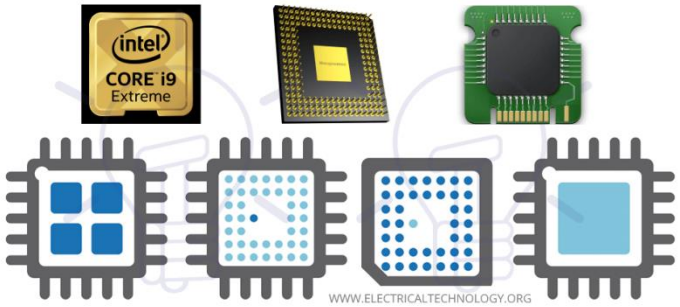


MİKROİŞLEMCİ SİSTEMLERİ

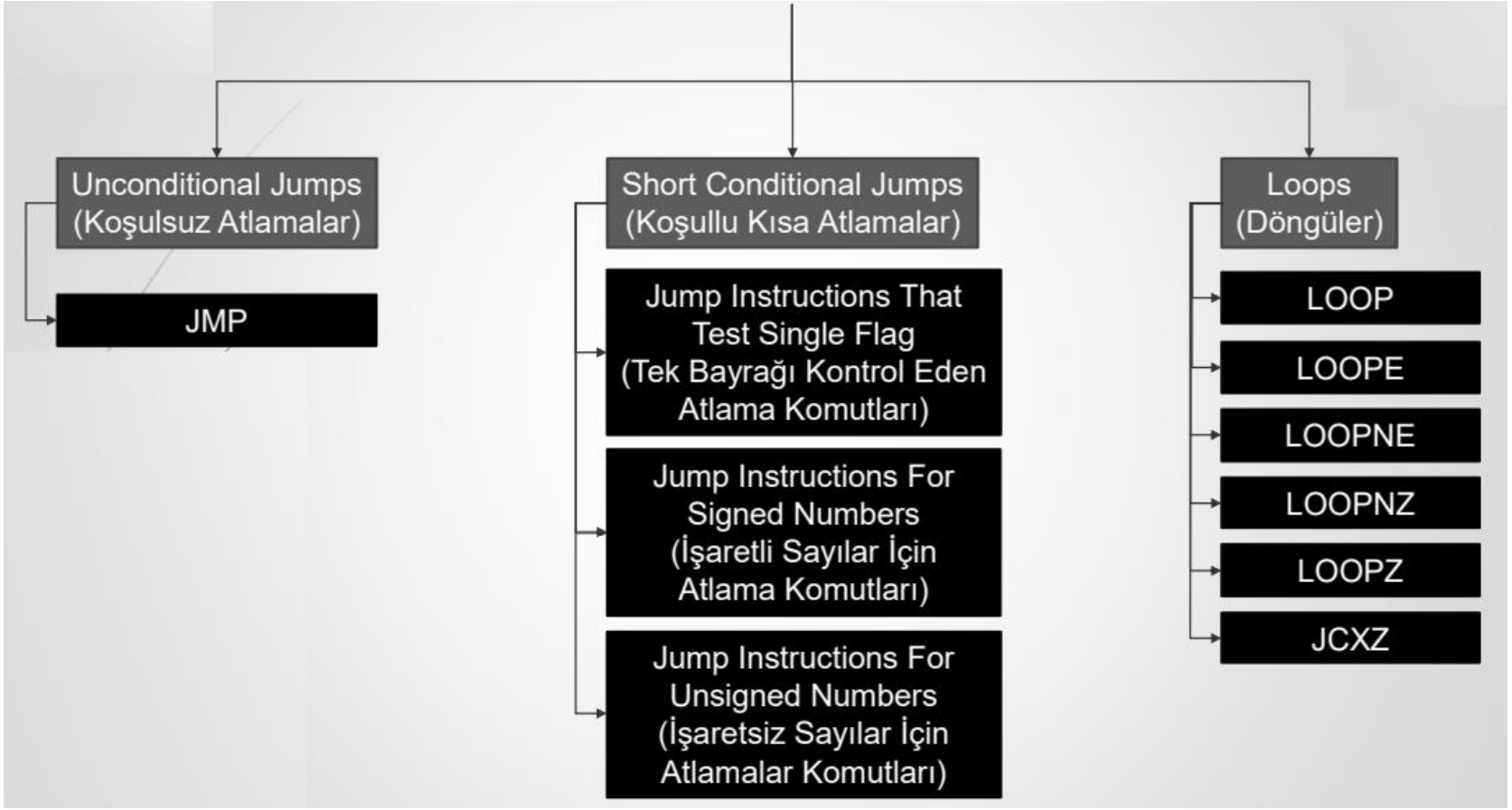
Dr. Öğr. Üyesi Meltem KURT PEHLİVANOĞLU

W-11

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **Program Akış Kontrol Türleri**



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **Döngü Komutları:**

instruction	operation and jump condition	opposite instruction
LOOP	decrease cx, jump to label if cx not zero.	DEC CX and JCXZ
LOOPE	decrease cx, jump to label if cx not zero and equal (zf = 1).	LOOPNE
LOOPNE	decrease cx, jump to label if cx not zero and not equal (zf = 0).	LOOPE
LOOPNZ	decrease cx, jump to label if cx not zero and zf = 0.	LOOPZ
LOOPZ	decrease cx, jump to label if cx not zero and zf = 1.	LOOPNZ
JCXZ	jump to label if cx is zero.	OR CX, CX and JNZ

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

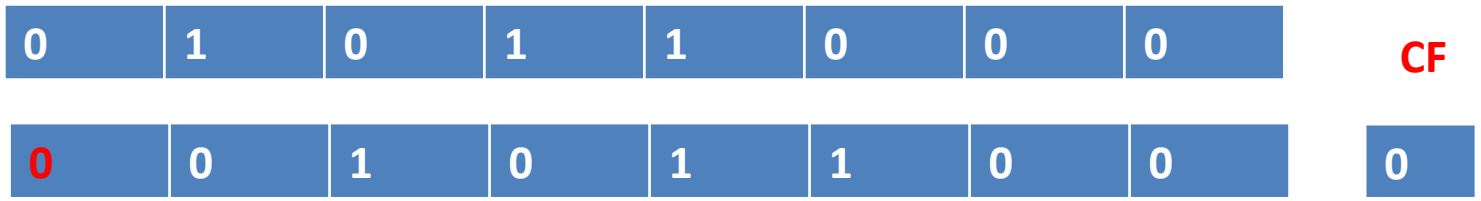
```
org 100h

MOV AL,2
MOV AH,2
MOV BL,1


SUB AL,AH   ; zf=1



dongu1:       ; zf=1 durumuna gecince bu dongu calismamali
INC BL
SUB AH,BL
loopnz dongu1



ret
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **KAYDIRMA VE DÖNDÜRME KOMUTLARI**

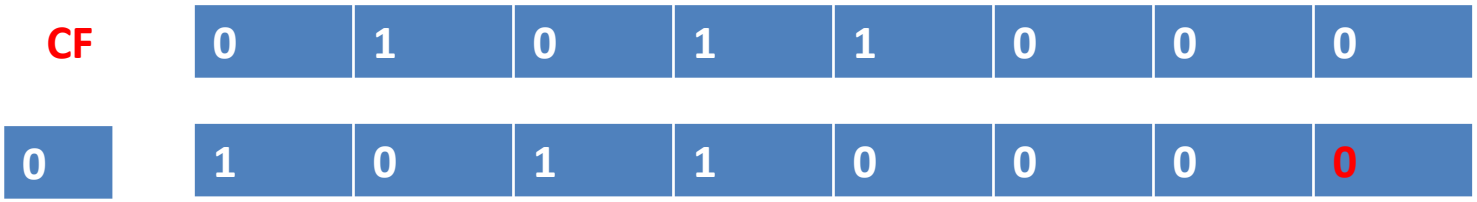
SHR: (Shift Right): SHR AL,1



OF=0 (7. bit değişmedi)

SHL: (Shift Left): SHL AL,1

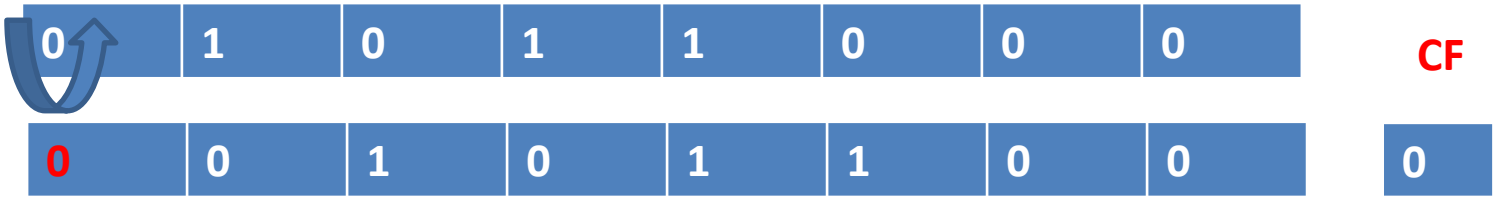
OF=1 (7. bit değişti)



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

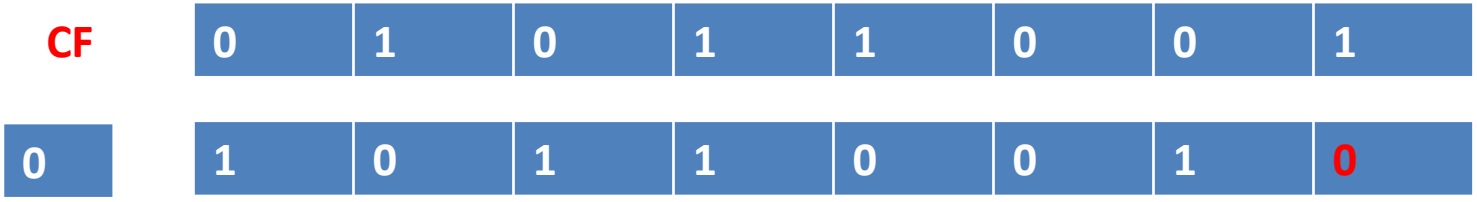
- **SAR (Shift Arithmetic Right): SAR AL,1**



OF=0 (7. bit hiç değişmez)

- **SAL (Shift Arithmetic Left):SAL AL,1**

(SHL komutuyla aynı işlemi yapıyor)

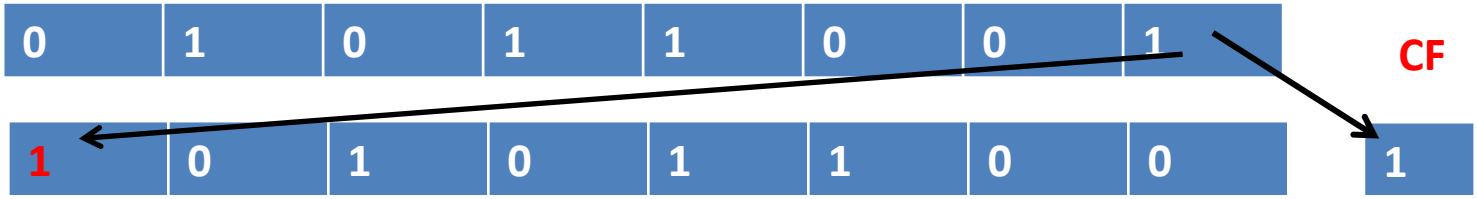


OF=1 (7. bit değişti)

8086 16-Bit Mikroişlemci

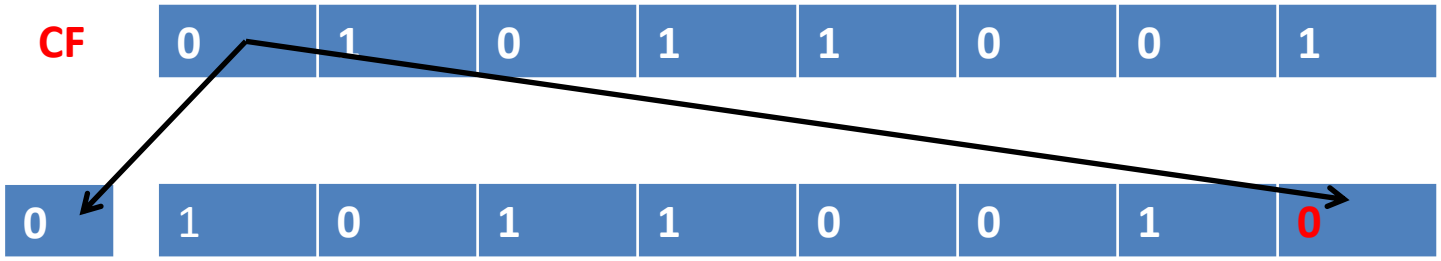
EMU 8086-MICROPROCESSOR EMULATOR

- **ROR (Rotate Right): ROR AL,1**



OF=1 (7. bit değişti)

- **ROL (Rotate Left): ROL AL,1**

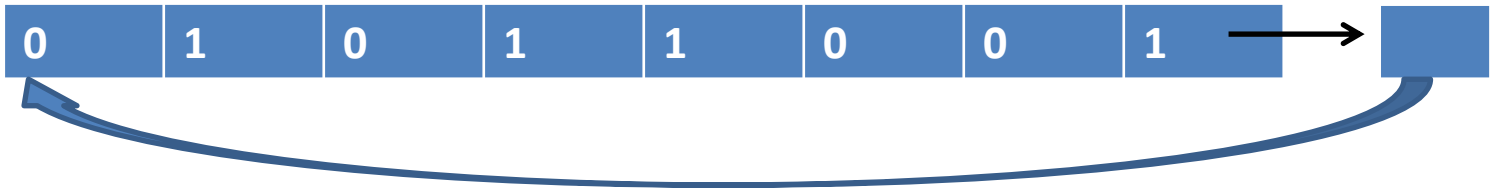


OF=1 (7. bit değişti)

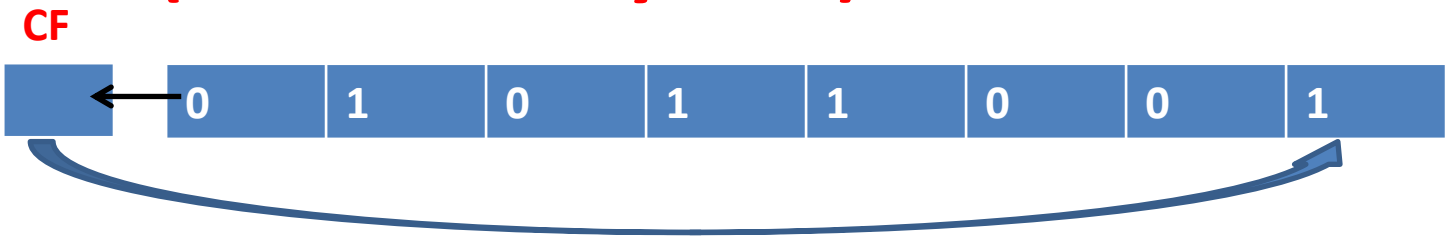
8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **RCR(Rotate Carry Right):** CF içindeki ilk değer 7. bit olarak başa döner



- **RCL(Rotate Carry Left):** CF içindeki ilk değer 0. bit olarak yazılır



RCR ve RCL de CF içindeki değer kullanılır daha sonra kaydırma yapılır

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

DS VE ES KULLANIMI

- COM dosyalarında tek bir bölüt vardır ve bu bölüt 64KB ile sınırlıdır
- Segment registerları başlangıçta aynı adresi işaret ederler

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

SREG: OFFSET

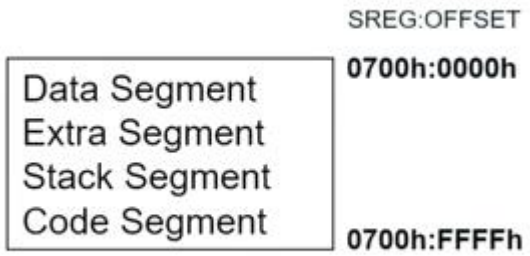
DS : SI

ES : DI

SS : SP

CS : IP

```
org 100h

LEA SI, sayi1
LEA DI,sayi2

MOV AL,[DI]
MOV BL,DS:[DI]
MOV CL,ES:[DI]

ret

sayi1 db 5
sayi2 db 4
sayi3 db 3
```



SREG: OFFSET

DS : SI

ES : DI

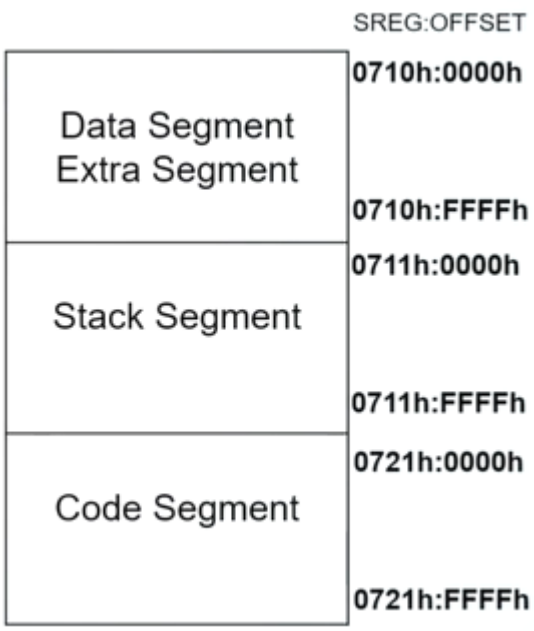
SS : SP

CS : IP

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- Exe dosyasında tüm segmentler ayrılmıştır.
- Her segment 64 KB yer kaplar




```
; multi-segment executable file template.

data segment
    ; veri tanimlamalar data segmentte yapilir

    sayi1 db 5
    sayi2 db 4
    sayi3 db 3

ends



stack segment
    dw   128  dup(0)
ends



code segment

start:
; set segment registers:
    mov ax, data
    mov ds, ax      ; DS ve ES ayni yeri gosterecek
    mov es, ax



    LEA SI, sayi1
    LEA DI,sayi2

    MOV AL,[DI]
    MOV BL,DS:[DI]
    MOV CL,ES:[DI]


    mov ax, 4c00h ; exit to operating system.
    int 21h       ; exit to operating system.
ends

end start ; set entry point and stop the assembler.
```

SREG: OFFSET

DS : SI

ES : DI

SS : SP

CS : IP

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
data segment

   sayi1 db 5
   sayi2 db 4
   sayi3 db 3

ends


extra segment
   sayi4 db 6
   sayi5 db 7

ends


stack segment
   dw   128  dup(0)
ends


code segment
start:
; set segment registers:
   mov ax, data
   mov ds, ax
   mov ax, extra
   mov es, ax


   LEA SI,sayi1
   LEA DI,sayi2

   MOV AL,[DI]
   MOV CL,ES:[DI]


 mov ax, 4c00h ; exit to operating system.
 int 21h      ; exit to operating system.
ends

end start ; set entry point and stop the assembler.
```

SREG: OFFSET
DS : SI
ES : DI
SS : SP
CS : IP

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **STRING (DİZİ) KOMUTLARI**

MOVSB: Operand almaz

DS:[SI] da bulunan byte olarak tanımlı veriyi ES:[DI] ile gösterilen yere kopyalar.

DS kaynak, ES hedef olur.

DF(direction flag) bayrağına göre SI ve DI güncellenir.
(DF=0 SI=SI+1 ve DI=DI+1, DF=1 SI=SI-1 ve DI=DI-1)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
data segment     ;DS:SI

   kaynak db 1,2,3,4


ends


extra segment   ; ES:DI

   hedef db 4 dup(0)

ends


stack segment
   dw   128  dup(0)
ends


code segment
start:
; set segment registers:
   mov ax, data
   mov ds, ax
   mov ax, extra
   mov es, ax


   cld   ; df=0
   mov cx,4
   lea SI,kaynak
   lea DI,hedef

   dongu:
   movsb  ; SI ve DI otomatik olarak artar
   loop dongu



 mov ax, 4c00h ; exit to operating system.
 int 21h       ; exit to operating system.
ends

end start ; set entry point and stop the assembler.
```

SREG: OFFSET
DS : SI
ES : DI
SS : SP
CS : IP

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

MOVSW: Operand almaz

DS:[SI] da bulunan word olarak tanımlı veriyi ES:[DI] ile gösterilen yere kopyalar.

DS kaynak, ES hedef olur.

DF(direction flag) bayrağına göre SI ve DI güncellenir.
(DF=0 SI=SI+2 ve DI=DI+2, DF=1 SI=SI-2 ve DI=DI-2)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

LODSB: Operand almaz

DS:[SI] da bulunan byte olarak tanımlı veriyi AL registerına kopyalar.

DS kaynak, AL hedef olur.

DF(direction flag) bayrağına göre SI güncellenir.

(DF=0 SI=SI+1, DF=1 SI=SI-1)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
data segment     ;DS:SI

   kaynak db 1,2,3,4


ends


extra segment   ; ES:DI



ends


stack segment
   dw   128  dup(0)
ends


code segment
start:
; set segment registers:
   mov ax, data
   mov ds, ax
   mov ax, extra
   mov es, ax


   cld   ; df=0
   mov cx,4
   lea SI,kaynak

   dongu:
   lodsb  ; SI otomatik olarak artar
   loop dongu



 mov ax, 4c00h ; exit to operating system.
 int 21h       ; exit to operating system.
ends

end start ; set entry point and stop the assembler.
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

LODSW: Operand almaz

DS:[SI] da bulunan word olarak tanımlı veriyi AX registerına kopyalar.

DS kaynak, AX hedef olur.

DF(direction flag) bayrağına göre SI güncellenir.

(DF=0 SI=SI+2, DF=1 SI=SI-2)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

STOSB: Operand almaz

AL içinde bulunan byte olarak tanımlı veriyi ES:[DI] ile gösterilen yere kopyalar.

AL kaynak, ES:[DI] hedef olur.

DF(direction flag) bayrağına göre DI güncellenir.

(DF=0 DI=DI+1, DF=1 DI=DI-1)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
data segment     ;DS:SI

ends


extra segment   ; ES:DI

    hedef db 4 dup(0)

ends


stack segment
   dw   128  dup(0)
ends


code segment
start:
; set segment registers:
   mov ax, data
   mov ds, ax
   mov ax, extra
   mov es, ax


   cld   ; df=0
   mov cx,4
   lea DI,hedef

   dongu:
   mov al,cl
   stosb  ; DI otomatik olarak artar
   loop dongu



 mov ax, 4c00h ; exit to operating system.
 int 21h       ; exit to operating system.
ends

end start ; set entry point and stop the assembler.
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

STOSW: Operand almaz

AX içinde bulunan word olarak tanımlı veriyi ES:[DI] ile gösterilen yere kopyalar.

AX kaynak, ES:[DI] hedef olur.

DF(direction flag) bayrağına göre DI güncellenir.

(DF=0 DI=DI+2, DF=1 DI=DI-2)

SREG: OFFSET
DS : SI
ES : DI
SS : SP
CS : IP

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

CMPSB: Operand almaz

DS:[SI] ile gösterilen yerde bulunan byte olarak tanımlı veriyi ES:[DI] ile gösterilen yerdeki veriyle karşılaştırır.

Temelde çıkarma işlemi yapar

DF(direction flag) bayrağına göre SI ve DI güncellenir.

(DF=0 SI=SI+1 ve DI=DI+1, DF=1 SI=SI-1 ve DI=DI-1)

SREG: OFFSET
DS : SI
ES : DI
SS : SP
CS : IP

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

CMPSW: Operand almaz

DS:[SI] ile gösterilen yerde bulunan word olarak tanımlı veriyi ES:[DI] ile gösterilen yerdeki veriyle karşılaştırır.

Temelde çıkarma işlemi yapar

DF(direction flag) bayrağına göre SI ve DI güncellenir.

(DF=0 SI=SI+2 ve DI=DI+2, DF=1 SI=SI-2 ve DI=DI-2)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

SCASB: Operand almaz

AL içinde bulunan byte olarak tanımlı veriyi ES:[DI] ile gösterilen yerdeki veriyle karşılaştırır.

Temelde çıkarma işlemi yapar

DF(direction flag) bayrağına göre DI güncellenir.

(DF=0 DI=DI+1, DF=1 DI=DI-1)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

SCASW: Operand almaz

AX içinde bulunan word olarak tanımlı veriyi ES:[DI] ile gösterilen yerdeki veriyle karşılaştırır.

Temelde çıkarma işlemi yapar

DF(direction flag) bayrağına göre DI güncellenir.

(DF=0 DI=DI+2, DF=1 DI=DI-2)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

TEKRARLAMA KOMUTLARI

- REP operand1 (operand1: MOVSB,MOVSW,LODSB,LODSW,STOSB,STOSWkomutlarının CX=0 olana kadar tekrarlar)

- REPE operand1 (operand1: CMPSB, CMPSW, SCASB, SCASW komutlarının ZF=1 olduğu sürece ve CX=0 olana kadar tekrarlar)

- REPZ operand1 (operand1: CMPSB, CMPSW, SCASB, SCASW komutlarının ZF=1 olduğu sürece ve CX=0 olana kadar tekrarlar)

(REPE-REPZ Kısaca eşitlik olduğu sürece tekrar etmesi gereken durumlarda)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

TEKRARLAMA KOMUTLARI

- REPNE operand1 (operand1: CMPSB, CMPSW, SCASB, SCASW komutlarının ZF=0 olduğu sürece ve CX=0 olana kadar tekrarlar)

- REPNZ operand1 (operand1: CMPSB, CMPSW, SCASB, SCASW komutlarının ZF=0 olduğu sürece ve CX=0 olana kadar tekrarlar)

(REPNE-REPNZ kısaca eşitlik olmadığı sürece tekrar etmesi gereken durumlarda)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
data segment     ;DS:SI

   kaynak db 1,2,3,4


ends


extra segment   ; ES:DI

   hedef db 4 dup(0)

ends


stack segment
   dw   128  dup(0)
ends


code segment
start:
; set segment registers:
   mov ax, data
   mov ds, ax
   mov ax, extra
   mov es, ax


   cld   ; df=0
   mov cx,4
   rep movsb



 mov ax, 4c00h ; exit to operating system.
 int 21h      ; exit to operating system.
ends

end start ; set entry point and stop the assembler.
```

KAYNAKLAR

8086 assembler tutorials/ part 7: program flow control/ Short Conditional Jumps