BİL 362 Mikroişlemciler: Kesmeler

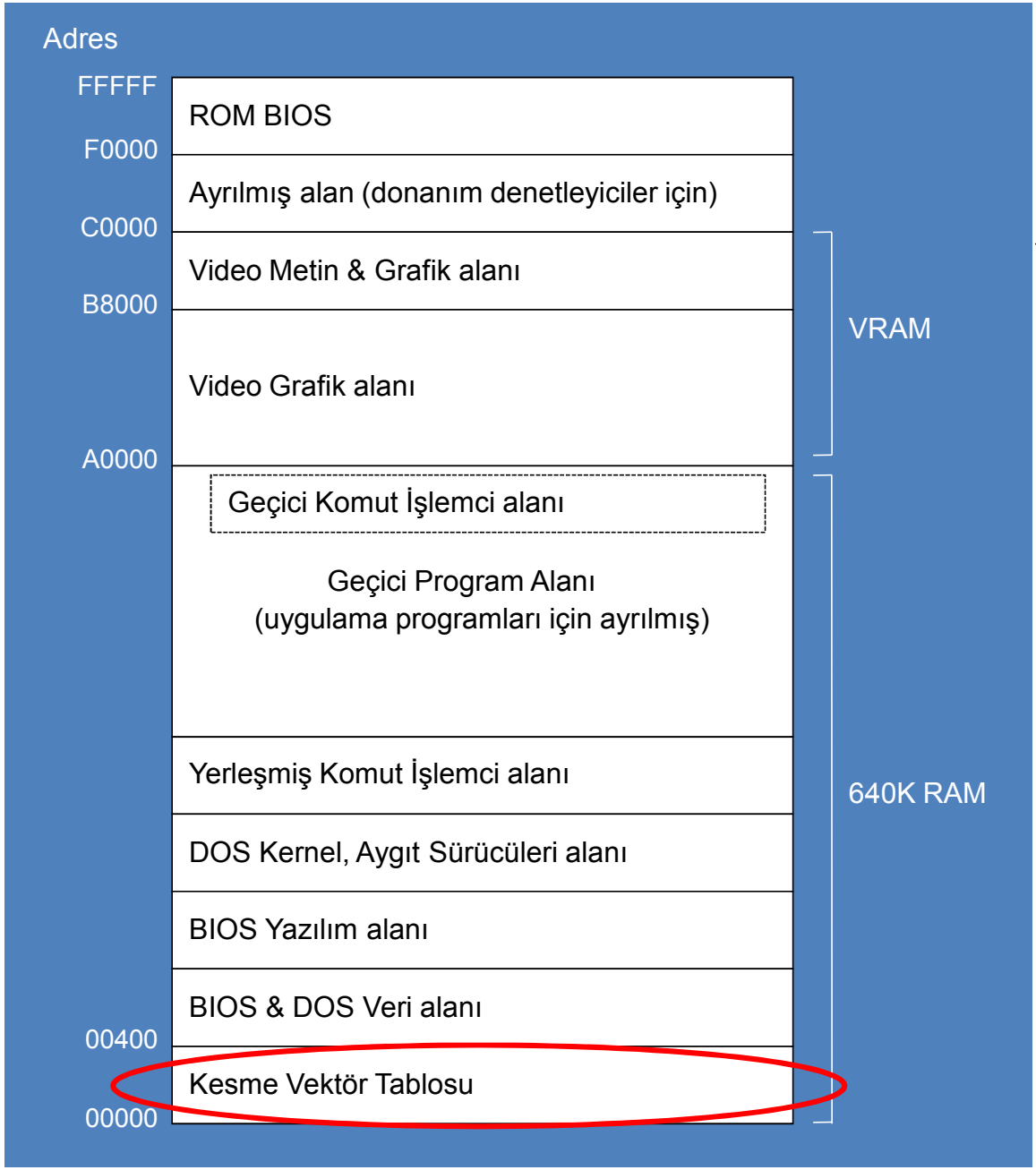
Ahmet Burak Can

abc@hacettepe.edu.tr

Gerçek-Adres Modu
(“Real-Address Mode”)

- Gerçek-adres (16-bit) modundaki programlarda:
  - Maksimum 1 MB adreslenebilir RAM
  - Tek-görevlilik
  - Bellek sınır koruma yok
  - Ofsetler 16 bit

- IBM PC-DOS: IBM-PC için ilk gerçek-adres modlu işletim sistemi
  - Daha sonra MS-DOS olarak Microsoft tarafından sahiplenildi


Adres
FFFFF
ROM BIOS
F0000
Ayrılmış alan (donanım denetleyiciler için)
C0000
Video Metin & Grafik alanı
B8000
Video Grafik alanı
VRAM
A0000
Geçici Komut İşlemci alanı
Geçici Program Alanı
(uygulama programları için ayrılmış)
Yerleşmiş Komut İşlemci alanı
DOS Kernel, Aygıt Sürücüleri alanı
BIOS Yazılım alanı
BIOS & DOS Veri alanı
640K RAM
00400
Kesme Vektör Tablosu
00000

INT Komutu

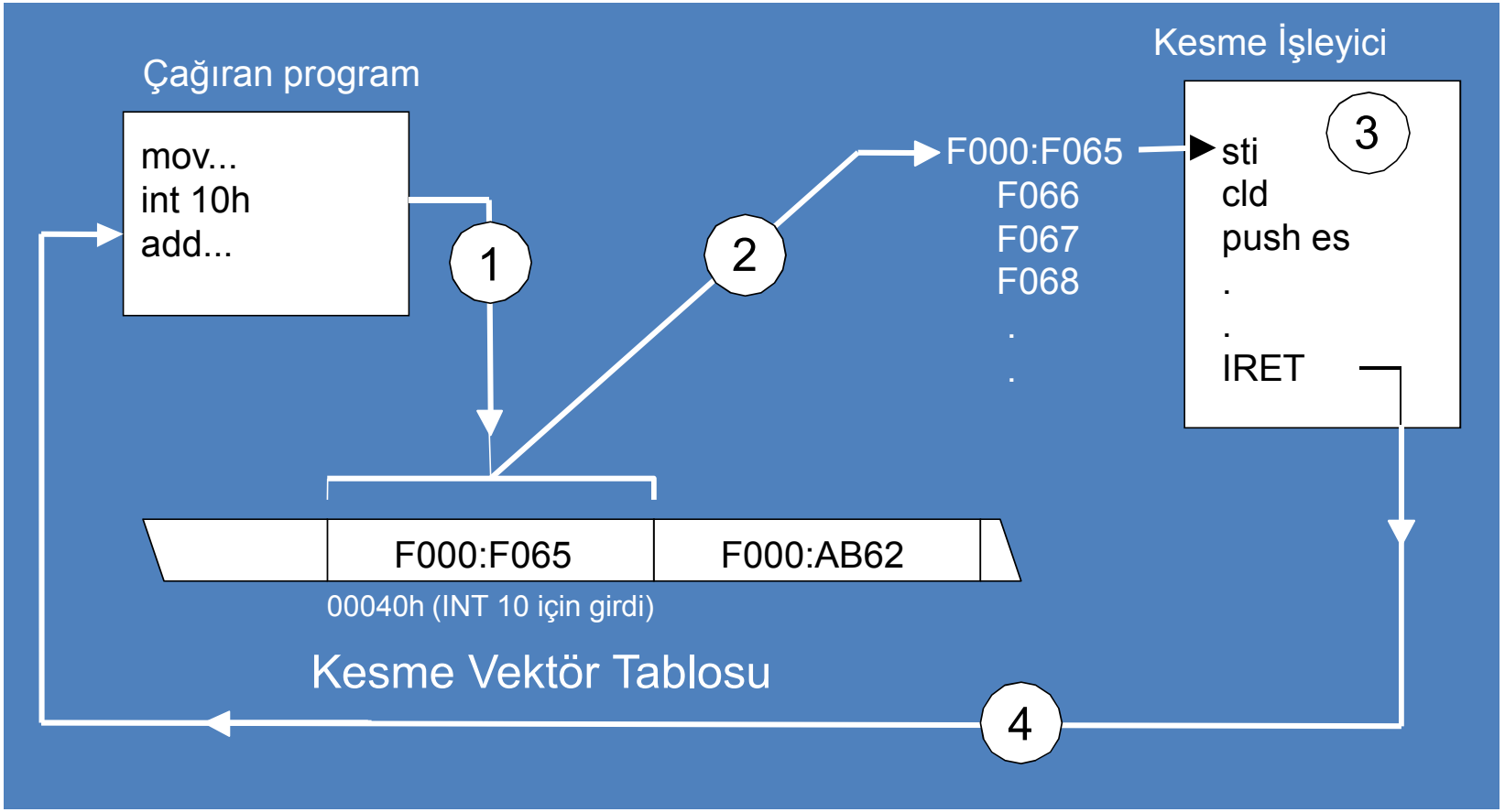
- INT ("call to interrupt procedure") komutu tanımlanmış yazılım kesmesini ("software interrupt") işletir.
  - Bayrakları ve dönüş adresini yığıta koyar.
  - İşleyici kesme yordamını bulmak için, Kesme Vektör Tablosunu kullanır.
- Kesmeyi ele alan koda, kesme işleyici ("interrupt handler") denir.
- Sözdizimi:

```
INT intno
(intno = 0..FFh)
```

Çağrılmadan önce, birtakım yazmaçların ilklendirilmesi gerekir.

Kesme Vektör Tablosu ("Interrupt Vector Table – IVT") her olası kesme işleyici için, 32-bit bölüt-ofset adresi içerir (kesme işleyici bölüt:ofset adresleri her makine için değişir).

Kesme Vektörleme İşlemi



Yaygın Kullanılan Kesmeler

- INT 10h Video Servisleri
- INT 16h Klavye Servisleri
- INT 17h Yazıcı Servisleri
- INT 1Ah Tarih ve Zaman
- INT 1Ch Kullanıcı-tanımlı zamanlayıcı (“Timer”) Kesmesi
- INT 21h MS-DOS Servisleri

INT 10H

- 00H : Video modu ayarlama
- 0EH : Ekrana karakter yazdırma
- 09H : Belli Özelliklerle Karakter Yazdırma

INT 10H (00H): Video Modu Ayarlama

- AH=00H
- AL yazmacına istenen mod yazılır. AL şu değerleri alabilir:

  00h - text mode. 40x25. 16 colors. 8 pages.
  03h - text mode. 80x25. 16 colors. 8 pages.
  13h - graphical mode. 40x25. 256 colors. 320x200 pixels. 1 page.

  Örnek:

```
mov al, 03h
mov ah, 00h
int 10h
```

INT 10H (0EH): Ekrana Karakter Yazdırma

- AH=0EH
- AL yazmacına ekrana yazdırılacak karakter yazılır.

Örnek:

```
mov al, 'a'
mov ah, 0eh
int 10h
```

INT 10H (09H): Belli Özelliklerle Karakter Yazdırma

- AH=09H
- AL yazmacına ekrana yazdırılacak karakter yazılır.
- CX: Karakterin tekrarlanma sayısı
- BL: Karakterin özelliği

Örnek:

```
mov al, 'a'
mov bl, 4
mov cx, 1
mov ah, 09h
int 10h
```

INT 10H (09H) için BL Yazmaç Değerleri

Hex	Renk
0	Siyah
1	Mavi
2	Yeşil
3	Camgöbeği
4	Kırmızı
5	Eflatun
6	Kahverengi
7	Açık gri
8	Koyu gri
9	Açık mavi
A	Açık yeşil
B	Açık camgöbeği
C	Açık kırmızı
D	Açık eflatun
E	Sarı
F	Beyaz

INT 16H (00H): Klavyeden Karakter Okuma

- AH=00H
- Okunan karakter AL yazmacına aktarılır.
- Örnek: Enter (0DH) karakteri girilene kadar girilenleri ekrana yazdıran bir program.

```
OKU:
    MOV AH,00H
    INT 16H
    CMP AL, 'Z'
    JE SON
    MOV AH, 0EH
    INT 10H
    JMP OKU
SON:
```

MS-DOS Sistem Çağrısı: INT 21h

- ASCII Kontrol Karakterleri
- Çıktı İşlevleri
- Girdi İşlevleri
- Tarih/Zaman İşlevleri

INT 4Ch: Süreci (“Programı”) Bitir

- İşleyen süreci (programı) bitirir ve çağıran sürece seçimli olarak 8-bit kod döndürür.
  - 0 dönüş kodu, genellikle başarılı sonlandırma anlamına gelir.

- Örnek:

```
mov ah,4Ch              ; süreci bitir
mov al,0                ; dönüş kodu
int 21h

; Aşağıdakine denktir:

.EXIT 0
```

ASCII Kontrol Karakterleri

- Çoğu INT 21h işlevi, aşağıdaki kontrol karakterlerine göre işletilir:
  - 08h – “Backspace” (sola doğru bir kolon kayar)
  - 09h – “Horizontal tab” (n kolon ileri atlar)
  - 0Ah – “Line feed” (bir sonraki çıktı satırına geçer)
  - 0Ch – “Form feed” (bir sonraki yazıcı sayfasına geçer)
  - 0Dh – “Carriage return” (en soldaki çıktı kolonuna geçer)
  - 1Bh – “Escape” karakteri

Seçilmiş Çıktı İşlevleri

- ASCII kontrol karakterleri:
  - 02h, 06h – Standart çıktıya karakter yazar
  - 05h – Varsayılan yazıcıya karakter yazar
  - 09h – Standart çıktıya dizgi yazar

(INT 21h İşlevleri) 02h ve 06h:

Standart Çıktıya Karakter Yazmak

Standart çıktıya “A” harfini yazmak için:

```
mov ah,02h
mov dl,'A'
int 21h
```

Standart çıktıya “backspace” karakteri yazmak için (06H işlevinde, DL’nin değeri 0...254 aralığında ise ekrana yazılır. 255 ise klavyeden okunur.):

```
mov ah,06h
mov dl,08h
int 21h
```

(ZF=0 ise AL, karakterini ASCII kodunu taşır.)

(INT 21h İşlevi) 05h:

Varsayılan Yazıcıya Karakter Yazmak

Varsayılan yazıcıya “A” harfini yazmak için:

```
mov ah,05h
mov dl,65
int 21h
```

Varsayılan yazıcıya “horizontal tab” karakteri yazmak için:

```
mov ah,05h
mov dl,09h
int 21h
```

(INT 21h İşlevi) 09h:
Standart Çıktıya Dizgi Yazmak

- Dizgi '$' karakteriyle sonlanmalıdır.

- DS, dizginin tanımlandığı bölüte referans etmeli ve DX, dizginin ofsetini içermelidir.

```
.data
string BYTE “Bu bir dizgidir.$”

.code
mov ax,@data
mov ds,ax

mov  ah,9
mov  dx,OFFSET string
int  21h
```

Seçilmiş Girdi İşlevleri

- ASCII kontrol karakterleri:
  - 01h, 06h – Standart girdiden karakter okur
  - 0Ah – Standart girdiden arabellekteki diziye okur.
  - 0Bh – Standart girdi arabelleğinin durumunu okur.
  - 3Fh – Dosya veya aygıttan okur

(INT 21h İşlevi) 01h:

Standart Girdiden Karakter Okumak

- Girilen karakteri yankılandırır.
- Arabellek boşsa karakterin girilmesini bekler.
- “Ctrl-Break” (^C) için kontrol eder.

```
.data
char BYTE ?
.code
mov ah,01h
int 21h
mov char,al
```

(AL, karakterini ASCII kodunu taşır.)

(INT 21h İşlevi) 06h:

Standart Girdiden Beklemeden Karakter Okumak

- Girilen karakteri yankılandırmaz.
- Girdi için beklemez (girilen karakteri kontrol etmek için sıfır bayrağını (ZF) kullanır).
- Örnek: Bir karakter basılana kadar döngüyü tekrar etmek.

```
.data
char BYTE ?
.code
L1: mov  ah,06h             ; klavye girdisi
    mov  dl,0FFh            ; girdi için bekleme
    int  21h
    jz   L1                 ; karakter yok: döngüyü tekrar et
    mov  char,al            ; karakter basildi: sakla
    call DumpRegs           ; yazmaçları görüntüle
```

(ZF=0 ise AL, karakterini ASCII kodunu taşır.)

(INT 21h İşlevi) 0Ah:
Standart Girdiden Arabellekteki Diziyi Okumak - 1

- Maksimum girdi boyunu tanımlayan ve girdi karakterlerini tutan bir yapının önceden oluşturulması gerekir.

- Örnek:

```
count = 80

KEYBOARD STRUCT
    maxInput BYTE count ; girilebilecek max karakter sayisi
    inputCount BYTE ?   ; gerçek girdi sayisi
    buffer BYTE count DUP(?)  ; girdi karakterlerini tutar
KEYBOARD ENDS
```

(Kesme işletildiğinde *buffer* , girdi karakterlerini taşır.)

(INT 21h İşlevi) 0Ah:

Standart Girdiden Arabellekteki Diziyi Okumak - 2

Kesmeyi işletmek için:

```
.data
kybdData KEYBOARD <>

.code
    mov ah,0Ah
    mov dx,OFFSET kybdData
    int 21h
```

“Enter” (veya ^C) tuşu basılana kadar girilen arabellekteki dizgiyi standart girdiden okur.

İşlev bittiğinde *inputCount* , okunan karakter sayısını döndürür.

(INT 21h İşlevi) 0Bh:
Standart Girdi Arabelleğinin Durumunu Okumak

- “Ctrl-Break” (^C) ile kesilebilir.
- Örnek: Bir tuş basılana kadar döngüyü tekrar et; basılan karakteri sakla.

```
L1:  mov ah,0Bh          ; arabelleğin durumunu oku
     int 21h
     cmp al,0            ; arabellek boş mu?
     je  L1              ; evet: tekrar dön
     mov ah,1            ; hayır: tuşu oku
     int 21h
     mov char,al         ; ve sakla
```

(Bekleyen karakter varsa AL=0FFh, yoksa AL=0)

Örnek: Dizgi Şifreleme

Standart girdiden okur, her baytı şifreler ve standart çıktıya yazar.

```
XORVAL = 239                ; 0-255 arasında herhangi bir değer
.code
main PROC
     mov  ax,@data
     mov  ds,ax
L1:  mov  ah,6              ; doğrudan klavyeden girdi
     mov  dl,0FFh           ; karakter için bekleme
     int  21h               ; AL = karakter
     jz   L2                ; ZF = 1 ise çık
     xor  al,XORVAL
     mov  ah,6              ; çıktıya yaz
     mov  dl,al
     int  21h
     jmp  L1                ; döngüyü tekrar et
L2:  .exit
```

Örnek: Bir Karakter Dizisi Okuma

Aşağıdaki kod kesimi, standart girdiden bir dizgiyi okur ve sonu “null” ile biten bir karakter dizisi döndürür.

```
.data
    buffer BYTE 20 DUP(?)
.code
    mov  si,OFFSET buffer
    mov  cx,LENGTHOF buffer
    dec  cx     ; “null” bayt için yer ayır
L1: mov  ah,1   ; işlev: klavyeden girdi
    int  21h    ; karakteri AL’de döndürür
    cmp  al,0Dh; satır sonu mu?
    je   L2     ; evet: bitir
    mov  [si],al       ; hayır: karakteri sakla
    inc  si     ; arabellek imlecini arttır
    loop L1     ; CX=0 olana kadar dön
L2: mov  BYTE PTR [si],0      ; “null” baytı yerleştir
    ret
```

Tarih/Zaman İşlevleri

- ASCII kontrol karakterleri:
  - 2Ah – Sistem tarihini okur.
  - 2Bh – Sistem tarihini belirler.
  - 2Ch – Sistem zamanını okur.
  - 2Dh – Sistem zamanını belirler.

(INT 21h İşlevi) 2Ah:
Sistem Tarihini Okumak

- Yılı CX, ayı DH, günü DL ve haftanın gününü (pazar = 0) AL yazmacında döndürür.

```
mov  ah,2Ah
int  21h
mov  year,cx
mov  month,dh
mov  day,dl
mov  dayOfWeek,al
```

(INT 21h İşlevi) 2Bh:
Sistem Tarihini Belirlemek

- Tarih günleme başarılı ise AL=0, değilse AL=0FFh döner.

```
mov  ah,2Bh
mov  cx,year
mov  dh,month
mov  dl,day
int  21h
cmp  al,0
jne  failed
```

(INT 21h İşlevi) 2Ch:
Sistem Zamanını Okumak

- Saati (0-23) CH, dakikayı (0-59) CL, saniyeyi (0-59) DH ve saniyenin yüzde birini (0-99) DL yazmacında döndürür.

```
mov  ah,2Ch
int  21h
mov  hours,ch
mov  minutes,cl
mov  seconds,dh
```

(INT 21h İşlevi) 2Dh:
Sistem Zamanını Belirlemek

- Zaman günleme başarılı ise AL=0, değilse AL=0FFh döner.

```
mov   ah,2Dh
mov   ch,hours
mov   cl,minutes
mov   dh,seconds
int   21h
cmp   al,0
jne   failed
```

Standart MS-DOS Dosya Girdi/Çıktı Servisleri

- ASCII kontrol karakterleri:
  - 3Ch – Dosyayı oluşturur
  - 3Dh – Varolan dosyayı açar.
  - 3Eh – Dosyayı (“file handle”) kapatır.
  - 42h – Dosya imlecini (“file pointer”) hareket ettirir.
  - 3fh – Dosyadan okuma
  - 40h – Dosyaya yazma
  - 5706h – Dosyanın oluşturulduğu tarih ve zamanı okur.

(INT 21h İşlevi) 3Ch: Dosya Oluşturmak

- AH = 3Ch
- CX = özellikler (0 = normal, 1 = salt okunur, 2 = gizli, 4 = sistem, 16 = arşiv)
- DS:DX = dosya adının bölüt:ofseti

Örnek: Yeni Bir Dosya Oluşturmak

```
.data
     filename db "c:\myfile.txt", 0
     handle dw ?
.code
     mov ah, 3ch
     mov cx, 0
     mov dx, offset filename
     mov ah, 3ch
     int 21h
     jc err
     mov handle, ax
     jmp k
     err:
     ; ....
     k:
```

(Dosya oluşturma/açma başarılı ise CF=0, AX=dosya tanıtıcı, CX=yapılan işlem; başarısız ise CF=1 döner.)

(INT 21h İşlevi) 3Dh: Varolan Dosya Açmak

- AH = 3Dh
- AL = (0 = normal, 1 = write, 2 = read)
- DS:DX = dosya adının bölüt:ofseti

```
.data
     filename db "c:\myfile.txt", 0
     handle dw ?
.code
     mov al, 2
     mov dx, offset filename
     mov ah, 3dh
     int 21h
     jc err
     mov handle, ax
     jmp k
     err:
     ; ....
     k:
```

(INT 21h İşlevi) 3Eh:
Dosya Tanıtıcıyı Kapatmak

- Dosya açıldığı sırada INT 21h ile döndürülen aynı dosya tanıtıcı (“file handler”) kullanılır.

- Örnek:

```
.data
filehandle WORD ?
.code
     mov  ah,3Eh
     mov  bx,filehandle
     int  21h
     jc   failed
```

(Dosya kapatma başarılı ise CF=0, değilse CF=1 döner.)

(INT 21h İşlevi) 3Fh:
Dosya veya Aygıttan Okumak

- Bayt bloklarını okur.
- “Ctrl-Break” (^C) ile kesilebilir.
- Örnek: Klavyeden dizgi okumak.

```
.data
inputBuffer BYTE 127 dup(0)
bytesRead WORD ?
.code
mov  ah,3Fh
mov  bx,handle                    ; klavyeden okurken handle=0
mov  cx,127                       ; okunacak max bayt sayisi
mov  dx,OFFSET inputBuffer        ; hedef konum
int  21h
mov  bytesRead,ax                 ; karakter sayisini sakla
```

(AX, okunan bayt sayısını döndürür.)

(INT 21h Function) 40h:
Dosya veya Aygıta Dizgi Yazmak

Girdi: BX = dosya veya aygıt işleyici (“handler”) (konsol = 1),
CX = yazılacak bayt sayısı,
DS:DX = dizinin adresi

```
.data
mesaj “Dizgi yazıyoruz.”
baytSayisi WORD ?

.code
    mov ax, @data
    mov ds,ax

    mov ah,40h
    mov bx, handle                ;Ekrana yazarken handle=1
    mov cx,LENGTHOF mesaj
    mov dx,OFFSET mesaj
    int 21h
    mov baytSayisi,ax     ; ax = yazilan bayt sayisini döndürür
```

(INT 21h İşlevi) 42h:
Dosya İmlecini Hareket Ettirmek

Bir (metin veya ikili) dosyaya rastgele erişime izin verir.

```
mov   ah,42h
mov   al,0              ; başlangıca göre ofset
mov   bx,handle
mov   cx,offsetHi       ; CX:DX = 32-bit ofset
mov   dx,offsetLo
int   21h
```

AL imlecin ofsetinin nasıl hesaplandığını gösterir:

- 0: Dosyanın başlangıcına göre ofset
- 1: İmlecin mevcut durumuna göre ofset
- 2: Dosyanın sonuna göre ofset

(İmleç hareketi başarılı ise CF=0 ve DX:AX=başlangıca göre yeni konumun ofseti; başarısız ise CF=1 döner.)

Örnek: Yeni Dosya Açıp, Yazma, Okuma -1

```
.data
   filename db "myfile.txt", 0
   handle dw ?
   data db " hello files! "
   data_size=$-offset data
   buffer db 4 dup(' ')
.code
   mov ah, 3ch
   mov cx, 0
   mov dx, offset filename
   mov ah, 3ch
   int 21h                          ; create file...
   mov handle, ax

   mov bx, handle
   mov dx, offset data
   mov cx, data_size
   mov ah, 40h
   int 21h                          ; write to file...
```

Örnek: Yeni Dosya Açıp, Yazma, Okuma - 2

```
mov al, 0
mov bx, handle
mov cx, 0
mov dx, 7
mov ah, 42h
int 21h                         ; seek...

mov bx, handle
mov dx, offset buffer
mov cx, 4
mov ah, 3fh
int 21h                         ; read from file...

mov bx, handle
mov ah, 3eh
int 21h                         ; close file...
ret
```