

Bölüm 3: Assembly

Mikroişlemciler

Assembly dili

Basit bilgisayar modeli

Sistem yolu

CPU'nun içi

Genel amaçlı yazmaçlar

Kesim yazmaçları

Fiziksel adres

IP ve bayraklar

İlk program

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

Öğrenme Çıktıları

Assembly'nin yeri

Assembly dilinin makine dili ve yüksek seviyeli dillerle ilişkisini açıklamak.

Bilgisayar modeli

CPU, bellek, G/Ç aygıtları ve sistem yolunun görevlerini tanımlamak.

8086 yazmaçları

Genel amaçlı, kesim ve özel amaçlı yazmaçları ve 8/16 bit ilişkisini kullanmak.

Adres hesabı

Kesim:bağıl konum çiftinden 20 bitlik fiziksel adresi hesaplamak.

IP ve bayraklar

IP'nin nasıl ilerlediğini ve bir işlemin bayrakları nasıl etkilediğini izlemek.

İlk program

emu8086 ile basit bir .COM programını yazıp adım adım çalıştırmak.

Assembly Dili

- **Düşük seviyeli** bir programlama dilidir: her komut, işlemcinin tek bir makine komutuna karşılık gelir.
- Makine kodundaki sayılar yerine akılda kalıcı **komut kısaltmaları** (mnemonic) kullanılır:
`MOV` , `ADD` , `JMP` ...
- Bilgisayarın yapısı (yazmaçlar, bellek, sistem yolu) hakkında **bilgi sahibi olmayı** gerektirir.
- Bilgisayarın temel yapısını ve bir programın gerçekte nasıl çalıştığını **anlamak** açısından önemlidir.
- Her işlemci ailesinin kendi assembly dili vardır; bu derste **Intel 8086** ve **emu8086** kullanılır.

Tek satır, üç görünüm

Assembly: `MOV AX, 5`

Onaltılık makine kodu: `B8 05 00`

CPU'nun gördüğü:

`10111000 00000101 00000000`

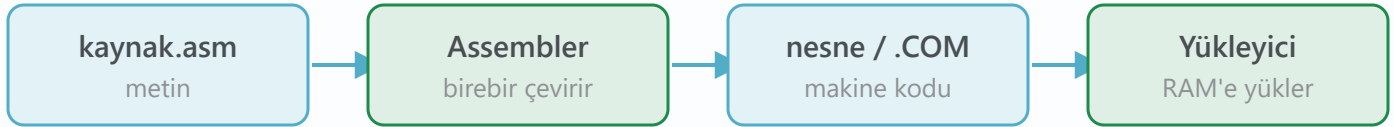
16 bitlik 0005h, bellekte düşük bayt önce (05 00) saklanır.

Dil Düzeyleri

Düzyey	Örnek	Kim anlar?
Yüksek seviyeli dil (C)	<code>x = 5; x++; return;</code>	İnsan; derleyici makine koduna çevirir
Assembly	<code>MOV AX, 5 · INC AX · RET</code>	İnsan; <i>assembler</i> birebir çevirir
Makine dili (onaltılık)	<code>B8 05 00 · 40 · C3</code>	CPU (ikili olarak)
Makine dili (ikili)	<code>10111000 00000101 00000000 · 01000000 · 11000011</code>	CPU

Bir C satırı çoğu zaman **birçok** makine komutuna dönüşür; bir assembly satırı ise (direktifler hariç) **tam olarak bir** makine komutudur. Bu yüzden assembly, işlemcinin “ne yaptığını” doğrudan gösterir.

Assembler ve emu8086



EXE için araya bağlayıcı (linker) girer: .obj → .exe

Assembler

Komut kısaltmalarını ve etiketleri makine koduna, değişken adlarını adreslere çevirir. Örnek: MASM, TASM, NASM.

emu8086

Editör + assembler + 8086 öykünücüsü (emulator). Yazmaçları, belleği ve bayrakları *adım adım* gösterir.

.COM / .EXE

.COM tek kesimli, en fazla 64 KB'lık basit biçimdir (ORG 100h). .EXE çok kesimli olabilir.

Bu derste örnekler emu8086 sözdizimiyle yazılır; çoğu MASM/TASM ile de uyumludur.

Neden Assembly?

Assembly dili ile oluşturulmuş bir program:

Artıları

- + Daha **az yer** kaplar (gereksiz kod yok).
- + Yalnızca yapması gereken işlemleri yaptığından **daha hızlıdır**.
- + Yüksek seviyeli dillerin yapamadığı işlemleri yapabilir: port G/Ç, kesme işleme, bayraklara doğrudan erişim.
- + Önyükleyici, aygıt sürücüsü, gömülü sistem gibi alanlarda vazgeçilmezdir.

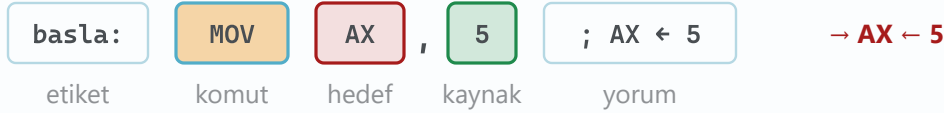
Eksileri

- **Taşınabilir değildir**: 8086 kodu ARM'da çalışmaz.
- Yazması, okuması ve hata ayıklaması zahmetlidir.
- Büyük projelerde verim düşüktür; modern derleyiciler çoğu zaman yeterince hızlı kod üretir.

Uygulamada genellikle **karma** yaklaşım kullanılır: program yüksek seviyeli dilde, zamanlaması kritik küçük parçalar assembly ile yazılır.

Komut Satırının Yapısı

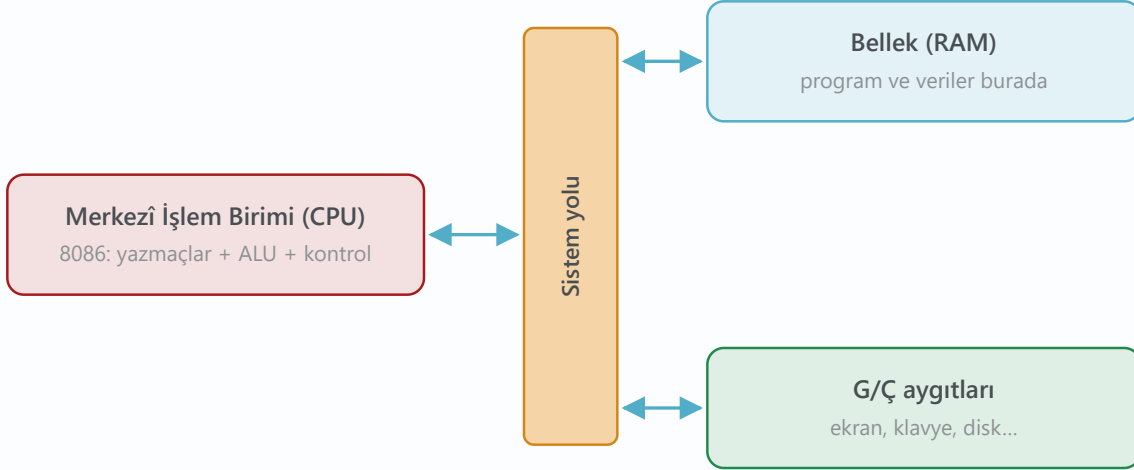
[etiket:] komut [hedef], [kaynak] [; yorum]



Satır	Etiket	Komut	İşlenenler	Yorum
basla: MOV AX, 5 ; AX ← 5	basla	MOV	AX, 5	; AX ← 5
INC AX	—	INC	AX	—
RET	—	RET	yok	—

- İki işlenenli komutlarda sıra **hedef, kaynaktır**: MOV AX, BX → AX ← BX.
- Assembler büyük-küçük harfe duyarlıdır: mov ax, bx ile MOV AX, BX aynıdır.
- ; işaretinden satır sonuna kadar her şey yorumdur ve makine koduna çevrilmez.

Basit Bilgisayar Modeli



Sistem yolu

Bilgisayarın bileşenlerini birbirine bağlar.

CPU

Bilgisayarın kalbi; hesaplamaların çoğu CPU içinde yapılır.

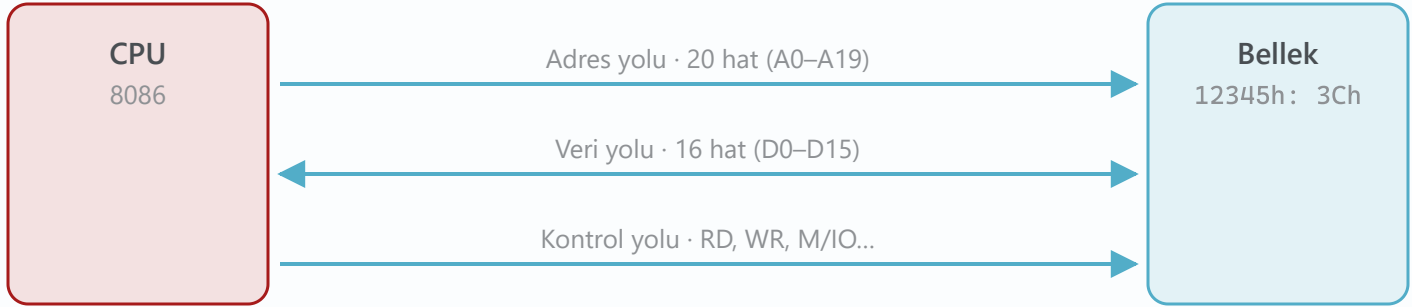
RAM

Programların yürütülmek üzere yüklendiği alan: komutlar ve veriler.

G/Ç aygıtları

Kullanıcıyla ve dış dünyayla iletişim.

Sistem Yolu



Adres yolu

CPU'dan belleğe/G/Ç'ye tek yönlü. 8086'da 20 hat → 2^{20} = **1 MB** adreslenebilir bellek.

Veri yolu

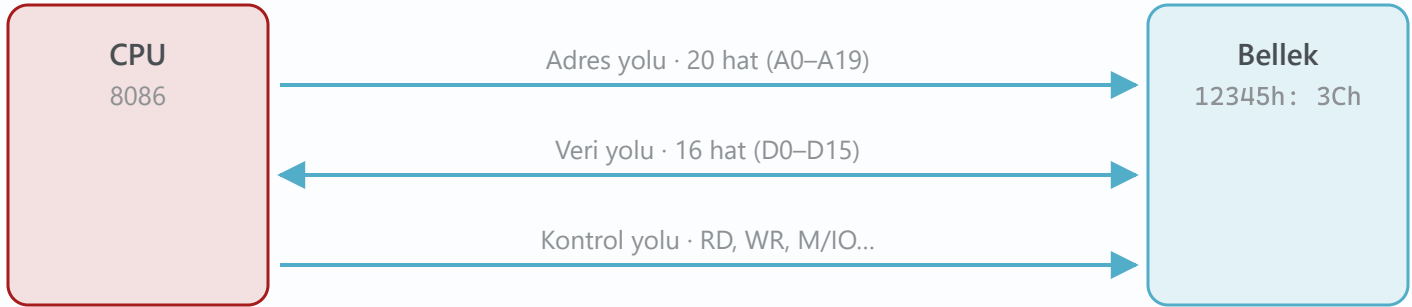
Çift yönlü. 8086'da 16 hat → tek seferde 1 bayt ya da 1 sözcük (word) aktarılır.

Kontrol yolu

İşlemin türünü bildirir: okuma (RD), yazma (WR), bellek mi G/Ç mı (M/IO), saat, kesme...

8086'da adres ve veri hatları aynı uçlardan zaman paylaşımı geçer (AD0–AD15).

Bellekten Okuma Adım Adım

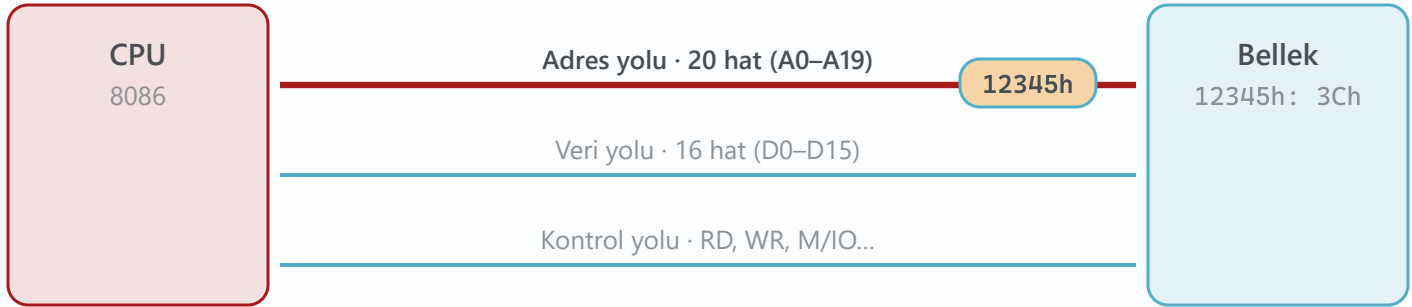


Komut: `MOV AL, [12345h fiziksel adresi]` . CPU'nun bir bayta ihtiyacı var; veri bellekte.

Adım 1 / 5

Yazma döngüsünde 3. adımda veriyi CPU koyar ve WR sinyali etkin olur.

Bellekten Okuma Adım Adım

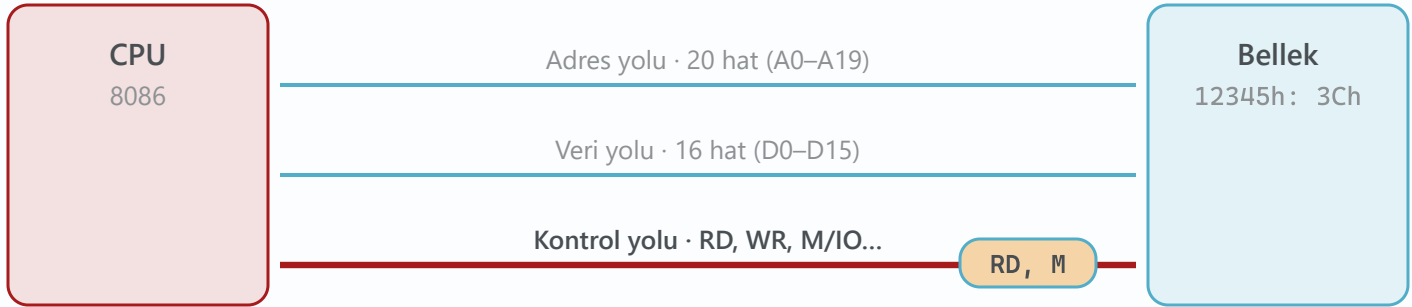


1. CPU, okunacak adresi **adres yoluna** koyar: 12345h.

Adım 2 / 5

Yazma döngüsünde 3. adımda veriyi CPU koyar ve WR sinyali etkin olur.

Bellekten Okuma Adım Adım



2. Kontrol yolu: “bellekten oku” (RD etkin, M/IO = bellek).

Adım 3 / 5

Yazma döngüsünde 3. adımda veriyi CPU koyar ve WR sinyali etkin olur.

Bellekten Okuma Adım Adım

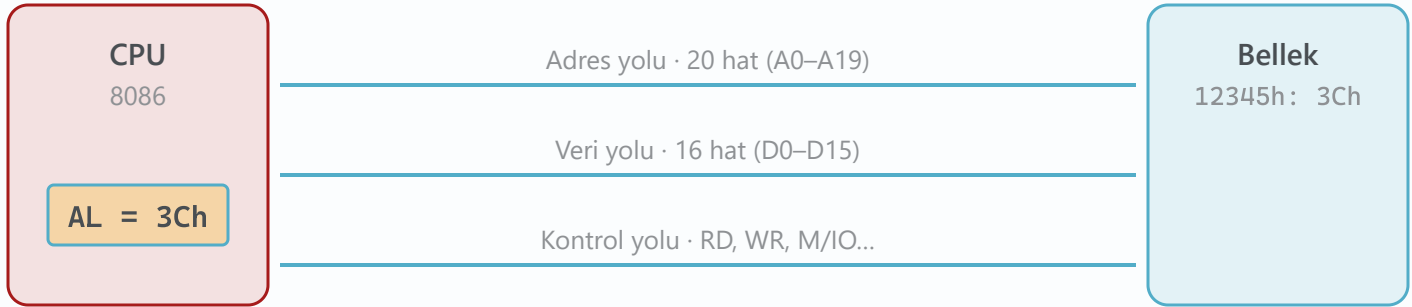


3. Bellek, 12345h'deki baytı (3Ch) **veri yoluna** koyar.

Adım 4 / 5

Yazma döngüsünde 3. adımda veriyi CPU koyar ve WR sinyali etkin olur.

Bellekten Okuma Adım Adım



4. CPU veriyi alır ve hedef yazmaca yazar: **AL = 3Ch**. 8086'da bu bir **yol döngüsü**dir (4 saat periyodu: T1–T4).

Adım 5 / 5

Yazma döngüsünde 3. adımda veriyi CPU koyar ve WR sinyali etkin olur.

Getir – Çöz – Yürüt

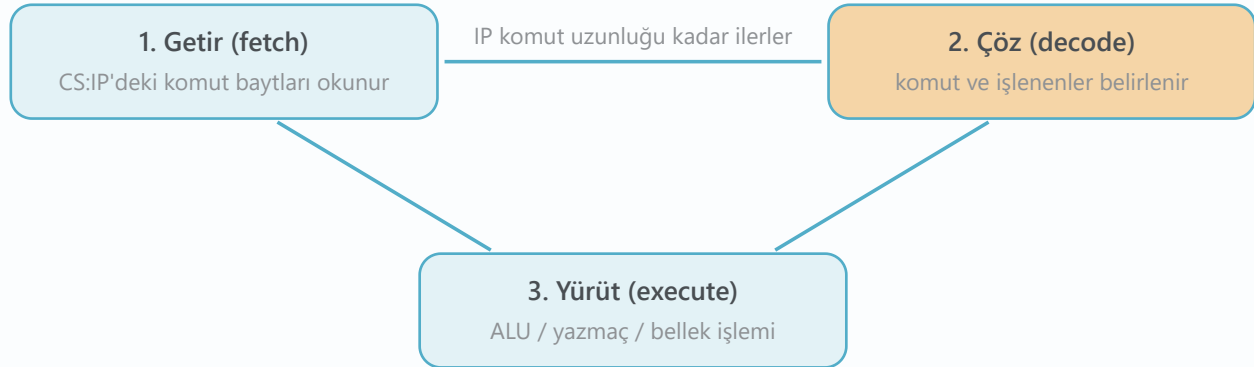


Getir: CPU, CS:IP adresindeki komut baytlarını bellekten okur ve **IP'yi komut uzunluğu kadar artırır.**

Adım 1 / 3

8086'da **BIU** (yol arayüz birimi) komutları önden getirip 6 baytlık **komut kuyruğuna** koyar; **EU** (yürütme birimi) bu sırada önceki komutu yürütür. Bu basit bir boru hattıdır (pipelining).

Getir – Çöz – Yürüt

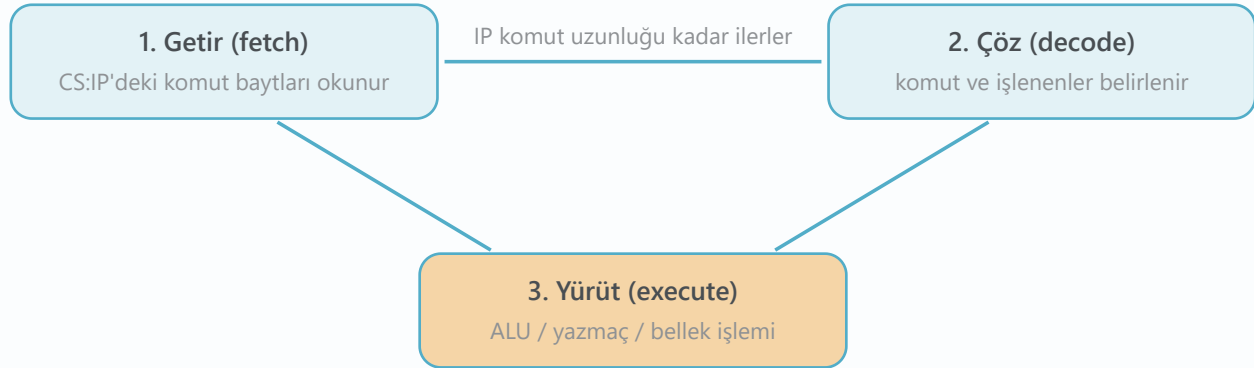


Çöz: İşlem kodu (opcode) çözülür: hangi işlem, hangi yazmaç/bellek işlenenleri?

Adım 2 / 3

8086'da **BIU** (yol arayüz birimi) komutları önden getirip 6 baytlık **komut kuyruğuna** koyar; **EU** (yürütme birimi) bu sırada önceki komutu yürütür. Bu basit bir boru hattıdır (pipelining).

Getir – Çöz – Yürüt



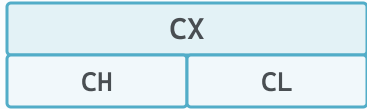
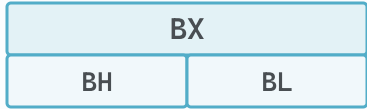
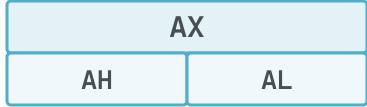
Yürüt: İşlem yapılır, sonuç yazılır, bayraklar güncellenir. Ardından döngü bir sonraki komutla tekrar başlar.

Adım 3 / 3

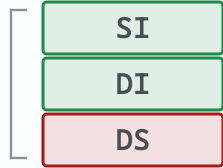
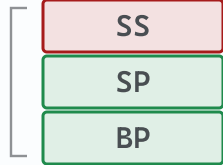
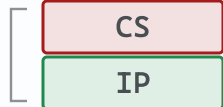
8086'da **BIU** (yol arayüz birimi) komutları önden getirip 6 baytlık **komut kuyruğuna** koyar; **EU** (yürütme birimi) bu sırada önceki komutu yürütür. Bu basit bir boru hattıdır (pipelining).

CPU'NUN İÇİ

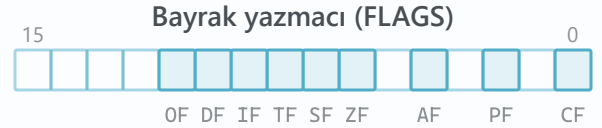
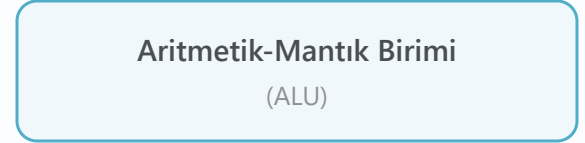
CPU'nun İçinde



genel amaçlı (16 = 8 + 8 bit)



kesim / işaretçi-indis



OF taşıma

DF yön

IF kesme

TF tuzak/adım

SF işaret

ZF sıfır

AF yardımcı elde

PF eşlik

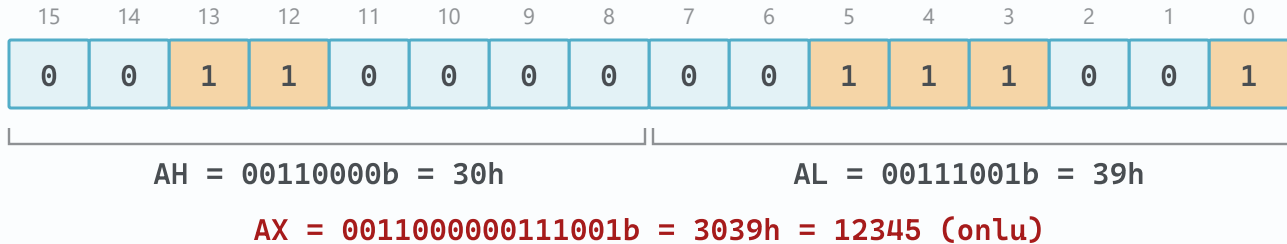
CF elde

9 bayrak kullanılır; diğer 7 bit ayrılmıştır

Mavi: genel amaçlı · kırmızı: kesim yazmaçları · yeşil: işaretçi/indis yazmaçları. Köşeli çizgiler birlikte çalışan çiftleri gösterir (CS:IP, SS:SP/BP, DS:SI/DI).

Genel Amaçlı Yazmaçlar

- 8086 CPU'da **8 adet** genel amaçlı yazmaç vardır: AX, BX, CX, DX, SI, DI, BP, SP.
- Her yazmacın bir adı ve tipik bir görevi vardır; ana amaç bir sayıyı (değişkeni) **geçici olarak saklamaktır**.
- Hepsi **16 bittir**. AX, BX, CX, DX ikişer adet **8 bitlik** yazmaçtan oluşur (üst: H, alt: L).
- CPU'nun içinde bulundukları için bellekten çok **daha hızlıdır**.



GENEL AMAÇLI YAZMAÇLAR

Yazmaçların Görevleri

Yazmaç	Adı	Tipik kullanımı
AX (AH, AL)	Akümülatör (Accumulator)	Aritmetik; MUL/DIV'in örtük işleneni; IN/OUT ile port verisi
BX (BH, BL)	Taban (Base)	Bellek adreslemede taban adres: [BX]
CX (CH, CL)	Sayaç (Count)	LOOP ve REP için sayaç; kaydırmalarda sayı CL'de
DX (DH, DL)	Veri (Data)	MUL/DIV'de sonucun üst yarısı; G/Ç port adresi
SI	Kaynak indis (Source Index)	Dizi/dizgi işlemlerinde kaynak adresi
DI	Hedef indis (Destination Index)	Dizi/dizgi işlemlerinde hedef adresi
BP	Taban işaretçisi (Base Pointer)	Yığındaki parametre ve yerel değişkenlere erişim
SP	Yığın işaretçisi (Stack Pointer)	Yığının tepesi; PUSH/POP/CALL/RET değiştirir

SI, DI, BP, SP'nin 8 bitlik yarıları yoktur. SP'yi genel amaçlı değişken gibi kullanmak yığını bozar.

Kod İzleme: AX, AH ve AL

► **ORG 100h**

```
2 MOV AX, 3039h ; AX = 3039h
3 MOV AL, 0FFh ; yalnızca AL
4 MOV AH, 12h ; yalnızca AH
5 INC AL ; FFh + 1
6 RET
```

AX 00 00
AH AL

Bayraklar

ZF=0

SF=0

PF=0

AF=0

CF=0

Başlangıç: AX = 0000h. **ORG 100h** makine koduna çevrilmez.

Adım 1 / 5

DEĞİŞKENLER

AX = 0000h

AH = 00h

AL = 00h

ZF = 0

SF = 0

PF = 0

AF = 0

CF = 0

Kod İzleme: AX, AH ve AL

1 **ORG** 100h

► **MOV** AX, 3039h ; AX = 3039h

3 **MOV** AL, 0FFh ; yalnızca AL

4 **MOV** AH, 12h ; yalnızca AH

5 **INC** AL ; FFh + 1

6 **RET**

AX

30	39
AH	AL

Bayraklar

ZF=0

SF=0

PF=0

AF=0

CF=0

MOV AX, 3039h : 16 bitin tamamı yazılır → AH = 30h, AL = 39h.

Adım 2 / 5

DEĞİŞKENLER

AX = 3039h

AH = 30h

AL = 39h

ZF = 0

SF = 0

PF = 0

AF = 0

CF = 0

Kod İzleme: AX, AH ve AL

```
1 ORG 100h
2 MOV AX, 3039h ; AX = 3039h
3 MOV AL, 0FFh ; yalnızca AL
4 MOV AH, 12h ; yalnızca AH
5 INC AL ; FFh + 1
6 RET
```

AX

30	FF
AH	AL

Bayraklar

ZF=0

SF=0

PF=0

AF=0

CF=0

MOV AL, 0FFh : yalnızca düşük bayt değişir → AX = **30FFh**.

Adım 3 / 5

DEĞİŞKENLER

AX = 30FFh

AH = 30h

AL = FFh

ZF = 0

SF = 0

PF = 0

AF = 0

CF = 0

Kod İzleme: AX, AH ve AL

```
1 ORG 100h
2 MOV AX, 3039h    ; AX = 3039h
3 MOV AL, 0FFh     ; yalnızca AL
4 MOV AH, 12h      ; yalnızca AH
5 INC AL           ; FFh + 1
6 RET
```

AX **12** **FF**
 AH AL

Bayraklar

ZF=0

SF=0

PF=0

AF=0

CF=0

MOV AH, 12h : yalnızca yüksek bayt değişir → AX = **12FFh**.
MOV bayrakları etkilemez.

Adım 4 / 5

DEĞİŞKENLER

AX = 12FFh

AH = 12h

AL = FFh

ZF = 0

SF = 0

PF = 0

AF = 0

CF = 0

Kod İzleme: AX, AH ve AL

```
1 ORG 100h
2 MOV AX, 3039h ; AX = 3039h
3 MOV AL, 0FFh ; yalnızca AL
4 MOV AH, 12h ; yalnızca AH
▶ INC AL ; FFh + 1
6 RET
```

AX

12	00
AH	AL

Bayraklar

ZF=1

SF=0

PF=1

AF=1

CF=0

INC AL : FFh + 1 = 00h (8 bit). Elde **AH'ye geçmez**: AX = **1200h**. ZF = 1; INC, CF'yi değiştirmez.

Adım 5 / 5

DEĞİŞKENLER

AX = 1200h

AH = 12h

AL = 00h

ZF = 1

SF = 0

PF = 1

AF = 1

CF = 0

AX Yazmacı Oyun Alanı

AX'e bir değer yazın; yüksek ve düşük baytları, ikili gösterimi ve onluk karşılıkları görün.

- AH ve AL ayrı yazmaçlar gibi kullanılabilir, ama **aynı** 16 bitin parçalarıdır.
- İşaretli yorumda bit 15 işaret bitidir (ikiye tümleyen).
- BX/BH/BL, CX/CH/CL, DX/DH/DL aynı şekilde çalışır.

AX = **3039** h

15	14	13	12	11	10	9	8
0	0	1	1	0	0	0	0
7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	1

AX = **3039h** · AH = **30h** · AL = **39h**
işaretsiz 12345 · işaretli 12345

Turuncu: AH (bit 15–8) · mavi: AL (bit 7–0)

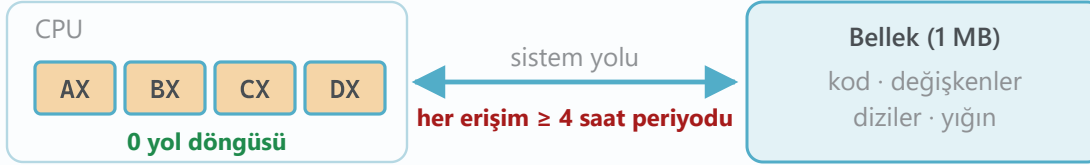
Yazmaç mı, Bellek mi?

Yazmaç

- CPU'nun içindedir; erişim için yol döngüsü gerekmez.
- Sayısı çok azdır (8086'da 14 adet 16 bitlik yazmaç).
- Ara sonuçlar ve sayacılar için idealdir.

Bellek

- CPU'nun dışındadır; her erişim en az bir yol döngüsü (8086'da 4 saat periyodu) sürer.
- 1 MB'a kadar büyüktür.
- Program kodu, değişkenler, diziler ve yığın buradadır.



İyi assembly kodu sık kullanılan değerleri **yazmaçlarda** tutar, belleğe yalnızca gerektiğinde gider. Örneğin bir döngü sayacını bellekte değil CX 'te saklamak döngüyü hızlandırır.

14 yazmaç: 8 genel amaçlı + 4 kesim + IP + FLAGS.

KESİM YAZMAÇLARI

Kesim Yazmaçları

Bellekteki erişilebilir **blokları** (kesim, segment) işaret etmek için kullanılır. Dört kesim yazmacı vardır:

CS · Code Segment

Geçerli programın (komutların) bulunduğu kesim. IP ile birlikte çalışır.

DS · Data Segment

Değişkenlerin tanımlandığı kesim. Bellek işlenenlerinin varsayılanı.

ES · Extra Segment

Kullanımı programcıya bağlı ek kesim; dizgi komutlarında hedef (ES:DI).

SS · Stack Segment

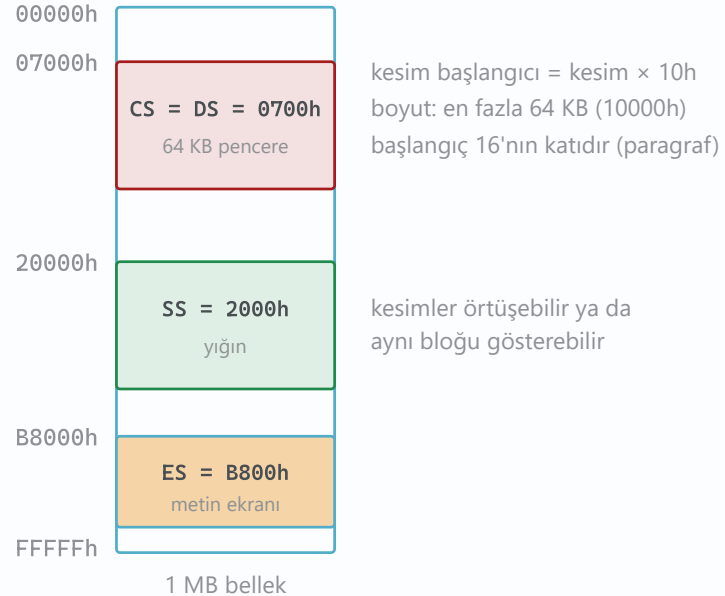
Yığını içeren kesim. SP ve BP ile birlikte çalışır.



Neden kesim? 8086'nın yazmaçları 16 bittir (en fazla 64 KB adresler) ama adres yolu 20 bittir (1 MB). İki 16 bitlik değer birleştirilerek 20 bitlik adres elde edilir.

KESİM YAZMAÇLARI

1 MB Bellek ve 64 KB Kesimler



Fiziksel Adres Hesabı

DS

1

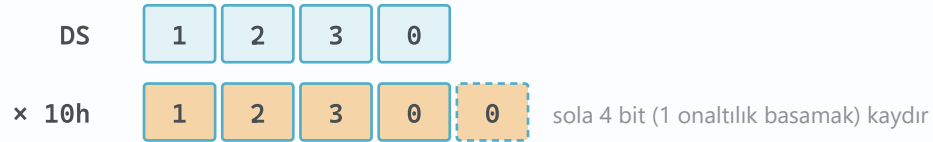
2

3

0

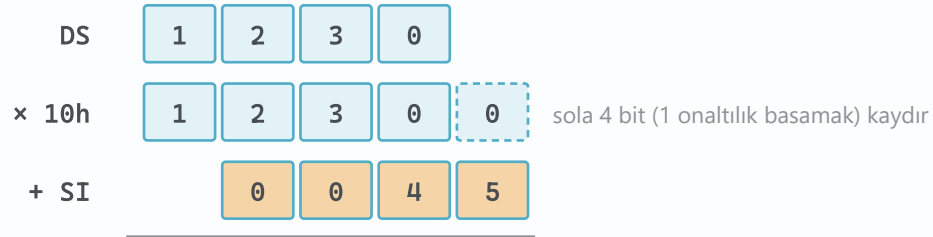
Fiziksel adresi **12345h** olan bayta erişmek için örneğin **DS = 1230h** ve **SI = 0045h** atanır.

Fiziksel Adres Hesabı



Kesim değeri **10h (16) ile çarpılır**: onaltılıkta sağa bir 0 eklemek, ikilide 4 bit sola kaydırmaktır → 12300h.

Fiziksel Adres Hesabı



Bağıl konum (offset) eklenmek üzere hizalanır: **0045h**.

Fiziksel Adres Hesabı



12300h + 0045h = 12345h. Gösterim: **1230:0045** (kesim:bağlı konum).

Aynı Adres, Farklı Çiftler

Kesim:Bağıl konum	Hesap	Fiziksel
1230:0045	12300 + 0045	12345h
1234:0005	12340 + 0005	12345h
1200:0345	12000 + 0345	12345h
1000:2345	10000 + 2345	12345h
0235:FFF5	02350 + FFF5	12345h

- Kesimler 16 baytta bir başlar, 64 KB uzunluktadır → **örtüşürler**.
- Bu nedenle bir fiziksel adrese 4096'ya kadar farklı kesim:bağıl konum çifti karşılık gelebilir.
- Normalize** biçimde bağıl konum 0–Fh arasındır: 1234:0005 .
- Toplam 20 biti aşarsa 8086'da adres **başa sarar**: FFFF:0010 → 00000h.

Kod İzleme: Üç Çift, Tek Bayt

```

1  ORG 100h
▶ MOV AX, 1230h
3  MOV DS, AX
4  MOV SI, 0045h
5  MOV BYTE PTR [SI], 7Ah
6  MOV AX, 1234h
7  MOV DS, AX
8  MOV BX, 0005h
9  MOV CL, [BX]      ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI]   ; 1000:2345
14 RET

```

fiziksel adres = kesim × 10h + bağıl konum

DS × 10h	?????h
+ SI	????h
= fiziksel	?

bellek: 12345h

00h

bulunan çiftler

AX = 1230h. Kesim yazmacına sabit doğrudan yazılamadığı için değer önce AX'e alınır.

Adım 1 / 12

DEĞİŞKENLER

AX = 1230h

DS = 0700h

ES = 0700h

SI = 0000h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```
1 ORG 100h
2 MOV AX, 1230h
3 MOV DS, AX
4 MOV SI, 0045h
5 MOV BYTE PTR [SI], 7Ah
6 MOV AX, 1234h
7 MOV DS, AX
8 MOV BX, 0005h
9 MOV CL, [BX] ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI] ; 1000:2345
14 RET
```

fiziksel adres = kesim × 10h + bağıl konum

DS × 10h	12300h
+ SI	????h
= fiziksel	?

bellek: 12345h

00h

bulunan çiftler

DS = **1230h**: kesim 12300h fiziksel adresinden başlar.

Adım 2 / 12

DEĞİŞKENLER

AX = 1230h

DS = 1230h

ES = 0700h

SI = 0000h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```
1 ORG 100h
2 MOV AX, 1230h
3 MOV DS, AX
4 MOV SI, 0045h
5 MOV BYTE PTR [SI], 7Ah
6 MOV AX, 1234h
7 MOV DS, AX
8 MOV BX, 0005h
9 MOV CL, [BX] ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI] ; 1000:2345
14 RET
```

fiziksel adres = kesim × 10h + bağıl konum

DS × 10h	12300h
+ SI	0045h
= fiziksel	12345h

bellek: 12345h

00h

bulunan çiftler

SI = 0045h: kesim içindeki bağıl konum.

Adım 3 / 12

DEĞİŞKENLER

AX = 1230h

DS = 1230h

ES = 0700h

SI = 0045h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```
1 ORG 100h
2 MOV AX, 1230h
3 MOV DS, AX
4 MOV SI, 0045h
5 ► MOV BYTE PTR [SI], 7Ah
6 MOV AX, 1234h
7 MOV DS, AX
8 MOV BX, 0005h
9 MOV CL, [BX] ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI] ; 1000:2345
14 RET
```

fiziksel adres = kesim × 10h + bağıl konum

DS × 10h	12300h
+ SI	0045h
= fiziksel	12345h

bellek: 12345h

7Ah

bulunan çiftler

1230:0045
→ 12345h

1230h × 10h + 0045h = **12345h**. Bu bayta **7Ah** yazıldı.

Adım 4 / 12

DEĞİŞKENLER

AX = 1230h

DS = 1230h

ES = 0700h

SI = 0045h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```

1 ORG 100h
2 MOV AX, 1230h
3 MOV DS, AX
4 MOV SI, 0045h
5 MOV BYTE PTR [SI], 7Ah
6 MOV AX, 1234h
7 MOV DS, AX
8 MOV BX, 0005h
9 MOV CL, [BX] ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI] ; 1000:2345
14 RET

```

fiziksel adres = kesim × 10h + bağıl konum

DS × 10h	?????h
+ BX	????h
= fiziksel	?

bellek: 12345h

7Ah

bulunan çiftler

1230:0045
→ 12345h

AX = 1234h: ikinci kesim değeri.

Adım 5 / 12

DEĞİŞKENLER

AX = 1234h

DS = 1230h

ES = 0700h

BX = 0000h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```

1  ORG 100h
2  MOV AX, 1230h
3  MOV DS, AX
4  MOV SI, 0045h
5  MOV BYTE PTR [SI], 7Ah
6  MOV AX, 1234h
7  ► MOV DS, AX
8  MOV BX, 0005h
9  MOV CL, [BX]      ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI]   ; 1000:2345
14 RET

```

fiziksel adres = kesim × 10h + bağıl konum

DS × 10h	12340h
+ BX	????h
= fiziksel	?

bellek: 12345h

7Ah

bulunan çiftler

1230:0045

→ 12345h

DS = **1234h**: kesim 12340h'den başlar; öncekinden yalnızca 4 bayt ileride.

Adım 6 / 12

DEĞİŞKENLER

AX = 1234h

DS = 1234h

ES = 0700h

BX = 0000h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```
1 ORG 100h
2 MOV AX, 1230h
3 MOV DS, AX
4 MOV SI, 0045h
5 MOV BYTE PTR [SI], 7Ah
6 MOV AX, 1234h
7 MOV DS, AX
▶ MOV BX, 0005h
9 MOV CL, [BX] ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI] ; 1000:2345
14 RET
```

fiziksel adres = kesim × 10h + bağlı konum

DS × 10h	12340h
+ BX	0005h
= fiziksel	12345h

bellek: 12345h

7Ah

bulunan çiftler

1230:0045
→ 12345h

BX = 0005h.

Adım 7 / 12

DEĞİŞKENLER

AX = 1234h

DS = 1234h

ES = 0700h

BX = 0005h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağlı konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```

1  ORG 100h
2  MOV AX, 1230h
3  MOV DS, AX
4  MOV SI, 0045h
5  MOV BYTE PTR [SI], 7Ah
6  MOV AX, 1234h
7  MOV DS, AX
8  MOV BX, 0005h
9  ► MOV CL, [BX] ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI] ; 1000:2345
14 RET

```

fiziksel adres = kesim × 10h + bağıl konum

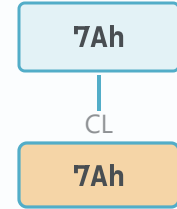
DS × 10h	12340h
+ BX	0005h
= fiziksel	12345h

bulunan çiftler

1230:0045	1234:0005
→ 12345h	→ 12345h

1234h × 10h + 0005h = **12345h**: aynı bayt! CL = **7Ah**.

bellek: 12345h



Adım 8 / 12

DEĞİŞKENLER

AX = **1234h**

DS = **1234h**

ES = **0700h**

BX = **0005h**

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```

1 ORG 100h
2 MOV AX, 1230h
3 MOV DS, AX
4 MOV SI, 0045h
5 MOV BYTE PTR [SI], 7Ah
6 MOV AX, 1234h
7 MOV DS, AX
8 MOV BX, 0005h
9 MOV CL, [BX] ; 1234:0005
▶ MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI] ; 1000:2345
14 RET

```

fiziksel adres = kesim × 10h + bağıl konum

ES × 10h	?????h
+ DI	????h
= fiziksel	?

bellek: 12345h

7Ah

bulunan çiftler

1230:0045	1234:0005
→ 12345h	→ 12345h

AX = 1000h.

Adım 9 / 12

DEĞİŞKENLER

AX = 1000h

DS = 1234h

ES = 0700h

DI = 0000h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```

1  ORG 100h
2  MOV AX, 1230h
3  MOV DS, AX
4  MOV SI, 0045h
5  MOV BYTE PTR [SI], 7Ah
6  MOV AX, 1234h
7  MOV DS, AX
8  MOV BX, 0005h
9  MOV CL, [BX]      ; 1234:0005
10 MOV AX, 1000h
    ► MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI]   ; 1000:2345
14 RET

```

fiziksel adres = kesim × 10h + bağlı konum

ES × 10h	10000h
+ DI	????h
= fiziksel	?

bellek: 12345h

7Ah

bulunan çiftler

1230:0045	1234:0005
→ 12345h	→ 12345h

ES = **1000h**. Bu kez DS'ye dokunmadan ek kesim kullanılacak.

Adım 10 / 12

DEĞİŞKENLER

AX = 1000h

DS = 1234h

ES = 1000h

DI = 0000h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağlı konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```
1 ORG 100h
2 MOV AX, 1230h
3 MOV DS, AX
4 MOV SI, 0045h
5 MOV BYTE PTR [SI], 7Ah
6 MOV AX, 1234h
7 MOV DS, AX
8 MOV BX, 0005h
9 MOV CL, [BX] ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI] ; 1000:2345
14 RET
```

fiziksel adres = kesim × 10h + bağıl konum

ES × 10h	10000h
+ DI	2345h
= fiziksel	12345h

bellek: 12345h

7Ah

bulunan çiftler

1230:0045	1234:0005
→ 12345h	→ 12345h

DI = 2345h.

Adım 11 / 12

DEĞİŞKENLER

AX = 1000h

DS = 1234h

ES = 1000h

DI = 2345h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Kod İzleme: Üç Çift, Tek Bayt

```

1  ORG 100h
2  MOV AX, 1230h
3  MOV DS, AX
4  MOV SI, 0045h
5  MOV BYTE PTR [SI], 7Ah
6  MOV AX, 1234h
7  MOV DS, AX
8  MOV BX, 0005h
9  MOV CL, [BX]      ; 1234:0005
10 MOV AX, 1000h
11 MOV ES, AX
12 MOV DI, 2345h
13 MOV DL, ES:[DI]   ; 1000:2345
14 RET

```

fiziksel adres = kesim × 10h + bağıl konum

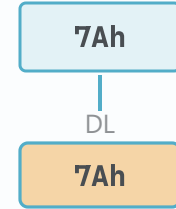
ES × 10h	10000h
+ DI	2345h
= fiziksel	12345h

bulunan çiftler

1230:0045	1234:0005	1000:2345
→ 12345h	→ 12345h	→ 12345h

ES:[DI] : 10000h + 2345h = **12345h** → DL = **7Ah**. Üç farklı kesim:bağıl konum çifti, tek fiziksel adres.

bellek: 12345h



Adım 12 / 12

DEĞİŞKENLER

AX = 1000h

DS = 1234h

ES = 1000h

DI = 2345h

Kesimler 16 baytta bir başlayıp örtüştüğü için aynı fiziksel bayta pek çok kesim:bağıl konum çiftiyle ulaşılır.

Fiziksel Adres Hesaplayıcı

Kesim h : Bağlı konum h

$$1230h \times 10h = 12300h$$

$$12300h + 0045h = \mathbf{12345h}$$

Normalize: **1234:0005**

B800:015E

1234:7890

0700:0100

FFFF:0010

Onaltılık en fazla 4 basamak girin. Hesap: fiziksel = kesim $\times$ 10h + bağlı konum (20 bit).

Varsayılan Kesim Eşleşmeleri

Bağlı konumu veren	Varsayılan kesim
IP (sıradaki komut)	CS
SP (yığın tepesi)	SS
BP içeren adresleme	SS
BX , SI , DI , sabit adres	DS
Dizgi komutlarında DI	ES

- Kesim ve genel amaçlı yazmaçlar birlikte kullanılarak bellekteki herhangi bir değere erişilir.
- Köşeli parantez içinde yazmaçlardan hesaplanan bağlı konuma **etkin adres** (EA) denir; fiziksel adres = kesim × 10h + EA.
- Adreslemede yalnızca **BX, SI, DI, BP** kullanılabilir; [BX] geçerli, [BL] , [BH] , [AX] geçersiz.
- Varsayılan değiştirilebilir: MOV AL, ES: [BX] (kesim önekli erişim).

Kod İzleme: [BX], [BP] ve ES:[BX]

```
1 ORG 100h
▶ MOV AX, 2000h
3 MOV DS, AX
4 MOV AX, 3000h
5 MOV ES, AX
6 MOV BX, 0010h
7 MOV BP, BX      ; aynı ofset
8 MOV BYTE PTR [BX], 11h
9 MOV BYTE PTR [BP], 22h
10 MOV BYTE PTR ES:[BX], 33h
11 RET
```

DS = 0700h

taban 07000h

+0010h = 07010h

00h

SS = 0700h

taban 07000h

+0010h = 07010h

00h

ES = 0700h

taban 07000h

+0010h = 07010h

00h

BX = 0000h BP = 0000h

AX = 2000h. .COM programında başlangıçta CS = DS = ES = SS = 0700h.

Adım 1 / 9

DEĞİŞKENLER

DS = 0700h

ES = 0700h

SS = 0700h

BX = 0000h

BP = 0000h

Bağıl konum aynı olsa da kesim farklıysa erişilen bayt farklıdır: BP içeren adres SS, BX/SI/DI içeren adres DS kullanır.

Kod İzleme: [BX], [BP] ve ES:[BX]

```
1 ORG 100h
2 MOV AX, 2000h
3 ► MOV DS, AX
4 MOV AX, 3000h
5 MOV ES, AX
6 MOV BX, 0010h
7 MOV BP, BX      ; aynı ofset
8 MOV BYTE PTR [BX], 11h
9 MOV BYTE PTR [BP], 22h
10 MOV BYTE PTR ES:[BX], 33h
11 RET
```

DS = 2000h

taban 20000h

+0010h = 20010h

00h

SS = 0700h

taban 07000h

+0010h = 07010h

00h

ES = 0700h

taban 07000h

+0010h = 07010h

00h

BX = 0000h BP = 0000h

DS = **2000h**: veri kesimi artık 20000h'de.

Adım 2 / 9

DEĞİŞKENLER

DS = **2000h**

ES = 0700h

SS = 0700h

BX = 0000h

BP = 0000h

Bağıl konum aynı olsa da kesim farklıysa erişilen bayt farklıdır: BP içeren adres SS, BX/SI/DI içeren adres DS kullanır.

Kod İzleme: [BX], [BP] ve ES:[BX]

```
1 ORG 100h
2 MOV AX, 2000h
3 MOV DS, AX
4 MOV AX, 3000h
5 MOV ES, AX
6 MOV BX, 0010h
7 MOV BP, BX ; aynı ofset
8 MOV BYTE PTR [BX], 11h
9 MOV BYTE PTR [BP], 22h
10 MOV BYTE PTR ES:[BX], 33h
11 RET
```

DS = 2000h

taban 20000h

+0010h = 20010h

00h

SS = 0700h

taban 07000h

+0010h = 07010h

00h

ES = 0700h

taban 07000h

+0010h = 07010h

00h

BX = 0000h BP = 0000h

AX = 3000h.

Adım 3 / 9

DEĞİŞKENLER

DS = 2000h

ES = 0700h

SS = 0700h

BX = 0000h

BP = 0000h

Bağıl konum aynı olsa da kesim farklıysa erişilen bayt farklıdır: BP içeren adres SS, BX/SI/DI içeren adres DS kullanır.

Kod İzleme: [BX], [BP] ve ES:[BX]

```
1 ORG 100h
2 MOV AX, 2000h
3 MOV DS, AX
4 MOV AX, 3000h
5 MOV ES, AX
6 MOV BX, 0010h
7 MOV BP, BX ; aynı ofset
8 MOV BYTE PTR [BX], 11h
9 MOV BYTE PTR [BP], 22h
10 MOV BYTE PTR ES:[BX], 33h
11 RET
```

DS = 2000h

taban 20000h

+0010h = 20010h

00h

SS = 0700h

taban 07000h

+0010h = 07010h

00h

ES = 3000h

taban 30000h

+0010h = 30010h

00h

BX = 0000h BP = 0000h

ES = **3000h**. SS değişmedi: 0700h.

Adım 4 / 9

DEĞİŞKENLER

DS = 2000h

ES = **3000h**

SS = 0700h

BX = 0000h

BP = 0000h

Bağıl konum aynı olsa da kesim farklıysa erişilen bayt farklıdır: BP içeren adres SS, BX/SI/DI içeren adres DS kullanır.

Kod İzleme: [BX], [BP] ve ES:[BX]

```
1 ORG 100h
2 MOV AX, 2000h
3 MOV DS, AX
4 MOV AX, 3000h
5 MOV ES, AX
6 MOV BX, 0010h
7 MOV BP, BX ; aynı ofset
8 MOV BYTE PTR [BX], 11h
9 MOV BYTE PTR [BP], 22h
10 MOV BYTE PTR ES:[BX], 33h
11 RET
```

DS = 2000h

taban 20000h

+0010h = 20010h

00h

SS = 0700h

taban 07000h

+0010h = 07010h

00h

ES = 3000h

taban 30000h

+0010h = 30010h

00h

BX = 0010h BP = 0000h

BX = 0010h.

Adım 5 / 9

DEĞİŞKENLER

DS = 2000h

ES = 3000h

SS = 0700h

BX = 0010h

BP = 0000h

Bağıl konum aynı olsa da kesim farklıysa erişilen bayt farklıdır: BP içeren adres SS, BX/SI/DI içeren adres DS kullanır.

Kod İzleme: [BX], [BP] ve ES:[BX]

```
1 ORG 100h
2 MOV AX, 2000h
3 MOV DS, AX
4 MOV AX, 3000h
5 MOV ES, AX
6 MOV BX, 0010h
7 ► MOV BP, BX ; aynı ofset
8 MOV BYTE PTR [BX], 11h
9 MOV BYTE PTR [BP], 22h
10 MOV BYTE PTR ES:[BX], 33h
11 RET
```

DS = 2000h

taban 20000h

+0010h = 20010h

00h

SS = 0700h

taban 07000h

+0010h = 07010h

00h

ES = 3000h

taban 30000h

+0010h = 30010h

00h

BX = 0010h BP = 0010h

BP = BX = 0010h. İki yazmaç aynı bağlı konumu tutuyor.

Adım 6 / 9

DEĞİŞKENLER

DS = 2000h

ES = 3000h

SS = 0700h

BX = 0010h

BP = 0010h

Bağıl konum aynı olsa da kesim farklıysa erişilen bayt farklıdır: BP içeren adres SS, BX/SI/DI içeren adres DS kullanır.

Kod İzleme: [BX], [BP] ve ES:[BX]

```
1 ORG 100h
2 MOV AX, 2000h
3 MOV DS, AX
4 MOV AX, 3000h
5 MOV ES, AX
6 MOV BX, 0010h
7 MOV BP, BX ; aynı ofset
8 ► MOV BYTE PTR [BX], 11h
9 MOV BYTE PTR [BP], 22h
10 MOV BYTE PTR ES:[BX], 33h
11 RET
```

DS = 2000h
taban 20000h
+0010h = 20010h
11h

BX = 0010h BP = 0010h

SS = 0700h
taban 07000h
+0010h = 07010h
00h

ES = 3000h
taban 30000h
+0010h = 30010h
00h

[BX] → DS

[BX] → varsayılan kesim **DS**: 20000h + 10h = **20010h** ← 11h.

Adım 7 / 9

DEĞİŞKENLER

DS = 2000h

ES = 3000h

SS = 0700h

BX = 0010h

BP = 0010h

Bağıl konum aynı olsa da kesim farklıysa erişilen bayt farklıdır: BP içeren adres SS, BX/SI/DI içeren adres DS kullanır.

Kod İzleme: [BX], [BP] ve ES:[BX]

```
1 ORG 100h
2 MOV AX, 2000h
3 MOV DS, AX
4 MOV AX, 3000h
5 MOV ES, AX
6 MOV BX, 0010h
7 MOV BP, BX      ; aynı ofset
8 MOV BYTE PTR [BX], 11h
9 MOV BYTE PTR [BP], 22h
10 MOV BYTE PTR ES:[BX], 33h
11 RET
```

DS = 2000h

taban 20000h

+0010h = 20010h

11h

BX = 0010h BP = 0010h

SS = 0700h

taban 07000h

+0010h = 07010h

22h

[BP] → SS

ES = 3000h

taban 30000h

+0010h = 30010h

00h

[BP] → varsayılan kesim **SS**: 07000h + 10h = **07010h** ← 22h.
Aynı ofset, başka bayt!

Adım 8 / 9

DEĞİŞKENLER

DS = 2000h

ES = 3000h

SS = 0700h

BX = 0010h

BP = 0010h

Bağıl konum aynı olsa da kesim farklıysa erişilen bayt farklıdır: BP içeren adres SS, BX/SI/DI içeren adres DS kullanır.

Kod İzleme: [BX], [BP] ve ES:[BX]

```
1 ORG 100h
2 MOV AX, 2000h
3 MOV DS, AX
4 MOV AX, 3000h
5 MOV ES, AX
6 MOV BX, 0010h
7 MOV BP, BX ; aynı ofset
8 MOV BYTE PTR [BX], 11h
9 MOV BYTE PTR [BP], 22h
▶ MOV BYTE PTR ES:[BX], 33h
11 RET
```

DS = 2000h

taban 20000h

+0010h = 20010h

11h

SS = 0700h

taban 07000h

+0010h = 07010h

22h

ES = 3000h

taban 30000h

+0010h = 30010h

33h

ES:[BX] → ES

BX = 0010h BP = 0010h

ES: önceki varsayılanı değiştirir: 30000h + 10h = **30010h** ← 33h.

Adım 9 / 9

DEĞİŞKENLER

DS = 2000h

ES = 3000h

SS = 0700h

BX = 0010h

BP = 0010h

Bağıl konum aynı olsa da kesim farklıysa erişilen bayt farklıdır: BP içeren adres SS, BX/SI/DI içeren adres DS kullanır.

Özel Amaçlı Yazmaçlar

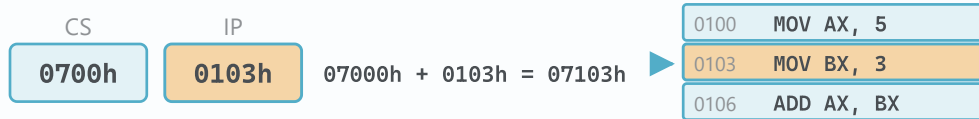
Mikroişlemcinin belirli görevleri için ayrılmıştır; genel amaçlı yazmaçlar gibi **MOV ile doğrudan** erişilemez.

IP · Instruction Pointer

- CS ile birlikte çalışır: **CS:IP**, **getirilecek bir sonraki komutu** gösterir.
- Diğer işlemcilerde *Program Counter* (PC) olarak da bilinir.
- Her komut getirildiğinde komut uzunluğu kadar artar.
- Yalnızca JMP, CALL, RET, INT, IRET ve koşullu atlamalar değiştirir.

FLAGS · Bayrak yazmacı

- İşlemlerden sonra CPU tarafından **otomatik** güncellenir.
- Koşullu atlamalarla programın akışını (kontrolü) diğer bölümlere aktarmak için kullanılır.
- Sonucun durumunu gösteren bitlere **durum bayrakları**, CPU'yu yönlendiren bitlere **kontrol bayrakları** denir.

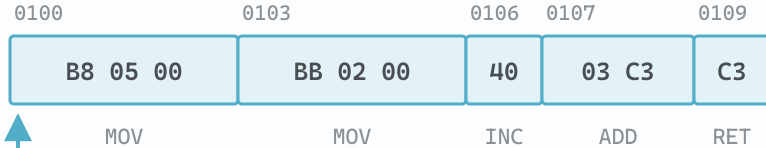


Not: 8086'da IP, o anda yürütülen komutu değil, **sıradaki** komutu gösterir (komut kuyruğu nedeniyle daha da ileride okuma yapılabilir).

Kod İzleme: IP Nasıl İlerler?

► **ORG 100h**

```
2 MOV AX, 5      ; 3 bayt
3 MOV BX, 2      ; 3 bayt
4 INC AX         ; 1 bayt
5 ADD AX, BX     ; 2 bayt
6 RET           ; 1 bayt
```



IP = 0100h

Program CS:0100h'ye yüklenir; **IP = 0100h** ilk komutu gösterir.

Adım 1 / 6

DEĞİŞKENLER

IP = 0100h

AX = 0000h

BX = 0000h

SP = FFEh

Kod İzleme: IP Nasıl İlerler?

1 **ORG** 100h

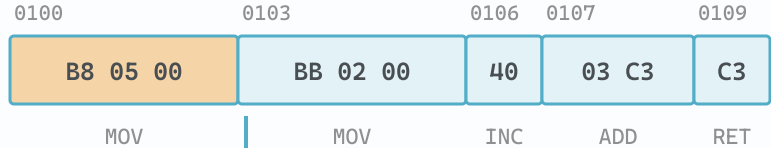
► **MOV** AX, 5 ; 3 bayt

3 **MOV** BX, 2 ; 3 bayt

4 **INC** AX ; 1 bayt

5 **ADD** AX, BX ; 2 bayt

6 **RET** ; 1 bayt



IP = 0103h

MOV AX, 5 (B8 05 00) getirilir: **IP = 0103h**, yürütülünce AX = 0005h.

Adım 2 / 6

DEĞİŞKENLER

IP = 0103h

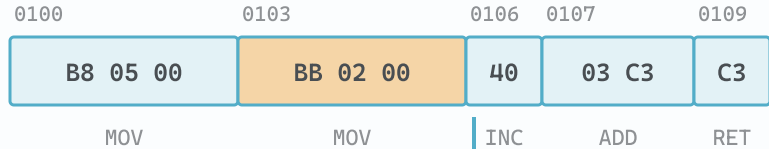
AX = 0005h

BX = 0000h

SP = FFEh

Kod İzleme: IP Nasıl İlerler?

```
1 ORG 100h
2 MOV AX, 5      ; 3 bayt
3 ► MOV BX, 2     ; 3 bayt
4 INC AX          ; 1 bayt
5 ADD AX, BX      ; 2 bayt
6 RET            ; 1 bayt
```



IP = 0106h

3 bayt daha: **IP = 0106h**, BX = 0002h.

Adım 3 / 6

DEĞİŞKENLER

IP = 0106h

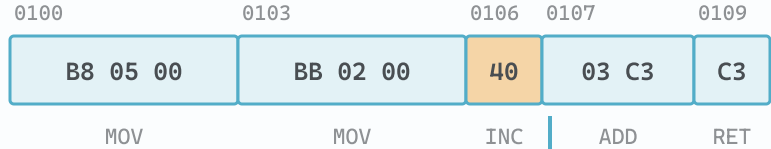
AX = 0005h

BX = 0002h

SP = FFEh

Kod İzleme: IP Nasıl İlerler?

```
1 ORG 100h
2 MOV AX, 5      ; 3 bayt
3 MOV BX, 2      ; 3 bayt
4 INC AX         ; 1 bayt
5 ADD AX, BX     ; 2 bayt
6 RET           ; 1 bayt
```



IP = 0107h

INC AX tek bayttır (40h): **IP = 0107h**, AX = 0006h.

Adım 4 / 6

DEĞİŞKENLER

IP = 0107h

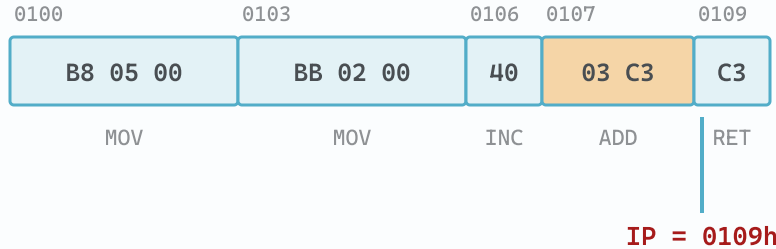
AX = 0006h

BX = 0002h

SP = FFEh

Kod İzleme: IP Nasıl İlerler?

```
1 ORG 100h
2 MOV AX, 5      ; 3 bayt
3 MOV BX, 2      ; 3 bayt
4 INC AX         ; 1 bayt
5 ADD AX, BX     ; 2 bayt
6 RET           ; 1 bayt
```



ADD AX, BX 2 bayt: **IP = 0109h**, AX = 0008h.

Adım 5 / 6

DEĞİŞKENLER

IP = 0109h

AX = 0008h

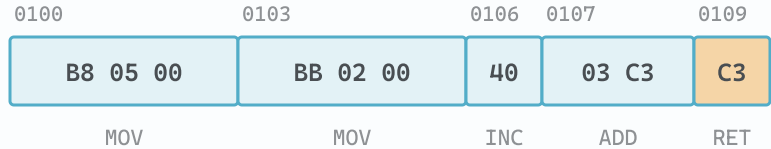
BX = 0002h

SP = FFEh

Kod İzleme: IP Nasıl İlerler?

```

1 ORG 100h
2 MOV AX, 5      ; 3 bayt
3 MOV BX, 2      ; 3 bayt
4 INC AX         ; 1 bayt
5 ADD AX, BX     ; 2 bayt
▶ RET           ; 1 bayt
    
```



IP = 0000h (PSP'ye dönüş)

RET yığından 0000h'yi çeker: **IP = 0000h**, SP = 0000h.
PSP:0000'daki INT 20h programı bitirir.

Adım 6 / 6

DEĞİŞKENLER

IP = 0000h

AX = 0008h

BX = 0002h

SP = 0000h

ÖZEL AMAÇLI YAZMAÇLAR

Bayrak Yazmacı

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
–	–	–	–	OF	DF	IF	TF	SF	ZF	–	AF	–	PF	–	CF

☐ durum bayrağı (sonuçtan etkilenir)

☐ kontrol bayrağı (programcı ayarlar)

Bayrak	Bit	1 olur, eğer...
CF elde (carry)	0	işaretsiz sonuç sığmadıysa: toplamada elde, çıkarmada ödünç
PF eşlik (parity)	2	sonucun düşük baytında çift sayıda 1 varsa
AF yardımcı elde	4	bit 3'ten bit 4'e elde/ödünç olduysa (BCD işlemleri)
ZF sıfır (zero)	6	sonuç sıfırsa
SF işaret (sign)	7	sonucun en anlamlı biti 1 ise (negatif)
OF taşma (overflow)	11	işaretsiz sonuç sığmadıysa
TF , IF , DF	8, 9, 10	adım adım çalışma, kesmelere izin, dizgi yönü (kontrol)

Bayraklar Nasıl Oluşur?

MOV AL, 7Fh · MOV BL, 01h · ADD AL, BL

AL	0111 1111	7Fh	+127
+ BL	0000 0001	01h	+1
= AL	1000 0000	80h	işaretsiz 128, işaretli -128

CF = 0 · bit 7'den dışarı elde çıkmadı (işaretsiz 127 + 1 = 128 sığar).

Bayraklar Nasıl Oluşur?

MOV AL, 7Fh · MOV BL, 01h · ADD AL, BL

AL	0111 1111	7Fh	+127
+ BL	0000 0001	01h	+1
= AL	1000 0000	80h	işaretsiz 128, işaretli -128

CF = 0 · bit 7'den dışarı elde çıkmadı (işaretsiz 127 + 1 = 128 sığar).

ZF = 0 · sonuç 80h sıfır değil.

Bayraklar Nasıl Oluşur?

MOV AL, 7Fh · MOV BL, 01h · ADD AL, BL

AL	0111 1111	7Fh	+127
+ BL	0000 0001	01h	+1
= AL	1000 0000	80h	işaretsiz 128, işaretli -128

CF = 0 · bit 7'den dışarı elde çıkmadı (işaretsiz 127 + 1 = 128 sığar).

ZF = 0 · sonuç 80h sıfır değil.

SF = 1 · sonucun bit 7'si 1 → işaretli yorumda negatif.

Bayraklar Nasıl Oluşur?

MOV AL, 7Fh · MOV BL, 01h · ADD AL, BL

AL	0111 1111	7Fh	+127
+ BL	0000 0001	01h	+1
= AL	1000 0000	80h	işaretsiz 128, işaretli -128

CF = 0 · bit 7'den dışarı elde çıkmadı (işaretsiz 127 + 1 = 128 sığar).

ZF = 0 · sonuç 80h sıfır değil.

SF = 1 · sonucun bit 7'si 1 → işaretli yorumda negatif.

OF = 1 · iki pozitif sayının toplamı negatif çıktı → işaretli taşma (+128, 8 bite sığmaz).

Bayraklar Nasıl Oluşur?

MOV AL, 7Fh · MOV BL, 01h · ADD AL, BL

AL	0111 1111	7Fh	+127
+ BL	0000 0001	01h	+1
= AL	1000 0000	80h	işaretsiz 128, işaretli -128

CF = 0 · bit 7'den dışarı elde çıkmadı (işaretsiz $127 + 1 = 128$ sığar).

ZF = 0 · sonuç 80h sıfır değil.

SF = 1 · sonucun bit 7'si 1 → işaretli yorumda negatif.

OF = 1 · iki pozitif sayının toplamı negatif çıktı → işaretli taşma (+128, 8 bite sığmaz).

PF = 0 · 1000 0000'da bir adet 1 var (tek sayı).

Bayraklar Nasıl Oluşur?

MOV AL, 7Fh · MOV BL, 01h · ADD AL, BL

AL	0111 1111	7Fh	+127
+ BL	0000 0001	01h	+1
= AL	1000 0000	80h	işaretsiz 128, işaretli -128

CF = 0 · bit 7'den dışarı elde çıkmadı (işaretsiz $127 + 1 = 128$ sığar).

ZF = 0 · sonuç 80h sıfır değil.

SF = 1 · sonucun bit 7'si 1 → işaretli yorumda negatif.

OF = 1 · iki pozitif sayının toplamı negatif çıktı → işaretli taşma ($+128$, 8 bite sığmaz).

PF = 0 · 1000 0000'da bir adet 1 var (tek sayı).

AF = 1 · düşük nibble: $Fh + 1h = 10h$ → bit 3'ten bit 4'e elde.

Bayrak Hesaplayıcı

AL = h BL = h

$$7Fh + 01h = \textcolor{red}{80h} \text{ (10000000b)}$$

işaretsiz: $127 + 1 = 128 \rightarrow 128$ · işaretli: $127 + 1 = 128 \rightarrow -128$

CF	ZF	SF	OF	PF	AF
0	0	1	1	0	1

CF: işaretsiz toplam 255'i aştı mı? · **OF**: işaretli sonuç $-128 \dots 127$ dışında mı? · **PF**: sonuçta 1 adet 1 $\rightarrow$ tek · **AF**: alt 4 bitte elde var

8 bitlik ya da sonrası durum bayrakları. Değerleri onaltılık girin (00–FF).

Kod İzleme: Bayraklar

► **MOV** AL, 0F0h

2 **ADD** AL, 20h ; F0h + 20h

3 **SUB** AL, 10h

4 **SUB** AL, 1

5 **RET**

AX

00	F0
AH	AL

Bayraklar

CF=0	ZF=0	SF=0	OF=0	PF=0	AF=0
------	------	------	------	------	------

AL = F0h (işaretsiz 240, işaretli -16). MOV bayrakları değiştirmez.

Adım 1 / 5

DEĞİŞKENLER

AL = F0h

CF = 0

ZF = 0

SF = 0

OF = 0

PF = 0

AF = 0

Kod İzleme: Bayraklar

1 **MOV** AL, 0F0h

▶ **ADD** AL, 20h ; F0h + 20h

3 **SUB** AL, 10h

4 **SUB** AL, 1

5 **RET**

AX

00	10
AH	AL

Bayraklar

CF=1	ZF=0	SF=0	OF=0	PF=0	AF=0
------	------	------	------	------	------

F0h + 20h = 110h → AL = **10h**, dışarı taşan 1 → **CF = 1**. İşaretli:
-16 + 32 = 16 doğru → OF = 0.

Adım 2 / 5

DEĞİŞKENLER

AL = 10h

CF = 1

ZF = 0

SF = 0

OF = 0

PF = 0

AF = 0

Kod İzleme: Bayraklar

```

1 MOV AL, 0F0h
2 ADD AL, 20h ; F0h + 20h
3 ► SUB AL, 10h
4 SUB AL, 1
5 RET
    
```

AX 00 00
 AH AL

Bayraklar

CF=0 ZF=1 SF=0 OF=0 PF=1 AF=0

$10h - 10h = 00h \rightarrow \mathbf{ZF = 1}$, $PF = 1$ (sıfır adet 1 çifttir), ödünç yok
 $\rightarrow CF = 0$.

Adım 3 / 5

DEĞİŞKENLER

AL = 00h

CF = 0

ZF = 1

SF = 0

OF = 0

PF = 1

AF = 0

Kod İzleme: Bayraklar

```
1 MOV AL, 0F0h
2 ADD AL, 20h ; F0h + 20h
3 SUB AL, 10h
▶ SUB AL, 1
5 RET
```

AX

00	FF
AH	AL

Bayraklar

CF=1	ZF=0	SF=1	OF=0	PF=1	AF=1
------	------	------	------	------	------

$00h - 1 = \mathbf{FFh}$: ödünç alındı → **CF = 1**, SF = 1 (işaretli -1), PF = 1 (sekiz adet 1), AF = 1.

Adım 4 / 5

DEĞİŞKENLER

AL = **FFh**

CF = **1**

ZF = **0**

SF = **1**

OF = **0**

PF = **1**

AF = **1**

Kod İzleme: Bayraklar

```
1 MOV AL, 0F0h
2 ADD AL, 20h ; F0h + 20h
3 SUB AL, 10h
4 SUB AL, 1
▶ RET
```

AX

00	FF
AH	AL

Bayraklar

CF=1	ZF=0	SF=1	OF=0	PF=1	AF=1
------	------	------	------	------	------

RET ile DOS'a dönülür. Aynı bit deseni (FFh) işaretli 255, işaretsiz -1 olarak yorumlanabilir.

Adım 5 / 5

DEĞİŞKENLER

AL = FFh

CF = 1

ZF = 0

SF = 1

OF = 0

PF = 1

AF = 1

.COM Program Şablonu

```
ORG 100h          ; .COM: kod 0100h'den başlar

MOV AX, 5          ; AX = 5
MOV BX, 3          ; BX = 3
ADD AX, BX         ; AX = AX + BX = 8
MOV CL, AL         ; CL = 08h

RET               ; DOS'a dön

sayi DB 0         ; değişkenler RET'ten sonra
```

- **ORG 100h** : .COM programı kesimin 0100h adresine yüklenir; ilk 256 bayt (PSP) DOS'a aittir.
- Komutlar yukarıdan aşağı, **sırayla** yürütülür.
- **RET** : işletim sistemine geri döner.
- Değişkenler kodun **sonuna** (RET'ten sonra) konur ki CPU onları komut sanıp yürütmesin.
- .COM'da başlangıçta **CS = DS = ES = SS** , **IP = 0100h** , **SP = FFFEh** .

emu8086'da: **new** → **COM template**, kodu yaz, **emulate**; ardından **single step** ile her komutun yazmaçları nasıl değiştirdiğini izle.

Kod İzleme: .COM Programı Bellekte

► **ORG 100h** ; kod 0100h'den

```
2 MOV AX, 5
3 MOV BX, 3
4 ADD AX, BX ; AX = 8
5 MOV CL, AL ; CL = 08h
6 RET ; DOS'a dön
7 sayi DB 0 ; veri
```

IP = 0100h

SP = FFEh

AX

0000h

BX

0000h

CL

00h

CS:IP →

SS:SP →

kesim 0700h	
0000h	PSP (256 bayt)
0100h	MOV AX, 5
0103h	MOV BX, 3
0106h	ADD AX, BX
0108h	MOV CL, AL
010Ah	RET
010Bh	sayi = 00
⋮	
FFFEh	yığın: 0000h

Program 0700:0100'e yüklendi. CS = DS = ES = SS = 0700h, **IP = 0100h**, **SP = FFEh**; yığın tepesinde 0000h var.

Adım 1 / 6

DEĞİŞKENLER

AX = 0000h

BX = 0000h

CL = 00h

IP = 0100h

SP = FFEh

Komutlar 3, 3, 2, 2 ve 1 bayttır; IP her komutta uzunluğu kadar ilerler. Veriler RET'ten sonra durduğu için komut olarak yürütülmez.

Kod İzleme: .COM Programı Bellekte

```
1 ORG 100h      ; kod 0100h'den
▶ 2 MOV AX, 5
3 MOV BX, 3
4 ADD AX, BX      ; AX = 8
5 MOV CL, AL      ; CL = 08h
6 RET            ; DOS'a dön
7 sayi DB 0      ; veri
```

IP = 0103h

SP = FFEh

AX
0005h
BX
0000h
CL
00h

CS:IP

SS:SP

kesim 0700h	
0000h	PSP (256 bayt)
0100h	MOV AX, 5
0103h	MOV BX, 3
0106h	ADD AX, BX
0108h	MOV CL, AL
010Ah	RET
010Bh	sayi = 00
⋮	
FFFEh	yığın: 0000h

MOV AX, 5 (3 bayt): AX = 0005h, IP = 0103h.

Adım 2 / 6

DEĞİŞKENLER

AX = 0005h

BX = 0000h

CL = 00h

IP = 0103h

SP = FFEh

Komutlar 3, 3, 2, 2 ve 1 bayttır; IP her komutta uzunluğu kadar ilerler. Veriler RET'ten sonra durduğu için komut olarak yürütülmez.

Kod İzleme: .COM Programı Bellekte

```

1  ORG 100h      ; kod 0100h'den
2  MOV AX, 5
3  ► MOV BX, 3
4  ADD AX, BX      ; AX = 8
5  MOV CL, AL      ; CL = 08h
6  RET            ; DOS'a dön
7  sayi DB 0      ; veri

```

IP = 0106h

SP = FFEh

AX

0005h

CS:IP

BX

0003h

CL

00h

SS:SP

kesim 0700h	
0000h	PSP (256 bayt)
0100h	MOV AX, 5
0103h	MOV BX, 3
0106h	ADD AX, BX
0108h	MOV CL, AL
010Ah	RET
010Bh	sayi = 00
⋮	
FFFEh	yığın: 0000h

MOV BX, 3 (3 bayt): BX = 0003h, IP = **0106h**.

Adım 3 / 6

DEĞİŞKENLER

AX = 0005h

BX = 0003h

CL = 00h

IP = 0106h

SP = FFEh

Komutlar 3, 3, 2, 2 ve 1 bayttır; IP her komutta uzunluğu kadar ilerler. Veriler RET'ten sonra durduğu için komut olarak yürütülmez.

Kod İzleme: .COM Programı Bellekte

```

1  ORG 100h      ; kod 0100h'den
2  MOV AX, 5
3  MOV BX, 3
4  ► ADD AX, BX    ; AX = 8
5  MOV CL, AL     ; CL = 08h
6  RET           ; DOS'a dön
7  sayi DB 0     ; veri

```

IP = 0108h

SP = FFEh

AX

0008h

BX

0003h

CL

00h

CS:IP

SS:SP

ADD AX, BX (2 bayt): AX = **0008h**, IP = 0108h.

kesim 0700h

0000h PSP (256 bayt)

0100h MOV AX, 5

0103h MOV BX, 3

0106h ADD AX, BX

0108h MOV CL, AL

010Ah RET

010Bh **sayi = 00**

⋮

FFFEh yığın: 0000h

Adım 4 / 6

DEĞİŞKENLER

AX = 0008h

BX = 0003h

CL = 00h

IP = 0108h

SP = FFEh

Komutlar 3, 3, 2, 2 ve 1 bayttır; IP her komutta uzunluğu kadar ilerler. Veriler RET'ten sonra durduğu için komut olarak yürütülmez.

Kod İzleme: .COM Programı Bellekte

```
1 ORG 100h      ; kod 0100h'den
2 MOV AX, 5
3 MOV BX, 3
4 ADD AX, BX      ; AX = 8
5 MOV CL, AL      ; CL = 08h
6 RET            ; DOS'a dön
7 sayi DB 0      ; veri
```

IP = 010Ah

SP = FFEh

AX

0008h

BX

0003h

CL

08h

CS:IP

SS:SP

kesim 0700h

0000h PSP (256 bayt)

0100h MOV AX, 5

0103h MOV BX, 3

0106h ADD AX, BX

0108h MOV CL, AL

010Ah RET

010Bh sayi = 00

⋮

FFFEh yığın: 0000h

MOV CL, AL (2 bayt): CL = 08h, IP = 010Ah.

Adım 5 / 6

DEĞİŞKENLER

AX = 0008h

BX = 0003h

CL = 08h

IP = 010Ah

SP = FFEh

Komutlar 3, 3, 2, 2 ve 1 bayttır; IP her komutta uzunluğu kadar ilerler. Veriler RET'ten sonra durduğu için komut olarak yürütülmez.

Kod İzleme: .COM Programı Bellekte

```
1 ORG 100h      ; kod 0100h'den
2 MOV AX, 5
3 MOV BX, 3
4 ADD AX, BX      ; AX = 8
5 MOV CL, AL      ; CL = 08h
6 RET            ; DOS'a dön
7 sayi DB 0      ; veri
```

IP = 0000h

SP = 0000h

AX

0008h

BX

0003h

CL

08h

CS:IP

kesim 0700h	
0000h	PSP (256 bayt)
0100h	MOV AX, 5
0103h	MOV BX, 3
0106h	ADD AX, BX
0108h	MOV CL, AL
010Ah	RET
010Bh	sayi = 00
⋮	
FFFEh	yığın: 0000h

RET yığından 0000h'yi alır: IP = 0000h, SP = 0000h. PSP:0000'daki

INT 20h programı bitirir; 010Bh'deki **sayi** hiç yürütülmedi.

Adım 6 / 6

DEĞİŞKENLER

AX = 0008h

BX = 0003h

CL = 08h

IP = 0000h

SP = 0000h

Komutlar 3, 3, 2, 2 ve 1 bayttır; IP her komutta uzunluğu kadar ilerler. Veriler RET'ten sonra durduğu için komut olarak yürütülmez.

Yazım Kuralları

Sayı tabanları

25 veya 25d onluk · 19h onaltılık ·
00011001b ikili · 31o sekizli. Hepsı aynı
değer: 25.

Harfle başlayan onaltılık

Başına 0 konur: 0FFh, 0B800h. FFh
yazılırsa assembler onu bir *ad*
(etiket/değişken) sanır.

Karakterler

'A' → ASCII kodu 41h. MOV CL, 'A' ile
MOV CL, 41h aynı makine kodunu üretir.

İşaretli sayılar

-1 8 bitte FFh, 16 bitte FFFFh olarak saklanır
(ikiye tümleyen).

Büyük-küçük harf

Assembler duyarsızdır: mov, MOV, Mov
aynıdır; var1 ile VAR1 aynı addır.

Yorumlar

; sonrası yorumdur. Her satıra "ne" değil
"neden" yazmak iyi alışkanlıktır.

Sık Yapılan Hatalar

Hatalı	Neden?	Doğrusu
<code>MOV AL, BX</code>	Boyutlar farklı (8 ↔ 16 bit)	<code>MOV AL, BL</code> veya <code>MOV AX, BX</code>
<code>MOV DS, 1234h</code>	Kesim yazmacına sabit atanamaz	<code>MOV AX, 1234h</code> · <code>MOV DS, AX</code>
<code>MOV AL, [AX]</code>	Adreslemede yalnız BX, SI, DI, BP	<code>MOV BX, AX</code> · <code>MOV AL, [BX]</code>
<code>MOV AL, FFh</code>	FFh bir ad sanılır	<code>MOV AL, 0FFh</code>
<code>MOV IP, 200h</code>	IP'ye MOV ile yazılamaz	<code>JMP</code> ile atla
<code>MOV AL, 300</code>	300 > 255, 8 bite sığmaz	<code>MOV AX, 300</code>

Bu kurallar MOV komutunun işlenen kısıtlarından gelir.

ÖZET

Özet

Assembly

Makine koduyla birebir eşleşen düşük seviyeli dil; küçük ve hızlı kod, donanıma doğrudan erişim, ama taşınabilir değil.

Bilgisayar modeli

CPU + RAM + G/Ç, sistem yolu ile bağlı: adres (20 bit), veri (16 bit), denetim.

Genel amaçlı yazmaçlar

AX, BX, CX, DX (8 bitlik H/L yarılarıyla), SI, DI, BP, SP; hepsi 16 bit.

Kesim yazmaçları

CS, DS, ES, SS; fiziksel adres = kesim $\times$ 10h + bağlı konum.

IP

CS:IP sıradaki komutu gösterir; komut uzunluğu kadar ilerler.

Bayraklar

CF, PF, AF, ZF, SF, OF sonuca göre; TF, IF, DF kontrol için.

ÖZET

Alıştırmalar

1. $AX = 0ABCDh$ iken `MOV AH, 0` sonrası AX , AL ve BX değerleri nedir?
2. $DS = 2000h$, $BX = 1234h$ ise `[BX]` hangi fiziksel adrestir? Aynı adresi veren başka bir kesim:bağlı konum çifti yazın.
3. Fiziksel adres $B815Eh$ için normalize edilmiş kesim:bağlı konum nedir?
4. $AL = 80h$ iken `ADD AL, 80h` sonrası AL , CF , ZF , SF , OF ?
5. Program $0100h$ 'de başlar: $3 + 2 + 1 + 4$ baytlık dört komut yürütülünce IP ne olur?
6. `MOV CX, AL` ve `MOV ES, 0B800h` neden hatalıdır? Düzeltin.

► Cevaplar

Kaynaklar

- emu8086 belgeleri, *8086 assembler tutorial for beginners* (bölüm 1: numbering systems, bölüm 2: memory access).
- Intel, *8086 16-Bit HMOS Microprocessor* veri sayfası.
- Barry B. Brey, *The Intel Microprocessors*, 8. baskı, Pearson.
- Muhammad Ali Mazidi, Janice G. Mazidi, *The 80x86 IBM PC and Compatible Computers*, Pearson.
- Andrew S. Tanenbaum, *Modern Operating Systems*, 3. baskı (bilgisayar modeli).

Son

Bölüm 3: Assembly

Sonraki bölüm: Bölüm 4: Bellek Erişimi · [Terim sözlüğü](#)