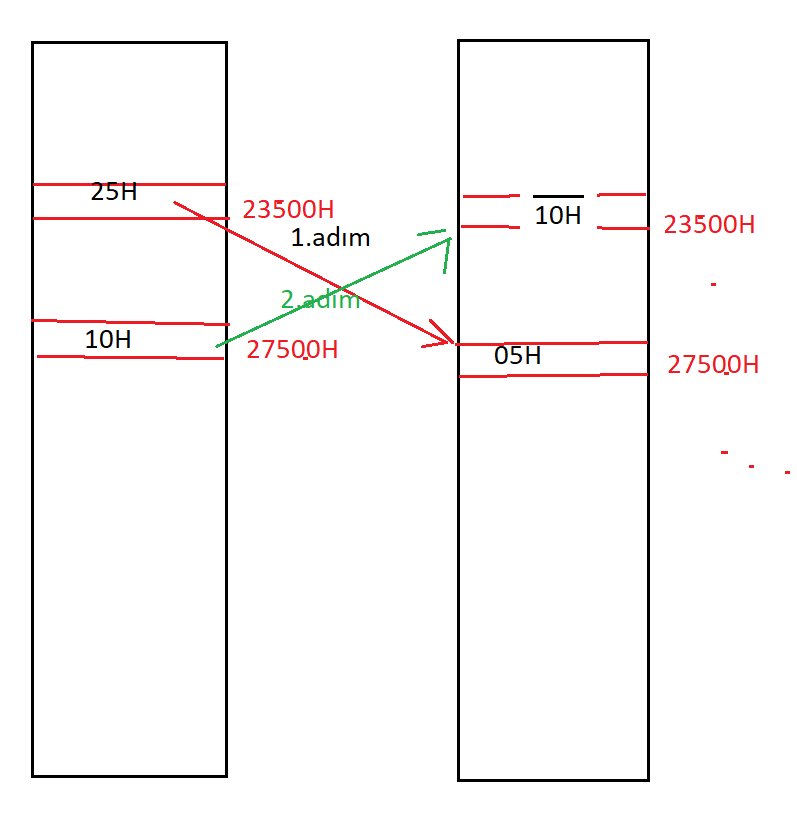
ÖRNEK: 27500H adresli hafıza hücresinde kayıtlı 8 bitlik sayının, en yüksek seviyeli 4 bitini sıfırladıktan sonra 23500H adresli hafıza hücresinde bulunan 8 bitlik sayının tersi ile yer değiştiren 8086 komut kümelerini yazınız.



En yüksek seviyeli 4 biti sıfırlama kodu:

BL: 25H = 0010 0101B (25H)

AND 0000 1111B (0FH)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. 0101B (05H)

En düşük seviyeli 4 biti sıfırlama kodu:

BL: 25H = 0010 0101B (25H)

AND 1111 0000B (F0H)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0010 0000B (20H)

23500H= (2000H)\*10H+3500H

27500H=(2000H)\*10H+7500H

MOV DX,2000H

MOV DS,DX //artık 20000H tabanındayız.

MOV BL, [7500H] // 27500H daki 8 bitlik sayı BL'ye yüklendi.

AND BL, 0FH // En yüksek seviyeli 4 bit sıfırlanmıştır.

NOT [7500H]

XCHG [7500H],BL //2.adım tamamlandı

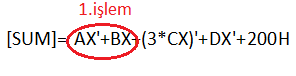
MOV [3500H],BL // 1.adım tamalandı

ÖRNEK: [SUM]= AX'+BX+(3\*CX)'+DX'+200H

Yukarıda verilen mantıksal ifadenin çözümünü [SUM] hafıza lokasyonuna yükleyen 8086 komut kümelerini yazınız.

*NOT: \* AND (VE) işlemi, + OR (VEYA) işlemidir.*

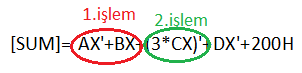
*1. işlem*



NOT AX

OR AX,BX // 1.işlem sonucu AX'dedir.

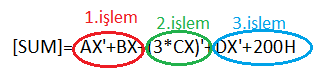
*2. işlem*



AND CX,03H

NOT CX // 2.işlem sonucu CX

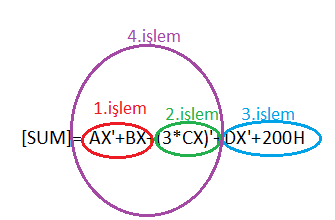
*3.işlem*

**

NOT DX

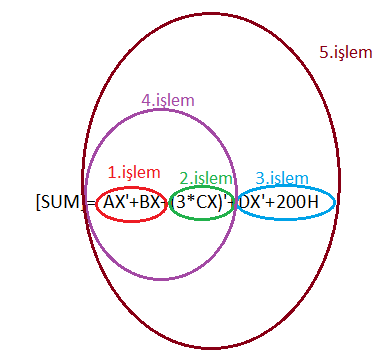
OR DX,200H // 3 işlem sonucu DX'de

*4.işlem (1 işlem ile 2. işlemi birleştirme)*

**

OR AX,CX // 4 işlem sonucu AX'de

*5.işlem (4 ile 3 işlemin birleştirilmesi)*

**

OR AX,DX // 5 işlemin sonucu AX'de

*6. işlem (mantıksal ifadenin sonucunun [SUM] yüklenmesi*

MOV [SUM], AX // Tüm ifadenin sonucu [SUM] hafıza lokasyonuna yüklenmiştir.

*Tüm kodlara tek bakış*

NOT AX

OR AX,BX // 1.işlem sonucu AX'dedir.

OR CX,03H

NOT CX // 2.işlem sonucu CX

NOT DX

OR DX,200H // 3 işlem sonucu DX'de

OR AX,CX // 4 işlem sonucu AX'de

OR AX,DX // 5 işlemin sonucu AX'de

MOV [SUM], AX // Tüm ifadenin sonucu [SUM] hafıza lokasyonuna yüklenmiştir.

*ÖDEV:*

(AX'+BX'+FFH)'+4\*BX+(3\*CX+AX)'+DX'

Yukarıda verilen mantıksal ifadenin sonucunun ilk düşük seviyeli 8 bitini sıfırladıktan sonra tersini 28FFFH adresli hafıza lokasyonuna yükleyen 8086 komut kümelerini yazınız.

*En düşük seviyeli 8 biti sıfırlama kodu:*

BL: 5625H = 0101 0110 0010 0101B (5625H)

AND 1111 1111 0000 0000B (FF00H)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0101 0110 0000 0000B (5600H)

28FFFH=(2000H)\*10H+8FFFH

MOV DX,2000H

MOV DS,DX // artık 20000H tabanındayız.

MOV DI,AX // AX iki yerde var, bu yüzden yedekliyoruz.

NOT AX

MOV SI,BX

NOT BX

OR AX,BX

OR AX, FFH

NOT AX // (AX'+BX'+FFH)' artık AX de

AND SI, 04H

OR AX,SI // (AX'+BX'+FFH)'+4\*BX ifadesi AX’de

AND CX,03H

OR CX,DI

NOT CX // (3\*CX+AX)' ifadesi CX’de

NOT DX

OR CX,DX // (3\*CX+AX)'+DX' ifadesi CX’de

OR AX,CX // Tüm ifade(AX'+BX'+FFH)'+4\*BX+(3\*CX+AX)'+DX' artık AX de

AND AX,FF00H // ifadenin sonucunun ilk 8 biti sıfırlandı

NOT AX

MOV [8FFFH],AX // tüm sonuç AX=[8FFFH] yani 28FFFH hafıza lokasyonuna yük.

Genişletme İfadeler:

CBW ve CWD komutları vardır.

***CBW:***

CBW (Convert Byte to Word) Byte’tı kelimeye dönüştür.

Açıklama: 8bit lik değerden 16 bitlik değer elde edilir.

Özet olarak; AL kaydedicisinin 1 byte’lık içeriğini AX’e genişletir. AL’deki değer pozitifse AH’ın tüm bitleri ‘0’ değerini alır. AL’deki değeri negatifse AH’ın tüm bitleri ‘1’ olur.

NOT: E.Y.B ‘0’ ise sayı pozitifdir, eğer ‘1’ ise negatifdir.

1 byte=8 bit

1 Kelime = 16bit

İsim Biçim Anlam

CBW CBW Byte kelimeye’e dönüşür.

ÖR\_1:

AL : 68H AH:XX // AH ın ilk başta ne olduğu önemli değil.

MOV AL, 68H // AL nin e.y.b (sayının pozitif mi negatif mi olduğuna bakar!)

CBW // AL: 0110 1000B sayı pozitif bu durumda AH: 0000 0000B olur.

Yukarıdaki ifadeler çalıştıktan sonra, AL:68 AH: 00H AX: 0000 0000 0110 1000B (0068H)

Ör\_2:

MOV AL, 9FH

CBW // AL: 1001 1111B AH: 1111 1111B AX: 1111 1111 1001 1111B = FF9FH

İfadeler çalıştığında AX=? AX=FF9FH

***CWD:*** (Convert Word to Double Word) Kelimeyi çift kelimeye dönüştür.

Açıklama: 16 bit lik değerden 32 bitlik değer elde edilir.

Özet olarak; AX kaydedicisinin 1 byte’lık içeriğini (DX;AX)’e genişletir. AX’deki değer pozitifse DX’ın tüm bitleri ‘0’ değerini alır. AX’deki değeri negatifse DX’ın tüm bitleri ‘1’ olur.

Ör\_1:

MOV AX, 67FDH

CWD // AX: 0110 0111 1111 1101B DX: 0000 0000 0000 0000B

Sırasıyla ifadeler çalıştığında (DX;AX)=? (0000H; 67FDH)

ÖR\_2:

MOV AL, FBH

CBW // AL: 1111 1011B AH: 1111 1111B AX: 1111 1111 1111 1011B

CWD // AX: 1111 1111 1111 1011B DX: 1111 1111 1111 1111B

İfadeler çalıştığında (DX;AX)=? (DX;AX)= (FFFBH; FFFFH)

**Aritmetik ifadeler:**

**#** Aritmetik ifadeler işaretli ve işaretsiz çarpım ve bölüm, arttırma, eksiltme, eldeli eldesiz toplama ile borçlu borçsuz çıkarma ifadelerinden oluşmaktadır.

Toplama/Çıkarma ifadeleri:

|  |  |  |
| --- | --- | --- |
| İsim | Biçim | Anlam |
| ADD | ADD H,K | H=H+K; hedef kaynak toplanır, sonuç hedefe yazılır. |
| ADC | ADC H,K | H=H+K+E; hedef kaynak varsa elde ile toplanır, sonuç hedefe yazılır. (E:elde) |
| SUB | SUB H,K | H=H-K; hedef’den kaynak çıkarılır sonuç hedefe yazılır. |
| SBB | SBB H,K | H=H-K-B; hedef’den kaynak varsa borç çıkraılır. Sonuç hedefe yazılır. |

ÖDEV:

Y=(AX’+BX+35H-(CX’+2FH)’)+DX’

Yukarıda verilen aritmetik ifadenin sonucunu FF0E0H adresine yükleyen 8086 komut kümelerini yazınız.