

BİL 362 Mikroişlemciler: Dizi ve Dizgi İşleme Komutları

Ahmet Burak Can

abc@hacettepe.edu.tr

İçerik

- Dizgi işleme komutları
- Örnek dizgi yordamları
- İki-boyutlu diziler

MOVSb, MOVSw ve MOVSD Komutları

- MOVSB, MOVSW ve MOVSD komutları; SI ile gösterilen bellek bölümündeki veriyi, DI ile gösterilen bellek bölümüne kopyalar.
- SI ve DI otomatik olarak arttırılır / azaltılır:
 - MOVSB 1 arttırır/azaltır
 - MOVSW 2 arttırır/azaltır
 - MOVSD 4 arttırır/azaltır
- Örnek:

```
.data
source DWORD 0FFFFFFFFh
target DWORD ?
.code
mov si,OFFSET source
mov di,OFFSET target
movsd
```

Yön Bayrağı (“Direction Flag”)

- Yön bayrağı, SI ve DI yazmaçlarının artıp azalmasını kontrol eder.
 - $DF = 0 \rightarrow$ SI ve DI arttırılır
 - $DF = 1 \rightarrow$ SI ve DI azaltılır
- Yön bayrağı CLD ve STD komutlarıyla değiştirilebilir.

CLD	; yön bayrağını temizler (0 yapar)
STD	; yön bayrağını 1 yapar

Tekrar (REP) Ön Eki (“Repeat Prefix”)

- MOVSB, MOVSW veya MOVSD komutlarından önce, REP (“repeat prefix”) ön eki kullanılabilir.
 - CX yazmacı tekrar sayısını tutar.
- Örnek: Hedeften kaynağa 20 çift-sözcük kopyala

```
.data
source DWORD 20 DUP(?)
target DWORD 20 DUP(?)
.code
cld                      ; yön = ileri
mov cx,LENGTHOF source  ; REP sayacı
mov si,OFFSET source
mov di,OFFSET target
rep movsd
```

Alıştırma

- Aşağıdaki çift-sözcük dizisindeki ilk elemanı silmek için MOVSD komutunu kullanın (izleyen tüm değerler birer öne kaymalıdır.)

array DWORD 1,1,2,3,4,5,6,7,8,9,10

```
.data
array DWORD 1,1,2,3,4,5,6,7,8,9,10
.code
cld
mov cx, (LENGTHOF array) - 1
mov si, OFFSET array+4
mov di, OFFSET array
rep movsd
```

CMPSB, CMPSW ve CMPSD Komutları

- CMPSB, CMPSW ve CMPSD komutları; SI ile gösterilen bellek bölümündeki veriyi, DI ile gösterilen bellek bölümündeki veriyle karşılaştırır.
 - CMPSB baytları karşılaştırır.
 - CMPSW sözcükleri karşılaştırır.
 - CMPSD çift-sözcükleri karşılaştırır.
- Tekrar ön eki (“repeat prefix”) genellikle kullanılır.
 - REPE (REPZ)
 - REPNE (REPNZ)

Örnek: Çift-Sözcük İkililerini Karşılaştırma

source > target ise, akış L1 etiketi ile devam eder,
değilse L2 etiketine atlar.

```
.data
source DWORD 1234h
target DWORD 5678h

.code
mov si,OFFSET source
mov di,OFFSET target
cmpsd                ; iki çift-sözcüğü karşılaştır
ja L1                ; source > target ise atlar
jmp L2                ; source <= target ise atlar
```


Alıştırma

- Önceki yansıdaki örneği source ve target veri etiketlerini “WORD” tipinde tanımlayarak değiştirin. Gerekli diğer değişiklikleri de yapın.

```
.data
source WORD 12h
target WORD 34h

.code
mov si,OFFSET source
mov di,OFFSET target
cmpsw                ; iki sözcüğü karşılaştır
ja L1                ; source > target ise atlar
jmp L2               ; source <= target ise atlar
```

İki Diziyi Karşılaştırma

İki dizinin karşılıklı elemanlarını karşılaştırmak için, REPE (“repeat while equal”) ön eki kullanılır.

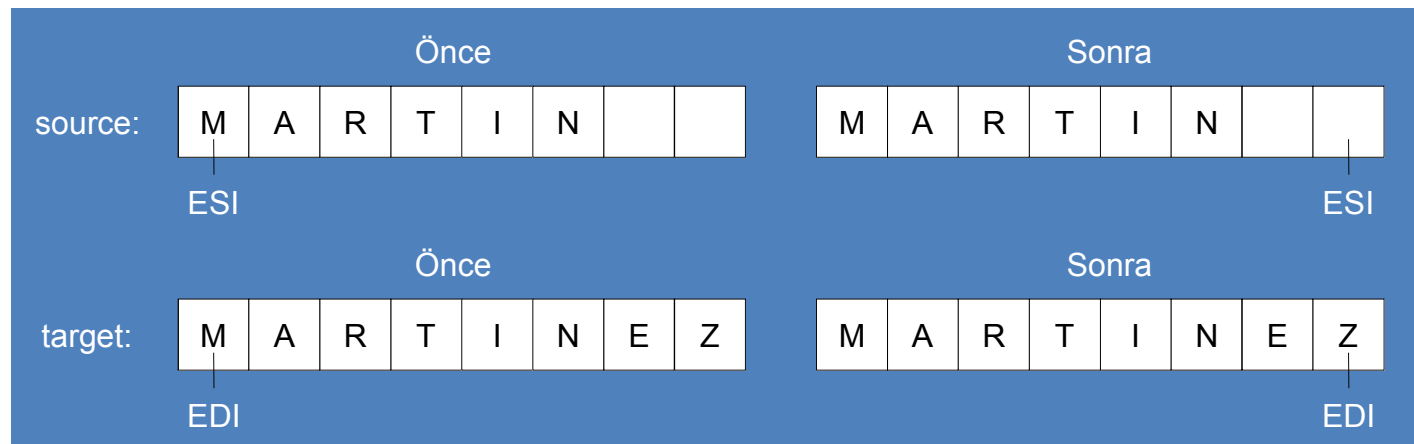
```
.data
COUNT EQU 10
source DWORD COUNT DUP(?)
target DWORD COUNT DUP(?)
.code
mov cx,COUNT                ; tekrar sayacı
mov si,OFFSET source
mov di,OFFSET target
cld                          ; yön = ileri
repe cmpsd                   ; eşitse tekrar et
je L1                        ; son karşılaştırılan
                             ; eşitse iki dizi eşittir.
```

Örnek: İki Dizgiyi Karşılaştırma - 1

Aşağıdaki program iki dizgiyi karşılaştırır (“source” ve “destination”).

(Source değerinin target değerinden küçük olmasına göre ilgili etikete atlar.)

```
.data  
source BYTE "MARTIN "  
target BYTE "MARTINEZ"
```



Örnek: İki Dizgiyi Karşılaştırma - 2

```
.data
    source BYTE "MARTIN  "
    target BYTE "MARTINEZ"
.code
main PROC
    cld                                ; yön = ileri
    mov si,OFFSET source
    mov di,OFFSET target
    mov cx,LENGTHOF source
    repe cmpsb
    jnb source_smaller
    mov dx, di                        ; source daha küçük değil
    jmp done
source_smaller:
    mov dx, si                        ; source daha küçük
done:
    exit
main ENDP
END main
```

SCASB, SCASW ve SCASD Komutları

- SCASB, SCASW ve SCASD komutları; AL/AX/EAX yazmacı içindeki değeri, DI yazmacı ile gösterilen bayt/sözcük/çift-sözcük tipindeki veriyle karşılaştırır.
- Yaygın aramalar:
 - Uzun bir dizi veya dizgi içindeki belirli bir elemanı aramak
 - Verilen bir değeri tutmayan ilk elemanı aramak

SCASB: Örnek

“alpha” dizgisinde “F” harfini arayalım.

```
.data
alpha BYTE "ABCDEFGH",0
.code
mov di,OFFSET alpha
mov al,'F' ; 'F'yi ara
mov cx,LENGTHOF alpha
cld
repne scasb ; eşit değilse tekrar et
jnz quit
dec di ; EDI 'F' i gösterir
```

STOSB, STOSW ve STOSD Komutları

- STOSB STOSW ve STOSD komutları; AL/AX/EAX yazmacı içeriğini, DI yazmacı ile gösterilen bellek adresine kopyalar.
- Örnek: Diziyi 0FFh ile dolduralım.

```
.data
count EQU 100
string1 BYTE count DUP(?)
.code
mov al, 0FFh           ; saklanacak değer
mov di, OFFSET string1 ; EDI hedefi gösterir
mov cx, count          ; karakter sayısı
cld                    ; yön = ileri
rep stosb              ; hedefi AL'nin içeriği ile doldur
```

LODSB, LODSW ve LODSD Komutları

- LODSB, LODSW ve LODSD komutları; SI yazmacı ile gösterilen bayt veya sözcük tipindeki bellek işleneni, AL/AX/EAX yazmacına yükler.
- Örnek:

```
.data
array BYTE 1,2,3,4,5,6,7,8,9
.code
    mov si,OFFSET array      ; ESI kaynağı gösterir
    mov cx,LENGTHOF array   ; karakter sayısı
    cld                     ; yön = ileri
L1:  lodsb                   ; baytı AL yazmacına yükle
    or al,30h                ; convert to ASCII
    mov ah,0Eh
    int 10h                  ; karakteri yaz
    loop L1
```


Örnek: Dizi Çarpma

Bir çift-sözcük dizisinin her elemanını sabit bir sayı ile çarpalım.

```
.data
array DWORD 1,2,3,4,5,6,7,8,9,10
multiplier DWORD 10
.code
    cld                        ; yön = ileri
    mov si,OFFSET array       ; kaynak için dizin yazmacı
    mov di, si                 ; hedef için dizin yazmacı
    mov cx,LENGTHOF array     ; döngü sayacı

L1: lodsd                     ; [SI] içeriğini AX'e yükle
    mul multiplier             ; sabit değer ile çarp
    stosd                      ; AX'i [DI] içeriğinde sakla
    loop L1
```

Alıştırma

- Bir dizideki her paketsiz BCD baytı, ASCII onlu bayta çevirerek yeni bir diziye kopyalayan programı yazın.

```
.data
array BYTE 1,2,3,4,5,6,7,8,9
dest  BYTE (LENGTHOF array) DUP(?)
```

```
    mov si,OFFSET array
    mov di,OFFSET dest
    mov cx,LENGTHOF array
    cld
L1:  lodsb                ; [SI]'ı AL'ye yükle
    or al,30h            ; baytı ASCII'ye çevir
    stosb                ; AL'yi [DI]'a sakla
    loop L1
```

Örnek Dizgi Yordamları

Karakter Dizgisi Karşılaştırma

- *string1* dizgisini *string2* dizgisi ile karşılaştırıp; Elde (CF) ve Sıfır (ZF) bayrakları kurulur.
 - *Karakter dizgilerinin sonunda 0 (sıfır) karakteri olduğunu varsayıyoruz.*
- İşlem sonucu:

```
string1 > string2 → CF=0, ZF=0
```

```
string1 < string2 → CF=1, ZF=0
```

```
string1 == string2 → ZF=1
```

Karakter Dizgisi Karşılaştırma

```
    mov si, OFFSET string1
    mov di, OFFSET string2
L1:  mov  al,[si]
      mov  dl,[di]
      cmp  al,0                ; string1'in sonu mu?
      jne  L2                  ; hayır
      cmp  dl,0                ; evet: string2'nin sonu mu?
      jne  L2                  ; hayır
      jmp  L3                  ; evet: ZF = 1, çık
L2:  inc  si                    ; bir sonraki eleman
      inc  di
      cmp  al,dl               ; karakterler eşit mi?
      je   L1                  ; evet: döngüye devam et
L3:
```

Karakter Dizgisi Uzunluğunu Bulma

- “null” ile biten bir dizginin uzunluğunu bulup, AX yazmacında döndüreceğiz
- Örnek:
 - Aşağıdaki dizi için uzunluk 7 olmalıdır (0 karakteri sayılmaz).

```
.data  
myString BYTE "abcdefg",0
```

Karakter Dizgisi Uzunluğunu Bulma

```
    mov di, OFFSET pString
    mov ax,0                ; karakter sayısı

L1:    cmp byte ptr [di],0    ; dizgi sonu mu?
        je L2                ; evet: bitir
        inc di                ; hayır: bir sonraki eleman
        inc ax                ; sayacı arttır
        jmp L1

L2:
```

Karakter Dizgisi Kopyalama

- “null” ile biten bir dizgiyi kaynak bellek konumundan (“source”) hedef bellek konumuna (“target”) kopyalar.
- Önce bir önceki sayfada verilen kod ile karakter dizgisi uzunluğu bulunur.
- Daha sonra dizgi uzunluğu kadar karakteri kopyalar.

Karakter Dizgisi Kopyalama

```
;Burada karakter dizgisinin boyunu  
;ax içinde döndüren kodu çalıştır.  
...  
  
mov cx, ax                ; REP sayacı  
inc cx                    ; "null" bayt için 1 ekle  
mov si, OFFSET source  
mov di, OFFSET target  
cld                        ; yön = ileri  
rep movsb                 ; dizgiyi kopyala
```

Karakter Dizgisi Tıraşlama (Trim) İşlemi

- “null” ile biten bir dizginin sonundan, verilen bir karakterin tüm tekrarlarını siler.
- Örnek:
 - Aşağıdaki örnekte, “#” karakteri silinince sonuç, myString=“Hello” olacaktır.

```
.data  
myString BYTE "Hello###",0  
char BYTE "#"
```

Karakter Dizgisi Tıraşlama (Trim) İşlemi - 2

- Birden çok koşulu kontrol eder (“#” işareti karakter yerine kullanılmıştır):
 - Dizgi boş olabilir.
 - Dizgi karakterden önce başka karakterler içerebilir (örnek: "Hello##").
 - Dizgi sadece “#” karakterini içerebilir.
 - Dizgi “#” karakterini içermeyebilir (örnek: "Hello" or "H").

Karakter Dizgisi Tıraşlama (Trim) İşlemi - 3

```
mov  di, OFFSET myString

;karakter dizgisinin uzunluğunu bulan kodu çalıştır.
...
cmp  ax,0                ; boş dizgi mi?
je   L2                  ; evet: çık
mov  cx, ax              ; hayır: sayaç = dizgi uzunluğu
dec  ax
add  di, ax              ; DI son karakteri gösterir
mov  al, char            ; kırpılacak karakter
std                      ; yön = geri
repe scasb               ; son kırpılan karakteri atla
jne  L1                  ; ilk karakter silindi mi?
dec  di                  ; DI'ı ayarla: ZF=1 && CX=0
L1: mov  BYTE PTR [di+2],0 ; "null" baytı yerleştir
L2:
```

Karakter Dizgisini Büyük Harfe Çevirme

- Bir dizgiyi tümüyle büyük harfe çevirir; herhangi bir değer döndürmez.

Örnek:

```
.data  
myString BYTE "Hello",0
```

Karakter Dizgisini Büyük Harfe Çevirme

```
L1: mov al,[si]           ; karakteri al
    cmp al,0             ; dizgi sonu mu?
    je L3                ; evet: bitir
    cmp al,'a'           ; 'a'dan küçük mü?
    jb L2                ; 'z'den büyük mü?
    cmp al,'z'           ; 'z'den büyük mü?
    ja L2                ; 'z'den büyük mü?
    dec BYTE PTR [si],32 ; karakteri büyük harfe çevir

L2: inc si               ; sonraki karakter
    jmp L1

L3:
```