

1) İki yazmaç içerisindeki değerleri değiştirmeyi, tablo halinde temsili ifadelerle (* ve !) gösterirsek;

Komut	AX	BX	CX	Açıklama
İLK DURUM	*	!		
MOV CX, AX	*	!	*	AX'deki değer CX'e atıldı
MOV AX, BX	!	!	*	BX'deki değer AX'e atıldı
MOV BX, CX	!	*	!	CX'deki değer BX'e atıldı

CX yazmacı burada geçici bellek birimi olarak kullanılmıştır.

2) İlk soruda yaptığımız işlemi tek bir komut ile şu şekilde gerçekleştirebiliriz;

XCHG AX, BX

3) Tablo halinde temsili ifadelerle (* ve !) gösterirsek, bu yer değiştirme şu kod ile yapılabilir;

Komut	DS:300	DS:400	AX	Açıklama
İLK DURUM	*	!		
MOV AX, [300]	*	!	*	300'ün gösterdiği yerdeki değer AX'e atıldı
XCHG AX, [400]	*	*	!	AX ile 400'ün gösterdiği yerdeki değer değiştirildi
MOV [300], AX	!	*	!	AX'in değeri, 300'ün gösterdiği yere atıldı

Sonuçta görüldüğü gibi, 300 ve 400'ün gösterdiği yerdeki değerler değiştirilmiş oldu.

4) **ADD AX,1** ve **INC AX** komutlarının ikisi de AX yazmacının değerini 1 artırır. **ADD AX,1** komutu hafızada 3 byte yer kaplarken, **INC AX** komutu 1 byte yer kaplar. Büyük bir programda, **INC** komutu belleği daha az işgal eder. Buna ek olarak, **INC** komutu bu işlemi daha hızlı sürede gerçekleştirir. Bu yüzden **INC** komutu daha avantajlıdır.

5) **LEA** (load effective address) komutu, göreceli konum adreslerini hedef operanda yükleyemeyi sağlar. **LEA AX, [BX]** komutunu kullandığımızda, BX yazmacının içeriği AX yazmacına kopyalanır. Yani, **MOV AX, BX** komutuyla aynıdır. **MOV AX, [BX]** komutunu kullandığımız zaman ise, DS: BX hafızasında bulunan veri AX yazmacının içerisine aktarılır.

6) **ADC** (Add With Carry) komutunda toplama işlemi yapılırken Carry de hesaba katılır ve eklenir. **ADC AX, BX** komutuna göre, AX ve BX yazmaçlarının içindeki değerler ve Carry Flag'deki değer de toplanarak AX yazmacına atanır.

a) Carry Flag 1 ise, $AX + BX + 1$ işleminin sonucu AX'e yazılır.

b) Carry Flag 0 ise, $AX + BX + 0$ işleminin sonucu AX'e yazılır.

7) **NEG** komutu ile sayının ikilik komplementi alınır. Bu arada programda bayrakların değeri de değişir.

PL → NG : Sayının işaretinin ilk durumda pozitif, sonra negatif olduğunu gösterir.

NA → AC : Not Auxiliary Carry durumdan, Auxiliary Carry durumuna geçildiğini gösterir.

PO → PE : Parity bit'in 0 iken 1 olduğunu gösterir.

NC → CY : Carry yok iken, Carry olan duruma geçildiğini gösterir.

8) $Z = A + B * (C - D)$ işleminin gerçekleşmesi için;

```
MOV BL, [501]
MOV CL, [502]
MOV DL, [503]
SUB CL, DL
MOV AL, CL
MUL BL
ADD AL, [500]
MOV [504], AL
```

Bu kodlar ile, önce yazmaçlarının içerisine bilgiler atanmıştır. SUB CL, DL komutu ile CL yazmacının içeriği CL –DL işleminin sonucu oldu. MUL BL komutu ile $AX = AL * BL$ işlemi gerçekleştirilir.

9) MUL komutu işaretli sayıların , IMUL komutu ise işaretli sayıların çarpılmasında kullanılır.

AL = 20H 32 (Decimal)

CL = 80H 128 (Decimal)

MUL CL → AX = 1000H değerine sahip olur.

Çünkü $AL * CL = 4096$ (Decimal) değerini verecektir.

$4096 = 1000H$ ‘ tir

IMUL CL → AX = F000 değerine sahip olur .

Çünkü **IMUL** komutu işaretli sayılar üzerinde çarpım yapar. MUL komutu sonucu ortaya çıkan 1000h = 0001 0000 0000 0000 ikili değeri IMUL komutu kullanımı ile işaretli sayı olarak algılanır. Bundan dolayı 1111 0000 0000 0000 = F000H değerine sahip olur.

Debug ortamında bayrakların durumuna bakarsak;

MUL CL sonucu ;

-a 100

0C7F:0100 mul cl

0C7F:0102

-t

AX=1000 BX=0000 CX=0080 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0C7F ES=0C7F SS=0C7F CS=0C7F IP=0102 OV UP EI PL NZ NA PO CY

0C7F:0102 04DA ADD AL,DA

IMUL CL sonucu ;

-a 100

0C7F:0100 IMUL CL

0C7F:0102

-t

AX=F000 BX=0000 CX=0080 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0C7F ES=0C7F SS=0C7F CS=0C7F IP=0102 OV UP EI PL NZ NA PO CY

0C7F:0102 04DA ADD AL,DA

10) a) AX ve BX yazmaçlarında saklanan değerleri sadece kaydırma işlevini kullanarak 2 ile bölme ;

-A 100

0C7F:0100 SHR AX , 1

0C7F:0102 SHR BX , 1

0C7F:0104

-T

AX=0020 BX=0041 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C7F ES=0C7F SS=0C7F CS=0C7F IP=0102 NV UP EI PL NZ NA PO NC
0C7F:0102 D1EB SHR BX,1

-T

AX=0020 BX=0020 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C7F ES=0C7F SS=0C7F CS=0C7F IP=0104 NV UP EI PL NZ NA PO CY
0C7F:0104 020B ADD CL,[BP+DI] SS:0000=CD

Bayrak durumlarına baktığımız zaman **SHR BX , 1** komutu ile elde yok durumundan (NC), elde var durumuna (CY) geçmiştir.

AX ve BX yazmaçlarında saklanan değerleri sadece kaydırma işlevini kullanarak 2 ile çarpma ;

-a

0C7F:0104 SHL AX , 1

0C7F:0106 SHL BX , 1

0C7F:0108

-T

AX=0040 BX=0020 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C7F ES=0C7F SS=0C7F CS=0C7F IP=0106 NV UP EI PL NZ NA PO NC
0C7F:0106 D1E3 SHL BX,1

-T

AX=0040 BX=0040 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C7F ES=0C7F SS=0C7F CS=0C7F IP=0108 NV UP EI PL NZ NA PO NC
0C7F:0108 022604F0 ADD AH,[F004] DS:F004=00

Yazmaç içerikleri AX = BX = 0040H olurken, bayrak durumu tekrar elde yok (NC) durumuna geçerken, sıfır bayrağı değişmemiştir.

AX yazmacının içeriğini 1 artırıp, AX ve BX yazmaçlarının içeriklerini ve bayrak durumlarını kontrol edelim ;

-a

0C7F:0108 INC AX

0C7F:0109

-T

AX=0041 BX=0040 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0C7F ES=0C7F SS=0C7F CS=0C7F IP=0109 NV UP EI PL NZ NA PE NC

0C7F:0109 26 ES:

0C7F:010A 04F0 ADD AL,F0

AX yazmacının içeriği 1 arttı, BX yazmacının içeriği ve bayrak durumları değişmemiştir.

AX ve BX yazmaçlarının içeriğini birbiriyle değiştirirsek ;

-a

0C7F:0109 XCHG AX , BX

0C7F:010B

-T

AX=0040 BX=0041 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0C7F ES=0C7F SS=0C7F CS=0C7F IP=010B NV UP EI PL NZ NA PE NC

0C7F:010B F0 LOCK

0C7F:010C 0204 ADD AL,[SI] DS:0000=CD

AX ve BX yazmaçlarının içerikleri değişti, bayrak durumlarında bir değişiklik olmamıştır.

b) 16 (decimal) → 0000 0000 0001 0000 (binary) → 0010 (hex)

O halde -16 (decimal) → 1111 1111 1111 0000 → FFF0 (hex) olur.

-A

0C7F:010B SHR AX , 1

0C7F:010D

-T

AX=7FF8 BX=0041 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0C7F ES=0C7F SS=0C7F CS=0C7F IP=010D OV UP EI PL NZ NA PO NC

0C7F:010D 0404 ADD AL,04

Bu sayede AX yazmacının içeriğine ikiye bölmüş olduk.