

# MİKROİŞLEMCİLER (BLM202)

## DERS- 11

Dr. Bilgin YAZLIK, RTTP, PMP



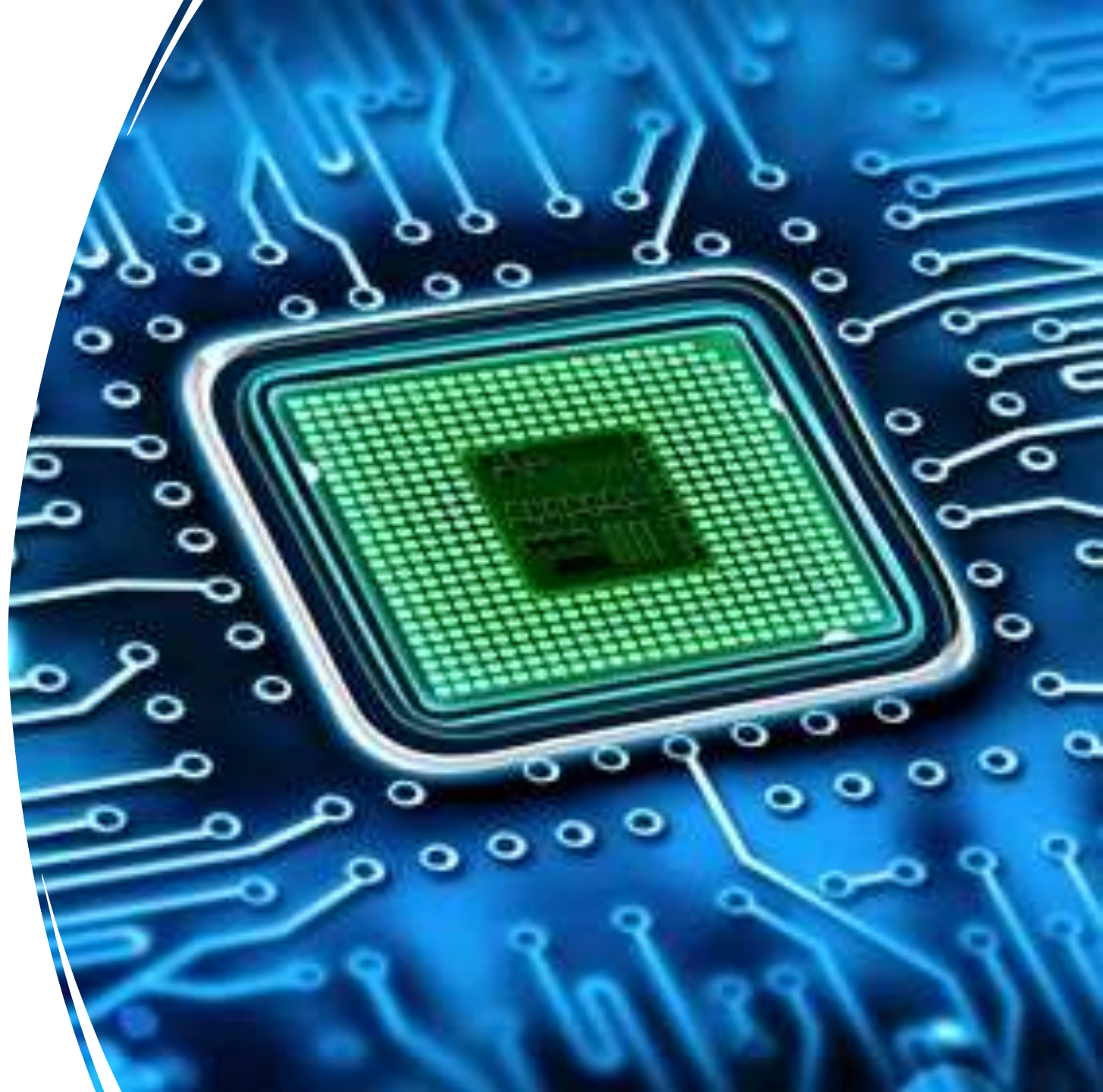
**BİLGİSAYAR MÜHENDİSLİĞİ**



# 11. DERS

---

- Alt Programlar
- Kesmeler



# Alt Programlar (Procedure)

Büyük programları tek bir kod bloğu şeklinde tanımlamak, programın gerçekleştirimi ve bakımı açısından sorunludur. Alt programlar kullanılarak bu sorunun üstesinden gelinebilir.

Programların işleyişi bir ana alt programla başlar, daha sonra ana alt Programdan çağrılan alt programlar kendisine tanımlanmış işlemleri yapıp, çağrıldıkları yere geri dönerler.

# CALL Komutu

- CALL komutu, JMP komutu gibi programın sıra düzensel akışını değiştirerek, komuta parametre olarak verilen bir alt programdan komut işletimine devam edilmesini sağlar.
- CALL komutu ile çağrılan alt program, tanımlanmış işleri bitirdikten sonra, ilk çağrıldığı adrese RET komutu ile geri döner.
- CALL komutu ile bir alt program çağrılırken dönüş adresi yığıta atılır.





# CALL NEAR – CALL FAR

- Eğer CALL komutu ile çağrılan alt program, çağrının yapıldığı bölüt içinde tanımlanmışsa **CALL NEAR** çağrısı yapılır. Bu durumda dönüş adresi olarak yığita **IP yazmacı** atılır.
- Eğer CALL komutu ile çağrılan alt program, başka bir bölüt içinde tanımlanmışsa, **CALL FAR** çağrısı yapılır. Bu durumda dönüş adresi olarak yığita şu yazmaçlar atılır:
  - **CS yazmacı**
  - **IP yazmacı**



# TANIMLAMA

- dene1 ve dene2 adlı iki alt program tanımı şu şekilde yapılır:
- **dene1 proc far**
- ....
- **call dene2**
- ....
- **ret**
- **dene1 endp**
- **dene2 proc**
- ....
- **ret**
- **dene2 endp**





# ÖRNEK

- ORG 100H
- CALL M1
- MOV AX, 2
- RET ; işletim sistemine dön
- M1 PROC
- MOV BX, 5
- RET ; çağrıldığı yere dön.
- M1 ENDP
- END

# ÖRNEK

- ORG 100h
- MOV AL, 1
- MOV BL, 2
- CALL m2
- CALL m2
- CALL m2
- CALL m2
- RET ;işletim sistemine dön
- m2 PROC
- MUL BL ;  $AX = AL * BL$ .
- RET ; return to caller.
- m2 ENDP
- END





# ALT PROGRAMLARA DEĞER AKTRMA

---

- Alt Programlara Değer Aktarma
  - Alt Programlara şu şekilde değer aktarılabilir:
    - Registerları kullanarak
    - Alt Programlarca erişilebilir bellek alanları tanımlayarak



# REGISTER KULLANARAK DEĞER AKTARMA

- Code segment
- Mov dx,011FFH
- Call dene
- hlt
- Dene proc
- Xor dx,0FFFFH
- Mov ax,dx
- ret
- Dene endp
- ends

# ERİŞİLEBİLİR BELLEK ALANLARI TANIMLAYARAK DEĞER AKTARMA

- Veri bölümü içinde bir bellek alanı tanımlanıp, Alt Programda bu bellek alanı üzerinde işlem yapılabilir.
- ORG 100H
- .DATA
- VAR1 DW 1
- VAR2 DW 3
- .CODE
- **CALL DENE**
- HLT
- **DENE PROC**
- MOV AX, VAR1
- OR AX, VAR2
- MOV VAR1, AX
- **RET**
- DENE ENDP

# KESMELER

## INT Komutu

- INT (“call to interrupt procedure”) komutu tanımlanmış yazılım kesmesini (“software interrupt”) işletir.
  - Bayrakları ve dönüş adresini yığıtaya 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).





## KESMELER

### 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

## INT 10H

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



## INT 10H

### 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

### INT 10H (0EH): Ekrana Karakter Yazdırma

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

Örnek:

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



# INT 10 H - ÖZELLİKLER

## 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
```


# ÖZELLİKLER

## 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



# EKRANA METİN YAZDIRMA

- .MODEL SMALL
  - .STACK 100H
  - .DATA
  - ;The string to be printed
  - STRING DB 'This is a sample string', '\$'
  - .CODE
  - MAIN PROC FAR
  - MOV AX,@DATA
  - MOV DS,AX
  - ; load address of the string
  - LEA DX,STRING
  - ;output the string
  - ;loaded in dx
  - MOV AH,09H
  - INT 21H
  - ;interrupt to exit
  - MOV AH,4CH
  - INT 21H
  - MAIN ENDP
  - END MAIN
- 

# KLAVYEDEN KARAKTER OKUMA

## 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:

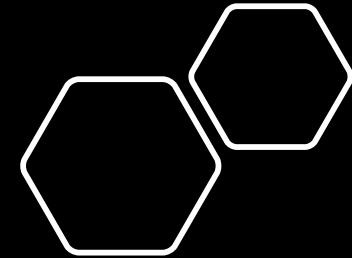


# INT 21H

## MS-DOS Sistem Çağrısı: INT 21h

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

AH	Description	AH	Description
01	<a href="#">Read character from STDIN</a>	02	<a href="#">Write character to STDOUT</a>
05	<a href="#">Write character to printer</a>	06	<a href="#">Console Input/Output</a>
07	<a href="#">Direct char read (STDIN), no echo</a>	08	<a href="#">Char read from STDIN, no echo</a>
09	<a href="#">Write string to STDOUT</a>	0A	<a href="#">Buffered input</a>
0B	<a href="#">Get STDIN status</a>	0C	<a href="#">Flush buffer for STDIN</a>
0D	<a href="#">Disk reset</a>	0E	<a href="#">Select default drive</a>
19	<a href="#">Get current default drive</a>	25	<a href="#">Set interrupt vector</a>
2A	<a href="#">Get system date</a>	2B	<a href="#">Set system date</a>
2C	<a href="#">Get system time</a>	2D	<a href="#">Set system time</a>
2E	<a href="#">Set verify flag</a>	30	<a href="#">Get DOS version</a>
35	<a href="#">Get Interrupt vector</a>		
36	<a href="#">Get free disk space</a>	39	<a href="#">Create subdirectory</a>
3A	<a href="#">Remove subdirectory</a>	3B	<a href="#">Set working directory</a>
3C	<a href="#">Create file</a>	3D	<a href="#">Open file</a>
3E	<a href="#">Close file</a>	3F	<a href="#">Read file</a>
40	<a href="#">Write file</a>	41	<a href="#">Delete file</a>
42	<a href="#">Seek file</a>	43	<a href="#">Get/Set file attributes</a>
47	<a href="#">Get current directory</a>	4C	<a href="#">Exit program</a>
4D	<a href="#">Get return code</a>	54	<a href="#">Get verify flag</a>
56	<a href="#">Rename file</a>	57	<a href="#">Get/Set file date</a>





# INT 21H - Örnek

- **AH = 2Ah - GET SYSTEM DATE**
- Return: CX = year (1980-2099)  
DH = month DL = day AL = day of week (00h=Sunday)

MOV AH, 2AH

INT 21H



# INT 21H - Örnek

- **AH = 2Ch - GET SYSTEM TIME**
- Return: CH = hour CL = minute DH = second DL = 1/100 seconds

```
MOV AH, 2CH
```

```
INT 21H
```



INT 21H;  
MOV AH,  
4CH

## 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
```

# Kaynaklar

- Feza Buzluca, İTÜ Ders Notları, Bilgisayar Mimarisi
- Wikipedia
- Emel Soylu, Kadriye Öz, Karabük Üniversitesi, Mikroişlemciler Ders Notları
- 1) [Bilgisayar Mimarisi – Doç. Dr. Şirzat KAHRAMANLI](#)
- 2) [Ders Notları – Yrd. Doç. Dr. Rifat KURBAN](#)
- Wikipedia
- <https://edukedar.com/difference-between-cisc-and-risc/>
- Dr. B. B. Hegde First Grade College, Kundapura

