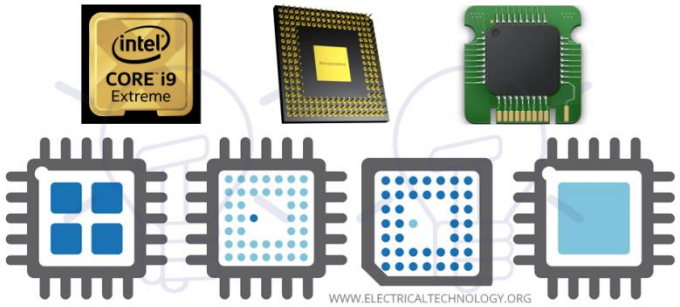


MİKROİŞLEMCİLER

Dr. Meltem KURT PEHLİVANOĞLU

W-12

MİKROİŞLEMCİLER

Digital Logic +

Digital Design +

Computer Architecture +

Microprocessors +

Microcontrollers +

Assembly Language Programming(8086)

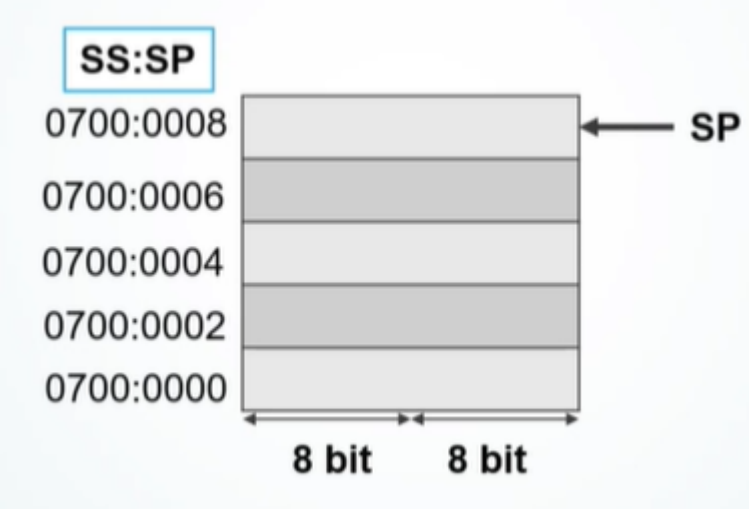
8086 16-Bit Mikroişlemci

- 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

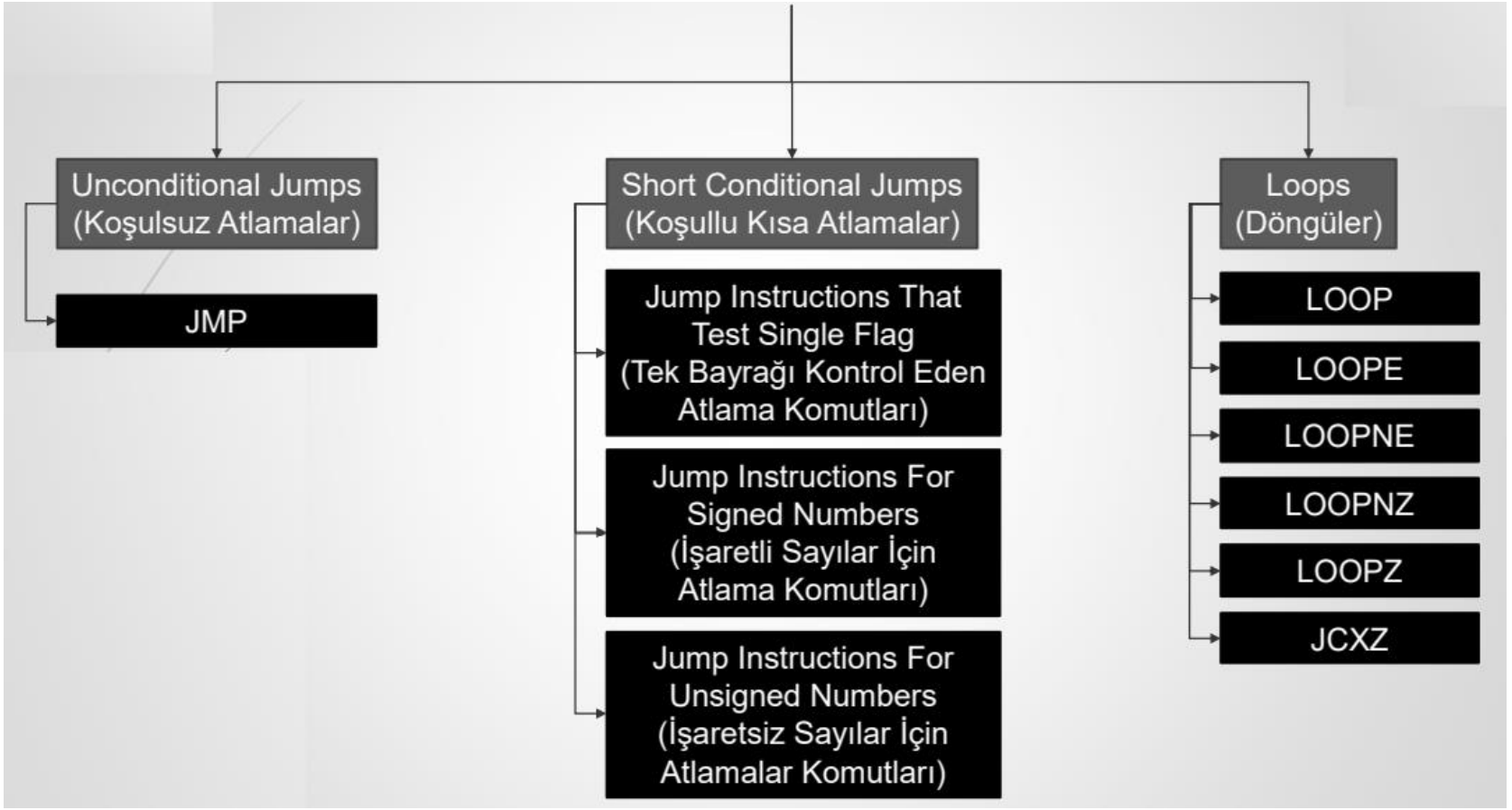
EMU 8086-MICROPROCESSOR EMULATOR

- 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

- **Program Akış Kontrol Türleri**



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **Koşulsuz Dallanma: JMP**

Code Segment üzerinde istenilen yere (64 KB alan içinde) atlayabilir.

```
org 100h
MOV AX,2025h

jmp label1

label2:
MOV AL,32h
ret

label1:
MOV AH,0F3h
jmp label2
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **CMP:** İki değer eşitse (SUB komutundaki işlemi yapar aslında) ZF=1 olur sonuç herhangi bir yerde tutulmaz. Şartlı dallanmada sık kullanılan bir komuttur.

CMP operand1,operand2

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- Dallanma sınırı: CS üzerinden 127 byte ileri, 128 byte geri gidebilir
- Bu sınırı aşmak için kullanılan koşulun tam tersi ile birlikte JMP komutu kullanılabilir. Emülatörün güncel sürümlerinde bu işlemi otomatik olarak yapmaktadır.
- Eğer değişim -127/+128 değerlerini aşmıyorsa kullanılan koşulun kendisi kullanılır.

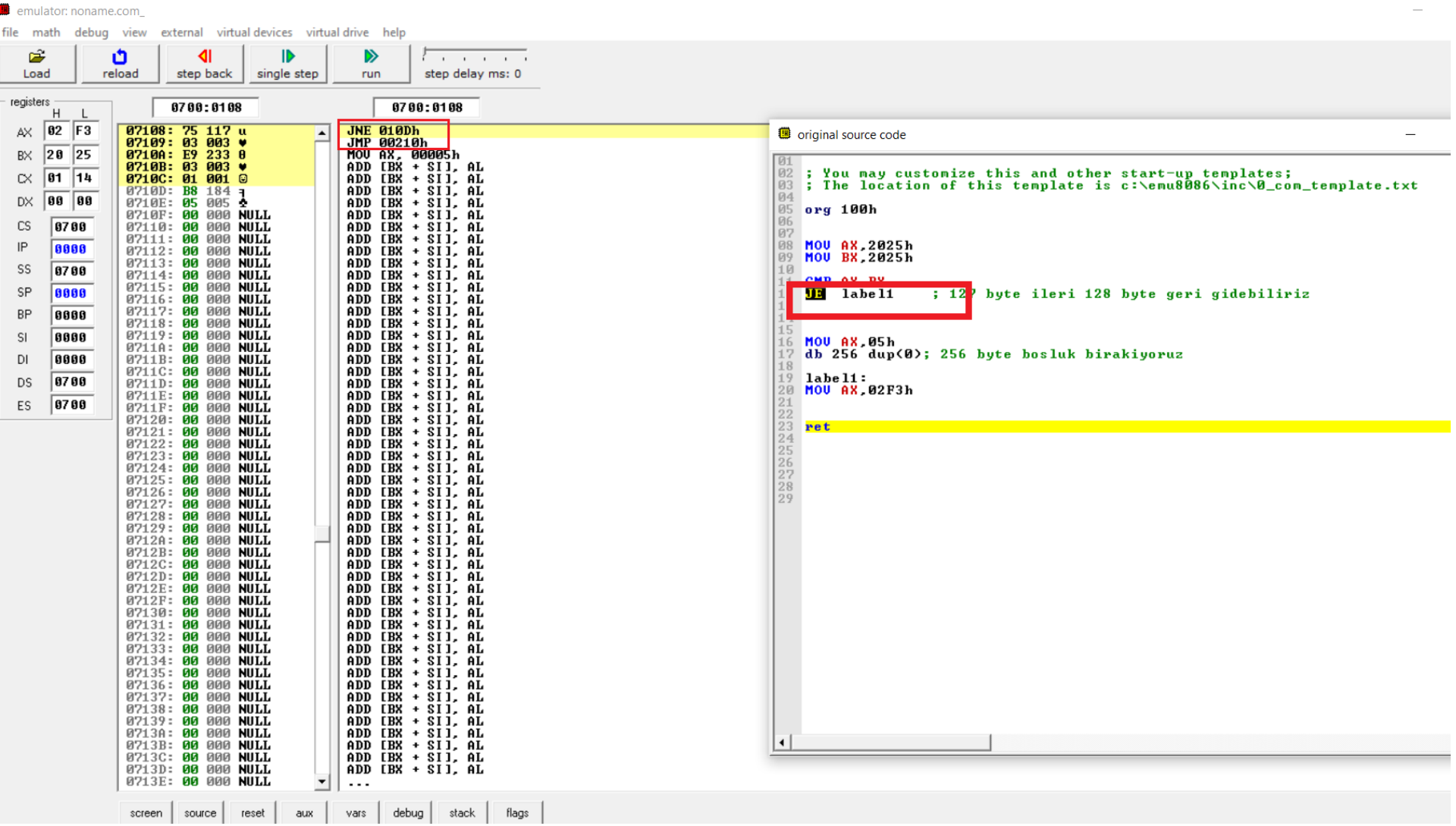
8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h

MOV AX,2025h
MOV BX,2025h

CMP AX,BX
JE  label1    ; 127 byte ileri 128 byte geri gidebiliriz

MOV AX,05h
db 256 dup(0); 256 byte bosluk birakiyoruz

label1:
MOV AX,02F3h

ret
```

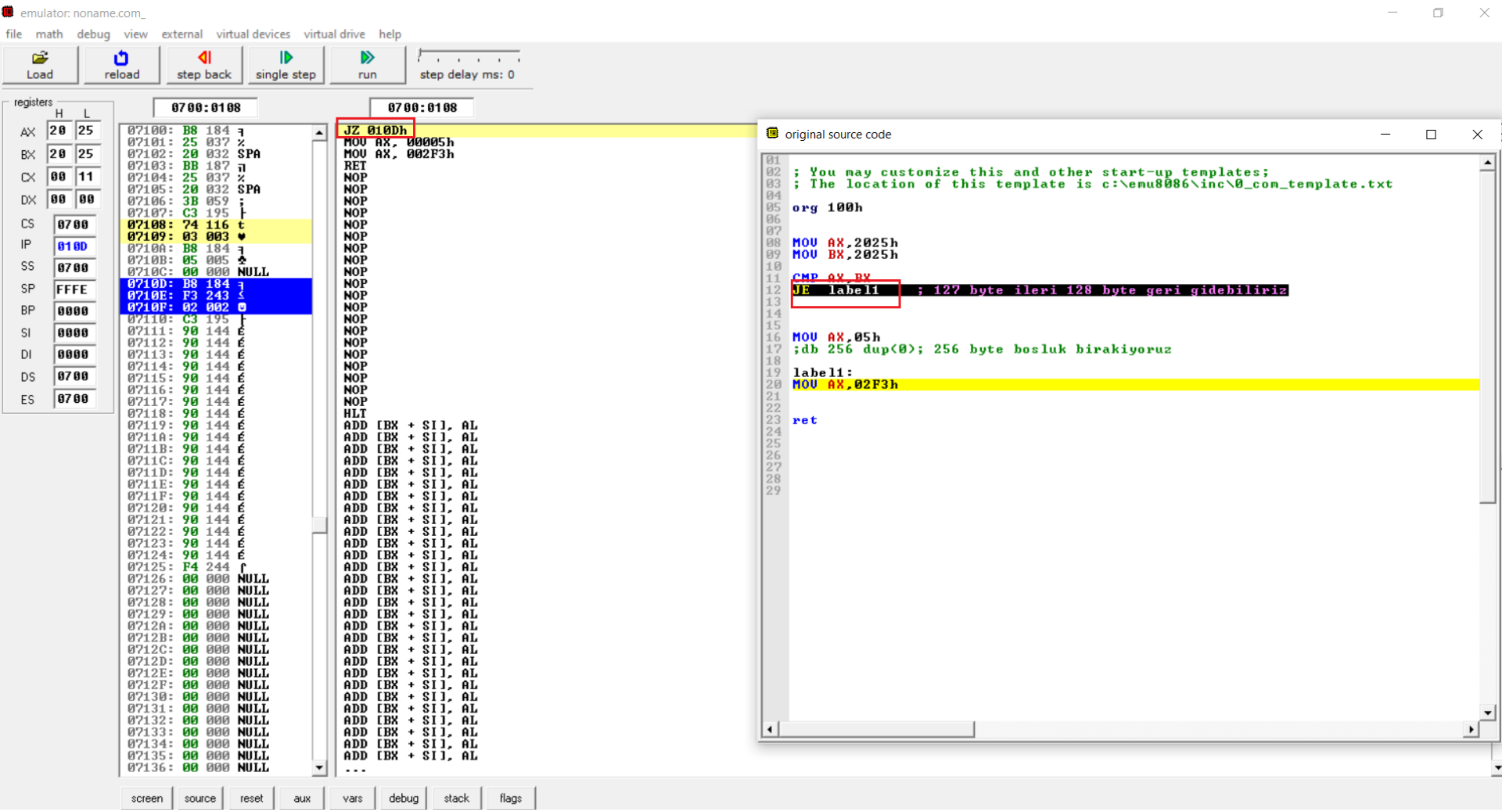
8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **Tek operasyon kodu:**

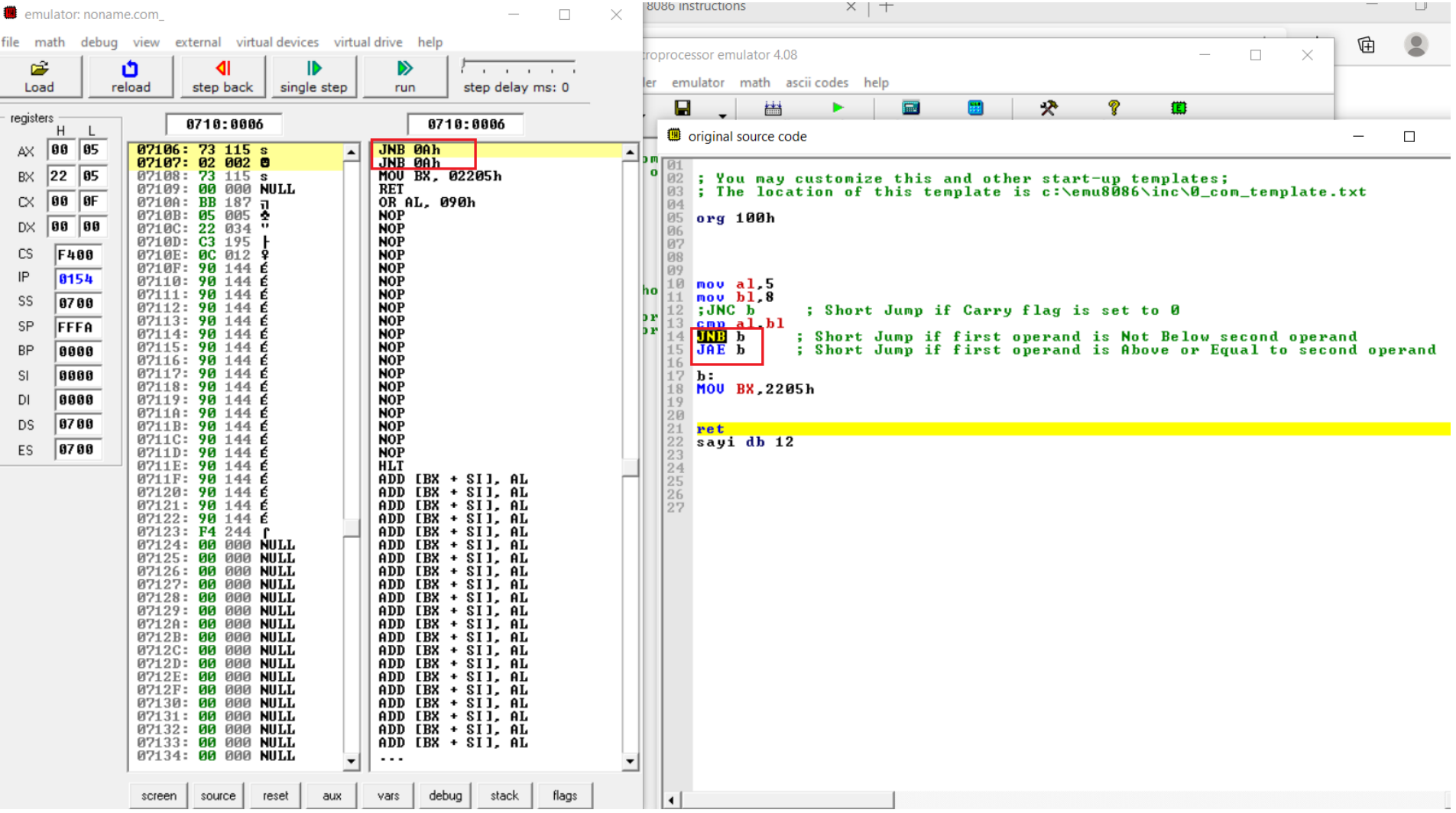
Bazı komutlar aynı işi yapmalarına rağmen farklı isimler ile adlandırılır. Programcının komutları daha kolay anlamasını sağlar. Ancak yapılan işlem aynı olduğu için tek bir opcode üretilir.

JNB, JAE, JNC komutları JNB komutuna dönüşür (temelde aynı işlemi yapıyorlar instruction isimleri farklı) ve aynı opcode a sahiptir 73 (http://www.mlsite.net/8086/#oper_b)

```
org 100h

mov al,5
mov bl,8
;JNC b     ; Short Jump if Carry flag is set to 0
cmp al,bl
JNB b     ; Short Jump if first operand is Not Below second operand
JAE b     ; Short Jump if first operand is Above or Equal to second operand

b:
MOV BX,2205h

ret
sayi db 12
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **Tek Bayrağı Kontrol Eden Dallanma Komutları**

Instruction	Description	Condition	Opposite Instruction
JZ , JE	Jump if Zero (Equal).	ZF = 1	JNZ, JNE
JC , JB, JNAE	Jump if Carry (Below, Not Above Equal).	CF = 1	JNC, JNB, JAE
JS	Jump if Sign.	SF = 1	JNS
JO	Jump if Overflow.	OF = 1	JNO
JPE, JP	Jump if Parity Even.	PF = 1	JPO
JNZ , JNE	Jump if Not Zero (Not Equal).	ZF = 0	JZ, JE
JNC , JNB, JAE	Jump if Not Carry (Not Below, Above Equal).	CF = 0	JC, JB, JNAE
JNS	Jump if Not Sign.	SF = 0	JS
JNO	Jump if Not Overflow.	OF = 0	JO
JPO, JNP	Jump if Parity Odd (No Parity).	PF = 0	JPE, JP

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h

mov al,5
mov bl,8

JNZ c    ; ZF=0 ise c ye dallanir, ZF degistirildiginde asagidaki komut isler
JMP b

c:
MOV BX,3131h

;JZ b

db 256 dup(0)


b:
MOV BX,2205h

ret
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **İşaretli Sayılar İçin Dallanma Komutları**

Instruction	Description	Condition	Opposite Instruction
JE , JZ	Jump if Equal (=). Jump if Zero.	ZF = 1	JNE, JNZ
JNE , JNZ	Jump if Not Equal (<>). Jump if Not Zero.	ZF = 0	JE, JZ
JG , JNLE	Jump if Greater (>). Jump if Not Less or Equal (not <=).	ZF = 0 and SF = OF	JNG, JLE
JL , JNGE	Jump if Less (<). Jump if Not Greater or Equal (not >=).	SF <> OF	JNL, JGE
JGE , JNL	Jump if Greater or Equal (>=). Jump if Not Less (not <).	SF = OF	JNGE, JL
JLE , JNG	Jump if Less or Equal (<=). Jump if Not Greater (not >).	ZF = 1 or SF <> OF	JNLE, JG

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h

MOV AL,-5
MOV AH,-20


CMP AL, AH   ; AL deki degerle AH daki degeri karsilastirip AL deki deger AH dan kucuk mu buyuk mu esit mi
JL kucuk
JG buyuk
JMP esit


kucuk:
MOV BL, 4
jmp bitir

buyuk:
MOV BL,5
jmp bitir

esit:
MOV BL,6
jmp bitir

bitir:
ret
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- **İşaretsiz Sayılar İçin Dallanma Komutları**

Instruction	Description	Condition	Opposite Instruction
JE , JZ	Jump if Equal (=). Jump if Zero.	ZF = 1	JNE, JNZ
JNE , JNZ	Jump if Not Equal (<>). Jump if Not Zero.	ZF = 0	JE, JZ
JA , JNBE	Jump if Above (>). Jump if Not Below or Equal (not <=).	CF = 0 and ZF = 0	JNA, JBE
JB , JNAE, JC	Jump if Below (<). Jump if Not Above or Equal (not >=). Jump if Carry.	CF = 1	JNB, JAE, JNC
JAE , JNB, JNC	Jump if Above or Equal (>=). Jump if Not Below (not <). Jump if Not Carry.	CF = 0	JNAE, JB
JBE , JNA	Jump if Below or Equal (<=). Jump if Not Above (not >).	CF = 1 or ZF = 1	JNBE, JA

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h

MOV AL,"C"
MOV AH,"c"


CMP AL, AH
JE esit
JMP esitdegil

geri:
MOV BL,30h
jmp bitir



esit:
MOV BL,AL
jmp bitir


esitdegil:
MOV BL,AH
jmp geri

bitir:
ret
```

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

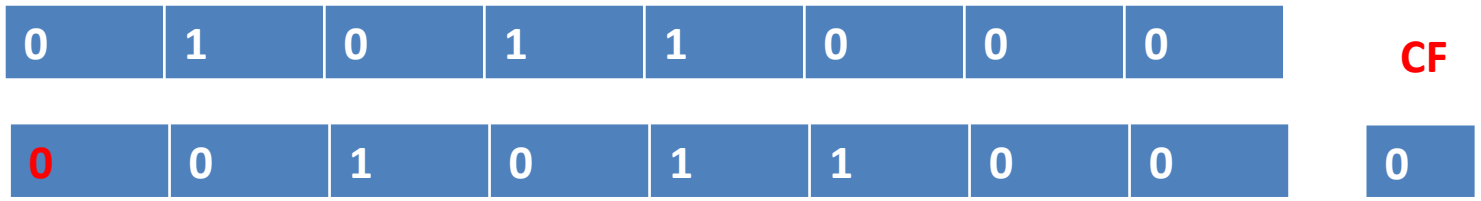
```
org 100h

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**

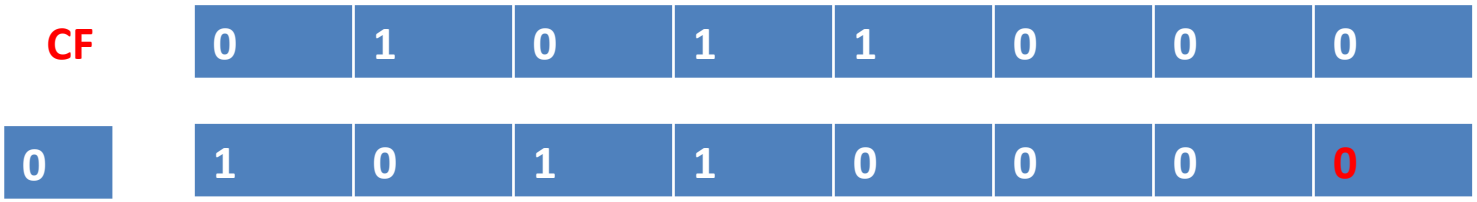
SHR: (Shift Right): SHR AL,1



OF=0 (7. bit değişmedi)

SHL: (Shift Left): SHL AL,1

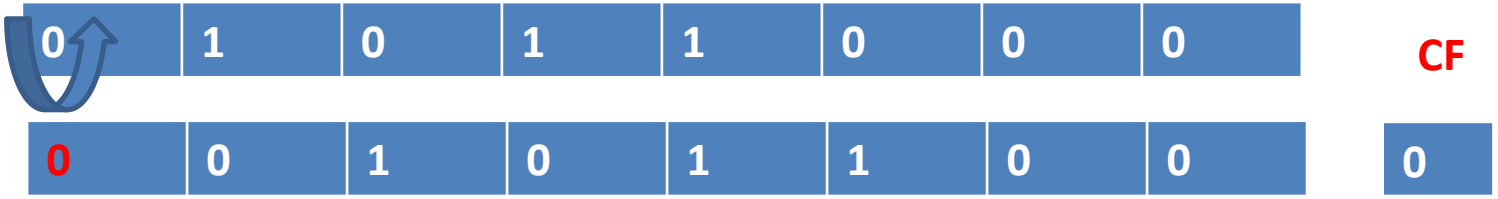
OF=1 (7. bit değişti)



8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

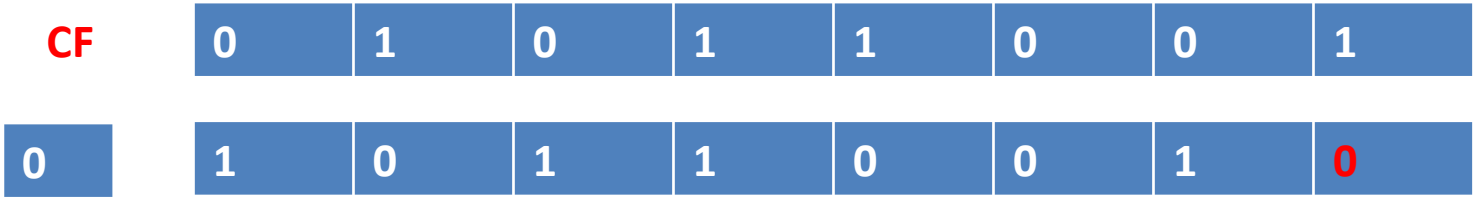
- **SAR (Shift Arithmetic Right): SAR AL,1**



OF=0 (7. bit hiç değişmez)

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

(SHL komutuyla aynı işlemi yapıyor)

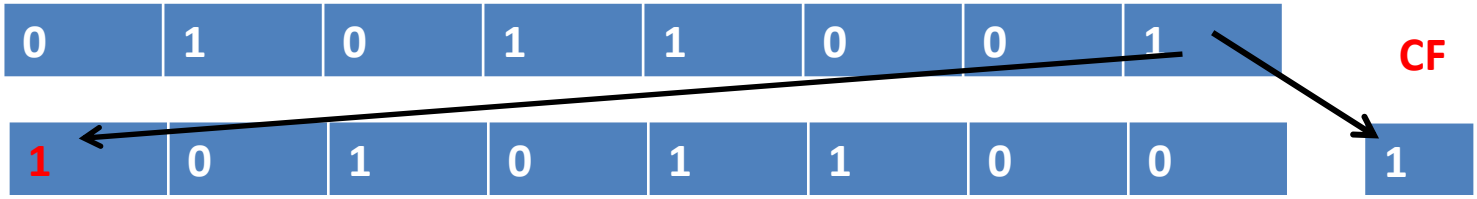


OF=1 (7. bit değişti)

8086 16-Bit Mikroişlemci

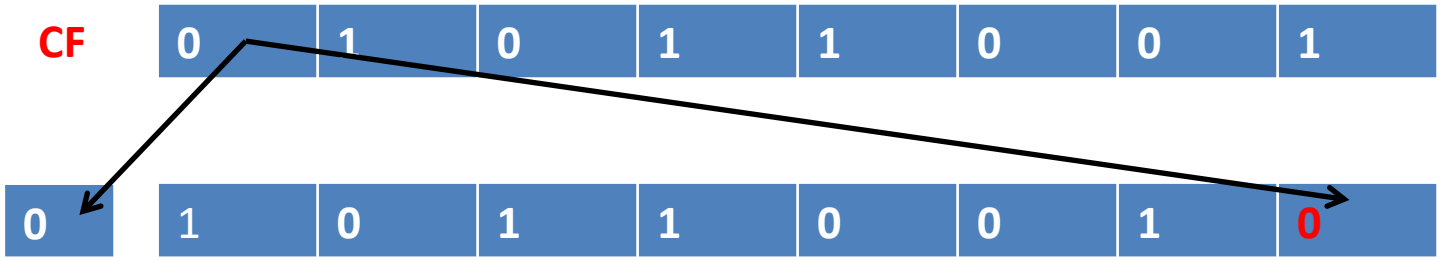
EMU 8086-MICROPROCESSOR EMULATOR

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



OF=1 (7. bit değişti)

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

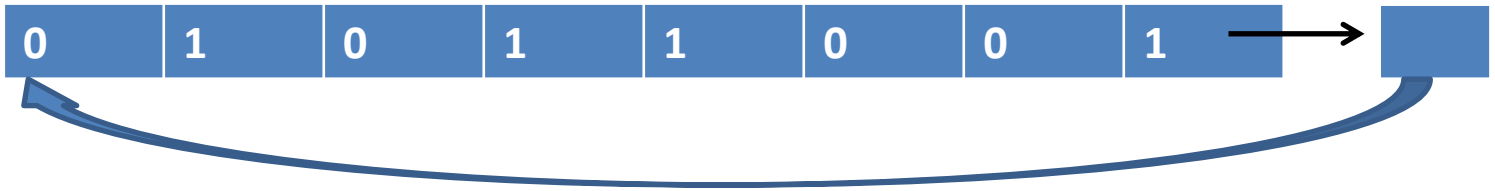


OF=1 (7. bit değişti)

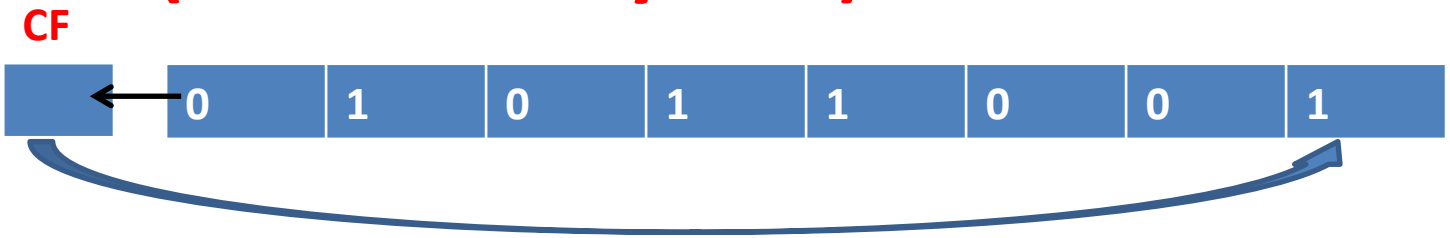
8086 16-Bit Mikroişlemci

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

- SORU1: sayi=11001000b içindeki ‘1’ bitlerinin sayısını bulan ve bu sayıyı birsayisi değişkeninde saklayan 8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h

MOV CX,8
MOV AL, sayi

dongu:
JCXZ bitir
SHR AL,1  ; CF eger 1 se
JC birarttir
LOOP dongu


birarttir:
DEC CX
INC birsayisi
JMP dongu


bitir:
ret

sayi db 11001000b
birsayisi db 0
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU2: sayilar=2,5,4,-9 dizisindeki her bir elemanın değerini 2 arttıran programı loop döngüsü kullanmadan 8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h


MOV CX,4
MOV SI,0

dongu:
add [sayilar+SI],2
INC SI
DEC CX   ; CX degerini azaltmazsaniz sonsuz donguye girer
JCXZ  bitir  ;CX=0 oldugunda bitir etiketine gider
JMP dongu


bitir:

ret


sayilar db 2,5,4,-9
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU3: 8-bitlik işaretsiz iki sayının toplanması sonucunda eğer taşma varsa tasma değişkeninin değeri 1 yapılsın diğer durumda 0 olsun 8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h


MOV AL,0F2h
MOV BL,0DEh

ADD AL,BL
JC  elde       ;CF=1 yani tasma var bu durumda elde etiketine gider
JMP bitir



elde:
MOV tasma,1
MOV CL,tasma
JMP bitir

bitir:
ret

tasma db 0
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU4: sayilar=-12,5,-6,9,-13,8 dizisindeki negatif sayıları negatif, pozitif sayıları pozitif dizisine atan 8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h


MOV CX,6
MOV SI,0
MOV DI,0
MOV BP,0


dongu:
JCXZ bitir              ; CX=0 sifirlandi mi kontrol ediliyor
MOV BL,[sayilar+SI]
CMP BL,0
JL negatifetiket        ; eger BL 0 dan kucukse negatif etiketine git
JMP pozitifetiket
LOOP dongu
JMP bitir


negatifetiket:
INC SI
DEC CX
MOV [negatif+DI],BL
INC DI
JMP dongu


pozitifetiket:
INC SI
DEC CX
MOV [pozitif+BP],BL
INC BP
JMP dongu


bitir:
ret

sayilar db -12,5,-6,9,-13,8
negatif db 6 dup(?)
pozitif db 6 dup(?)
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU5:

kelime1=bukelime kelime2=bukelimE

- kelime1 ve kelime2 değişkenlerini harf harf kontrol eden bu iki kelime birbiriyle aynıysa kontrol değişkenini 1 yapan diğer durumda 0 yapan
- Ayrıca aynı olan harf sayisini aynikontrol değişkeninde tutan

8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h

MOV CX,8
MOV SI,0


dongu:
JCXZ bitir
MOV AL,[kelime1+SI]
CMP [kelime2+SI],AL
JE ayni
INC SI
LOOP dongu
JMP bitir


ayni:
INC SI
DEC CX
INC aynikontrol
JCXZ kontroldegistir
JMP dongu



kontroldegistir:
MOV kontrol,1
JCXZ bitir
JMP dongu



bitir:
ret

kelime1 db 'bukelime'
kelime2 db 'bukelimE'
kontrol db 0
aynikontrol db 0
```

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

- SORU6: sayi=11001001b sayısını 8-bitini sayi2 dizisine aktaran 8086 Assembly kodunu yazınız

8086 16-Bit Mikroişlemci

EMU 8086-MICROPROCESSOR EMULATOR

```
org 100h
MOV CX,8
MOV AL, sayi
MOV SI,7


dongu:
JCXZ bitir
SHR AL,1 ; 8. biti CF de


JC bir ; CF=1 ise 1 etiketi
JMP sifir ; CF=0 ise sifir etiketi
LOOP dongu


bir:
DEC CX
STC          ; CF=1 oldu
RCR sayi3,1

DEC SI
JMP dongu



sifir:
DEC CX
CLC     ; CF=0
RCR sayi3,1
DEC SI
JMP dongu


bitir:


ret
sayi db 11001000b
sayi3 db 00000000b
```

KAYNAKLAR

- http://www.mlsite.net/8086/#oper_b

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