

KARABÜK ÜNİVERSİTESİ  
TEKNOLOJİ FAKÜLTESİ MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ



# MTM 305 MİKROİŞLEMCİLER

Arş. Gör. Emel SOYLU  
Arş. Gör. Kadriye ÖZ

# 8086 Mimarisi (Tekrar)

## Adresleme Modları

# 8086 Mimarisi

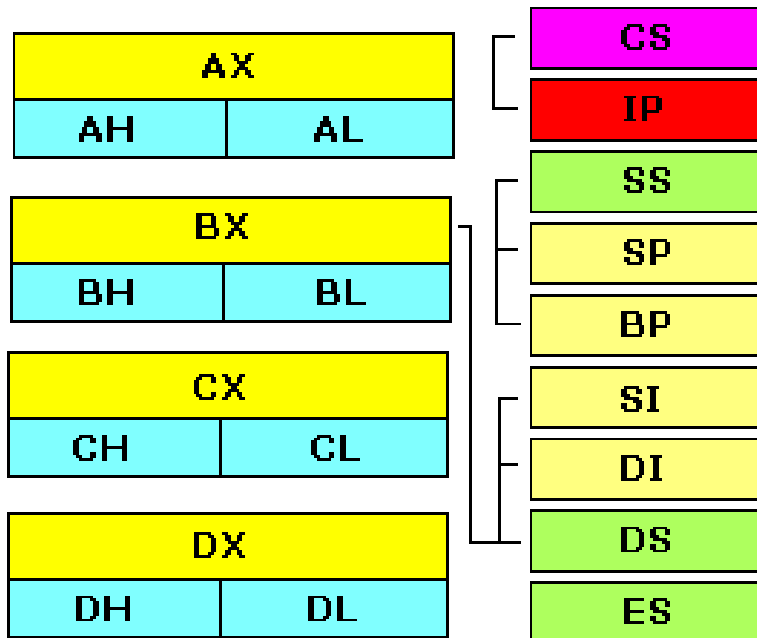
- 8086'da bulunan tüm iç register'lar ve veri yolları 16 bitlik genişliktedir.

# Adresleme

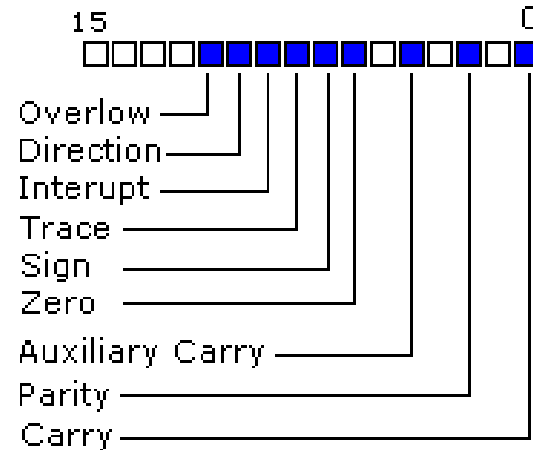
- Veri yolu 16-bit genişliğindedir. Adres yolu ise 20-bit genişliğindedir
- 8086, 1MB'lık hafıza bloğunu adresleyebilir ( $1M = 2^{20}$ )
- Ancak, en fazla adreslenebilir hafıza uzayı 64 KB büyüklüğündedir. Çünkü tüm dahili register'lar 16-bit büyüklüğündedir. 64 KB'lık sınırların dışında programlama yapabilmek için extra operasyonlar kullanmak gerekir

# 8086 Bileşenleri

## Central Processing Unit (or CPU)



## Arithmetic & Logical Unit (or ALU)



# Register'lar

- Register'lar, CPU içerisinde bulunduklarından dolayı, hafıza bloğuna göre oldukça hızlıdırlar.
- Hafıza bloğuna erişim için sistem veri yollarının kullanılması gereklidir.
- Register'daki verilerin ulaşılması için çok çok küçük bir zaman dilimi yeterli olur.
- Bu sebeple, değişkenlerin, register'larda tutulmasına çalışılmalıdır.
- Register grupları genellikle oldukça kısıtlıdır ve çoğu register'ın önceden tanımlanmış görevleri bulunur. Bu nedende, kullanımları çok sınırlıdır. Ancak, yine de hesaplamalar için geçici hafıza birimi olarak kullanılmak için en ideal birimlerdir.

# Register Tipleri

1. Genel Amaçlı Register'lar
2. İndis Register'ları
3. Özel Amaçlı Register'lar

# 1. Genel Amaçlı Register'lar

- 8086 CPU'da, 8 genel amaçlı register bulunur. Her register'ın ayrı bir ismi bulunur:
  - **AX** - accumulator register – akümülatör (**AH / AL**).
  - **BX** - the base address register – adres başlangıcı (**BH / BL**).
  - **CX** - the count register – sayma (**CH / CL**).
  - **DX** - the data register – veri (**DH / DL**).
  - **SI** - source index register – kaynak indisi.
  - **DI** - destination index register – hedef indisi.
  - **BP** - base pointer – temel gösterici.
  - **SP** - stack pointer – yığıt gösterici.



# Genel Amaçlı Register'lar (devam)

- Bu register'lar, isminin belirttiği amaçlar için kullanılmak zorunda değildir. Programcı, genel amaçlı register'ları istediği gibi kullanabilir.
- Register'ların ana amacı, bir değişkeni tutmaktır.
- Yukarıdaki register'ların tamamı 16-bitliktir.
- 4 genel amaçlı register (AX, BX, CX, DX), iki 8-bitlik register olarak kullanılabilir.
  - Örneğin eğer **AX=3A39h** ise, bu durumda **AH=3Ah** ve **AL=39h** olur.
  - 8-bitlik register'ları değiştirdiğiniz zaman, 16-bitlik register'lar da değişmiş olur.

## 2. Segment Register'ları

- **CS** – (Code Segment) Mevcut programın bulunduğu bölümü işaretler.
- **DS** – (Data Segment) Genellikle programda bulunan değişkenlerin bulunduğu bölümü işaretler.
- **ES** – (Extra Segment) Bu register'ın kullanımı, kullanıcıya bırakılmıştır.
- **SS** – (Stack Segment) yığının bulunduğu bölümü işaretler.

# Segment Register'ları (devam)

- Segment register'larında herhangi bir veriyi depolamak mümkündür. Ancak bu, güzel bir fikir değildir.
- Segment register'larının özel amaçları vardır. Hafızada ulaşılabilir bazı bölümleri işaretler.

# Segment Register'ları (devam)

- Segment register'ları, genel amaçlı register'ları ile birlikte çalışarak hafızada herhangi bir bölgeyi işaretleyebilir. Örneğin, fiziksel adres **12345h** (heksadesimal) işaretlenmesi isteniyor ise, **DS = 1230h** ve **SI = 0045h** olmalıdır.
- CPU, segment register'ı 10h ile çarpar ve genel amaçlı register'da bulunan değeri de ilave eder ( $1230h \times 10h + 45h = 12345h$ ).

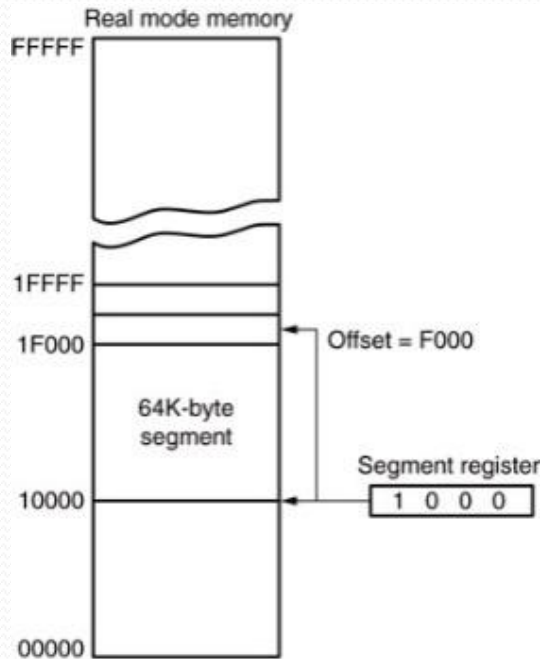
# Segment Register'ları (devam)

- 2 register tarafından oluşturulmuş olan adrese, **effective address** (efektif adres) ismi verilir.
- **BX, SI ve DI register'ları**, **DS** ile birlikte çalışır; **BP** ve **SP** register'ları ise **SS** ile birlikte çalışır.
- Diğer genel amaçlı register'lar, efektif adres oluşturmak için kullanılmazlar. Ayrıca, BX efektif adres oluşturulmasında kullanılırken, BH ve BL kullanılmaz.

# Segment ve Offset

- Tüm hafıza adresleri segment adresine offset adresi ilave edilmesi ile bulunur.
  - **segment adresi:** Herhangi bir 64 KB'lık hafıza bölümünün başlangıcını gösterir.
  - **offset adresi:** 64 KB'lık hafıza bölümünde herhangi bir satırı belirtir.

# Segment ve Offset (devam)



- Segment register'ı 1000h değerine sahip ise, 10000h ile 1FFFFh aralığında bir hafıza adresine karşılık gelir.
  - 64KB'lık bir aralıktır
- Offset değeri F000h ise 1F000h adresindeki hafıza satırına karşılık gelir.

# Segment ve Offset (devam)

- Başlangıç adresi belli ise, bitiş adresi FFFFh ilave edilerek bulunur.
  - Çünkü segment register'ı 64 KB'lık bir bölümü işaretler.
- Offset adresi, segment adresine ilave edilir.
- Segment ve offset adresleri 1000:2000 biçiminde de yazılabilir.
  - Bu durumda segment adresi 1000H ve offset'te 2000H'dir.



# Öntanımlı Segment ve Offset Register'ları

- CS:IP

- CS, kod bölümünün başlangıcına işaret eder. IP, kod bölümü içerisinde bir sonraki komutun bulunduğu hafıza adresine işaret eder.
- Eğer  $CS=1400H$  ve  $IP=1200H$ , mikroişlemci, bir sonraki komutu  $1400H + 1200H = 15200H$  adresinden okur.

## Öntanımlı Segment ve Offset Register'ları (devam)

- SS:SP veya SS:BP
  - Yığın bölümünü kullanan komutlar kullanır.
- DS register'ı, BX, DI, SI, 8-bit'lik veya 16-bit'lik sayı ile birlikte
- ES:DI (string komutları için)

### 3. Özel Amaçlı Register'lar

- **IP** –instruction pointer – komut işaretleyicisi.
  - **IP** register'ı CS ile birlikte, halihazırdaki çalıştırılan komutu işaretler.
- **Flags (Bayrak) register**
  - **Flags register**, CPU tarafından, matematiksel operasyonlardan sonra otomatik olarak değiştirilir. Bu register sayesinde, elde edilen sonucun çeşidi ve durumu ile ilgili bilgiler, program tarafından kullanılabilir.

### 3. Özel Amaçlı Register'lar

- Genellikle, bu register'lara doğrudan erişim bulunmaz.
- Ancak, sistem register'larının değerleri, daha sonra öğreneceğiniz bazı metotlar sayesinde değiştirilebilir.

# Flag Bit'leri

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				O	D	I	T	S	Z		A		P		C

- **C (carry):** toplama işleminden oluşan elde ve çıkarma işleminden oluşan ödünçleri tutar.
  - Ayrıca, hata durumlarını gösterir
- **P (parity):** bir sayıda bulunan 1'lerin tek sayıda mı yoksa çift sayıda mı olduğunu belirtir.
  - 0 tek parity; 1 çift parity.
  - Eğer bir sayı 3 tane binary 1 bit var ise, sayı tek parity'ye sahiptir.
  - Eğer bir sayıda hiç 1 bit yok ise, sayı çift parity'ye sahiptir.

# Flag Bit'leri (devam)

- **A (auxiliary carry):** 3 ve 4. bit pozisyonları için geçerli olmak üzere, toplama işleminden oluşan elde ve çıkarma işleminden oluşan ödünçleri tutar.
- **Z (zero):** aritmetik veya mantık operasyonunun sonucunun sıfır olup olmadığı bilgisini tutar.
- **S (sign):** çalıştırılan aritmetik veya mantık operasyonunun sonucunda elde edilen sayının aritmetik yönünü (pozitif veya negatif) belirtir.


# Flag Bit'leri (devam)

- **T (trap):** çip üzeri debug özelliğine olanak tanır.
- **I (interrupt):** INTR (**i**nterrupt **r**equ~~e~~st – interrupt isteği) girişini kontrol eder.
- **D (direction):** DI ve/veya SI register'ları için arttırma veya azaltma modlarından birini seçer.
- **O (overflow):** iki yönlü sayının toplamı veya çıkarılması durumlarında kullanılır.
  - overflow olması, sonucun, 16-bitlik kapasiteyi aştığını gösterir.



# Adresleme Modları





Adresleme modları belleğin nasıl kullanıldığını, belleğe nasıl erişileceğini ve verilerin belleğe nasıl yerleştirileceğini belirler. Aşağıda verilen anlatımda kullanılan kısaltmaları anlamaları şu şekildedir.

Register: Kaydedici (Yazmaç)

Memory: Bellek

Immediate : Acil veri, doğrudan veri

# Acil Adresleme( Immediate Addressing)

## (Anlık Adresleme)

Doğrudan sabit bir değer bir kaydediciye aktarılır. Sabit değerın büyüklüğü ile register uyumlu olmalıdır. **Örneğin** 8 bitlik bir kaydediciye 16 bitlik bir değer yüklenemez.

Genel Kullanımı:

KOMUT register, immediate

Örnek:

MOV CL, 16h

MOV DI, 2ABFh

MOV AL, 4567h ; Yanlış

## Kaydedici Adresleme (Register Addressing) (Yazmaç Adresleme)

Bu adresleme modunda her iki operand (işlenen) de kaydedicidir.

Genel kullanımı:

KOMUT register, register

Örnek:

MOV AL, BL

INC BX

DEC AL

SUB DX, CX

## Doğrudan adresleme(Direct addressing)

Doğrudan bir adres değeri kullanılır. Bir adresten bir kaydediciye veri aktarımı gerçekleştirilir. Bir başka ifade ile operandlardan birisi adres belirtir.

Genel kullanımı:

KOMUT register, memory veya

KOMUT memory, register

Örnek:

MOV AX, [1000]

TOPLAM DW 20 ; Burada TOPLAM bir adrestir. 20 sayısı bu adresin içindeki değerdir.

MOV AX, TOPLAM ; TOPLAM adresindeki değeri AX e at.

MOV TOPLAM, AX

## Dolaylı adresleme(Indirect addressing) (Kaydediciye dayalı dolaylı adresleme)

Etkin adres değeri (offset) BX, BP, SI, DI kaydedicilerinden birinde bulunur.

Genel kullanımı:

KOMUT register, [BX/BP/SI/DI] veya

KOMUT [BX/BP/SI/DI], register

Örnek:

```
MOV AX,[SI]
```

```
MOV BX,1000 ;
```

```
SUB DX, [BX]
```

```
MOV [SI], AL
```

```
TABLO DB 5,9,0,3,-7
```

```
MOV BX, OFFSET TABLO ; Bunun yerine LEA BX,TABLO kullanılabilir.
```

```
MOV AX, [BX] ; Kaydedici dolaylı adresleme var
```

## Dolaylı adresleme(Indirect addressing) (Kaydediciye dayalı dolaylı adresleme) (2)

LEA komutu (Load Effective Address) ofset adresi bir kaydediciye yüklemek için kullanılır.

Yukardaki iki komut yerine aşağıdaki komut kullanılabilir.

MOV AX, TABLE ; doğrudan adresleme var. İlk iki değer AX e yüklenir.

Herhangi bir bellek bölgesi dolaylı bir adresleme ile adreslenip içine sabit bir değer atanmak istendiği zaman atanacak değerın uzunluğu byte ptr ve word ptr ile belirtilmelidir.

MOV [BX], 12 ; yanlış 00

MOV byte ptr [BX],12 ; doğru 12 sabit değeri bayt olarak BX in gösterdiği adrese yerleşecek

MOV word ptr [BX],1234 ;doğru 1234 sabit değeri word olarak BX in gösterdiği adrese yerleşecek

## İndisli adresleme (Indexed addressing)

Dolaylı adreslemede köşeli parantez içinde kalan BX/BP/SI/DI kaydedicilerine bir indis değeri eklenerek kullanılır.

Genel kullanımı:

KOMUT register, [BX/BP/SI/DI+indis] veya

KOMUT [BX/BP/SI/DI+indis], register

### Örnek:

MOV AX, [SI+4]

ADD [DI-6],CX

MOV CX, [SI+DI+7] ;

MOV AX, [BX+DI]

MOV AX, [SI-7] ; bu komut yerine MOV AX, [SI]-7 de kullanılabilir.

MOV DI, 2

MOV AL, TABLE [DI] ;AL kaydedicisine TABLE dizisinin 2. Elemanını yükler.

# DİKKAT

1-) KOMUT memory, memory

Şeklinde bir kullanım geçerli değildir. Bir bellek bölgesinde başka bir bellek bölgesine veri aktarımı yoktur.

2-) Sabit bir değer doğrudan Segment Registerine atanamaz.

MOV DS,1234 Yanlıştır. Bunun yerine aşağıdaki kullanım geçerlidir.

MOV AX, 1234 ; önce kaydediciye

MOV DS, AX ; sonra segment kaydedicisine değer atanır.

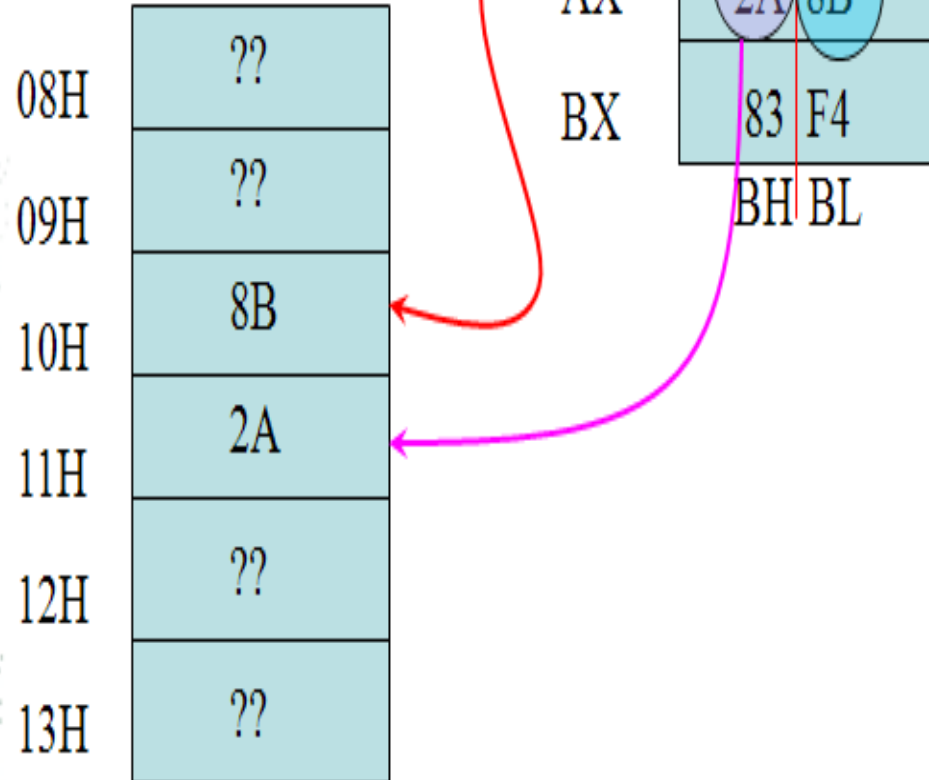


## 16 Bitlik bir verinin belleğe yerleşmesi

Belleğin her bir hücresi 8 bit olduğundan 16 bitlik verinin düşük değerlikli 8 bitlik kısmı adresin düşük değerlikli kısmına, yüksek değerlikli 8 bitlik kısmı adresin yüksek değerlikli kısmına yerleşir. Aşağıda verilen örnekte AX kaydedicisinde 2A8BH değeri yer almaktadır. Düşük değerlikli değer AL de yer almakta ve 10H adresine yerleşmektedir. Yüksek değerlikli kısımda yer alan AH daki değer 2AH değeri ise 11H adresine yerleşmektedir.

- MOV AX, 2A8BH

- MOV [10H], AX



Benzer olarak 32bitlik bir veriyi(doubleword) bu şekilde düşünerek yerleştirebilirsiniz.

Örneğin:

SAYI DB 1A2F56FFH verisini 100H adresinden itibaren yerleştirecek olursa şu şekilde olur.

Adres	veri
100	FF
101	56
102	2F
103	1A
104	
105	
106	

# Dizi Tanımlaması ve Elemanlarına Erişimi

DIZI DB 5, 6, 7, 8, 9, 0, -6, -9, 3 şeklinde tanımlaması yapılır.

LEA SI, DIZI ; şeklinde dizinin başlangıç adresini SI kaydedicisine alınır.

**Örnek:** Bir dizide bulunan elemanların toplamını bulan program kodunu yazınız.

```
.MODEL SMALL
```

```
.STACK 64
```

```
.DATA
```

```
DIZI DB 5, 6, 7, 8, 9, 0, -6, -9, 3, 8
```

```
SONUC DW ?
```

```
.CODE
```

```
ANA PROC FAR
```

```
MOV AX, @DATA
```

```
MOV DS, AX
```

```
MOV AL, 0
```

```
MOV CX, 10
```

```
LEA SI, DIZI ; DIZI nin baslangic offset  
adresini SI ya yüklenir
```

```
BAS:
```

```
MOV BL, [SI]
```

```
ADC AL, BL
```

```
INC SI
```

```
LOOP BAS
```

```
MOV SONUC, AL
```

```
MOV AH, 4CH
```

```
INT 21H
```

```
ANA ENDP
```

```
END ANA
```

## Örnekler

Aşağıdaki komutların adresleme modlarını bulun

- MOV DH,[BX+DI+20H]
- MOV AL,BL
- JMP etiket1
- MOV SP,BP
- MOV AX,dizi
- MOV CH,[BP+SI]
- MOV AX,dosya[BX+DI]
- MOV [DI],BH
- MOV AX,44H
- MOV [BX+SI],SP
- MOV AL,sayı
- MOV AX,[DI+100H]
- MOV BL,44
- MOV dizi[SI],BL
- MOV liste[SI+2],CL
- MOV CX,[BX]



Beni dinlediğiniz için teşekkür ederim.