. 8051 ailesinin adresleyebileceği harici veri hafıza alanı 64 KByte ile sınırlıdır. Aşağıda harici veri hafıza kullanılan 8051 tabanlı bir sisteme ait blok diyagram verilmiştir.

8051 tabanlı mikrodenetleyicilerde dahili ve harici veri hafızaya erişim komutları farklıdır. (Dahili hafızaya "MOV" harici hafızaya ise "MOVX" komutu ile erişilir.) Şekil 6'da gösterildiği gibi gerekli donanım ve devre bağlantıları yapılması durumunda "MOVX" komutu ile CPU harici veri hafızaya erişmek (yazma/okuma) için gerekli tüm işlemleri otomatik olarak gerçekleştirir.



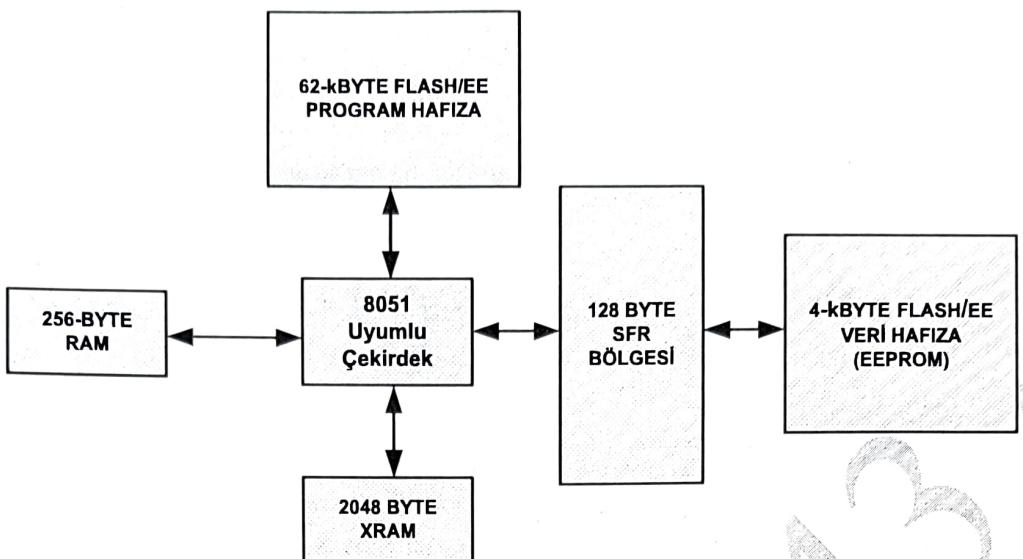
Şekil 25. Harici Veri hafızaya (64-kByte) erişim için gerekli arayüz

### **Harici Hafıza Erişim Pinleri:**

ALE/PROG (Address Latch Enable), PSEN (Program Store Enable) ve  $\overline{EA}$ /VPP (External Access) pinleri harici hafıza erişim sürecinde kullanılan pinlerdir. ALE/PROG pini harici hafızaya (program/veri) erişimlerde adresin düşük değerli (A7-A0) Byte'ını, adres tutucusuna tutturmak amacıyla kullanılır. PSEN pini harici program hafızaya erişimlerde kullanılır.  $\overline{EA}$ /VPP pini program hafızanın seçiminde, harici ( $\overline{EA}$ /VPP=0) veya dahili ( $\overline{EA}$ /VPP=1), kullanılır. Bu pin mutlaka GND veya Vcc'pinine bağlanmalıdır, boşta bırakılmamalıdır.

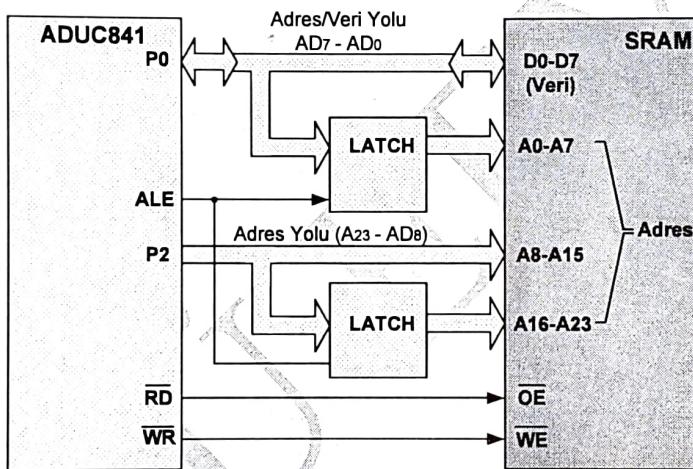
### **ADUC841 Mikrodenetleyicisi Hafıza Yapısı**

ADUC841 mikrodenetleyicisi aşağıda gösterildiği gibi 62-kByte program hafızaya, 256-Byte veri hafızaya, 2-kByte dahili XRAM hafızaya ve 4-kbyte EEPROM veri hafızaya sahiptir.



Şekil 26. Aduc841 Hafıza birimleri

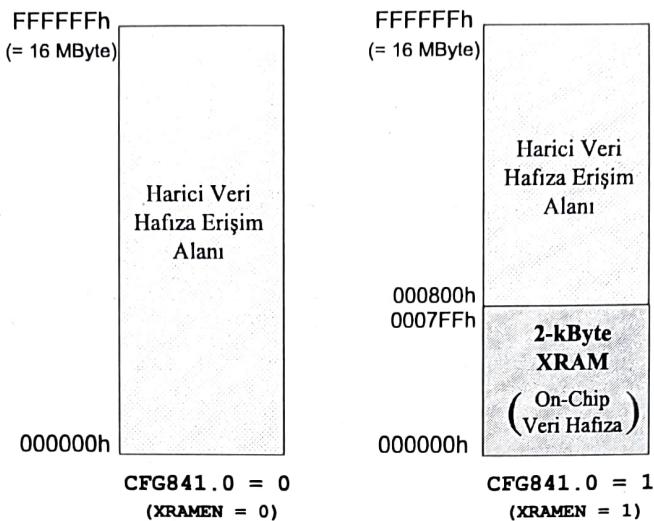
Dahili program (62-kByte) ve veri hafızanın (256 Byte) kullanımı standart 8051 ile aynıdır. Ancak Aduc841'in "MOVX" komutu ile adresleyebileceğiniz harici veri hafıza alanı 16-MByte gibi oldukça yüksek bir değere sahiptir.



Şekil 27. Aduc841 Harici Veri hafızaya (16-MByte) erişim için gerekli arayüz

**XRAM Hafıza:** Aduc841 mikrodenetleyicisi 256-Byte genel amaçlı RAM birimine ilave olarak tümdevre üzerinde (on-chip) 2-KByte (2048-Byte) büyüğünde veri hafızaya sahiptir. Tümdevre üzerinde olmasına rağmen bu ilave veri hafızaya erişim sadece "MOVX" komutları ile gerçekleşir. Hafıza tümdevre üzerinde olduğundan dahili "MOVX" komutları ile erişilebildiğinden XRAM hafıza olarak adlandırılır. Dahili XRAM kullanılabilmesi için CFG841 SFR'sinde bulunan XRAM yetkilendirme biti "**xramen**" setlenmelidir (CFG841.0 = 1). Bu durumda şekilde gösterildiği gibi dahili XRAM harici veri hafıza adres alanının alt 2-KByte'lık kısmını kullanır. Dahili XRAM'e erişilirken P0 ve P2 portları kullanılmadığından bu portlar genel amaçlı kullanım için serbesttirler.

2-kByte'lık dahili XRAM hafızanın yetersiz olması durumunda harici RAM hafıza entegreleri ilave edilerek veri hafıza alanı artırılabilir.



Şekil 28. Harici ve dahili XRAM kullanımı

Dahili XRAM'ın üst 1792-Byte'luk kısmı gerekli ayarlamalar yapıldığı durumda SP alanı olarak kullanılabilir.

#### Flash EEPROM Veri Hafıza:

Aduc841 mimarisinde 4-kByte kapasiteye sahip Flash/EE veri hafızası (EEPROM) bulunur. Flash/EE veri hafıza her biri 4-Byte'luk alana sahip 1024 sayfadan oluşur ( $4 \times 1024 = 4096$  Byte). Flash/EE veri hafızanın 4096-Byte'nın her birine tek tek Byte düzeyinde erişilebileceği gibi 4-Byte'lük sayfalar halinde erişilebilir.

SAYFA ADRESİ (EARDRH/L)	BYTE 1 (0FFCH)	BYTE 2 (0FFDH)	BYTE 3 (OFFEH)	BYTE 4 (0FFFH)
	BYTE 1 (0FF8H)	BYTE 2 (0FF9H)	BYTE 3 (OFFAH)	BYTE 4 (0FFBH)
	BYTE 1 (000CH)	BYTE 2 (000DH)	BYTE 3 (000EH)	BYTE 4 (000FH)
	BYTE 1 (0008H)	BYTE 2 (0009H)	BYTE 3 (000AH)	BYTE 4 (000BH)
	BYTE 1 (0004H)	BYTE 2 (0005H)	BYTE 3 (0006H)	BYTE 4 (0007H)
	BYTE 1 (0000H)	BYTE 2 (0001H)	BYTE 3 (0002H)	BYTE 4 (0003H)
Byte adresleri parantez içerisinde verilmiştir.				
	EDATA1 SFR	EDATA2 SFR	EDATA3 SFR	EDATA4 SFR

Şekil 29. Dahili Flash veri hafıza

## Program Hafıza

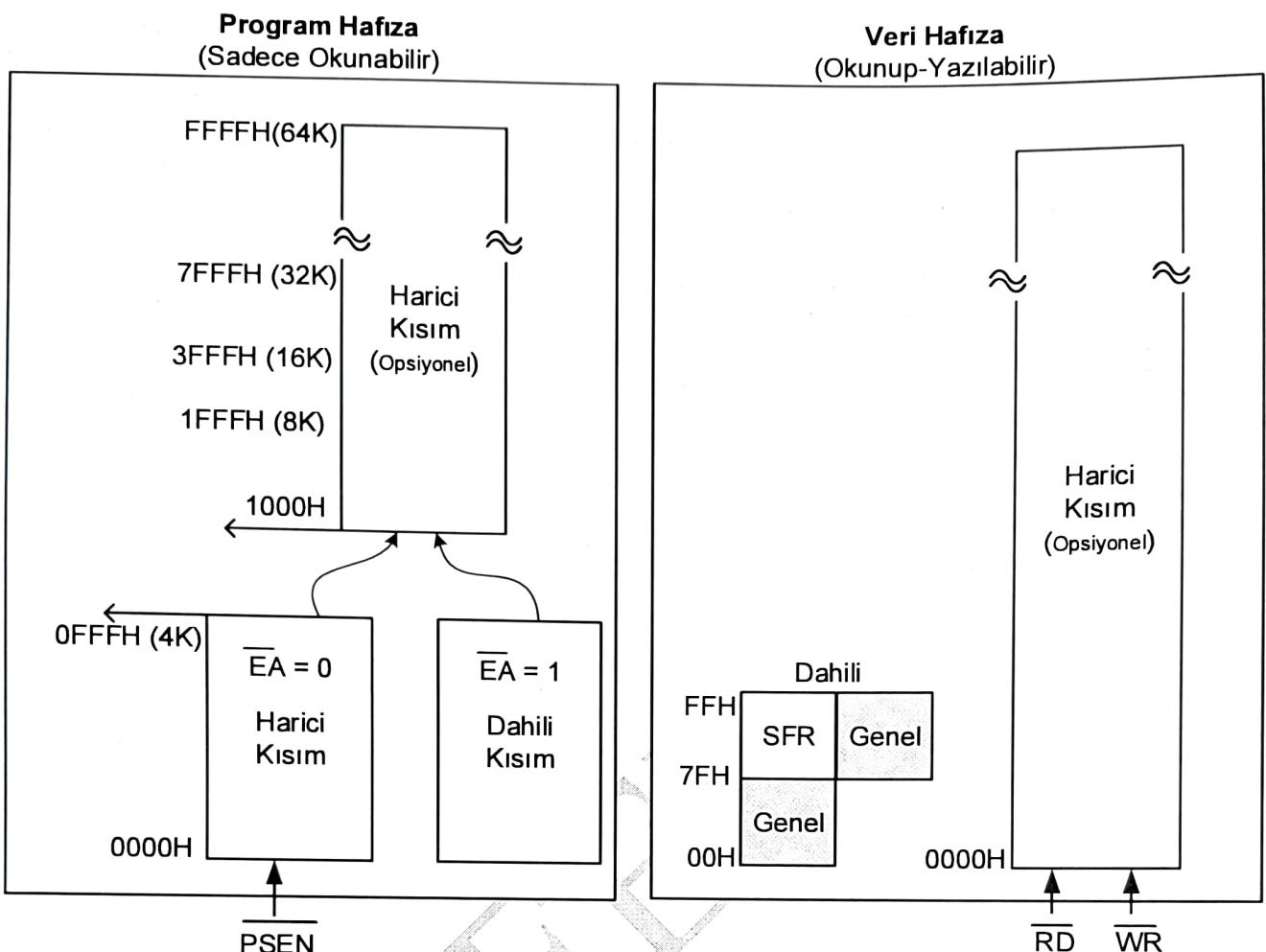
Bir mikrodenetleyicinin gerçekleştirmesi istenen işlevler uygun bir programlama dilinde (Assembly, C, vb.) yazılıp derlenerek “makine kodu” olarak adlandırılan kodlar elde edilir. Program (kod) hafıza mikrodenetleyicinin koşturacağı makine kodlarının tutulduğu kalıcı (non-volatile) hafıza birimidir. Sadece okunabilir olan (ROM-Read Only Memory) program hafıza mikrodenetleyici mimarisinde (dahili, internal) bulunabileceği gibi harici (external) olarakda mikrodenetleyiciye eklenebilir.

8051 ailesinin bazı üyeleri tümdevre üzerinde ROM Program Hafızaya sahiptir. 8051 ve 8052, 4K ve 8K fabrika-maskeli (factory-masked) ROM hafızalara sahiptir. 80C515A-5 ve 80C517A-5 ürünlerinin her biri 32K tümdevre üzeri ROM'a sahiptir. 8751 ve 8752, 8051 ve 8052'in EPROM'lu sürümleridir. 8031, 8032, 80C535A ve 80C537A, sırasıyla 8051, 8052, 80C515 ve 80C517A'nın ROM'suz sürümleridir. Standart 8051 mikrodenetleyicisinde 4-KByte (0000h – 0FFFh) olan dahili program hafızanın büyütüğü günümüzde üretilen 8051 tabanlı mikrodenetleyicilerde çok daha yüksek değerlere (ADUC-841 için 62-KByte) ulaşmıştır.

Dahili program hafızanın bulunmadığı veya yetersiz olduğu durumlarda makine kodları harici program hafızada saklanıp buradan okunabilir. 8051 mikrodenetleyici ailesinde program hafızanın seçimi (dahili/harici)  $\overline{EA}$  (External Access) pininin lojik değerine bağlı olarak aşağıdaki gibi yapılır.

$\overline{EA} = 1$  olması durumunda mikrodenetleyici dahili program hafızayı kullanır. Dahili program hafızanın değeri aşıldığında (standart 8051'de 0FFFh=4-KByte) ise makine kodları otomatik olarak harici program hafızadan okunur.

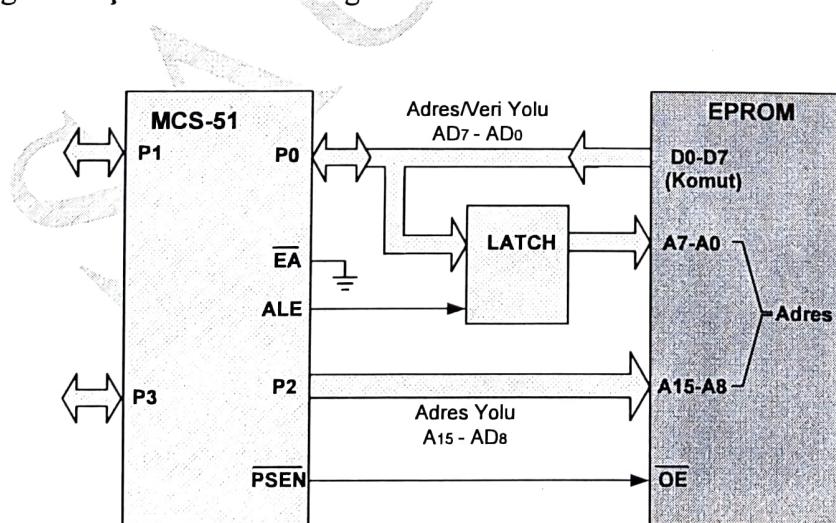
$\overline{EA} = 0$  olması durumunda ise mikrodenetleyici dahili program hafızaya sahip olsa bile harici program hafıza kullanılır. Makine kodları harici program hafızadan okunur.



Şekil 30. Aduc841'e ait Program (Dahili/Harici) ve Veri (Dahili/Harici) Hafıza yapıları

Aşağıda komutların harici program hafızadan koşturulabilmesi için gerekli donanım yapısını gösteren blok diyagram verilmiştir.

Harici ROM okuma sinyali PSEN, bütün harici program erişimlerinde kullanılır. Buna karşın, dahili program erişimlerinde aktif değildir.



Şekil 31. Harici Program hafıza kullanımı için gerekli bağlantı yapısı

Harici Program Hafıza için tipik bir donanım yapısı örneği Şekil 12'de görülmektedir. Port 0 ve Port 2'nin toplam 16-bit giriş/çıkış hattı, Program Hafıza erişimleri için, yol (bus) işlemlerine atanır. Harici program hafızaya erişimde P0 ve P2 (8-bit + 8-bit = 16-bit) portları harici program hafızadan koşturulacak komutu adreslemek için kullanılır. Adresin P0 adresin düşük byte ( $PCL=A_7-A_0$ ) değeri ALE (Address Latch Enable) komutu ile tutucuya (latch) tutturulur. P2 ise yüksek byte'ını ( $PCH=A_{15}-A_8$ ) oluşturur ve adres yoluna çıkarır. Daha sonra PSEN sinyali aktif edilerek EPROM'dan komut byte'ı mikrodenetleyiciye okunur.

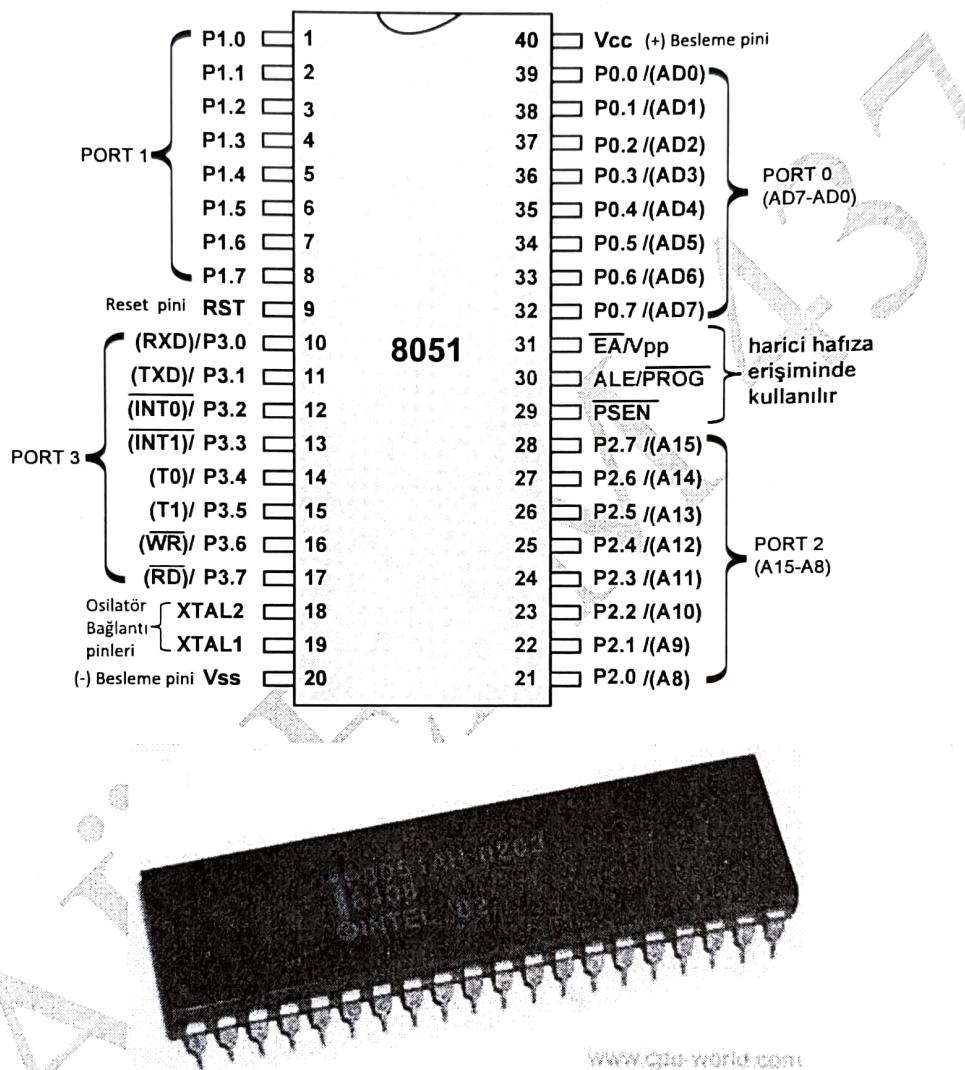
8051 ailesi 16-bit uzunluğunda adres yoluna sahiptir. Dolayısıyla adresleyebileceğि (kullanılabilecek) program hafıza (dahili veya harici) değeride 64KByte (0000h – 0FFFFh) ile sınırlıdır. Standart 8051 4-KByte büyüğünde dahili program hafıza sahip iken ailenin gelişmiş üyelerinde 62-KByte büyüğünde flash program hafıza bulunmaktadır.



## BÖLÜM 6 8051 TÜMDEVRE UÇ FONKSİYONLARI

### 8051 Tümdevre Uç Fonksiyonları

Standart 8051 mikrodenetleyicisi aşağıda gösterildiği gibi 40-uçlu (pin) tümdevre yapısına sahiptir. Bu pinlerden 32 tanesi 8-bitlik giriş/çıkış portları (P0,P1,P2 ve P3) için tahsis edilmiştir. Sınırlı sayıdaki pin sayısından dolayı standart giriş/çıkış işlemeye ilave olarak pinler aşağıda detaylı olarak belirtilen ilave fonksiyonlara sahiptirler.



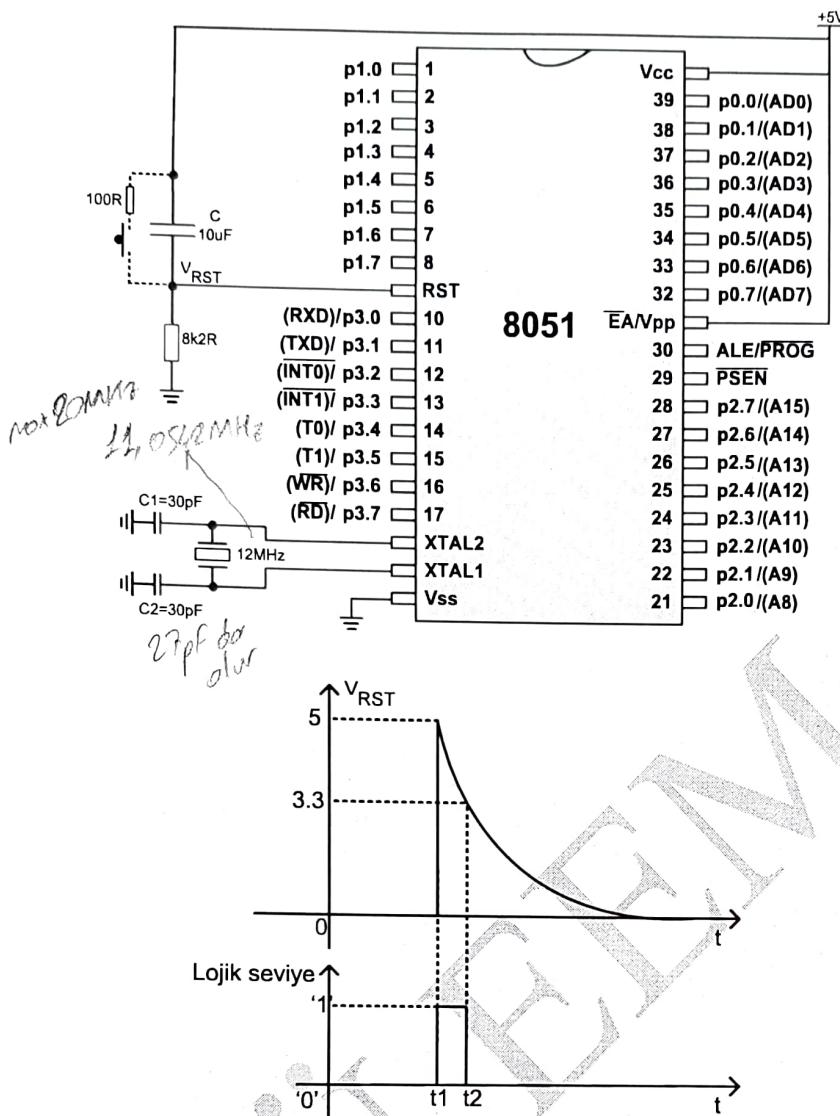
Şekil 32. 8051'e ait 40-uçlu tümdevre ve gerçek-zaman görüntüsü

#### 6.1. Besleme Uçları

8051' in beslemesi 20 (Vss, -) ve 40 (Vcc, +5V) numaralı pinlerden yapılır.

#### 6.2. RST (RESET)- Power on Reset

Mikrodenetleyicinin besleme gerilimi uygulanıp kararlı çalışmaya başlayabilmesi için RST pinine geçerli bir reset işaretü uygulanmalıdır. Geçerli reset işaretü için RST pini (8051-de 9 nolu pin) en az 2 makine çevrimi süresince Lojik 1 seviyesinde kalmalı ve daha sonra Lojik 0 seviyesine getirilmelidir. Reset işaretü farklı devreler kullanılarak üretilebilir. Aşağıda bir RC devresi kullanılarak gerçekleştirilen reset devresi verilmiş, besleme geriliminin uygulanması ile birlikte RST pinine etkiyen işaretin genlik ve lojik değişimi gösterilmiştir.



Şekil 33. Reset devresi ve reset işaretinin genlik ve Lojik değişimi

$t=t_1$  anında sisteme +5V besleme uygulandığında başlangıçta boş olan kondansatör RC zaman sabitine bağlı olarak dolmaya başlayacaktır. Bu durumda RST pinindeki lojik işaretin değişimi yukarıda gösterildiği gibi olacaktır. R ve C değerleri  $t_2-t_1 \geq 2$  makine çevrimi olacak şekilde seçilmelidir. **Geçerli bir reset işlemi sonrasında;**

- SP'in içeriği (gösterdiği adres) **07h**,
- PC değeri **0000h** olur. PC okunacak komutların kayıtlı Program Hafıza'daki (Code ROM) satırları okur ve geçerli reset sinyal geldiğinde PC 0000h'den itibaren okumaya başlar.
- Port (P0-P3) tutucuları **FFh** olur.

### 6.3. Osilatör Girişleri (XTAL1 ,XTAL2) %33,33

Mikroişlemcinin çalışabilmesi için sistem saat üreticinden gelen periyodik saat darbelerine ihtiyaç duyulur. Saat darbelerinin (zamanlama işaretinin) üretilebilmesi için XTAL1 ve XTAL2 pinlerine Şekil 2'de gösterildiği bir kristal ve 2 kondansatörden oluşan harici bir devre bağlanır. Uygulama gereksinimine göre farklı frekans ve tolerans değerlerine sahip kristaller kullanılabilir. Ancak kristal frekansı üretici firma tarafından belirlenen sınır değeri aşmamalıdır.

#### 6.4. Harici Hafıza Erişim Pinleri

ALE/PROG (Address Latch Enable), PSEN (Program Store Enable) ve  $\overline{EA}$ /VPP (External Access) pinleri harici hafıza erişim sürecinde kullanılan pinlerdir. ALE/PROG pini harici hafızaya (program/veri) erişimlerde adresin düşük değerli (A7-A0) Byte'ını, adres tutucusuna tutturmak amacıyla kullanılır. PSEN pini harici program hafızaya erişimlerde kullanılır.  $\overline{EA}$ /VPP pini program hafızanın seçiminde, harici ( $\overline{EA}$ /VPP=0) veya dahili ( $\overline{EA}$ /VPP=1), kullanılır. Bu pin mutlaka GND veya Vcc'pinine bağlanmalıdır, boşta bırakılmamalıdır.

#### 6.5. Giriş/Çıkış Pinleri

8051 mikrodenetleyicisinde 8-bitlik 4 adet (P0,P1,P2 ve P3) çift yönlü (hem giriş hem de çıkış, I/O) olarak kullanılabilen port bulunur. P1 dışındaki tüm portlar giriş/çıkış işlevi yanı sıra aşağıda detaylı olarak belirtilen ilave fonksiyonlara sahiptir. Portlar bir bütün (8-bit) olarak okunup yazılabilceğ gibi portların her bir pine ayrı ayrı bağımsız olarak bit tabanlı komutları kullanarak erişmek de mümkündür.

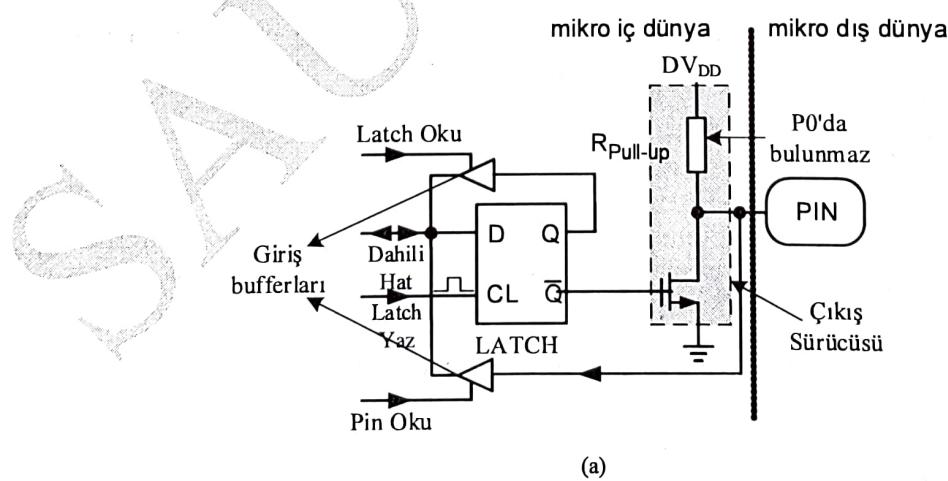
**NOT:** Reset sonrası tüm pinler Lojik 1 seviyesine çekilerek giriş olarak atanır.

KAYDEDİCİ	RESET DEĞERİ (BINARY)
P0	11111111
P1	11111111
P2	11111111
P3	11111111

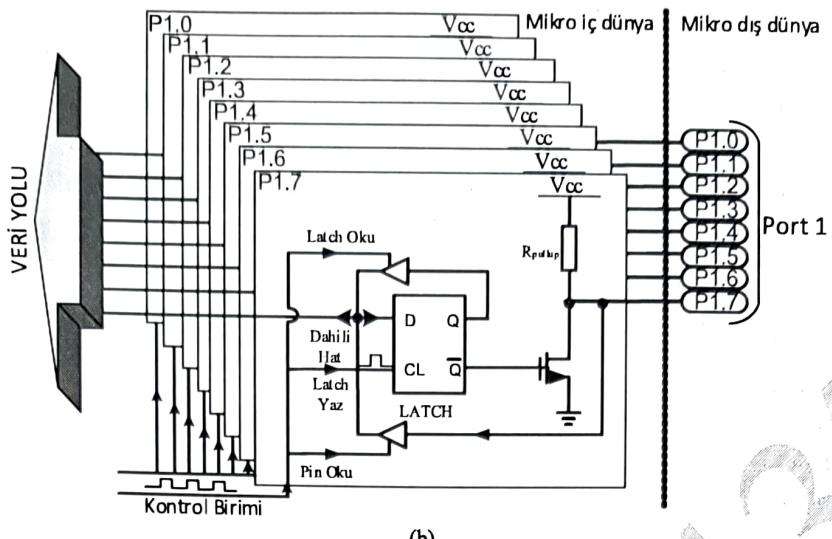
##### 6.5.1 Port 1 (P1)

P1 portu 8 bitlik bir giriş/çıkış portudur. Dahili pull-up dirençlerine sahip olan P1 portunun I/O dışında ilave bir fonksiyonu yoktur.

Dış dünya ile mikrodenetleyici arasında Lojik (0 veya 1) veri传递ası sağlayan port 1'e ait pinlerin her biri Şekil 3'de gösterildiği gibi bir tutucuya (Latch), bir çıkış sürücüsüne (driver) ve giriş tampon (buffer) elemanlarına sahiptir.



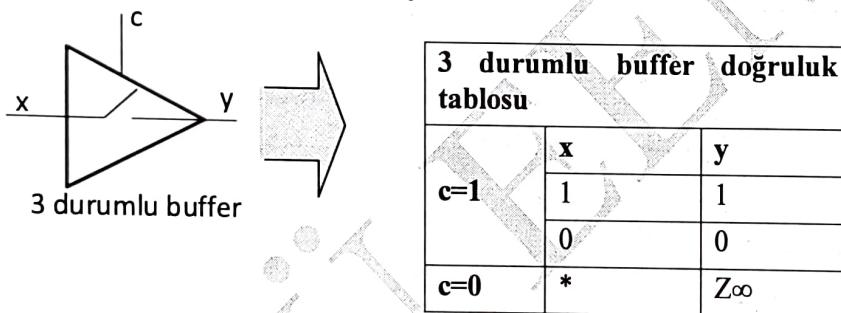
mosfetin re dizer  
R pull-up a göre değişik olmalı



Şekil 34. (a) P1 portu için 1-bit (P1.x) basitleştirilmiş donanım yapısı (b) P1 portuna ait basitleştirilmiş donanım yapısı

Şekil 3, P1 portuna ait 1-bit (Örn. Port 1.1) giriş-çıkış için basitleştirilmiş donanım yapısını göstermektedir.

Mikrodenetleyicide bulunan tampon (buffer) 3 durumlu (3 state) yapıya sahiptir. Aşağıda giriş tamponun yapısı ve doğruluk tablosu verilmiştir.



Mikroişlemcide bazı komutlar port'a ait fiziksel pinlerin değerini okurken bazıları ise o pinler'e ait tutucuların durumlarını okumaktadır. 8051 mikrodenetleyicisinde portlara ait pinlerin okumasında iki adet durum söz konusudur;

1. Pin'e (fiziksel pin) ait değeri okuma
2. Tutucu'yu okuma

Aşağıda Tabloda verilen komutların kullanımında “komut <hedef>,<kaynak>” durumu göz önünde bulundurulduğunda eğer;

- Hedef operand herhangi bir port veya portun bir biti ise komutun koşturulmasında ilk önce tutucu değeri okunur, okunan veri üzerinde işlem yapılır ve oluşan yeni değer tekrar tutucuya yazılır. Bu işlemlerin hepsi donanım tarafından otomatik olarak yerine getirilir.
- Eğer port veya portun bir biti kaynak operand ise bu durumda komutun koşturulmasında portun fiziksel pinlerindeki değer okunur.

**Örneğin;** ANL komutunun “ANL P1, A” şeklindeki kullanımında P1 portu hedef operand’dır. Bu durumda komut koşturulurken donanım P1 portuna ait tutucuyu (latch) okur, akümülatördeki değer ile lojik ve işlemeye tabi tutar ve ardından lojik ve işleminin sonucunu P1 tutucularına yazar. Fakat, ANL komutunun “ANL A, P1” şeklindeki kullanımında ise P1 portu kaynak operand’dır. Bu durumda komut koşturulurken donanım P1 portuna fizikselse pinleri okur akümülatördeki değer ile lojik ve işlemeye tabi tutar ve ardından lojik ve işleminin sonucu akümülatöre yazılır.

#### Pin Okuma komutları:

Komut	Açıklama
MOV a,Px	İlgili Port'u okur ve P1 fizikselse pinden gelen bilgiyi a'ya kaydeder
JNB Px.y,<adres>	Px.y lojik 0 ise ilgili adrese dallan, JNB P2.3, X1
JB Px.y,<adres>	Px.y lojik 0 ise ilgili adrese dallan, JB P1.2,X2
MOV C, Px.y	Pinin durumunu Carry bitine aktar, MOV C, P2.2

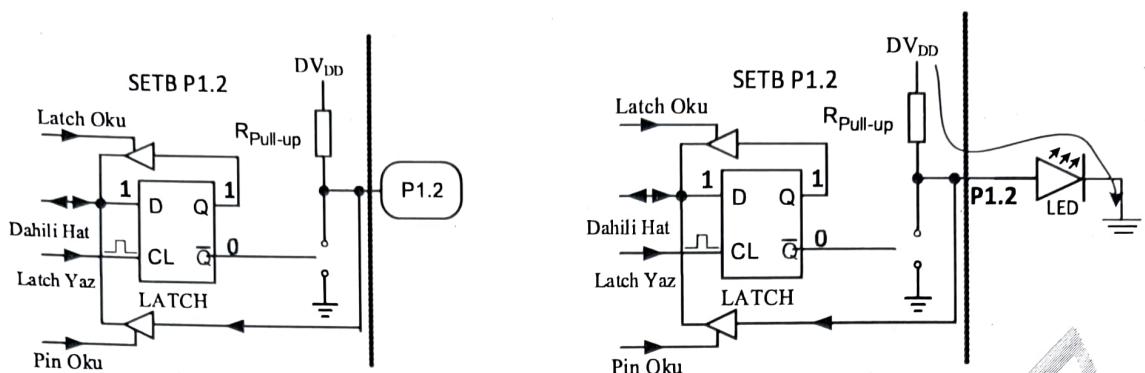
#### Oku/Değiştir/Yaz (Read/Modify/Write) komutları- Tutucu üzerinde işlem yapan komutlar:

Komut	Açıklama
ANL Px, a	Lojik “VE” işlemi <b>ANL P1, A</b>
ORL Px, a	Lojik “VEYA” işlemi <b>ORL P2, #55h</b>
XRL Px, a	Lojik “ÖZEL-VEYA” işlemi <b>XRL P3, #0AAh</b>
JBC Px.x, <adres>	Eğer bit=1 ise adrese dallan sonra biti sıfırla <b>JBC P2.0, devam</b>
CPL Px.x	Biti tersle <b>CPL P3.2</b>
INC Px	<b>INC P2</b>
DEC Px	<b>DEC P3</b>
DJNZ Px, <adres>	<b>DJNZ P2, git</b>
MOV Px.y, C *	<b>MOV P2.3, C</b>
CLR Px.y *	<b>CLR P3.6</b>
SETB Px.y *	<b>SETB P0.3</b>

\*Bu komutların icrasında ilk önce portun tutucuları (8-Bit'in tamamı) okunur, ilgili bitin değeri değiştirilir ve ardından oluşan yeni Byte değeri tutuculara yazılır.

Not: Tablo'da x, 0,1,2, ve 3'ü göstermektedir y ise 0,1,2...7'yi göstermektedir. (P0-P3)

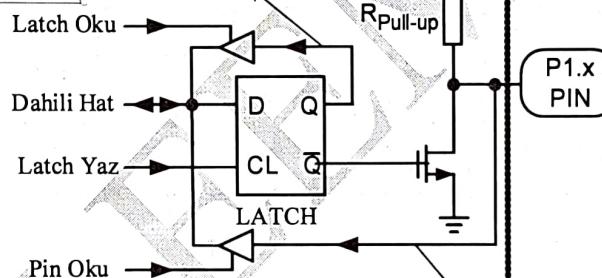
**Örnek:** setb P1.2 komutu ile 8051 mikrodenetleyicisinin P1 portunun P1.2 bitinde otomatik olarak gerçekleşen lojik seviyeler aşağıda temsilen gösterilmiştir.



**NOT:** "Pin oku" ve "Latch oku", "Latch yaz" sinyalleri ilgili komut ID (Instruction Decoder) çözüldükten sonra mimarideki "kontrol birimi" tarafından üretilir.

### Latch okuma

Komut	
ANL Px, a	INC Px
ORL Px, a	DEC Px
XRL Px, a	DJNZ Px, <adres>
JBC Px.x, <adres>	MOV Px.y, C
CPL Px.x	CLR Px.y
SETB Px.y	



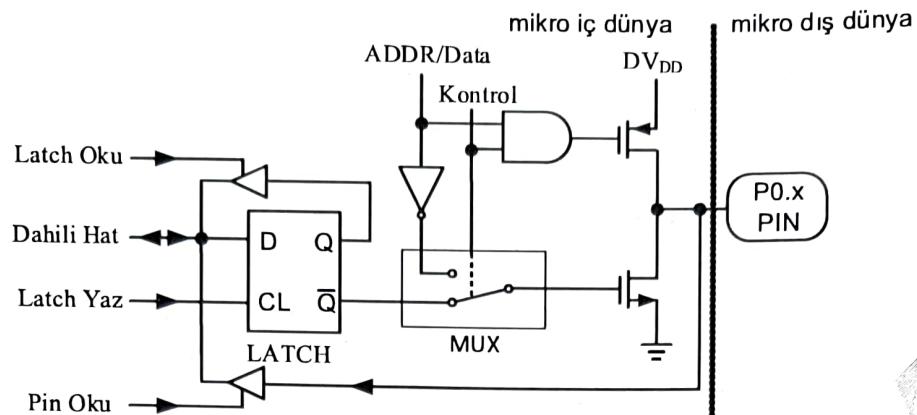
### Pin okuma

Komut
MOV a,Px
JNB Px.y,<adres>
JB Px.y,<adres>
MOV C, Px.y

Sekil 35. Port 1'in bir pini üzerinde Latch ve pin okuma komutlarının gösterilmesi

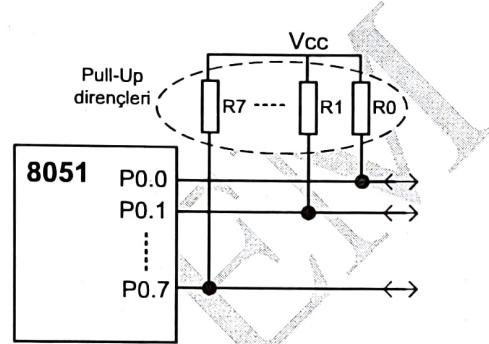
## 6.5.2 Port 0 (P0)

P0 standart I/O işlemi dışında harici hafıza erişiminde de kullanılan açık-kollektörlü (open-drain) yapıya sahip porttur. P0 aynı zamanda, harici Program Hafızaya ve Veri Hafızaya erişimlerde, seçmeli (multiplexed) düşük byte ( $A_7-A_0$ ) değerini (Adres/Veri Yolu) olarak da kullanılabilir. Bu durumda çalışırken, P0, dışarı lojik 1 işaretini göndermede, dahili pull-up'lar olarak çalışan FET transistörlerini kullanır. Adresleme sonrasında ise harici hafızaya yazılacak veya harici hafızadan okunacak veri P0 üzerinden taşınır. Aşağıda Port 0'ın herhangi bir pini için basitleştirilmiş iç yapı verilmiştir. P0 uclarına program ile 1 yazılması durumunda, yüksek empedanslı giriş uçları olarak kullanılabilir.



Şekil 36. Port 0 detaylı iç yapısı, 1-bit için.

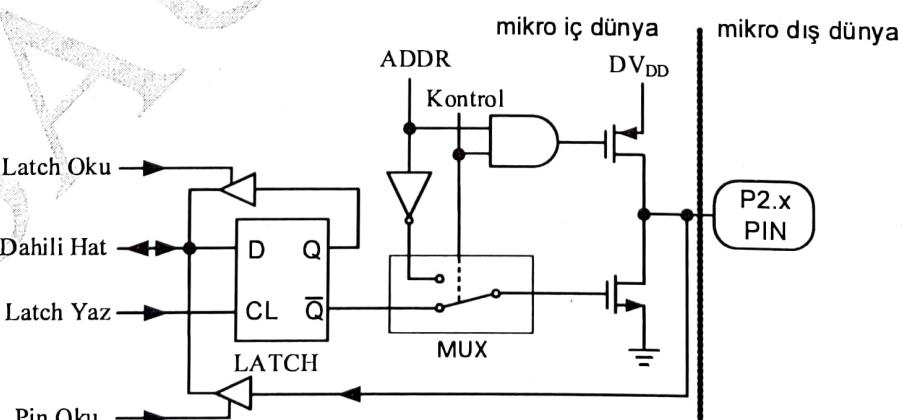
P0 diğer portlardan farklı olarak açık-kollektörlü (open-drain) olarak tasarlanmıştır. Başka bir ifade ile P0 pinleri dahili pull-up dirençlerine sahip değildir. Bu haliyle P0 pinleri sadece giriş olarak kullanılabilir. P0 pinlerinin çıkış olarak da kullanılabilmesi için aşağıda gösterildiği gibi harici pull-up dirençleri eklenmesi gereklidir.



Şekil 37. Port 0 harici pull-up direnç bağlantısı.

### 6.5.3. Port 2 (P2)

P2 portu I/O işlemi dışında harici hafıza erişiminde de kullanılır. P2 harici hafıza erişimlerde erişilmek istenen hafıza adresinin yüksek byte değerini ( $A_{15}-A_8$ ) oluşturur. Standart 8051 entegresi P2 ve P0 portlarını kullanarak 16-bitlik hafıza adresi ( $A_{15}-A_0$ ) üzerinden 64 kByte'lık harici hafızayı adresleyebilmektedir.

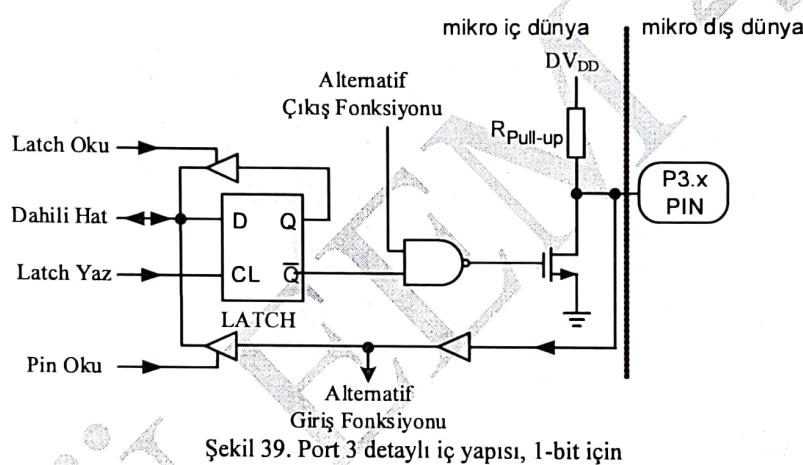


Şekil 38. Port 2 detaylı iç yapısı, 1-bit için.

#### 6.5.4. Port 3 (P3)

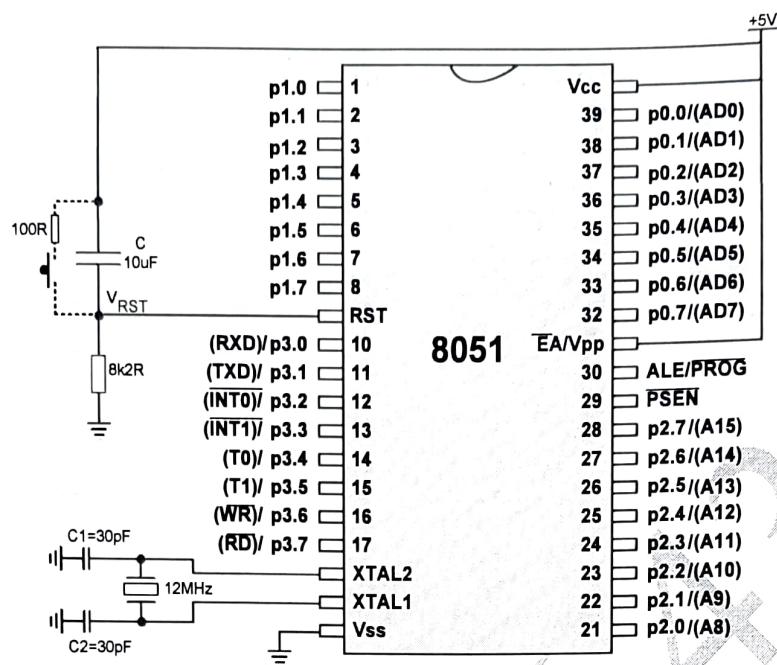
P3 portuna ait pinlerin her biri giriş/çıkış işlemi dışında ilave fonksiyonlara sahiptir. Aşağıda tabloda P3 pinlerine ait ilave fonksiyonlar belirtilmiştir.

Pin	İsim	İkincil Fonksiyon
P3.0	RxD	Seri haberleşme veri giriş pini
P3.1	TxD	Seri haberleşme veri çıkış pini
P3.2	INT0	Dış (harici) kesme 0 ve T/C- 0 için harici start/stop girişi
P3.3	INT1	Dış (harici) kesme 1 ve T/C- 1 için harici start/stop girişi
P3.4	T0	T/C- 0 için harici darbe girişi
P3.5	T1	T/C- 1 için harici darbe girişi
P3.6	WR	Harici hafıza yazma işaretü
P3.7	RD	Harici hafızadan okuma işaretü



Şekil 39. Port 3 detaylı iç yapısı, 1-bit için

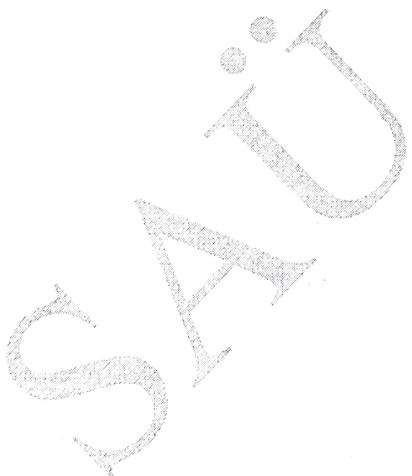
- 8051 mikrodenetleyicisinin çalışabilmesi için gerekli olan asgari bağlantılar (Besleme, Reset devresi, Osilatör ) Şekil 9'da gösterilmiştir.

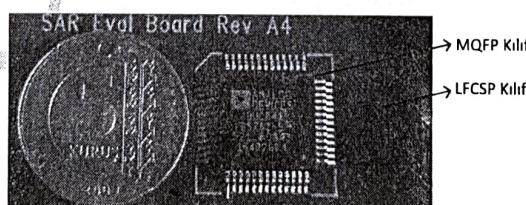
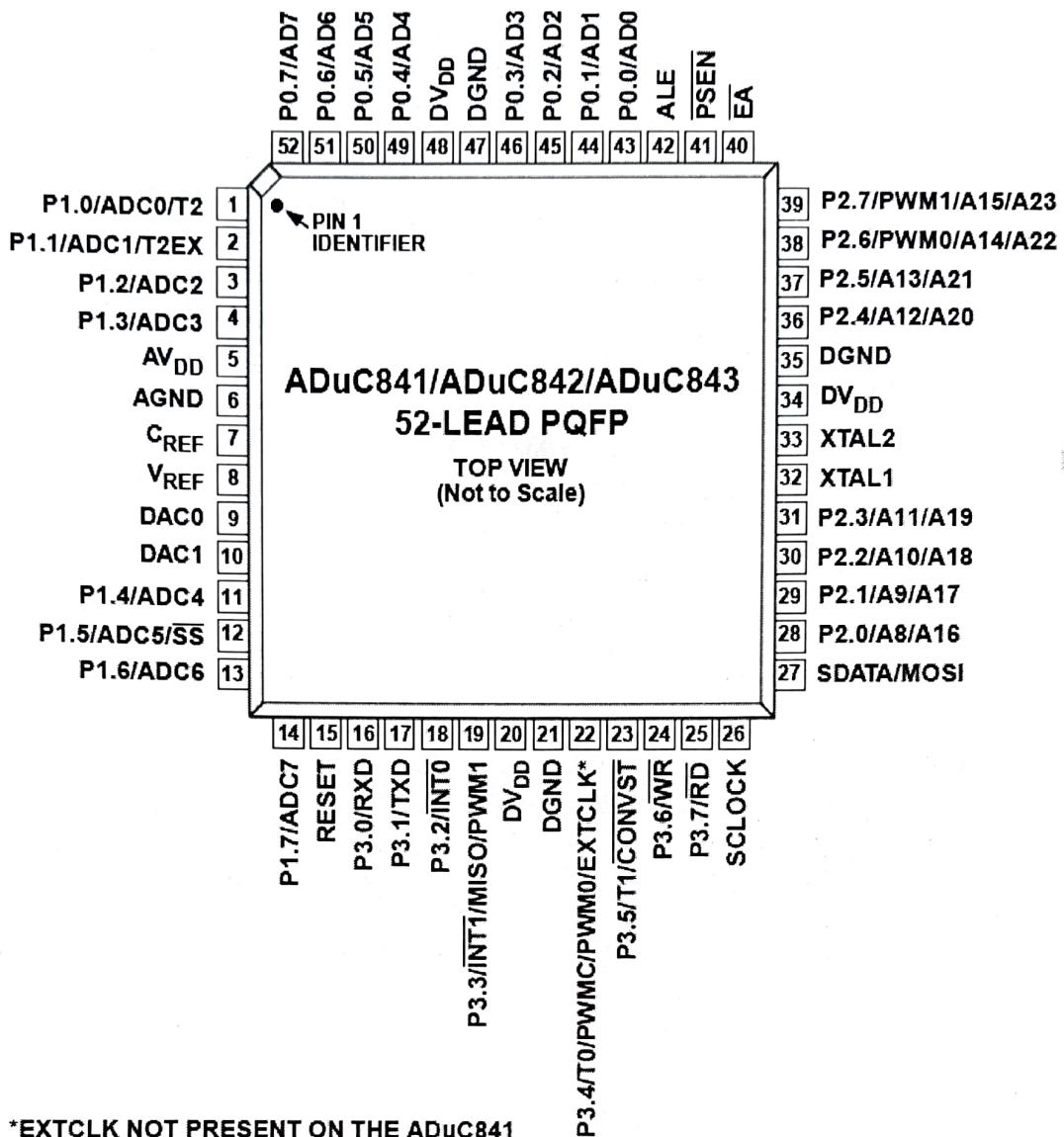


Şekil 40. 8051 mikrodenetleyicisinin çalışabilmesi için gerekli olan asgari bağlantı

## 6.6. Aduc841 Tümdevre Uç Fonksiyonları

Aduc841'in 52-pinli tümdevre yapısı aşağıda gösterilmiştir. Standart 8051 ile kıyasla fazladan 12 adet pine sahiptir. İlave pinlerin 6 tanesi Aduc841'in ilave çevre birimlerinin (DAC0, DAC1, SDATA, MOSI, CREF, VREF) giriş çıkış uçları diğer 6 tanesi ise ilave çevre birimlerinin beslemesi (AVDD, AGND, DVDD, DGND) içindir.





Sekil 41. Aduc 841 Tümdevre uç fonksiyonları ve gerçek zaman görüntüsü

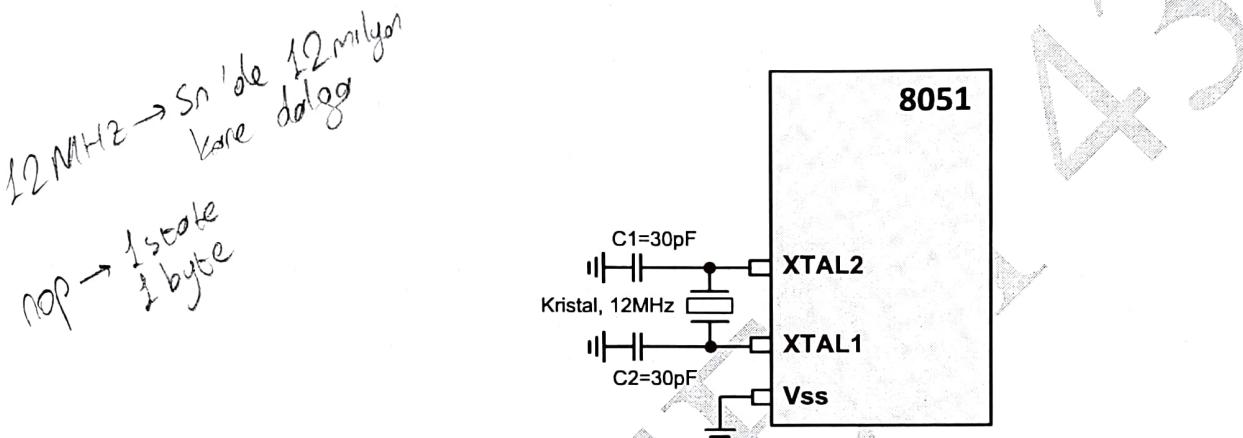
Aduc841 mimarisinde P0, P2 ve P3 portlarının kullanımı standart 8051 ile aynıdır. Ancak P2 ve P3 portlarının bazı pinlerine Şekil 9'da gösterildiği gibi ilave alternatif fonksiyonlar yüklenmiştir. Benzer şekilde P1 portuna standart 8051'den farklı olarak ilave fonksiyonlar (Analog giriş kanalları ve T/C-2 kontrol girişleri) yüklenmiştir. Ayrıca Aduc841'de P1 portu port mimarisinde yapılan değişikliklerden dolayı digital çıkış olarak kullanılamaz.

## BÖLÜM 7 ZAMANLAYICI/SAYICI YAPISI:

### 7.1. Sistem Saat üreteci ve Makine Çevrimi

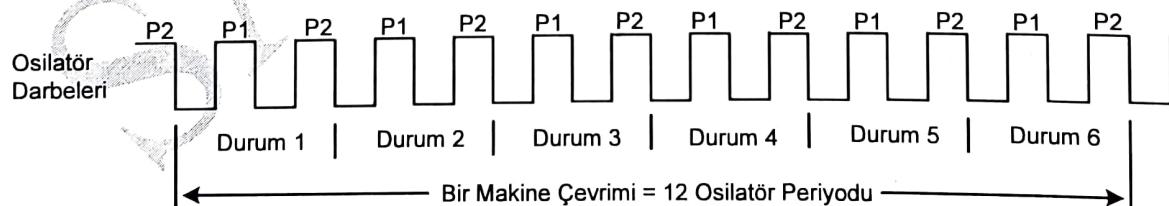
Bilgi:

Saat üreteci bir mikrodenetleyicinin fonksiyonlarını yerine getirebilmesi için gerekli olan saat darbelerini üreten birimdir. Bu saat darbeleri mikrodenetleyicinin kalp atışları gibidir, mikrodenetleyicini bütün fonksiyonları bu saat darbeleri sayesinde gerçekleştirir. Tüm 8051 mikrodenetleyicilerinde tümdevre-üzeri saat üreteci (dahili osilatör) mevcuttur. Dahili osilatörün saat darbelerini üretmesi için XTAL1 ve XTAL2 pinlerine Şekil 1 de gösterildiği gibi kristal veya seramik resonatör ve iki kondensatör harici olarak bağlanmalıdır. Uygulama gereksinimine göre farklı frekans değerlerine sahip kristaller kullanılabilir. Mikrodenetleyiciye bağlanan kristal frekansı üretici firma tarafından belirlenen maksimum değeri aşmamalıdır. (8051-16 Mhz, ADUC841-20 Mhz max.)



Saat (osilatör) darbelerinin frekansı XTAL1 ve XTAL2 pinlerine bağlanan kristal frekansı ile aynıdır. 8051 makine çevrimi (periyodu) aşağıda gösterildiği gibi 6 durumdan, her durumda faz 1 ve faz 2 (P1 ve P2 ) şeklinde 2 osilatör periyodundan oluşur.

Sonuç olarak her durum 2 osilatör periyodu uzunluğunda olduğundan 8051'in bir makine çevrimi 12 osilatör periyodundan oluşur. Ders boyunca standart 8051 için kristal frekansı 12 MHz olarak kabul edilerek, bir makine çevriminin süresi, 12 osilatör periyodu = 1 us alınacaktır.



Makine çevrim süresi mikrodenetleyicinin işlemlerini yerine getirme hızını belirler. 8051, 255 farklı işlem koduna (opcode) sahip olup 111 komut olarak gruplandırılmıştır. Örneğin, byte tabanlı veri transfer (MOV) komutları için, 15 farklı işlem kodu vardır. Bu farklılık, komutlardaki kaynak ve hedef byte'lardan ve adresleme modlarından kaynaklanır. 255 işlem kodundan, 159 tanesinin yürütülmesi 1 makine çevrimi içinde gerçekleşir. Diğer işlem kodlarından 51 tanesi 2 makine çevrimi, 43 tanesi 3 makine çevrimi ve 2

tanesi de 4 makine çevrimi süresinde yürütülür. Sık olarak kullanılan komutlar, sadece 1 makine çevrimi gerektirdiği için (1 makine çevrimi = 1us) 8051 saniyede yaklaşık 1 Milyon komut yürütübilme (1 MIPS—Million Instructions Per Second) kapasitesine sahip bir işlemci olarak belirtilir.

Genel olarak standart 8051'de herhangi bir komutun koşturma süresi, Cycle= komutun koşturulması için gerekli makine çevrimi olmak üzere,

$$T = \frac{\text{Cycle} * 12}{\text{Kristal Frekansı}} \quad (\text{sn})$$

ifadesi ile hesaplanabilir.

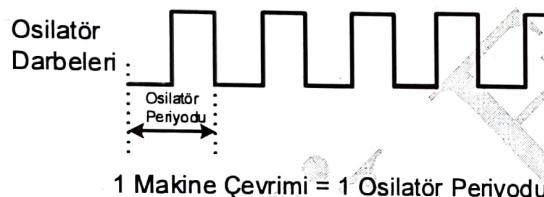
Örneğin “mov R0, 45h” komutu 2 makine çevriminde koşturulur (Kod Tablosunu inceleyiniz). Dolayısıyla bu komutun 12 MHz kristal takılı olan 8051 mikrodenetleyicisi için koşturulma süresi

$$T = \frac{2 * 12}{12 * 10^6} = 2 * 10^{-6} \text{ sn} = 2\mu\text{s}$$

olarak bulunur.

## 7.2 Aduc841 Makine Çevrimi

Maksimum çalışma frekansı 20-Mhz olan Aduc841 mikrodenetleyicisinde 1 makine çevrimi 1 osilatör periyodundan oluşur. Dolayısı ile Aduc841 aynı kristal frekansı değerinde bile standart 8051'den 12 kat daha hızlı işlem gerçekleştirir. (Standart 8051'de 1 makine çevrimi=12 osilatör periyodu)



Aduc841 için bir komutun koşturma süresi,

$$T = \frac{\text{Cycle}}{\text{Kristal Frekansı}} \quad (\text{sn})$$

ifadesi ile hesaplanır.

Bu durumda “mov R0, 45h” komutunun 12 MHz kristal takılı olan Aduc841 mikrodenetleyicisinde koşturulma süresi

$$T = \frac{2}{12 * 10^6} \approx 0.167 * 10^{-6} \text{ sn} \approx 0.167\mu\text{s}$$

olarak bulunur.

Gördüğü gibi aynı kristal frekans değerinde Aduc841 standart 8051'e oranla 12 kat ( $2/0.167 = 12$ ) hızlı işlem yapmaktadır. Bunun sebebi yukarıda belirtildiği gibi standart 8051'de 1 makine çevrimi=12 osilatör periyodu iken Aduc841'de 1 makine çevrimi=1 osilatör periyodu olmasıdır.

**NOT:** Geliştirme kartlarında yer alan Aduc841 mikrodenetleyicilerinde 11.0592 MHz kristal takılmıştır. Bu durumda 1 makine çevrimi

$$T = \frac{1}{11.0592 * 10^6} = 90.4 * 10^{-9} \text{ sn} = 90 \text{ ns}$$

olarak bulunur.

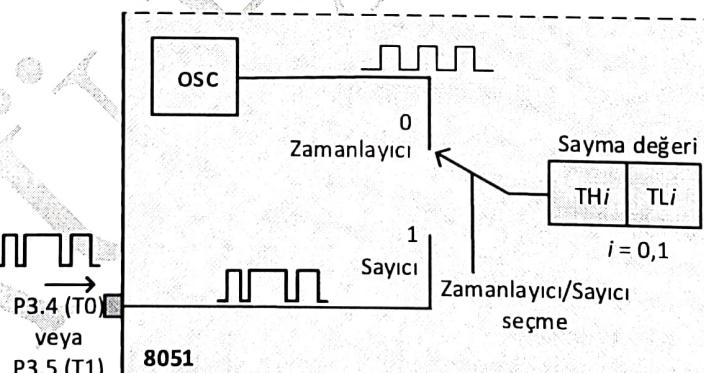
### 7.3. Zamanlayıcı/Sayıci

Mikrodenetleyici tabanlı uygulamalarda belirli bir zaman gecikmesi oluşturmak veya belirli bir zaman aralığını ölçmek veya bir olayın gerçekleşme sayısını saymak gibi ihtiyaçlar oluşabilir. Bu tür işlemlerin gerçekleştirilmesi amacı ile mikrodenetleyici mimarilerinde zamanlayıcı/sayıci (Timer/Counter, T/C) çevre birimi bulunur.

$\text{CT} \rightarrow \text{Counter Timer}$

ADUC841 ve 8051 mimarisinde zamanlayıcı/sayıci-0 (Timer/Counter, T/C-0) ve zamanlayıcı/sayıci-1 (T/C-1) olarak adlandırılan, yukarı yönde sayan, 2-adet 16-bit T/C bulunur. (ADUC841'de T/C 2 Zamanlayıcı 2 de bulunur, bu zamanlayıcı ileride anlatılacaktır.) Özellikleri;

- Yani T/C tek bir elemandır.
- T/C start verilirse aşağıdan yukarıya doğru sayar.
- İstenirse sayıciya ilk değer verilebilir bu durumda ilk verilen değerden yukarıya doğru sayar.
- Zamanlayıcı/sayıci birimi aşağıda gösterildiği gibi aynı donanımı kullandığından herhangi bir anda ya zamanlayıcı ya da sayıci olarak kullanılabilir.
- Zamanlayıcı/sayıci kullanmanın avantajı:
- Merkezi işlem birimi CPU yazılımın gerektiği anlar dışında sayma işlemi ile ilgilenmediği için sayma işlemleri işlemcinin performansını etkilemez.



Şekil 43. Zamanlayıcı/Sayıci basitleştirilmiş blok diyagramı

Anahtar 0 konumuna bırakıldığından her makine çevriminde (1 osilatör darbesi= 1 makine çevrimi) THi-TLi saklayıcılarında tutulan sayma değeri 1 artar. Başka bir ifade ile bu çalışma modunda sistem saatinden (Osc) gelen periyodik darbeler sayılır. Sayma değerinin artışı periyodik olduğundan herhangi bir andaki sayma değeri başlangıçtan itibaren geçen süreyi gösterecektir. Bu nedenle bu çalışma modu "zamanlayıcı" olarak adlandırılır. ADUC841'de (11.0592 MHz kristal kullanıldığında) 1 makine çevrimi = 90.4 nsn'dir. Dolayısıyla ADUC841'de anahtarın 0 konumunda her 90.4 nsn'de THi-TLi saklayıcılarında tutulan sayma değeri 1 artacaktır.

Anahtar 1 konumuna ayarlandığında ise dış dünyadan gelen işaretin her 1-0 geçişinde sayma değeri 1 artar. Başka bir ifade ile bu çalışma modunda harici lojik darbeler sayılır. Bu nedenle bu çalışma modu

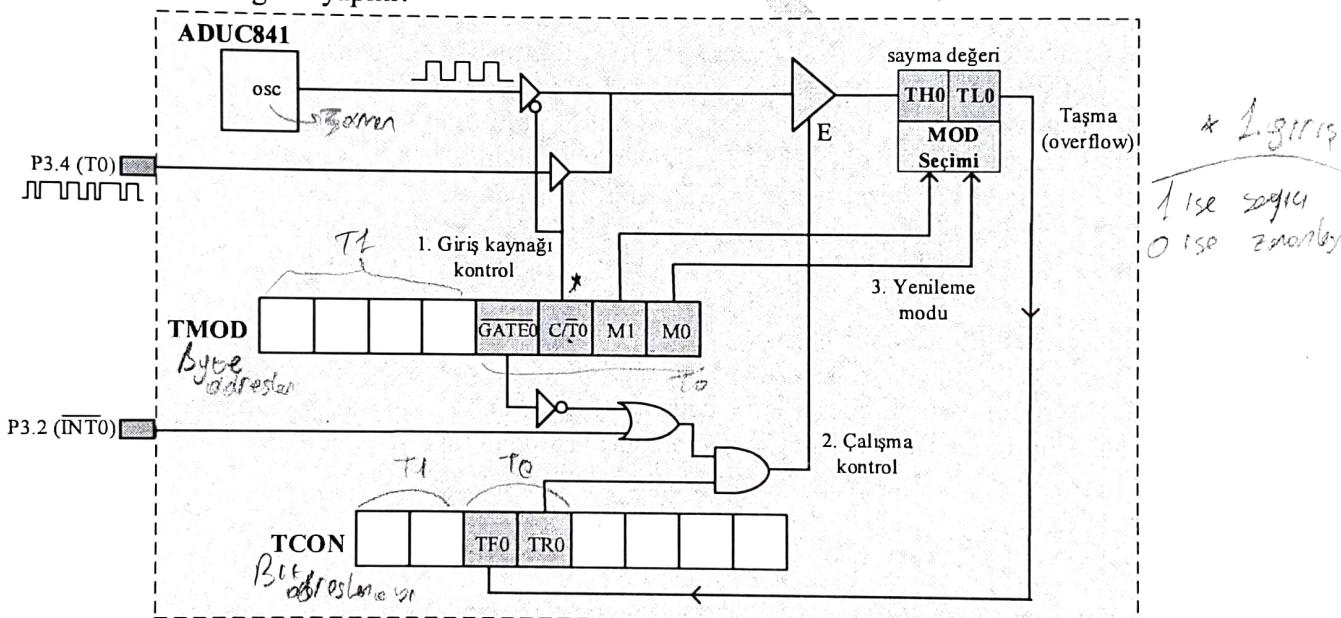
“sayıcı” olarak adlandırılır. Sayıcı çalışma modu için harici lojik darbeler mikrodenetleyicinin bu amaç için atanmış 2 pinine {Sayıcı-0 için P3.4(T0) ve Sayıcı-1 için P3.5 (T1)} bağlanabilir.

Her iki zamanlayıcı/sayıçı için de 2'şer adet anlık sayma değerinin tutulduğu 8-bitlik veri SFR'sı, T/C-0 için TH0 ve TL0, T/C-1 için TH1 ve TL1, vardır. Bu saklayıcıların reset sonrasında değerleri 00h'dır. Dolayısı ile zamanlayıcı/sayıçı'lar için sayma başlangıç değeri 0000h'dır. Ancak sayma değerinin belli bir değerden başlatılmasının uygun veya gerekli olduğu durumlar olabilir. Bu gibi durumlarda TH<sub>i</sub> ve TL<sub>i</sub> saklayıcılarına istenilen sayma başlangıç değeri yüklenerek bu değerden itibaren sayması sağlanabilir. Benzer şekilde zamanlayıcı/sayıçı'ların herhangi bir andaki sayma değeri TH<sub>i</sub> ve TL<sub>i</sub> ( $i=0,1$ ) saklayıcılarından okunabilir.

4 farklı çalışma moduna sahip olan T/C-0 ve T/C-1'in amaca uygun kullanılabilmesi için çalışma ayarlarının doğru olarak yapılması gereklidir. Çalışma ayarlarını:

- i) giriş kaynağının belirlenmesi
  - ii) çalışma kontrolünün belirlenmesi
  - iii) çalışma modunun belirlenmesi

şeklinde özetlemek mümkündür. ADUC841 mimarisinde bulunan T/C-0 ve T/C-1'in çalışma ayarları aşağıda verilen basitleştirilmiş blok diyagramında gösterildiği gibi TMOD ve TCON SFR'lerinde bulunan kontrol bitleri aracılığı ile yapılır.

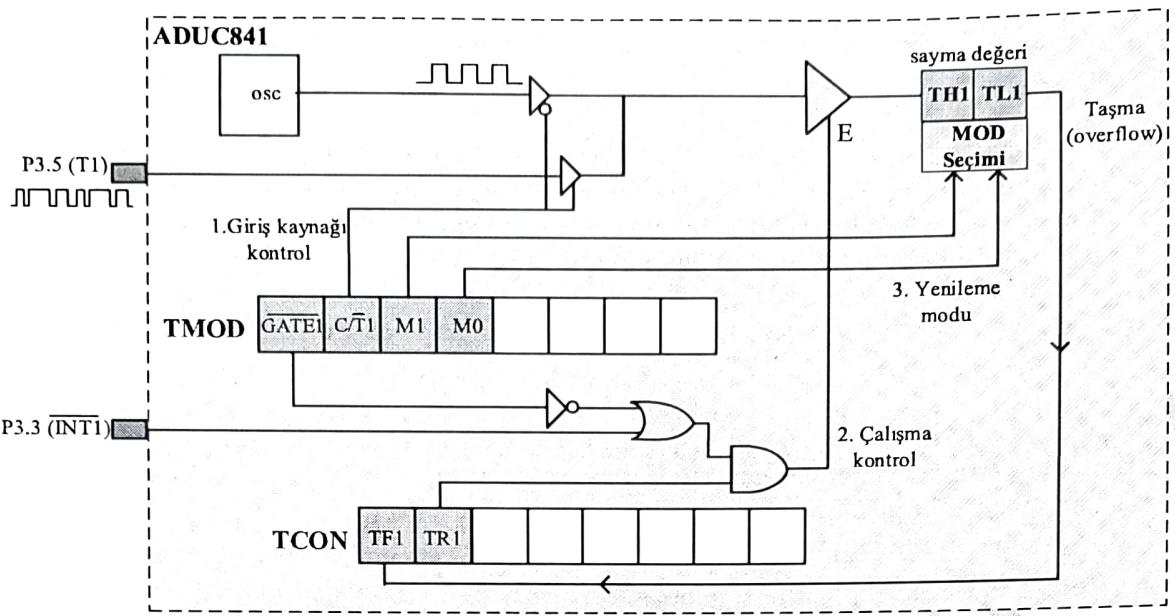


Şekil 44. Zamanlayıcı/Sayıçı-0 detaylı blok diyagramı

anayıcı/Sayıçı-0 detaylı blok diyagramı  
 setb TR0 %100 olacak  
~~EOTE → 0~~) 2. çalışma Kısıtları

TFO 11se 255 ohnser

Times I can syn. - so dec. because now definate



Şekil 45. Zamanlayıcı/Sayıcı-1 detaylı blok diyagramı

#### Giriş kaynağının belirlenmesi:

Zamanlayıcı/sayıcılar için giriş kaynağı ya osilatörden gelen saat darbeleri yada harici pinden {T/C-0 için P3.4(T0) ve T/C-1 için P3.5 (T1)} gelen lojik darbelerdir. Giriş kaynağının seçimi TMOD saklayıcısında bulunan  $C/\bar{T}i$  ( $i=0,1$ ) kontrol biti ile gerçekleştirilir. Verilen kontrol blok diyagamlarından görüldüğü gibi  $C/\bar{T}i$  biti temizlendiğinde ( $C/\bar{T}i=0$ ) giriş kaynağı olarak osilatörden gelen saat darbeleri seçilecek ve böylece ilgili T/C zamanlayıcı olarak ayarlanmış olacaktır. Benzer şekilde gibi  $C/\bar{T}i$  biti setlendiğinde ( $C/\bar{T}i=1$ ) T/C harici lojik darbeleri sayma üzere sayıci olarak ayarlanmış olacaktır.

#### Çalışma kontrolünün belirlenmesi:

T/C-0 ve T/C-1 için yukarıda verilen blok diyagamlarından görüldüğü gibi seçili olan giriş kaynağının sayma değerini artırabilmesi için çalışma kontrolünü sağlayan 3-durumlu tampon elemanın yetkilendirilmiş olması ( $E=1$ ) gereklidir. Bu amaçla yetkilendirme işaretini üreten lojik devre incelendiğinde;

$$E = (\overline{GATE}_i \text{ veya } INT_i) \text{ ve } TR_i, i=0,1$$

ifadesinin  $TR_i = 1$  olarak ayarlanmak koşulu ile 2 farklı durum için “ $E=1$ ” olabileceği dolayısıyla zamanlayıcı/sayıcılar için çalışma kontrolünün 2 farklı şekilde yapılabileceği görülür.

i)  $\overline{GATE}_i = 0$  iken  $TR_i = 1$  yapıldığı anda  $E=1$  olacağından 3-durumlu tampon elemanı yetkilendirilmiş olur. Böylece seçili olan giriş kaynağından gelen darbeler sayma değerini artırır. Başka bir ifade ile zamanlayıcı/sayıci aktif olur.  $TR_i = 0$  yapıldığı anda ise zamanlayıcı/sayıci pasif duruma geçer. Sayma değeri değişmez, son değerini korur.

ii)  $\overline{GATE}_i = 1$  iken  $TR_i = 1$  yapılması zamanlayıcı/sayıciyi aktif yapmak için yeterli değildir. Bu durumda zamanlayıcı/sayıcinın aktif olabilmesi için P3.2 (INT0) {T/C-1 için P3.3(INT1)} pininden gelen işaret Lojik 1 seviyesinde olmalıdır. P3.2 (INT0) pininden gelen işaret Lojik 0 seviyesinde olduğu sürece zamanlayıcı/sayıci pasif durumda kalır.

Yukarıda açıklanan çalışma modlarından birincisi yazılımla start/stop diğeri ise harici start/stop çalışma olarak adlandırılır. Ancak her iki çalışma modunda da  $TR_i=0$  olduğu sürece zamanlayıcı/sayıcinın pasif

durumda kalacağına dikkat edilmelidir.

### Çalışma modunun belirlenmesi:

T/C-0 ve T/C-1 TMOD saklayıcısının M0 ve M1 bitleri ile belirlenen 4 farklı çalışma moduna sahiptir.

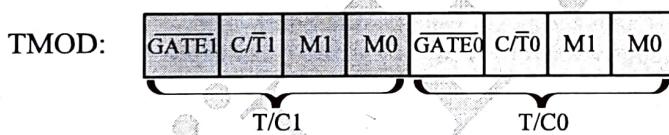
M1	M0	Mod	Açıklama
0	0	0	13-bit zamanlayıcı/sayıci modu.
0	1	1	16-bit zamanlayıcı/sayıci modu.
1	0	2	8-bit otomatik yüklemeli zamanlayıcı/sayıci modu.
1	1	3	Bağımsız çalışma modu. T/C-1 bu çalışma modunda durdurulur, son sayma değeri TH1 ve TL1 saklayıcılarında tutulmaya devam eder. T/C-0 ise 2-adet bağımsız 8-bit'lik zamanlayıcı veya 1-adet 8-bit'lik zamanlayıcı ve 1-adet 8-bit'lik sayıci olarak çalışır. TL0, T/C-0 kontrol bitleri ile kontrol edilen ve taşıma durumunda TF0 bayrağını setleyen 8-bit'lik zamanlayıcı veya sayıci olarak kullanılabilir. TH0 ise T/C-1 kontrol bitleri ile kontrol edilen ve taşıma durumunda TF1 bayrağını setleyen 8-bit'lik zamanlayıcı olarak kullanılabilir.

### Zamanlayıcı/Sayıci Taşması

Sayma değeri maksimum değere ulaştıktan (8-bit için FFh = 255, 16-bit için FFFFh = 65535) sonra gelen ilk darbe ile “taşma” (Overflow) gerçekleşir ve TCON saklayıcısında bulunan ilgili taşıma bayrağı ( $TF_i$ ,  $i=0,1$ ) donanım tarafından setlenir,  $TF_i=1$ . **Zamanlayıcı/sayıci kesmesi kullanılmıyor ise taşıma bayrağı  $TF_i$  yazılımla temizlenmelidir.** Taşma sonrasında  $TH_i$  ve  $TL_i$  saklayıcılarının değeri tekrar 00h' e döner.

### TMOD Saklayıcısı

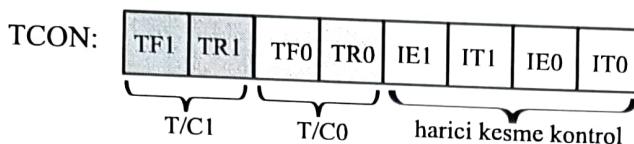
Zamanlayıcı/Sayıcların çalışma ayarlarının yapıldığı kontrol bitlerini içerir. Aşağıda gösterildiği gibi düşük 4-bit'i T/C-0 için yüksek 4-bit'i ise T/C-1 için kullanılan kontrol bitleridir. TMOD SFR'si **bit adreslenemez**.



Bit	İsim	Açıklama
7	$GATE1$	T/C-1 için kapı kontrolü. $GATE1=0$ iken, TR1 =1 yapılması ile T/C-1 aktif olur. $GATE1=1$ yapılması durumunda, T/C-1'in aktif olabilmesi için TR1=1 yapılmalı ve P3.3(INT1) pini lojik '1' seviyesinde olmalıdır.
6	$C/\bar{T}1$	T/C-1 için giriş kaynağını (zamanlayıcı/sayıci ?) seçme biti. $C/\bar{T}1 =0$ Zamanlayıcı, $C/\bar{T}1 =1$ Sayıcı
5	M1	T/C-1 için üstte verilen tabloya uygun olarak çalışma modunu seçme bitleridir.
4	M0	
3	$GATE0$	T/C-0 için kapı kontrolü. T/C-0 için $GATE1$ ile aynı işlevle sahiptir. T/C-0 için harici start/stop işaretini P3.2(INT0) pininden gelmektedir.
2	$C/\bar{T}0$	T/C-0 için giriş kaynağını (zamanlayıcı/sayıci ?) seçme biti. $C/\bar{T}0 =0$ Zamanlayıcı, $C/\bar{T}0 =1$ sayıci
1	M1	T/C-0 için üstte verilen tabloya uygun olarak çalışma modunu seçme bitleridir.
0	M0	

## TCON Saklayıcısı

TCON SFR'sinin yüksek 4-bit'i zamanlayıcı/sayıcılarla, düşük 4-bit'i ise harici kesme kaynakları ile ilgilidir. TCON saklayıcısı **bit adreslenebilir**.



Bit	İsim	Açıklama
7	TF1	T/C-1 taşıma bayrağı. T/C-1 taşıması durumunda donanım tarafından otomatik olarak setlenir. Kesme yapısı kullanılması durumunda program ilgili kesme hizmet altprogramına (Interrupt Service Routine, ISR) dallandığında yine donanım tarafından temizlenir. Kesme yapısı kullanılmadığında ise taşıma sonrasında yazılımla temizlenmelidir.
6	TR1	T/C-1 çalışma biti. TR1 = 0 durumunda T/C-1 pasiftir. TR1 = 1 durumunda ise T/C-1'in aktif olması $\overline{GATE}_1$ bitinin veya P3.3(INT1) pininin durumuna bağlıdır. Yukarıda verilen blok diyagramlarını ve açıklamaları inceleyiniz.
5	TF0	T/C-0 taşıma bayrağı. T/C-0 için TF1 ile aynı işlev sahiptir.
4	TR0	T/C-0 çalışma biti. T/C-0 için TR1 ile aynı işlev sahiptir.
3	IE1	Harici kesme1 oluşturgunda donanım tarafından setlenir. İlgili ISR'den dönüş komutu (RETI) koşturulduğunda donanım tarafından temizlenir.
2	IT1	Harici kesme1 (INT1) için kesme algılama kontrol bitidir. Harici kesme1 girişi P3.3 pinidir. IT1=1 yapıldığında P3.3 pininden gelen işaretin düşen kenarında (1-0 geçişinde) kesme oluşur. IT1=0 yapıldığında P3.3 pininden gelen işaretin Lojik 0 olması durumunda kesme üretilir. Bu durumda giriş işaretin Lojik 0 olduğu sürece sürekli kesme üretilir.
1	IE0	Harici kesme0 oluşturgunda donanım tarafından setlenir. ISR'den dönüş komutu (RETI) koşturulduğunda donanım tarafından temizlenir.
0	IT0	Harici kesme0 (INT0) için kesme algılama kontrol bitidir. İşlevi IT1 ile aynıdır. Harici kesme0 girişi P3.2 pinidir.

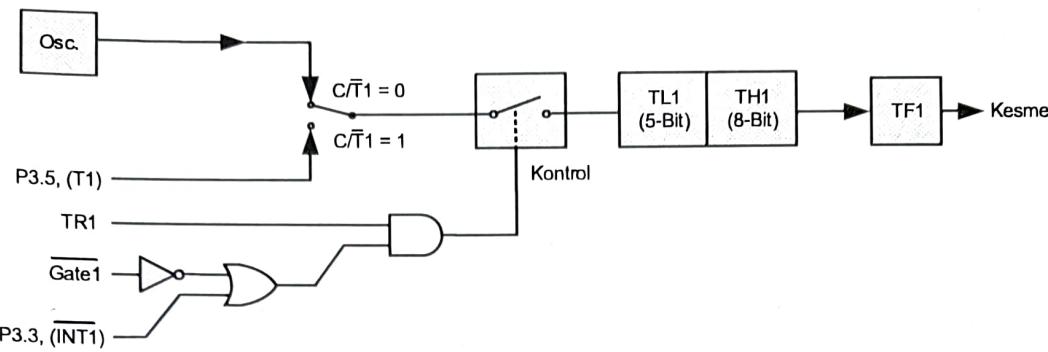
T/C-0 ve T/C-1 4 farklı çalışma modu aşağıda detaylı olarak incelenmiştir. Çalışma modlarına ait şekiller T/C-1 için verilmiştir.

### Zamanlayıcı/Sayıçı Algoritması:

1. Sayma Modu seçilir. TMOD (M0-M1 doğruluk tablosuna göre)
2. Harici veya Dahili (makine çevrimleri) sayma seçimine uygun mod seçilir. (TMOD'daki  $C/\bar{T}$ 'den seçim yapılır.)
3. Sayıcı Star/Stop yazılımsal mı yoksa harici mi yapılacak karar verilip bu doğrultuda  $\overline{GATE}$  ayarı yapılmalıdır.

### 7.3.1 Mod-0: 13-bit zamanlayıcı/sayıçı

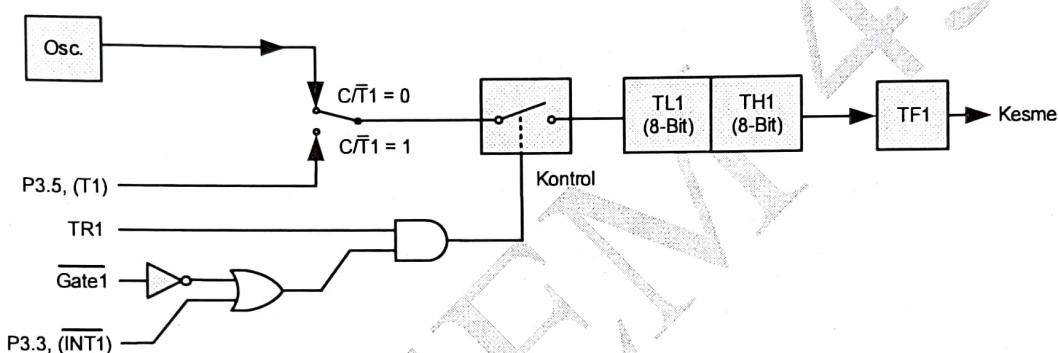
13-bit Zamanlayıcı/sayıçı TLi'nin düşük 5-bit'i ve THi (8-bit)'nin beraber kullanılması ile elde edilir. TLi 'nin sayma aralığı xxx0000b' den xxx1111b (=31d) dir. Dolayısı ile THi nin değeri sayma giriş 8191'e ulaştıktan sonra gelen ilk darbe ile taşıma gerçekleşir, TFi bayrağı setlenir, TLi ve THi saklayıcılarının değeri 0'a düşer.



Şekil 46 T/C1 için Mod-0 blok diyagramı

### 7.3.2 Mod-1: 16-bit zamanlayıcı/sayıci

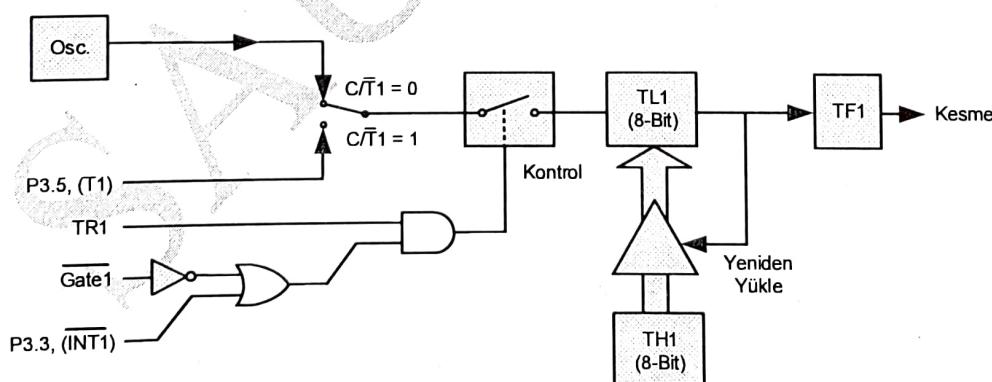
Aşağıda gösterildiği gibi Mod-1'in Mod-0'dan tek farkı 16-bit olmasıdır. Dolayısı ile Mod-1 için maksimum sayma değeri 65535 dir.



Şekil 47. T/C1 için Mod-1 blok diyagramı

### 7.3.4. Mod-2: 8-bit otomatik yüklemeli zamanlayıcı/sayıci

Bu modda TLi anlık sayma değerini tutar. TLi saklayıcısı 255 değerine ulaştıktan sonra sayma giriş kaynağından gelen ilk darbe ile taşar. Taşma bayrağı (TFi) setlenir ve THi'den tutulan değer sayma başlangıç değeri olarak donanım tarafından otomatik olarak TLi'ye aktarılır. TLi yüklenen bu değerden itibaren saymaya devam eder.



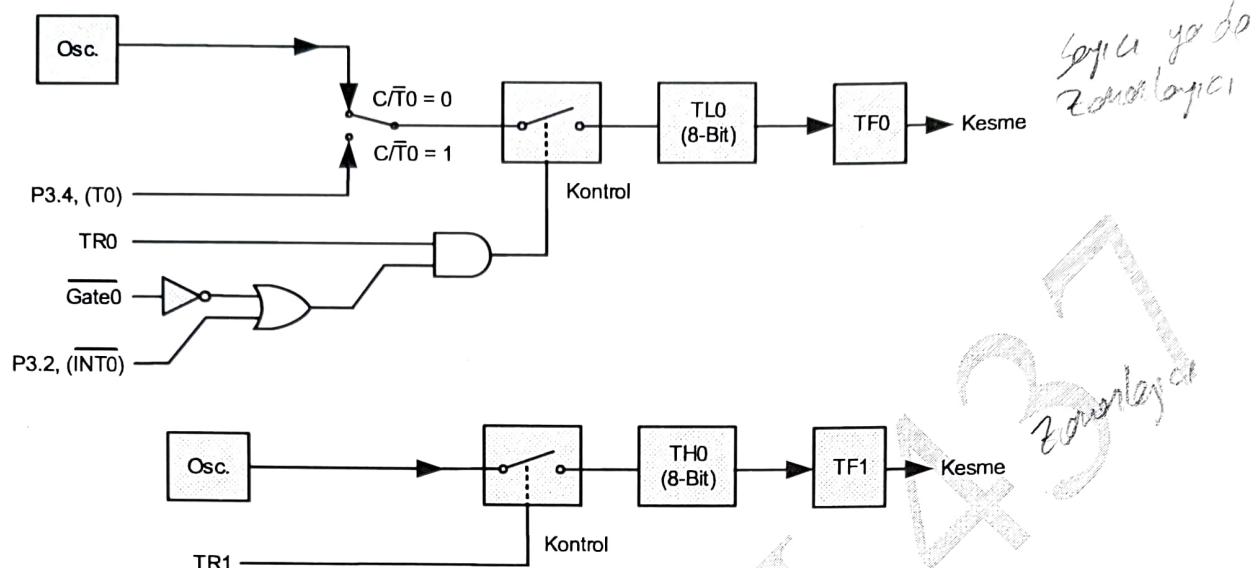
Şekil 48. T/C1 için Mod-2 blok diyagramı

### 7.3.4. Mod-3: Bağımsız çalışma modu

Bu çalışma modunda T/C-1 durdurulur, son sayma değeri TH1 ve TL1 saklayıcılarında tutulmaya devam eder. T/C-0 ise aşağıda gösterildiği gibi 2-adet bağımsız 8-bit'lik zamanlayıcı veya 1-adet 8-bit'lik zamanlayıcı ve 1-adet 8-bit'lik sayıci olarak çalışır.

TL0, T/C-0 kontrol bitleri ile kontrol edilen ve taşıma durumunda TF0 bayrağını setleyen 8-bit'lik

zamanlayıcı veya sayıcı olarak kullanılabilir. TH0 ise çalışma kontrolü TR1 biti ile gerçekleştirilen ve taşıma durumunda TF1 bayrağını setleyen 8-bit'lik zamanlayıcı olarak kullanılabilir.



Şekil 49. T/C Mod-3 blok diyagramı

#### 7.4. Timer 2

**7.4.1 T2CON Saklayıcısı** standart 8051 de 8052 ve ADuC812'de var  
Zamanlayıcı/Sayıçı-2'nin çalışma ayarlarının yapıldığı T2CON SFR'si bit adreslenebilir.  
setb

T2CON: 

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	CNT2	CAP2
-----	------	------	------	-------	-----	------	------

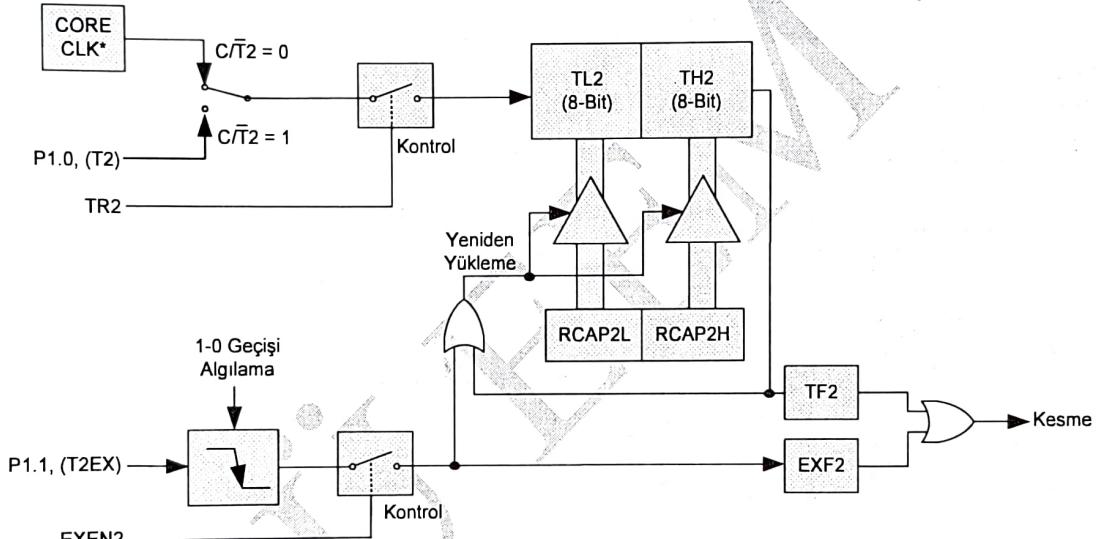
Bit	İsim	Açıklama
7	TF2	Timer 2 taşıma bayrağı; Timer2 taşılığında donanım tarafından setlenir. Yazılım tarafından temizlenir. RCLK=1 veya TCLK=1 olduğu durumlarda setlenmez.
6	EXF2	EXEN2 =1 iken yakalama (Capture) veya yeniden yükleme (Reload) olaylarından biri gerçekleştiğinde T2EX (P1.1)'den gelen sinyalin düşen kenarında (▼) donanım tarafından setlenir. Yazılımla temizlenir.
5	RCLK	RCLK (Receive Clock Enable Bit) Yazılım tarafından setlendiğinde seri portun Mode1 ve Mode3 çalışmalarında seri kanaldan veri almak için gerekli saat darbeleri Timer2'nin taşıma biti tarafından sağlanır. Temizlendiğinde bu işlem için Timer1'in taşıma biti kullanılır.
4	TCLK	TCLK (Transmit Clock Enable) Seri kanaldan veri gönderme için RCLK ile aynı şekilde kullanılır.
3	EXEN2	EXEN2 Timer2 seri kanal için saat üretici olarak kullanılmadığı durumlarda, yazılım tarafından EXEN=1 yapılarak T2EX (P1.1) pininden gelen sinyalin düşen kenarında (▼) yakalama veya yeniden yükleme olayının gerçekleşmesi sağlanır.
2	TR2	TR2 Setlendiğinde Timer2 çalışmaya başlar, temizlendiğinde Timer2 durur.
1	CNT2	CNT2 setlendiğinde T2 pininden gelen harici sinyalleri saymak için <u>Sayıçı</u> olarak ayarlanır. Temizlendiğinde ise <u>Zamanlayıcı</u> olarak kullanılır.
0	CAP2	CAP2 Setlendiğinde (eger aynı anda EXEN2=1 ise) T2EX (P1.1)'den gelen sinyalin düşen kenarında (▼) yakalama işlemi olur.

Zamanlayıcı/Sayıçı-2'nin T2CON saklayıcısı tarafından belirlenen 3 çalışma modu aşağıda verilmiştir.

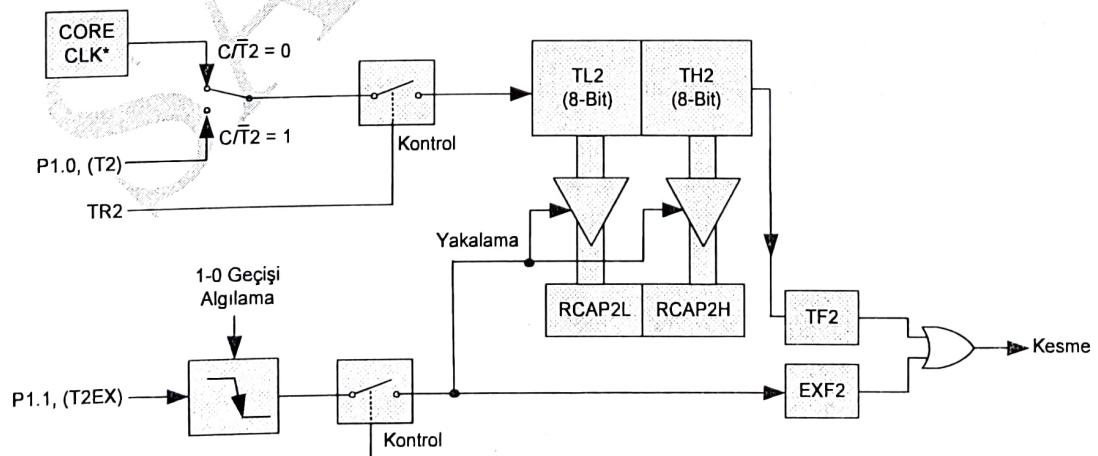
RCLK (veya) TCLK	CAP2	TR2	Çalışma Modu
0	0	1	16-Bit yeniden yükleme <i>Mod-3 16 Bit</i>
0	1	1	16-Bit yakalama
1	X	1	Seri haberleşme için Baud-Rate üreticisi
X	X	0	Timer 2 Çalışmaz

Zamanlayıcı/Sayıcı2'nin kendisine ait 4 adet 8-bitlik veri hafızası (TH2-TL2, RCAP2H-RCAP2L) vardır. TH2 ve TL2, Zamanlayıcı/Sayıcı2'nin çalışma esnasındaki sayma değerini tutar. Dolayısıyla herhangi bir anda Zamanlayıcı/Sayıcı2'nin değeri bu saklayıcılardan okunabilir.

RCAP2H ve RCAP2L saklayıcıları ise otomatik yükleme (autoreload) modunda TH2 ve TL2'ye yüklenen değerleri tutar. Yakalama (capture) modunda ise, yakalama komutu geldiği anda TH2 ve TL2'in değerleri bu saklayıcılara aktarılır. Aşağıdaki her iki çalışma durumuna ait blok diyagram gösterilmiştir.



Şekil 47. Zamanlayıcı/Sayıcı-2 16-Bit “yeniden yükleme modu” blok diyagramı



Şekil 48. Zamanlayıcı/Sayıcı-2 16-Bit “yakalama modu” blok diyagramı

## BÖLÜM 8 KESMELER (INTERRUPTS)

“Kesme”, isminden de anlaşılacağı üzere yürütülmekte olan işlemi kesen (geçici olarak durdurulan) ve derhal cevaplandırılması gereken, iç veya dış kaynağa bağlı olarak yazılımsal/donanımsal bir talep olarak düşünülebilir. Bu talep dış dünyadan (harici) gelebileceği gibi mikrodenetleyici içerisindeki birimlerden de (dahili) gelebilir.

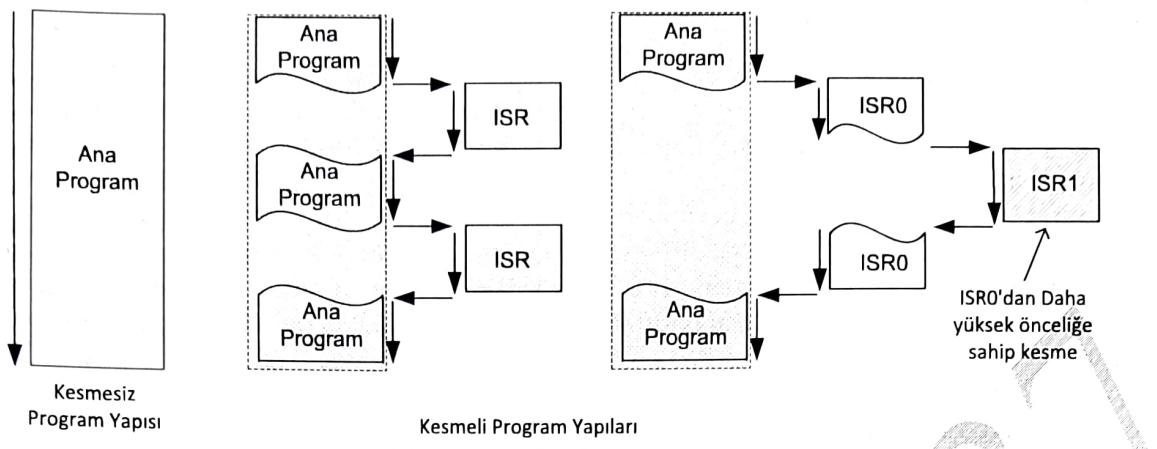
Normal çalışmada mikrodenetleyici herhangi bir durumun (T/C taşması, harici bir işaretin 1-0 değişimi, seri kanaldan veri gönerilip-alınması) gerçekleşip gerçekleşmediğini yoklama (polling) yaparak belirleyebilir. Örneğin TF0 bayrağı koşullu dallanma komutları ile yoklanarak T/C-0’ın taşması kontrol edilebilir. Benzer şekilde RI veya TI bayrakları yoklanarak seri kanaldan veri gönderiminin/alımının tamamlanıp tamamlanmadığı belirlenebilir. Yoklama ile yapılan işlemde mikrodenetleyici herhangi bir anda sadece bir değişkenin değerini kontrol edebilir. Bir değişkenin kontrolü esnasında bir diğer değişkenin değeri değişim能力和 mikrodenetleyici bu durumu tarama sıklığına bağlı olarak belirli bir gecikme ile tespit edebilir.

Kesme yapısı mikrodenetleyicinin *göreceli olarak* aynı anda birden fazla işlemi yapmasını mümkün kılar. Gerçekte işlemci belirli bir anda sadece tek bir komut icra edebilir. Ancak, kesme yapılarının sağlamış olduğu bir özellik olarak, yürütülmekte olan program parçası gelen kesme talebi doğrultusunda geçici olarak durdurulup daha öncelikli bir başka program parçası yürütülür ardından tekrar durdurulan program parçasına geri dönülebilir. Bu durum işlemcinin asenkron olarak birden fazla olayı, durumu gözlemleyememesini, kontrol edebilmesini mümkün kılar.

Besleme gerilimi uygulanıp geçerli bir reset işaretini üretildikten sonra mikrodenetleyicinin program (flash) hafızasında tutulan bir programın normal şartlar altında yürütülmesi, 0000h adresinden başlayarak yukarıdan aşağıya sıra ile tüm komutların koşturulması şeklindedir. Gerekli durumlarda koşullu (cjne, jb, jnb, vb.) veya koşulsuz (sjmp, ljmp, acall) dallanma komutları ile programın akışı yazılım ile yönlendirilir.

Kesme yapılarında ise yürütülmekte olan program oluşan kesme ile birlikte geçici olarak durdurulur, program sayacının (Program Counter, PC) değeri yığına yüklenir ve ilgili kesme vektör adresine dallanılır. Bu noktaya kadar olan tüm işlemler donanım tarafından gerçekleştirilir. Bu noktadan sonra programın akışı gelen kesme talebinin karşılandığı ilgili kesme hizmet programına (Interrupt Service Routine, ISR) yönlendirilir.

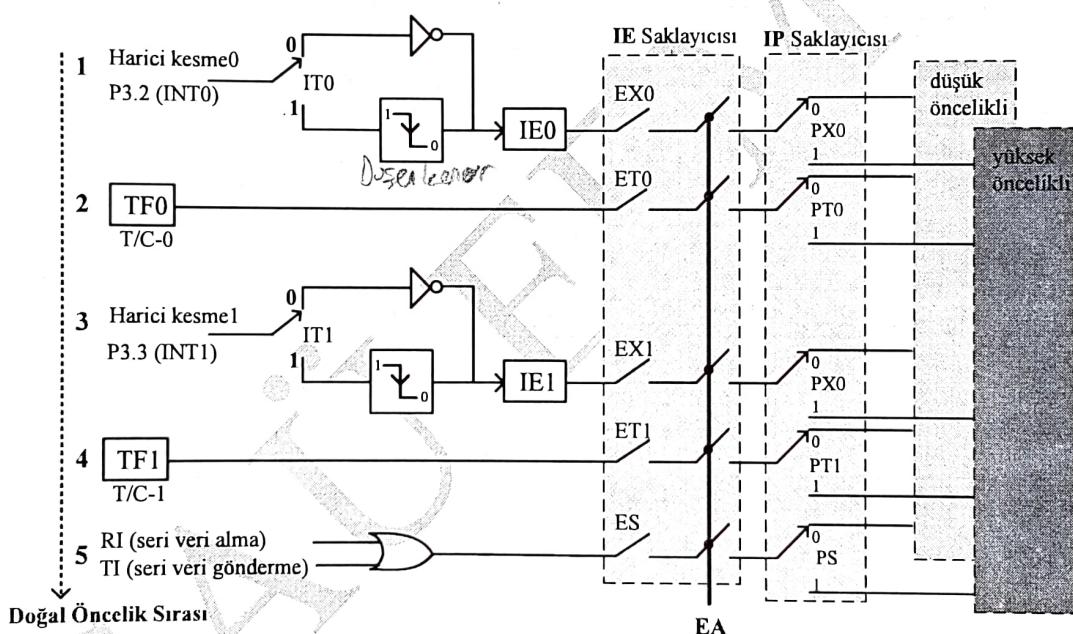
ISR ilgili kesme oluştugunda donanım tarafından çağrılan bir alt program olarak düşünülebilir. Yapılacak olan işlemler tamamlandıında “reti” (Return Interrupt) komutu ile ISR sonlandırılır. “reti” komutu koşturulduğunda program ISR’ den çıkar, dönüş adresi (kesme oluştugunda programın durdurulduğu nokta) yığından okunarak PC’ ye aktarılır ve bu adresten itibaren programın yürütülmesine devam edilir. Aşağıda kesmeli ve kesmesiz örnek programların işleyişi gösterilmiştir.



Şekil 50. Kesmeli ve kesmesiz program yapıları

### 8.1.Kesme Kaynakları ve Öncelik Seviyesi

Standart 8051 mikrodenetleyicisi 2-adet harici kesme (int0 ve int1), 2-adet zamanlayıcı/sayıçısı kesmesi (TF0 ve TF1) ve 1-adet seri haberleşme kesmesi olmak üzere toplam 5-adet kesme kaynağına sahiptir. Aşağıda kesme yapısı için basitleştirilmiş blok diyagramı verilmiştir.



Şekil 51. Kesme ayarları blok diyagramı

Harici kesme 0 (int0) ve harici kesme 1 (int1) sırasıyla P3.2 ve P3.3 pinlerine harici olarak uygulanan işaretler tarafından oluşturulan kesme kaynaklarıdır. Blok diyagramında gösterildiği gibi TCON saklayıcısında bulunan  $IT_i$ ,  $i=0,1$  kontrol bitinin değerine göre harici kesme kaynakları düşen kenar (1-0 geçiş) veya seviye (Lojik 0) tetiklemeli olarak ayarlanır.

Zamanlayıcı/sayıçısı kesme kaynakları T/C-0 için TF0 ve T/C-1 için TF1'dir. Zamanlayıcı/sayıçısı taşılığında  $TF_i$ ,  $i=0,1$  bayrağı donanım tarafından setlenir. Gerekli ayarların yapılması durumunda  $TF_i$  bayrağının setlenmesi kesme üretilmesine neden olur.

Seri veri alma işlemi ve seri veri gönderme işlemi için tanımlanmış ortak tek bir kesme kaynağı vardır. Seri veri alma işlemi tamamlandığında RI bayrağı, seri veri gönderme işlemi tamamlandığında ise TI bayrağı donanım tarafından setlenir. RI veya TI bayrağının setlenmesi seri haberleşme kesmesini

meydana getirir.

Mikrodenetleyici mimarisinde her makine çevriminde kesme talebi olup olmadığı donanım tarafından kontrol edilir. Herhangi bir kesme talebi algılandığında program ilgili ISR'ye dallanarak kesmeye cevap verir. Aynı anda birden fazla kaynaktan kesme talebi gelebilir. Bu durumda ilk olarak hangi kesmeye cevap verileceği öncelik seviyesine bağlıdır. İşlemci yüksek öncelik seviyesine sahip kesmeden başlayarak tüm kesme taleplerine cevap verir. 8051 mimarisinde yazılımla belirlenen 2 öncelik seviyesi vardır: yüksek öncelik ve düşük öncelik seviyesi. Eğer gelen kesmelerin programlanan öncelik seviyesi aynı ise bu durumda doğal öncelik sırasına göre kesmelere cevap verilir. Yukarıda verilen blok diyagramında gösterildiği gibi doğal öncelik sırası en yüksek olan harici kesme 0, en düşük olan ise seri haberleşme kesmesidir.

Programın akışı bir kesme hizmet alt programında iken yeni bir kesme talebi geldiğinde gelen kesmenin öncelik seviyesi yüksek ise işletilmekte olan ISR durdurulur, gelen ISR'ye cevap verilir ve durdurulan ISR'ye geri dönülerek ISR tamamlanır.

Bir ISR devam ederken gelen kesmenin öncelik seviyesi aynı ise bu yeni kesmeye ISR tamamlandıktan sonra cevap verilir. Bir komutun icrası esnasında (bir makine çevriminden daha uzun komutlar) kesme gelmesi durumunda komutun icrası bitirilir daha sonra ISR'ye dallanılır.

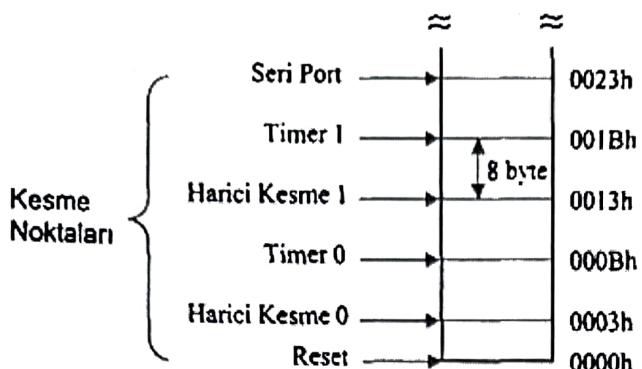
Herhangi bir kesme oluştuğunda program sayacının (Program Counter, PC) değeri yiğina (SP) yüklenir ve programın akışı donanım tarafından ilgili kesme vektör adresine yönlendirilir. Her bir kesme kaynağı için tanımlanmış olan kesme vektör adreslerinin değeri aşağıda belirtilmiştir.

Tablo 6. Kesme vektör adresleri

Kesme Kaynağı	Vektör Adresi	Doğal Sırası	Öncelik
<b>RESET</b>	<b>0000h</b>		
IE0 (Harici kesme0, int0)	0003h	1 (En yüksek)	
TF0 (Zamanlayıcı/Sayıçı-0)	000Bh	2	
IE1 (Harici kesme1, int1)	0013h	3	
TF1 (Zamanlayıcı/Sayıçı-1)	001Bh	4	
RI-TI (Seri Haberleşme)	0023h	5 (En düşük)	

8051 ailesinde donanım sıfırlaması işleminden sonra, işlemcinin program yürütmeye başlangıç adresi 0000h'tır. Şekil 3, Program Hafızanın düşük adresli bölümünün harmasını göstermektedir. Bu bölgede, işlemcinin yürütmekte olduğu komutu bir dış veya iç birimden gelen bir istekle kesip yürüteceği program kodları bulunur.

018 0000  
5JMP basic ayar



Şekil 52. MCS-51 Program Hafıza kesme başlangıç noktaları

Bu şeviden görüldüğü gibi, her kesmeye Program Hafızasında sabit bir bölge atanmıştır. CPU, bir kesme gelmesi sonucu, kesme hizmet programının başlangıç noktası olan bu alanlardan birine, kesme tipine göre dallanır. Örneğin, Harici Kesme 0 (External Interrupt 0) için, 0003h adresi donanım tarafından atanmıştır. Eğer Harici Kesme 0 kullanılmak isteniyorsa, onun kesme hizmet programı 0003h adresinden başlamalıdır. Eğer kesmeler kullanılmayacaksa, bu hafıza bölgesi genel amaçlı Program Hafıza alanı olarak kullanıma açıktır.

Kesme hizmet alanları 8-byte aralıklarla yerleştirilmiştir: 0003h Harici Kesme 0 için, 000Bh Zamanlayıcı 0 için, 0013h Harici Kesme 1 için, 001Bh Zamanlayıcı 1 gibi. Eğer bir ISR kısa ise, bu 8-byte aralıkta yer alabilir. 8-byte'tan uzun ise ve diğer kesmeler de kullanımdaysa, daha sonra gelen kesme alanlarını geçmek için, kesme hizmet programında bir dalandırma komutu (JMP) kullanılmalıdır.

## 8.2.Reset Kesmesi

Mikrodenetleyicinin RST pinine uygulanan gerilim değerinin Lojik 1 seviyesine çıkması durumunda işlemci resetlenmiş olur. Başka bir ifade ile reset kesmesi oluşur. Reset kesmesi oluştuğunda programın akışı 0000h adresine, programın başlangıcına, yönlendirilerek programın başlangıçtan itibaren koşturulması sağlanır. Diğer kesmelerden farklı olarak **Reset kesmesi yazılım ile pasif yapılamaz** ve PC değeri saklanmaz. Normal çalışma durumunda RST pini Lojik 0 seviyesinde tutulur.

Standart 8051'de kesme yetkilendirme (interrupt enable) ve kesme öncelik (interrupt priority) ayarı için 2 adet kontrol SFR'si vardır: IE ve IP. Ayrıca zamanlayıcı/sayıcılar kısmında tanıtılan TCON saklayıcısının düşük 4 biti harici kesmeler (int0 ve int1) ile ilgilidir.

### Kesmeye ait örnek program yapısı:

```

Org 00h
Sjmp basla
org 0003H ; kesme vektör adresi, harici kesme 0
sjmp harici_kesme ; kesme sonucu gidilecek program adı
org 000BH ; kesme vektör adresi, timer0 kesmesi
sjmp timer_kesme ; kesme sonucu gidilecek program adı
basla:
setb ea
setb et0

```

setb ex0

X: jmp x ; sonsuz döngü

harici\_kesme :

.....  
.....  
reti

timer\_kesme :

.....  
.....  
reti  
end

### 8.3.IE Saklayıcısı

Kesme kaynakları için yetkilendirme kontrolü bu SFR üzerinden belirlenir. Başlangıçta tüm kesmeler pasif durumdadır. İlgili kontrol biti setlenerek kullanılacak olan kesme kaynağı aktif hale getirilir. Ancak bu yeterli değildir. Herhangi bir kesmenin kullanılabilmesi için genel yetkilendirme (Enable All, EA) kontrol bitide setlenmelidir. Kesme kaynaklarını yetkilendirme bitlerini içeren IE SFR'si bit adreslenebilir.

IE: 

EA	-	-	ES	ET1	EX1	ET0	EX0
----	---	---	----	-----	-----	-----	-----

Bit	İsim	Açıklama
7	EA	EA=1 tüm kesmeler aktif, EA=0 ise tüm kesmeler pasif.
6	-	Rezerve
5	-	Rezerve
4	ES	ES = 1/0 seri veri alma/gönderme kesmesi aktif/pasif.
3	ET1	ET1 = 1/0 zamanlayıcı/sayıci-1 kesmesi aktif/pasif.
2	EX1	EX1 = 1/0 harici kesme 1 aktif/pasif.
1	ET0	ET0 = 1/0 zamanlayıcı/sayıci-1 kesmesi aktif/pasif.
0	EX0	EX0 = 1/0 harici kesme 0 aktif/pasif.

### 8.4.IP Saklayıcısı

Kesme kaynakları doğal öncelik dışında yazılım ile ayarlanabilen, yüksek ve düşük öncelik olarak adlandırılan 2 öncelik seviyesine sahiptirler. Her bir kesme kaynağı için öncelik seviyesi IP saklayıcısı üzerinden belirlenir. Başlangıçta tüm kesmeler düşük öncelik seviyesindedir. İlgili kontrol bitinin setlenmesi ile istenen kesme kaynağına yüksek öncelik seviyesi kazandırılır. IP SFR'si bit adreslenebilir.

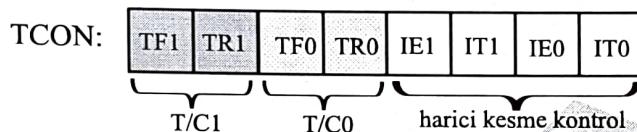
IP: 

-	-	-	PS	PT1	PX1	PT0	PX0
---	---	---	----	-----	-----	-----	-----

Bit	İsim	Açıklama
7	-	Rezerve
6	-	Rezerve
5	-	Rezerve
4	PS	PS = 1/0 seri veri alma/gönderme yüksek/düşük öncelik seviyeli.
3	PT1	PT1 = 1/0 zamanlayıcı/sayıçı-1 yüksek/düşük öncelik seviyeli.
2	PX1	PX1 = 1/0 harici kesme1 yüksek/düşük öncelik seviyeli.
1	PT0	PT0 = 1/0 zamanlayıcı/sayıçı-1 kesmesi yüksek/düşük öncelik seviyeli.
0	PX0	PX = 1/0 harici kesme0 yüksek/düşük öncelik seviyeli.

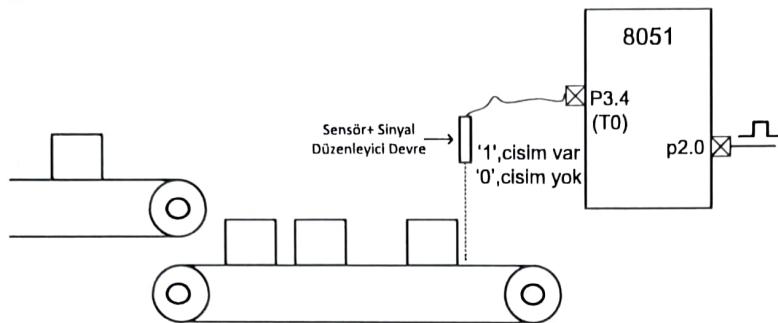
### 8.5.TCON Saklayıcısı

TCON saklayıcısının yüksek 4-bit'i zamanlayıcı/sayıclarla, düşük 4-bit'i ise harici kesme kaynakları ile ilgilidir. TCON SFR'si **bit adreslenebilir**.



Bit	İsim	Açıklama
7	TF1	Açıklamalar için Syf. 112'ye bakınız.
6	TR1	
5	TF0	
4	TR0	
3	IE1	Harici kesme1 oluştduğunda donanım tarafından setlenir. İlgili ISR'den dönüş komutu (RETI) koşturulduğunda donanım tarafından temizlenir.
2	IT1	Harici kesme1 (INT1) için kesme algılama kontrol bitidir. IT1=1 yapıldığında P3.3 pininden gelen işaretin düşen kenarında (1-0 geçişinde) kesme oluşur. IT1=0 yapıldığında P3.3 pininden gelen işaretin Lojik 0 olması durumunda kesme üretilir. Bu durumda giriş işaretinin Lojik 0 olduğu sürece sürekli kesme üretilir.
1	IE0	Harici kesme0 oluştduğunda donanım tarafından setlenir. ISR'den dönüş komutu (RETI) koşturulduğunda donanım tarafından temizlenir.
0	IT0	Harici kesme0 (INT0) için kesme algılama kontrol bitidir. İşlevi IT1 ile aynıdır.

**Örnek:**



Yukarıda verilen sistemde konveyör hattan her 1000 ürün geçişinde p2.0'dan pozitif darbe üretelecektir. Gerekli programı kesme yapısı kullanılarak yazınız.

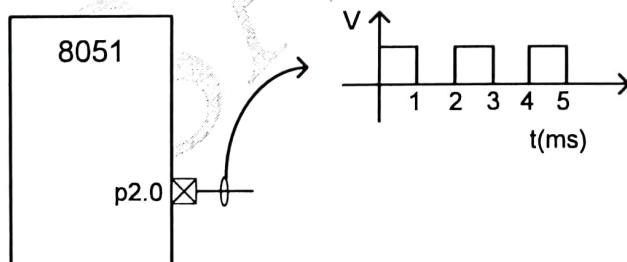
```

org      00h
sjmp    basla
org      0Bh
sjmp    sayici0
basla: mov   tmod, #00000101b ; 16bit, sayıcı0 ayarı.
        mov   dptr,#64535d ;65535-1000
        mov   th0,dph
        mov   tl0,dpl
        setb  et0           ;Z/S-0 kesmesi aktif
        setb  ea            ;seçili kesmeler aktif
        setb  tr0           ;Z/S-0 start
        clr   p2.0
x:     sjmp  x           ;ana program bitti

sayici0: setb  p2.0
        clr   p2.0
        mov   tl0,dpl
        mov   th0,dph
        reti
end.

```

**Örnek:**



P2.0 pininden şekilde verildiği gibi bir kare dalga üretilmek istenmektedir. Gerekli yazılımı kesme yapısını kullanarak gerçekleştiriniz.

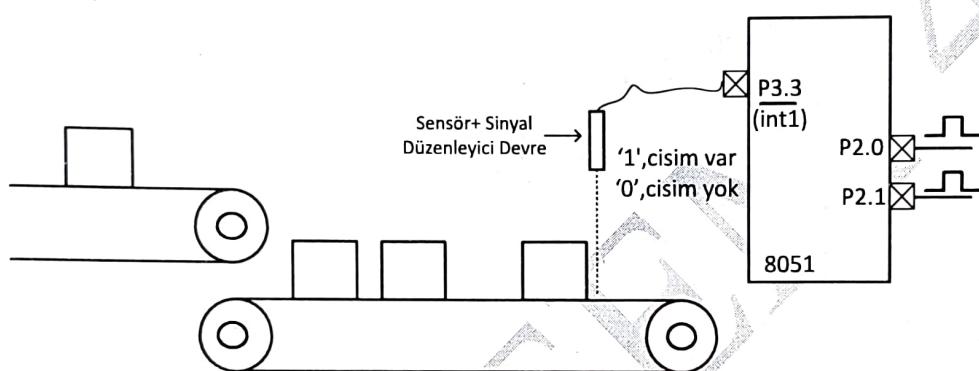
```
org      00h
```

```

sjmp basla
org 0Bh
sjmp timer0
basla: tmov, #00000001b ; 16bit, zamanlayıcı0 ayarı.
        mov dptr,#64535d ;65535-1000
        mov th0,dph
        mov tl0,dpl
        setb et0           ; Z/S-0 kesmesi aktif
        setb ea            ; seçili kesmeler aktif
        setb tr0           ; Z/S-0 start
x:    sjmp x           ;ana program bitti

timer0:cpl p2.0
        mov tl0,dpl
        mov th0,dph
        reti
;
```

**Örnek:**



Bant hızı sabit olan bir konveyör sisteminde geçen kutular uzunluklarına (geçiş süresi) göre tasniflenmektedir. Kutu boyu 50ms ve altı A sınıfı (p2.0 üzerinden 1 adet darbe üretecektir); 50ms üzeri ise B sınıfıdır (p2.1 üzerinden 1 adet darbe üretecektir). Gerekli assembly programını kesme ISR kullanarak gerçekleştiriniz.

**1. Yol:**

```

org 00h
sjmp basla
org 13h
sjmp kesme1
basla:mov tmov,#10010000b      ;harici start/stop timer1,
                                ;16bit timer ayarı
        mov tl1,#00h
        mov th1,#00h
        setb ex1           ;harici kesme1 aktif
        setb it1            ;harici kesme1 düşen kenar tetiklemeli
        setb ea             ;seçili kesmeler aktif
        clr p2.0
        clr p2.1
        setb tr1
        mov dptr,#50000d ;50ms değeri
        mov 40h,dpl
        mov 41h,dph
;
```

```

x:      sjmp  x           ;ana prog. görevi bitti
kesme1: mov   42h, tl1
        mov   a,th1
        mov   tl1,#00h
        mov   th1,#00h
        cjne a,41h,x1    ;cjne komutunun elde bayrağına olan
        mov   a,42h ;etkisini inceleyiniz..
        cjne a,42h,x1
uygun:  setb  p2.0
        clr   p2.0
        reti
x1:     jc    uygun
hata:  setb  p2.1
        clr   p2.1
        reti
end.

```

## 2. Yol:

```

org  00h
sjmp basla
org  0Bh
sjmp timer1
org  13h
sjmp kesme1
basla: mov  tmod,#10010000b ;harici start/stop timer1,
        ;16bit timer ayarı
        mov  dptr,#15535d ;65535d-50000d
        mov  tl1,dpl
        mov  th1,dph
        setb ex1          ;harici kesme1 aktif
        setb it1          ;harici kesme1 düşen kenar tetiklemeli
        setb et1          ;z/s-1 kesmesi aktif
        setb ea           ;seçili kesmeler aktif
        setb pt1          ;z/s-1 kesmesi yüksek öncelikli
        clr   p2.0
        clr   p2.1
        setb tr1
        clr   00h          ;bit adreslenebilen bölge 00h biti temizlendix: sjmp x
;ana prog. bitti
kesme1: mov  tl1,dpl
        mov  th1,dph
        jnb  00h,uygun
        clr   00h
        setb p2.1
        clr   p2.1
        reti
uygun:  setb p2.0
        clr   p2.0
        reti
timer1: setb 00h ;z/s-1 kesmesi oluştu.. kutu>50ms
        reti
end.

```

## 8.6. ADUC841 Kesme Yapısı

Standart 8051 kesme yapısına ilave olarak 6 adet kesme kaynağına sahiptir. Aduc841 için kesme vektör adresleri ve doğal öncelik sırası aşağıda verilmiştir.

Tablo 7. Aduc841 mikrodenetleyicisi için kesme kaynakları ve öncelik sırası

Kesme Kaynağı	Doğal Öncelik	Açıklama
PSMI	1 (En yüksek)	Besleme kaynağı izleme kesmesi
WDS	2	Watchdog Zamanlayıcı kesmesi
IE0	3	Harici kesme 0
ADCI	4	ADC kesmesi
TF0	5	T/C 0 kesmesi
IE1	6	Harici kesme 1
TF1	7	T/C 1 kesmesi
ISPI/I2CI	8	SPI/I2C kesmesi
RI + TI	9	Seri haberleşme kesmesi
TF2 + EXF2	10	T/C 2 kesmesi
TII	11 (En düşük)	TIC Sayıcı kesmesi

Tablo 3. Aduc841 mikrodenetleyicisi için kesme vektör adresleri

Kesme Kaynağı	Vektör Adresi
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH
ADCI	0033H
ISPI/I2CI	003BH
PSMI	0043H
TII	0053H
WDS	005BH

Aduc841 mikrodenetleyicisinde kesme yetkilendirme ve kesme öncelik ayarı için IE ve IP'ye ilave olarak IEIP2 SFR'si bulunmaktadır. Ayrıca IE ve IP saklayıcılarının standart 8051'de boş bırakılan bitleri Aduc841'in ilave kesme kaynakları için aşağıdaki gibi kullanılmıştır.

IE: 

EA	EADC	ET2	ES	ET1	EX1	ET0	EX0
----	------	-----	----	-----	-----	-----	-----

Bit	İsim	Açıklama
7	EA	EA=1 tüm kesmeler aktif, EA=0 ise tüm kesmeler pasif.
6	EADC	EADC = 1/0 ADC kesmesi aktif/pasif.
5	ET2	ET2 = 1/0 zamanlayıcı/sayıçı-1 kesmesi aktif/pasif.
4	ES	ES = 1/0 seri veri alma/gönderme kesmesi aktif/pasif.
3	ET1	ET1 = 1/0 zamanlayıcı/sayıçı-1 kesmesi aktif/pasif.
2	EX1	EX1 = 1/0 harici kesme1 aktif/pasif.
1	ET0	ET0 = 1/0 zamanlayıcı/sayıçı-1 kesmesi aktif/pasif.
0	EX0	EX = 1/0 harici kesme 0 aktif/pasif.

IP: 

-	PADC	PT2	PS	PT1	PX1	PT0	PX0
---	------	-----	----	-----	-----	-----	-----

Bit	İsim	Açıklama
7	-	Rezerve
6	PADC	PADC = 1/0 ADC kesmesi yüksek/düşük öncelik seviyeli.
5	PT2	PT2 = 1/0 zamanlayıcı/sayıçı-2 yüksek/düşük öncelik seviyeli.
4	PS	PS = 1/0 seri veri alma/gönderme yüksek/düşük öncelik seviyeli.
3	PT1	PT1 = 1/0 zamanlayıcı/sayıçı-1 yüksek/düşük öncelik seviyeli.
2	PX1	PX1 = 1/0 harici kesme1 yüksek/düşük öncelik seviyeli.
1	PT0	PT0 = 1/0 zamanlayıcı/sayıçı-1 kesmesi yüksek/düşük öncelik seviyeli.
0	PX0	PX = 1/0 harici kesme 0 yüksek/düşük öncelik seviyeli.

### 8.6.1. IEIP2 Saklayıcısı

Aduc841'in ilave kesme kaynaklarının yetkilendirme ve öncelik ayarlarının yapıldığı IEIP2 SFR'si bit adreslenemez.

IEIP2: 

-	PTI	PPSM	PSI	-	ETI	EPSMI	ESI
---	-----	------	-----	---	-----	-------	-----

Bit	İsim	Açıklama
7	-	Rezerve
6	PTI	PTI = 1/0 zaman aralığı kesmesi yüksek/düşük öncelik seviyeli.
5	PPSM	PPSM = 1/0 Besleme kaynağı izleme kesmesi yüksek/düşük öncelik seviyeli.
4	PSI	PSI = 1/0 SPI/I2C kesmesi yüksek/düşük öncelik seviyeli.
3	-	Bu bit daima sıfır olarak bırakılmalıdır.
2	ETI	ETI = 1/0 zaman aralığı kesmesi aktif/pasif.
1	EPSMI	EPSMI = 1/0 Besleme kaynağı izleme kesmesi aktif/pasif.
0	ESI	ESI = 1/0 SPI/I2C kesmesi 0 aktif/pasif.

## **Kesme'nin işleme konulması:**

Kesme bayrakları her makine çevriminde örneklenmektedir. Bir sonraki makine çevriminde geçerli kesme istekleri arasından seçenek belirlenerek donanım tarafından LCALL oluşturulur. Gelen bir kesme isteğinin karşılanması ancak şu olası durumlar bloke edebilir:

- Eşdeğer yada yüksek öncelikli bir kesmenin işlemde olması.
- Seçmenin gerçekleştiği makine çevriminde o sırada işlenmekte olan komutun sona ermiyor olması.
- İşlenmekte olan komutun kesmeden geri dönme (reti) yada IE, IP kaydedicilerine yazan bir komut olması.

İlk koşul sürdürdüce yeni gelen kesme işleme konamaz. İkincisinde işlenen komutun bitmesi beklenir. Sonuncu koşuldaki reti yada IE, IP'ye erişen komutlardan sonra en az bir komut daha işlenmelidir. Eğer herhangi bir nedenle bir kesme bloke edilmişse koşullar gerçekleşip sırası gelinceye kadar ilgili kesme bayrağı etkin durumda bekletilmelidir. Eğer yanıtlanamayan bir kesme isteği bloke edilmesi sona erdiğinde tekin durumda değilse işleme konmaz. Çünkü her makine çevriminde yer alan seçme işlemi yeniden yapılmaktadır ve işleme konmayan bir kesme bayrağı hatırlanamaz.

Bir kesme işleme konduğunda Program Sayıcı'nın içeriği yiğita yerleştirilmekte ve uygun kesme hizmet çevrimine dallanılmaktadır. Donanımla gerçekleşen bu işlemler sırasında durum belirtici PSW yiğita depolanmaz. Bazı durumlarda kesme hizmetinin başlaması ile ilgili kesme bayrağı sıfırlanır. Ama seri G/C kapısı yada zamanlayıcı-2'nin bayrakları yazılımla sıfırlanmalıdır.

Kesme altprogramı reti ile son buluncaya kadar sürmektedir. Rastlanan bir reti ile eski program sayıcının iki byte'ı yiğittan alınarak yeniden yerine yüklenir. Dikkat edilmesi gereken nokta kesmeden dönüldüğünde kesme düzeneğindeki bilgilerin (bayraklar) henüz kesmenin süregü biçimde kaldığıdır. Bunlar yazılımla sıfırlanmalıdır. Eğer bu yapılmazsa kesme düzeneği kesmenin sona erdiğini fark edememektedir.

Dış kesmelerden birisinin gerçekleşebilmesi için bazı koşulların yerine getirilmesi gereklidir. Dış kesmenin düşen kenar/alçak seviye tiplerinden hangisine duyarlı olacağı TCON kaydedicisindeki IT0, IT1 bitleriyle belirlenmektedir. IT0=1 için kesme-0 düşen kenarla tetiklenmektedir. Bu durumda kesme-0 girişinin ardında gelen iki makine çevrimindeki örneklemeler sırasında önce "1" sonra "0" olması gereklidir. Kesme bayrakları her makine çevriminde yalnız bir kez örneklenmektedir. Kesme-0 girişinin en az bir makine çevrimi "1" olarak kaldıkten sonra en az bir makine çevrimi boyunca "0" olması ile kesme-0 bayrağı etkin duruma geçmesi garantiye alınmış olur. IE0 ilgili kesme altprogramı işleme girince donanım tarafından sıfırlanacaktır. IT1=0 için ise kesme-1 girişi alçak seviyeye getirildiğinde IE1=0 olacaktır. Kesme işleme konup ilgili kesme altprogramına dallanma işlemi gerçekleşene kadar seviye ile tetiklenen kesme girişi "0" olarak tutulmalı, kesme programı sona ermeden önce "1" seviyesine ulaşmalıdır.

Bir kesme isteğinin yanıtlanması için gerekli en kısa süre 3 makine çevrimidir. İlk çevrimde kesme için seçme işlemi gerçekleşmekte, ilgili vektörlerle gösterilen altprograma dallanma işlemi ise 2 makine çevrimi sürmektedir. En uzun süre için ise kesme isteği geldiğinde reti yada IE, IP kaydedicilerinden birisine yazan bir komut işlemiyor olası ve izleyen komutun da bir çarpma yada bölme olması gerekmektedir. Böylece bir kesme isteğin karışılanması için gereken süre 3 makine çevriminden çok, 9 makine çevriminden azdır.

## BÖLÜM 9 ADUC841 ÇEVRE BİRİMLERİ

Mikrodenetleyici mimarilerinde dahili olarak bulunan çevre birimleri endüstriyel uygulamalarda duyulan ihtiyaçlara ve teknolojik gelişmelere paralel olarak sürekli olarak gelişmekte ve mikrodenetleyici mimarilerine yeni çevre birimleri ilave edilmektedir.

İlk olarak Intel firması tarafından 1980 yılında üretilmeye başlayan 8051 ailesi de sürekli olarak gelişmekte ve standart 8051 mimarisinin temel özellikleri korunarak endüstriyel uygulamalardaki ihtiyaçlara cevap verebilecek 8051 ailesinin yeni nesil üyeleri birçok firma tarafından üretilmektedir.

Bu doğrultuda günümüzde mimarisinde,

- Seri Kanal Haberleşmesi
- yüksek hızlı analog giriş (ADC)
- analog çıkış (DAC)
- darbe genişlik modülasyon (Pulse Width Modulation, PWM)
- ilave hafıza (dahili XRAM, flash veri hafıza)
- Uzun ömürlü (100 yıl) ve büyük kapasiteli Flash Program hafıza
- ilave kesme kaynakları
- Çift DPTR
- yüksek hızlı haberleşme birimleri ( $I^2C$ , SPI, CAN)
- kısır-döngü zamanlayıcısı (Watdog Timer)
- besleme gerilimi izleme (Power Supply Monitor, PSM)
- uykuya modu (Power-down)
- devre üzerinde programlanabilme (In System Programming, ISP)

vb. bir çok özelliği barındıran 8051 tabanlı mikrodenetleyiciler çeşitli firmalar tarafından üretilmektedir.

### 9.1. Seri Kanal Haberleşmesi

Modern mikrodenetleyici mimarilerinde en az bir tane evrensel asenkron alıcı/verici birimi (Universal Asynchronous Receiver/Transmitter, UART) bulunmaktadır. Bu sayede mikrodenetleyiciler diğer sayısal cihazlarla (PC, PLC,  $\mu$ C, vb.) standart RS-232 (veya RS-485) seri haberleşmesi yapabilirler. RS-232 bilgisayar dışındaki cihazların bilgisayar ile haberleşmelerinde kullanılan iletişim standartlarından birisidir. RS-232 temel olarak bir seri iletişim tekniğidir.

Seri veri iletişimini; tek-yönlü (simplex), çift-yönlü (duplex) ve eşzamanlı çift-yönlü (full-duplex) olarak gerçekleştirebilir.

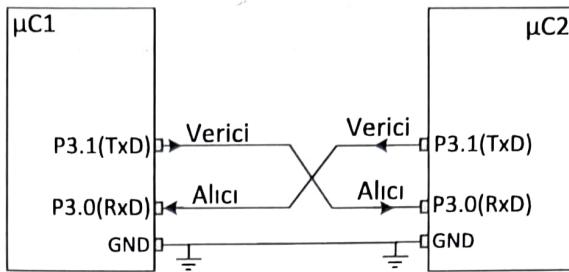


UART birimi üzerinden iki mikrodenetleyici arasında çift-taraflı (duplex) veri iletişiminiabilmesi için gerekli asgari bağlantı şekli aşağıda verilmiştir.

1. Baudrate → 4800  
 Bluetooth → 1200 (mbit)  
 Kablolu → 3600

3600 bps → bit per second

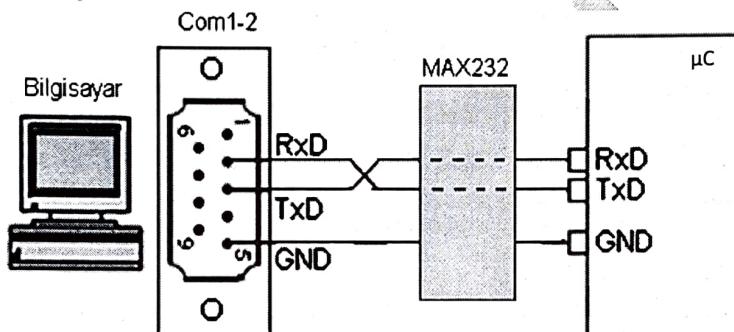
120  
 240  
 480  
 3600  
 115200



Şekil 53. Çift-taraflı UART haberleşmesi için minimum bağlantı yapısı

Gördüğü gibi iki mikrodenetleyicinin seri veri gönderme (TxD= Transmit Data) ve seri veri alma (RxD= Receive Data) uçları birbirine çapraz bağlanır. GND uçları da karşılıklı birbirine bağlanır.

Mikrodenetleyiciler ve bilgisayarlar arasında asenkron seri haberleşmede Lojik seviye farkından dolayı (mikrodenetleyicilerde 0 – 5V, bilgisayarlarda ise ±15V) mikrodenetleyici pinleri doğrudan bilgisayarın ilgili pinlerine (COM port) bağlanmamalıdır. Bilgisayar ile mikrodetleyici arasında seri haberleşme için gerekli Lojik seviye dönüşümünü sağlamak üzere bir gerilim dönüştürücü (örneğin MAX232) kullanılır. Aşağıda bilgisayar ile mikrodetleyici arasında çift-taraflı veri iletişimini yapabilmesi için gerekli asgari bağlantı şekli verilmiştir.

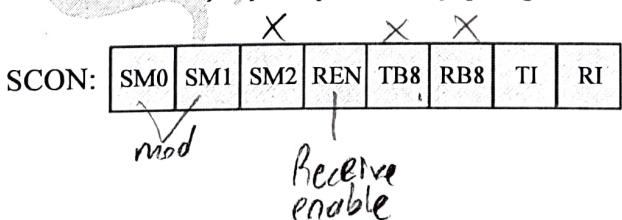


Şekil 54. μC ile bilgisayarın seri haberleşme bağlantı şekli

8051 mimarisinde UART seri portu için ayrılmış 2 adet SFR (SBUF ve SCON) vardır. SCON seri port ayarlarının yapıldığı saklayıcıdır. Seri porta gönderilen veya seri porttan gelen veriler ise SBUF üzerinden işlem görür. SBUF saklayıcısı tek bir SFR adresine sahiptir. Ancak mikrodenetleyici donanımında veri gönderme ve alma işlemi fiziksel olarak iki ayrı SBUF saklayıcısı bulunur. Seri haberleşme için gerekli ayarlar yapıldıktan sonra SBUF'a yazmak seri veri iletişimini başlatır.

### 9.1.1. SCON Saklayıcısı

UART biriminin çalışma ayarlarının yapıldığı SCON SFR'si bit adreslenebilir.



Bit	İsim	Açıklama	
7	SM0	SM1 ve SM0 bitleri aşağıdaki tabloya göre seri haberleşme modunu belirler.	
	SM0	SM1	Çalışma modu
6	SM1	0 0 Mod-0 : Kaymalı Kaydedici, baud-rate sabit: fosc/12 0 1 Mod-1: 8-bit UART, baud-rate TH1 ile ayarlanabilir // 1 0 Mod-2: 9-bit UART, baud-rate sabit: fosc/32 veya fosc/16 1 1 Mod-3: 9-bit UART, baud-rate TH1 ile ayarlanabilir	
5	SM2	Setlendiğinde çok işlemcili haberleşme (Mod-2 ve Mod-3) yetkilendirilmiş olur. Mod-0 kullanılırken SM2=0 yapılmalıdır. Mod-1'de SM2 setlenir ise geçerli bir STOP biti alınana kadar RI biti setlenmez, SM2=0 yapılır ise veri Byte'ı alınır alınmaz RI donanım tarafından setlenir.	
4	REN	Setlendiğinde seri kanaldan veri alma işlemi yetkilendirilmiş olur. Temizlendiğinde seri veri alma işlemi durur.	
3	TB8	2. ve 3. Modda (çok işlemcili haberleşme) iletilen verinin 9. Bitidir.	
2	RB8	2. ve 3. Modda (çok işlemcili haberleşme) alınan verinin 9. Bitidir. Mod-1'de stop biti RB8'e atanır.	
1	TI	Seri veri gönderme işlemi tamamlandığında donanım tarafından setlenir. Yeni veri göndermeden önce yazılımla temizlenmesi gereklidir.	
0	RI	Seri veri alma işlemi tamamlandığında donanım tarafından setlenir. SBUFtan veri okunduktan sonra yeni veri alma işlemi için yazılımla temizlenmesi gereklidir.	

Seri haberleşmede veri iletişim hızı (BaudRate, bit/s) ADUC841 mikrokontrolör için aşağıdaki gibi belirlenir,

$$BaudRate = \frac{2^{SMOD}}{32} \times \frac{f_{osc}}{[256 - TH1]} \text{ bps} \times \frac{1}{12} \rightarrow \text{Standart } 8051$$

bps = bit per second = 1 saniyede iletilen bit sayısı

şeklindedir. Formüllerde "SMOD" PCON SFR sinin 7. bitidir. TH1 ise T/C1'in yüksek veri byte'ıdır.

Örneğin ADUC841 için Baudrate=9600bps olarak ayarlanmak istenir ise: (SMOD=0 ve fosc=11.0592MHz olmak üzere) TH1'e yüklenmesi gereken değer aşağıdaki gibi hesaplanır.

$$9600 = \frac{2^0}{32} \times \frac{11.0592 \times 10^6}{[256 - TH1]} \rightarrow 256 - TH1 = 36 \rightarrow TH1 = 220'd = DC'h$$

İki cihazın seri kanal üzerinden doğru bir şekilde haberleşebilmeleri için her iki taraftada yapılan ayarlamalar (Baud Rate, Haberleşme modu vs.) aynı olmalıdır.

## 9.2. CFG841 Saklayıcısı

ADUC841 mimarisinde yer alan çeşitli çevre birimlerinin ayarının yapıldığı CFG841 SFR'si **bit adreslenemez**. Reset sonrası başlangıç değeri 10h'dır.

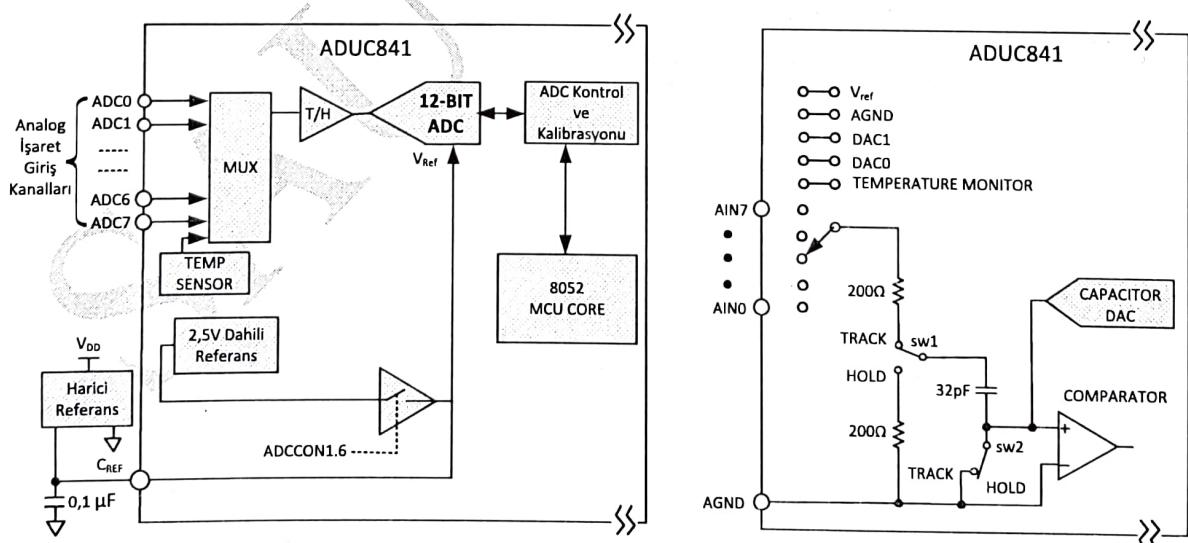
CFG841:	EXSP	PWPO	DBUF	EPM2	EPM1	EPM0	MSPI	XRAMEN
---------	------	------	------	------	------	------	------	--------

Bit	İsim	Açıklama
7	EXSP	Setlendiğinde genişletilmiş SP yapısı (11-bit) aktif olur. Temizlendiğinde ise standart SP yapısı (8-bit) seçilmiş olur.
6	PWPO	Setlendiğinde P3.4 ve P3.3 pinleri PWM çıkışları olarak atanır. Temizlendiğinde ise P2.6 ve P2.7 PWM çıkışları olarak atanır.
5	DBUF	Setlendiğinde dahili DAC çıkış tamponları by-pass edilir. Temizlendiğinde ise DAC çıkış tamponları aktif duruma geçer.
4	EPM2	EPM2, EPM1 ve EPM0 bitleri aşağıdaki tabloya göre PWM saat frekansı için bölmeye faktörünü belirler.
3	EPM1	
2	EPM0	
1	MSPI	Setlendiğinde SPI haberleşme biriminin MISO, MOSI ve SCLOCK fonksiyonlarını sırasıyla P3.3, P3.4 ve P3.5 pinlerine atanır.
0	XRAMEN	Setlendiğinde dahili XRAM kullanıma açılır ve harici veri hafıza adres alanının alt 2-kByte 'lık kısmına yerleştirilir. Temizlendiğinde ise dahili XRAM kullanıma kapalıdır.

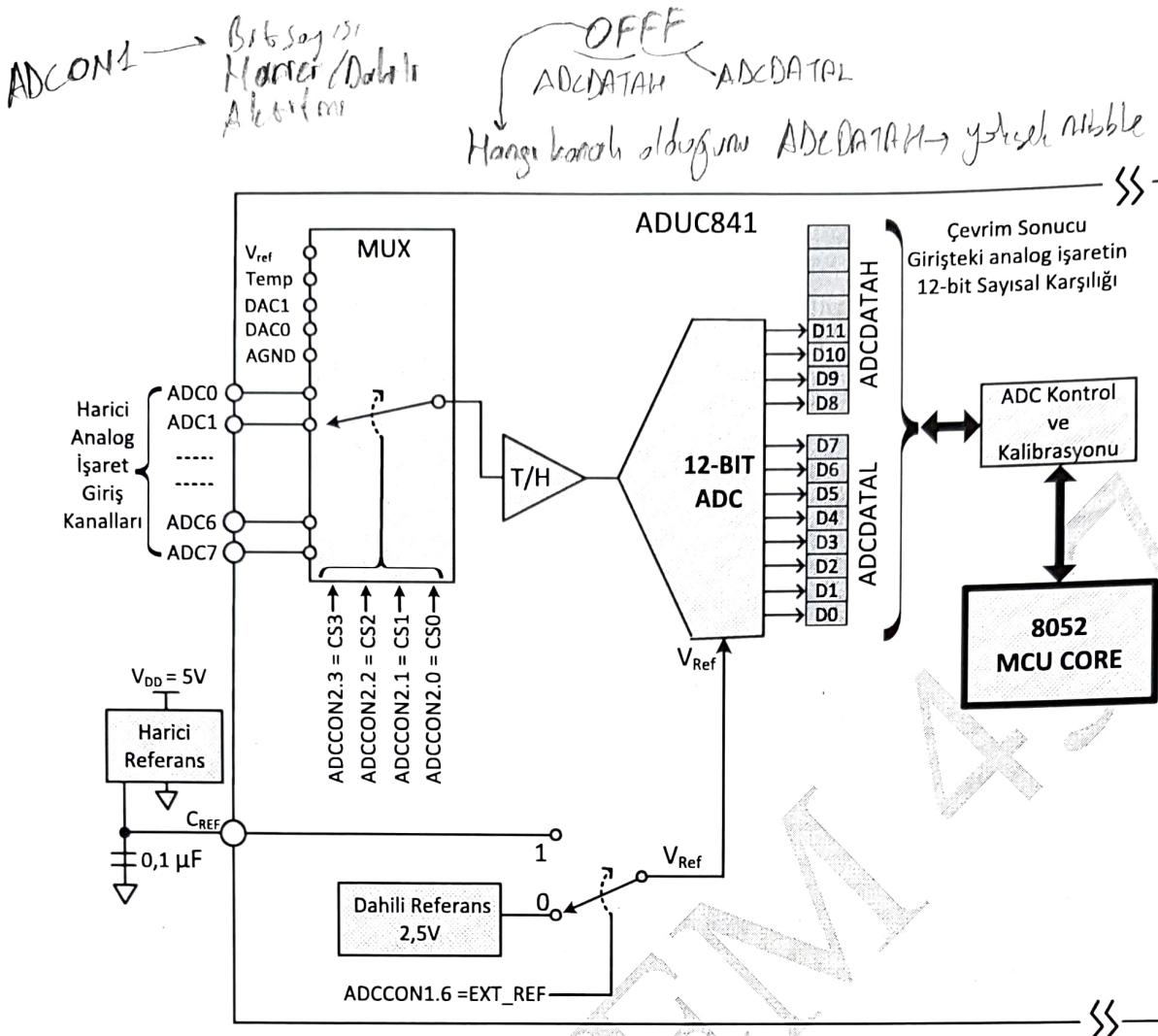
EPM2	EPM1	EPM0	Bölme Faktörü
0	0	0	<b>32</b>
0	0	1	<b>64</b>
0	1	0	<b>128</b>
0	1	1	<b>256</b>
1	0	0	<b>512</b>
1	0	1	<b>1024</b>

### 9.3. Analog-Sayısal Dönüştürücü Birimi (Analog-Digital Converter, ADC)

ADUC841 mimarisinde, 12-bit, 8 analog giriş kanalı, 420 kSPS çevrim hızı (420.000 çevrim/saniye), tek-kaynak besleme, izleyici ve tutucu (TRACK and HOLD, T/H), dahili referans kaynağı ve kalibrasyon edilebilme gibi özelliklere sahip olan ardışıl-yaklaşım tipi (Successive Approximation Register, SAR) bir ADC mevcuttur. ADC 'nin blok diyagramı aşağıda verilmiştir.



Şekil 55. Aduc841 mimarisinde yer alan ADC blok diyagramı



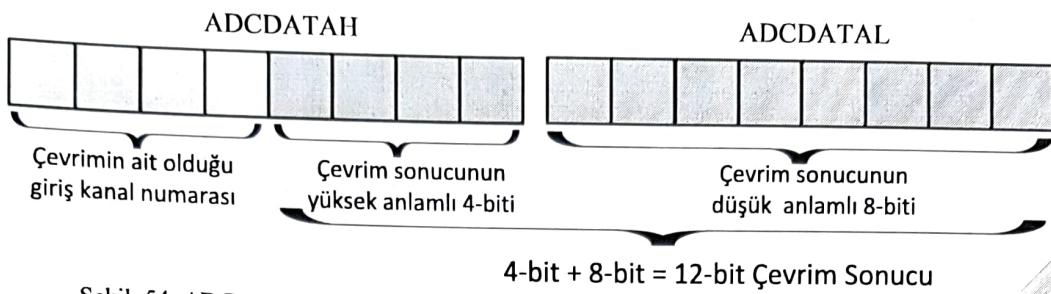
Şekil 56. Aduc841 mimarisinde yer alan ADC ayrıntılı blok diyagramı

ADC yapısında bulunan MUX, T/H ve Referans Gerilim kaynağı bloklarının işlevi aşağıda kısaca açıklanmıştır.

1. **MUX (Çoğullayıcı):** Blok diyagramında gösterildiği gibi ADC'nin birden fazla analog giriş kanalı mevcuttur (ADC0...ADC7, DAC0, DAC1, vd). Herhangi bir anda giriş kanallarından sadece bir tanesi aktif olabilir. Başka bir ifade ile ADC herhangi bir anda giriş kanallarından sadece bir tanesini sayısal olarak dönüştürebilir. MUX, seçme girişleri (ADCCON2 saklayıcısındaki CS0..CS3 bitleri) aracılığı ile belirlenen giriş kanalını T/H'a iletir.
2. **T/H (TRACK and HOLD, İzleyici ve tutucu):** Girişindeki analog işareti ADC'ye çevrime başla sinyali gelene kadar izler (trace). ADC'ye çevrime başla sinyali geldiği anda T/H girişinde o an bulunan analog değeri tutar (hold) ve çevrim bitene kadar bu değeri saklar. Aksi halde, yani T/H kullanılmadığı durumda, ADC çevrimi esnasında girişteki analog işaret değişeceğinden çevrim sonucu hatalı olabilir.
3. **Referans Gerilim kaynağı:** ADC'lerin analog bilgiyi sayısal olarak dönüştürebilmeleri için referans gerilimi, Vref, gereksinimleri vardır. ADUC841 mimarisinde ADC için gerekli olan referans gerilim mimari içerisinde (internal, Vref=2.5V) bulunan kaynaktan sağlanabileceği gibi entegrenin uygun pinine (Cref) bağlanan harici (external) bir kaynaktanda sağlanabilir. Harici referans kaynağı gerilim değeri Vref=1V-AVDD arasında olmalıdır. ADC'nin doğru olarak sayısal olarak dönüştürebileceği giriş gerilim değeri 0V – (Referans Gerilim)'dır. Çevrim öncesinde ADC'nin hangi referans kaynağını kullanacağı ADCCON1.6 =EXT\_REF biti ile belirlenir.

ADC' nin çalışma ayarları (analog giriş kanalının, referans kaynağının seçilmesi, çevrimin başlatılması vb.) ADCCON1, ADCCON2 ve ADCCON3 kontrol SFR'leri kullanılarak yapılır. ADC çevrimi tamamlandığında elde edilen 12-Bit'lik çevrim sonucu ise donanım tarafından otomatik olarak

ADCDATAH ve ADCDATAH veri SFR'lerine yüklenir. ADCDATAH SFR'sinin yüksek değerli 4-Bit'lik kısmı çevrim sonucunun ait olduğu analog giriş kanalını belirtir.



Şekil 54. ADC çevrim sonucunun ADCDATAH/L SFR'lerinde saklanması

### 9.3.1. ADCCON1 Saklayıcısı

Çevrim ve data yakalama sürelerinin ayarlandığı, çevrime başlama modunun seçildiği ve ADC'nin on/off yapıldığı kontrol SFR'sidir. ADCCON1 SFR'si bit adreslenemez.

ADCCON1:	MD1	EXT_REF	CK1	CK0	AQ1	AQ0	T2C	EXC
	0	0	0	0	1	1	1	1

Bit	İsim	Açıklama															
7	MD1	Setlendiğinde ADC enerjilendirilir, kullanıma hazır hale gelir (Power On). Temizlendiğinde ise ADC beslemesi kapatılır (Power Off).															
6	EXT_R EF	Setlendiğinde ADC için <u>harici referans</u> kaynağı seçilir. Bu durumda işlemcinin ilgili pinine (Cref) harici bir referans kaynağı bağlanmalıdır. Temizlendiğinde <u>dahili referans</u> kaynağı seçilir															
5	CK1	ADC'nin çalışma frekansı, $f_{ADC}$ , mikrokontrolöre bağlı olan harici kristal frekansının ( $fosc$ ) CK0 ve CK1 bitleri ile belirlenen katsayıya ( $MCLK$ ) bölünmesi ile ( $f_{ADC} = fosc / MCLK$ ) elde edilir. Bölme katsayısı $400kHz < f_{ADC} < 8.38 MHz$ şartı sağlanacak şekilde aşağıdaki tabloya göre belirlenir.															
4	CK0	<table border="1"> <thead> <tr> <th>CK1</th> <th>CK0</th> <th>MCLK</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>32</td> </tr> <tr> <td>0</td> <td>1</td> <td>4</td> </tr> <tr> <td>1</td> <td>0</td> <td>8</td> </tr> <tr> <td>1</td> <td>1</td> <td>2</td> </tr> </tbody> </table> $f_{ADC} = 6MHz$	CK1	CK0	MCLK	0	0	32	0	1	4	1	0	8	1	1	2
CK1	CK0	MCLK															
0	0	32															
0	1	4															
1	0	8															
1	1	2															
3	AQ1	AQ1 ve AQ0 bitleri kullanılarak izleyici ve tutucunun (T/H) giriş işaretini yakalaması için kaç adet ADC saat darbesi kullanılacağı aşağıdaki tabloya göre belirlenir. Üretici firma 3 veya 4 saat darbesinin kullanılmasını önermektedir.															
2	AQ0	<table border="1"> <thead> <tr> <th>AQ1</th> <th>AQ0</th> <th>ADC Clk Sayısı</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>2</td> </tr> <tr> <td>1</td> <td>0</td> <td>3</td> </tr> <tr> <td>1</td> <td>1</td> <td>4</td> </tr> </tbody> </table> $16 \times \frac{1}{6MHz} + (4 \times \frac{1}{6MHz})$	AQ1	AQ0	ADC Clk Sayısı	0	0	1	0	1	2	1	0	3	1	1	4
AQ1	AQ0	ADC Clk Sayısı															
0	0	1															
0	1	2															
1	0	3															
1	1	4															
1	T2C	Setlendiğinde ADC'nin çevrime başlaması Zamanlayıcı/Sayıçı-2'nin taşıma biti (TF2) ile kontrol edilir. Zamanlayıcı/Sayıçı-2 taşılığında TF2 donanım tarafından otomatik olarak setlenir ve ADC çevrime başlar. Çevrim bittiğinde TF2 yazılımla temizlenmelidir. Daha detaylı bilgi için T2CON saklayıcısını inceleyiniz.															
0	EXC	Setlendiğinde ADC'nin çevrime başlaması P3.5 (CONVST) pininden gelen sinyal ile kontrol edilir. P3.5 pininden gelen sinyalin ADC çevrimini başlatılabilmesi için minimum 100ns Lojik 0 mertebesinde kalmalıdır.															

### 9.3.2. ADCCON2 Saklayıcısı

Analog giriş kanalının ve çevrim modunun seçildiği ADCCON2 SFR'si bit adreslenebilir.

ADCCON2:	ADC1	DMA	CCONV	SCONV	CS3	CS2	CS1	CS0
----------	------	-----	-------	-------	-----	-----	-----	-----

Bit	İsim	Açıklama																																																																											
7	ADCI	ADC kesme bayrağıdır. Çevrim tamamlandığında donanım tarafından otomatik olarak setlenir. Program ADC kesme vektörüne (ORG 0033H) dallandığında yine donanım tarafından temizlenir. Kesme alt programı kullanılmadığı durumlarda ise yazılım tarafından temizlenmelidir.																																																																											
6	DMA	DMA (Direct Memory Access = Doğrudan Hafızaya Erişim) ADC nin bu özelliği deneylerde kullanılmayacağından DMA = 0 yapılacaktır.																																																																											
5	CCON V	CCONV = 1 <u>yapıldığı anda</u> ADC sürekli çevrim modunda çalışmaya başlar. Bir çevrim tamamlandığında otomatik olarak yeni bir çevrim başlar.																																																																											
4	SCO NV	SCONV = 1 <u>yapıldığı anda</u> ADC tek bir çevrim yapar ve durur. Çevrim tamamlandığında donanım tarafından SCONV=0'a çekilir. <u>Yeni bir çevrim için tekrar SCONV = 1 yapılmalıdır.</u>																																																																											
3	CS3	CS0,CS1,CS2,CS3 bitleri ile analog giriş kanalı aşağıdaki tabloya göre belirlenir.																																																																											
2	CS2																																																																												
1	CS1																																																																												
0	CS0	<table border="1"> <thead> <tr> <th>CS 3</th> <th>CS 2</th> <th>CS 1</th> <th>CS 0</th> <th>Analog Giriş Kanalı</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>Adc0</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>Adc1</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>Adc2</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>Adc3</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>Adc4</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>Adc5</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> <td>Adc6</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>Adc7</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>Sıcaklık İzleyicisi</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> <td>DAC0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>DAC1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>1</td> <td>AGND</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>Vref</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>DMA STOP</td> </tr> </tbody> </table>	CS 3	CS 2	CS 1	CS 0	Analog Giriş Kanalı	0	0	0	0	Adc0	0	0	0	1	Adc1	0	0	1	0	Adc2	0	0	1	1	Adc3	0	1	0	0	Adc4	0	1	0	1	Adc5	0	1	1	0	Adc6	0	1	1	1	Adc7	1	0	0	0	Sıcaklık İzleyicisi	1	0	0	1	DAC0	1	0	1	0	DAC1	1	0	1	1	AGND	1	1	0	0	Vref	1	1	1	1	DMA STOP
CS 3	CS 2	CS 1	CS 0	Analog Giriş Kanalı																																																																									
0	0	0	0	Adc0																																																																									
0	0	0	1	Adc1																																																																									
0	0	1	0	Adc2																																																																									
0	0	1	1	Adc3																																																																									
0	1	0	0	Adc4																																																																									
0	1	0	1	Adc5																																																																									
0	1	1	0	Adc6																																																																									
0	1	1	1	Adc7																																																																									
1	0	0	0	Sıcaklık İzleyicisi																																																																									
1	0	0	1	DAC0																																																																									
1	0	1	0	DAC1																																																																									
1	0	1	1	AGND																																																																									
1	1	0	0	Vref																																																																									
1	1	1	1	DMA STOP																																																																									

### 9.3.3. ADCCON3 Saklayıcısı

ADC'nin kalibrasyon ayarlarının yapıldığı ADCCON3 SFR'si bit adreslenemez.

ADCCON3:	BUSY	RSVD	AVGS1	AVGS0	RSVD	RSVD	TYPICAL	SCAL
----------	------	------	-------	-------	------	------	---------	------

Bit	İsim	Açıklama
7	BUSY	BUSY bitinin içeriği okunabilir fakat yazılımla değiştirilemez (Read Only). BUSY biti ADC çevrimi devam ederken veya ADC nin kalibrasyonu yapılırken <u>donanım</u> tarafından setlenir. Çevrim veya kalibrasyon bittiğinde yine <u>donanım</u> tarafından temizlenir.

6	RSVD	Bu bit donanım gereği daima 0 olmalıdır
5	AVGS1	ADC'nin kalibrasyonu ile ilgilidir. Deneylerde kullanılmayacağı için burada açıklanmamıştır
4	AVGS0	ADC'nin kalibrasyonu ile ilgilidir. Deneylerde kullanılmayacağı için burada açıklanmamıştır
3	RSVD	Bu bit donanım gereği daima 0 olmalıdır
2	RSVD	Bu bit kalibrasyon süresince daima 1 olmalıdır
1	TYPICAL	ADC'nin kalibrasyonu ile ilgilidir. Deneylerde kullanılmayacağı için burada açıklanmamıştır
0	SCAL	ADC'nin kalibrasyonu ile ilgilidir. Deneylerde kullanılmayacağı için burada açıklanmamıştır

#### ADC çevriminin başlatılması:

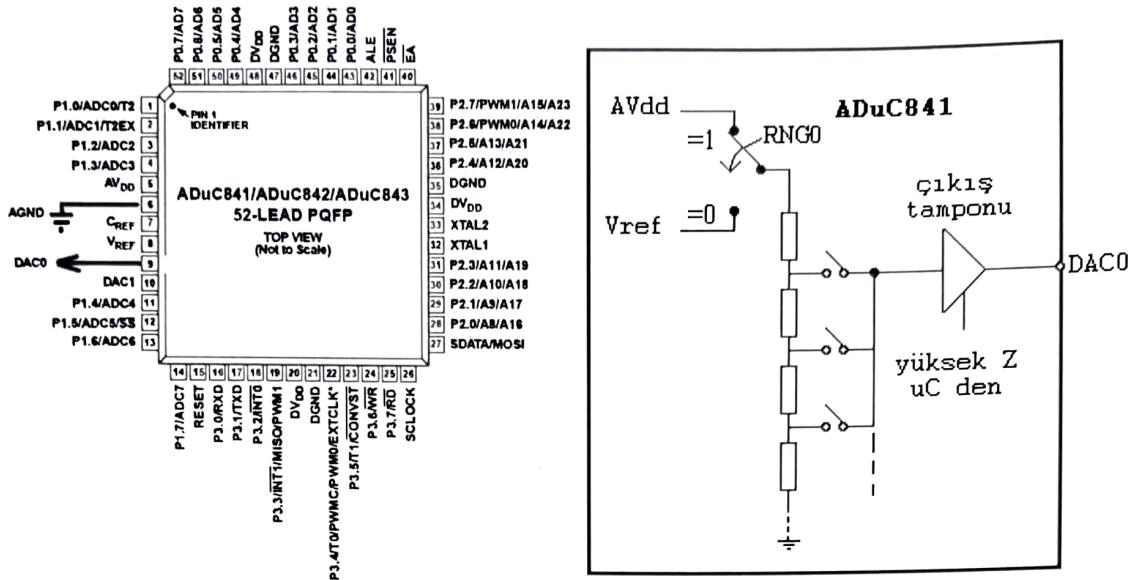
**ADCCON1** ve **ADCCON2** SFR'leri için yukarıda verilen açıklamalar incelediğinde ADUC841 mimarisinde bulunan ADC'nin çevrime başlatılması için 4 farklı yöntemin bulunduğu görülür. Bunlar;

- i) Harici işaret ile çevrim başlatma: ADCCON1.0 = EXC biti setlendiği durumda ADC P3.5 (CONVST) pini aracılığı ile çevrime başlatılabilir.
- ii) Zaman tabanlı çevrim başlatma: ADCCON1.1 = T2C biti setlendiği durumda ADC Zamanlayıcı/Sayıçı 2'nin taşıma bayrağı TF2 aracılığı ile çevrime başlatılabilir.
- iii) Tek çevrim modu: ADCCON2.4 = SCONV setlendiği anda ADC tek bir çevrim yapar ve durur.
- iv) Sürekli çevrim modu: ADCCON2.5 = CCONV setlendiği anda ADC sürekli çevrim modunda çalışmaya başlar.

**Çevrim Süresi:** ADC'nin analog giriş kanalındaki işaretin örneklenip sayısal çıkışlarında hazır hale gelmesi için harcanan zaman ADC'nin çevrim süresini verir. Çevrim süresi ADC'lerin seçiminde dikkat edilmesi gereken önemli kriterlerden biridir. ADUC841 mikrokontrolöründe bulunan 12-bitlik ADC'nin çevrim süresi, 16 ADC saat darbesi + veri yakalama için seçilen ADC saat darbesi (1...4) olarak hesaplanır. (**1 ADC saat darbesi = 1/ f<sub>ADC</sub> dir.**) ADC'nin çalışma frekansı (f<sub>ADC</sub>) ADDCON1 deki CK0-CK1 bitleri ile ayarlanır. Dolayısıyla ADUC841 mimarisinde bulunan ADC'nin çevrim süresi ADC için seçilen çalışma frekansına ve T/H için ayarlanan saat darbesi sayısına bağlı olarak değişir. Yukarıda belirtildiği gibi maksimum çevrim hızı 420.000 çevrim/saniyedir.

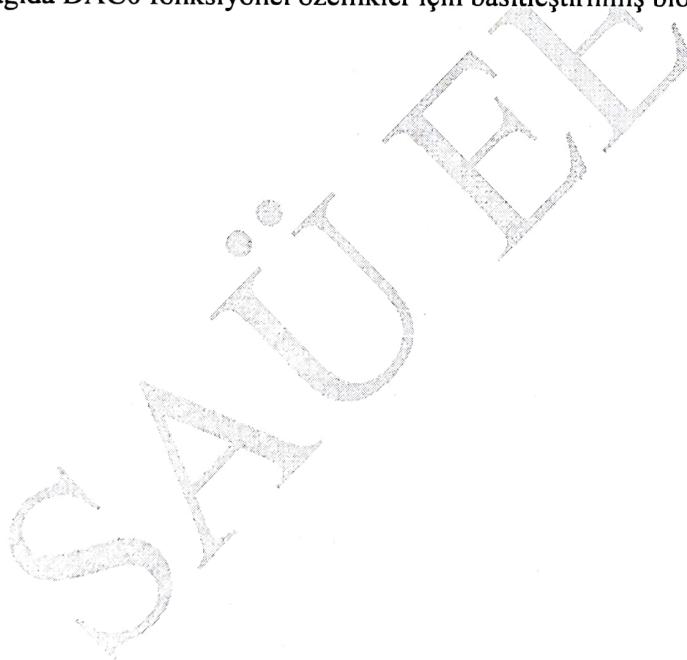
#### 9.4. Sayısal - Analog Dönüşürücü Birimi (Digital - Analog Converter, DAC)

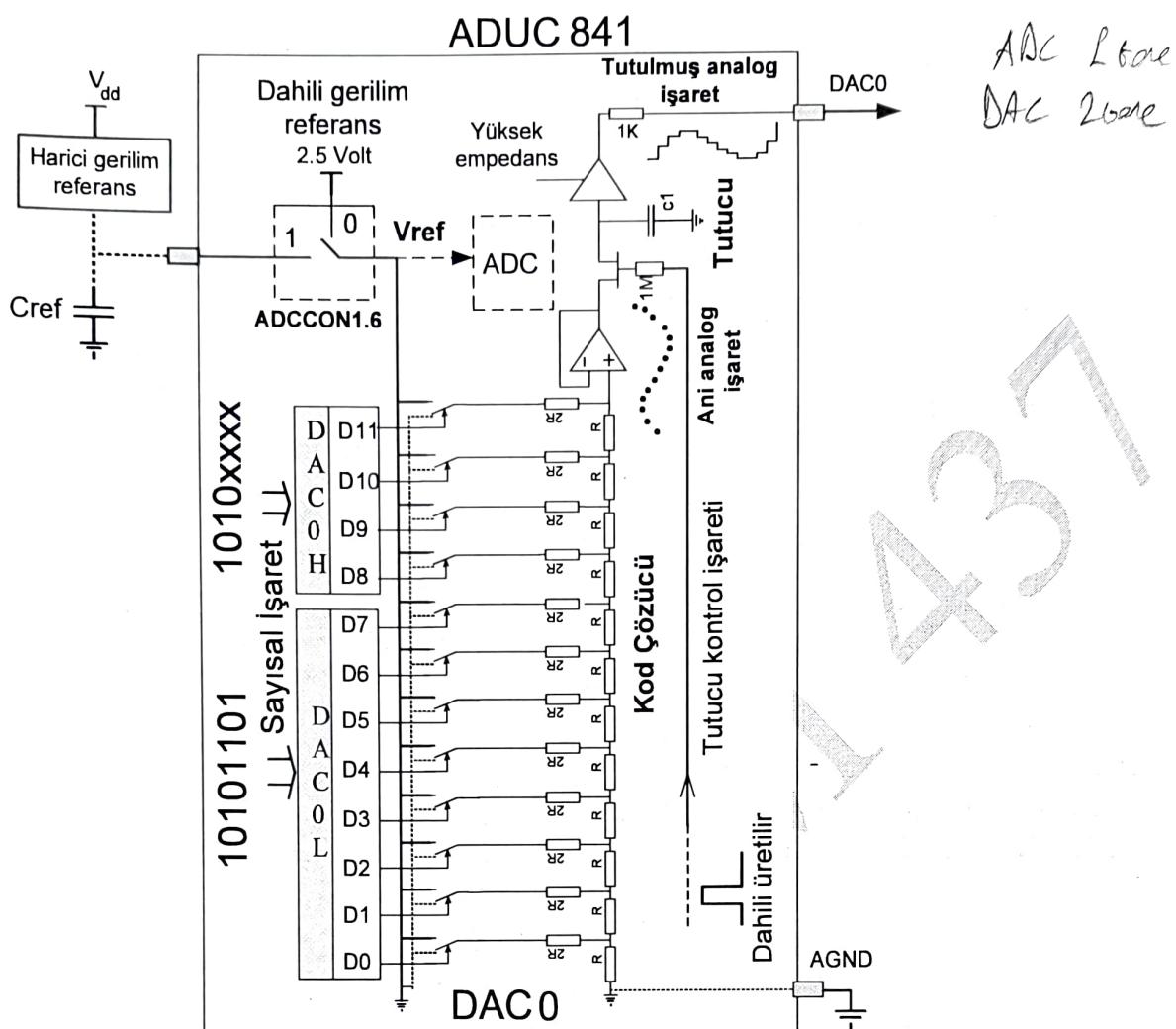
ADUC841 mimarisinde iki adet, 12-Bit gerilim çıkışlı direnç-dizisi (resistor string) türünde DAC (DAC0 ve DAC1) modülü bulunmaktadır. DAC'ların çıkış gerilim aralığı 0V – Vref (=2.5V) ve 0V – AVDD olmak üzere iki farklı alternatif sahiptir. Ancak çıkış gerilim aralığı 0V – Vref seçilmesi durumunda DAC'ların doğru çalışabilmesi için ADC'de mutlaka enerjilendirilmelidir (Power-On). Aşağıda ADUC841 mimarisinde yer alan DAC'ların basitleştirilmiş blok diyagramı verilmiştir.



Şekil 57. ADuC841 mikrokontrolör ve mimarisinde bulunan DAC modülünün basitleştirilmiş yapısı.

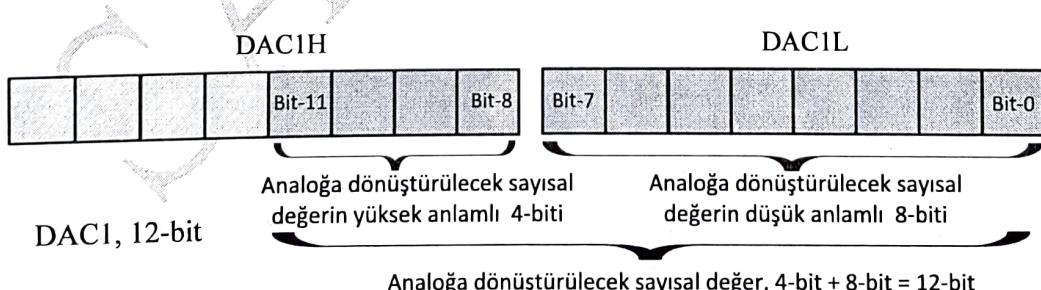
ADUC841 mimarisinde bulunan DAC birimleri, lojik ‘1’ ve ‘0’ lardan oluşan 12-bit uzunluğundaki o andaki sayısal işaretin ani analog işaretine çeviren *kod çözücü* ve bu ani analog işaretin bir sonraki ani analog işaret gelinceye kadar sabit tutan *tutucu* olmak üzere iki temel bloktan oluşan donanımsal bir birimdir. ADUC841 mimarisinde bulunan DAC0 ve DAC1 donanımsal olarak özdeşdir. Aşağıda DAC0 fonksiyonel özellikler için basitleştirilmiş blok olarak verilmiştir.

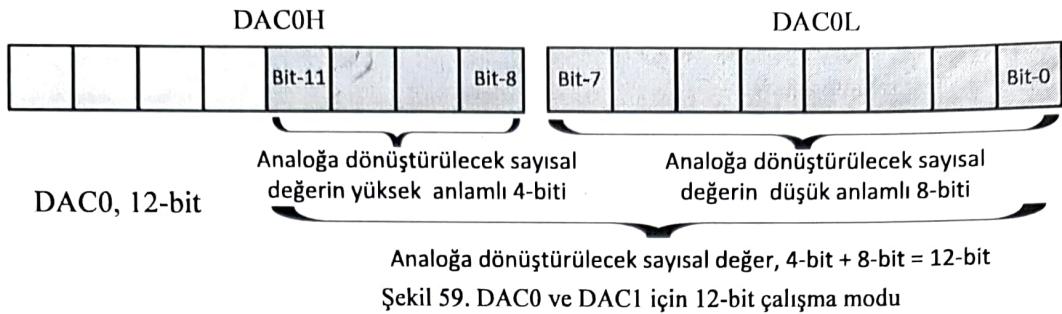




Şekil 58. DAC0 basitleştirilmiş blok şeması

ADUC841 mimarisinde bulunan her iki DAC’ın çalışma ayarları DACCON SFR’si üzerinden yapılır. Her iki DAC 12-Bit veya 8-Bit modunda kullanılabilir. 12-Bit çalışma modundan analoga dönüştürülecek sayısal veriler DAC1 için DAC1H- DAC1L ve DAC0 için DAC0H- DAC0L veri SFR’lerine aşağıda gösterildiği gibi yazılır.

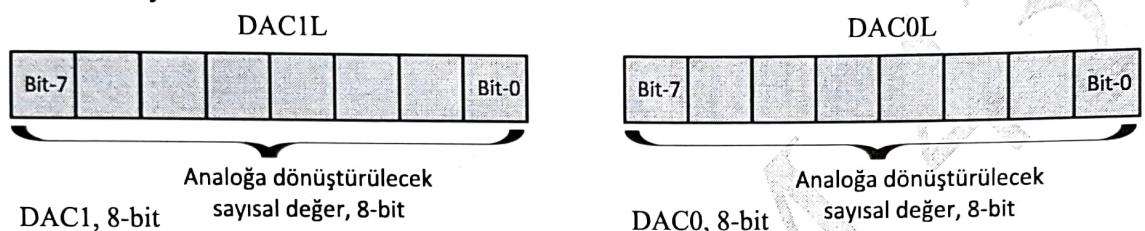




Şekil 59. DAC0 ve DAC1 için 12-bit çalışma modu

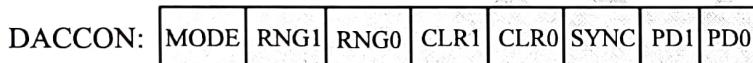
12-Bit asenkron çalışma modunda DACiL'e veri yazıldığı anda ilgili DAC çıkıştı güncellenir. Dolayısıyla bu çalışma modunda ilk önce DACiH'e ardından DACiL'e veri yazılmalıdır.

8-Bit çalışma modunda analoga dönüştürülecek sayısal veriler DAC1 için DAC1L ve DAC0 için DAC0L SFR'lerine yazılır.



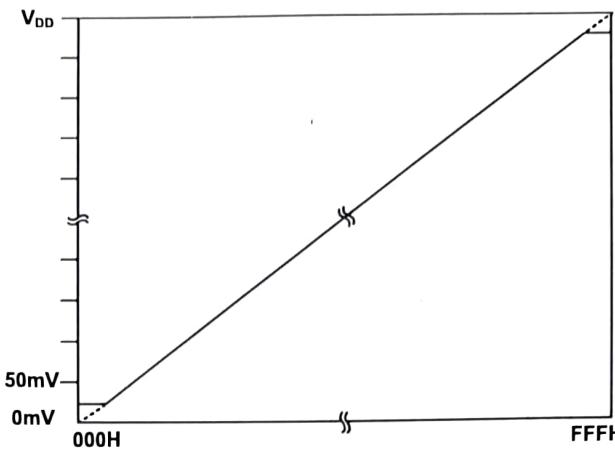
#### 9.4.1. DACCON Saklayıcısı

Aduc841 mimarisinde bulunan her iki DAC modülünün (DAC0 ve DAC1) çalışma ayarlarının yapıldığı DACCON SFR'si **bit adreslenemez**.



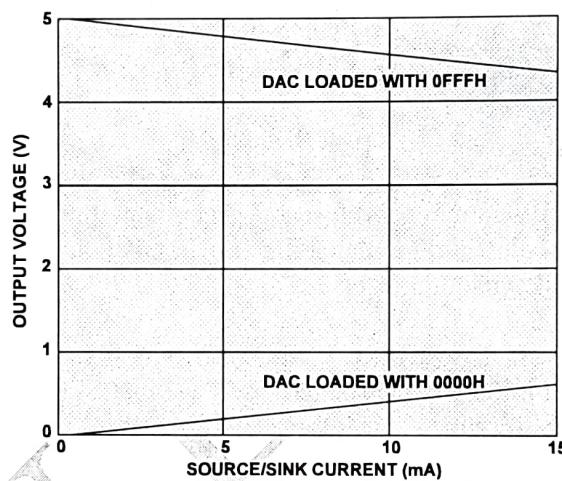
Bit	İsim	Açıklama
7	MODE	Setlendiğinde her iki DAC 8-bit modda, temizlendiğinde ise her iki DAC 12-bit modda çalışır
6	RNG1	Setlendiğinde DAC1 çıkış gerilim aralığı 0 V – AVDD, temizlendiğinde ise 0V-Vref (2.5V) olarak seçilir. <b>RNG=0 durumunda DAC'ın doğru çalışabilmesi için donanım gereği ADC'de aktif yapılmalıdır, (Power-On).</b>
5	RNG0	DAC0 çıkış gerilim aralığı seçme bitidir. Çalışması RNG1 ile aynıdır.
4	CLR1	Setlendiğinde DAC1 çıkışı DAC1H/L tutulan sayısal veriye karşılık gelen analog değere sahip olur. Temizlendiğinde ise DAC1 çıkışı 0V'a çekilir.
3	CLR0	DAC0 çıkış silme bitidir. Çalışması CLR1 ile aynıdır.
2	SYNC	Setlendiğinde, DACiL ( $i=0,1$ ) kaydedicisine veri yazıldığı anda DACi çıkıştı güncellenir. <i>Sync 1 → Direkt çıkış</i> Temizlendiğinde ise, her iki DAC senkron çalışma moduna girer. Bu modda, öncelikle DACiL/H kaydedicilerine veri aktarılır; daha sonra da SYNC biti lojik '1' yapılarak her iki DAC çıkışı aynı anda güncellenebilir.
1	PD1	Setlendiğinde DAC1 aktif olur (Power-On). Temizlendiğinde DAC1 pasif duruma geçer.
0	PD0	DAC0 çıkış aktif/pasif seçme bitidir. Çalışması PD1 ile aynıdır.

DAC çıkış işaretleri Şekil 55'de gösterildiği gibi Aduc841 mimarisinde bulunan dahili çıkış tamponları üzerinden dış dünyaya verilir. Şekil 59'daki grafikten, çıkış tamponunun maksimum ve minimuma yakın değerleri için doğrusallığının bozulduğu görülmektedir.



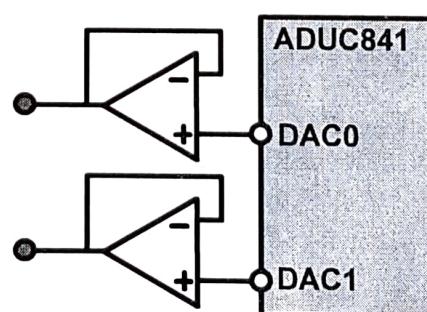
Şekil 60. DAC çıkışının sayısal değere göre değişimi

Üç noktalardaki doğrusal olmama problemi DAC çıkışları yüklenikçe Şekil 58'de gösterildiği gibi dahada artacaktır.



Şekil 61. DAC çıkışının sayısal değere göre değişimi

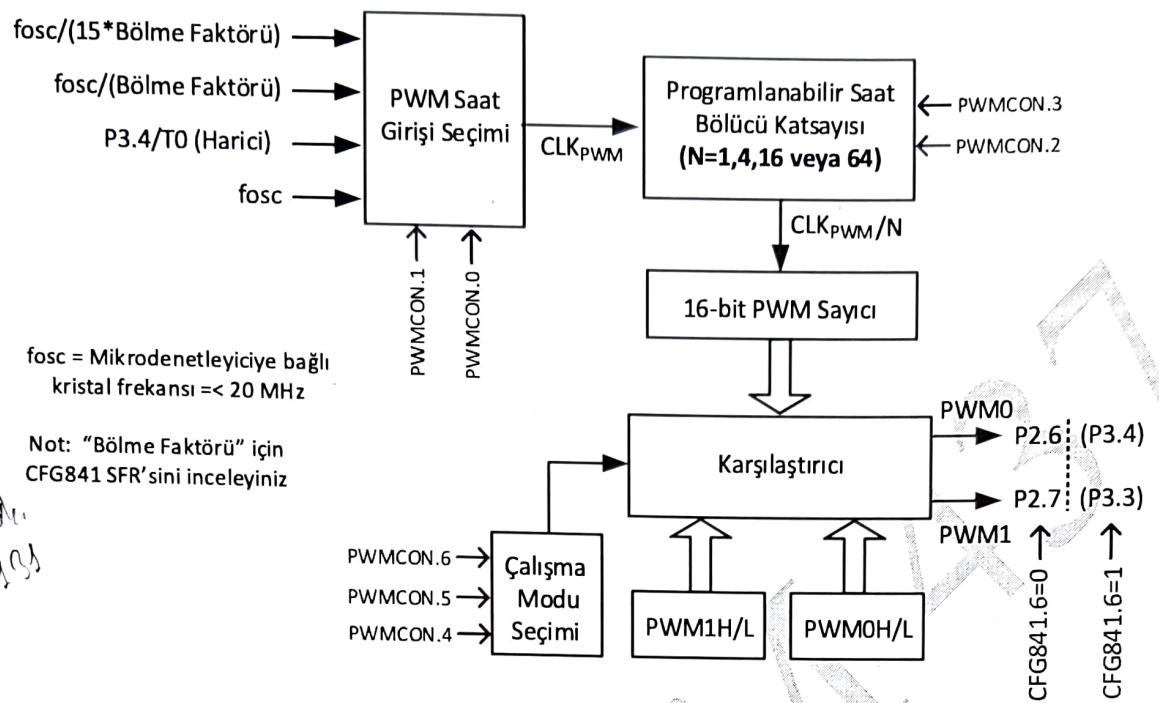
Bu nedenle hassas tasarımlarda, CFG841 kaydedicisinde bulunan DBUF biti setlenerek dahili tamponlar devre dışı bırakılabilir. Bu durumda DAC çıkışlarına Şekil 61'de gösterildiği gibi harici tamponlar bağlanmalıdır.



Şekil 62. DAC çıkışının sayısal değere göre değişimi

## 9.5. Darbe-Genişlik Modülasyon (Pulse-Width Modulator, PWM) Birimi

ADUC841 mimarisinde çözünürlük ve giriş kaynak saati yazılımla ayarlanabilen, 6 farklı çalışma moduna sahip PWM birimi bulunur. PWM için basitleştirilmiş blok diyagramı aşağıda verilmiştir.



Şekil 63. PWM blok diyagramı

Şekilde gösterildiği gibi PWM için 4 farklı saat girişi kullanılabilir. PWMCON SFR'si aracılığı ile seçilen saat girişine ait saat darbeleri programlanabilir (1,4,16,64) bölücüden geçirilerek 16-bit uzunluğundaki PWM sayacı için giriş kaynağını oluşturur.

PWM biriminin çalışma ayarları (çalışma modu, giriş saatı, bölücü katsayısı) PWMCON SFR'si aracılığı ile yapılır. PWM işaretinin duty cycle değeri ise PWM0H/L, PWM1H/L veri SFR'lerine, seçilen saat girişi ve programlanabilir bölücü katsayısanı bağlıdır.

PWMCON SFR'sine veri yazıldığı anda PWM sayıcısının değeri resetlenir (PWM sayacı=0). PWM sayacı girişindeki kaynak saat darbeleri ile 0'dan itibaren yukarı yönde sayar. PWM mimarisinde bulunan karşılaştırıcı anlık sayma değerini PWM0H/L ve PWM1H/L SFR'lerine yazılımla atanır. Değerler ile karşılaştırarak PWM çıkışlarının Lojik seviyesini belirler. PWM sayacının anlık sayma değeri çalışma modu için belirlenen üst değere ulaştıktan sonra tekrar sıfır düşer.

PWM çıkış pinleri P2.6 ve P2.7 (CFG841.6 = PWPO = 0 için) veya P3.4 ve P3.3 (CFG841.6 = PWPO = 1 için) olarak ADUC841 konfigürasyon saklayıcısı (CFG841) aracılığı ile belirlenir.

PWM'in 16-bit'lik çalışma modlarında (Mod 1,3,4 ve 6) ilk önce PWM0L ve PWM1L SFR'lerine ardından PWM0H ve PWM1H SFR'lerine değer yazılmalıdır. PWM0H/L ve PWM1H/L SFR'lerine yazılan yeni değer devam etmekte olan PWM periyodu tamamlandıktan sonra dikkate alınır.

### 9.5.1. PWMCON Saklayıcısı

PWM biriminin çalışma ayarlarının yapıldığı PWMCON SFR'si bit adreslenemez.

PWMCON:	SNGL	MD2	MD1	MD0	CDIV1	CDIV0	CSEL1	CSEL0
---------	------	-----	-----	-----	-------	-------	-------	-------

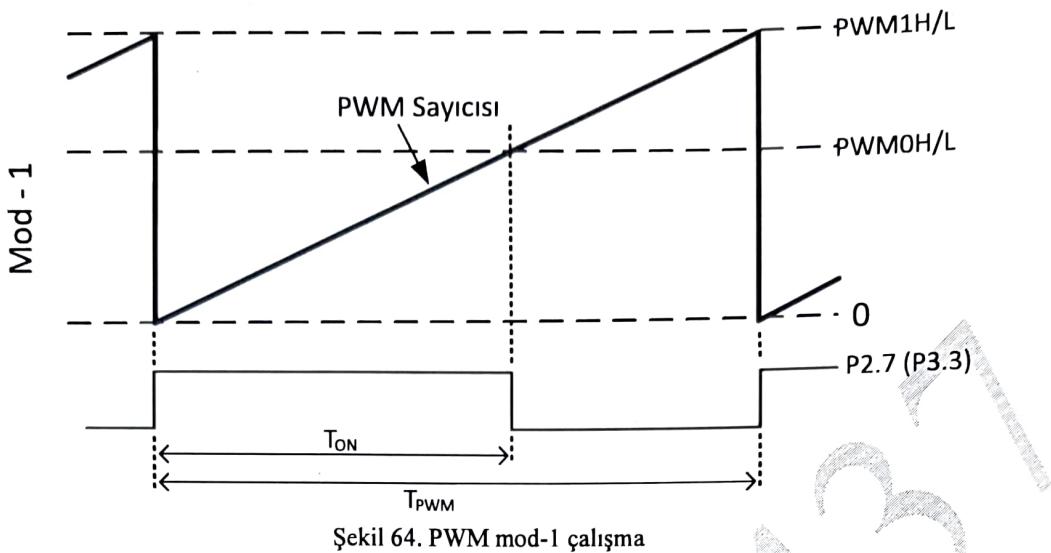
## PHASES ALER

Bit	İsim	Açıklama			
7	SNGL	Setlendiğinde tek çıkışlı PWM işaretü üretilir. P2.6 (veya P3.4) genel amaçlı kullanıma bırakılır. Çift çıkışlı çalışma modları için SNGL = 0 yapılmalıdır.			
6	MD2	MD 0, MD 1 ve MD 2 bitleri aşağıdaki tabloya göre PWM çalışma modunu belirler.			
5	MD1	MD 2   MD1   MD 0   Çalışma Modu			
4	MD0	0	0	0	<b>Mod 0 :PWM Kapalı</b>
		0	0	1	<b>Mod 1: P2.7 (veya P3.3) pininden tek çıkışlı PWM</b>
		0	1	0	<b>Mod 2: Çift çıkışlı (twin) 8-bit PWM</b>
		0	1	1	<b>Mod 3: Çift çıkışlı (twin) 16-bit PWM</b>
		1	0	0	<b>Mod 4: Çift NRZ 16-bit <math>\Sigma/\Delta</math> DAC</b>
		1	0	1	<b>Mod 5: Çift çıkışlı (dual) 8-bit PWM</b>
		1	1	0	<b>Mod 6: Çift RZ 16-bit <math>\Sigma/\Delta</math> DAC</b>
		1	1	1	<b>Rezerve</b>
3	CDIV1	CDIV1 ve CDIV0 bitleri aşağıdaki tabloya göre PWM Saat darbeleri bölgüsü katsayısını belirler.			
2	CDIV0	CDIV 1	CDIV 0	Bölgüsü Katsayısı, N	
		0	0	1	
		0	1	4	
		1	0	16	
1	CSEL1	1	1	64	
		CSEL1 ve CSEL0 bitleri aracılığıyla aşağıdaki tabloya göre PWM Saat kaynağı seçilir. <b>Not: Bölme Faktörü CFG841 SFR'sinden belirlenir.</b>			
0	CSEL0	CSEL 1	CSEL 0	PWM giriş saati, CLK <sub>PWM</sub>	
		0	0	$f_{osc}/(15 * \text{Bölme Faktörü})$	
		0	1	$f_{osc}/\text{Bölme Faktörü}$	
		1	0	<b>Harici giriş, P3.4</b>	
		1	1	$f_{osc}$	

### 9.5.2. PWM Çalışma Modları

**Mod 0:** PWM kullanıma kapalı, devre dışıdır. PWM pinleri genel amaçlı kullanıma açıktır.

**Mod 1:** Bu çalışma modunda hem darbe genişlik süresi ( $T_{on}$ ) hem de PWM çözünürlüğü yazılımla ayarlanabilir. Maksimum PWM çözünürlüğü 16-bit'dir.



Şekil 64. PWM mod-1 çalışma

- Şekilde gösterildiği gibi bu modda PWM işaretinin periyodu ( $T_{PWM}$ )  $PWM1H/L$ ,  $T_{ON}$  süresi ise  $PWM0H/L$  SFR'leri ile ayarlanır.

Yukarıda belirtildiği gibi PWMCON SFR'sine veri yazıldığında PWM sayacısının değeri sıfırlanır. PWM sayacı=0 olduğunda PWM çıkışı donanım tarafından Lojik 1 seviyesine çekilir. PWM sayacısının değeri Şekil 61'de gösterildiği gibi kaynak saat darbeleri ile artar. PWM sayacı= $PWM0H/L$  olduğu anda PWM çıkışı donanım tarafından Lojik 0 seviyesine çekilir. PWM sayacının değeri artmaya devam eder ve PWM sayacı= $PWM1H/L$  olduktan sonra gelen ilk saat darbesi ile PWM sayacı=0'a düşer ve PWM çıkışı donanım tarafından tekrar Lojik 1 seviyesine çekilir.

Mod 1'de  $PWM1H/L$  'a yüklenen değerin küçülmesi çıkış işaretinin frekansını artırırken PWM çözünürlüğü azalır. Örneğin  $PWM1H/L$  SFR'lerine 65535 yazıldığında 16-bit çözünürlüğünde PWM elde edilirken maksimum PWM frekansi 266 Hz olur. 12-bit çözünürlüğünde PWM ( $PWM1H/L = 4095$ ) ise maksimum PWM frekansi 4096 Hz olur.

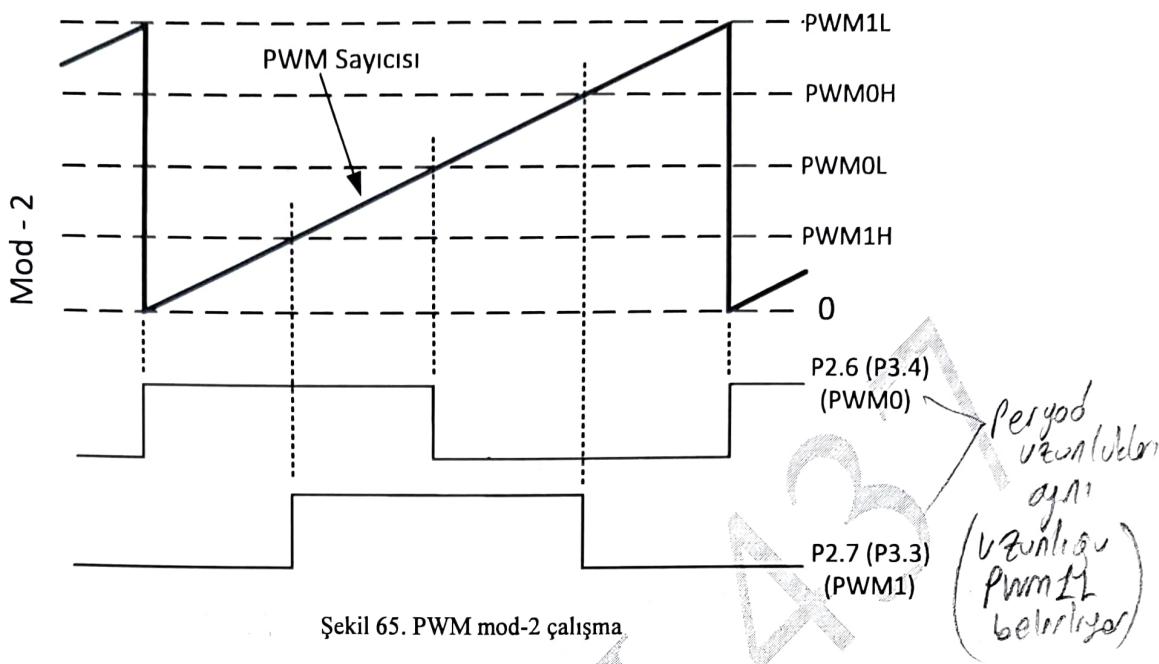
Mod 1'de PWM işareti P2.7 (CFG841.6 = PWPO = 0) veya P3.3 (CFG841.6 = PWPO = 1) pininden dış dünyaya aktarılır.

**Mod 2:** Bu çalışma modunda PWM0 ve PWM1 olmak üzere iki PWM çıkışı kullanılır. PWM çıkışlarının  $T_{ON}$  süreleri ayrı ayrı ayarlanabilir. Her iki PWM'in çözünürlüğü ise yazılımla ortak olarak ayarlanabilir. Maksimum PWM çözünürlüğü 8-bit'dir. PWM işaretleri P2.6 ve P2.7 (veya P3.4 ve P3.3) pinlerinden aşağıda gösterildiği gibi dış dünyaya aktarılır.

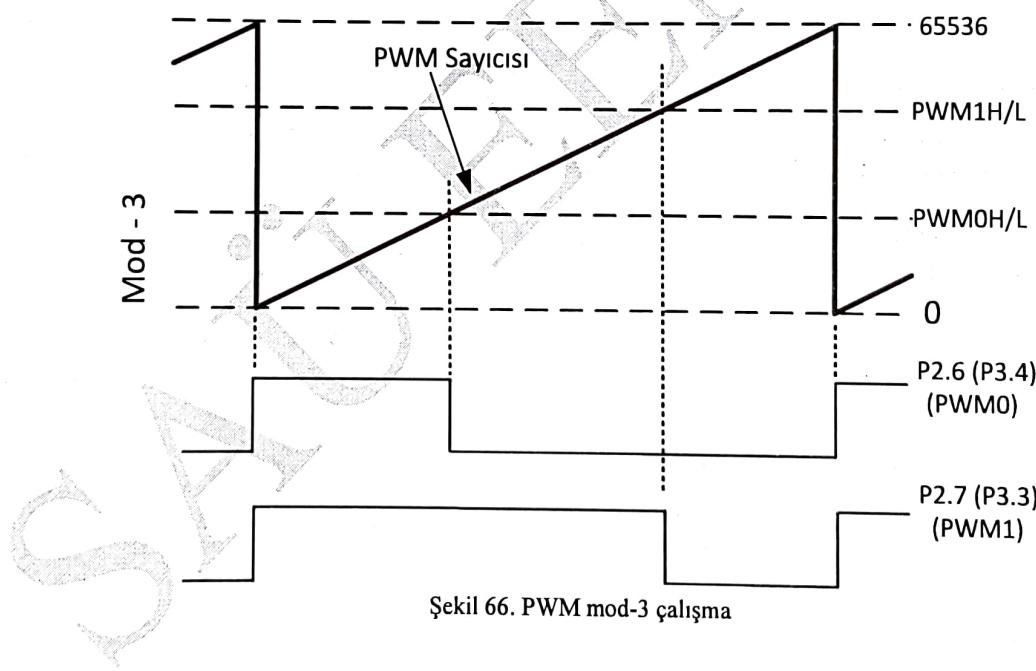
$PWM1L$  SFR'si her iki her iki PWM işaretinin çözünürlüğünü ve periyodunu  $T_{PWM}$  belirler. PWM0 için  $T_{ON}$  süresi  $PWM0L$  SFR'si ile PWM1 için  $T_{ON}$  süresi ise  $PWM0H - PWM1H$  SFR'leri arasındaki farkla ayarlanır.  $PWM1H = 0$  yapıldığında her iki PWM senkronize çalışır yani PWM sayacı = 0 olduğunda her iki PWM çıkışı aynı anda Lojik 1 seviyesine çekilir.

PWM1 Gon  $\rightarrow$  PWM0H - PWM1H

PWM0 Gon  $\rightarrow$  PWM0L

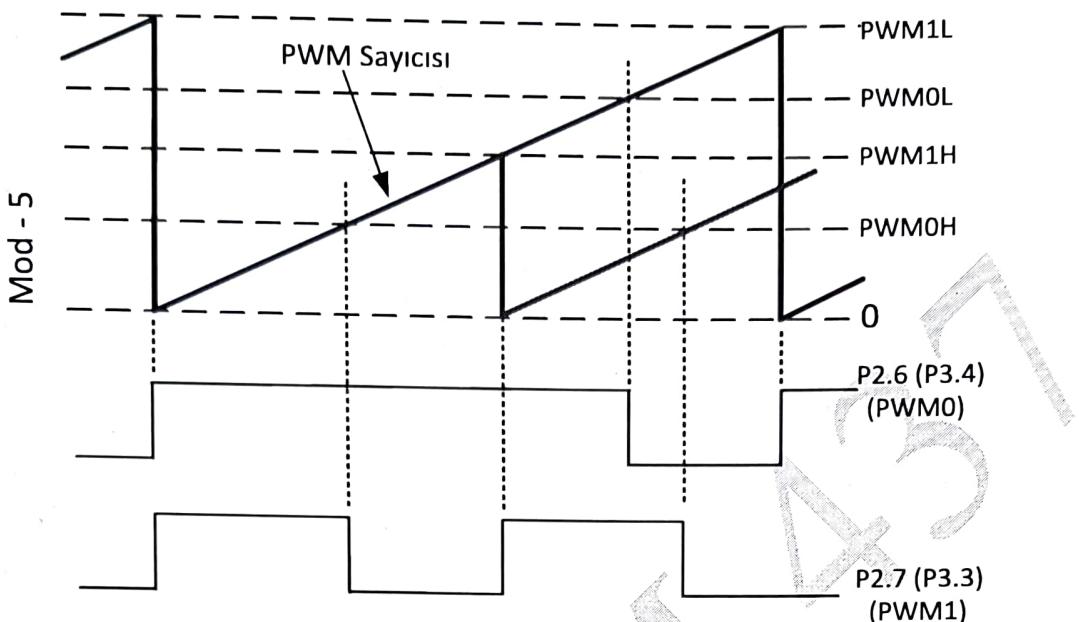


**Mod 3:** Bu çalışma modunda PWM0 ve PWM1 çıkışlarının Ton süreleri ayrı ayrı yazılımla ayarlanabilir. PWM çözünürlükleri ise sabit 16-bit'dir. Başka bir ifade ile PWM sayacının sayma aralığı, 0'dan 65536'ya olmak üzere, sabittir. PWM işaretleri P2.6 ve P2.7 (veya P3.4 ve P3.3) pinlerinden aşağıda gösterildiği gibi dış dünyaya aktarılır.



Şekilden görüldüğü gibi her iki PWM çıkışı senkronize çalışır, PWM sayacı = 0 olduğunda her iki PWM çıkışı donanım tarafından Lojik 1 seviyesine çekilir. PWM0 için TON süresi PWM0H/L PWM1 için TON süresi ise PWM1H/L SFR'leri ile ayarlanır.

**Mod 5:** Bu çalışma modunda PWM0 ve PWM1 çıkışlarının hem TON süreleri hemde çözünürlükleri birbirinden bağımsız olarak ayarlanabilir. Maksimum PWM çözünürlüğü 8-bit'dir. PWM işaretleri P2.6 ve P2.7 (veya P3.4 ve P3.3) pinlerinden aşağıda gösterildiği gibi dış dünyaya aktarılır.



Şekil 67. PWM mod-5 çalışma

Şekilden görüldüğü gibi PWM0 için  $T_{ON}$  süresi PWM0L, çözünürlük ise PWM1L SFR'sinden ayarlanır. Benzer şekilde PWM1 için  $T_{ON}$  süresi PWM0H, çözünürlük ise PWM1H SFR'sinden ayarlanır.

## 9.6. Kısır-Döngü Zamanlayıcısı (Watchdog Timer)

Mikrodenetleyiciler çalışma anında programlama hatası, elektriksel problemler, vb. çeşitli nedenlerden dolayı sonsuz bir kısır-döngü içerisinde girebilir. Bu durumda mikrodenetleyici donanımsal bir reset uygulanmadığı sürece bu kısır döngü içerisinde kalacak ve yapması gereken işlemleri yerine getiremeyecektir. Gelişmiş mikrodenetleyicilerde bu gibi durumlardan sakınmak amacıyla kısır-döngü zamanlayıcıları (Watchdog Timer, WDT) bulunur.

Aduc841 mimarisinde bulunan WDT bir kez yetkilendirildiğinde her makine çevriminde sayma değeri donanım tarafından otomatik olarak 1 artırılır. Yazılımla belirlenen "zaman aşım periyodu" içerisinde sayma değeri yenilenmez (0'a çekilmez) ise mikrodenetleyici donanım tarafından resetlenir veya WDT kesmesi oluşur. WDT kesmesi yazılımla pasif yapılamaz.

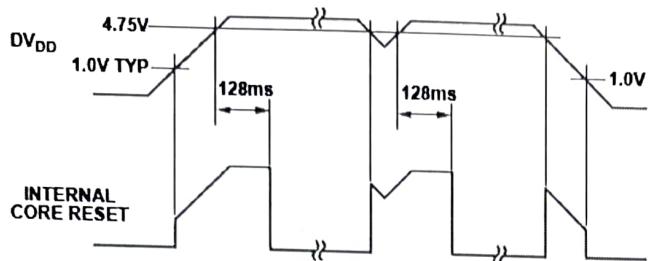
WDT çalışma ayarlarının yapıldığı WDCON SFR'si **bit adreslenebilir**.

WDCON:	PRE3	PRE2	PRE1	PRE0	WDI	WDS	WDE	WDWR
--------	------	------	------	------	-----	-----	-----	------

Bit	İsim	Açıklama					
7	PRE3	PRE3 – 0 bitleri WDT zaman aşım periyodunu aşağıdaki tabloya göre belirler.					
6	PRE2	PRE	PRE	PRE	PRE	Zaman Periyodu (t <sub>WD</sub> , ms)	Açıklama
		3	2	1	0		
5	PRE1	0	0	0	0	15.6	Reset veya Kesme
		0	0	0	1	31.2	Reset veya Kesme
		0	0	1	0	62.5	Reset veya Kesme
		0	0	1	1	125	Reset veya Kesme
		0	1	0	0	250	Reset veya Kesme
4	PRE0	0	1	0	1	500	Reset veya Kesme
		0	1	1	0	1000	Reset veya Kesme
		0	1	1	1	2000	Reset veya Kesme
		1	0	0	0	0.0	Derhal reset
		PRE3-0 > 1000					
		Rezerve					
3	WDIR	Setlendiğinde WDT zaman aşım periyodu dolduğunda WDT kesmesi oluşur. Sabit yüksek önceliğe sahip olan WDT kesmesi daima aktiftir, EA biti ile pasif yapılamaz. Alternatif zamanlayıcı olarak kullanılabilir. Temizlendiğinde WDT zaman aşım periyodu dolduğunda mikrodenetleyici resetlenir.					
2	WDS	Zaman aşım periyodu dolduğunda donanım tarafından setlenir. Harici donanımsal reset veya yazılımla temizlenebilir. WDT reseti ile temizlenmez.					
1	WDE	Setlendiğinde hem WDT kullanıma açılır hemde WDT sayacı temizlenir. Eğer bu bit zaman aşım periyodu süresi içerisinde tekrar setlenmez ise WDIR bitinin değerine bağlı olarak WDT kesmesi veya WDT reseti oluşur. Yazılımla temizlenebildiği gibi, WDT reseti, donanımsal reset ve PSM kesmesi ile de temizlenir. Temizlendiğinde WDT kullanıma kapatılır.					
0	WDWR	WDCON SFR'sine véri yazabilmek için aşağıda örnekte gösterildiği gibi ilk önce WDWR biti setlenmeli ve hemen ardından (arada başka hiçbir komut kullanmadan) WDCON SFR'sine istenen değer yüklenebilir. WDCON'a değer yazarken tüm kesmelerin pasif yapılması önerilir. CLR EA ; tüm kesmeler pasif SETB WDWR ; WDCON'a yazabilmek için WDWR = 1 yapıldı MOV WDCON, #72H ;WDCON'a yazıldı SETB EA ; Gerekli ise tüm kesmeler aktif...					

## 9.7. Power-On Reset (POR)

ADUC841 entegresi 3V ve 5V besleme gerilimi ile çalışan farklı versiyonlara sahiptir. 3V besleme ile çalışan versiyonlarda besleme gerilimi 2.7V – 3.6V arasında kaldığı sürece entegre normal çalışmasını sürdürür. 5V besleme ile çalışan versiyonlarda ise besleme gerilimi  $5V \pm 5\%$  aralığında olmalıdır. 5V'luk versiyonlarda dijital besleme ( $DV_{DD}$ ) değeri 4.5V altına düşüğünde POR mimarisi  $DV_{DD}$  ile beslenen dijital birimleri Reset'te tutar.  $DV_{DD}$  4.5V üstüne çıktıktan 128ms sonra dijital birimler Reset durumundan serbest bırakılarak normal çalışmaya bırakılır. Bu sürede besleme geriliminin kararlı bir şekilde minimum 4.75V'a ulaştığından emin olunmalıdır. Benzer şekilde enerji kesintilerinde de  $DV_{DD} < 1.0V$  olana kadar POR birimi işlemciyi reset'te tutar.



### 9.8. Besleme Gerilimi İzleyicisi (Power Supply Monitor, PSM)

Yetkilendirilmesi durumunda mikrodenetleyicinin dijital beslemesini ( $DV_{DD}$ ) gözlemler. Güç besleme pinlerinden herhangi birinin değeri yazılımla belirlenen eşik gerilim (3.08V veya 2.93V) değerinin altına düşüğünü bildirir ve gerekli ayarlar yapılır (Bakınız IEIP2 SFR) ise kesme üretir. PSM'nin doğru çalışabilmesi için analog besleme  $AV_{DD} > 2.7V$  olmalıdır. PSM enerji kesintilerine karşı kullanıcının bazı önemli verileri imkan sağlar. PSM ayarlarının yapıldığı PSMCON SFR'si bit adreslenebilir.

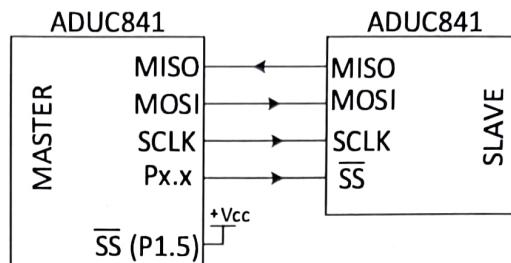
PSMCON:	-	CMPD	PSMI	TPD1	TPD0	-	-	PSMEN
---------	---	------	------	------	------	---	---	-------

Bit	İsim	Açıklama															
7	-	Rezerve															
6	CMP D	$DV_{DD}$ karşılaştırıcısının durumunu gösteren bu bit aadece okunabildir (read-only), değeri yazılımla değiştirilemez. Bu bit değerinin "1" olması $DV_{DD}$ beslemesinin yazılımla seçilmiş olan eşik değerinin üstünde, "0" olması ise eşik değerin altında olduğunu belirtir.															
5	PSMI	CMPA(Analog) veya CMPD(Digital) bitlerinden herhangi biri Lojik 0 olduğunda donanım tarafından PSMI biti setlenir. Bu bit gerekli ayarlar yapılmış ise kesme oluşturur. CMPA ve veya CMPD tekrar Lojik 1 seviyesine dönüp satabil hale geldiğinde 250 ms'lik bir zamanlayıcı yine donanım tarafından çalıştırılır. Zamanlayıcı taşılığında (250 ms süre geçince) PSMI biti donanım tarafından temizlenir. CMPA ve CMPD Lojik 1 ise PSMI biti yazılımla da temizlenebilir.															
4	TPD1	TPD1 ve TPD0 bitleri $DV_{DD}$ beslemesi için eşik değeri aşağıdaki gibi belirler.															
		<table border="1"> <thead> <tr> <th>TPD1</th> <th>TPD0</th> <th>Eşik Değeri (V)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Rezerve</td> </tr> <tr> <td>0</td> <td>1</td> <td>3.08</td> </tr> <tr> <td>1</td> <td>0</td> <td>2.93</td> </tr> <tr> <td>1</td> <td>1</td> <td>Rezerve</td> </tr> </tbody> </table>	TPD1	TPD0	Eşik Değeri (V)	0	0	Rezerve	0	1	3.08	1	0	2.93	1	1	Rezerve
TPD1	TPD0	Eşik Değeri (V)															
0	0	Rezerve															
0	1	3.08															
1	0	2.93															
1	1	Rezerve															
3	TPD0																
2	-	Rezerve															
1	-	Rezerve															
0	PSME N	Setlendiğinde PSM yetkilendirilir, aktif duruma geçer. Temizlendiğinde ise PSM yapısı pasif duruma geçer.															

### 9.9. Seri Çevresel Arayüz Haberleşmesi (SPI):

SPI (Serial Peripheral Interface) 8 bit veri alışverişini sağlayan senkron arayüz endüstri standartıdır. Motorola tarafından ismi verilmiştir. SPI'da 8 bit veri iletimi ve alımı aynı anda senkron olarak yapılır (Full-duplex).

ADUC841 mikrodenetleyicisinde mimarisinde SPI donanım olarak mevcuttur. SPI bağlantısı Şekil 1'de gösterildiği gibi 4 pin üzerinden gerçekleştir MISO (Master In/Slave Out Data), MOSI (Master Out/Slave In Data), SCLK (Serial Clock) ve SS (Slave Select Pin) pinlere ait ayrıntılı açıklamalar aşağıda verilmiştir.



Şekil 68. Master ve Slave mikrodenetleyicilerinin SPI bağlantı şekli

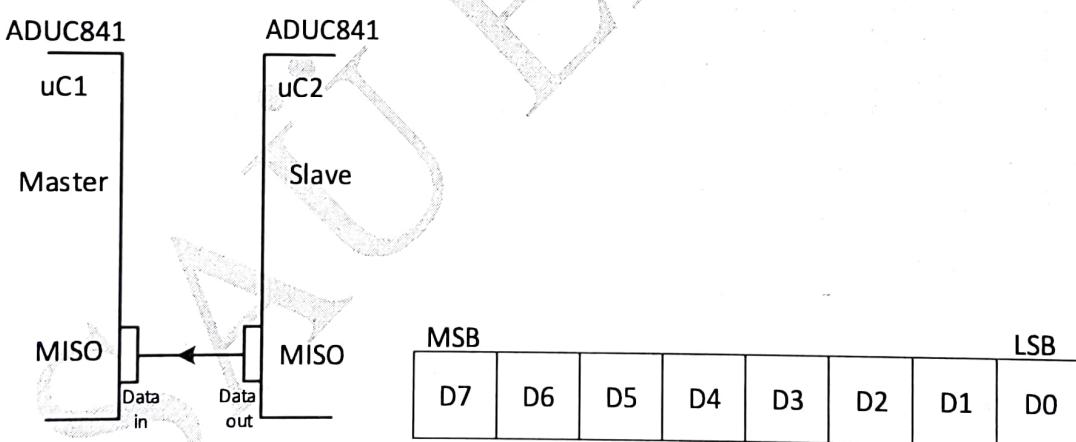
SPI pinleri ve I<sup>2</sup>C pinleri ortak kullanılır, bu nedenle herhangi bir anda aynı pinler üzerinden ancak bir arayüz etkin olabilir. Ancak CFG841'deki 1. bit MSPI biti lojik "1" olursa SPI haberleşme pinleri MISO, MOSI ve SCLOCK fonksiyonlarını sırasıyla P3.3, P3.4 ve P3.5 pinlerine atandığı için I<sup>2</sup>C'de aynı anda aktif yapılabılır.

SPI portu Master (ana) veya Uydu (slave) olarak seçilebilir.

SPI biriminin ayarları SPICON kaydedicisi yardımı ile yapılır.

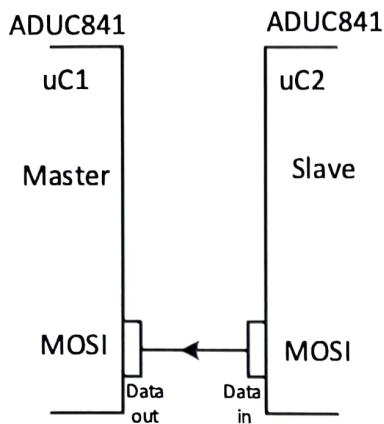
### 1) MISO (Master In/Slave Out Data)

MISO pini MASTER modda giriş hattı - SLAVE modda çıkış hattı olarak atanır. Aşağıdaki şekilde görüldüğü gibi MASTER'daki MISO hattı Slave'deki MISO hattına bağlanmalıdır. 8 bitlik veri, ilk bit MSB (Most Significant Bit) olacak şekilde seri olarak aktarılır.



### 2) MOSI (Master Out/Slave In Data)

MOSI pini MASTER modda çıkış hattı - SLAVE modda giriş hattı olarak atanır. Aşağıdaki şekilde görüldüğü gibi SLAVE'deki MOSI hattı MASTER'daki MOSI hattına bağlanmalıdır. 8 bitlik veri, ilk bit MSB (Most Significant Bit) olacak şekilde seri olarak aktarılır.



### 3) SCLOCK (Serial Clock) I/O Pin:

SCLOCK (Serial Clock) pininden MOSI ve MISO'dan senkron (eş zamanlı) seri veri alışverişi için saat darbeleri üretilir. SCLOCK saat darbeleri MOSI ve MISO veri hatları üzerinden iletilen ve alınan verileri senkronlamak için kullanılır. Tek bir veri biti her SCLOCK periyodunda ilettilir veya alınır. Bir bytelik (8-bit) veri 8 tane SCLOCK darbesi ile gönderilir veya alınır. SCLOCK pini MASTER modda çıkış, SLAVE modda giriş hattı olarak atanır.

- CPOL ve CPHA bitlerinin durumuna göre iletlenen veri yükselen veya düşen kenar saat darbelerinde ilettilir bu nedenle bu bitlerin hem SLAVE hem de MASTER için aynı olarak ayarlanması gereklidir.

Master modda;

- Bit rate (Bit hızı)
- Polarite (yon)
- Phase of clock (saat darbeleri fazı): CPOL, CPHA, SPR0 ve SPR1 bitleri ile SPICON'da ayarlanır.

Slave modda:

- Phase (Faz)
- Polarite,

CPHA, CPOL bitleri ile SPICON'da ayarlanır.

Hem MASTER hem SLAVE modda SCLOCK'un bir köşesinde veri ilettilir ve diğer köşesinde alınır. Bundan dolayı CPHA ve CPOL değerleri MASTER ve SLAVE'de aynı olmak zorundadır.

- 4)  **$\overline{SS}$  (Slave Select Pin)-Uydu seçim biti:** P1.5 pini işlemci Power on resetinden sonra ADC5 girişi (analog giriş) olarak donanımsal olarak ayarlanır. SLAVE işlemci seçimi için P1.5 pini yazılımla Lojik 0 yapılarak dijital giriş ( $\overline{SS}$ ) olarak ayarlanmalıdır. (CLR P1.5)
  - ✓ Slave modda  $\overline{SS}$  pini düşük lojik seviyede (Lojik 0) iken veri alınır ve gönderilir.  
(Zamanlama diyagramını inceleyiniz.)



- ✓ Tek MASTER-multi SLAVE modu vardır. (Birden çok MASTER olamaz)
- ✓ CPHA=1 ise  $\overline{SS}$  girişi sürekli olarak Lojik 0 seviyesinde tutulabilir.
- ✓ CPHA=0 ise  $\overline{SS}$  pinin 8 bit uzunlığundaki verinin ilk biti alınmadan veya gönderilmeden önce Lojik 0 son bit alındıktan veya gönderildikten sonra Lojik 1 yapmak zorundadır.
- ✓ SS pininin Lojik seviyesini slave modda SPICON'daki SPR0 biti ile okunabilir.

**SPICON:** SPI ayarlarının yapıldığı SFR'dir. Bit adreslenemez. Resetten sonraki değeri 00h'tır.

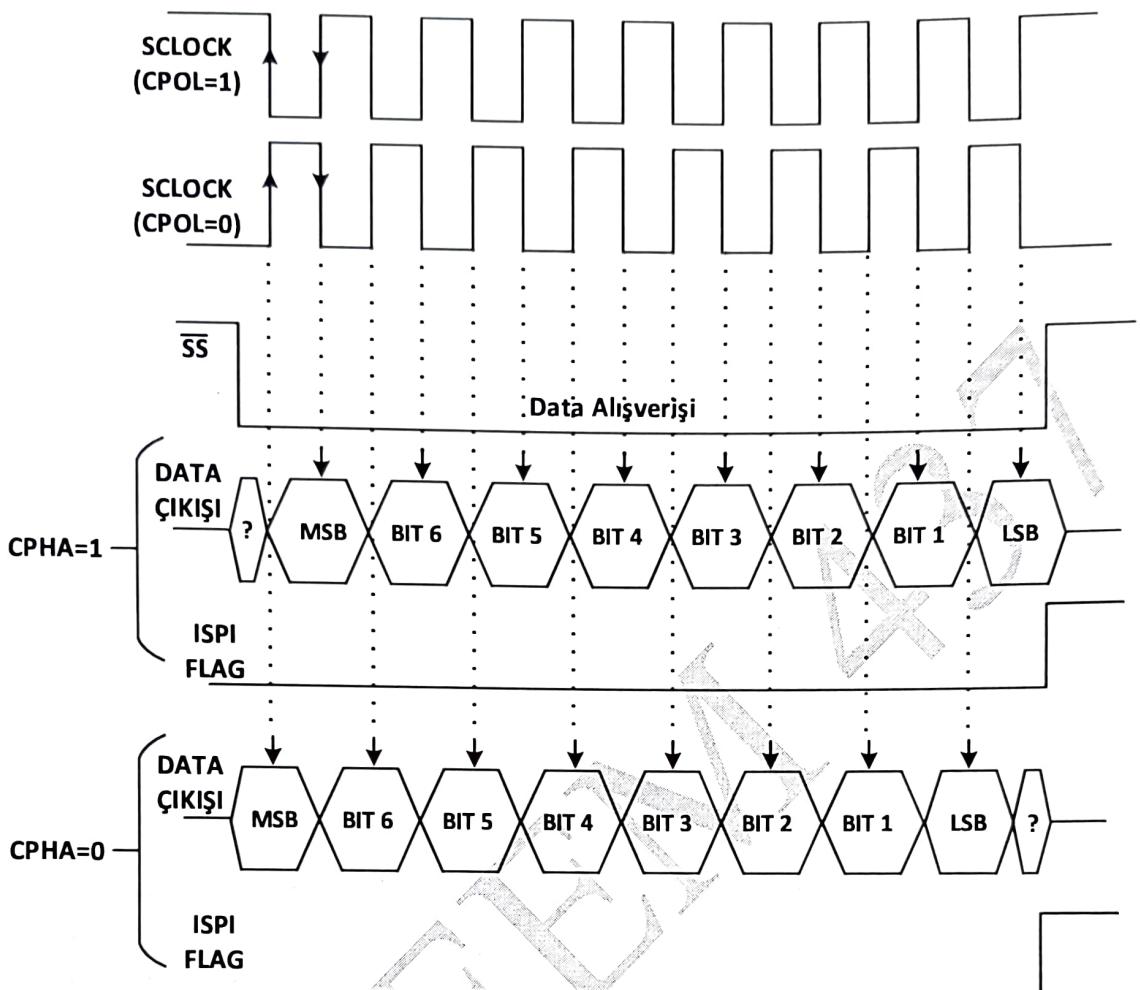
SPICON: 

ISPI	WGL	SPE	SPIM	CPOL	CPHA	SPR1	SPR0
------	-----	-----	------	------	------	------	------

Bit	İsim	Açıklama															
7	ISPI	SPI kesme bitidir. Her transferden (alma veya gönderme) sonra mikrodenetleyici tarafından setlenir. Kullanıcı veya dolaylı olarak SPIDAT SFR'si okunarak temizlenir.															
6	WGL	SPI iletimi sürerken SPIDAT'a veri yazılırsa mikrodenetleyici tarafından setlenir. Yazılımla temizlenir. (Yazma çıkışma bitidir) Veri gönderme işlemi bitmeden veri yazılırsa meydana gelir.															
5	SPE	SPI Enable, SPE=1 SPI yetkilidir. SPE=0 I2C yetkilidir. Eğer CFG841'in 1. biti MSPI pini Lojik "1" yapılırsa, I2C otomatik olarak yetkin olur.															
4	SPIM	SPI Master/Slave seçim bitidir. "1" yapıldığında Master modda (SCLOCK çıkış), "0" yapıldığında Slave Modda (SCLOCK çıkış) çalışılır.															
3	CPOL	Saat darbeleri polarite seçim biti, Lojik "1" iken SCLOCK saat darbeleri Lojik 1-0-1 şeklinde üretilir. Lojik "0" iken SCLOCK saat darbeleri Lojik 0-1-0 şeklinde üretilir. (Zamanlama diyagramına bkz.)															
2	CPHA	Saat darbeleri faz seçim biti, Lojik "1" iken veri ilk gelen SCLOCK saat darbesi değişiminde iletılır. Lojik "0" iken veri izleyen SCLOCK saat darbesi değişiminde iletılır. (Zamanlama diyagramına bkz.)															
1	SPR1	SPR1 SPR0 Clock bit iletim hızı															
0	SPR0	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>SPR1</td> <td>SPR0</td> <td>Clock bit iletim hızı</td> </tr> <tr> <td>0</td> <td>0</td> <td><math>f_{osc}/2</math></td> </tr> <tr> <td>0</td> <td>1</td> <td><math>f_{osc}/4</math></td> </tr> <tr> <td>1</td> <td>0</td> <td><math>f_{osc}/8</math></td> </tr> <tr> <td>1</td> <td>1</td> <td><math>f_{osc}/16</math></td> </tr> </table>	SPR1	SPR0	Clock bit iletim hızı	0	0	$f_{osc}/2$	0	1	$f_{osc}/4$	1	0	$f_{osc}/8$	1	1	$f_{osc}/16$
SPR1	SPR0	Clock bit iletim hızı															
0	0	$f_{osc}/2$															
0	1	$f_{osc}/4$															
1	0	$f_{osc}/8$															
1	1	$f_{osc}/16$															

Slave Modda (SPIM=0)  $\overline{SS}$  pinin lojik seviyesi SPR0 üzerinden okunabilir.

Tüm modlar için SPI zamanlama diyagramı aşağıda verilmiştir;



Şekil 69. Tüm modlar için SPI veri alış verisi zamanlama diyagramı

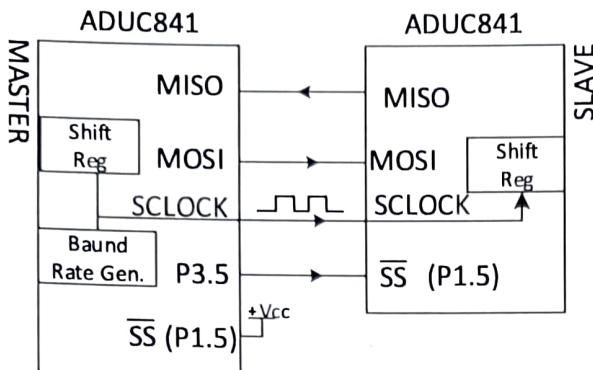
**SPI Master Mode:** Master modda SCLOCK pini her zaman çıkıştır. SPIDAT kaydedicisine veri yazar yazmaz 8 saat darbesi üretilir. 8 saat darbesi sonunda veri tamamen iletilmiş olur. ISPI bayrağı otomatik olarak setlenir ve SPI kesmesi aktif edilmiş ise kesme alt programına dallanılır ve alınan byte bilgisi giriş kaydedicisi SPIDAT'da hazır olur.

**SPI Slave Mode:** Slave Modda SCLOCK pin her zaman girişir. İletim, alım işlemi tamamlandıktan sonra shift kaydedicisindeki değer SPIDAT'ta tutulur. Eğer CPHA=1 ise 8 adet saat darbesinden sonra iletim sona ere. CPHA=0 ise  $\overline{SS}=1$  olduğunda sona erer.

- SPI kesme vektör adresi 3Bh'tir.

**ÖRNEK:** İki ADUC841 mikrodenetleyicisi SPI haberleşmesi ile haberleşecektir. Master'dan Slave' e 0'dan 10'a kadar olan sayılar sırasıyla gönderilecek, Slave'den alınan veriler ise 40h ile 50h arasındaki adreslere yazılacaktır. Slave mikrodeneleyicisinde Master'dan alınan veriler 40h ile 50h arasına kaydedecek ve Master'a 0'dan 10'a kadar sayılar sırası ile gönderecektir. SPI kesmesi kullanılacaktır.

Master ve Slave için programları ayrı ayrı yazınız. (Bitrate= fosc/16, CPHA=1, CPOL=0)



### MASTER için:

```
#include <Aduc841.h>
```

```
FLAG BIT 00H
```

```
ORG 00H
```

```
SJMP START
```

```
ORG 3BH
```

```
SJMP SPI_KESME
```

```
START:
```

```
MOV SPICON,#00110111B
```

; BITRATE= FOSC/16, CPHA=1, CPOL=0, MASTER  
; MOD SEÇİLDİ, SPI PORT YETKILENDİRİLDİ

```
MOV IEIP2,#01H
```

; SPI KESMESİ YETKILENDİRİLDİ, GENEL ;  
YETKILENDİRME

```
SETB EA
```

```
MOV R0,#00H
```

; SS PIN AYARLANDI

```
MOV R1,#40H
```

```
CLR P3.5
```

```
TRANSMIT:
```

```
INC R0
```

```
MOV SPIDAT,R0
```

```
SETB FLAG
```

```
XX:
```

```
JB FLAG, XX
```

```
MOV A,R0
```

```
CJNE A,#0AH,TRANSMIT
```

```
MOV R0,#00H
```

```
SJMP TRANSMIT
```

```
SPI_KESME:
```

```
SETB P3.5
```

```
CLR FLAG  
MOV @R1,SPIDAT  
INC R1  
CJNE R1,#50D,CONT  
MOV R1,#40H
```

CONT:

```
RETI  
END
```

**SLAVE için:**

```
#include <Aduc841.h>  
FLAG BIT 00H  
ORG 00H  
SJMP START  
ORG 3BH  
SJMP SPI_KESME
```

START:

```
MOV SPICON,#00100100B
```

; BITRATE= FOSC/16, CPHA=1, CPOL=0, SLAVE  
; MOD SEÇİLDİ, SPI PORT YETKİLENDİRİLDİ  
; SPI KESMESİ YETKİLENDİRİLDİ, GENEL  
; YETKİLENDİRME

```
MOV IEIP2,#01H
```

; SS PIN AYARLANDI

```
SETB EA
```

```
MOV R0,#00H
```

```
MOV R1,#40H
```

```
CLR P1.5
```

TRANSMIT:

```
INC R0
```

```
MOV SPIDAT,R0
```

```
SETB FLAG
```

XX:

```
JB FLAG,XX
```

```
MOV A,R0
```

```
CJNE A,#0AH,TRANSMIT
```

```
MOV R0,#00H
```

```
SJMP TRANSMIT
```

SPI\_KESME:

```
SETB P3.5
```

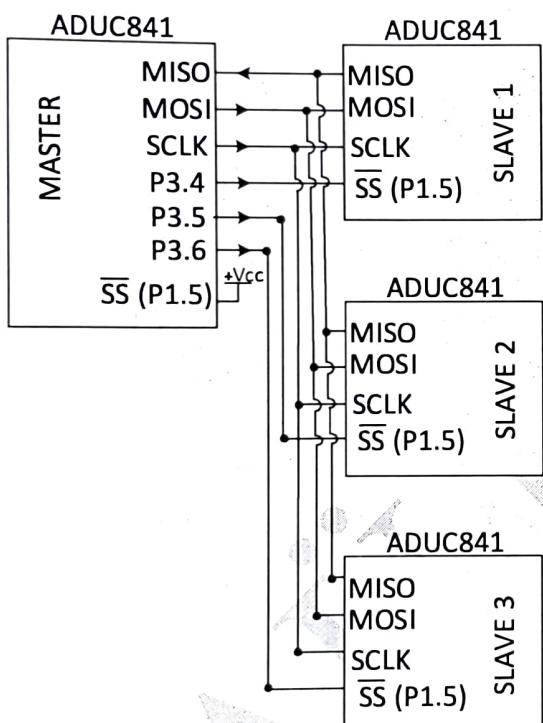
CLR FLAG

```
MOV @R1,SPIDAT  
INC R1  
CJNE R1,#50D,CONT  
MOV R1,#40H
```

CONT:

```
RETI  
END
```

Bir MASTER çoklu SLAVE işlemci SPI haberleşmesi için örnek bağlantı şéklı aşağıda gösterilmiştir.



**NOT:** Master'dan Slave mikrodenetleyicilerine giden Slave Select ( $\overline{SS}$ ) pin çıkışları yukarıdaki Şekilde görüldüğü gibi P3.4-P3.5 ve P3.6 pinleri olarak gösterilmiştir. Ancak bu pinler yazılımcının isteğine bağlı olarak boşta olan herhangi bir başka pine de bağlanabilir. Slave'e bağlanmış olan  $\overline{SS}$  girişleri bütün Slave'lerde mutlaka P1.5 pin girişine bağlanmalıdır. Master'daki  $\overline{SS}$  (P1.5) pini ise beslemeye bağlanmalıdır.

## Kaynaklar

---

1. H.Gümüşkaya, Mikroişlemciler ve 8051 Ailesi, Alfa Yayıncılık, 1998.
2. 8051 Data Book, Philips Semiconductors
3. K.J.Ayala, The 8051 Microcontroller Architecture, Programming and Applications, West Publishing, 1991.
4. A.T.Özcerit, M.Çakıroğlu, C.Bayılmış, 8051 Mikodenetleyici Uygulamaları, Papatya Yayıncılık, 2005.
5. A.Özdemir, Mikroişlemciler I Ders notları, 2005.
6. A.Özdemir, Mikroişlemciler II Ders notları, 2006.
7. I.S.MacKenzie, The 8051 Microcontroller, Prentice Hall, 1995.
8. Aduc841 Datasheet, Analog Devices.

