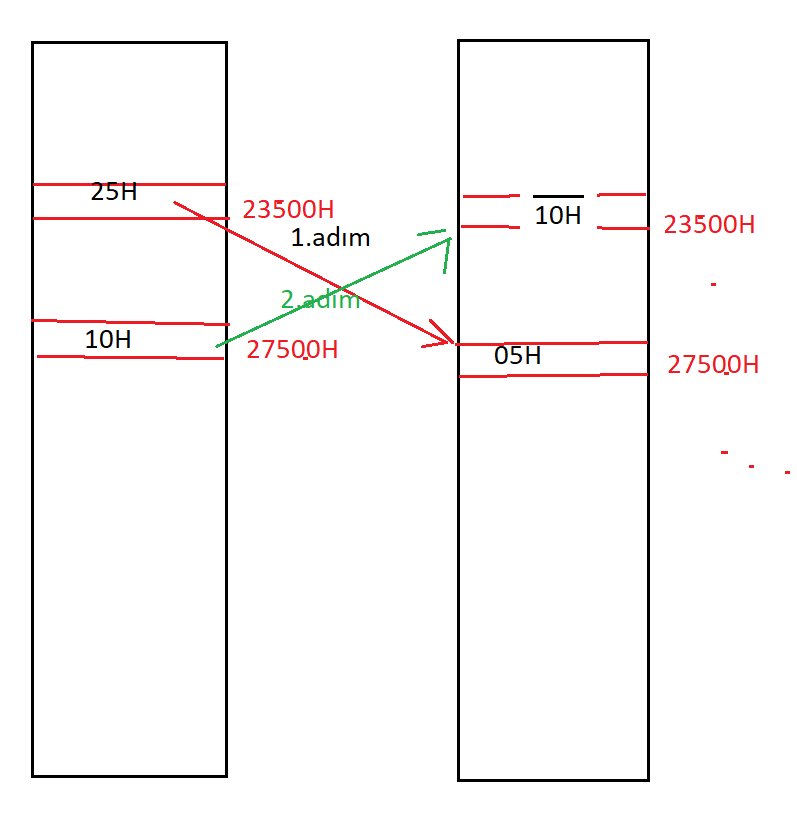
ÖRNEK: 27500H adresli hafıza hücresinde kayıtlı 8 bitlik sayının, en yüksek seviyeli 4 bitini sıfırladıktan sonra 23500H adresli hafıza hücresinde bulunan 8 bitlik sayının tersi ile yer değiştiren 8086 komut kümelerini yazınız.



En yüksek seviyeli 4 biti sıfırlama kodu:

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

AND 0000 1111B (0FH)

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

0000 0101B (05H)

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

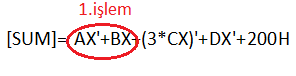
MOV [3500H],BL // 1.adım tamalandı

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.*

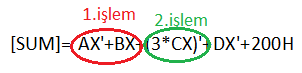
*1. işlem*



NOT AX

OR AX,BX // 1.işlem sonucu AX'dedir.

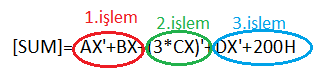
*2. işlem*



OR CX,03H

NOT CX // 2.işlem sonucu CX

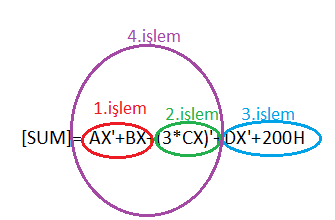
*3.işlem*

**

NOT DX

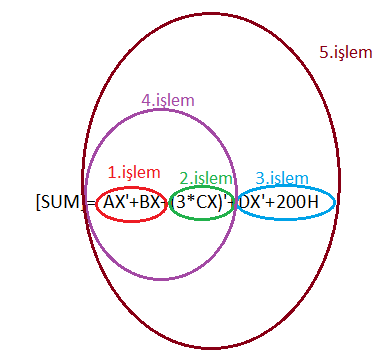
OR DX,200H // 3 işlem sonucu DX'de

*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.

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 artık AX de

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

NOT AX

MOV [8FFFH],AX // tüm sonuç 28FFFH da.