

BİL 362 Mikroişlemciler: Yordamlar ve CALL Komutu

Ahmet Burak Can

abc@hacettepe.edu.tr

Yordamlar (Procedure)

- Büyük programları tek bir kod bloğu şeklinde tanımlamak, programın gerçekleştirimi ve bakımı açısından sorunludur.
- Bu nedenle, 8086 çevirici dilinde programları yordamlar (*procedure*) şeklinde tanımlama imkanı sağlanmıştır.
- Programların işleyişi bir ana yordamla başlar, daha sonra ana yordamdan çağrılan alt yordamlar kendisine tanımlanmış işlemleri yapıp, çağrıldıkları yere geri dönerler.

CALL Komutu

- CALL komutu, JMP komutu gibi programın sıradüzensel akışını değiştirerek, komuta parametre olarak verilen bir yordamdan komut işletimine devam edilmesini sağlar.
- CALL komutu ile çağrılan yordam, tanımlanmış işleri bitirdikten sonra, ilk çağrıldığı adrese RET komutu ile geri döner.
- CALL komutu ile bir yordam çağrılırken dönüş adresi yığta atılır.

CALL NEAR ve CALL FAR Komutları

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

Yordam Tanımları

- dene1 ve dene2 adlı iki yordam tanımı şu şekilde yapılır:

```
dene1 proc far
```

```
....
```

```
call dene2
```

```
....
```

```
ret
```

```
dene1 endp
```

```
dene2 proc
```

```
....
```

```
ret
```

```
dene2 endp
```

Yordamlara Değer Aktarma

- Yordamlara şu şekilde değer aktarılabilir:
 - Yazmaçları kullanarak
 - Yordamlarca erişilebilir bellek alanları tanımlayarak
 - Yığita işlenecek veriyi atarak
 - Yığita işlenecek verinin adresini atarak

Yazmaç Kullanarak Yordama Değer Aktarma

- Yordamda işlenecek veri bir yazmaca aktarılır ve yordam çağrılır. Örnek:

```
code segment
    mov ax, @data
    mov ds, ax
    mov dx, 011FFH
    call dene
    hlt

dene proc
    xor dx, 0FFFFH
    mov ax, dx
    ret
dene endp
ends
```

Yordamlarca Erişilebilir Bellek Alanları Tanımlayarak Yordama Değer Aktarma

- Veri bölümü içinde bir bellek alanı tanımlanıp, yordamda bu bellek alanı üzerinde işlem yapılabilir. Örnek:

```
data segment
    var1 dw ?
    var2 dw ?
ends

code segment
    mov ax, @data
    mov ds, ax
    call dene
    hlt
    dene proc
        mov ax, var1
        or ax, var2
        mov var1, ax
        ret
    dene endp
ends
```

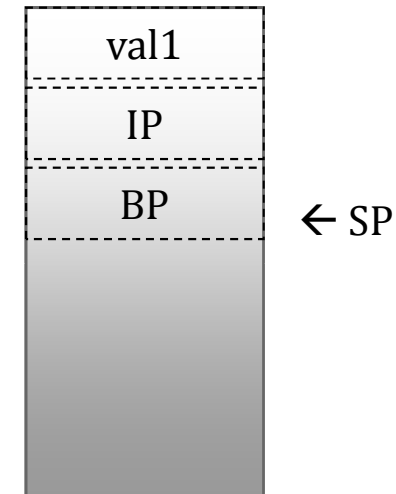

Yığıttan Yordama Değer Aktarma

- Yordama aktarılacak değerler yığita atılır. Yordam içinde yığita atılan değerler okunup işlem yapılır.
- Çağırılan yordamın aynı bölüt içinde olup olmamasına göre yığita atılan değerler değişir.
- Eğer aynı bölüt içinde çağrı yapılmışsa (CALL NEAR), dönüş adresi olarak yığita sadece IP yazmacının içeriği atılır.
- Eğer bölüt dışında bir yordam çağrısı yapılmışsa (CALL FAR), dönüş adresi olarak yığita CS ve IP yazmaçlarının içeriği atılır.
- CALL FAR veya NEAR olmasına göre yığıttan okuma yöntemi değişir.

CALL NEAR Çağrısı ile Yığıttan Yordama Değer Aktarma

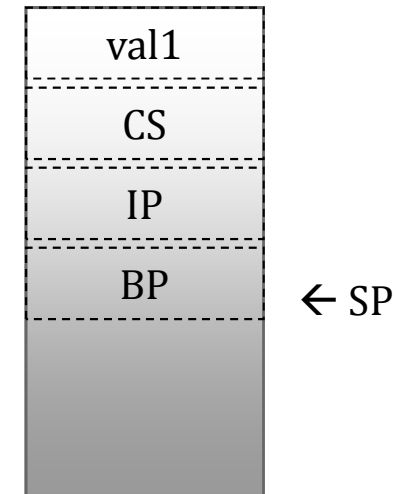
```
data segment
    val1 dw ?
ends
code segment
    mov ax, @data
    mov ds, ax
    push val1
    call dene
    hlt

    dene proc
        push bp
        mov bp, sp
        mov ax, [bp+4]
        xor ax, 1111H
        mov [bp+4], ax
        pop bp
        ret
    dene endp
ends
```



CALL FAR Çağrısı ile Yığıttan Yordama Değer Aktarma

```
data segment
    val1 dw ?
ends
code segment
    mov ax, @data
    mov ds, ax
    push val1
    call dene
    hlt
ends
code2 segment
    dene proc far
        push bp
        mov bp, sp
        mov ax, [bp+6]
        xor ax, 1111H
        mov [bp+6], ax
        pop bp
        ret
    dene endp
ends
```

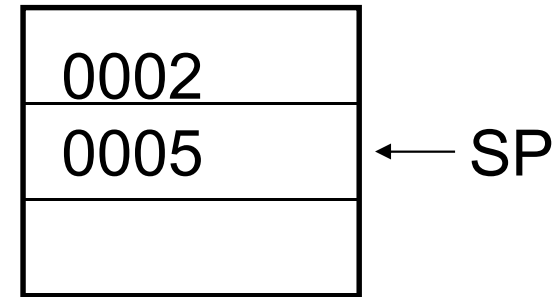


Yığıtta İşlenecek Verinin Adresini Göndererek Yordama Değer Aktarma

```
.data  
count DW 5  
arr    DW 5 DUP(0)
```

```
.code  
push OFFSET arr  
push count
```

(offset arr)
(count)



CALL çağrısı öncesi yığın

Yığıtta İşlenecek Verinin Adresini Göndererek Yordama Değer Aktarma

- **ArrayFill** yordamının, uzunluğu ve adresi verilen, 16-bitlik sayılardan oluşan bir diziye sıfır değeri ataması istenmektedir.
- Yordamı çağırırsan kod kesiminde, dizinin adresi ve dizideki elemanların sayısı yığita atılmakta ve yordama parametre olarak verilmektedir. Yordamın çağrıldığı yer :

```
.data
count db 100
array WORD 100 DUP(?)

.code
    push OFFSET array
    push count
    call ArrayFill
```

Yığıtta İşlenecek Verinin Adresini Göndererek Yordama Değer Aktarma

Böylece ArrayFill yordamı, dizinin adını bilmeden, dizinin içini sıfır değeri doldurabilmektedir. CALL NEAR çağrısı yapıldığı varsayılarakyordam şu şekilde yazılabilir.

```
ArrayFill PROC
    push bp
    mov  bp,sp
    mov  si,[bp+6]
    mov  cx,[bp+4]
dongu:
    mov WORD PTR [si], 0
    add si, 2
    loop dongu
    pop bp
    ret
endp
```

