MİKROİŞLEMCİLER LAB.

1.DENEY: BASİT BİR PROGRAMI OLUŞTURMA VE ÇALIŞTIRMA

YENİ KOMUTLAR

MOV [operand1],[operand2]
Açıklama : operand2'nin içeriği operand1'e kopyalanır.
Algoritma : operand1 = operand2

ADD [operand1],[operand2]
Açıklama : operand1 ile operand2 toplanır. Sonuç operand1'e yazılır.
Algoritma : operand1 = operand1 + operand2

SUB [operand1],[operand2]
Açıklama : operand1'den operand2 çıkarılır. Sonuç operand1'e yazılır.
Algoritma : operand1 = operand1 - operand2

ÖRNEKLER

1. Bellekteki **0100:1000h** ve **0100:2000h** adreslerine **34h** değerini yazacak bir program yazınız.

```
;program
MOV [1000h], 34h
MOV [2000h], 34h

;isletim sistemine donus
MOV AH, 4Ch
INT 21h
```

2. **CL** ve **DL** registerlarındaki değerlerin yerlerini değiştirecek bir program yazınız.

```
;ilk degerler
MOV CL, 0CCh
MOV DL, 0DDh

;program
MOV AL, CL
MOV CL, DL
MOV DL, AL

;isletim sistemine donus
MOV AH, 4Ch
INT 21h
```

3. **0100:0500h** bellek adresindeki **9Bh** değeri ile **0100:0501h** bellek adresindeki **52h** değerini toplayan ve sonucu **0100:0502h** bellek adresine yazan bir program yazınız.

```
;program
MOV [0500h], 9Bh
MOV [0501h], 52h
```

1.DENEY: BASİT BİR PROGRAMI OLUŞTURMA VE ÇALIŞTIRMA

```
MOV AL, [0500h]
MOV AH, [0501h]
ADD AL, AH

MOV [0502h], AL

;isletim sistemine donus
MOV AH, 4Ch
INT 21h
```

MİKROİŞLEMCİLER LAB.

2.DENEY: ADRESLEME MODLARI

AÇIKLAMALAR

```
DS = 0100h
BX = 1000h
DI = 2000h
DIZI = 05BCh
```

MOD	ÖRNEK	ADRES
Immediate addressing	ADD CH, 43h	-
Register addressing	ADD DL, CL	-
Direct addressing	SUB byte ptr [1200h], 20h	DS*10h + 1200h = 02200h
Register indirect addressing	MOV AL, [BX]	DS*10h + BX = 02000h
Base-plus-index addressing	ADD CX, [BX+DI]	DS*10h + BX + DI = 04000h
Register relative addressing	MOV AX, [DI+05BCh] MOV AL, DIZI [DI]	DS*10h + DI + 05BCh = 035BCh
Base relative-plus-index addressing	SUB DX, DIZI [BX+DI]	DS*10h + BX + DI + 05BCh = 045BCh

MİKROİŞLEMCİLER LAB.

3.DENEY: KARŞILAŞTIRMA VE ATLAMA KOMUTLARININ KULLANIMI

YENİ KOMUTLAR

CMP [operand1], [operand2]

Açıklama : *operand1*'den *operand2* çıkartılır. Sonuç hiçbir yerde saklanmaz. Sadece ilgili *flag*'ların (OF, SF, ZF, AF, PF, CF) değerleri değişir.

Algoritma : operand1 - operand2

J?? [label]

Açıklama : Eğer ?? ile gösterilen önerme doğru ise, programda *label* ile gösterilen yere atlar. Olası Komutlar; JA, JAE, JB, JBE, JC, JE, JG, JGE, JL, JLE, JNA, JNAE, JNB, JNBE, JNC, JNE, JNG, JNGE, JNL, JNLE, JNO, JNP, JNS, JNZ, JO, JP, JS, JZ, ...

JMP [label]

Açıklama : Programda *label* ile gösterilen satıra koşulsuz atlama yapar.

Algoritma : jump to label

LOOP [label]

Açıklama : *CX register*'ını bir azaltır. Ardından, eğer *CX* sıfır değilse programda *label* ile gösterilen yere atlar. Başka bir ifadeyle *label* ile *LOOP* komutu arasındaki kodlar *CX*'in değeri defa işletilir.

Algoritma :
- CX = CX - 1
- if CX <> 0 then
  jump to label
  else
  no jump, continue

INC [operand]

Açıklama : *operand*'ın değerini bir arttırır.

Algoritma : operand = operand + 1

DEC [operand]

Açıklama : *operand*'ın değerini bir azaltır.

Algoritma : operand = operand – 1

ÖRNEKLER

1. **0100:1000h** adresini inceleyecek bir program yazınız. Bu program eğer bu adresteki değer **C2h** ise **0100:1100h** adresine **FFh** değerini, eğer **C2h** değilse **0100:1100h** adresine **AAh** değerini yazacak.

```
;ilk degerler
MOV   byte ptr [1000h], 0ABh

;program
CMP   byte ptr [1000h], 0C2h
JE    esit
MOV   byte ptr [1100h], 0AAh
JMP   bitis
esit:
MOV   byte ptr [1100h], 0FFh
```

```
bitis:
;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

2. Bellekte **0100:1000h** adresinden yukarıya doğru değeri **00h** olmayan bir baytlık bilgiler den oluşan bir seri depolanmıştır. Bu seri **00h** değeri ile sonlandırılmıştır. Bu seriyi **0100:2000h** adresinden yukarıya doğru kopyalacak bir program yazınız.

```
;ilk degerler
MOV   byte ptr [1000h], 12h
MOV   byte ptr [1001h], 34h
MOV   byte ptr [1002h], 56h
MOV   byte ptr [1003h], 78h
MOV   byte ptr [1004h], 9Ah
MOV   byte ptr [1005h], 0BCh
MOV   byte ptr [1006h], 0DEh
MOV   byte ptr [1007h], 0F0h
MOV   byte ptr [1008h], 00h

;program
DIZI  EQU 1000h
MOV   BX, 1000h
MOV   DI, 0000h

dongu:
MOV   AL, DIZI[DI]
CMP   AL, 00h
JE    bitis

MOV   DIZI[BX+DI], AL
INC   DI
JMP   dongu

bitis:
;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

3. **0100:2000h** ile **0100:200Fh** arasındaki bellek adreslerindeki bir *word*'lük değerleri inceleyen bir program yazınız. Bu değerlerden en büyüğünü **0100:1000h** adresine kopyalayınız.

```
;ilk degerler
MOV   word ptr [2000h], 1234h
MOV   word ptr [2002h], 5678h
MOV   word ptr [2004h], 9ABCh
MOV   word ptr [2006h], 0DEF1h
MOV   word ptr [2008h], 2345h
MOV   word ptr [200Ah], 6789h
```

3.DENEY: KARŞILAŞTIRMA VE ATLAMA KOMUTLARININ KULLANIMI

```
MOV   word ptr [200Ch], 0ABCDh
MOV   word ptr [200Eh], 0EF12h

;program
DIZI  EQU 2000h
MOV   BX, 1000h
MOV   DI, 0000h
MOV   CX, 0007h

MOV   AX, DIZI[DI]
ADD   DI, 0002h

dongu:
MOV   DX, DIZI[DI]
CMP   AX, DX
JGE   buyuk_esit
MOV   AX, DX
buyuk_esit:
ADD   DI, 0002h
LOOP  dongu

MOV   [BX], AX

;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

4. **0100:1500h** ile **0100:15FFh** arasındaki bellek adreslerini aşağıdaki tabloda gösterildiği gibi yükleyen bir program yazınız.

0100:1500h	**00h**
0100:1501h	**01h**
⋮	⋮
0100:15FDh	**FDh**
0100:15FEh	**FEh**
0100:15FFh	**FFh**

```
;program
DIZI  EQU 1500h
MOV   BX, 0000h
MOV   CX, 0100h

dongu:
MOV   DIZI[BX], BL
INC   BL
LOOP  dongu

;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

3.DENEY: KARŞILAŞTIRMA VE ATLAMA KOMUTLARININ KULLANIMI

5. **0100:3000h** adresinde herhangi bir değer olabilir. Bu değeri **C3h** değerinden küçükse birer birer arttırarak yada **C3h** değerinden büyükse birer birer azaltarak **C3h** değerine getirecek bir program yazınız.

```
;ilk degerler
MOV   byte ptr [3000h], 0ABh

;program
CMP   byte ptr [3000h], 0C3h
JG    buyuk

kucuk:
CMP   byte ptr [3000h], 0C3h
JE    bitis
INC   byte ptr [3000h]
JMP   kucuk

buyuk:
CMP   byte ptr [3000h], 0C3h
JE    bitis
DEC   byte ptr [3000h]
JMP   buyuk

bitis:
;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

MİKROİŞLEMCİLER LAB.

4.DENEY: VERİ TAŞIMA KOMUTLARI (STRING VERİ TRANSFERİ)

YENİ KOMUTLAR

STOSB

Açıklama : *AL* registerındaki baytı *ES:[DI]* adresine kaydet. *DI* registerını güncelle.

Algoritma :
- ES:[DI] = AL
- if DF = 0 then
  DI = DI + 1
  else
  DI = DI - 1

STOSW

Açıklama : *AX* registerındaki wordü *ES:[DI]* adresine kaydet. *DI* registerını güncelle.

Algoritma :
- ES:[DI] = AX
- if DF = 0 then
  DI = DI + 2
  else
  DI = DI - 2

LODSB

Açıklama : *DS:[SI]* adresindeki baytı *AL* registerına yükle. *SI* registerını güncelle.

Algoritma :
- AL = DS:[SI]
- if DF = 0 then
  SI = SI + 1
  else
  SI = SI - 1

LODSW

Açıklama : *DS:[SI]* adresindeki wordü *AX* registerına yükle. *SI* registerını güncelle.

Algoritma :
- AX = DS:[SI]
- if DF = 0 then
  SI = SI + 2
  else
  SI = SI - 2

MOVSB

Açıklama : *DS:[SI]* adresindeki baytı *ES:[DI]* adresine kopyala. *SI* ve *DI* registerlarını güncelle.

Algoritma :
- ES:[DI] = DS:[SI]
- if DF = 0 then
  SI = SI + 1
  DI = DI + 1
  else
  SI = SI - 1
  DI = DI - 1

4.DENEY: VERİ TAŞIMA KOMUTLARI (STRING VERİ TRANSFERİ)

MOVSW

Açıklama : *DS:[SI]* adresindeki wordü *ES:[DI]* adresine kopyala. *SI* ve *DI* registerlarını güncelle.

Algoritma : • ES:[DI] = DS:[SI]
• if DF = 0 then
SI = SI + 2
DI = DI + 2
else
SI = SI - 2
DI = DI - 2

CLD

Açıklama : *DF*'yi sıfırla. *SI* ve *DI* registerları *chain instruction*lar (*STOSB, STOSW, LODSB, LODSW, MOVSB, MOVSW*) tarafından arttırılacaktır.

Algoritma : • DF = 0

STD

Açıklama : *DF*'yi setle. *SI* ve *DI* registerları *chain instruction*lar (*STOSB, STOSW, LODSB, LODSW, MOVSB, MOVSW*) tarafından azaltılacaktır.

Algoritma : • DF = 1

REP [chain instruction]

Açıklama : *chain instruction*'ı (*STOSB, STOSW, LODSB, LODSW, MOVSB, MOVSW*) *CX* defa tekrarla.

Algoritma : • CX = CX - 1
• if CX <> 0 then
Do chain instruction
else
continue

ÖRNEKLER

1. **0100:0300h** ile **0100:03FFh** arasındaki bellek adreslerine **BCDEh** değerini yazacak bir program yazınız.

```
;program
CLD
MOV   DI, 0300h
MOV   CX, 0080h
MOV   AX, 0BCDEh
REP   STOSW

;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

2. **0100:0300h** ile **0100:03FFh** arasındaki bellek adreslerinden **1234h** çıkartıcak bir program yazınız.

```
;ilk degerler
CLD
MOV   DI, 0300h
MOV   CX, 0080h
```

4.DENEY: VERİ TAŞIMA KOMUTLARI (STRING VERİ TRANSFERİ)

```
MOV   AX, 0BCDEh
REP   STOSW

;program
CLD
MOV   SI, 0300h
MOV   DI, 0300h
MOV   CX, 0080h
MOV   DX, 1234h

dongu:
LODSW
SUB   AX, DX
STOSW
LOOP  dongu

;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

3. **0100:0300h** ile **0100:03FFh** arasındaki bellek adreslerindeki değerleri **0100:3500h** ile **0100:35FFh** arasındaki bellek adreslerine kopyalayacak bir program yazınız.

```
;ilk degerler
CLD
MOV   DI, 0300h
MOV   CX, 0080h
MOV   AX, 0BCDEh
REP   STOSW

;program
CLD
MOV   SI, 0300h
MOV   DI, 3500h
MOV   CX, 0080h
REP   MOVSW

;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

4. Data segment içinde **34h** değerini arayan bir program yazınız. Bu programda **34h** değerinin bulunduğu offset adresleri **0100:1000h** adresinden yukarıya doğru yazılacak ve kaç kere bulunduğu **CX**'de tutulacak.

```
;ilk degerler
MOV   byte ptr[0500h], 34h
MOV   byte ptr[0600h], 34h
MOV   byte ptr[0700h], 34h

;program
CLD
```

4.DENEY: VERİ TAŞIMA KOMUTLARI (STRING VERİ TRANSFERİ)

```
MOV   SI, 0000h
MOV   CX, 0000h
MOV   DI, 1000h
MOV   DL, 34h

dongu:
LODSB
CMP   AL, DL
JNE   bulunamadi

MOV   AX,SI
DEC   AX
STOSW
INC   CX

bulunamadi:
CMP   SI, 0000h
JNE   dongu

;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

MİKROİŞLEMCİLER LAB.

5.DENEY: TEMEL ARİTMETİK KOMUTLAR - 1

YENİ KOMUTLAR

ADD [operand1], [operand2]
Açıklama : *operand1* ile *operand2* toplanır, sonuç *operand1*'e yazılır.
Algoritma : • operand1 = operand1 + operand2

SUB [operand1], [operand2]
Açıklama : *operand1*'den *operand2* çıkartılır, sonuç *operand1*'e yazılır.
Algoritma : • operand1 = operand1 - operand2

ADC [operand1], [operand2]
Açıklama : *operand1, operand2 ve CF (carry flag)* toplanır, sonuç *operand1*'e yazılır.
Algoritma : • operand1 = operand1 + operand2 + CF

SBB [operand1], [operand2]
Açıklama : *operand1'*den *operand2* ve *CF* çıkartılır, sonuç *operand1*'e yazılır.
Algoritma : • operand1 = operand1 - operand2 - CF

ÖRNEKLER

1. **12125656h** ile **9876FFFFh** sayılarını toplayıp **0100:5000h** adresine kaydedecek bir program yazınız.

```
;program
MOV   AX, 1212h
MOV   BX, 5656h
MOV   CX, 9876h
MOV   DX, 0FFFFh

ADD   BX, DX
ADC   AX, CX

MOV   [5000h], BX
MOV   [5002h], AX

;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

2. **98765432h** sayısından **12345678h** sayısını çıkartıp **0100:5000h** adresine kaydedecek bir program yazınız.

```
;program
MOV   AX, 9876h
MOV   BX, 5432h
MOV   CX, 1234h
MOV   DX, 5678h

SUB   BX, DX
SBB   AX, CX
```

```
MOV   [5000h], BX
MOV   [5002h], AX

;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```

3. **SI** ile gösterilen bellek adesinden başlayıp **DI** ile gösterilen bellek adresine kadarki bir baytlık verileri toplayan bir program yazınız. Sonucu **DX-BX** register kombinasyonunda saklayınız.

```
;ilk degerler
MOV   SI, 0000h
MOV   DI, 0100h

;program
CLD
MOV   CX, DI
SUB   CX, SI
INC   CX
MOV   BX, 0000h
MOV   DX, 0000h

dongu:
LODSB
ADD   BL, AL
ADC   BH, 00h
ADC   DX, 0000h
LOOP  dongu

;isletim sistemine donus
MOV   AH, 4Ch
INT   21h
```