

Bölüm 4: Bellek Erişimi

Mikroişlemciler

Adresleme modları

İleri kaydırma

Etkin adres

Kesim:bağıl konum

byte ptr / word ptr

MOV kuralları

Video belleği örneği

XCHG

8/16 bit kopyalama

Sayı tabanları

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

Öğrenme Çıktıları

Adresleme modları

BX, SI, DI, BP ve ileri kaydırma ile geçerli bellek işlenenleri oluşturmak.

Adres hesabı

Etkin adresi ve kesim:bağlı konumdan fiziksel adresi hesaplamak.

Veri boyutu

`byte ptr` / `word ptr` ile erişim boyutunu belirtmek; little-endian düzeni okumak.

MOV ve XCHG

Geçerli işlenen kombinasyonlarını ayırt etmek ve kopyalama/takas programlarını izlemek.

ADRESLEME MODLARI

Bellek Erişimi

- Belleğe erişmek için **BX, SI, DI, BP** yazmaçları kullanılır.
- Bu yazmaçlar (ve bir sabit) `[ ]` sembolleri içinde birleştirilerek farklı bellek konumlarına erişilir; bu birleştirmelere **adresleme modları** denir.

Yalnız yazmaç(lar)	+ d8 (8 bit kaydırma)	+ d16 (16 bit kaydırma)
[BX + SI]	[BX + SI + d8]	[BX + SI + d16]
[BX + DI]	[BX + DI + d8]	[BX + DI + d16]
[BP + SI]	[BP + SI + d8]	[BP + SI + d16]
[BP + DI]	[BP + DI + d8]	[BP + DI + d16]
[SI]	[SI + d8]	[SI + d16]
[DI]	[DI + d8]	[DI + d16]
d16 (doğrudan adres)	[BP + d8]	[BP + d16]
[BX]	[BX + d8]	[BX + d16]

24 kombinasyon, 8086 makine kodundaki mod/rm alanının tüm bellek seçenekleridir.

`[BP]` tek başına yazılırsa assembler onu `[BP + 0]` ($d8 = 0$) olarak kodlar.

Kombinasyon Kuralı



her sütundan en fazla bir seçim · BP seçilirse kesim SS, değilse DS

Kurallar

- **BX ve BP** aynı modda bir araya gelmez.
- **SI ve DI** aynı modda bir araya gelmez.
- SS kesimi BP ile, DS kesimi BP içermeyen modlarla kullanılır.

Örnekler

- `[BX+5]` → taban + d8 (DS)
- `[BX+SI]` → taban + indis (DS)
- `[DI+BX-4]` → `[BX+DI-4]`, sıra önemsiz (DS)
- `[BP+SI+300]` → taban + indis + d16 (SS)

Yer Değiştirme (Displacement)

d8

8 bit işaretli yer değiştirme ($-128 \dots +127$).

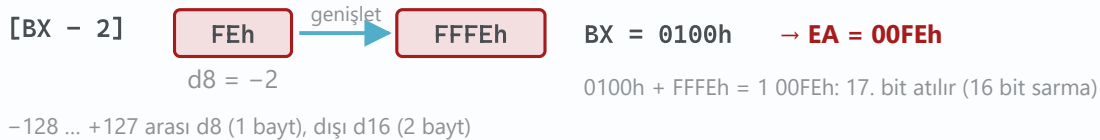
Örnek: 22, 55h, -1.

d16

16 bit işaretli yer değiştirme. Örnek: 300,

5517h, -259.

- Yer değiştirme bir sayı ya da bir **değişkenin bağlı konumu** (offset) olabilir: $[BX + \text{dizi}]$.
- Birden fazla değer varsa assembler **tek bir değer** hesaplar: $[BX + 2 + 3] \rightarrow [BX + 5]$.
- Yer değiştirme $[]$ içinde veya dışında yazılabilir; assembler aynı makine kodunu üretir: $[BX+5] = 5[BX] = [BX]+5$.
- İşaretli bir değerdir; pozitif ya da negatif olabilir: $[SI-1]$.
- Assembler değer $-128 \dots 127$ aralığındaysa kısa (d8), değilse uzun (d16) biçimi seçer.



Etkin adres 16 bit ile hesaplanır ve **başa sarar**: $BX = FFFFh$ iken $[BX+2] \rightarrow EA = 0001h$.

ADRESLEME MODLARI

Geçerli mi, Geçersiz mi?

Geçerli	Açıklama
[BX]	yazmaç dolaylı
[1234h]	doğrudan adres
[SI+10]	indisli
[BP+DI-2]	tabanlı-indisli (SS)
ES: [BX+SI]	kesim önekli

Geçersiz	Neden?
[AX], [CX], [DX]	adreslemede kullanılamaz
[BL], [BH]	8 bitlik yazmaç adres olamaz
[BX+BP]	iki taban yazmacı
[SI+DI]	iki indis yazmacı
[SP], [BX*2]	8086'da yok (80386+)

Tablo kullanılarak tüm geçerli kombinasyonlar oluşturulabilir: taban (BX/BP) + indis (SI/DI) + kaydırma, her biri isteğe bağlı.

ADRESLEME MODLARI

Adresleme Modu Türleri

Mod	Örnek	İşlenen nerede?
Anlık (immediate)	MOV AX, 5	komutun içinde (sabit)
Yazmaç (register)	MOV AX, BX	yazmaçta
Doğrudan (direct)	MOV AX, [1000h]	DS:1000h
Yazmaç dolaylı	MOV AX, [BX]	DS:BX
Tabanlı (based)	MOV AX, [BP+4]	SS:BP+4
İndisli (indexed)	MOV AL, [SI+3]	DS:SI+3
Tabanlı-indisli	MOV AL, [BX+DI]	DS:BX+DI
Tabanlı-indisli + yer değiştirme	MOV AL, [BX+SI+8]	DS:BX+SI+8

İlk ikisi belleğe gitmez; geri kalanlar etkin adres (EA) hesaplar. Fiziksel adres = kesim × 10h + EA.

Kod İzleme: Altı Adresleme Modu

```
1 ORG 100h
2 MOV BX, OFFSET tablo
3 MOV SI, 2
4 MOV AH, 7 ; anlık
5 MOV AL, [tablo+1] ; doğrudan
6 MOV CL, [BX] ; dolaylı
7 MOV CH, [BX+4] ; tabanlı
8 MOV DL, [BX+SI] ; taban+indis
9 MOV DH, [BX+SI+3] ; + d8
10 RET
11 tablo DB 11h,22h,33h,44h,55h,66h
```

tablo (DS:0116h ...)



BX = 0116h SI = 0000h

BX = **0116h**: **tablo** 'nun bağlı konumu (OFFSET derleme anında hesaplanır).

Adım 1 / 8

DEĞİŞKENLER

BX = 0116h

SI = 0000h

AX = 0000h

CX = 0000h

DX = 0000h

.COM programında DS = 0700h alındı; fiziksel adres = DS × 10h + EA.

Kod İzleme: Altı Adresleme Modu

```
1 ORG 100h
2 MOV BX, OFFSET tablo
3 MOV SI, 2
4 MOV AH, 7 ; anlık
5 MOV AL, [tablo+1] ; doğrudan
6 MOV CL, [BX] ; dolaylı
7 MOV CH, [BX+4] ; tabanlı
8 MOV DL, [BX+SI] ; taban+indis
9 MOV DH, [BX+SI+3] ; + d8
10 RET
11 tablo DB 11h,22h,33h,44h,55h,66h
```

tablo (DS:0116h ...)



BX = 0116h SI = 0002h

SI = 2: indis yazmacı.

Adım 2 / 8

DEĞİŞKENLER

BX = 0116h

SI = 0002h

AX = 0000h

CX = 0000h

DX = 0000h

.COM programında DS = 0700h alındı; fiziksel adres = DS × 10h + EA.

Kod İzleme: Altı Adresleme Modu

```
1 ORG 100h
2 MOV BX, OFFSET tablo
3 MOV SI, 2
4 MOV AH, 7 ; anlık
5 MOV AL, [tablo+1] ; doğrudan
6 MOV CL, [BX] ; dolaylı
7 MOV CH, [BX+4] ; tabanlı
8 MOV DL, [BX+SI] ; taban+indis
9 MOV DH, [BX+SI+3] ; + d8
10 RET
11 tablo DB 11h,22h,33h,44h,55h,66h
```

tablo (DS:0116h ...)



anlık

değer komutun içinde: 07h

Anlık mod: 7 komutun içindedir; belleğe gidilmez. AH = 07h.

Adım 3 / 8

DEĞİŞKENLER

BX = 0116h

SI = 0002h

AX = 0700h

CX = 0000h

DX = 0000h

.COM programında DS = 0700h alındı; fiziksel adres = DS × 10h + EA.

Kod İzleme: Altı Adresleme Modu

```
1 ORG 100h
2 MOV BX, OFFSET tablo
3 MOV SI, 2
4 MOV AH, 7 ; anlık
5 MOV AL, [tablo+1] ; doğrudan
6 MOV CL, [BX] ; dolaylı
7 MOV CH, [BX+4] ; tabanlı
8 MOV DL, [BX+SI] ; taban+indis
9 MOV DH, [BX+SI+3] ; + d8
10 RET
11 tablo DB 11h,22h,33h,44h,55h,66h
```

tablo (DS:0116h ...)



Doğrudan mod: adres komutun içinde: [0117h] → AL = 22h.

Adım 4 / 8

DEĞİŞKENLER

BX = 0116h

SI = 0002h

AX = 0722h

CX = 0000h

DX = 0000h

.COM programında DS = 0700h alındı; fiziksel adres = DS × 10h + EA.

Kod İzleme: Altı Adresleme Modu

```
1 ORG 100h
2 MOV BX, OFFSET tablo
3 MOV SI, 2
4 MOV AH, 7 ; anlık
5 MOV AL, [tablo+1] ; doğrudan
6 MOV CL, [BX] ; dolaylı
7 MOV CH, [BX+4] ; tabanlı
8 MOV DL, [BX+SI] ; taban+indis
9 MOV DH, [BX+SI+3] ; + d8
10 RET
11 tablo DB 11h,22h,33h,44h,55h,66h
```

tablo (DS:0116h ...)



yazmaç dolaylı

EA = BX = 0116h

fiziksel = 07000h + 0116h = 07116h

Yazmaç dolaylı: EA = BX = 0116h → CL = 11h.

Adım 5 / 8

DEĞİŞKENLER

BX = 0116h

SI = 0002h

AX = 0722h

CX = 0011h

DX = 0000h

.COM programında DS = 0700h alındı; fiziksel adres = DS × 10h + EA.

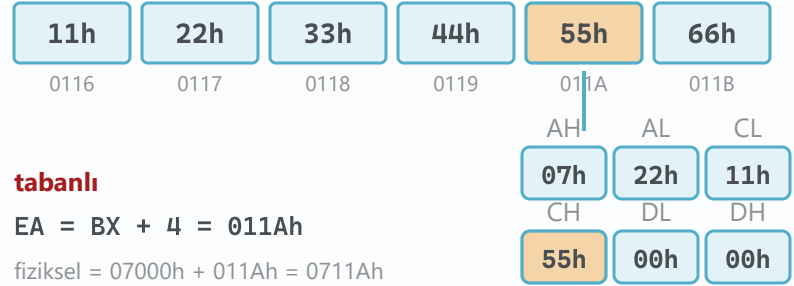
Kod İzleme: Altı Adresleme Modu

```

1  ORG 100h
2  MOV BX, OFFSET tablo
3  MOV SI, 2
4  MOV AH, 7 ; anlık
5  MOV AL, [tablo+1] ; doğrudan
6  MOV CL, [BX] ; dolaylı
7  MOV CH, [BX+4] ; tabanlı
8  MOV DL, [BX+SI] ; taban+indis
9  MOV DH, [BX+SI+3] ; + d8
10 RET
11 tablo DB 11h,22h,33h,44h,55h,66h

```

tablo (DS:0116h ...)



Tabanlı: EA = BX + 4 = 011Ah → CH = **55h**.

Adım 6 / 8

DEĞİŞKENLER

BX = 0116h

SI = 0002h

AX = 0722h

CX = **5511h**

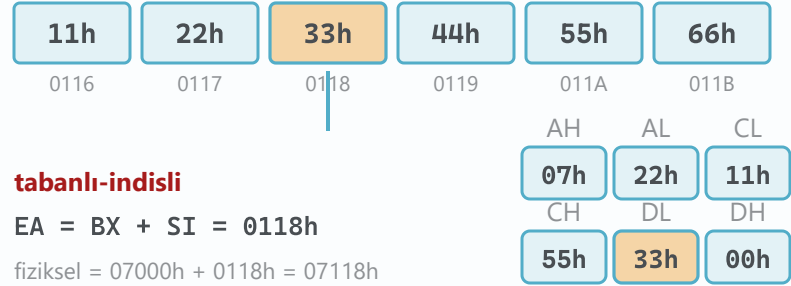
DX = 0000h

.COM programında DS = 0700h alındı; fiziksel adres = DS × 10h + EA.

Kod İzleme: Altı Adresleme Modu

```
1 ORG 100h
2 MOV BX, OFFSET tablo
3 MOV SI, 2
4 MOV AH, 7 ; anlık
5 MOV AL, [tablo+1] ; doğrudan
6 MOV CL, [BX] ; dolaylı
7 MOV CH, [BX+4] ; tabanlı
8 MOV DL, [BX+SI] ; taban+indis
9 MOV DH, [BX+SI+3] ; + d8
10 RET
11 tablo DB 11h,22h,33h,44h,55h,66h
```

tablo (DS:0116h ...)



Tabanlı-indisli: $EA = BX + SI = 0118h \rightarrow DL = 33h$.

Adım 7 / 8

DEĞİŞKENLER

BX = 0116h

SI = 0002h

AX = 0722h

CX = 5511h

DX = 0033h

.COM programında $DS = 0700h$ alındı; fiziksel adres = $DS \times 10h + EA$.

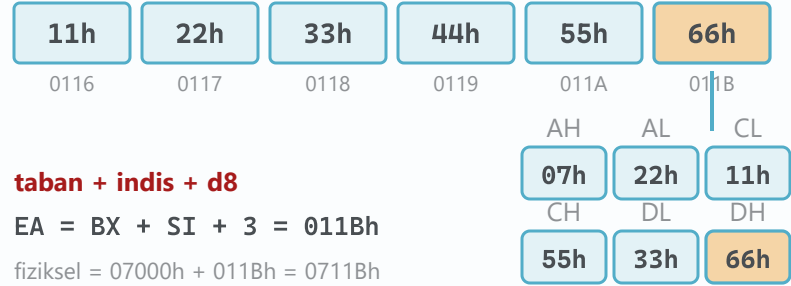
Kod İzleme: Altı Adresleme Modu

```

1 ORG 100h
2 MOV BX, OFFSET tablo
3 MOV SI, 2
4 MOV AH, 7 ; anlık
5 MOV AL, [tablo+1] ; doğrudan
6 MOV CL, [BX] ; dolaylı
7 MOV CH, [BX+4] ; tabanlı
8 MOV DL, [BX+SI] ; taban+indis
9 MOV DH, [BX+SI+3] ; + d8
10 RET
11 tablo DB 11h,22h,33h,44h,55h,66h

```

tablo (DS:0116h ...)



Taban + indis + d8: $EA = BX + SI + 3 = 011Bh \rightarrow DH = 66h$.
BP kullanılmadığı için kesim hep DS.

Adım 8 / 8

DEĞİŞKENLER

BX = 0116h

SI = 0002h

AX = 0722h

CX = 5511h

DX = 6633h

.COM programında DS = 0700h alındı; fiziksel adres = DS × 10h + EA.

Etkin Adres Adım Adım

MOV AL, [BX + SI + 4] · DS = 1000h, BX = 0200h, SI = 0030h

BX

0200

Taban yazmacı BX = 0200h.

Etkin Adres Adım Adım

MOV AL, [BX + SI + 4] · DS = 1000h, BX = 0200h, SI = 0030h

BX	0200
+ SI	0030

İndis yazmacı SI = 0030h eklenir.

Etkin Adres Adım Adım

MOV AL, [BX + SI + 4] · DS = 1000h, BX = 0200h, SI = 0030h

BX	0200
+ SI	0030
+ d8	0004

8 bitlik ileri kaydırma: +4.

Etkin Adres Adım Adım

`MOV AL, [BX + SI + 4]` · DS = 1000h, BX = 0200h, SI = 0030h

BX	0200
+ SI	0030
+ d8	0004
EA	0234

Etkin adres EA = 0200h + 0030h + 0004h = **0234h** (16 bit).

Etkin Adres Adım Adım

MOV AL, [BX + SI + 4] · DS = 1000h, BX = 0200h, SI = 0030h

BX	0200	Fiziksel adres
+ SI	0030	DS × 10h = 10000h
+ d8	0004	+ EA = 0234h
		10234h
EA	0234	AL ← [10234h]

BP yok → kesim **DS**. Fiziksel adres = 10000h + 0234h = **10234h**. AL bu adresteki baytı alır.

BP ile Erişim

MOV AX, [BP + DI - 2] · SS = 2000h, BP = 0100h, DI = 0010h

BP

0100

Taban yazmacı BP = 0100h.

BP ile Erişim

MOV AX, [BP + DI - 2] · SS = 2000h, BP = 0100h, DI = 0010h

BP	0100
+ DI	0010

İndis yazmacı DI = 0010h eklenir.

BP ile Erişim

MOV AX, [BP + DI - 2] · SS = 2000h, BP = 0100h, DI = 0010h

BP	0100
+ DI	0010
+ d8	FFFE

İşaretli yer değiştirme -2 (makine kodunda FEh).

BP ile Erişim

MOV AX, [BP + DI - 2] · SS = 2000h, BP = 0100h, DI = 0010h

BP	0100
+ DI	0010
+ d8	FFFE
EA	010E

$$EA = 0100h + 0010h - 2 = \mathbf{010Eh}.$$

BP ile Erişim

MOV AX, [BP + DI - 2] · SS = 2000h, BP = 0100h, DI = 0010h

BP	0100	Fiziksel adres
+ DI	0010	SS × 10h = 20000h
+ d8	FFFE	+ EA = 010Eh
		2010Eh
EA	010E	AX ← [2010Eh], [2010Fh]

BP kullanıldığı için kesim **SS**: 20000h + 010Eh = **2010Eh**. AX ← [2010Eh] (2 bayt: 2010Eh ve 2010Fh).

Etkin Adres Hesaplayıcı

[BX ▼ + SI ▼ + 4] varsayılan kesim ▼

BX

0200

BP

0100

SI

0030

DI

0010

DS

1000

ES

B800

SS

2000

CS

0700

[BX + SI + 0004h]

EA = BX (0200h) + SI (0030h) + 0004h = **0234h**

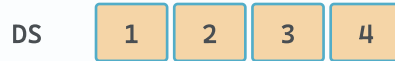
DS × 10h + EA = 10000h + 0234h = **10234h**

Mod: **tabanlı-indsli + kaydırma** · kaydırma: d8 · kesim: **DS** (varsayılan DS)

Yazmaç değerleri onaltılık; yer değiştirme onluk (**-2**) ya da onaltılık (**1Ah**) girilebilir.

Kesim ve Bağıl Konum

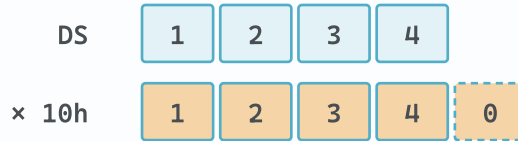
- Kesim yazmacındaki (CS, DS, SS, ES) değere **kesim** (segment), adresi veren yazmaçlardaki (BX, SI, DI, BP) değere **bağıl konum** (offset) denir.
- Kesim belleğin bir **bloğunu**, bağıl konum o bloktaki **belirli bir konumu** temsil eder; ikisi birleşerek fiziksel adresi oluşturur.



DS = **1234h**, SI = **7890h**. Gösterim: **1234:7890** .

Kesim ve Bağıl Konum

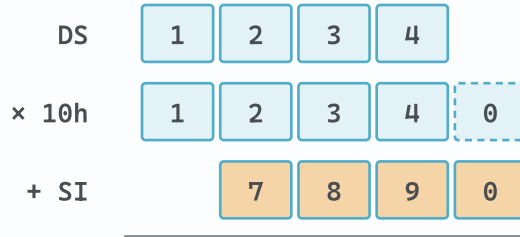
- Kesim yazmacındaki (CS, DS, SS, ES) değere **kesim** (segment), adresi veren yazmaçlardaki (BX, SI, DI, BP) değere **bağıl konum** (offset) denir.
- Kesim belleğin bir **bloğunu**, bağıl konum o bloktaki **belirli bir konumu** temsil eder; ikisi birleşerek fiziksel adresi oluşturur.



Kesim 10h ile çarpılır (4 bit sola kaydırma): **12340h**.

Kesim ve Bağlı Konum

- Kesim yazmacındaki (CS, DS, SS, ES) değere **kesim** (segment), adresi veren yazmaçlardaki (BX, SI, DI, BP) değere **bağlı konum** (offset) denir.
- Kesim belleğin bir **bloğunu**, bağlı konum o bloktaki **belirli bir konumu** temsil eder; ikisi birleşerek fiziksel adresi oluşturur.



Bağlı konum eklenmek üzere hizalanır: **7890h**.

Kesim ve Bağlı Konum

- Kesim yazmacındaki (CS, DS, SS, ES) değere **kesim** (segment), adresi veren yazmaçlardaki (BX, SI, DI, BP) değere **bağlı konum** (offset) denir.
- Kesim belleğin bir **bloğunu**, bağlı konum o bloktaki **belirli bir konumu** temsil eder; ikisi birleşerek fiziksel adresi oluşturur.

DS	1	2	3	4	
× 10h	1	2	3	4	0
+ SI		7	8	9	0
=	1	9	B	D	0

elde: 4 + 9 = D, 3 + 8 = B,
2 + 7 = 9, 1 + 0 = 1 → 19BD0h

12340h + 7890h = **19BD0h** fiziksel adres.

Kesim Öneki

Varsayılan kesim, köşeli parantez önüne **kesim öneki** yazılarak değiştirilebilir:

```
MOV AX, 0B800h
MOV ES, AX      ; ES = video belleği
MOV BX, 0
MOV AL, ES:[BX] ; ES:0000'dan oku
MOV AL, [BX]    ; DS:0000'dan oku
MOV DL, SS:[SI] ; SS:SI'dan oku
```

Önek, makine kodunda komutun önüne eklenen 1 baytlık bir koddur (ES: → 26h, CS: → 2Eh, SS: → 36h, DS: → 3Eh).

Ne zaman gerekir?

- DS'yi değiştirmeden başka bir kesimdeki veriye (ör. video belleği) erişmek.
- Yığındaki veriye BP dışındaki bir yazmaçla erişmek: **SS:[BX]**.
- Koddaki bir tabloya erişmek: **CS:[SI]**.

IP her zaman CS ile, SP her zaman SS ile kullanılır; bunlar değiştirilemez.

Kod İzleme: ES ile Video Belleği

1 **ORG** 100h

► **MOV** AX, 0B800h

3 **MOV** ES, AX ; ES = video

4 **MOV** BX, 00A0h ; satır 1

5 **MOV BYTE PTR** ES:[BX], 'K'

6 **MOV BYTE PTR** ES:[BX+1], 1Eh

7 **MOV** AL, [BX] ; DS:00A0h

8 **MOV** DL, ES:[BX] ; B800:00A0h

9 **RET**

DS = 0700h

07000h + 0000h = 07000h

00h

+0

ekran, satır 1:

ES = 0700h

07000h + 0000h = 07000h

20h

07h

+0

+1

AL

DL

00h

00h

AX = B800h: renkli metin ekranının kesimi. Video belleğinde bir hücre = karakter + renk.

Adım 1 / 7

DEĞİŞKENLER

DS = 0700h

ES = 0700h

BX = 0000h

AL = 00h

DL = 00h

Başlangıçta video hücresi 20h 07h (boşluk, gri) kabul edildi. Önek yalnızca o komutu etkiler; DS değişmez.

Kod İzleme: ES ile Video Belleği

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV ES, AX      ; ES = video
4 MOV BX, 00A0h   ; satır 1
5 MOV BYTE PTR ES:[BX], 'K'
6 MOV BYTE PTR ES:[BX+1], 1Eh
7 MOV AL, [BX]     ; DS:00A0h
8 MOV DL, ES:[BX]  ; B800:00A0h
9 RET
```

DS = 0700h

$07000h + 0000h = 07000h$

00h

+0

ekran, satır 1:

ES = B800h

$B8000h + 0000h = B8000h$

20h

07h

+0

+1

AL

00h

DL

00h

ES = **B800h**. DS hâlâ programın kesiminde (0700h): değişkenlere erişim bozulmaz.

Adım 2 / 7

DEĞİŞKENLER

DS = 0700h

ES = **B800h**

BX = 0000h

AL = 00h

DL = 00h

Başlangıçta video hücresi 20h 07h (boşluk, gri) kabul edildi. Önek yalnızca o komutu etkiler; DS değişmez.

Kod İzleme: ES ile Video Belleği

```

1 ORG 100h
2 MOV AX, 0B800h
3 MOV ES, AX      ; ES = video
4 MOV BX, 00A0h   ; satır 1
5 MOV BYTE PTR ES:[BX], 'K'
6 MOV BYTE PTR ES:[BX+1], 1Eh
7 MOV AL, [BX]     ; DS:00A0h
8 MOV DL, ES:[BX]  ; B800:00A0h
9 RET

```

DS = 0700h
07000h + 00A0h = 070A0h

00h
+0

ekran, satır 1:

ES = B800h
B8000h + 00A0h = B80A0h

20h 07h
+0 +1

AL DL
00h 00h

$BX = 00A0h = 160 = (1 \times 80 + 0) \times 2$: 1. satırın ilk hücresi.

Adım 3 / 7

DEĞİŞKENLER

DS = 0700h

ES = B800h

BX = 00A0h

AL = 00h

DL = 00h

Başlangıçta video hücresi 20h 07h (boşluk, gri) kabul edildi. Önek yalnızca o komutu etkiler; DS değişmez.

Kod İzleme: ES ile Video Belleği

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV ES, AX          ; ES = video
4 MOV BX, 00A0h       ; satır 1
5 MOV BYTE PTR ES:[BX], 'K'
6 MOV BYTE PTR ES:[BX+1], 1Eh
7 MOV AL, [BX]        ; DS:00A0h
8 MOV DL, ES:[BX]     ; B800:00A0h
9 RET
```

DS = 0700h

07000h + 00A0h = 070A0h

00h

+0

ekran, satır 1:

K

ES = B800h

B8000h + 00A0h = B80A0h

4Bh

07h

+0

+1

AL

00h

DL

00h

ES: öneki (makine kodunda 26h baytı): B8000h + 00A0h = **B80A0h** ← 'K' (4Bh).

Adım 4 / 7

DEĞİŞKENLER

DS = 0700h

ES = B800h

BX = 00A0h

AL = 00h

DL = 00h

Başlangıçta video hücresi 20h 07h (boşluk, gri) kabul edildi. Önek yalnızca o komutu etkiler; DS değişmez.

Kod İzleme: ES ile Video Belleği

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV ES, AX      ; ES = video
4 MOV BX, 00A0h   ; satır 1
5 MOV BYTE PTR ES:[BX], 'K'
6 MOV BYTE PTR ES:[BX+1], 1Eh
7 MOV AL, [BX]    ; DS:00A0h
8 MOV DL, ES:[BX] ; B800:00A0h
9 RET
```

DS = 0700h

07000h + 00A0h = 070A0h

00h

+0

ekran, satır 1:

K

ES = B800h

B8000h + 00A0h = B80A0h

4Bh

1Eh

+0

+1

AL

00h

DL

00h

B80A1h ← **1Eh**: mavi zemin üzerine sarı yazı. Ekranda sarı bir K belirir.

Adım 5 / 7

DEĞİŞKENLER

DS = 0700h

ES = B800h

BX = 00A0h

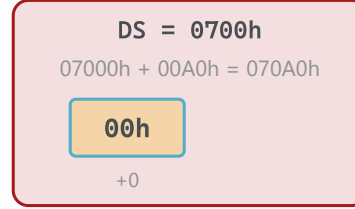
AL = 00h

DL = 00h

Başlangıçta video hücresi 20h 07h (boşluk, gri) kabul edildi. Önek yalnızca o komutu etkiler; DS değişmez.

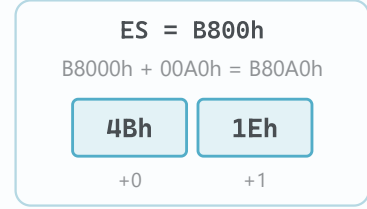
Kod İzleme: ES ile Video Belleği

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV ES, AX          ; ES = video
4 MOV BX, 00A0h       ; satır 1
5 MOV BYTE PTR ES:[BX], 'K'
6 MOV BYTE PTR ES:[BX+1], 1Eh
7 MOV AL, [BX]        ; DS:00A0h
8 MOV DL, ES:[BX]     ; B800:00A0h
9 RET
```



ekran, satır 1:

K



AL

00h

DL

00h

Önek yok → varsayılan **DS**: 07000h + 00A0h = 070A0h okunur (programın PSP alanı, burada 00h). Aynı ofset, başka bayt.

Adım 6 / 7

DEĞİŞKENLER

DS = 0700h

ES = B800h

BX = 00A0h

AL = 00h

DL = 00h

Başlangıçta video hücresi 20h 07h (boşluk, gri) kabul edildi. Önek yalnızca o komutu etkiler; DS değişmez.

Kod İzleme: ES ile Video Belleği

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV ES, AX      ; ES = video
4 MOV BX, 00A0h   ; satır 1
5 MOV BYTE PTR ES:[BX], 'K'
6 MOV BYTE PTR ES:[BX+1], 1Eh
7 MOV AL, [BX]    ; DS:00A0h
8 MOV DL, ES:[BX] ; B800:00A0h
9 RET
```

DS = 0700h

07000h + 00A0h = 070A0h

00h

+0

ekran, satır 1:

K

ES = B800h

B8000h + 00A0h = B80A0h

4Bh

1Eh

+0

+1

AL

00h

DL

4Bh

ES: [BX] yeniden video belleğine gider: DL = **4Bh** ('K').

Adım 7 / 7

DEĞİŞKENLER

DS = 0700h

ES = B800h

BX = 00A0h

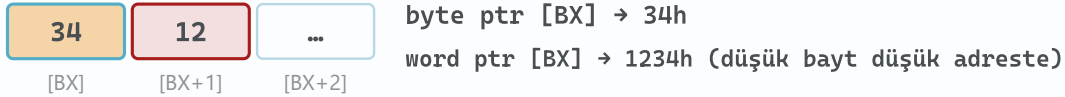
AL = 00h

DL = 4Bh

Başlangıçta video hücresi 20h 07h (boşluk, gri) kabul edildi. Önek yalnızca o komutu etkiler; DS değişmez.

Veri Türü Belirtme

- Derleyiciye erişilecek verinin boyutunu bildirmek için **ön ekler** kullanılır:
 - `byte ptr [BX]` : bir bayt (8 bit) erişimi.
 - `word ptr [BX]` : iki bayt (16 bit, bir sözcük) erişimi.
- emu8086 daha kısa örnekleri de destekler:
 - `b. [SI]` : SI'nın gösterdiği adresteki değeri bir bayt olarak yorumlar.
 - `w. [DI]` : DI'nın gösterdiği adresteki değeri iki bayt olarak yorumlar.
- Bazı durumlarda assembler türü **kendisi** çıkarır: diğer işlenen bir yazmaçsa boyutu odur.



Komut	Boyut nereden belli?
<code>MOV [BX], AL</code>	AL 8 bit → bayt yazılır
<code>MOV [BX], AX</code>	AX 16 bit → sözcük yazılır
<code>MOV [BX], 5</code>	belirsiz! → <code>MOV BYTE PTR [BX], 5</code> ya da <code>MOV WORD PTR [BX], 5</code>

Kod İzleme: byte ptr ve word ptr

► **MOV** BX, 0200h

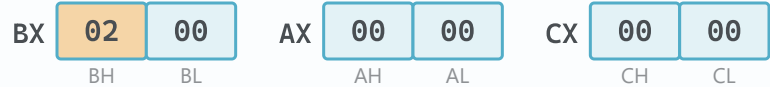
2 **MOV WORD PTR** [BX], 1234h

3 **MOV BYTE PTR** [BX], 0FFh

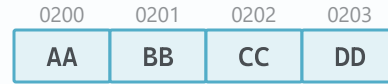
4 **MOV WORD PTR** [BX], 0FFh

5 **MOV** AX, [BX] ; sözcük

6 **MOV** CL, [BX+1] ; bayt



DS:0200 ... DS:0203



BX = 0200h. Bellekte önceki değerler (AA BB CC DD) duruyor.

Adım 1 / 6

DEĞİŞKENLER

BX = 0200h

AX = 0000h

CL = 00h

w[0200] = BBAAh

w[0202] = DDCCh

Kod İzleme: byte ptr ve word ptr

1 **MOV** BX, 0200h

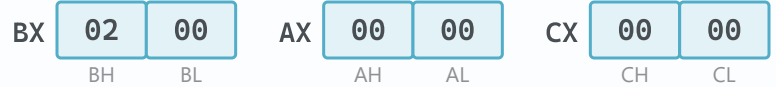
► **MOV WORD PTR** [BX], 1234h

3 **MOV BYTE PTR** [BX], 0FFh

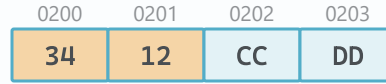
4 **MOV WORD PTR** [BX], 0FFh

5 **MOV** AX, [BX] ; sözcük

6 **MOV** CL, [BX+1] ; bayt



DS:0200 ... DS:0203



word ptr: 2 bayt yazılır. Düşük bayt (34h) düşük adrese: **34 12**.

Adım 2 / 6

DEĞİŞKENLER

BX = 0200h

AX = 0000h

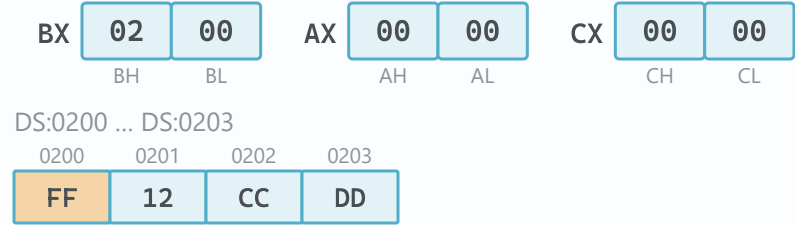
CL = 00h

w[0200] = 1234h

w[0202] = DDCCh

Kod İzleme: byte ptr ve word ptr

```
1 MOV BX, 0200h
2 MOV WORD PTR [BX], 1234h
3 ► MOV BYTE PTR [BX], 0FFh
4 MOV WORD PTR [BX], 0FFh
5 MOV AX, [BX] ; sözcük
6 MOV CL, [BX+1] ; bayt
```



byte ptr: yalnızca 0200h değişir → FF 12. 0201h'ye dokunulmaz.

Adım 3 / 6

DEĞİŞKENLER

BX = 0200h

AX = 0000h

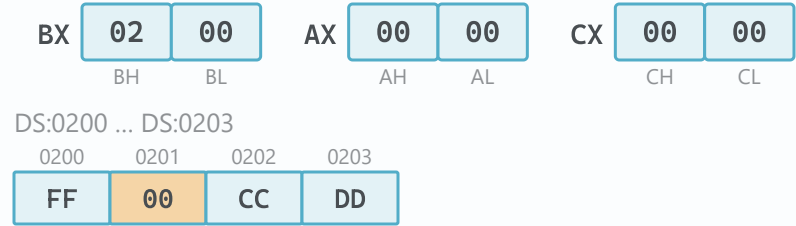
CL = 00h

w[0200] = 12FFh

w[0202] = DDCCh

Kod İzleme: byte ptr ve word ptr

```
1 MOV BX, 0200h
2 MOV WORD PTR [BX], 1234h
3 MOV BYTE PTR [BX], 0FFh
▶ 4 MOV WORD PTR [BX], 0FFh
5 MOV AX, [BX] ; sözcük
6 MOV CL, [BX+1] ; bayt
```



Aynı sabit, **word ptr**: 00FFh yazılır → **FF 00**. 0201h de değişti!

Adım 4 / 6

DEĞİŞKENLER

BX = 0200h

AX = 0000h

CL = 00h

w[0200] = 00FFh

w[0202] = DDCCh

Kod İzleme: byte ptr ve word ptr

```
1 MOV BX, 0200h
2 MOV WORD PTR [BX], 1234h
3 MOV BYTE PTR [BX], 0FFh
4 MOV WORD PTR [BX], 0FFh
▶ 5 MOV AX, [BX] ; sözcük
6 MOV CL, [BX+1] ; bayt
```



AX 16 bit olduğundan boyut belli: AX = **00FFh** (0201h yüksek bayt).

Adım 5 / 6

DEĞİŞKENLER

BX = 0200h

AX = 00FFh

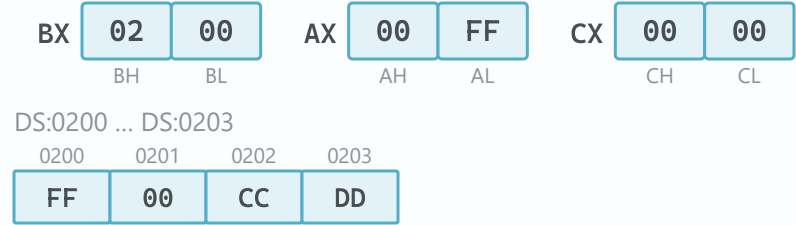
CL = 00h

w[0200] = 00FFh

w[0202] = DDCCh

Kod İzleme: byte ptr ve word ptr

```
1 MOV BX, 0200h
2 MOV WORD PTR [BX], 1234h
3 MOV BYTE PTR [BX], 0FFh
4 MOV WORD PTR [BX], 0FFh
5 MOV AX, [BX] ; sözcük
▶ MOV CL, [BX+1] ; bayt
```



CL 8 bit: [BX+1] = 0201h'deki bayt → CL = **00h**.

Adım 6 / 6

DEĞİŞKENLER

BX = 0200h

AX = 00FFh

CL = 00h

w[0200] = 00FFh

w[0202] = DDCCh

Little-Endian Düzen

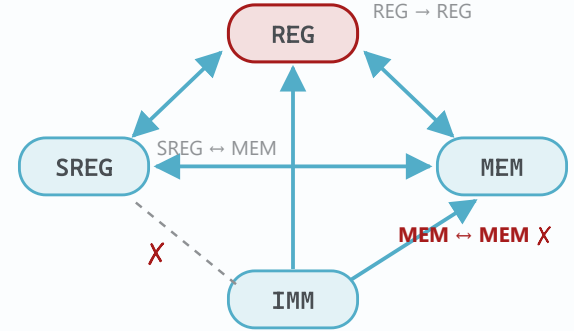


- 8086 **little-endian**dir: 16 bitlik bir değerin **düşük baytı düşük adreste**, yüksek baytı bir sonraki adreste saklanır.
- `MOV [0200h], AX` → `[0200h] = AL`, `[0201h] = AH`. `MOV AX, [0200h]` tersini yapar.
- emu8086'nın bellek penceresinde sözcükler bu yüzden "ters" görünür: `1234h` → `34 12`.
- Bu kural makine kodundaki sabitler için de geçerlidir: `MOV AX, 1234h` → `B8 34 12`.

MOV KOMUTU

MOV Komutu

- **MOV hedef, kaynak** : kaynaktaki değeri hedefe **kopyalar**; kaynak değişmez.
- **Kaynak**: anlık değer (sabit), genel amaçlı yazmaç, kesim yazmacı ya da bellek konumu.
- **Hedef**: genel amaçlı yazmaç, kesim yazmacı (CS hariç) ya da bellek konumu.
- **CS ve IP** yazmaçlarına değer atamak için kullanılmaz.
- İki işlenenin **boyutu aynı** olmalıdır: ikisi de bayt ya da ikisi de sözcük.
- İki işlenen birden bellek olamaz.
- MOV **bayrakları etkilemez**.



MOV AX, BX

BX'teki değeri AX'e kopyalar. BX değişmez.

MOV [SI], 10h

SI'nın gösterdiği adrese (DS:SI) 10h değerini yazar. Boyut belirsiz olduğundan doğrusu `MOV BYTE PTR [SI], 10h`.

MOV KOMUTU

Desteklenen İşlenenler

Genel amaçlı

MOV REG, memory
MOV memory, REG
MOV REG, REG
MOV memory, immediate
MOV REG, immediate

Kesim yazmaçları

MOV SREG, memory
MOV memory, SREG
MOV REG, SREG
MOV SREG, REG

REG	AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP
SREG	DS, ES, SS ve CS (CS yalnızca ikinci işlenen, yani kaynak olarak)
memory	[BX] , [BX+SI+7] , değişken...
immediate	5 , -24 , 3Fh , 10001101b ...

Dikkat: **MOV SREG, immediate** ve **MOV SREG, SREG** yoktur; değer önce bir genel yazmaca alınır.

MOV KOMUTU

MOV: Geçerli mi?

Komut	?	Komut	?
MOV [BX], [SI]	1	MOV AX, CS	6
MOV DS, ES	2	MOV SP, BX	7
MOV DS, 100h	3	MOV BL, DX	8
MOV CS, AX	4	MOV AL, [SI+BP]	9
MOV [BX], 5	5	MOV 5, AX	10

► Cevaplar

PROGRAM ÖRNEĞİ

Program Örneği

```
ORG 100h          ; .com programı için
MOV AX, 0B800h     ; AX = B800h
MOV DS, AX         ; DS = B800h
MOV CL, 'A'        ; CL = 41h
MOV CH, 11011111b  ; CH = DFh
MOV BX, 15Eh       ; BX = 015Eh
MOV [BX], CX       ; B800:015E ← CX
RET               ; DOS'a dön
```

- Program, **video belleği** üzerinde belirli bir konuma 'A' harfini yazar.
- **MOV** , kaynak ve hedef arasında değer kopyalar.
- **ORG** , programın başlangıç adresini belirler (makine koduna çevrilmez).
- **;** yorum satırları için kullanılır.
- DS'ye sabit doğrudan atanamadığı için B800h önce AX'e yüklenir.

Komut	Makine kodu	Komut	Makine kodu
MOV AX, 0B800h	B8 00 B8	MOV CH, 11011111b	B5 DF
MOV DS, AX	8E D8	MOV BX, 15Eh	BB 5E 01
MOV CL, 'A'	B1 41	MOV [BX], CX · RET	89 0F · C3

Kod İzleme: Ekrana 'A' Yazmak

► **ORG** 100h

```
2 MOV AX, 0B800h
3 MOV DS, AX
4 MOV CL, 'A'
5 MOV CH, 11011111b
6 MOV BX, 15Eh
7 MOV [BX], CX
8 RET
```

AX

00	00
AH	AL

 CX

00	00
CH	CL

 BX

00	00
BH	BL

DS

0B56

B800:015C ... 015F (video belleği)

015C	015D	015E	015F
20	07	20	07

Ekran boş: her hücre **20h** (boşluk) + **07h** (siyah zemin üzerine gri yazı).

Adım 1 / 8

DEĞİŞKENLER

AX = 0000h

DS = 0B56h

CH = 00h

CL = 00h

BX = 0000h

[B800:015E] = 20h

[B800:015F] = 07h

Kod İzleme: Ekrana 'A' Yazmak

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV DS, AX
4 MOV CL, 'A'
5 MOV CH, 11011111b
6 MOV BX, 15Eh
7 MOV [BX], CX
8 RET
```

AX **B8** 00 AH AL CX 00 00 CH CL BX 00 00 BH BL

DS **0B56**

B800:015C ... 015F (video belleği)

015C	015D	015E	015F
20	07	20	07

AX = **B800h** (renkli metin ekranının kesimi).

Adım 2 / 8

DEĞİŞKENLER

AX = **B800h**

DS = **0B56h**

CH = **00h**

CL = **00h**

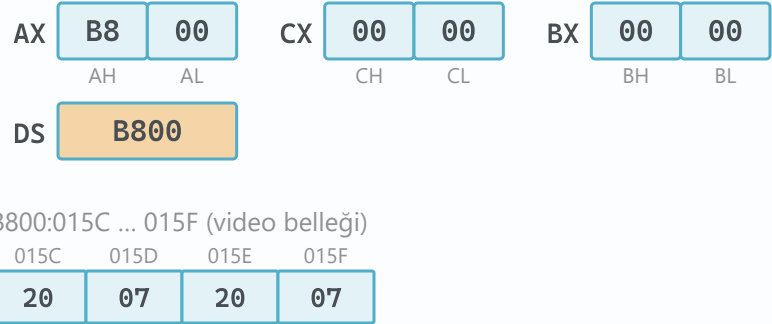
BX = **0000h**

[B800:015E] = **20h**

[B800:015F] = **07h**

Kod İzleme: Ekrana 'A' Yazmak

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV DS, AX
4 MOV CL, 'A'
5 MOV CH, 11011111b
6 MOV BX, 15Eh
7 MOV [BX], CX
8 RET
```



DS = AX = **B800h**. Artık [] erişimleri video belleğine gider.

Adım 3 / 8

DEĞİŞKENLER

AX = B800h DS = B800h CH = 00h CL = 00h BX = 0000h

[B800:015E] = 20h [B800:015F] = 07h

Kod İzleme: Ekrana 'A' Yazmak

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV DS, AX
4 ► MOV CL, 'A'
5 MOV CH, 11011111b
6 MOV BX, 15Eh
7 MOV [BX], CX
8 RET
```



CL = 'A' = **41h** (ASCII kodu).

Adım 4 / 8

DEĞİŞKENLER

AX = B800h DS = B800h CH = 00h **CL = 41h** BX = 0000h

[B800:015E] = 20h [B800:015F] = 07h

Kod İzleme: Ekrana 'A' Yazmak

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV DS, AX
4 MOV CL, 'A'
5 MOV CH, 11011111b
6 MOV BX, 15Eh
7 MOV [BX], CX
8 RET
```



CH = 11011111b = **DFh**: renk (öznitelik) baytı.

Adım 5 / 8

DEĞİŞKENLER

AX = B800h DS = B800h CH = DFh CL = 41h BX = 0000h

[B800:015E] = 20h [B800:015F] = 07h

Kod İzleme: Ekrana 'A' Yazmak

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV DS, AX
4 MOV CL, 'A'
5 MOV CH, 11011111b
6 MOV BX, 15Eh
7 MOV [BX], CX
8 RET
```



BX = **015Eh** = 350: ekrandaki konumun bağlı adresi.

Adım 6 / 8

DEĞİŞKENLER

AX = B800h

DS = B800h

CH = DFh

CL = 41h

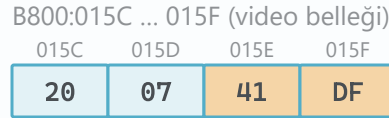
BX = 015Eh

[B800:015E] = 20h

[B800:015F] = 07h

Kod İzleme: Ekran'a 'A' Yazmak

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV DS, AX
4 MOV CL, 'A'
5 MOV CH, 11011111b
6 MOV BX, 15Eh
7 MOV [BX], CX
8 RET
```



CX = DF41h sözcük olarak yazılır: [015Eh] = **41h** ('A'), [015Fh] = **DFh** (renk). Fiziksel: B8000h + 15Eh = **B815Eh**.

Adım 7 / 8

DEĞİŞKENLER

AX = B800h DS = B800h CH = DFh CL = 41h BX = 015Eh

[B800:015E] = 41h

[B800:015F] = DFh

Kod İzleme: Ekrana 'A' Yazmak

```
1 ORG 100h
2 MOV AX, 0B800h
3 MOV DS, AX
4 MOV CL, 'A'
5 MOV CH, 11011111b
6 MOV BX, 15Eh
7 MOV [BX], CX
RET
```



RET: DOS'a dönülür; 'A' harfi ekranda kalır.

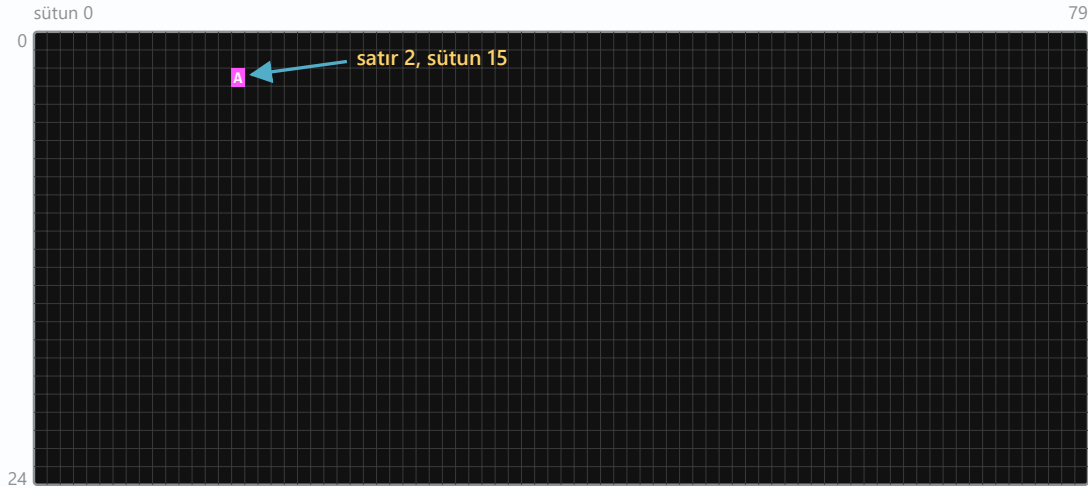
Adım 8 / 8

DEĞİŞKENLER

AX = B800h DS = B800h CH = DFh CL = 41h BX = 015Eh

[B800:015E] = 41h [B800:015F] = DFh

Metin Modu Video Belleği



Konum

80×25 metin ekranında her hücre **2 bayttır**:
karakter + renk. Bağlı adres = (satır × 80 +
sütun) × 2 → (2 × 80 + 15) × 2 = 350 = **15Eh**.

Renk baytı DFh = 1101 1111b

Bit 7: yanıp sönme (emu8086'da parlak
zemin) · bit 6–4 = 101: macenta zemin · bit 3–
0 = 1111: parlak beyaz yazı.

Bellekten Bir Değeri Yazmaca Taşıma

► **ORG** 100h

2 **MOV** AX, [1000h] ; A1 00 10

3 **RET**

AX

00

00

IP

0100

AH

AL

DS:1000 ...

1000

1001

1002

CD

AB

00

DS = 0700h. DS:1000h'de **CD AB** baytları var. IP = 0100h.

Adım 1 / 3

DEĞİŞKENLER

AX = 0000h

IP = 0100h

w[1000h] = ABCDh

Bellekten Bir Değeri Yazmaca Taşıma

1 **ORG** 100h

▶ **MOV** AX, [1000h] ; A1 00 10

3 **RET**

AX

AB

CD

IP

0103

AH

AL

DS:1000 ...

1000

1001

1002

CD

AB

00

MOV AX, [1000h] : iki bayt okunur, little-endian → AX = **ABCDh**. Komut 3 bayt → IP = 0103h.

Adım 2 / 3

DEĞİŞKENLER

AX = **ABCDh**

IP = **0103h**

w[1000h] = **ABCDh**

Bellekten Bir Değeri Yazmaca Taşıma

```
1 ORG 100h
2 MOV AX, [1000h] ; A1 00 10
▶ RET
```

AX

AB	CD
AH	AL

 IP

0103

DS:1000 ...

1000	1001	1002
CD	AB	00

Bellek değişmedi; MOV yalnızca **kopyalar**. Fiziksel adres: 07000h + 1000h = 08000h.

Adım 3 / 3

DEĞİŞKENLER

AX = ABCDh

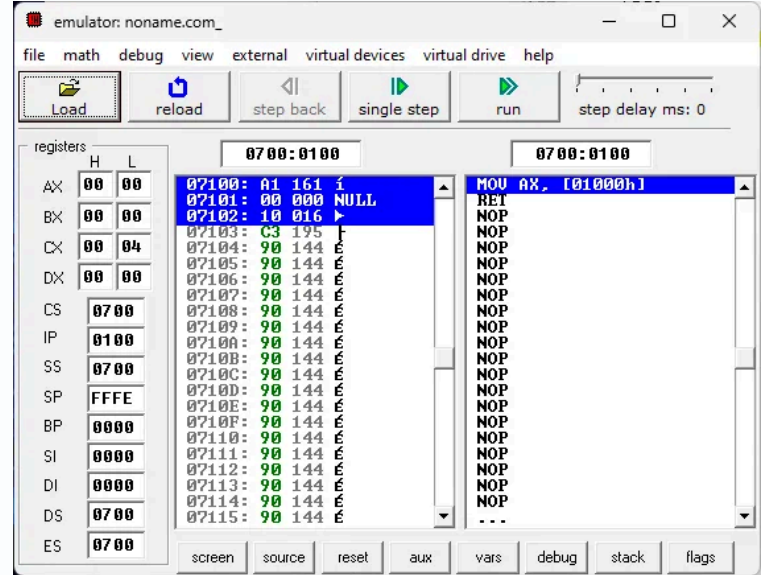
IP = 0103h

w[1000h] = ABCDh

MOV ÖRNEKLERİ

emu8086'da Görünüm

- Sol: yazmaçlar (H/L yarıları ayrı). CS = DS = 0700h, IP = 0100h, SP = FFFEh.
- Orta: bellek (fiziksel adres 07100h = 0700:0100). A1 00 10 komutun 3 baytıdır.
- Sağ: ayrıştırılmış (disassembled) kod: MOV AX, [01000h], ardından RET (C3).
- single step** ile komut yürütülür; değişen yazmaçlar renklenir.
- Bellek boş olduğundan bu ekranda AX = 0000h kalır.



emu8086 öykünücü penceresi

Yazmaçtaki Değeri Belleğe Taşıma

► **ORG** 100h

```
2 MOV BX, 1234h      ; BB 34 12
3 MOV [2000h], BX    ; 89 1E 00 20
4 RET
```

BX

00

00

IP

0100

BH

BL

DS:2000 ...

2000

2001

2002

00

00

00

DS:2000h boş (00 00).

Adım 1 / 4

DEĞİŞKENLER

BX = 0000h

IP = 0100h

w[2000h] = 0000h

Yazmaçtaki Değeri Belleğe Taşıma

1 **ORG** 100h

▶ **MOV** BX, 1234h ; BB 34 12

3 **MOV** [2000h], BX ; 89 1E 00 20

4 **RET**

BX

12

34

IP

0103

BH

BL

DS:2000 ...

2000

2001

2002

00

00

00

BX = **1234h**; IP = 0103h.

Adım 2 / 4

DEĞİŞKENLER

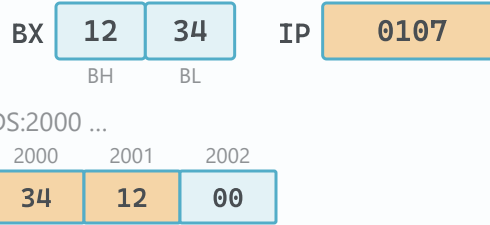
BX = **1234h**

IP = **0103h**

w[2000h] = **0000h**

Yazmaçtaki Değeri Belleğe Taşıma

```
1 ORG 100h
2 MOV BX, 1234h ; BB 34 12
▶ MOV [2000h], BX ; 89 1E 00 20
4 RET
```



BX belleğe yazılır: [2000h] = **34h**, [2001h] = **12h**. Komut 4 bayt
→ IP = 0107h.

Adım 3 / 4

DEĞİŞKENLER

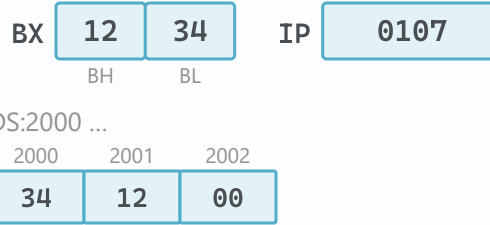
BX = **1234h**

IP = **0107h**

w[2000h] = **1234h**

Yazmaçtaki Değeri Belleğe Taşıma

```
1 ORG 100h
2 MOV BX, 1234h      ; BB 34 12
3 MOV [2000h], BX    ; 89 1E 00 20
▶ RET
```



BX değişmedi. Fiziksel adres: 07000h + 2000h = **09000h**.

Adım 4 / 4

DEĞİŞKENLER

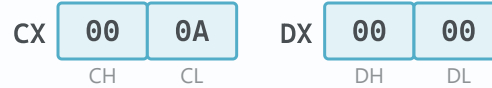
BX = 1234h

IP = 0107h

w[2000h] = 1234h

İki Yazmaç Değerini Takas

```
► MOV CX, 10 ; CX = 000Ah  
2 MOV DX, 20 ; DX = 0014h  
3 XCHG CX, DX ; takas
```



CX = 10 = **000Ah** (sayılar varsayılan olarak onluktur).

Adım 1 / 3

DEĞİŞKENLER

CX = 000Ah

DX = 0000h

İki Yazmaç Değerini Takas

1 **MOV** CX, 10 ; CX = 000Ah

▶ **MOV** DX, 20 ; DX = 0014h

3 **XCHG** CX, DX ; takas

CX

00

0A

CH

CL

DX

00

14

DH

DL

DX = 20 = **0014h**.

Adım 2 / 3

DEĞİŞKENLER

CX = 000Ah

DX = 0014h

İki Yazmaç Değerini Takas

1 **MOV** CX, 10 ; CX = 000Ah

2 **MOV** DX, 20 ; DX = 0014h

► **XCHG** CX, DX ; takas

CX

00

CH

14

CL

DX

00

DH

0A

DL

XCHG iki işleneni **tek komutta** değiştirir: CX = 0014h, DX = 000Ah. Bayraklar etkilenmez.

Adım 3 / 3

DEĞİŞKENLER

CX = 0014h

DX = 000Ah

XCHG Kuralları

XCHG hedef, kaynak

- Yazmaç ↔ yazmaç: `XCHG AX, BX`
- Yazmaç ↔ bellek: `XCHG AL, [SI]`
- Bellek ↔ bellek **yok**.
- Kesim yazmaçları ve sabitler kullanılamaz.
- Boyutlar aynı olmalı; bayraklar değişmez.
- `XCHG AX, reg` tek baytlık komuttur
(`XCHG AX, AX` = `NOP` = 90h).

XCHG olmasaydı takas için **geçici** bir yazmaç gerekirdi:

```
MOV AX, CX    ; AX ← CX (yedek)
MOV CX, DX    ; CX ← DX
MOV DX, AX    ; DX ← eski CX
```

3 komut ve bir yazmaç daha harcanır; XCHG bunu tek komutta yapar.

Kod İzleme: XCHG ile Takas

```
1 ORG 100h
2 MOV SI, OFFSET sayi
3 MOV AL, 5Ah
4 XCHG AL, [SI]      ; AL ↔ [SI]
5 MOV CX, 1111h
6 MOV DX, 2222h
7 MOV AX, CX          ; yedek
8 MOV CX, DX
9 MOV DX, AX          ; takas
10 XCHG CX, DX        ; tek komut
11 RET
12 sayi DB 0A5h
```

AX	CX	DX	sayi · 0116h
0000h	0000h	0000h	A5h

SI = 0116h: **sayi** 'nın adresi; bellekte sayi = **A5h**.

Adım 1 / 9

DEĞİŞKENLER

SI = 0116h

AX = 0000h

CX = 0000h

DX = 0000h

sayi = A5h

XCHG bayrakları değiştirmez; bellek ↔ bellek ve kesim yazmacı ile takas yoktur.

Kod İzleme: XCHG ile Takas

```
1 ORG 100h
2 MOV SI, OFFSET sayi
3 MOV AL, 5Ah
4 XCHG AL, [SI]      ; AL ↔ [SI]
5 MOV CX, 1111h
6 MOV DX, 2222h
7 MOV AX, CX          ; yedek
8 MOV CX, DX
9 MOV DX, AX          ; takas
10 XCHG CX, DX        ; tek komut
11 RET
12 sayi DB 0A5h
```

AX	CX	DX	sayi · 0116h
005Ah	0000h	0000h	A5h

AL = 5Ah.

Adım 2 / 9

DEĞİŞKENLER

SI = 0116h

AX = 005Ah

CX = 0000h

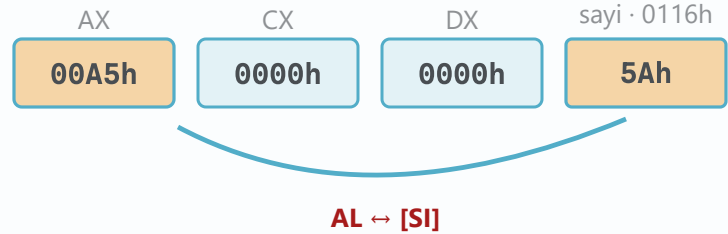
DX = 0000h

sayi = A5h

XCHG bayrakları değiştirmez; bellek ↔ bellek ve kesim yazmacı ile takas yoktur.

Kod İzleme: XCHG ile Takas

```
1 ORG 100h
2 MOV SI, OFFSET sayi
3 MOV AL, 5Ah
4 ► XCHG AL, [SI] ; AL ↔ [SI]
5 MOV CX, 1111h
6 MOV DX, 2222h
7 MOV AX, CX ; yedek
8 MOV CX, DX
9 MOV DX, AX ; takas
10 XCHG CX, DX ; tek komut
11 RET
12 sayi DB 0A5h
```



XCHG AL, [SI] : AL ↔ bellek. AL = **A5h**, sayi = **5Ah**. Ara yazmaç gerekmedi.

Adım 3 / 9

DEĞİŞKENLER

SI = 0116h

AX = 00A5h

CX = 0000h

DX = 0000h

sayi = 5Ah

XCHG bayrakları değiştirmez; bellek ↔ bellek ve kesim yazmacı ile takas yoktur.

Kod İzleme: XCHG ile Takas

```
1 ORG 100h
2 MOV SI, OFFSET sayi
3 MOV AL, 5Ah
4 XCHG AL, [SI] ; AL ↔ [SI]
5 MOV CX, 1111h
6 MOV DX, 2222h
7 MOV AX, CX ; yedek
8 MOV CX, DX
9 MOV DX, AX ; takas
10 XCHG CX, DX ; tek komut
11 RET
12 sayi DB 0A5h
```

AX	CX	DX	sayi · 0116h
00A5h	1111h	0000h	5Ah

CX = 1111h.

Adım 4 / 9

DEĞİŞKENLER

SI = 0116h

AX = 00A5h

CX = 1111h

DX = 0000h

sayi = 5Ah

XCHG bayrakları değiştirmez; bellek ↔ bellek ve kesim yazmacı ile takas yoktur.

Kod İzleme: XCHG ile Takas

```
1 ORG 100h
2 MOV SI, OFFSET sayi
3 MOV AL, 5Ah
4 XCHG AL, [SI]      ; AL ↔ [SI]
5 MOV CX, 1111h
6 MOV DX, 2222h
7 MOV AX, CX          ; yedek
8 MOV CX, DX
9 MOV DX, AX          ; takas
10 XCHG CX, DX       ; tek komut
11 RET
12 sayi DB 0A5h
```

AX	CX	DX	sayi · 0116h
00A5h	1111h	2222h	5Ah

DX = 2222h. Şimdi aynı takas XCHG olmadan yapılacaktır.

Adım 5 / 9

DEĞİŞKENLER

SI = 0116h

AX = 00A5h

CX = 1111h

DX = 2222h

sayi = 5Ah

XCHG bayrakları değiştirmez; bellek ↔ bellek ve kesim yazmacı ile takas yoktur.

Kod İzleme: XCHG ile Takas

```
1 ORG 100h
2 MOV SI, OFFSET sayi
3 MOV AL, 5Ah
4 XCHG AL, [SI]      ; AL ↔ [SI]
5 MOV CX, 1111h
6 MOV DX, 2222h
7 ► MOV AX, CX      ; yedek
8 MOV CX, DX
9 MOV DX, AX        ; takas
10 XCHG CX, DX      ; tek komut
11 RET
12 sayi DB 0A5h
```

AX	CX	DX	sayi · 0116h
1111h	1111h	2222h	5Ah

1. AX ← CX

3 komut + yardımcı yazmaç

MOV AX, CX : CX'in yedeği AX'te (AX'in eski değeri kayboldu).

Adım 6 / 9

DEĞİŞKENLER

SI = 0116h

AX = 1111h

CX = 1111h

DX = 2222h

sayi = 5Ah

XCHG bayrakları değiştirmez; bellek ↔ bellek ve kesim yazmacı ile takas yoktur.

Kod İzleme: XCHG ile Takas

```
1 ORG 100h
2 MOV SI, OFFSET sayi
3 MOV AL, 5Ah
4 XCHG AL, [SI]      ; AL ↔ [SI]
5 MOV CX, 1111h
6 MOV DX, 2222h
7 MOV AX, CX          ; yedek
8 MOV CX, DX          ; takas
9 MOV DX, AX          ; tek komut
10 XCHG CX, DX
11 RET
12 sayi DB 0A5h
```

AX	CX	DX	sayi · 0116h
1111h	2222h	2222h	5Ah

2. CX ← DX

3 komut + yardımcı yazmaç

MOV CX, DX : CX = 2222h. CX'in eski değeri yalnızca AX'te duruyor.

Adım 7 / 9

DEĞİŞKENLER

SI = 0116h

AX = 1111h

CX = 2222h

DX = 2222h

sayi = 5Ah

XCHG bayrakları değiştirmez; bellek ↔ bellek ve kesim yazmacı ile takas yoktur.

Kod İzleme: XCHG ile Takas

```
1 ORG 100h
2 MOV SI, OFFSET sayi
3 MOV AL, 5Ah
4 XCHG AL, [SI]      ; AL ↔ [SI]
5 MOV CX, 1111h
6 MOV DX, 2222h
7 MOV AX, CX          ; yedek
8 MOV CX, DX
9 ► MOV DX, AX         ; takas
10 XCHG CX, DX        ; tek komut
11 RET
12 sayi DB 0A5h
```

AX	CX	DX	sayi · 0116h
1111h	2222h	1111h	5Ah

3. DX ← AX

3 komut + yardımcı yazmaç

`MOV DX, AX` : DX = 1111h. Takas 3 komut ve bir yazmaç harcadı.

Adım 8 / 9

DEĞİŞKENLER

SI = 0116h

AX = 1111h

CX = 2222h

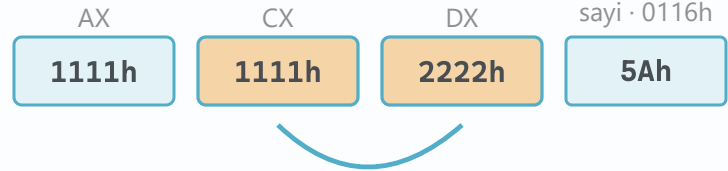
DX = 1111h

sayi = 5Ah

XCHG bayrakları değiştirmez; bellek ↔ bellek ve kesim yazmacı ile takas yoktur.

Kod İzleme: XCHG ile Takas

```
1 ORG 100h
2 MOV SI, OFFSET sayi
3 MOV AL, 5Ah
4 XCHG AL, [SI]      ; AL ↔ [SI]
5 MOV CX, 1111h
6 MOV DX, 2222h
7 MOV AX, CX          ; yedek
8 MOV CX, DX
9 MOV DX, AX          ; takas
▶ XCHG CX, DX        ; tek komut
11 RET
12 sayi DB 0A5h
```



CX ↔ DX: 1 komut

XCHG CX, DX : tek komutta geri takas: CX = **1111h**, DX = **2222h**; AX dokunulmadan kaldı.

Adım 9 / 9

DEĞİŞKENLER

SI = 0116h

AX = 1111h

CX = 1111h

DX = 2222h

sayi = 5Ah

XCHG bayrakları değiştirmez; bellek ↔ bellek ve kesim yazmacı ile takas yoktur.

Bir Yazmaca Başka Yazmacın İçeriğini Kopyalama

► **MOV** SI, 3000h ; SI = adres

2 **MOV** DI, SI ; DI = SI

3 **MOV** AL, [SI] ; DS:3000h

4 **MOV** [DI+1], AL ; DS:3001h

SI 3000

DI 0000

AX 00 00
AH AL

DS:3000 ...

3000	3001	3002
5A	00	00

SI'ya bir **bellek adresi** (bağlı konum) atanır: 3000h.

Adım 1 / 4

DEĞİŞKENLER

SI = 3000h

DI = 0000h

AL = 00h

Bir Yazmaca Başka Yazmacın İçeriğini Kopyalama

1 **MOV** SI, 3000h ; SI = adres

► **MOV** DI, SI ; DI = SI

3 **MOV** AL, [SI] ; DS:3000h

4 **MOV** [DI+1], AL ; DS:3001h

SI 3000

DI 3000

AX 00 00
AH AL

DS:3000 ...

3000	3001	3002
5A	00	00

DI = SI = **3000h**. Kopyalanan şey adrestir, bellekteki veri değil.

Adım 2 / 4

DEĞİŞKENLER

SI = 3000h

DI = 3000h

AL = 00h

Bir Yazmaca Başka Yazmacın İçeriğini Kopyalama

```
1 MOV SI, 3000h ; SI = adres
2 MOV DI, SI    ; DI = SI
3 ► MOV AL, [SI] ; DS:3000h
4 MOV [DI+1], AL ; DS:3001h
```

SI 3000

DI 3000

AX 00

5A

AH

AL

DS:3000 ...

3000

3001

3002

5A

00

00

Artık SI ve DI aynı hücreyi gösterir: AL = [3000h] = **5Ah**.

Adım 3 / 4

DEĞİŞKENLER

SI = 3000h

DI = 3000h

AL = 5Ah

Bir Yazmaca Başka Yazmacın İçeriğini Kopyalama

```
1 MOV SI, 3000h ; SI = adres
2 MOV DI, SI    ; DI = SI
3 MOV AL, [SI]  ; DS:3000h
▶ MOV [DI+1], AL ; DS:3001h
```

SI 3000

DI 3000

AX 00

5A

AH

AL

DS:3000 ...

3000

3001

3002

5A

5A

00

[DI+1] = 3001h'ye AL yazılır → **5A 5A**.

Adım 4 / 4

DEĞİŞKENLER

SI = 3000h

DI = 3000h

AL = 5Ah

Bir Bellek Adresine Sabit Bir Değeri Yazma

► **ORG 100h**

2 **MOV BYTE PTR** [4000h], 65 ; 'A'

3 **MOV WORD PTR** [4002h], 65

4 **RET**

IP **0100**

DS:4000 ...

4000	4001	4002	4003
00	99	00	99

DS:4000h ... 4003h: 00 99 00 99.

Adım 1 / 4

DEĞİŞKENLER

IP = **0100h**

[4000h] = **00h**

[4003h] = **99h**

Bir Bellek Adresine Sabit Bir Değeri Yazma

```
1 ORG 100h
```

```
▶ MOV BYTE PTR [4000h], 65 ; 'A'
```

```
3 MOV WORD PTR [4002h], 65
```

```
4 RET
```

IP **0105**

DS:4000 ...

4000	4001	4002	4003
41	99	00	99

65 = 41h = 'A' tek bayt olarak yazılır. Makine kodu **C6 06 00 40 41** (5 bayt) → IP = 0105h. [4001h] değişmez.

Adım 2 / 4

DEĞİŞKENLER

IP = **0105h**

[4000h] = **41h**

[4003h] = **99h**

Bir Bellek Adresine Sabit Bir Değeri Yazma

```
1 ORG 100h
2 MOV BYTE PTR [4000h], 65 ; 'A'
3 MOV WORD PTR [4002h], 65
4 RET
```

IP 010B

DS:4000 ...

4000	4001	4002	4003
41	99	41	00

WORD PTR: 0041h yazılır → [4002h] = 41h, [4003h] = **00h** (99h ezildi). Komut 6 bayt → IP = 010Bh.

Adım 3 / 4

DEĞİŞKENLER

IP = 010Bh

[4000h] = 41h

[4003h] = 00h

Bir Bellek Adresine Sabit Bir Değeri Yazma

```
1 ORG 100h
2 MOV BYTE PTR [4000h], 65 ; 'A'
3 MOV WORD PTR [4002h], 65
▶ RET
```

IP 010B

DS:4000 ...

4000	4001	4002	4003
41	99	41	00

Sabitin boyutu yoktur; boyutu **PTR** öneki belirler.

Adım 4 / 4

DEĞİŞKENLER

IP = 010Bh

[4000h] = 41h

[4003h] = 00h

Bir Yazmacın Düşük Baytını Diğerine Kopyalama

► **MOV** AX, 1234h ; AH = 12h, AL = 34h

2 **MOV** BX, 0FFFFh

3 **MOV** BL, AL ; yalnız BL

AX

12

34

AH

AL

BX

00

00

BH

BL

AX = **1234h**: AH = 12h, AL = 34h.

Adım 1 / 3

DEĞİŞKENLER

AX = **1234h**

BX = **0000h**

BL = **00h**

Bir Yazmacın Düşük Baytını Diğerine Kopyalama

1 **MOV** AX, 1234h ; AH = 12h, AL = 34h

► **MOV** BX, 0FFFFh

3 **MOV** BL, AL ; yalnız BL



Etkiyi görmek için BX = FFFFh.

Adım 2 / 3

DEĞİŞKENLER

AX = 1234h

BX = FFFFh

BL = FFh

Bir Yazmacın Düşük Baytını Diğerine Kopyalama

1 **MOV** AX, 1234h ; AH = 12h, AL = 34h

2 **MOV** BX, 0FFFFh

▶ **MOV** BL, AL ; yalnız BL

AX

12

34

AH

AL

BX

FF

34

BH

BL

AL (AX'in **düşük baytı**) BL'ye kopyalanır: BL = 34h. BH değişmez
→ BX = **FF34h**.

Adım 3 / 3

DEĞİŞKENLER

AX = 1234h

BX = FF34h

BL = 34h

Bir Yazmacın Yüksek Baytına Ekleme

► **MOV** AX, 00FFh ; AH = 00h, AL = FFh

2 **ADD** AH, 01h ; yalnız AH

3 **ADD** AL, 01h ; yalnız AL

4 **MOV** AX, 01FFh

5 **ADD** AX, 1 ; 16 bit

AX 00 FF
 AH AL

Bayraklar

CF=0

ZF=0

SF=0

OF=0

PF=0

AF=0

AX = **00FFh**.

Adım 1 / 5

DEĞİŞKENLER

AX = **00FFh**

AH = **00h**

AL = **FFh**

CF = **0**

ZF = **0**

SF = **0**

OF = **0**

PF = **0**

AF = **0**

Bir Yazmacın Yüksek Baytına Ekleme

```
1 MOV AX, 00FFh ; AH = 00h, AL = FFh
```

```
▶ ADD AH, 01h ; yalnız AH
```

```
3 ADD AL, 01h ; yalnız AL
```

```
4 MOV AX, 01FFh
```

```
5 ADD AX, 1 ; 16 bit
```

AX 01 FF
 AH AL

Bayraklar

CF=0

ZF=0

SF=0

OF=0

PF=0

AF=0

AH = 00h + 01h = 01h → AX = **01FFh**. Sonuç sıfır değil, elde yok.

Adım 2 / 5

DEĞİŞKENLER

AX = 01FFh

AH = 01h

AL = FFh

CF = 0

ZF = 0

SF = 0

OF = 0

PF = 0

AF = 0

Bir Yazmacın Yüksek Baytına Ekleme

```
1 MOV AX, 00FFh ; AH = 00h, AL = FFh
```

```
2 ADD AH, 01h ; yalnız AH
```

```
▶ ADD AL, 01h ; yalnız AL
```

```
4 MOV AX, 01FFh
```

```
5 ADD AX, 1 ; 16 bit
```

AX 01 00
 AH AL

Bayraklar

CF=1

ZF=1

SF=0

OF=0

PF=1

AF=1

AL = FFh + 1 = 00h, **CF = 1**, ZF = 1. Elde AH'ye **geçmez**: AX = **0100h**.

Adım 3 / 5

DEĞİŞKENLER

AX = 0100h

AH = 01h

AL = 00h

CF = 1

ZF = 1

SF = 0

OF = 0

PF = 1

AF = 1

Bir Yazmacın Yüksek Baytına Ekleme

```
1 MOV AX, 00FFh ; AH = 00h, AL = FFh
2 ADD AH, 01h ; yalnız AH
3 ADD AL, 01h ; yalnız AL
▶ MOV AX, 01FFh
5 ADD AX, 1 ; 16 bit
```

AX **01** **FF**
AH AL

Bayraklar

CF=1

ZF=1

SF=0

OF=0

PF=1

AF=1

Karşılaştırma için AX yeniden 01FFh. MOV bayrakları değiştirmez.

Adım 4 / 5

DEĞİŞKENLER

AX = **01FFh**

AH = **01h**

AL = **FFh**

CF = **1**

ZF = **1**

SF = **0**

OF = **0**

PF = **1**

AF = **1**

Bir Yazmacın Yüksek Baytına Ekleme

```
1 MOV AX, 00FFh ; AH = 00h, AL = FFh
2 ADD AH, 01h ; yalnız AH
3 ADD AL, 01h ; yalnız AL
4 MOV AX, 01FFh
▶ ADD AX, 1 ; 16 bit
```

AX 02 00
 AH AL

Bayraklar

CF=0

ZF=0

SF=0

OF=0

PF=1

AF=1

16 bitlik toplama: $01FFh + 1 = 0200h$; elde AL'den AH'ye geçti, CF = 0.

Adım 5 / 5

DEĞİŞKENLER

AX = 0200h

AH = 02h

AL = 00h

CF = 0

ZF = 0

SF = 0

OF = 0

PF = 1

AF = 1

İkili, Sekizlik ve Onaltılık Değer Atama

```
► mov al, 00000101b ; ikili
2 mov bl, 0ah        ; onaltılık
3 mov cl, 10o        ; sekizli
4 add al, bl          ; 5 + 10 = 15
5 sub al, cl          ; 15 - 8 = 7
```

AX 00 05 BX 00 00 CX 00 00
 AH AL BH BL CH CL

Bayraklar

CF=0

ZF=0

SF=0

OF=0

PF=0

AF=0

00000101b = 5 → AL = 05h.

Adım 1 / 5

DEĞİŞKENLER

AL = 05h

BL = 00h

CL = 00h

CF = 0

ZF = 0

SF = 0

OF = 0

PF = 0

AF = 0

İkili, Sekizlik ve Onaltılık Değer Atama

```
1 mov al, 00000101b ; ikili
▶ mov bl, 0ah        ; onaltılık
3 mov cl, 10o        ; sekizli
4 add al, bl          ; 5 + 10 = 15
5 sub al, cl          ; 15 - 8 = 7
```

AX 00 05 BX 00 0A CX 00 00
 AH AL BH BL CH CL

Bayraklar

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

0Ah = **10** → BL = 0Ah. Harfle başladığı için başta 0 var.

Adım 2 / 5

DEĞİŞKENLER

AL = 05h BL = 0Ah CL = 00h CF = 0 ZF = 0
SF = 0 OF = 0 PF = 0 AF = 0

İkili, Sekizlik ve Onaltılık Değer Atama

```
1 mov al, 00000101b ; ikili
2 mov bl, 0ah        ; onaltılık
3 ► mov cl, 10o      ; sekizli
4 add al, bl          ; 5 + 10 = 15
5 sub al, cl          ; 15 - 8 = 7
```

AX	00	05	BX	00	0A	CX	00	08
	AH	AL		BH	BL		CH	CL

Bayraklar

CF=0

ZF=0

SF=0

OF=0

PF=0

AF=0

10o (sekizli) = $1 \times 8 + 0 = 8 \rightarrow \text{CL} = 08\text{h}$.

Adım 3 / 5

DEĞİŞKENLER

AL = 05h

BL = 0Ah

CL = 08h

CF = 0

ZF = 0

SF = 0

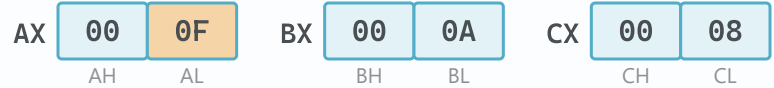
OF = 0

PF = 0

AF = 0

İkili, Sekizlik ve Onaltılık Değer Atama

```
1 mov al, 00000101b ; ikili
2 mov bl, 0ah        ; onaltılık
3 mov cl, 10o        ; sekizli
▶ add al, bl          ; 5 + 10 = 15
5 sub al, cl          ; 15 - 8 = 7
```



Bayraklar



AL = 5 + 10 = 15 = **0Fh**. PF = 1 (0000 1111: dört adet 1).

Adım 4 / 5

DEĞİŞKENLER

AL = 0Fh

BL = 0Ah

CL = 08h

CF = 0

ZF = 0

SF = 0

OF = 0

PF = 1

AF = 0

İkili, Sekizlik ve Onaltılık Değer Atama

```
1 mov al, 00000101b ; ikili
2 mov bl, 0ah        ; onaltılık
3 mov cl, 10o        ; sekizli
4 add al, bl          ; 5 + 10 = 15
▶ sub al, cl          ; 15 - 8 = 7
```

AX 00 07 BX 00 0A CX 00 08
 AH AL BH BL CH CL

Bayraklar

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

AL = 15 - 8 = 7 = **07h**. Ödünç yok → CF = 0; PF = 0 (üç adet 1).

Adım 5 / 5

DEĞİŞKENLER

AL = 07h

BL = 0Ah

CL = 08h

CF = 0

ZF = 0

SF = 0

OF = 0

PF = 0

AF = 0

MOV ÖRNEKLERİ

Sayı Yazımı

Taban	Sonek	Örnek	Onluk değer
Onluk (decimal)	yok ya da <code>d</code>	<code>25</code> , <code>25d</code>	25
İkili (binary)	<code>b</code>	<code>00011001b</code>	25
Sekizli (octal)	<code>o</code> ya da <code>q</code>	<code>31o</code>	25
Onaltılık (hexadecimal)	<code>h</code>	<code>19h</code> , <code>0FFh</code>	25, 255
Karakter	<code>' '</code>	<code>'A'</code>	65 (41h)

Onaltılık sayı bir harfle (A–F) başlıyorsa başına **0** eklenir: `0Ah` , `0B800h` . Aksi hâlde `Ah` , `AH` yazmacıyla; `B800h` bir etiketle karışır. emu8086 ayrıca `0x1000` biçimini de kabul eder.

ÖZET

Özet

Adresleme

[taban + indis + yer değiştirme]; taban BX/BP, indis SI/DI, yer değiştirme d8/d16 — her biri isteğe bağlı.

Fiziksel adres

kesim $\times$ 10h + EA; 1234:7890 $\rightarrow$ 19BD0h.

MOV

Kopyalar, bayrakları etkilemez; bellek $\leftrightarrow$ bellek, kesim $\leftrightarrow$ sabit, CS/IP hedef yok.

Kesim seçimi

BP varsa SS, yoksa DS; önekle değiştirilebilir (ES:[BX]).

Boyut

byte ptr / word ptr (b. / w.); yazmaç işlenen boyutu belirler. Little-endian.

XCHG

İki işleneni tek komutta takas eder; kesim ve sabit yok.

ÖZET

Alıştırmalar

1. DS = 3000h, BX = 0010h, SI = 0005h: `[BX+SI+20h]` için EA ve fiziksel adres?
2. SS = 4000h, BP = FFFh: `[BP+4]` için EA ve fiziksel adres?
3. `[BX+BP]` , `[DI+SI]` , `[BX+DI+300]` , `[DX]` modlarından hangileri geçerli?
4. `[0500h] = 78h`, `[0501h] = 56h` ise `MOV AX, [0500h]` sonrası AX? `MOV AL, [0501h]` sonrası AL?
5. Ekranın satır 10, sütun 40 konumuna yeşil zemin üzerine sarı 'X' yazan kodu yazın (renk 2Eh).
6. DS'yi değiştirmeden ES ile video belleğine yazmak için önceki programı nasıl değiştirirsiniz?

► Cevaplar

Kaynaklar

- emu8086 belgeleri, *8086 assembler tutorial for beginners*, bölüm 2: memory access, MOV instruction.
- Intel, *8086/8088 User's Manual*, adresleme modları ve komut kodlaması.
- Barry B. Brey, *The Intel Microprocessors*, 8. baskı, Pearson, bölüm 3 (Addressing Modes).
- IBM PC metin modu video belleği (B800h) ve öznitelik baytı belgeleri.

Son

Bölüm 4: Bellek Erişimi

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