



MİKROİŞLEMCİLER

Dr. Meltem KURT PEHLİVANOĞLU

W-11

MİKROİŞLEMCİLER

Digital Logic +

Digital Design +

Computer Architecture +

Microprocessors +

Microcontrollers +

Assembly Language

Programming(8086)

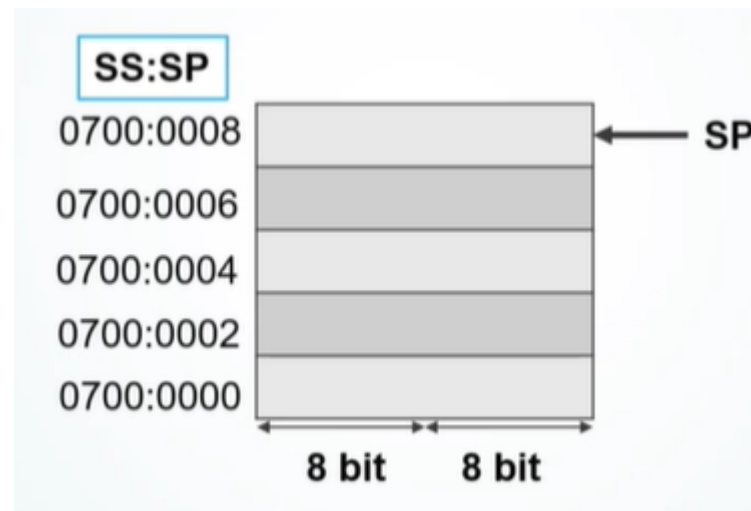
8086 16-Bit Mikroişlemci

- Segment ve adres register çiftleri:
- CS:IP
- SS:SP SS:BP
- DS:BX DS:SI
- DS:DI
- ES:DI

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 0000.
- 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 contents. 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 pointer (IP) address. The instruction at 07114 is highlighted, showing a MOV instruction.
- Assembly Window:** Shows the assembly code being executed. The instruction at 07114 is highlighted, showing a MOV instruction.
- Stack Window:** Shows the stack contents, with the top of the stack at 0700:FFFF.

The assembly code in the Assembly Window is as follows:

```
01 01 ; You may customize this and other start-up tem
02 02 ; The location of this template is c:\emu8086\i
03 03
04 04
05 05 org 100h
06 06
07 07 MOV AX,2323H
08 08 ;PUSH AX
09 09 MOV BX,0BCD2H
10 10 MOV CX,4543H
11 11 MOV DX,1234H
12 12 MOV SI,1215H
13 13 MOV DI,6463H
14 14
15 15 PUSHA
16 16
17 17 ;MOV AX,4646H
18 18 POPA
19 19 POP AX
20 20 ret
21 21
22 22
23 23
24 24
25 25
26 26
27 27
```

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 (Instruction Pointer) is highlighted at 0115.
- Memory Pane:** Displays the memory contents starting from address 0700:0115. The instruction at 0700:0115 is highlighted in blue.
- Assembly Pane:** Shows the assembly code being executed. The instruction `ret` at address 0700:0115 is highlighted in yellow.
- Stack Pane:** Shows the stack contents, with the top of the stack at 0700:0000.

The assembly code in the Assembly Pane is as follows:

```
01 ; You may customize this and other start-up templates
02 ; The location of this template is c:\emu8086\i
03
04
05 org 100h
06
07 MOV AX, 2323H
08 ; PUSH AX
09 MOV BX, 0BCD2H
10 MOV CX, 4543H
11 MOV DX, 1234H
12 MOV SI, 1215H
13 MOV DI, 6463H
14
15 PUSHA
16
17 ; MOV AX, 4646H
18 POPA
19 POP AX
20 ret
21
22
23
24
25
26
27
```

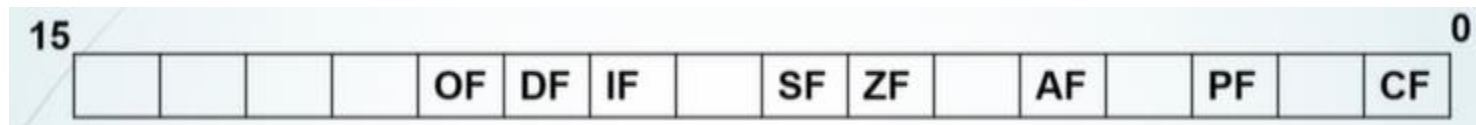
The memory contents in the Memory Pane are as follows:

Address	Value
0700:0115	B8 184 7
0700:0116	23 035 #
0700:0117	23 035 #
0700:0118	BB 187 7
0700:0119	D2 210 7
0700:011A	BC 188 7
0700:011B	B9 185 7
0700:011C	43 067 C
0700:011D	45 069 E
0700:011E	BA 186 7
0700:011F	34 052 4
0700:0120	12 018 4
0700:0121	BE 190 4
0700:0122	15 021 5
0700:0123	12 018 4
0700:0124	BF 191 4
0700:0125	63 099 c
0700:0126	64 100 d
0700:0127	60 096 a
0700:0128	61 097 a
0700:0129	58 088 X
0700:012A	C3 195 f
0700:012B	90 144 E
0700:012C	90 144 E
0700:012D	90 144 E
0700:012E	90 144 E
0700:012F	90 144 E
0700:0130	90 144 E
0700:0131	90 144 E
0700:0132	90 144 E
0700:0133	90 144 E
0700:0134	90 144 E
0700:0135	90 144 E
0700:0136	90 144 E
0700:0137	90 144 E
0700:0138	90 144 E
0700:0139	90 144 E
0700:013A	90 144 E
0700:013B	90 144 E
0700:013C	90 144 E
0700:013D	90 144 E
0700:013E	90 144 E
0700:013F	90 144 E
0700:0140	90 144 E
0700:0141	90 144 E
0700:0142	90 144 E
0700:0143	90 144 E
0700:0144	90 144 E
0700:0145	90 144 E
0700:0146	90 144 E
0700:0147	90 144 E
0700:0148	90 144 E
0700:0149	90 144 E
0700:014A	90 144 E
0700:014B	90 144 E
0700:014C	90 144 E
0700:014D	90 144 E
0700:014E	90 144 E
0700:014F	90 144 E
0700:0150	90 144 E
0700:0151	90 144 E
0700:0152	90 144 E
0700:0153	90 144 E
0700:0154	90 144 E
0700:0155	90 144 E
0700:0156	90 144 E
0700:0157	90 144 E
0700:0158	90 144 E
0700:0159	90 144 E
0700:015A	90 144 E
0700:015B	90 144 E
0700:015C	90 144 E
0700:015D	90 144 E
0700:015E	90 144 E
0700:015F	90 144 E
0700:0160	90 144 E
0700:0161	90 144 E
0700:0162	90 144 E
0700:0163	90 144 E
0700:0164	90 144 E
0700:0165	90 144 E
0700:0166	90 144 E
0700:0167	90 144 E
0700:0168	90 144 E
0700:0169	90 144 E
0700:016A	90 144 E
0700:016B	90 144 E
0700:016C	90 144 E
0700:016D	90 144 E
0700:016E	90 144 E
0700:016F	90 144 E
0700:0170	90 144 E
0700:0171	90 144 E
0700:0172	90 144 E
0700:0173	90 144 E
0700:0174	90 144 E
0700:0175	90 144 E
0700:0176	90 144 E
0700:0177	90 144 E
0700:0178	90 144 E
0700:0179	90 144 E
0700:017A	90 144 E
0700:017B	90 144 E
0700:017C	90 144 E
0700:017D	90 144 E
0700:017E	90 144 E
0700:017F	90 144 E
0700:0180	90 144 E
0700:0181	90 144 E
0700:0182	90 144 E
0700:0183	90 144 E
0700:0184	90 144 E
0700:0185	90 144 E
0700:0186	90 144 E
0700:0187	90 144 E
0700:0188	90 144 E
0700:0189	90 144 E
0700:018A	90 144 E
0700:018B	90 144 E
0700:018C	90 144 E
0700:018D	90 144 E
0700:018E	90 144 E
0700:018F	90 144 E
0700:0190	90 144 E
0700:0191	90 144 E
0700:0192	90 144 E
0700:0193	90 144 E
0700:0194	90 144 E
0700:0195	90 144 E
0700:0196	90 144 E
0700:0197	90 144 E
0700:0198	90 144 E
0700:0199	90 144 E
0700:019A	90 144 E
0700:019B	90 144 E
0700:019C	90 144 E
0700:019D	90 144 E
0700:019E	90 144 E
0700:019F	90 144 E
0700:01A0	90 144 E
0700:01A1	90 144 E
0700:01A2	90 144 E
0700:01A3	90 144 E
0700:01A4	90 144 E
0700:01A5	90 144 E
0700:01A6	90 144 E
0700:01A7	90 144 E
0700:01A8	90 144 E
0700:01A9	90 144 E
0700:01AA	90 144 E
0700:01AB	90 144 E
0700:01AC	90 144 E
0700:01AD	90 144 E
0700:01AE	90 144 E
0700:01AF	90 144 E
0700:01B0	90 144 E
0700:01B1	90 144 E
0700:01B2	90 144 E
0700:01B3	90 144 E
0700:01B4	90 144 E
0700:01B5	90 144 E
0700:01B6	90 144 E
0700:01B7	90 144 E
0700:01B8	90 144 E
0700:01B9	90 144 E
0700:01BA	90 144 E
0700:01BB	90 144 E
0700:01BC	90 144 E
0700:01BD	90 144 E
0700:01BE	90 144 E
0700:01BF	90 144 E
0700:01C0	90 144 E
0700:01C1	90 144 E
0700:01C2	90 144 E
0700:01C3	90 144 E
0700:01C4	90 144 E
0700:01C5	90 144 E
0700:01C6	90 144 E
0700:01C7	90 144 E
0700:01C8	90 144 E
0700:01C9	90 144 E
0700:01CA	90 144 E
0700:01CB	90 144 E
0700:01CC	90 144 E
0700:01CD	90 144 E
0700:01CE	90 144 E
0700:01CF	90 144 E
0700:01D0	90 144 E
0700:01D1	90 144 E
0700:01D2	90 144 E
0700:01D3	90 144 E
0700:01D4	90 144 E
0700:01D5	90 144 E
0700:01D6	90 144 E
0700:01D7	90 144 E
0700:01D8	90 144 E
0700:01D9	90 144 E
0700:01DA	90 144 E
0700:01DB	90 144 E
0700:01DC	90 144 E
0700:01DD	90 144 E
0700:01DE	90 144 E
0700:01DF	90 144 E
0700:01E0	90 144 E
0700:01E1	90 144 E
0700:01E2	90 144 E
0700:01E3	90 144 E
0700:01E4	90 144 E
0700:01E5	90 144 E
0700:01E6	90 144 E
0700:01E7	90 144 E
0700:01E8	90 144 E
0700:01E9	90 144 E
0700:01EA	90 144 E
0700:01EB	90 144 E
0700:01EC	90 144 E
0700:01ED	90 144 E
0700:01EE	90 144 E
0700:01EF	90 144 E
0700:01F0	90 144 E
0700:01F1	90 144 E
0700:01F2	90 144 E
0700:01F3	90 144 E
0700:01F4	90 144 E
0700:01F5	90 144 E
0700:01F6	90 144 E
0700:01F7	90 144 E
0700:01F8	90 144 E
0700:01F9	90 144 E
0700:01FA	90 144 E
0700:01FB	90 144 E
0700:01FC	90 144 E
0700:01FD	90 144 E
0700:01FE	90 144 E
0700:01FF	90 144 E

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

- **LAHF(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

ARİTMETİKSEL KOMUTLAR:

- **ADD** operand1,operand2

$\text{operand1} = \text{operand1} + \text{operand2}$

- **ADC** operand1,operand2

$\text{operand1} = \text{operand1} + \text{operand2} + \text{CF}$ (carry flag)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h
```

```
mov al,10  
add al,20 ; al=1E
```

```
mov al,255 ; maksimum alınacak deger  
add al,1 ; IF=1 zaten varsayılan geliyor  
; CF=1  $255+1=256$  isaretsiz sayılarda tasma oldu  
; ZF=1 oldu  $256-256=0$  islem sonucu 0 oldugundan ZF aktif oldu  
; PF=1 1 bitlerin sayısı çiftse aktifti, 8 bitte 0 tane 1 var çift olarak goruyor PF aktif oluyor  
; AF=1 sondaki 4 bitte tasma oldugundan AF aktif olur  $1111\ 1111 + 0001$  toplaması
```

```
mov al,255  
add al,5 ;  $255+5=260-256=4$  AL de 04 bulunur
```

```
mov al,-2  
add al,255 ;  $255-2=253$  ; AL de FD
```

```
mov ax,258  
add ax,5 ;  $258+5=263$  ; AX 01 07
```

```
add sayi1,3 ; sayi1=253  
mov bl,sayi1 ; BL=FD 253 HEX karsiligi
```

```
add [sayi1],5 ;  $253+5=258-256=2$   
mov bh,sayi1 ; BH=02 olur
```

```
ret
```

```
sayi1 db 250
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **SUB** operand1,operand2

operand1=operand1-operand2

- **SBB** operand1,operand2

operand1=operand1-operand2-CF

```
org 100h
```

```
mov al,1
```

```
sub al,3 ; -2 256-2=254 FE olur
```

```
ret
```

```
sayi1 db 250
```

```
org 100h
```

```
mov al,1
```

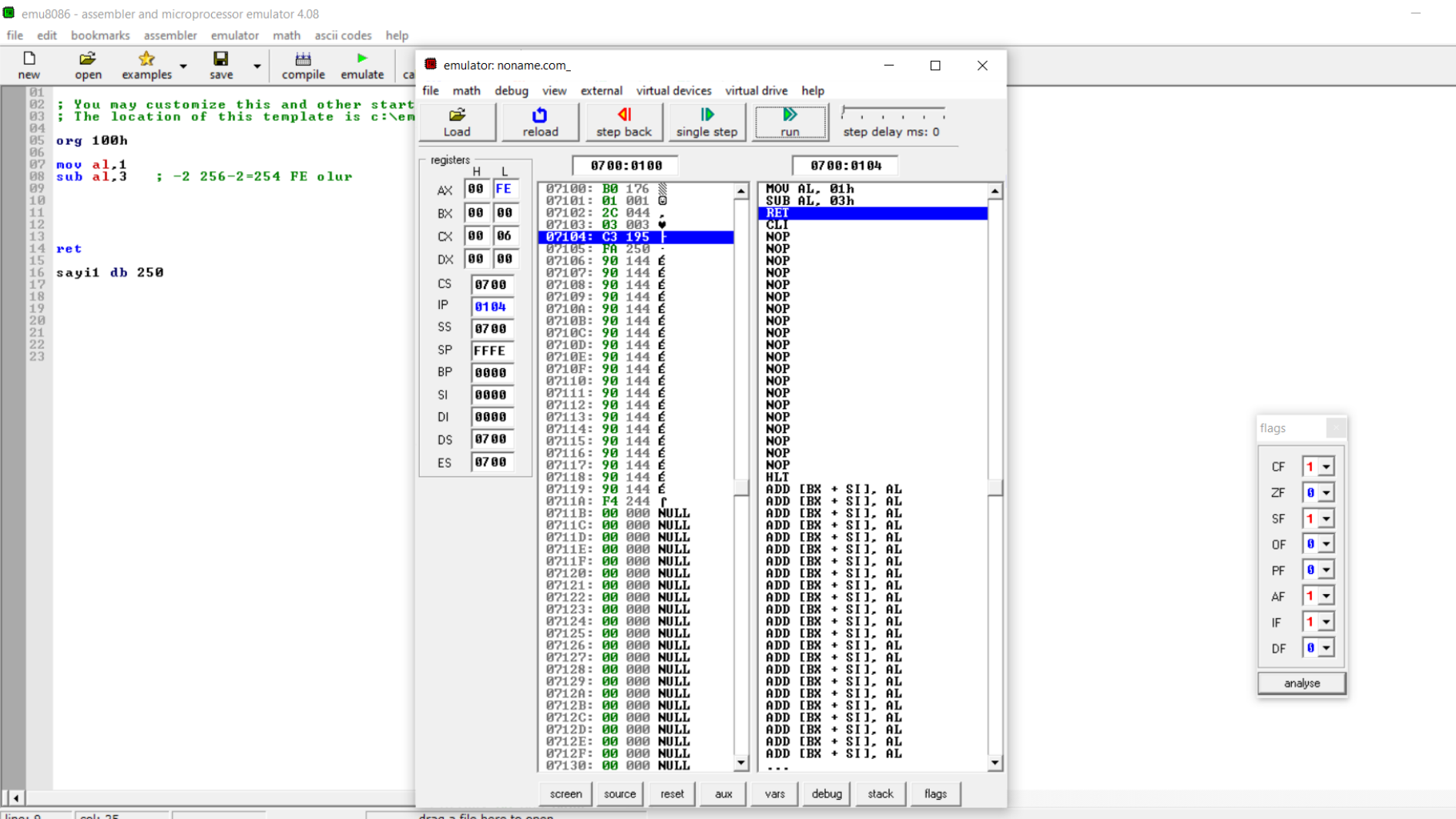
```
sub al,3 ; -2 256-2=254 FE olur  
; CF=1
```

```
sbb al,1 ; FE-1-1= FC
```

```
ret
```

```
sayi1 db 250
```

EMU 8086-MICROPROCESSOR EMULATOR



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

emu8086 - assembler and microprocessor emulator 4.08

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

- Registers:** A table showing the current values of 16-bit registers. The highlighted register is **AX**, which contains **00 FC**.
- Memory:** A list of memory addresses and their contents. The highlighted address is **07106: C3 195**.
- Source Code:** A window showing the assembly code being executed. The highlighted line is **ret**.
- Flags:** A window showing the status of various flags. The highlighted flag is **CF**, which is set to **0**.

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. The status bar at the bottom shows the current screen (screen), source, reset, aux, vars, debug, stack, and flags.

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **MUL** 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 sonucunun yüksek değeri (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** 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

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** işaretli sayılarda bölme

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

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

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **IDIV** 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 ;00011200 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

- **CBW:** 8-bitlik değeri 16-bitlik değere genişletir

Operand almaz, yüksek değerli 8. biti AH içine yayar

- **CWD:** 16-bitlik değeri 32-bitlik değere genişletir

Operand almaz, AX içindeki yüksek değerli 16. biti DX içine yayar

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

org 100h

mov al,-3 ;FD=11111101 8. biti 1 AH icindeki 8 biti de 1 yapar (1111 1111) o yuzden
AH FF olur
cbw

mov ax,0
mov ax,0FF5Fh ;FF5F: 11111111 01011111 16. bit 1 oldugu icin
cwd ;DX icini 1 ile doldurur DX=FF FF (1111 1111 1111 1111) olur

ret
sayi1 db 5

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

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU: İşaretsiz sayılar= 2,4,6,3 dizisindeki elemanların her birinin küpünü alan ve bu elemanları kup dizisine (word tipinde) yazdıran 8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

org 100h

```
MOV CX,4  
MOV SI,0  
MOV DI,0
```

dongu:

```
MOV AL,0  
MOV BL,0  
MOV AL, [sayilar+SI]  
MOV BL,AL ; BX= AX  
MUL BL ; AX= AX*AX  
MUL BL ; AX= AX*AX*AX  
MOV kup+DI,AX
```

```
INC SI  
ADD DI,2
```

LOOP dongu

ret

sayilar db 2,4,6,3
kup dw 4 dup(?)

org 100h

```
MOV CX,4  
MOV SI,0  
MOV DI,0
```

dongu:

```
MOV AL,0  
MOV BL,0  
MOV AL, [sayilar+SI]  
MOV BL,AL ; BX= AX  
IMUL BL ; AX= AX*AX  
IMUL BL ; AX= AX*AX*AX  
MOV kup+DI,AX
```

```
INC SI  
ADD DI,2
```

LOOP dongu

ret

sayilar db -2,4,6,3 ; isaretli eleman oldugunda imul kullanilir
kup dw 4 dup(?)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU: sayılar= 2,4,6,-2 dizisindeki elemanların her birinin indis değerlerini elemandan çıkararak sonucu fark dizisine yazan 8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h
```

```
MOV CX,4
```

```
MOV SI,0
```

```
MOV DL,0
```

```
dongu:
```

```
MOV AL,0
```

```
MOV BL,0
```

```
MOV AL, [sayilar+SI]
```

```
MOV BL,DL
```

```
SUB AL,BL ; sonuc AL de
```

```
MOV fark+SI,AL
```

```
INC SI
```

```
INC DL
```

```
LOOP dongu
```

```
ret
```

```
sayilar db 2,4,6,-2
```

```
fark db 4 dup(?)
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU: cumle='buyukharfyazilacak' dizisindeki tüm elemanları büyük harfe çevirip bunu buyukharf dizisine aktaran 8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h
```

```
MOV CX,18
```

```
MOV SI,0
```

```
dongu:
```

```
MOV AL, [cumle+SI]
```

```
AND AL,11011111b ;maskeleme islemi
```

```
MOV buyukharf+SI,AL
```

```
INC SI
```

```
LOOP dongu
```

```
ret
```

```
cumle db 'buyukharfyazilacak'
```

```
buyukharf db 18 dup(?)
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU: sayilar= 10,20,30,40 dizisindeki elemanların her birinin sadece son 4 bitindeki değerleri alıp bunları sayilar2 dizisine yazan 8086 Assembly kodunu yazınız

Örn: 20= 0001 0100 → 0100 (4) olacak

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h
```

```
MOV CX,4
```

```
MOV SI,0
```

```
dongu:
```

```
MOV AL, [sayilar+SI]
```

```
AND AL,00001111b
```

```
MOV sayilar2+SI,AL
```

```
INC SI
```

```
LOOP dongu
```

```
ret
```

```
sayilar db 10,20,30,40
```

```
sayilar2 db 4 dup(?)
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU: sayilar= 10,20,30,40 dizisindeki elemanları sayilar2= 3,4,7,6 dizisindeki sayılara bölen ve sadece kalanları kalan dizisine yazan 8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

org 100h

MOV CX,4

MOV SI,0

dongu:

MOV AX,0

MOV BX,0

MOV AL, [sayilar+SI]

MOV BL, [sayilar2+SI]

DIV BL ; kalan AH icinde

MOV [kalan+SI],AH

INC SI

LOOP dongu

ret

sayilar db 10,20,30,40

sayilar2 db 3,4,7,6

kalan db 4 dup(?)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

SORU: Stack üzerine **veri1=BCDEh**, **veri2=AB03h** değerlerini yazın. Daha sonra bu verileri **AX** registerına çekip veri1 ve veri2 nin **düşük değerlikli 8-bitini düşük** dizisine, **yüksek değerlikli 8-bitini yüksek** dizisine yazınız.

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

org 100h	org 100h	org 100h
PUSH 0BCDEh	MOV AX,veri1	MOV AX,veri1
PUSH 0AB03h	PUSH AX	PUSH AX
	MOV AX,veri2	MOV AX,veri2
	PUSH AX	PUSH AX
		; dongusuz yapmak istersem
MOV CX,2	MOV CX,2	
MOV SI,0	MOV SI,0	POP AX
	; dongulu kullanim	MOV buyuk[0],AH
dongu:	dongu:	MOV kucuk[0],AL
POP AX	POP AX	
MOV buyuk+SI,AH	MOV buyuk+SI,AH	
MOV kucuk+SI,AL	MOV kucuk+SI,AL	
INC SI	INC SI	POP AX
LOOP dongu	LOOP dongu	MOV buyuk[1],AH
		MOV kucuk[1],AL
ret	ret	ret
buyuk db 2 dup(?)	veri1 dw 0BCDEh	veri1 dw 0BCDEh
kucuk db 2 dup(?)	veri2 dw 0AB03h	veri2 dw 0AB03h
	buyuk db 2 dup(?)	buyuk db 2 dup(?)
	kucuk db 2 dup(?)	kucuk db 2 dup(?)

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
01 ; multi-segment executable file template.
02
03
04
05
06
07 data segment ; veri tanimlanalari buraya yapiyorum
08
09     veri1 dw 0BCDEh
10     veri2 dw 0AB03h
11     buyuk db 2 dup{?}
12     kucuk db 2 dup{?}
13
14 ends
15
16
17
18
19 stack segment
20     dw 128 dup{0} ; 128 tane bellek hucresi ayirdik icinde 16 bit veri tutabilir
21 ends
22
23
24
25
26 code segment
27 start:
28 ; set segment registers:
29     mov ax, data
30     mov ds, ax
31     mov es, ax
32
33 ; add your code here
34     MOV AX,veri1
35     PUSH AX
36     MOV AX,veri2
37     PUSH AX
38
39
40     MOV CX,2
41     MOV SI,0
42 ; dongulu kullanim
43 dongu:
44     POP AX
45     MOV buyuk+SI,AX
46     MOV kucuk+SI,AL
47     INC SI
48     LOOP dongu
49
50
51
52     mov ax, 4c00h ; isletim sistemin donus
53     int 21h
54 ends
55
56
57 end start ; set entry point and stop the assembler.
58
```