



# MİKROİŞLEMCİ SİSTEMLERİ

Dr. Öğr. Üyesi Meltem KURT PEHLIVANOĞLU

W-5

# 8086 Mikroişlemci

Segment ve adres register çiftleri:

<b>CS</b>	<b>IP</b>
<b>SS</b>	<b>SP</b>
	<b>BP</b>
<b>DS</b>	<b>BX</b>
	<b>SI</b>
	<b>DI</b>
<b>ES</b>	<b>DI</b>

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **MUL OPERAND1** (işaretsiz sayılarda çarpma)

$AX = AL * operand1$  (8-bit iki operandın çarpımı: sonuc AX te tutulur, operand1 register veya ram hücresi olabilir)

$(DX\ AX) = AX * operand1$  (16-bit iki operandın çarpımı: DX ve AX te tutulur 16-biti aşma durumuna karşı DX te kullanılır, operand1 register veya ram hücresi olabilir)

- Çarpma sonucunun yüksek değerli (AH=0 (8-bit için), DX=0 (16-bit için)) kısmı 0 olduğu zaman CF=OF=0 olur

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

org 100h

mov al,6

mov bl,2 ; ikinci carpilacak degeri baska registra atamalisiniz

mul bl ; carpim sonucu AX te AX=000C

mov al,255

mov bl,2

mul bl ; AX=01FE olur unsigned=510

mov ax,2

mul [sayi1] ; sonucu DX AX te gorurum DX=0000 AX=0200 (unsigned= 512)

mov ax,65535 ; yani max 16-bitle ifade edebilecegim deger FFFF

mul sayi1 ; 65535\*256=16776960 = 00FFFF00h DX=00FF AX=FF00

ret

sayi1 dw 256

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **IMUL OPERAND1** işaretli sayılarda çarpma

$AX = AL * \text{operand1}$  (8-bit iki operandın çarpımı: sonuc AX te tutulur, operand1 register veya ram hücresi olabilir)

$(DX\ AX) = AX * \text{operand1}$  (16-bit iki operandın çarpımı: DX ve AX te tutulur 16-biti aşma durumuna karşı DX te kullanılır, operand1 register veya ram hücresi olabilir)

- Çarpma sonucu 8-bit (-128,+127) 16-bit (-32768, +32767) olduğu zaman CF=OF=0 olur

sayi dw 1000000000000000b (-32768)

sayi2 dw 0111111111111111b (+32767)

sayi3 db 10000000b (-128)

sayi4 db 01111111b (+127)

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h
```

```
mov ax,-2
```

```
mov bx,20
```

```
imul bx ; DX=FFFF AX=FFD8 -40
```

```
ret
```

```
sayi1 dw 256
```

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **DIV OPERAND1** işaretsiz sayılarda bölme

$AL = AX / \text{operand1}$  (8-bit sonuc: AH=kalan  
operand1 register veya ram hücresi olabilir)

$AX = (DX\ AX) / \text{operand1}$  (16-bit sonuc: DX=kalan,  
operand1 register veya ram hücresi olabilir)

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **IDIV OPERAND1** işaretli sayılarda bölme

$AL = AX / \text{operand1}$  (8-bit sonuc: AH=kalan  
operand1 register veya ram hücresi olabilir)

$AX = (DX\ AX) / \text{operand1}$  (16-bit sonuc: DX=kalan,  
operand1 register veya ram hücresi olabilir)



# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

org 100h

mov ax,115

div [sayi1] ; sonuc AL de kalan AH da tutulur AH=0 ise tam bolunuyor

mov dx,0001h

mov ax,1200h ;00011200h sayisi 70144 decimal

mov bx,5h ;  $70144/5 = 14028$  sayisi HEX olarak 36CC olur (AX=36CC) Kalan=4  
(DX=0004)

div bx

mov ax,-20

mov bl,10

idiv bl

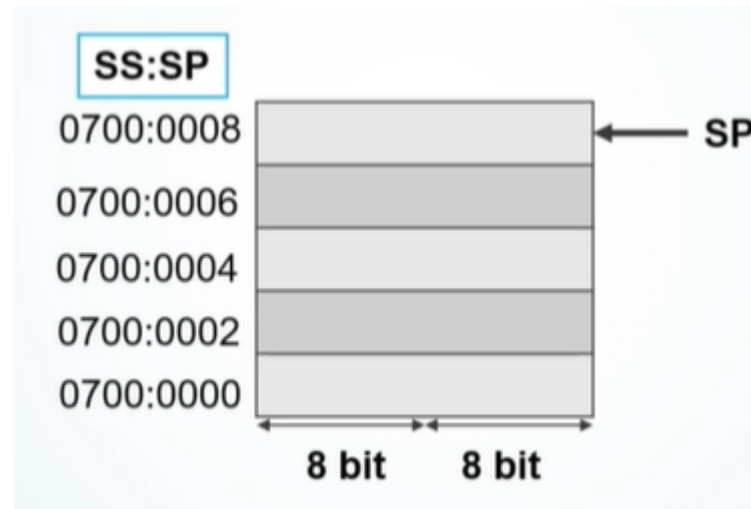
ret

sayi1 db 5

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- Stack (yığın): Geçici verileri tutmak için kullanılan bellek alanıdır.
- LIFO (Last In First Out) mantığı ile çalışır. Yani son giren ilk çıkar
- Normalde her RAM hücresi 8 bit (1 byte) yer kaplıyor ancak Stack içinde her eleman 16 bit (2 byte) olarak tutuluyor. Diğer bir ifadeyle ardışık 8 bitlik 2 RAM hücresi işgal eder.



# 8086 16-Bit Mikroişlemci

## **EMU 8086-MICROPROCESSOR EMULATOR**

- SP Stack için ayrılan alanın en başını işaret eder.
- Küçük değerlikli 8-bit önce olmak üzere Stack e yerleştirme yapılır
- PUSH Stack e yazma
- POP Stack ten bilgi alma

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **PUSH** operand1
  - $SP = SP - 2$
- **POP** operand1
  - $SP = SP + 2$

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **PUSHA** : operand almaz, tüm genel amaçlı (AX,BX,CX,DX,DI,SI,BP,SP) registerların değerlerini stack üzerine yükler. SP değeri komut kullanılmadan önceki haliyle yüklenir.
- Bu komuttan sonra herhangi bir stack işlemi yapılmamalıdır

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **POPA** : operand almaz, genel amaçlı (AX,BX,CX,DX,DI,SI,BP) registerların değerlerini stack den alıp yükler. **SP değeri geri yüklenmez**
- **SP değeri geri yüklenmiyor çünkü eğer geri yüklense SP kaldığı yerden farklı bir konuma geçer. SP değeri geri yüklenmediği için, SP ın kaldığı yer kaybolmamış olur.**
- Bu komuttan sonra herhangi bir stack işlemi yapılmamalıdır

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

org 100h

MOV AX,2323H

;PUSH AX

MOV BX,0BCD2H

MOV CX,4543H

MOV DX,1234H

MOV SI,1215H

MOV DI,6463H

PUSHA

;MOV AX,4646H

POPA

POP AX ; **burda Stack islemi yapildigi icin SP konumunun degistigi gozlemlenebilir**

ret

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

The screenshot displays the EMU 8086 Microprocessor Emulator interface, which is divided into several panes:

- Registers Pane:** Shows the current values of the 8086 registers. The AX register contains 0000, BX contains 0000, CX contains 0000, DX contains 0000, SI contains 1215, DI contains 6463, and the other registers (CS, IP, SS, SP, BP, DS, ES) contain 0700.
- Memory Pane:** Displays the memory contents starting from address 0700. The memory is organized into two columns, each showing addresses, hex values, and ASCII characters. The first column shows addresses from 0700 to 07130, and the second column shows addresses from 0700 to 07130. The memory is currently empty, showing NULL values.
- Assembly Code Pane:** Shows the assembly code being executed. The code starts with a comment: "; You may customize this and other start-up templates. The location of this template is c:\emu8086\i". The code then defines the origin as 100h and contains the following instructions:

```
org 100h
MOV AX, 2323H
; PUSH AX
MOV BX, 0BCD2H
MOV CX, 4543H
MOV DX, 1234H
MOV SI, 1215H
MOV DI, 6463H
PUSH A
POP AX
RET
```
- Stack Pane:** Shows the stack memory. The stack is currently empty, showing NULL values.



# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

The screenshot displays the EMU 8086 Microprocessor Emulator interface, which is divided into several windows:

- Registers Window:** Shows the current state of the 8086 registers. The CS register is highlighted at 0700, and the IP register is at 0114. The stack pointer (SP) is at FFFE.
- Memory Window:** Displays the memory contents at the current instruction address (0700:0114). The instruction at 07114 is highlighted: `MOV AX, 02323h`.
- Assembly Window:** Shows the assembly code being executed. The instruction `MOV AX, 02323h` is highlighted in yellow.
- Stack Window:** Shows the stack contents, with the top of the stack at 0700:FFFF.

The main window also includes a menu bar (file, math, debug, view, external, virtual devices, virtual drive, help) and a toolbar with buttons for Load, reload, step back, single step, run, and step delay ms: 0.

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

The screenshot displays the EMU 8086 Microprocessor Emulator interface, which is divided into several panes:

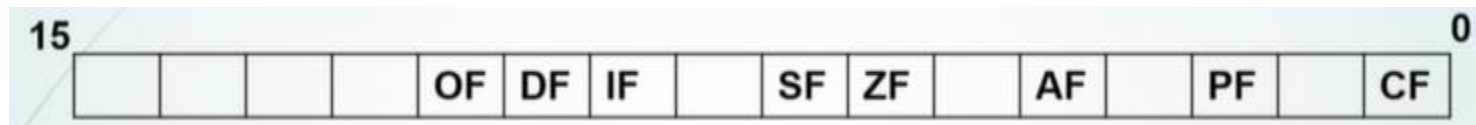
- Registers Pane:** Shows the current values of the 8086 registers. The IP register is highlighted with a value of 0115.
- Memory Pane:** Displays the memory contents at address 0700:0115. The memory is filled with a sequence of instructions, including MOV, PUSH, POP, and ADD.
- Assembly Code Pane:** Shows the original source code of the program. The instruction `ret` is highlighted in yellow.
- Stack Pane:** Shows the stack contents, with the top of the stack at address 0700:0000 containing the value 20CD.

The interface also includes a menu bar (file, math, debug, view, external, virtual devices, virtual drive, help) and a toolbar with buttons for Load, reload, step back, single step, and run. A status bar at the bottom shows the current step delay in milliseconds (0).

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

### 8086 16-bit Flag Register



Overflow Flag (OF) – Taşma Bayrağı

Direction Flag (DF) – Yön Bayrağı

Interrupt Enable Flag (IF) – Kesme Aktif Bayrağı

Sign Flag (SF) – İşaret Bayrağı

Zero Flag (ZF) – Sıfır Bayrağı

Auxiliary Carry Flag (AF) – Yardımcı Elde Bayrağı

Parity Flag (PF) – Benzerlik(Eşlik) Bayrağı

Carry Flag (CF) – Elde Bayrağı

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- CF(carry flag): Elde varsa 1 olur.
- ZF(zero flag): Herhangi bir işlem sonucunda 0 elde ediliyorsa ZF 1 olur
- SF(sign flag): ALU tarafından gerçekleştirilen bir işlemin sonucu eğer negatif çıkıyorsa SF 1 olur
- OF(overflow flag): İşaretli sayılarda işlem sonucu işaretli sayı aralığını aşıyorsa taşma bayrağı 1 olur (8-bitlik işaretli sayılar için en küçük değer -128, en büyük değer +127)
- PF(parity flag): İşlem sonucunda bulunan '1' bitlerinin sayısı çift ise PF 1 olur. **Sonuç 16-bit olsa bile düşük değerlikli 8-bit ele alınır.**
- AF(auxiliary flag): İşaretsiz sayılarda yapılan işlemlerdeki düşük değerlikli 4 bitte taşma meydana gelirse AF 1 olur.
- DF(direction flag): Diziler gibi ardışık verilerde özellikle string işlemlerinde kullanılan komutların ileri yönlü mü yoksa geri yönlü mü çalışacağını belirlemek için kullanılır. DF=0 iken ileri yönlü (düşük adresten yüksek adrese) işlem yapılır, DF=1 iken geri yönlü (yüksek adresten düşük adrese). Varsayılan 0 değeridir.
- IF (interrupt flag): Varsayılan olarak aktif bu sayede kesmelere izin veriyor. Örneğin klavyeden değer okuma, ekrana metin yazdırma vb.

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

### Bayrak Değişim Komutları:

- **STC(Set Carry flag):** Carry flag aktif duruma=1 getirir.
- **CLC(Clear Carry flag):** Carry flag pasif duruma=0 getirir.
- **CMC(Complement Carry flag):** Carry flag aktifse pasif, pasifse aktif yapar.
  
- **STD(Set Direction flag):** Direction flag aktif duruma=1 getirir
- **CLD(Clear direction flag):** Direction flag pasif duruma=0 getirir

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **STI (Set Interrupt enable flag):** Interrupt flag aktif duruma=1 getirir (işlemci donanımsal kesmelere izin verilir, varsayılan 1 gelir)
- **CLI (Clear Interrupt enable flag):** Interrupt flag pasif duruma=0 getirir
- **LAHF(Load AH from 8 low bits of Flags register):** Flag registerın düşük değerli 8 bitini AH kaydedicisine aktarır. Bayrak değerleri değişmez. 1,3,5. bitler rezerve edilmiş

AH bit:

SF	ZF	0	AF	0	PF	1	CF
7	6	5	4	3	2	1	0

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **SAHF(Store AH register into low 8 bits of Flags register):** AH içinde bulunan değer flag register içine yüklenir

SF	ZF	0	AF	0	PF	1	CF
7	6	5	4	3	2	1	0

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **PUSHF** : Flag register değerlerini stack üzerine yazar.
- operand almaz,
- SP değeri iki azalır  $SP=SP-2$
- Flag register 16-bittir



# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **POPF** : Flag register değerini stack üzerinden okur ve yazar.
- operand almaz,
- SP değeri iki artar  $SP=SP+2$
- Flag register 16-bittir

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

### MANTIKSAL İŞLEMLER

- **AND** operand1, operand2 : mantıksal ve işlemi yapar, **sonuc operand1 de tutulur**

CF=0, OF=0 olur ZF, SF, PF işlem durumuna göre değişir

```
org 100h
```

```
; b karakterini B ye donusturmek b=98 B=66
```

```
; 1. durum: -32 = 11100000 ile toplanabilir (98-32=66)
```

```
; 2. durum: 32 ile maskeleme 00100000 diger bir ifadeyle (11011111) degeriyle and leriz
```

```
mov al, 'b'
```

```
mov bl,11100000b
```

```
add al,bl ; deger ilk operand olan AL de tutulur
```

```
;2. durum
```

```
mov cl, 'b'
```

```
and cl,11011111b ; deger ilk operand olan CL de tutulur
```

```
ret
```

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **OR** operand1, operand2 : mantıksal veya işlemi yapar, **sonuc operand1 de** tutulur

CF=0, OF=0 olur ZF, SF, PF işlem durumuna göre değişir

org 100h

; B karakterini b ye donusturmek B=66 b=98

; 1. durum: 32 = 00100000 ile toplanabilir (66+32=98)

; 2. durum: 32 00100000 OR lanir

mov al, 'B'

mov bl,00100000b

add al,bl ; deger ilk operand olan AL de tutulur

;2. durum

mov cl, 'B'

or cl,00100000b ; deger ilk operand olan CL de tutulur

ret

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **XOR** operand1, operand2 : mantıksal XOR işlemi yapar, **sonuc operand1 de** tutulur

CF=0, OF=0 olur ZF, SF, PF işlem durumuna göre değişir

```
org 100h
```

```
mov bl,00100000b
```

```
xor bl,10101001b
```

```
ret
```

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **NOT** operand1: operand1 deki değerin bitlerinin tersini alır

Hiçbir bayrağa etki etmez.

```
org 100h
```

```
mov bl,00100000b
```

```
not bl
```

```
ret
```

# 8086 16-Bit Mikroişlemci

## EMU 8086-MICROPROCESSOR EMULATOR

- **TEST** operand1,operand2: operand1,operand2 mantıksal AND işlemine tabi tutulur, ancak sonuç herhangi bir yerde saklanmaz, sadece bayraklar etkilenir

CF=0, OF=0 olur ZF, SF, PF işlem durumuna göre değişir

org 100h

mov al,10000101b

test al,00000001b ; 10000101 AND 00000001 = 00000001 ZF=0

test al,00000010b ; 10000101 AND 00000010 = 00000000 AL deki deger 0 oldugu icin ZF=1  
; islem sonucunda bulunan 1 bitlerinin sayisi cift oldugundan PF 1 olur

ret