

Bölüm 5: Kesmeler

Mikroişlemciler

Kesme kavramı

Kesme türleri

Vektör tablosu

INT ve IRET

Alt fonksiyonlar

INT 10h ekran

INT 16h klavye

INT 21h DOS

Örnek: ikili yazdırma

Donanım kesmeleri

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

Öğrenme Çıktıları

Kavram

Kesmenin ne olduğunu, yoklamadan (polling) farkını ve kesme türlerini açıklamak.

Mekanizma

Kesme vektör tablosunu kullanarak `INT n` ile hangi adrese gidileceğini hesaplamak.

Yığın

`INT` ve `IRET` sırasında FLAGS, CS ve IP'nin yığına nasıl yazılıp geri alındığını izlemek.

Uygulama

BIOS (`INT 10h` , `INT 16h`) ve DOS (`INT 21h`) hizmetleriyle ekrana yazan, tuş bekleyen programlar yazmak.

Kesme Nedir?

- **Kesme (interrupt)**, işlemcinin o anki işini geçici olarak bırakıp özel bir yordamı çalıştırmasını sağlayan olaydır.
- Çalıştırılan yordama **kesme hizmet yordamı** denir: **ISR** (interrupt service routine).
- ISR bitince işlemci, kaldığı komuttan devam eder; ana program kesildiğini fark etmez.
- Yazılım kesmeleri, programlamayı kolaylaştıran **hazır hizmetlerdir**: karakter yazdırmak için kod yazmak yerine bir kesme çağrılır.
- Ekran, klavye, disk sürücüsü, bellek yönetimi gibi donanımlarla çalışan kesme işlevleri vardır.
- Kesmeler **donanım** tarafından da tetiklenebilir (klavyeye basılması, zamanlayıcı).

Günlük hayattan

Kitap okurken telefon çalar: sayfayı işaretlersiniz (dönüş adresini saklama), telefonu açarsınız (ISR), kapatınca işaretli sayfadan okumaya devam edersiniz (IRET).



Yoklama mı, Kesme mi?

Yoklama (polling)



CPU sürekli aygıtı sorar; çoğu soru boşa gider, veri geç fark edilebilir.
aygıt: "veri hazır"

Kesme (interrupt)



Yoklama

- Basit, ek donanım gerekmez.
- CPU zamanı boşa harcanır.
- Olay, sorgu aralığına bağlı olarak geç algılanır.

Kesme

- + CPU yalnızca olay olduğunda meşgul edilir.
- + Olaya hızlı tepki verilir.
- + Vektör tablosu, yığın ve ISR düzeni gerekir.

Kesme Türleri

Yazılım kesmesi

Program içindeki **INT n** komutuyla bilerek çağrılır. BIOS ve DOS hizmetleri bu yolla kullanılır. **INT 3** (kesme noktası) ve **INT0** (OF=1 ise INT 4) de yazılım kesmesidir.

Donanım kesmesi

Dış aygıt bir pini sürer. **INTR** maskelenebilir (IF=0 iken dikkate alınmaz), **NMI** maskelenemez (ör. bellek eşlik hatası, güç kaybı).

İç kesme (istisna)

İşlemcinin kendisi üretir: sıfıra bölme / bölüm taşması → **INT 0**, TF=1 iken her komuttan sonra tek adım → **INT 1**.

Tip	Kaynak	Açıklama
00h	iç	Bölme hatası (DIV / IDIV sıfıra bölme veya bölüm sığmıyor)
01h	iç	Tek adım (TF = 1) — hata ayıklayıcılar kullanır
02h	donanım	NMI pini (maskelenemez)
03h	yazılım	Kesme noktası; tek baytlık INT 3 (CCh)
04h	yazılım	INT0 : OF = 1 ise taşıma kesmesi
05h–FFh	—	Intel 05h–1Fh'yi ayırmıştır; PC'de BIOS, DOS ve kullanıcı programları kullanır

Kesme Vektör Tablosu (IVT)

- Belleğin en başındaki **1 KB**: **00000h–003FFh**.
- 256 kesme tipinin her biri için **4 baytlık** bir **vektör** (ISR'nin adresi) tutulur.
- Düşük adresli sözcük **IP** (bağlı konum), yüksek adresli sözcük **CS** (kesim) değeridir; ikisi de little-endian.
- Tip n için vektörün adresi: $n \times 4$.
- Vektörü değiştirmek, kesmeyi başka bir yordama yönlendirmek demektir.

adres (0000:....)	+0, +1	+2, +3	
0000h	IP	CS	INT 00h
0004h	IP	CS	INT 01h
0008h	IP	CS	INT 02h
000Ch	IP	CS	INT 03h
	⋮		
0040h	IP = F065h	CS = F000h	INT 10h
	⋮		
0084h	IP	CS	INT 21h
	⋮		
03FCh	IP	CS	INT FFh

256 giriş × 4 bayt = 1024 bayt (0000:0000 – 0000:03FF)

INT 10h vektörü için gösterilen F000:F065 değeri, orijinal IBM PC BIOS'undaki video yordamının adresidir (örnek).

Vektör Adresi Adım Adım

INT 10h

kesme tipi $n = 10h = 16$

Program `INT 10h` komutunu çalıştırır. Tip numarası **$n = 10h$** (16).

Adım 1 / 4

Vektör Adresi Adım Adım

$$10h \times 4 = 40h$$

$$16 \times 4 = 64 \rightarrow \text{adres } 0000:0040h$$

Vektör adresi **n × 4**: $10h \times 4 = \mathbf{40h}$. (Onaltılıkta $\times 4$, iki bit sola kaydırmaktır.)

Adım 2 / 4

Vektör Adresi Adım Adım

0040h	65h	} IP = F065h
0041h	F0h	
0042h	00h	} CS = F000h
0043h	F0h	

Bu 4 bayt okunur: düşük sözcük **IP = F065h**, yüksek sözcük **CS = F000h** (little-endian: önce düşük bayt).

Vektör Adresi Adım Adım

$$\text{fiziksel adres} = \text{CS} \times 16 + \text{IP}$$

$$\text{F0000h} + \text{F065h} = \text{FF065h}$$

Yığına dönüş bilgisi yazıldıktan sonra CS:IP = F000:F065 yüklenir; ISR, fiziksel **FF065h** adresinden çalışmaya başlar.

Demo: Vektör Hesaplayıcı

Bir kesme tipi girin (00h–FFh): tablodaki konumu ve kesmenin bilinen görevi gösterilir.

INT h

Hesap	$21h \times 4 = 0084h$ ($33 \times 4 = 132$)
IP	0000:0084h ve 0000:0085h
CS	0000:0086h ve 0000:0087h
Görev	DOS: genel hizmetler (AH ile seçilir)

INT 21h → vektör 0000:0084h

Hesap her zaman aynıdır: $\text{adres} = n \times 4$. IP bu adreste, CS ise $n \times 4 + 2$ adresindedir.

INT n Nasıl Çalışır?

1. FLAGS yazmacı yığına itilir: $SP \leftarrow SP - 2$, $[SS:SP] \leftarrow \text{FLAGS}$.

Adım 1 / 6

Yığına toplam **6 bayt** (3 sözcük) yazılır. IF'nin eski değeri FLAGS ile birlikte saklandığı için IRET onu da geri yükler.

INT n Nasıl Çalışır?

1. FLAGS yazmacı yığına itilir: $SP \leftarrow SP - 2$, $[SS:SP] \leftarrow \text{FLAGS}$.
2. **IF = 0** ve **TF = 0** yapılı: ISR başka maskelenebilir kesmelerle ve tek adımla kesilmez.

Adım 2 / 6

Yığına toplam **6 bayt** (3 sözcük) yazılır. IF'nin eski değeri FLAGS ile birlikte saklandığı için IRET onu da geri yükler.

INT n Nasıl Çalışır?

1. FLAGS yazmacı yığına itilir: $SP \leftarrow SP - 2$, $[SS:SP] \leftarrow FLAGS$.
2. **IF = 0** ve **TF = 0** yapılı: ISR başka maskelenebilir kesmelerle ve tek adımla kesilmez.
3. CS yığına itilir.

Adım 3 / 6

Yığına toplam **6 bayt** (3 sözcük) yazılır. IF'nin eski değeri FLAGS ile birlikte saklandığı için IRET onu da geri yükler.

INT n Nasıl Çalışır?

1. FLAGS yazmacı yığına itilir: $SP \leftarrow SP - 2$, $[SS:SP] \leftarrow FLAGS$.
2. **IF = 0** ve **TF = 0** yapılı: ISR başka maskelenebilir kesmelerle ve tek adımla kesilmez.
3. CS yığına itilir.
4. IP (INT'ten *sonraki* komutun adresi) yığına itilir.

Adım 4 / 6

Yığına toplam **6 bayt** (3 sözcük) yazılır. IF'nin eski değeri FLAGS ile birlikte saklandığı için IRET onu da geri yükler.

INT n Nasıl Çalışır?

1. FLAGS yazmacı yığına itilir: $SP \leftarrow SP - 2$, $[SS:SP] \leftarrow FLAGS$.
2. **IF = 0** ve **TF = 0** yapılı: ISR başka maskelenebilir kesmelerle ve tek adımla kesilmez.
3. CS yığına itilir.
4. IP (INT'ten *sonraki* komutun adresi) yığına itilir.
5. $IP \leftarrow [n \times 4]$, $CS \leftarrow [n \times 4 + 2]$: ISR başlar.

Adım 5 / 6

Yığına toplam **6 bayt** (3 sözcük) yazılır. IF'nin eski değeri FLAGS ile birlikte saklandığı için IRET onu da geri yükler.

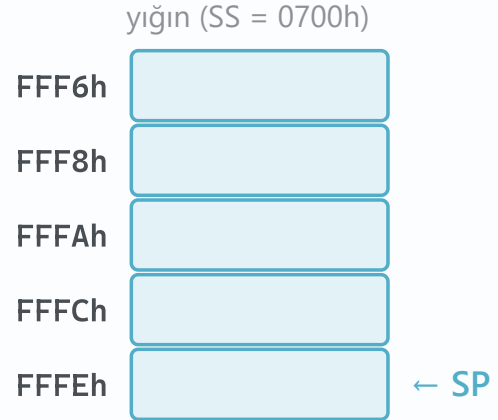
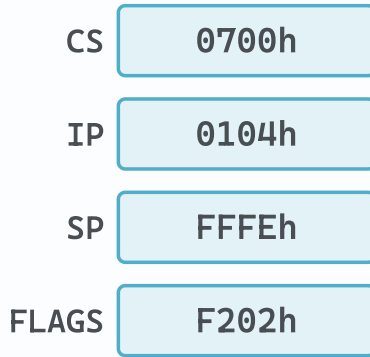
INT n Nasıl Çalışır?

1. FLAGS yazmacı yığına itilir: $SP \leftarrow SP - 2$, $[SS:SP] \leftarrow FLAGS$.
2. **IF = 0** ve **TF = 0** yapılır: ISR başka maskelenebilir kesmelerle ve tek adımla kesilmez.
3. CS yığına itilir.
4. IP (INT'ten *sonraki* komutun adresi) yığına itilir.
5. $IP \leftarrow [n \times 4]$, $CS \leftarrow [n \times 4 + 2]$: ISR başlar.
6. ISR sonunda **IRET**: IP, CS ve FLAGS sırasıyla yığından geri alınır, program kaldığı yerden sürer.

Adım 6 / 6

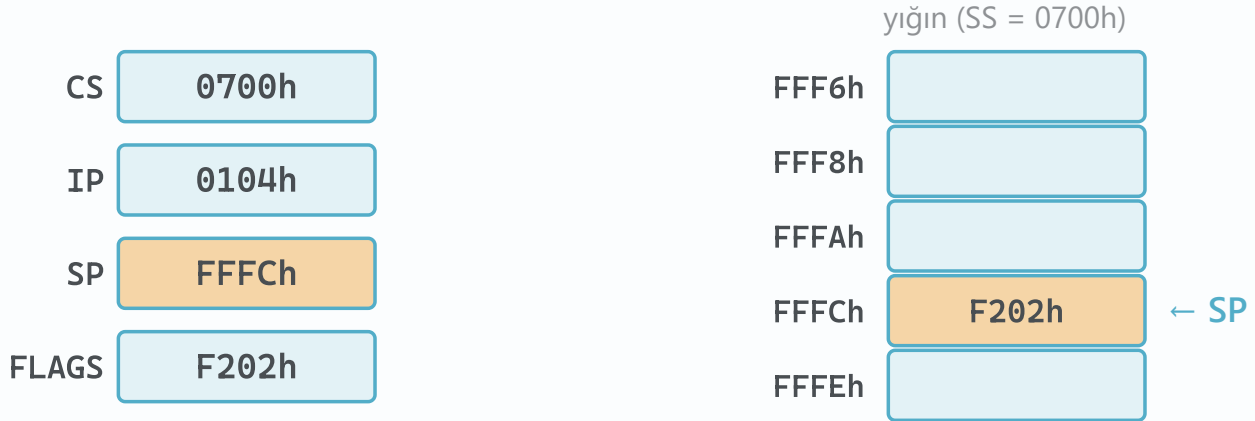
Yığına toplam **6 bayt** (3 sözcük) yazılır. IF'nin eski değeri FLAGS ile birlikte saklandığı için IRET onu da geri yükler.

INT ve IRET: Yığın Adım Adım



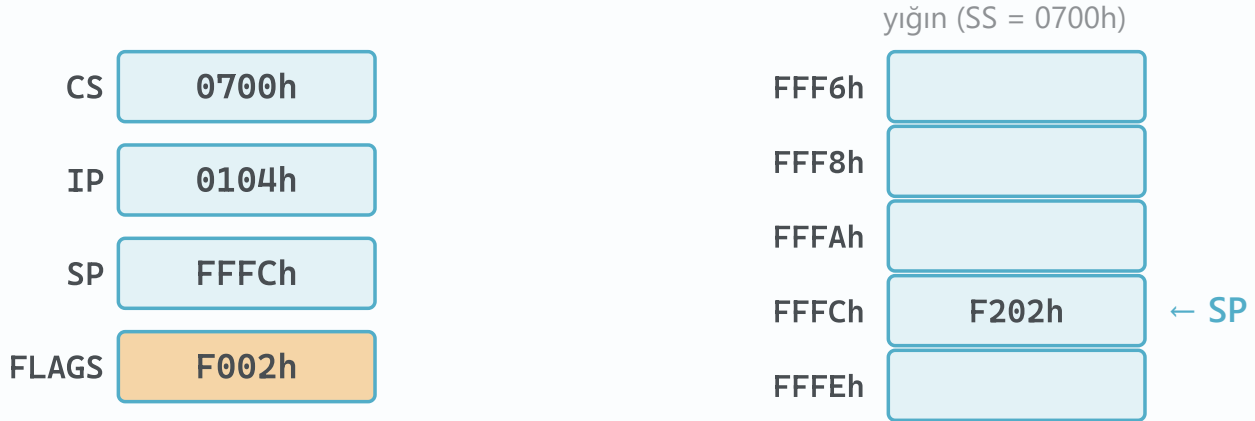
INT 10h 0700:0104'te. SP = FFFEh, FLAGS = F202h (IF = 1; 8086'da bit 1 ve 12–15. bitler hep 1'dir).

INT ve IRET: Yığın Adım Adım



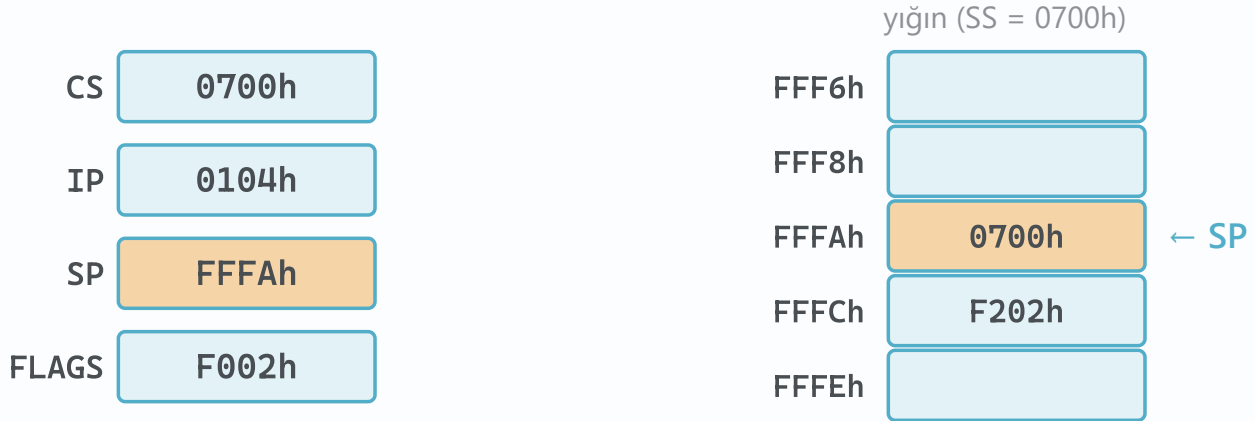
FLAGS itilir: SP = FFFCh, [FFFCh] = F202h.

INT ve IRET: Yığın Adım Adım



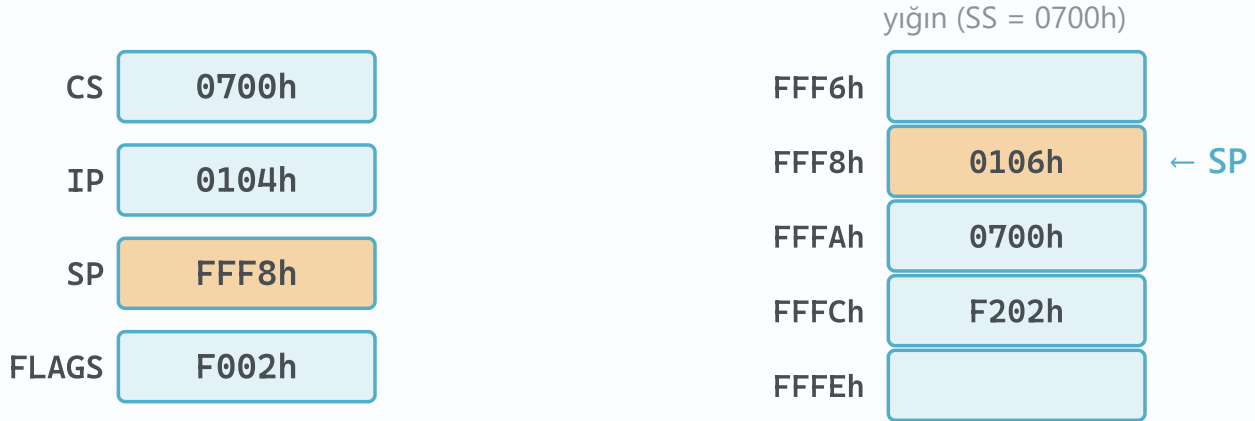
IF = 0, TF = 0 → FLAGS = F002h. Eski değer yığında güvende.

INT ve IRET: Yığın Adım Adım



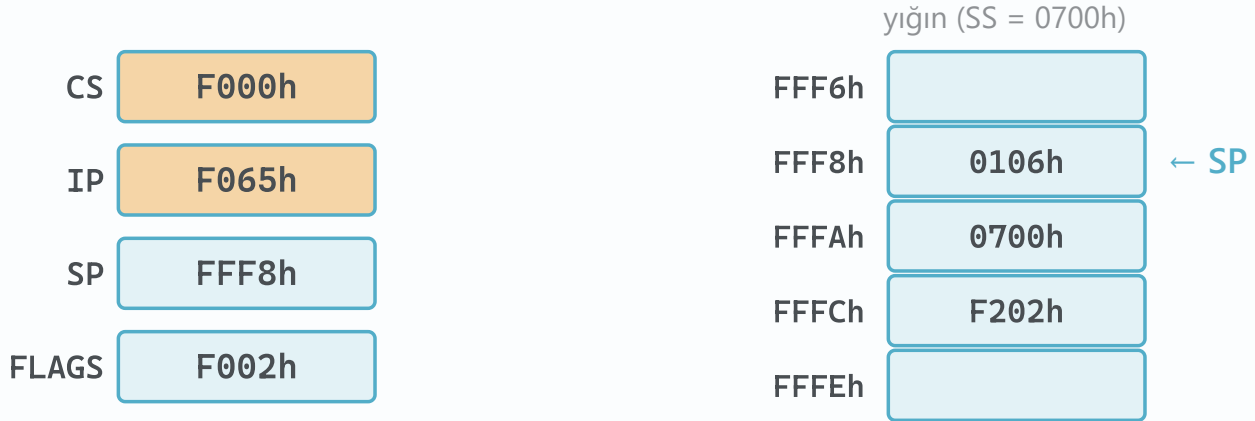
CS itilir: SP = FFFAh, [FFFAh] = 0700h.

INT ve IRET: Yığın Adım Adım



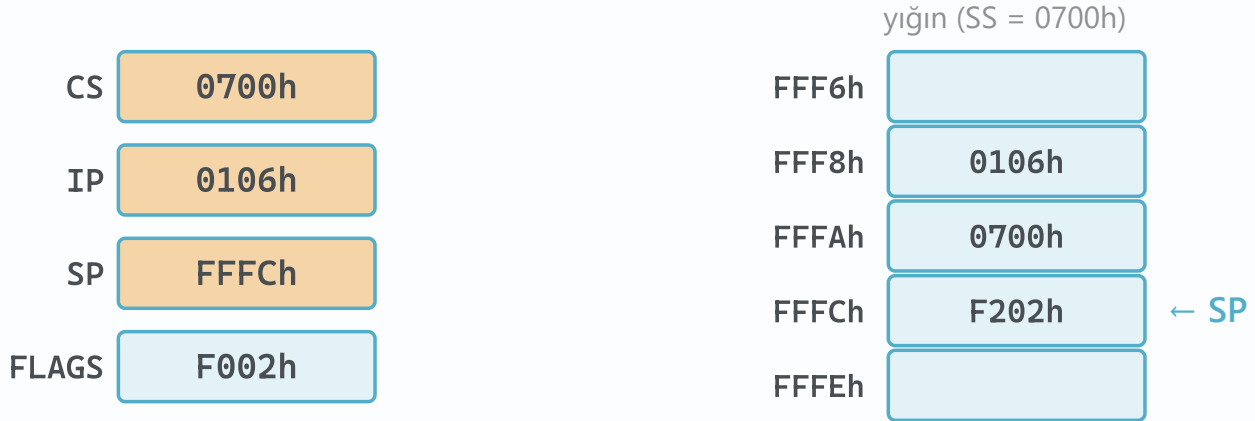
Dönüş adresi itilir: INT 10h 2 bayt olduğu için **0106h**. SP = FFF8h.

INT ve IRET: Yığın Adım Adım



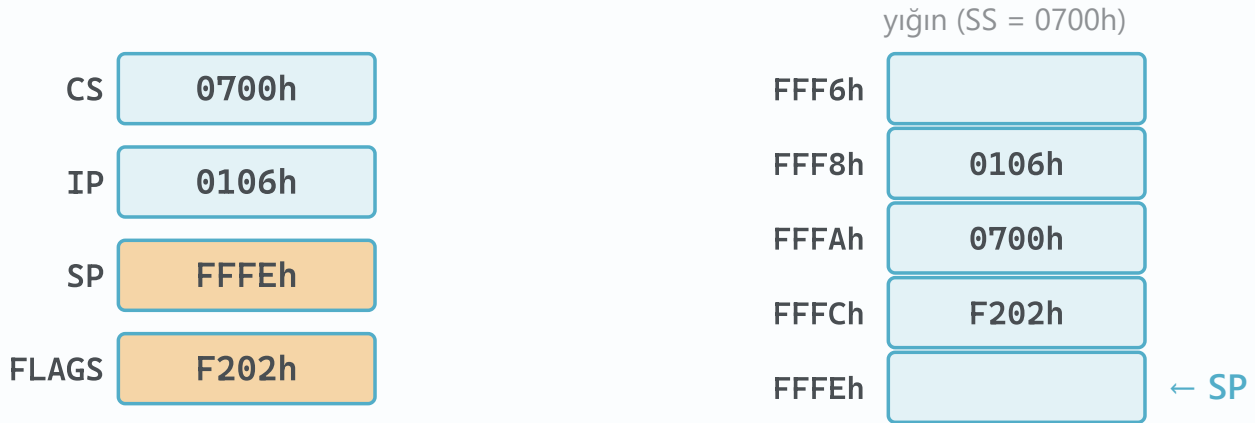
CS:IP ← vektör (F000:F065). ISR çalışır ve karakteri yazar.

INT ve IRET: Yığın Adım Adım



IRET: IP ← 0106h, CS ← 0700h (SP = FFFCh).

INT ve IRET: Yığın Adım Adım



FLAGS ← F202h, SP = FFFEh. Program 0106h'den devam eder. (Yığındaki eski değerler silinmez, yalnızca SP geri çekilir.)

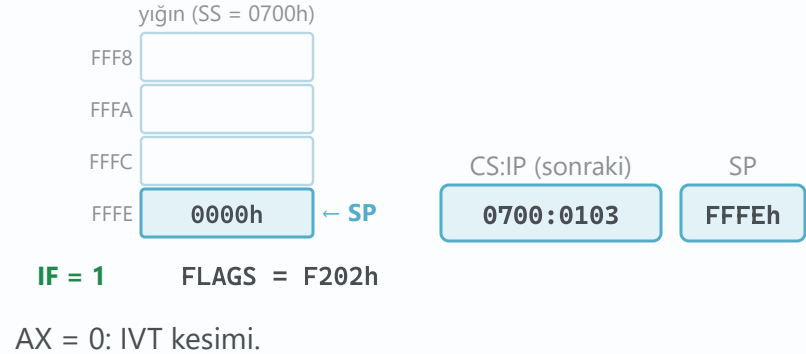
INT/IRET ile CALL/RET

	CALL / RET	INT n / IRET
Hedef adres	komutun içinde (etiket)	vektör tablosundan ($n \times 4$)
Yığına yazılan	IP (yakın) veya CS + IP (uzak)	FLAGS + CS + IP (6 bayt)
Bayraklar	değişmez	IF = 0, TF = 0 yapılır
Dönüş	RET : IP (ve CS) geri alınır	IRET : IP, CS, FLAGS geri alınır
Komut boyu	3 bayt (yakın CALL)	2 bayt (INT 3 : 1 bayt)
Kim çağırır?	yalnız program	program, işlemci (istisna) veya donanım

ISR'yi RET ile bitirmek hatadır: yakın RET yalnız IP'yi alır; CS ve FLAGS yığında kalır (SP 4 bayt kayar) ve IF eski hâline dönmez. ISR'ler daima **IRET** ile biter.

Kod İzleme: IRET Yerine RET

```
1 org 100h
2 mov ax, 0
3 mov es, ax
4 mov word ptr es:[61h*4], offset isr
5 mov word ptr es:[61h*4+2], cs
6 int 61h
7 hlt ; SP = ?
8 isr:
9 ret ; HATA!
```



Adım 1 / 7

DEĞİŞKENLER

CS = 0700h

IP = 0103h

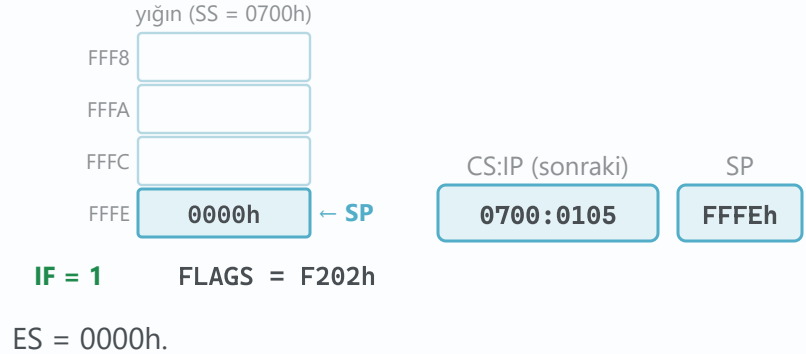
SP = FFEh

IF = 1

RETF bile yetmez: IP ve CS alınır ama FLAGS (2 bayt) yığında kalır ve IF eski hâline dönmez.

Kod İzleme: IRET Yerine RET

```
1 org 100h
2 mov ax, 0
3 mov es, ax
4 mov word ptr es:[61h*4], offset isr
5 mov word ptr es:[61h*4+2], cs
6 int 61h
7 hlt ; SP = ?
8 isr:
9 ret ; HATA!
```



Adım 2 / 7

DEĞİŞKENLER

CS = 0700h

IP = 0105h

SP = FFFEh

IF = 1

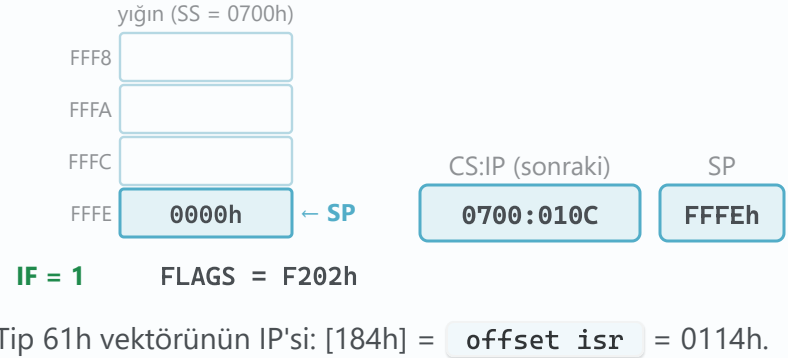
RETF bile yetmez: IP ve CS alınır ama FLAGS (2 bayt) yığında kalır ve IF eski hâline dönmeyiz.

Kod İzleme: IRET Yerine RET

```

1 org 100h
2 mov ax, 0
3 mov es, ax
4 ► mov word ptr es:[61h*4], offset isr
5 mov word ptr es:[61h*4+2], cs
6 int 61h
7 hlt ; SP = ?
8 isr:
9 ret ; HATA!

```



Adım 3 / 7

DEĞİŞKENLER

CS = 0700h

IP = 010Ch

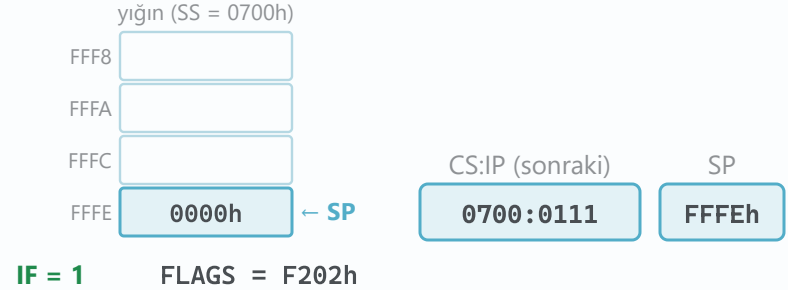
SP = FFEh

IF = 1

RETF bile yetmez: IP ve CS alınır ama FLAGS (2 bayt) yığında kalır ve IF eski hâline dönmez.

Kod İzleme: IRET Yerine RET

```
1 org 100h
2 mov ax, 0
3 mov es, ax
4 mov word ptr es:[61h*4], offset isr
5 mov word ptr es:[61h*4+2], cs
6 int 61h
7 hlt ; SP = ?
8 isr:
9 ret ; HATA!
```



CS: [186h] = 0700h. Vektör 0700:0114.

Adım 4 / 7

DEĞİŞKENLER

CS = 0700h

IP = 0111h

SP = FFEh

IF = 1

RETF bile yetmez: IP ve CS alınır ama FLAGS (2 bayt) yığında kalır ve IF eski hâline dönmez.

Kod İzleme: IRET Yerine RET

```

1 org 100h
2 mov ax, 0
3 mov es, ax
4 mov word ptr es:[61h*4], offset isr
5 mov word ptr es:[61h*4+2], cs
6 int 61h
7 hlt ; SP = ?
8 isr:
9 ret ; HATA!

```



IF = 0 **FLAGS = F002h**

int 61h : yığına FLAGS (F202h), CS (0700h), IP (0113h) —
6 bayt; SP = FFF8h, IF = 0.

Adım 5 / 7

DEĞİŞKENLER

CS = 0700h

IP = 0114h

SP = FFF8h

IF = 0

RETF bile yetmez: IP ve CS alınır ama FLAGS (2 bayt) yığında kalır ve IF eski hâline dönmez.

Kod İzleme: IRET Yerine RET

```

1 org 100h
2 mov ax, 0
3 mov es, ax
4 mov word ptr es:[61h*4], offset isr
5 mov word ptr es:[61h*4+2], cs
6 int 61h
7 hlt ; SP = ?
8 isr:
9 ret ; HATA!

```



ISR'deki yakın **ret** (IRET olmalıydı) yalnız **IP**'yi alır (0113h): CS ve FLAGS yığında kaldı. SP = **FFFAh**, beklenen FFFeh'den 4 bayt eksik; IF = 0 kaldı.

Adım 6 / 7

DEĞİŞKENLER

CS = 0700h

IP = 0113h

SP = FFFAh

IF = 0

RETF bile yetmez: IP ve CS alınır ama FLAGS (2 bayt) yığında kalır ve IF eski hâline dönmez.

Kod İzleme: IRET Yerine RET

```

1 org 100h
2 mov ax, 0
3 mov es, ax
4 mov word ptr es:[61h*4], offset isr
5 mov word ptr es:[61h*4+2], cs
6 int 61h
7 hlt ; SP = ?
8 isr:
9 ret ; HATA!

```



IF = 0 FLAGS = F002h

`hlt` 'e gelindi ama kesmeler kapalı (IF = 0) ve yığın kaymış: bu noktadan sonra bir `ret` 0700h'yi dönüş adresi sanırdı. ISR daima **IRET** ile biter.

Adım 7 / 7

DEĞİŞKENLER

CS = 0700h

IP = 0114h

SP = FFFAh

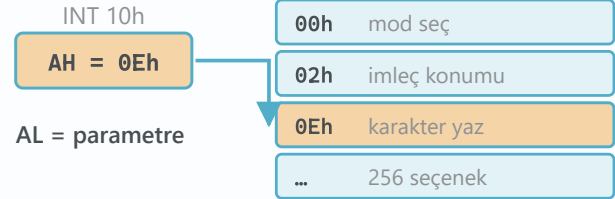
IF = 0

RETF bile yetmez: IP ve CS alınır ama FLAGS (2 bayt) yığında kalır ve IF eski hâline dönmez.

INT Komutu ve Alt Fonksiyonlar

- Yazılım kesmesi **INT değer** komutuyla çağrılır; *değer* 0–255 (0–0FFh) arasında bir sayıdır.
- Bir kesmenin birçok **alt fonksiyonu** olabilir.
- Alt fonksiyon, kesme çağrılmadan önce **AH** yazmacına yazılarak seçilir.
- AH 8 bit olduğundan her kesmenin en fazla **256** alt fonksiyonu olabilir: $256 \times 256 = 65\,536$ farklı hizmet (teorik üst sınır).
- Diğer yazmaçlar (AL, BX, CX, DX, DS:DX ...) **parametre** iletmek ve sonuç almak için kullanılır.

```
MOV AH, 0Eh ; alt fonksiyon
MOV AL, 'A'  ; parametre
INT 10h      ; hizmeti çağır
```



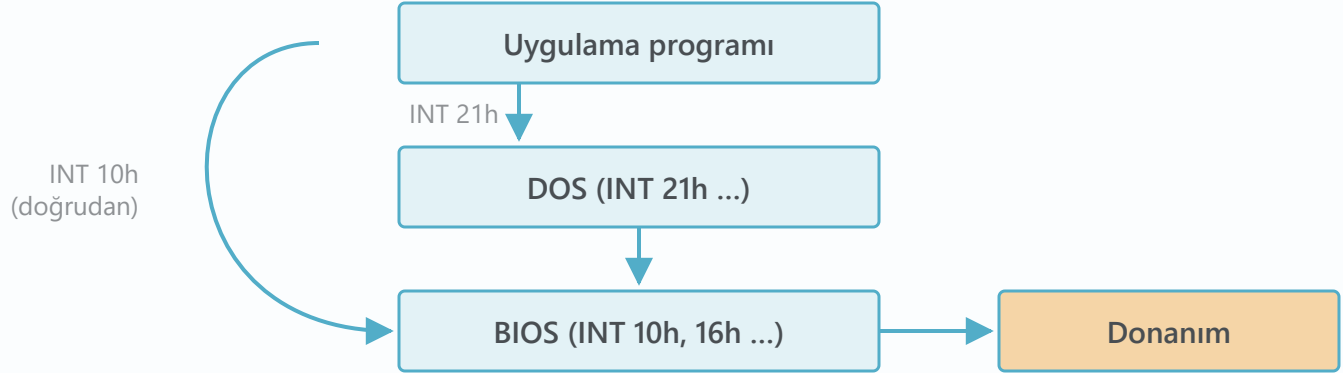
Bazı hizmetler AL'yi de alt-alt fonksiyon olarak kullanır (ör. INT 10h, AX = 4F00h : VESA). Kesme, sonucu çoğunlukla AL/AX'te ya da bayraklarda (CF, ZF) döndürür.

Sık Kullanılan Kesmeler

Kesme	Katman	Görev	Örnek alt fonksiyonlar
INT 10h	BIOS	Video / ekran	00h mod seç, 02h imleç konumu, 0Eh karakter yaz (teletype)
INT 13h	BIOS	Disk	02h sektör oku, 03h sektör yaz
INT 16h	BIOS	Klavye	00h tuş bekle ve oku, 01h tuş var mı?
INT 1Ah	BIOS	Saat	00h tik sayacını oku
INT 20h	DOS	Programı sonlandır	—
INT 21h	DOS	DOS hizmetleri	01h tuş oku, 02h karakter yaz, 09h dizgi yaz, 4Ch çık
INT 33h	sürücü	Fare	00h sıfırla, 03h konum ve düğmeler

emu8086 bu kesmelerin önemli bir kısmını taklit eder; tam liste için emu8086 yardımındaki “supported interrupts” sayfasına bakılabilir.

BIOS ve DOS Katmanları



BIOS

ROM'daki temel giriş/çıkış sistemi. İşletim sisteminden bağımsızdır; açılıшта vektör tablosunu doldurur.

DOS

İşletim sistemi hizmetleri: dosya, bellek, program yükleme/sonlandırma. Kendi işi için çoğu zaman BIOS'u çağırır.

Ekrana Yazı Yazma

```
ORG    100h
MOV    AH, 0Eh    ; alt fonksiyon seç.
MOV    AL, 'H'    ; ASCII kodu: 72
INT     10h        ; yazdır!
MOV    AL, 'e'    ; ASCII kodu: 101
INT     10h        ; yazdır!
MOV    AL, 'l'    ; ASCII kodu: 108
INT     10h        ; yazdır!
MOV    AL, 'l'    ; ASCII kodu: 108
INT     10h        ; yazdır!
RET
```

- **INT 10h** , ekrana karakter yazdırmak ve imleci hareket ettirmek için kullanılır.
- **0Eh** alt fonksiyonu (teletype): AL'deki ASCII kodunu imlecin olduğu yere yazar ve imleci bir ilerletir.
- Her karakter için AL'ye ASCII kodu atanır, ardından **INT 10h** çağrılır.
- AH, kesme tarafından değiştirilmediği için yalnızca bir kez ayarlanır.
- Program ekrana **Hell** yazar; "Hello" için iki komut daha gerekir.

Kod İzleme: INT 10h / 0Eh

```
1  ORG      100h
▶  MOV     AH, 0Eh    ; alt fonksiyon seç.
3  MOV     AL, 'H'    ; ASCII kodu: 72
4  INT      10h       ; yazdır!
5  MOV     AL, 'e'    ; ASCII kodu: 101
6  INT      10h       ; yazdır!
7  MOV     AL, 'l'    ; ASCII kodu: 108
8  INT      10h       ; yazdır!
9  MOV     AL, 'l'    ; ASCII kodu: 108
10 INT      10h       ; yazdır!
11 RET
```

-

AH

0Eh

AL

00h

AH = 0Eh: teletype alt fonksiyonu seçildi.

Adım 1 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 00h

ÇIKTI



Kod İzleme: INT 10h / 0Eh

```
1  ORG    100h
2  MOV    AH, 0Eh    ; alt fonksiyon seç.
▶ 3  MOV    AL, 'H'    ; ASCII kodu: 72
4  INT     10h        ; yazdır!
5  MOV    AL, 'e'    ; ASCII kodu: 101
6  INT     10h        ; yazdır!
7  MOV    AL, 'l'    ; ASCII kodu: 108
8  INT     10h        ; yazdır!
9  MOV    AL, 'l'    ; ASCII kodu: 108
10 INT     10h        ; yazdır!
11 RET
```

-

AH 0Eh

AL 48h

'H' = 72

AL ← 'H' = 72 = 48h.

Adım 2 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 48h

ÇIKTI



Kod İzleme: INT 10h / 0Eh

```
1  ORG    100h
2  MOV    AH, 0Eh    ; alt fonksiyon seç.
3  MOV    AL, 'H'    ; ASCII kodu: 72
4  ► INT   10h       ; yazdır!
5  MOV    AL, 'e'    ; ASCII kodu: 101
6  INT     10h       ; yazdır!
7  MOV    AL, 'l'    ; ASCII kodu: 108
8  INT     10h       ; yazdır!
9  MOV    AL, 'l'    ; ASCII kodu: 108
10 INT     10h       ; yazdır!
11 RET
```

H_

AH 0Eh

AL 48h

'H' = 72

INT 10h : 'H' yazılır, imleç bir sağa geçer.

Adım 3 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 48h

ÇIKTI

H

Kod İzleme: INT 10h / 0Eh

```
1  ORG    100h
2  MOV    AH, 0Eh    ; alt fonksiyon seç.
3  MOV    AL, 'H'    ; ASCII kodu: 72
4  INT     10h        ; yazdır!
5  ► MOV    AL, 'e'    ; ASCII kodu: 101
6  INT     10h        ; yazdır!
7  MOV    AL, 'l'    ; ASCII kodu: 108
8  INT     10h        ; yazdır!
9  MOV    AL, 'l'    ; ASCII kodu: 108
10 INT     10h        ; yazdır!
11 RET
```

H_

AH 0Eh

AL 65h

'e' = 101

AL ← 'e' = 101 = 65h.

Adım 4 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 65h

ÇIKTI

H

Kod İzleme: INT 10h / 0Eh

```
1  ORG    100h
2  MOV    AH, 0Eh    ; alt fonksiyon seç.
3  MOV    AL, 'H'    ; ASCII kodu: 72
4  INT     10h       ; yazdır!
5  MOV    AL, 'e'    ; ASCII kodu: 101
▶  INT     10h       ; yazdır!
7  MOV    AL, 'l'    ; ASCII kodu: 108
8  INT     10h       ; yazdır!
9  MOV    AL, 'l'    ; ASCII kodu: 108
10 INT     10h       ; yazdır!
11 RET
```

He_

AH = 0Eh

AL = 65h

'e' = 101

INT 10h : 'e' yazılır, imleç bir sağa geçer.

Adım 5 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 65h

ÇIKTI

He

Kod İzleme: INT 10h / 0Eh

```
1  ORG    100h
2  MOV    AH, 0Eh    ; alt fonksiyon seç.
3  MOV    AL, 'H'    ; ASCII kodu: 72
4  INT     10h        ; yazdır!
5  MOV    AL, 'e'    ; ASCII kodu: 101
6  INT     10h        ; yazdır!
7  ► MOV    AL, 'l'    ; ASCII kodu: 108
8  INT     10h        ; yazdır!
9  MOV    AL, 'l'    ; ASCII kodu: 108
10 INT     10h        ; yazdır!
11 RET
```

He_

AH = 0Eh

AL = 6Ch

'l' = 108

AL ← 'l' = 108 = 6Ch.

Adım 6 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 6Ch

ÇIKTI

He

Kod İzleme: INT 10h / 0Eh

```
1  ORG    100h
2  MOV    AH, 0Eh    ; alt fonksiyon seç.
3  MOV    AL, 'H'    ; ASCII kodu: 72
4  INT     10h        ; yazdır!
5  MOV    AL, 'e'    ; ASCII kodu: 101
6  INT     10h        ; yazdır!
7  MOV    AL, 'l'    ; ASCII kodu: 108
8  ► INT     10h        ; yazdır!
9  MOV    AL, 'l'    ; ASCII kodu: 108
10 INT     10h        ; yazdır!
11 RET
```

Hel_

AH = 0Eh

AL = 6Ch

'l' = 108

INT 10h : 'l' yazılır, imleç bir sağa geçer.

Adım 7 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 6Ch

ÇIKTI

Hel

Kod İzleme: INT 10h / 0Eh

```
1  ORG    100h
2  MOV    AH, 0Eh    ; alt fonksiyon seç.
3  MOV    AL, 'H'    ; ASCII kodu: 72
4  INT     10h       ; yazdır!
5  MOV    AL, 'e'    ; ASCII kodu: 101
6  INT     10h       ; yazdır!
7  MOV    AL, 'l'    ; ASCII kodu: 108
8  INT     10h       ; yazdır!
9  ► MOV    AL, 'l'    ; ASCII kodu: 108
10 INT     10h       ; yazdır!
11 RET
```

He**l**_

AH = 0Eh

AL = 6Ch

'l' = 108

AL ← 'l' = 108 = 6Ch.

Adım 8 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 6Ch

ÇIKTI

He**l**

Kod İzleme: INT 10h / 0Eh

```
1  ORG    100h
2  MOV    AH, 0Eh    ; alt fonksiyon seç.
3  MOV    AL, 'H'    ; ASCII kodu: 72
4  INT     10h        ; yazdır!
5  MOV    AL, 'e'    ; ASCII kodu: 101
6  INT     10h        ; yazdır!
7  MOV    AL, 'l'    ; ASCII kodu: 108
8  INT     10h        ; yazdır!
9  MOV    AL, 'l'    ; ASCII kodu: 108
▶  INT     10h        ; yazdır!
11 RET                ; işletim sistemine geri dön.
```

Hell_

AH = 0Eh

AL = 6Ch

'l' = 108

INT 10h : 'l' yazılır, imleç bir sağa geçer.

Adım 9 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 6Ch

ÇIKTI

Hell

Kod İzleme: INT 10h / 0Eh

```
1  ORG    100h
2  MOV    AH, 0Eh    ; alt fonksiyon seç.
3  MOV    AL, 'H'    ; ASCII kodu: 72
4  INT     10h       ; yazdır!
5  MOV    AL, 'e'    ; ASCII kodu: 101
6  INT     10h       ; yazdır!
7  MOV    AL, 'l'    ; ASCII kodu: 108
8  INT     10h       ; yazdır!
9  MOV    AL, 'l'    ; ASCII kodu: 108
10 INT     10h       ; yazdır!
▶ RET                                ; işletim sistemine geri dön.
```

Hell_

AH = 0Eh

AL = 6Ch

RET : COM programı DOS'a döner. Ekranda **Hell** kalır.

Adım 10 / 10

DEĞİŞKENLER

AH = 0Eh

AL = 6Ch

ÇIKTI

Hell

INT 10h: Diğer Alt Fonksiyonlar

AH	İşlev	Girdi	Not
00h	Video modunu seç	AL = mod (03h: 80×25 metin)	ekranı da temizler
02h	İmleci konumlandır	BH = sayfa, DH = satır, DL = sütun	sol üst köşe (0, 0)
03h	İmleç konumunu oku	BH = sayfa	DH, DL'de döner
06h	Pencereyi yukarı kaydır	AL = satır sayısı (0: temizle), BH = renk, CX–DX = köşeler	ekran temizleme
09h	Karakter + renk yaz	AL = karakter, BL = renk, CX = tekrar	imleç ilerlemez
0Eh	Teletype yaz	AL = karakter (BH = sayfa)	imleç ilerler, CR/LF işlenir

Metin modunda ekran 80 sütun × 25 satırdır; her hücre bir ASCII kodu ve bir renk (öznitelik) baytı tutar.

Demo: Metinden INT 10h Koduna

Bir metin yazın: her karakter için gereken `MOV AL` / `INT 10h` çifti ve ASCII kodları üretilir.

Metin

Hello

```
MOV AH, 0Eh
```

```
MOV AL, 'H' ; 72 = 48h
```

```
INT 10h
```

```
MOV AL, 'e' ; 101 = 65h
```

```
INT 10h
```

```
MOV AL, 'l' ; 108 = 6Ch
```

```
INT 10h
```

```
MOV AL, 'l' ; 108 = 6Ch
```

```
INT 10h
```

```
MOV AL, 'o' ; 111 = 6Fh
```

```
INT 10h
```

```
RET
```

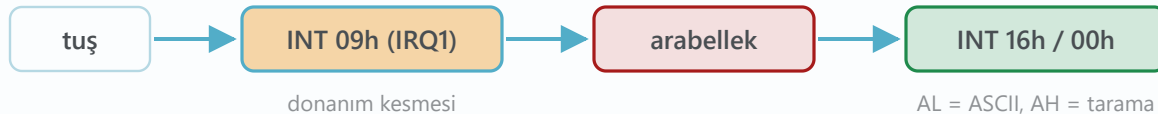
Hello_

12 komut, 5 kesme çağrısı

Klavye Tuşunu Bekleme

```
ORG 100h
; wait for any key press:
MOV AH, 0
INT 16H
RET
```

- **INT 16h** , BIOS klavye hizmetidir.
- **AH = 00h**: bir tuşa basılana kadar **bekler**; tuş arabellekten alınır.
- Dönüşte **AL = ASCII kodu**, **AH = tarama kodu** (scan code). Ör. 'a' → AL = 61h, AH = 1Eh.
- Ekranı bir şey yazmaz (yankı yoktur).
- **AH = 01h**: beklemeden kontrol eder; tuş yoksa **ZF = 1**, varsa ZF = 0 ve AX'te tuş (arabellekten silinmez).
- Program sonundaki **MOV AH, 0 / INT 16h** , çıktı ekranının kapanmadan önce okunabilmesini sağlar.



Kod İzleme: Tuş Oku ve Yankıla

```
1  ORG 100h
2  ► MOV AH, 0      ; tuş bekle
3  INT 16h          ; AL = ASCII, AH =
   tarama
4  MOV AH, 0Eh      ; teletype
5  INT 10h          ; AL'yi yaz
6  RET
```

AH = 0: "tuş bekle ve oku" alt fonksiyonu.

Adım 1 / 6

DEĞİŞKENLER

AH = 00h

AL = 00h

ÇIKTI



Kod İzleme: Tuşu Oku ve Yankıla

```
1  ORG 100h
2  MOV AH, 0      ; tuş bekle
3  ► INT 16h      ; AL = ASCII, AH = tarama
4  MOV AH, 0Eh    ; teletype
5  INT 10h        ; AL'yi yaz
6  RET
```

INT 16h : program burada **bekler**; kullanıcı henüz tuşa basmadı.

Adım 2 / 6

DEĞİŞKENLER

AH = 00h

AL = 00h

ÇIKTI



Kod İzleme: Tuşu Oku ve Yankıla

```
1 ORG 100h
2 MOV AH, 0      ; tuş bekle
3 INT 16h        ; AL = ASCII, AH = tarama
4 MOV AH, 0Eh    ; teletype
5 INT 10h        ; AL'yi yaz
6 RET
```

Kullanıcı **a** tuşuna bastı: AL = 61h ('a'), AH = 1Eh (A tuşunun tarama kodu). Ekranda henüz bir şey yok.

Adım 3 / 6

DEĞİŞKENLER

AH = 1Eh

AL = 61h

ÇIKTI



Kod İzleme: Tuşu Oku ve Yankıla

```
1 ORG 100h
2 MOV AH, 0      ; tuş bekle
3 INT 16h        ; AL = ASCII, AH =
  tarama
4 ► MOV AH, 0Eh   ; teletype
5 INT 10h        ; AL'yi yaz
6 RET
```

AH = 0Eh. AL hâlâ 61h: okunan karakter doğrudan yazdırılabilir.

Adım 4 / 6

DEĞİŞKENLER

AH = 0Eh

AL = 61h

ÇIKTI



Kod İzleme: Tuşu Oku ve Yankıla

```
1 ORG 100h
2 MOV AH, 0      ; tuş bekle
3 INT 16h        ; AL = ASCII, AH =
  tarama
4 MOV AH, 0Eh    ; teletype
▶ INT 10h        ; AL'yi yaz
6 RET
```

INT 10h : 'a' ekrana yazılır (yankı).

Adım 5 / 6

DEĞİŞKENLER

AH = 0Eh

AL = 61h

ÇIKTI

a

Kod İzleme: Tuşu Oku ve Yankıla

```
1 ORG 100h
2 MOV AH, 0      ; tuş bekle
3 INT 16h        ; AL = ASCII, AH =
  tarama
4 MOV AH, 0Eh    ; teletype
5 INT 10h        ; AL'yi yaz
▶ RET
```

RET : DOS'a dönüş.

Adım 6 / 6

DEĞİŞKENLER

AH = 0Eh

AL = 61h

ÇIKTI

a

INT 21h Hizmetleri

AH	İşlev	Girdi	Çıktı
01h	Tuş oku, ekrana yankıla	—	AL = ASCII
02h	Karakter yaz	DL = karakter	AL = DL
09h	Dizgi yaz	DS:DX = '\$' ile biten dizgi	—
0Ah	Satır oku (arabellekli)	DS:DX = arabellek	arabellek dolar
4Ch	Programı sonlandır	AL = dönüş kodu	—

```
MOV AH, 2      ; karakter yaz
MOV DL, '*'
INT 21h        ; ekranda: *
```

```
MOV AH, 9      ; dizge yaz
LEA DX, msg
INT 21h
msg DB 'Merhaba$'
```

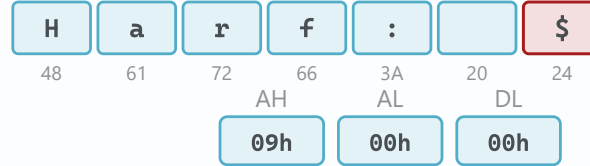
Kod İzleme: 09h, 01h ve 02h

```

1 org 100h
▶ mov ah, 9
3 lea dx, soru
4 int 21h          ; dizgiyi yaz
5 mov ah, 1
6 int 21h          ; oku + yankı
7 mov dl, al
8 inc dl           ; sonraki harf
9 mov ah, 2
10 int 21h         ; DL'yi yaz
11 ret
12 soru db 'Harf: $'

```

soru (DS:0115h)



09h dizgi

ekran: _

AH = 09h: "\$ ile biten dizgiyi yaz" hizmeti.

Adım 1 / 9

DEĞİŞKENLER

AH = 09h

AL = 00h

DX = 0000h

ÇIKTI

Girdi olarak k tuşu varsayıldı. Hizmet her seferinde AH ile seçilir; INT 21h AH'yi korur ama AL'yi değiştirebilir.

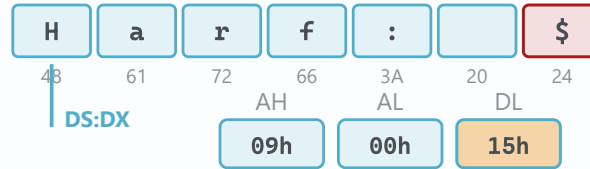
Kod İzleme: 09h, 01h ve 02h

```

1 org 100h
2 mov ah, 9
▶ 3 lea dx, soru
4 int 21h ; dizgiyi yaz
5 mov ah, 1
6 int 21h ; oku + yankı
7 mov dl, al
8 inc dl ; sonraki harf
9 mov ah, 2
10 int 21h ; DL'yi yaz
11 ret
12 soru db 'Harf: $'

```

soru (DS:0115h)



09h dizgi

ekran: _

lea dx, soru : DX = **0115h**. DS:DX dizginin ilk baytını gösterir.

Adım 2 / 9

DEĞİŞKENLER

AH = 09h

AL = 00h

DX = 0115h

ÇIKTI



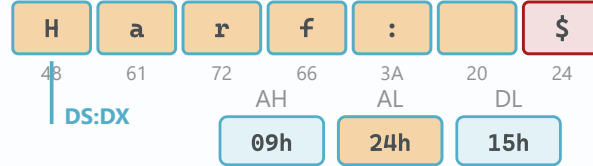
Girdi olarak k tuşu varsayıldı. Hizmet her seferinde AH ile seçilir; INT 21h AH'yi korur ama AL'yi değiştirebilir.

INT 21H: DOS · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: 09h, 01h ve 02h

```
1 org 100h
2 mov ah, 9
3 lea dx, soru
4 ► int 21h ; dizgiyi yaz
5 mov ah, 1
6 int 21h ; oku + yankı
7 mov dl, al
8 inc dl ; sonraki harf
9 mov ah, 2
10 int 21h ; DL'yi yaz
11 ret
12 soru db 'Harf: $'
```

soru (DS:0115h)



ekran: Harf: _

int 21h : 'Harf: ' yazılır; \$ (24h) yazılmaz, yalnızca sonu işaretler.
Dönüşte AL = 24h.

Adım 3 / 9

DEĞİŞKENLER

AH = 09h

AL = 24h

DX = 0115h

ÇIKTI

Harf:

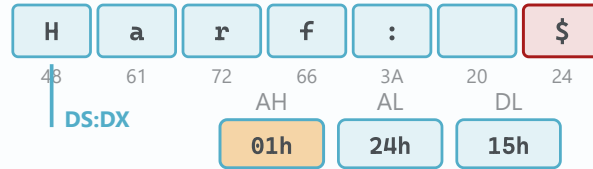
Girdi olarak k tuşu varsayıldı. Hizmet her seferinde AH ile seçilir; INT 21h AH'yi korur ama AL'yi değiştirebilir.

INT 21H: DOS · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: 09h, 01h ve 02h

```
1 org 100h
2 mov ah, 9
3 lea dx, soru
4 int 21h ; dizgiyi yaz
5 ► mov ah, 1
6 int 21h ; oku + yankı
7 mov dl, al
8 inc dl ; sonraki harf
9 mov ah, 2
10 int 21h ; DL'yi yaz
11 ret
12 soru db 'Harf: $'
```

soru (DS:0115h)



01h oku

ekran: Harf: _

AH = 01h: tuş oku ve ekrana yankıla.

Adım 4 / 9

DEĞİŞKENLER

AH = 01h

AL = 24h

DX = 0115h

ÇIKTI

Harf:

Girdi olarak k tuşu varsayıldı. Hizmet her seferinde AH ile seçilir; INT 21h AH'yi korur ama AL'yi değiştirebilir.

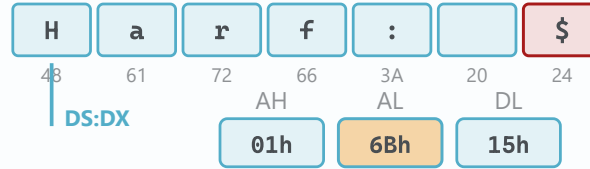
Kod İzleme: 09h, 01h ve 02h

```

1 org 100h
2 mov ah, 9
3 lea dx, soru
4 int 21h          ; dizgiyi yaz
5 mov ah, 1
6 int 21h          ; oku + yankı
7 mov dl, al
8 inc dl          ; sonraki harf
9 mov ah, 2
10 int 21h         ; DL'yi yaz
11 ret
12 soru db 'Harf: $'

```

soru (DS:0115h)



01h oku

ekran: Harf: k_

Kullanıcı **k** tuşuna bastı: AL = **6Bh**, harf ekranda da görüldü.

Adım 5 / 9

DEĞİŞKENLER

AH = 01h

AL = 6Bh

DX = 0115h

ÇIKTI

Harf: k

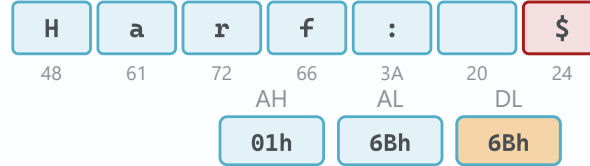
Girdi olarak k tuşu varsayıldı. Hizmet her seferinde AH ile seçilir; INT 21h AH'yi korur ama AL'yi değiştirebilir.

INT 21H: DOS · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: 09h, 01h ve 02h

```
1 org 100h
2 mov ah, 9
3 lea dx, soru
4 int 21h ; dizgiyi yaz
5 mov ah, 1
6 int 21h ; oku + yankı
7 mov dl, al
8 inc dl ; sonraki harf
9 mov ah, 2
10 int 21h ; DL'yi yaz
11 ret
12 soru db 'Harf: $'
```

soru (DS:0115h)



01h oku

ekran: Harf: k_

DL = AL = 6Bh: 02h hizmeti karakteri DL'de bekler.

Adım 6 / 9

DEĞİŞKENLER

AH = 01h

AL = 6Bh

DX = 016Bh

ÇIKTI

Harf: k

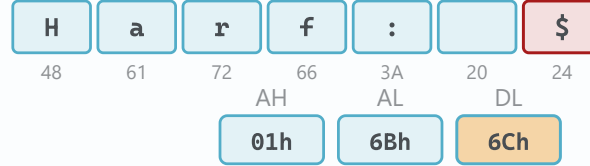
Girdi olarak k tuşu varsayıldı. Hizmet her seferinde AH ile seçilir; INT 21h AH'yi korur ama AL'yi değiştirebilir.

INT 21H: DOS · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: 09h, 01h ve 02h

```
1 org 100h
2 mov ah, 9
3 lea dx, soru
4 int 21h          ; dizgiyi yaz
5 mov ah, 1
6 int 21h          ; oku + yankı
7 mov dl, al
8 ▶ inc dl         ; sonraki harf
9 mov ah, 2
10 int 21h         ; DL'yi yaz
11 ret
12 soru db 'Harf: $'
```

soru (DS:0115h)



ekran: Harf: k_

DL = 6Ch ('l'): ASCII'de harfler sıralıdır.

Adım 7 / 9

DEĞİŞKENLER

AH = 01h

AL = 6Bh

DX = 016Ch

ÇIKTI

Harf: k

Girdi olarak k tuşu varsayıldı. Hizmet her seferinde AH ile seçilir; INT 21h AH'yi korur ama AL'yi değiştirebilir.

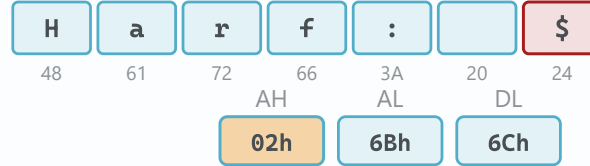
Kod İzleme: 09h, 01h ve 02h

```

1 org 100h
2 mov ah, 9
3 lea dx, soru
4 int 21h          ; dizgiyi yaz
5 mov ah, 1
6 int 21h          ; oku + yankı
7 mov dl, al
8 inc dl           ; sonraki harf
9 ► mov ah, 2
10 int 21h         ; DL'yi yaz
11 ret
12 soru db 'Harf: $'

```

soru (DS:0115h)



02h yaz

ekran: Harf: k_

AH = 02h: tek karakter yaz.

Adım 8 / 9

DEĞİŞKENLER

AH = 02h

AL = 6Bh

DX = 016Ch

ÇIKTI

Harf: k

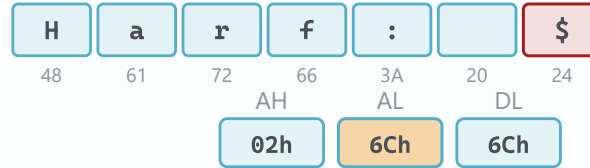
Girdi olarak k tuşu varsayıldı. Hizmet her seferinde AH ile seçilir; INT 21h AH'yi korur ama AL'yi değiştirebilir.

INT 21H: DOS · KODU ADIM ADIM ÇALIŞTIR

Kod İzleme: 09h, 01h ve 02h

```
1 org 100h
2 mov ah, 9
3 lea dx, soru
4 int 21h ; dizgiyi yaz
5 mov ah, 1
6 int 21h ; oku + yankı
7 mov dl, al
8 inc dl ; sonraki harf
9 mov ah, 2
▶ int 21h ; DL'yi yaz
11 ret
12 soru db 'Harf: $'
```

soru (DS:0115h)



02h yaz

ekran: Harf: kl_

int 21h : 'l' yazıldı; dönüşte AL = DL. Ekranda **Harf: kl**.

Adım 9 / 9

DEĞİŞKENLER

AH = 02h

AL = 6Ch

DX = 016Ch

ÇIKTI

Harf: kl

Girdi olarak k tuşu varsayıldı. Hizmet her seferinde AH ile seçilir; INT 21h AH'yi korur ama AL'yi değiştirebilir.

INT 10h/0Eh mi, INT 21h/02h mi?

BIOS · INT 10h, AH = 0Eh

- Karakter **AL**'de.
- Doğrudan video BIOS'u; DOS gerekmez.
- Açılış (boot) kodunda da çalışır.
- Renk (grafik modda BL) verilebilir.

DOS · INT 21h, AH = 02h

- Karakter **DL**'de.
- DOS çağırısı; çıktı yönlendirilebilir (dosyaya, yazıcıya).
- Dönüşte AL = DL olur.
- Ctrl+Break denetimi yapılır.

İkisi de bir karakter yazar ve imleci ilerletir; seçim, programın hangi ortamda çalışacağına bağlıdır. Aynı programda karıştırılabilir.

ÖRNEK PROGRAM

Toplama, Çıkarma ve Sonucu Yazdırma

```
org 100h
mov al, 5      ; bin=00000101b
mov bl, 10     ; hex=0ah or bin=00001010b
; 5 + 10 = 15 (decimal), 0fh, 00001111b
add bl, al
; 15 - 1 = 14 (decimal), 0eh, 00001110b
sub bl, 1
mov cx, 8      ; sonucu ikilik tabanda yazdır
```

```
print:
mov ah, 2      ; yazdırma fonksiyonu
mov dl, '0'
test bl, 10000000b ; ilk biti test et
jz zero
mov dl, '1'
zero:
int 21h
shl bl, 1
loop print
; b (binary suffix) yazdır
mov dl, 'b'
int 21h
; klavye tuşu basılmasını bekle
mov ah, 0
int 16h
ret
```

- $5 + 10 - 1 = 14 = 0Eh$ hesaplanır; sonuç BL'dedir.
- Döngü 8 kez (CX = 8) çalışır: her turda BL'nin **en anlamlı biti** (MSB) test edilir, '0' ya da '1' yazılır, BL bir sola kaydırılır.
- En sonda 'b' eki yazılır ve bir tuşa basılmasını beklenir. Beklenen çıktı: **00001110b**.

ÖRNEK PROGRAM

TEST ve SHL ile Bit Okuma

BL 7 6 5 4 3 2 1 0
0 0 0 0 1 1 1 0 AND → 00000000

mask 1 0 0 0 0 0 0 0 ZF = 1 (bit 7 = 0)

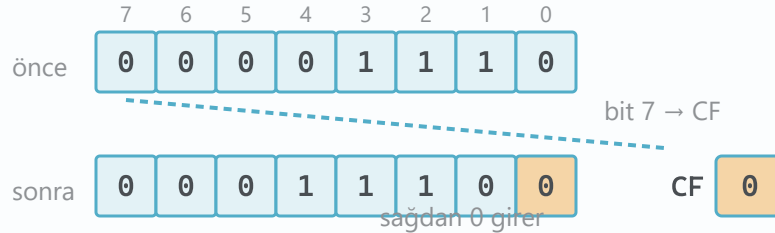
TEST BL, 10000000b : BL ile maske VE'lenir ama BL **değişmez**. Sonuç 0 ise bit 7 = 0 →
ZF = 1, JZ atlar.

Adım 1 / 3

Alternatif: SHL BL,1 ardından JC ile CF'ye bakmak. Ancak bu sıralamada kaydırma, yazdırmadan önce yapılmalıdır.

ÖRNEK PROGRAM

TEST ve SHL ile Bit Okuma



SHL BL, 1 : tüm bitler bir sola kayar; çıkan bit 7 **CF**'ye gider, bit 0'a 0 girer. Bir sonraki bit, test edilecek konuma (bit 7) gelir.

Adım 2 / 3

Alternatif: SHL BL,1 ardından JC ile CF'ye bakmak. Ancak bu sıralamada kaydırma, yazdırmadan önce yapılmalıdır.

ÖRNEK PROGRAM

TEST ve SHL ile Bit Okuma

tur:	1	2	3	4	5	6	7	8
bit:	0	0	0	0	1	1	1	0

8 tur sonunda BL'nin bitleri MSB'den LSB'ye sırayla yazılmış olur: **00001110**, ardından **b**.

Adım 3 / 3

Alternatif: SHL BL,1 ardından JC ile CF'ye bakmak. Ancak bu sıralamada kaydırma, yazdırmadan önce yapılmalıdır.

Kod İzleme (1/2): Hesaplama

```
1 org 100h
2 ► mov al, 5      ; bin=00000101b
3 mov bl, 10       ; hex=0ah or bin=00001010b
4 ; 5 + 10 = 15 (decimal), 0fh, 00001111b
5 add bl, al
6 ; 15 - 1 = 14 (decimal), 0eh, 00001110b
7 sub bl, 1
8 mov cx, 8        ; sonucu ikilik tabanda yazdır
```

AL = 5 = 05h = 00000101b. MOV bayrakları etkilemez.

Adım 1 / 5

DEĞİŞKENLER

AL = 05h

BL = 00h

CX = 0000h

CF = 0

ZF = 0

SF = 0

PF = 0

AF = 0

Kod İzleme (1/2): Hesaplama

```
1 org 100h
2 mov al, 5      ; bin=00000101b
▶ 3 mov bl, 10    ; hex=0ah or bin=00001010b
4 ; 5 + 10 = 15 (decimal), 0fh, 00001111b
5 add bl, al
6 ; 15 - 1 = 14 (decimal), 0eh, 00001110b
7 sub bl, 1
8 mov cx, 8      ; sonucu ikilik tabanda yazdır
```

BL = 10 = 0Ah = 00001010b.

Adım 2 / 5

DEĞİŞKENLER

AL = 05h

BL = 0Ah

CX = 0000h

CF = 0

ZF = 0

SF = 0

PF = 0

AF = 0

Kod İzleme (1/2): Hesaplama

```
1 org 100h
2 mov al, 5      ; bin=00000101b
3 mov bl, 10     ; hex=0ah or bin=00001010b
4 ; 5 + 10 = 15 (decimal), 0fh, 00001111b
▶ add bl, al
6 ; 15 - 1 = 14 (decimal), 0eh, 00001110b
7 sub bl, 1
8 mov cx, 8      ; sonucu ikilik tabanda yazdır
```

ADD BL, AL : BL = 0Ah + 05h = **0Fh** (15).
Taşma yok (CF = 0), sonuçta dört 1 biti var → PF = 1.

Adım 3 / 5

DEĞİŞKENLER

AL = 05h

BL = **0Fh**

CX = 0000h

CF = 0

ZF = 0

SF = 0

PF = **1**

AF = 0

Kod İzleme (1/2): Hesaplama

```
1 org 100h
2 mov al, 5      ; bin=00000101b
3 mov bl, 10     ; hex=0ah or bin=00001010b
4 ; 5 + 10 = 15 (decimal), 0fh, 00001111b
5 add bl, al
6 ; 15 - 1 = 14 (decimal), 0eh, 00001110b
▶ sub bl, 1
8 mov cx, 8      ; sonucu ikilik tabanda yazdır
```

SUB BL, 1 : BL = **0Eh** (14) = 00001110b. Üç
1 biti → PF = 0. Ödünç yok: CF = 0.

Adım 4 / 5

DEĞİŞKENLER

AL = **05h**

BL = **0Eh**

CX = **0000h**

CF = **0**

ZF = **0**

SF = **0**

PF = **0**

AF = **0**

Kod İzleme (1/2): Hesaplama

```
1 org 100h
2 mov al, 5      ; bin=00000101b
3 mov bl, 10     ; hex=0ah or bin=00001010b
4 ; 5 + 10 = 15 (decimal), 0fh, 00001111b
5 add bl, al
6 ; 15 - 1 = 14 (decimal), 0eh, 00001110b
7 sub bl, 1
▶ mov cx, 8      ; sonucu ikilik tabanda yazdır
```

CX = 8: **LOOP** sayacı, her bit için bir tur.

Adım 5 / 5

DEĞİŞKENLER

AL = **05h**

BL = **0Eh**

CX = 0008h

CF = **0**

ZF = **0**

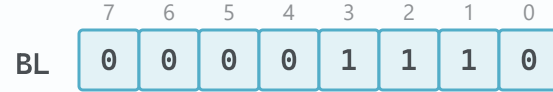
SF = **0**

PF = **0**

AF = **0**

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
▶ mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



Tur 1: AH = 2 (DOS karakter yaz).

Adım 1 / 26

DEĞİŞKENLER

AH = 02h

DL = 00h

BL = 0Eh

CX = 0008h

ZF = 0

CF = 0

ÇIKTI



Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 ► mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```

7 6 5 4 3 2 1 0
BL 0 0 0 0 1 1 1 0

CF 0 DL '0'

Varsayılan karakter DL = '0' (30h).

Adım 2 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = 0Eh

CX = 0008h

ZF = 0

CF = 0

ÇIKTI

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 ► test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



TEST : bit 7 = 0 → ZF = 1, CF = 0.

Adım 3 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = 0Eh

CX = 0008h

ZF = 1

CF = 0

ÇIKTI

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



ZF = 1 → JZ zero atlar; DL '0' kalır.

Adım 4 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = 0Eh

CX = 0008h

ZF = 1

CF = 0

ÇIKTI



Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 ► int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```

7 6 5 4 3 2 1 0
BL 0 0 0 0 1 1 1 0

CF 0 DL '0'

INT 21h : DL'deki '0' yazılır.

Adım 5 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = 0Eh

CX = 0008h

ZF = 1

CF = 0

ÇIKTI

0

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 ► shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```

7 6 5 4 3 2 1 0
BL 0 0 0 1 1 1 0 0

CF 0 DL '0'

SHL BL,1 : BL = 1Ch, çıkan bit CF = 0.

Adım 6 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = 1Ch

CX = 0008h

ZF = 0

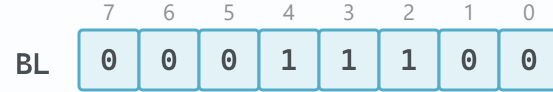
CF = 0

ÇIKTI

0

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 1000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
▶ loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



LOOP : CX = 7 ≠ 0 → print 'e dön.

Adım 7 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = 1Ch

CX = 0007h

ZF = 0

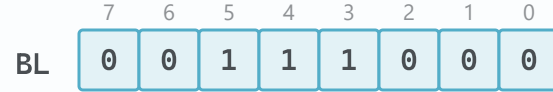
CF = 0

ÇIKTI

0

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
▶ loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



Tur 2 (özet): bit = 0, '0' yazıldı, BL = 38h. **LOOP** : CX = 6
≠ 0 → **print** 'e dön.

Adım 8 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = 38h

CX = 0006h

ZF = 0

CF = 0

ÇIKTI

00

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
▶ loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```

7 6 5 4 3 2 1 0
BL 0 1 1 1 0 0 0 0

CF 0 DL '0'

Tur 3 (özet): bit = 0, '0' yazıldı, BL = 70h. LOOP : CX = 5
≠ 0 → print 'e dön.

Adım 9 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = 70h

CX = 0005h

ZF = 0

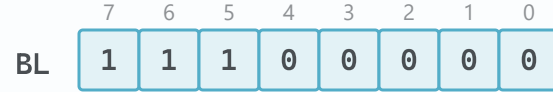
CF = 0

ÇIKTI

000

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
▶ loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



Tur 4 (özet): bit = 0, '0' yazıldı, BL = E0h. **LOOP** : CX = 4
≠ 0 → **print** 'e dön.

Adım 10 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = E0h

CX = 0004h

ZF = 0

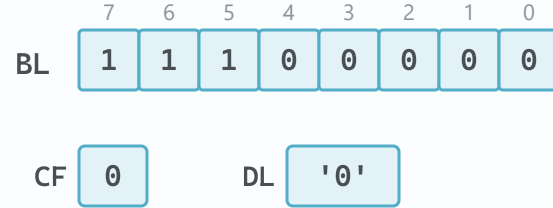
CF = 0

ÇIKTI

0000

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
▶ mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



Tur 5: AH = 2 (DOS karakter yaz).

Adım 11 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = E0h

CX = 0004h

ZF = 0

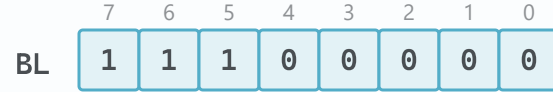
CF = 0

ÇIKTI

0000

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 ► mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



Varsayılan karakter DL = '0' (30h).

Adım 12 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = E0h

CX = 0004h

ZF = 0

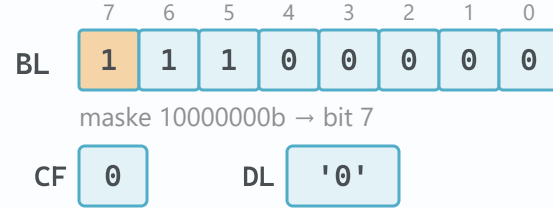
CF = 0

ÇIKTI

0000

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 ► test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



TEST : bit 7 = 1 → ZF = 0, CF = 0.

Adım 13 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = E0h

CX = 0004h

ZF = 0

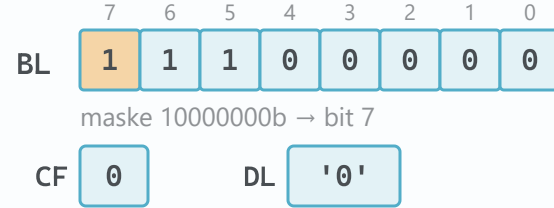
CF = 0

ÇIKTI

0000

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



ZF = 0 → JZ atlamaz, sıradaki komut çalışır.

Adım 14 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = E0h

CX = 0004h

ZF = 0

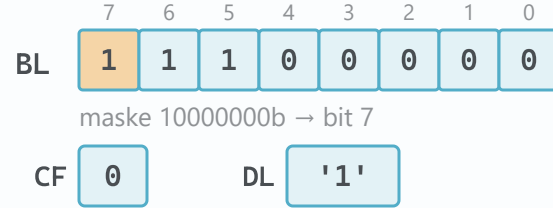
CF = 0

ÇIKTI

0000

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
▶ mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



DL = '1' (31h).

Adım 15 / 26

DEĞİŞKENLER

AH = 02h

DL = 31h

BL = E0h

CX = 0004h

ZF = 0

CF = 0

ÇIKTI

0000

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 ► int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```

BL

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

CF 0 DL '1'

INT 21h : DL'deki '1' yazılır.

Adım 16 / 26

DEĞİŞKENLER

AH = 02h

DL = 31h

BL = E0h

CX = 0004h

ZF = 0

CF = 0

ÇIKTI

00001

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 ► shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```

BL

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

CF 1 DL '1'

SHL BL,1 : BL = C0h, çıkan bit CF = 1.

Adım 17 / 26

DEĞİŞKENLER

AH = 02h

DL = 31h

BL = C0h

CX = 0004h

ZF = 0

CF = 1

ÇIKTI

00001

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
▶ loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```

7 6 5 4 3 2 1 0
BL 1 1 0 0 0 0 0 0

CF 1 DL '1'

LOOP : CX = 3 ≠ 0 → print 'e dön.

Adım 18 / 26

DEĞİŞKENLER

AH = 02h

DL = 31h

BL = C0h

CX = 0003h

ZF = 0

CF = 1

ÇIKTI

00001

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
▶ loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```

7 6 5 4 3 2 1 0
BL 1 0 0 0 0 0 0 0

CF 1 DL '1'

Tur 6 (özet): bit = 1, '1' yazıldı, BL = 80h. LOOP : CX = 2
≠ 0 → print 'e dön.

Adım 19 / 26

DEĞİŞKENLER

AH = 02h

DL = 31h

BL = 80h

CX = 0002h

ZF = 0

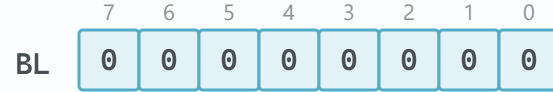
CF = 1

ÇIKTI

000011

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
▶ loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



Tur 7 (özet): bit = 1, '1' yazıldı, BL = 00h. **LOOP** : CX = 1
≠ 0 → **print** 'e dön.

Adım 20 / 26

DEĞİŞKENLER

AH = 02h

DL = 31h

BL = 00h

CX = 0001h

ZF = 1

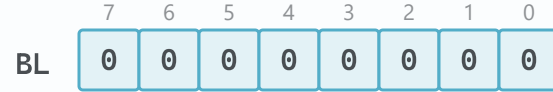
CF = 1

ÇIKTI

0000111

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
▶ loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



Tur 8 (özet): bit = 0, '0' yazıldı, BL = 00h. **LOOP** : CX = 0
→ döngü biter.

Adım 21 / 26

DEĞİŞKENLER

AH = 02h

DL = 30h

BL = 00h

CX = 0000h

ZF = 1

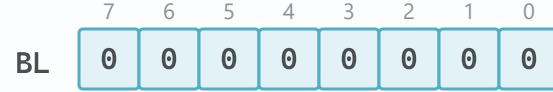
CF = 0

ÇIKTI

00001110

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
▶ mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



DL = 'b'. AH hâlâ 2.

Adım 22 / 26

DEĞİŞKENLER

AH = 02h

DL = 62h

BL = 00h

CX = 0000h

ZF = 1

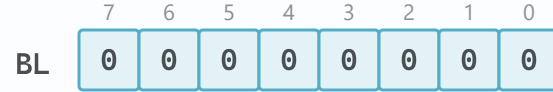
CF = 0

ÇIKTI

00001110

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 ► int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```



INT 21h : 'b' eki yazılır.

Adım 23 / 26

DEĞİŞKENLER

AH = 02h

DL = 62h

BL = 00h

CX = 0000h

ZF = 1

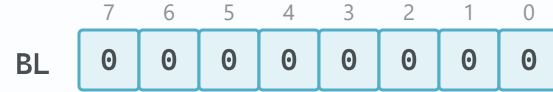
CF = 0

ÇIKTI

00001110b

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 ► mov ah, 0
16 int 16h
17 ret
```



AH = 0: tuş bekleme.

Adım 24 / 26

DEĞİŞKENLER

AH = 00h

DL = 62h

BL = 00h

CX = 0000h

ZF = 1

CF = 0

ÇIKTI

00001110b

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2 ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşa basılmasını bekle
15 mov ah, 0
16 int 16h
17 ret
```

7 6 5 4 3 2 1 0
BL 0 0 0 0 0 0 0 0

CF 0 DL 'b'

INT 16h : kullanıcı bir tuşa basana kadar beklenir (çıkıtı okunabilsin).

Adım 25 / 26

DEĞİŞKENLER

AH = 00h

DL = 62h

BL = 00h

CX = 0000h

ZF = 1

CF = 0

ÇIKTI

00001110b

Kod İzleme (2/2): İkili Yazdırma

```
1 print:
2 mov ah, 2    ; yazdırma fonksiyonu
3 mov dl, '0'
4 test bl, 10000000b ; ilk biti test et
5 jz zero
6 mov dl, '1'
7 zero:
8 int 21h
9 shl bl, 1
10 loop print
11 ; b (binary suffix) yazdır
12 mov dl, 'b'
13 int 21h
14 ; klavye tuşu basılmasını bekle
15 mov ah, 0
16 int 16h
▶ ret
```

7 6 5 4 3 2 1 0
BL 0 0 0 0 0 0 0 0

CF 0 DL 'b'

RET : program biter. Ekranda **00001110b**.

Adım 26 / 26

DEĞİŞKENLER

AH = 00h

DL = 62h

BL = 00h

CX = 0000h

ZF = 1

CF = 0

ÇIKTI

00001110b

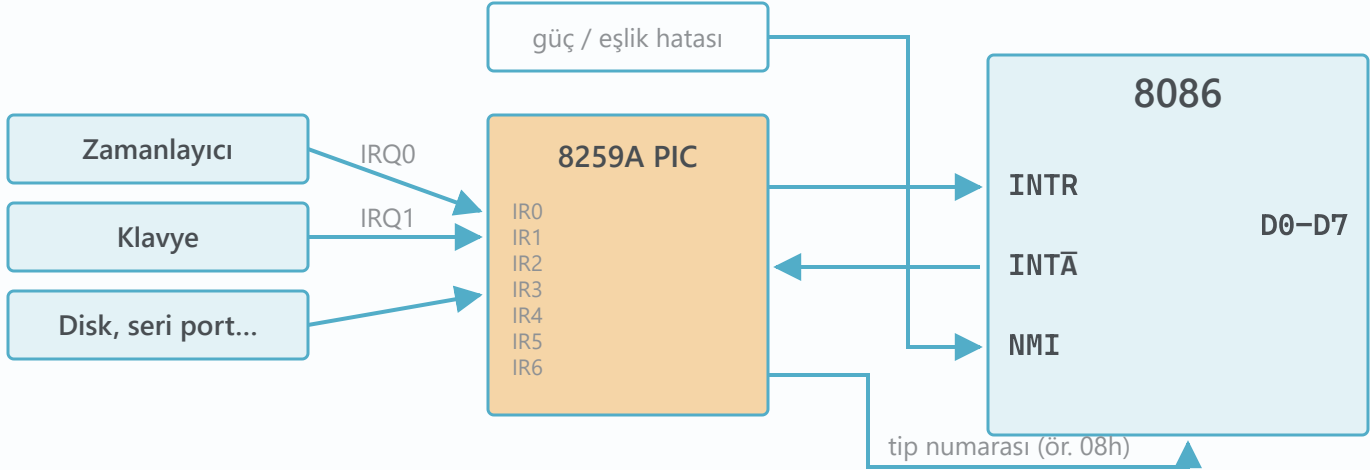
Ekran Çıktısı ve Notlar

```
00001110b_
```

emu8086 ekran penceresi (80×25)

- Sayının içi zaten ikilidir; yazdırmak, her biti **'0'/'1' karakterine** çevirmektir.
- `TEST` sonucu saklamadığı için BL bozulmaz; bozan yalnızca `SHL` 'dir (bilerek).
- `LOOP` CX'i azaltır; döngü içinde CX başka amaçla kullanılmamalıdır.
- `INT 21h/02h` AL'yi değiştirir (AL = DL); AL'de saklanan bir değer varsa önce korunmalıdır.

8086'nın Kesme Pinleri



INTR

Maskelenebilir istek. **IF = 1** ise kabul edilir. **STI** IF'yi 1, **CLI** 0 yapar.

INTA

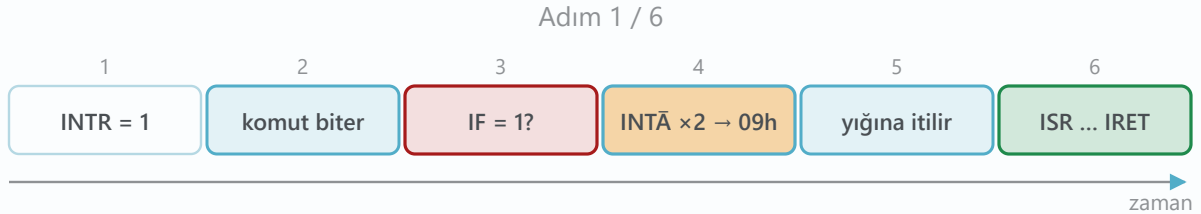
Kesme onayı (çıkış). İki darbe gönderilir; ikincisinde denetleyici tip numarasını D0-D7'ye koyar.

NMI

Maskelenemez; yükselen kenarla tetiklenir, her zaman **INT 2** olarak işlenir.

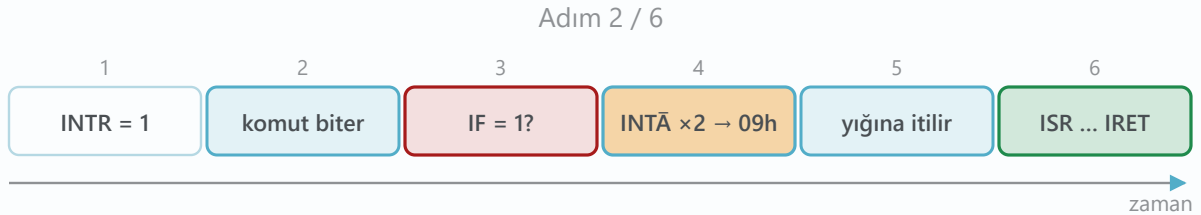
INTR Kabul Süreci

1. Aygıt (ör. klavye) IRQ1 hattını etkinleştirir; 8259A **INTR** pinini 1 yapar.



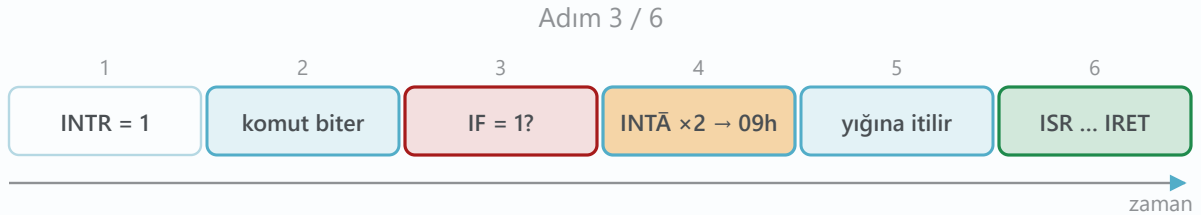
INTR Kabul Süreci

1. Aygıt (ör. klavye) IRQ1 hattını etkinleştirir; 8259A **INTR** pinini 1 yapar.
2. 8086 o anki komutu **bitirir**; kesmeler komut ortasında kabul edilmez.



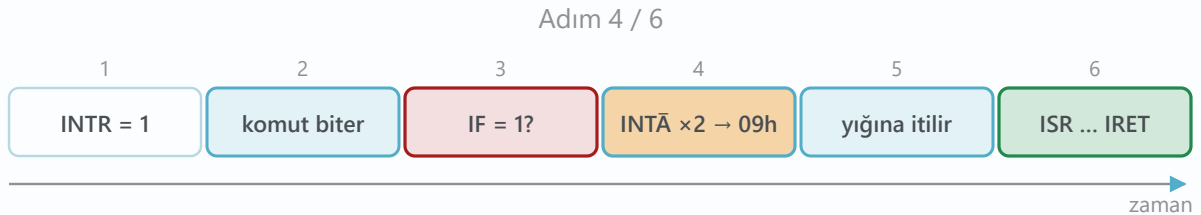
INTR Kabul Süreci

1. Aygıt (ör. klavye) IRQ1 hattını etkinleştirir; 8259A **INTR** pinini 1 yapar.
2. 8086 o anki komutu **bitirir**; kesmeler komut ortasında kabul edilmez.
3. IF = 0 ise istek bekletilir. IF = 1 ise kabul edilir.



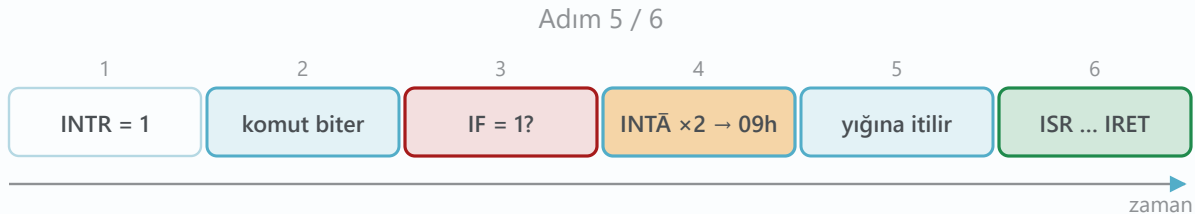
INTR Kabul Süreci

1. Aygıt (ör. klavye) IRQ1 hattını etkinleştirir; 8259A **INTR** pinini 1 yapar.
2. 8086 o anki komutu **bitirir**; kesmeler komut ortasında kabul edilmez.
3. IF = 0 ise istek bekletilir. IF = 1 ise kabul edilir.
4. CPU iki **INT $\bar{A}$** darbesi üretir; ikinci darbede 8259A tip numarasını (PC'de klavye için 09h) veri yoluna koyar.



INTR Kabul Süreci

1. Aygıt (ör. klavye) IRQ1 hattını etkinleştirir; 8259A **INTR** pinini 1 yapar.
2. 8086 o anki komutu **bitirir**; kesmeler komut ortasında kabul edilmez.
3. IF = 0 ise istek bekletilir. IF = 1 ise kabul edilir.
4. CPU iki **INT $\bar{A}$** darbesi üretir; ikinci darbeye 8259A tip numarasını (PC'de klavye için 09h) veri yoluna koyar.
5. Bundan sonrası **INT n** ile aynıdır: FLAGS, CS, IP yığına; IF = TF = 0; CS:IP $\leftarrow [n \times 4]$.



INTR Kabul Süreci

1. Aygıt (ör. klavye) IRQ1 hattını etkinleştirir; 8259A **INTR** pinini 1 yapar.
2. 8086 o anki komutu **bitirir**; kesmeler komut ortasında kabul edilmez.
3. IF = 0 ise istek bekletilir. IF = 1 ise kabul edilir.
4. CPU iki **INT $\bar{A}$** darbesi üretir; ikinci darbede 8259A tip numarasını (PC'de klavye için 09h) veri yoluna koyar.
5. Bundan sonrası **INT n** ile aynıdır: FLAGS, CS, IP yığına; IF = TF = 0; CS:IP $\leftarrow [n \times 4]$.
6. ISR işini bitirince 8259A'ya **EOI** (kesme sonu) gönderir ve **IRET** ile döner.

Adım 6 / 6



DONANIM KESMELERİ

Öncelik ve PC Kesme Haritası

8086 öncelik sırası

Sıra	Kaynak
1 (en yüksek)	Bölme hatası, <code>INT n</code> , <code>INT0</code>
2	NMI
3	INTR
4 (en düşük)	Tek adım (TF)

Aynı anda birden çok istek varsa, yüksek öncelikli olan önce hizmet görür.

IBM PC donanım kesmeleri

IRQ	Aygıt	Tip
0	Zamanlayıcı (18,2 Hz)	<code>08h</code>
1	Klavye	<code>09h</code>
3 / 4	Seri portlar	<code>0Bh</code> / <code>0Ch</code>
6	Disket	<code>0Eh</code>
7	Yazıcı	<code>0Fh</code>

Klavyeye basılınca önce donanım kesmesi `INT 09h` çalışır ve tuşu arabelleğe koyar; programın çağırdığı `INT 16h` bu arabellekten okur.

Kendi Kesme Yordamımız

```
org 100h
mov ax, 0
mov es, ax          ; ES = 0: IVT kesimi
cli                ; kesmeleri kapat
mov word ptr es:[90h*4], offset isr ; IP
mov word ptr es:[90h*4+2], cs      ; CS
sti                ; kesmeleri aç
int 90h             ; kendi kesmemiz
ret
isr:
mov ah, 0Eh
mov al, '*'
int 10h
iret                ; RET değil!
```

- Kullanılmayan bir tip seçilir: **90h**. Vektörü $90h \times 4 = 240h$ adresindedir.
- IVT 0000 kesimindedir; ES = 0 yapılır.
- Vektör yazılırken **CLI** / **STI** ile korunur: yarım yazılmış vektörle kesme gelmemelidir.
- **INT 90h** ekrana * yazar ve program devam eder.
- DOS altında **INT 21h**, AH = **25h** (vektör ata) / **35h** (vektör oku) daha güvenli yoldur.

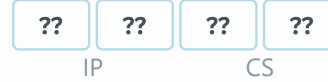
Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 ► mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 int 60h              ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)



CS:IP (sonraki)

0700:0103

FLAGS

F202h

IF = 1 ekran:

AX = 0. Vektör tablosu (IVT) 0000:0000'dadır; tip 60h'nin vektörü $60h \times 4 = 180h$ adresinde.

yığın (SS = 0700h)



Adım 1 / 12

DEĞİŞKENLER

AX = 0000h

ES = 0700h

SP = FFEh

FLAGS = F202h

IF = 1

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

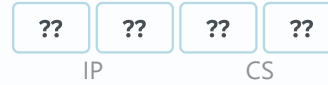
Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 ► mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 int 60h                ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)



CS:IP (sonraki)

0700:0105

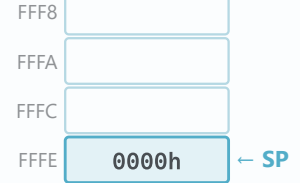
FLAGS

F202h

IF = 1 ekran:

ES = 0000h: ES:[180h] artık IVT'deki 0000:0180h baytını gösterir.

yığın (SS = 0700h)



Adım 2 / 12

DEĞİŞKENLER

AX = 0000h

ES = 0000h

SP = FFEh

FLAGS = F202h

IF = 1

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

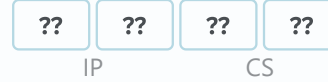
Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 ► cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 int 60h              ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)



CS:IP (sonraki)

0700:0106

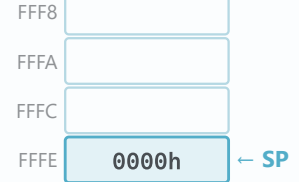
FLAGS

F002h

IF = 0 ekran:

cli : IF = 0. Vektör yarı yazılmışken donanım kesmesi gelmesin. FLAGS = F002h.

yığın (SS = 0700h)



Adım 3 / 12

DEĞİŞKENLER

AX = 0000h

ES = 0000h

SP = FFEh

FLAGS = F002h

IF = 0

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 ► mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 int 60h              ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)



CS:IP (sonraki)

0700:010D

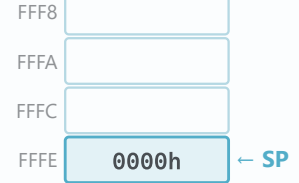
FLAGS

F002h

IF = 0 ekran:

Düşük sözcük = IP = offset isr = 0118h: bellekte 18 01 (little-endian).

yığın (SS = 0700h)



Adım 4 / 12

DEĞİŞKENLER

AX = 0000h

ES = 0000h

SP = FFEh

FLAGS = F002h

IF = 0

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 int 60h              ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)

18	01	00	07
----	----	----	----

IP

CS

vektör = 0700:0118h

CS:IP (sonraki)

FLAGS

0700:0112

F002h

IF = 0 ekran:

Yüksek sözcük = CS = 0700h: 00 07. Vektör artık 0700:0118.

yığın (SS = 0700h)

FFF8

FFFA

FFFC

FFFE

0000h

← SP

Adım 5 / 12

DEĞİŞKENLER

AX = 0000h

ES = 0000h

SP = FFFEh

FLAGS = F002h

IF = 0

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 ► sti
8 mov al, '#'
9 int 60h              ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)

18	01	00	07
----	----	----	----

IP

CS

vektör = 0700:0118h

CS:IP (sonraki)

FLAGS

0700:0113

F202h

IF = 1

ekran:

sti : IF = 1, kesmeler yeniden açık. FLAGS = **F202h**
(8086'da bit 1 ve 12–15. bitler hep 1).

yığın (SS = 0700h)

FFF8

FFFA

FFFC

FFFE

0000h

← SP

Adım 6 / 12

DEĞİŞKENLER

AX = 0000h

ES = 0000h

SP = FFEh

FLAGS = F202h

IF = 1

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 ► mov al, '#'
9 int 60h             ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)

18	01	00	07
----	----	----	----

IP

CS

vektör = 0700:0118h

CS:IP (sonraki)

FLAGS

0700:0115

F202h

IF = 1

ekran:

AL = '#' (23h): ISR'ye yazmaç ile iletilen parametre.

yığın (SS = 0700h)

FFF8

FFFA

FFFC

FFFE

0000h

← SP

Adım 7 / 12

DEĞİŞKENLER

AX = 0023h

ES = 0000h

SP = FFEh

FLAGS = F202h

IF = 1

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 ► int 60h           ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)

18	01	00	07
IP		CS	

vektör = 0700:0118h

CS:IP (sonraki)	FLAGS
0700:0118	F002h

IF = 0 ekran:

int 60h : FLAGS (F202h), CS (0700h), dönüş IP'si (0117h)
yığına; IF = 0; CS:IP ← [180h] = 0700:0118.

yığın (SS = 0700h)

FFF8	0117h	← SP
FFFA	0700h	
FFFC	F202h	
FFFE	0000h	

Adım 8 / 12

DEĞİŞKENLER

AX = 0023h

ES = 0000h

SP = FFF8h

FLAGS = F002h

IF = 0

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 int 60h              ; kendi kesmemiz
10 ret
11 isr:
12 ► mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)

18	01	00	07
----	----	----	----

IP

CS

vektör = 0700:0118h

CS:IP (sonraki)

FLAGS

0700:011A

F002h

IF = 0

ekran:

ISR içinde

ISR çalışıyor: AH = 0Eh (teletype).

yığın (SS = 0700h)

FFF8	0117h	← SP
FFFA	0700h	
FFFC	F202h	
FFFE	0000h	

Adım 9 / 12

DEĞİŞKENLER

AX = 0E23h

ES = 0000h

SP = FFF8h

FLAGS = F002h

IF = 0

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 int 60h              ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret

```

IVT 0000:0180h (tip 60h)

18	01	00	07
----	----	----	----

IP

CS

vektör = 0700:0118h

CS:IP (sonraki)

FLAGS

0700:011C

F002h

IF = 0

ekran: #

ISR içinde

int 10h : ekrana # yazılır. ISR içinden başka bir kesme çağrılabilir.

yığın (SS = 0700h)

FFF8	0117h	← SP
FFFA	0700h	
FFFC	F202h	
FFFE	0000h	

Adım 10 / 12

DEĞİŞKENLER

AX = 0E23h

ES = 0000h

SP = FFF8h

FLAGS = F002h

IF = 0

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

Kod İzleme: INT 60h ile Kendi ISR'miz

```

1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 int 60h              ; kendi kesmemiz
10 ret
11 isr:
12 mov ah, 0Eh
13 int 10h
► iret

```

IVT 0000:0180h (tip 60h)

18	01	00	07
----	----	----	----

IP

CS

vektör = 0700:0118h

CS:IP (sonraki)

FLAGS

0700:0117

F202h

IF = 1

ekran: #

ISR içinde

iret : IP = 0117h, CS = 0700h, FLAGS = F202h geri alınır; IF yeniden 1, SP = FFEh.

yığın (SS = 0700h)

FFF8

FFFA

FFFC

FFFE

0000h

← SP

Adım 11 / 12

DEĞİŞKENLER

AX = 0E23h

ES = 0000h

SP = FFEh

FLAGS = F202h

IF = 1

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

Kod İzleme: INT 60h ile Kendi ISR'miz

```
1 org 100h
2 mov ax, 0
3 mov es, ax          ; ES = 0: IVT
4 cli
5 mov word ptr es:[60h*4], offset isr
6 mov word ptr es:[60h*4+2], cs
7 sti
8 mov al, '#'
9 int 60h              ; kendi kesmemiz
▶ ret
11 isr:
12 mov ah, 0Eh
13 int 10h
14 iret
```

IVT 0000:0180h (tip 60h)

18	01	00	07
----	----	----	----

IP

CS

vektör = 0700:0118h

CS:IP (sonraki)

FLAGS

0700:0000

F202h

IF = 1

ekran: #

yığın (SS = 0700h)

FFF8

FFFA

FFFC

FFFE

ret : ana program kaldığı yerden sürdürdü ve DOS'a döner.

Adım 12 / 12

DEĞİŞKENLER

AX = 0E23h

ES = 0000h

SP = 0000h

FLAGS = F202h

IF = 1

Tip 60h kullanıcı programlarına ayrılmış tiplerdendir. Değerler emu8086 benzeri .COM yüklemesi (CS = 0700h) için.

ÖZET

Özet

Kesme

İşlemci işini bırakır, ISR'yi çalıştırır, **IRET** ile kaldığı yere döner.

Vektör tablosu

0000:0000–03FF, 256×4 bayt. Tip $n \rightarrow$ adres $n \times 4$ (IP), $n \times 4 + 2$ (CS).

INT n

FLAGS, CS, IP yığına ($SP - 6$), $IF = TF = 0$, $CS:IP \leftarrow$ vektör.

Alt fonksiyon

AH ile seçilir; parametreler diğer yazmaçlarda.

BIOS

INT 10h/0Eh AL'yi yazar; **INT 16h/00h** tuş bekler (AL = ASCII, AH = tarama kodu).

DOS

INT 21h/02h DL'yi yazar; 09h '\$' ile biten dizgiyi yazar; 4Ch çıkar.

ÖZET

Alıştırmalar

1. `INT 13h` vektörü hangi fiziksel adreslerde bulunur?
2. `INT 21h` çalışmadan önce `SP = FFFh` ise ISR içindeyken `SP` kaçtır? Yığında ne vardır?
3. Ekran **Hello** yazdırmak için kesmeler bölümündeki programı tamamlayın.
4. Ekran `INT 21h/09h` ile "Merhaba Dünya" yazan programı yazın.
5. Basılan tuşun ASCII kodunu ikili olarak yazan bir program tasarlayın.
6. ISR'nin `IRET` yerine `RET` ile bitmesi ne gibi sorunlara yol açar?

► İpuçları / cevaplar

Kaynaklar

- A. S. Tanenbaum, *Modern Operating Systems*, 3. baskı, Pearson, 2008.
- Intel, *The 8086 Family User's Manual*, 1979 — Interrupts bölümü.
- B. B. Brey, *The Intel Microprocessors*, 8. baskı, Pearson, 2009 — Bölüm 6 ve 12.
- emu8086 yardım belgeleri: "Interrupts" ve "Supported interrupts" sayfaları.
- R. Brown, *Ralf Brown's Interrupt List* (BIOS/DOS kesme başvurusu).

Son

Bölüm 5: Kesmeler

Sonraki bölüm: Bölüm 5: emu8086.inc Kütüphanesi · [Terim sözlüğü](#)