

MUL op1 1 tane operand alıyor

op1 = Byte

Ax  $\leftarrow$  AL  $\times$  op1

Mov [AL, 3]  $\times$  AL 'deki 3 kayboldu Burda AL gizli operand

Mov DL, 5

MUL DL; AL  $\times$  DL  $\rightarrow$  Ax

j Ax  $\leftarrow$  15

MUL OP  
word

$\underbrace{AX \times OP}_{16 \text{ bit}} \rightarrow \underbrace{Dx : Ax}_{32 \text{ bit}} \rightarrow$  Çarpım sonucunun  
düşük ondalı kismi  
Ax'te yüksek ondalı kismi  
Dx'te oluşuyor

Mov Dx, 17h

Mov Ax, 5  $\rightarrow$  Burda Ax gizli

Mov Bx, 2 operand

MUL Bx

$\rightarrow$  Burda Bx'in değeri bozulmuyor

MUL Ax, Bx  $\rightarrow$  Böyle bir operand yok

$Bx \rightarrow 2$   $Ax \rightarrow 10$   $Ax$ 'in değeri değişir  $Bx$ 'in  
değişmez

$Dx \rightarrow 0 \rightarrow Dx$ 'in  
değeri sıfır  
0 olur

$Dx$ 'in değerini  
korumak için  
`Mov Dx, 17h` dan  
sonra `push Dx`,  
Daha sonra `MUL Bx` ten  
sonra `POP Dx` yaparız

Görünmede  $Dx$   
 $\theta$ 'lanıyor bölmede  
ide  $Dx$ 'i bizim  
 $\theta$ 'lamamız gerekiyor

`Mov Ax, 0FFFFh`

`Mov Bx, 1000h`

$Bx, 00010000 0000 0000 b$

$Ax, 11111111 11111111 b$

$\hookrightarrow$  Bu işaretli sayıdır  
gündük biner korsis-  
tüşünde en ondalı  
biti 1'dir

`IMUL` ile `MUL`'un çalışma  
principi aynıdır

Mesela bir grupta bulunan  
genciklerin yaş ortalamasını  
olmak istiyorsun  
IMUL kullanamazsan Gündüz  
bir yaş zaten eksiz olamaz  
Sıcaklık derse IMUL kullanıcasın

DIY'de tek operand olsın

MUL 2 ; Böyle bir şey yapabiliyor muyuz

Ax  $\leftarrow$  AL \* 2  
→ DADDR ?

DIY op

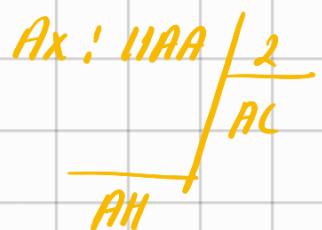
Byte → op  
16 bit  
Ax / op  
AL → 8 bit  
AH

Mov AH , 11h } Bunlar  
Mov AL , 0AAh } Bölkündükten

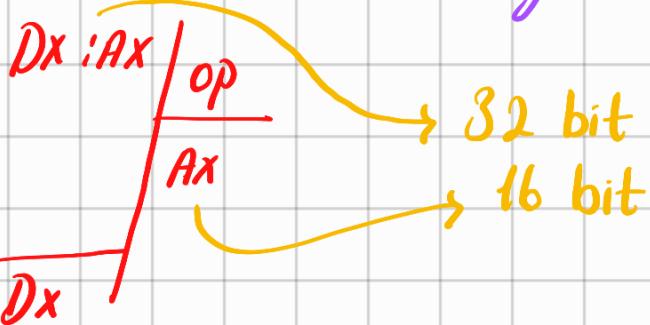
Mov CL, 2h

donro kayboldu

Div CL



Div op  
lн word  
1 byte 'tan  
büyük herhangi  
bir sayı word'tür



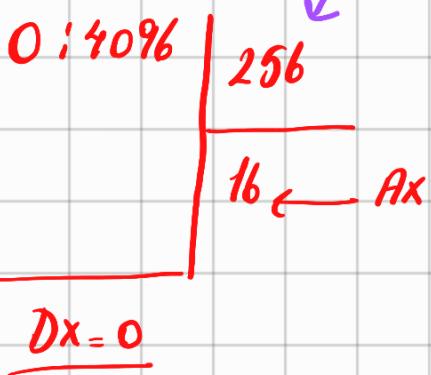
Mov CX, 0100h

δub Dx, Dx

Mov Ax, 1000h

Div CX

Bölmeden önce Dx'i kullanıcağı  
miz için Dx'i sı'lıdır



Jaxi → 1234ABCDh

high word  
DX AX

→ DX:AX

→ ABCDh

Mor AX, word Ptr [Sayı]

Mor DX, word Ptr [Sayı+2]

→ 1234h

IMUL ve IDIV mul ve DIV'a göre  
daha yavaş gelir o yüzden  
sayı işaretiz ise muhakkak  
DIV ya da MUL kullanıyo-  
ruz

## Dullanma Komutları

Programın akışını değiştiren komutlardır

\* Cmp komutu sonucu bayraklar değişiyordu  
bu bayraklarda oluşan sonucu göre  
gesitli döllenmeler gerçekleştiriyor

\* IP yarıncaının değerini değiştirirler

\* programınızın kodu kod segment  
denilen bellek alanının içerisinde

\* Koşullara bağlı reyo herhangi bir koşul  
olmadan döllenme gerçekleştirilebilir

\* Koşullu döllenme komutları genellikle  
CMP ile birlikte kullanılır

\* Koşuldu 2 döllenme komutlarında CMP'ye  
gerek olmuyor

\* Farklı isimlerdeki komutlar (mnemonic) aynı  
anlama gelebilmettedir.

→ MOV AX, 2

MOV BX, 2

CMP AX, BX

Ax - BX

② (E) → Burda JZ ya da JE kullanabiliriz  
ikisi de aynı birsey ifadesini ifade etmez

\* Koşullu döllenme komutlarının aralığı

8 bit ile sınırlıdır (128 byte peri veya 127 byte ileri 2 ip-  
lanabilir)

JXXX

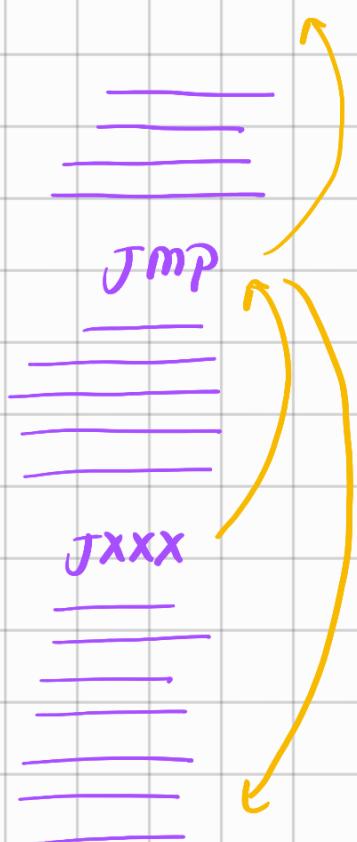
5 ← MOV      ——————  
6 ← ADD      ——————  
7 ← SUB      ——————

128 byte

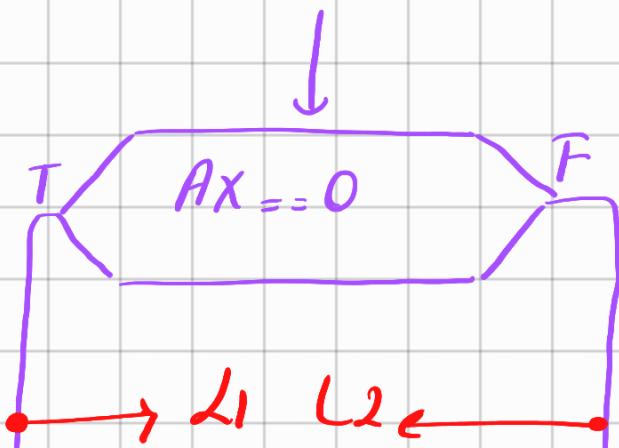
3 ← N      JXXX  
—  
—  
—  
—  
—  
—

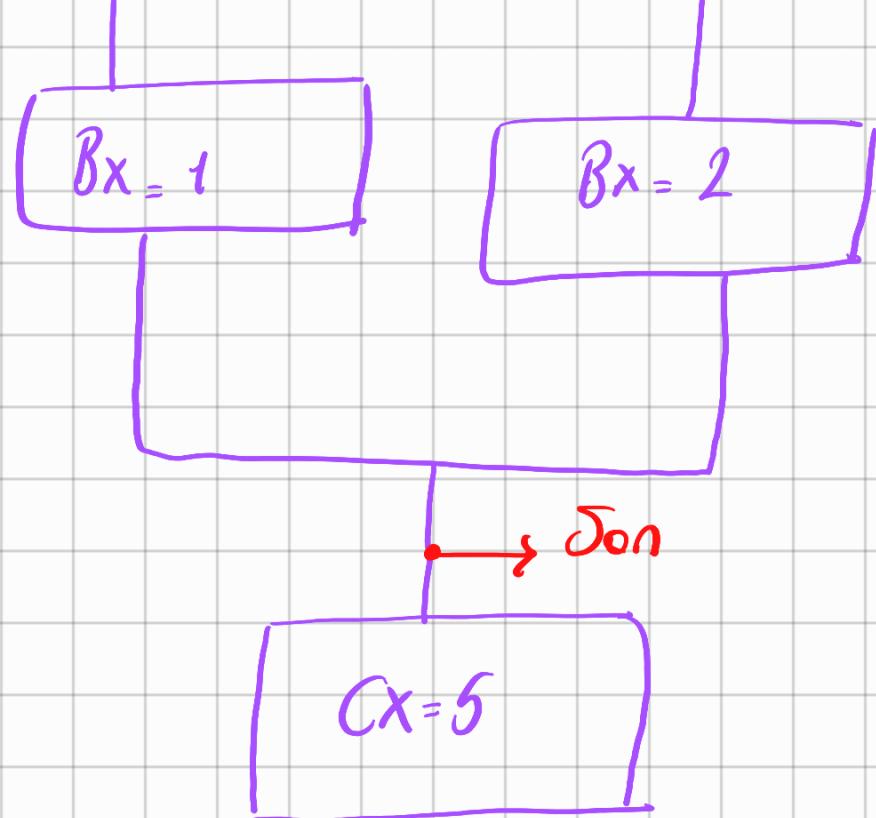
127 byte

eper daho yükset yerlere  
atlamak istiyorsam  
eper 128 byte ileri  
ya da periyede bir  
tane daho jmp  
komutu kaydırır



## Dallanma Örnekleri





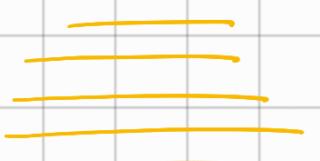
Cmp AX, 0

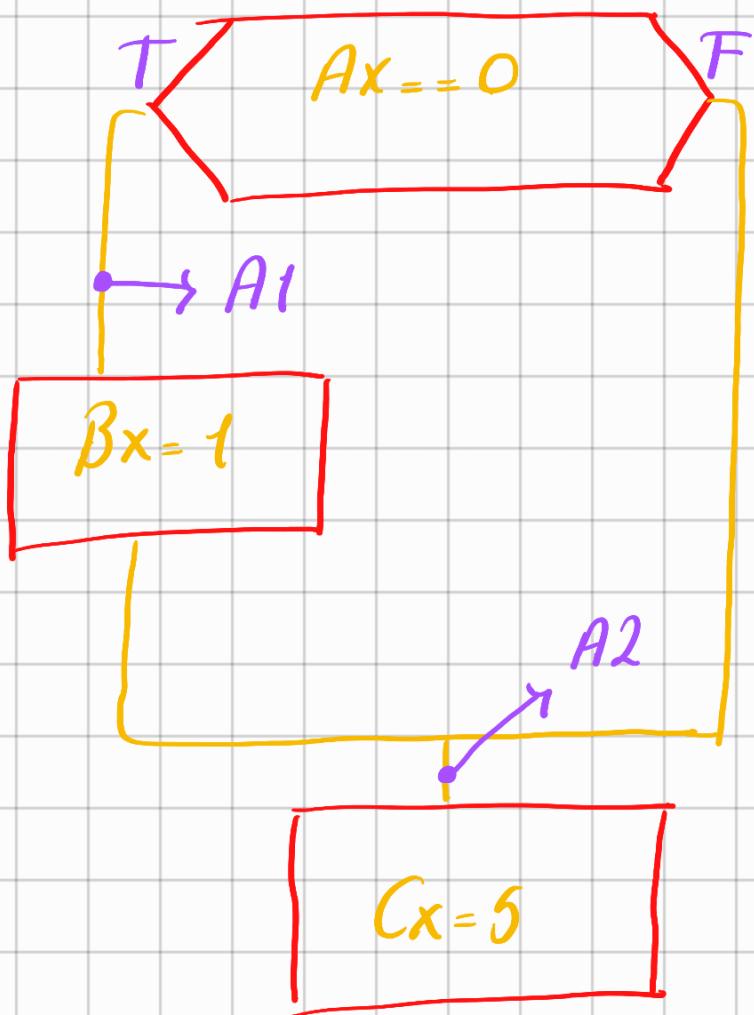
JF (JZ) L1 → eger esit degilse  
OLT satirdan  
Mov BX, 2 devam ediyor

Jmp JOn

L1 : Mov BX, 1

JOn : Mov CX, 5





*Cmp AX, 0*

*JE A1*

*Jmp A2*

*A1: Mov BX, 1*

*A2: Mov CX, 5*

---

*Cmp AX, 0*

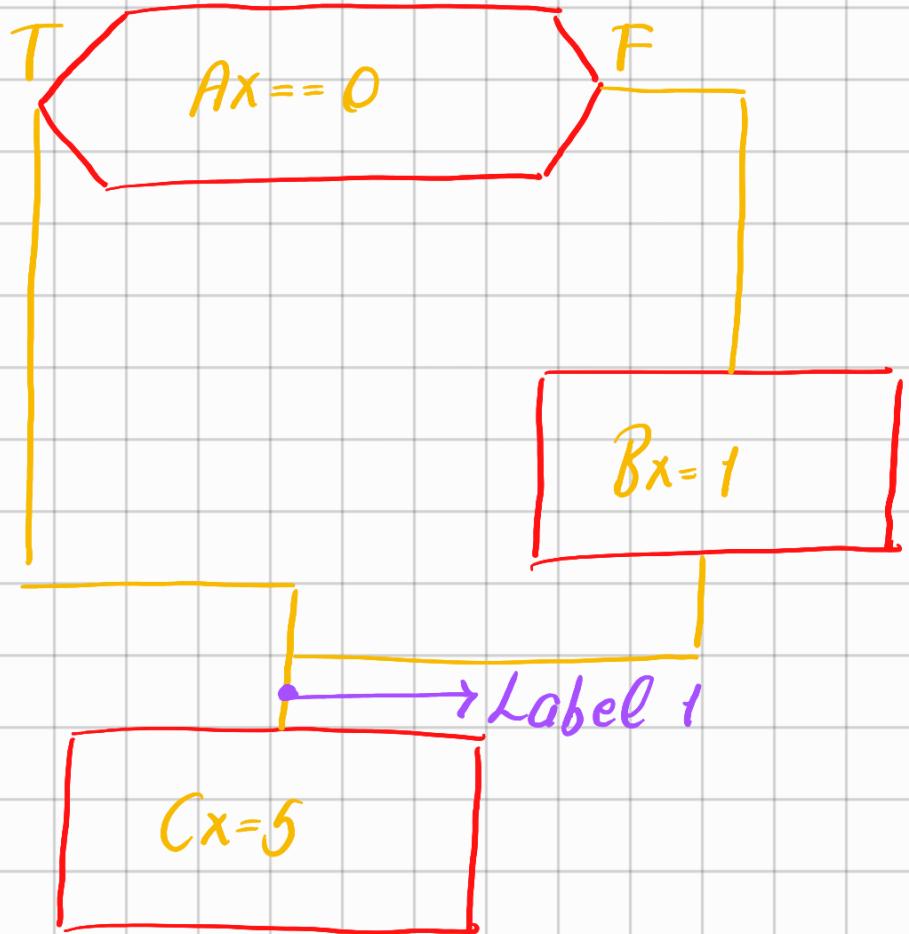
*JNE A2*

*Bu yukarıdaki koddan daha efektif*

Mov BX, 1

A2 : mov CX, 5

→ Kodu penellikle  
bos kolo göre  
y020r12



Cmp AX, 0

JE Label 1

Mov BX, 1

Label1 : mov CX, 5

\* Asla yorumlamamız gereken  
sey

CMP AX, 0

JE L1

JNE L2

X

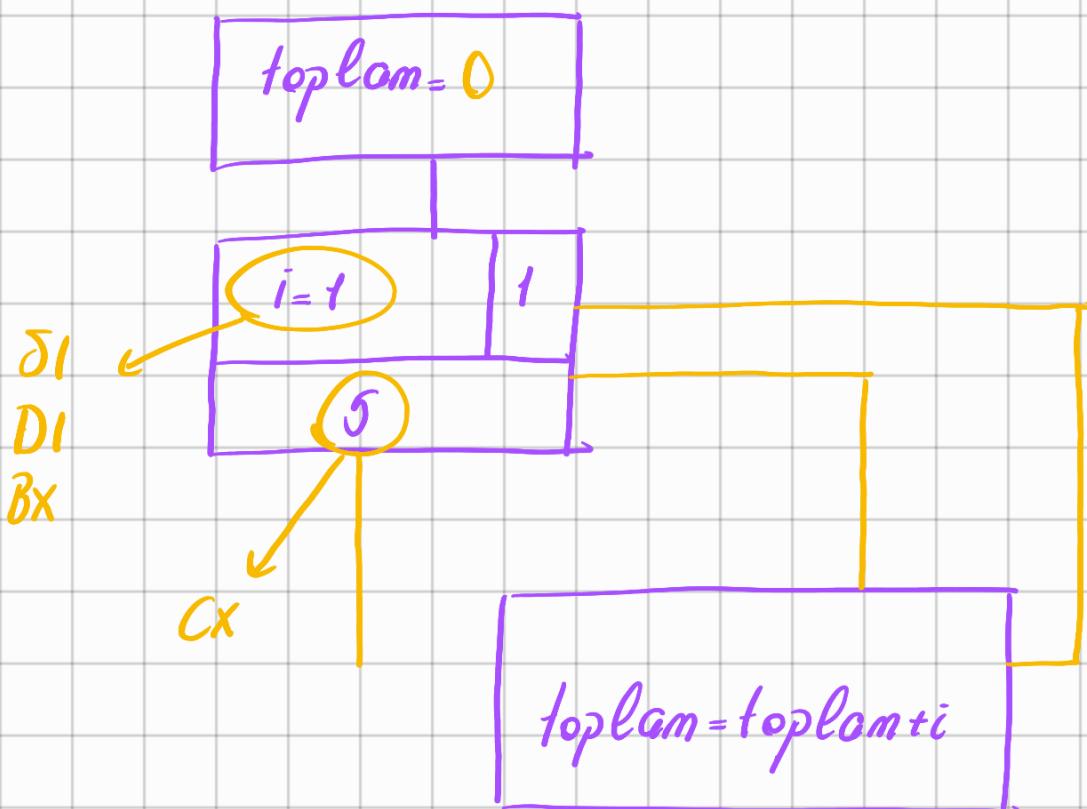
## Loop (FOR)

Cx bunun üzerinde dönüçet

Cx'in deperi 10 iğe 10 kere

dönüçet 20 iğe 20 kere

dönüçet



Mov toplam, 0

Mov Ax, 5

Mov S1, 1

L1: ADD toplam, S1

INC S1

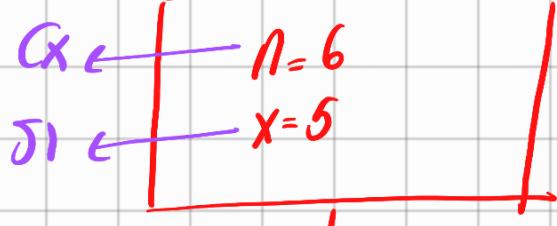
loop L1

→ Ax'in deperi 0 ise  
asapidon devam  
etmeye baslıyor

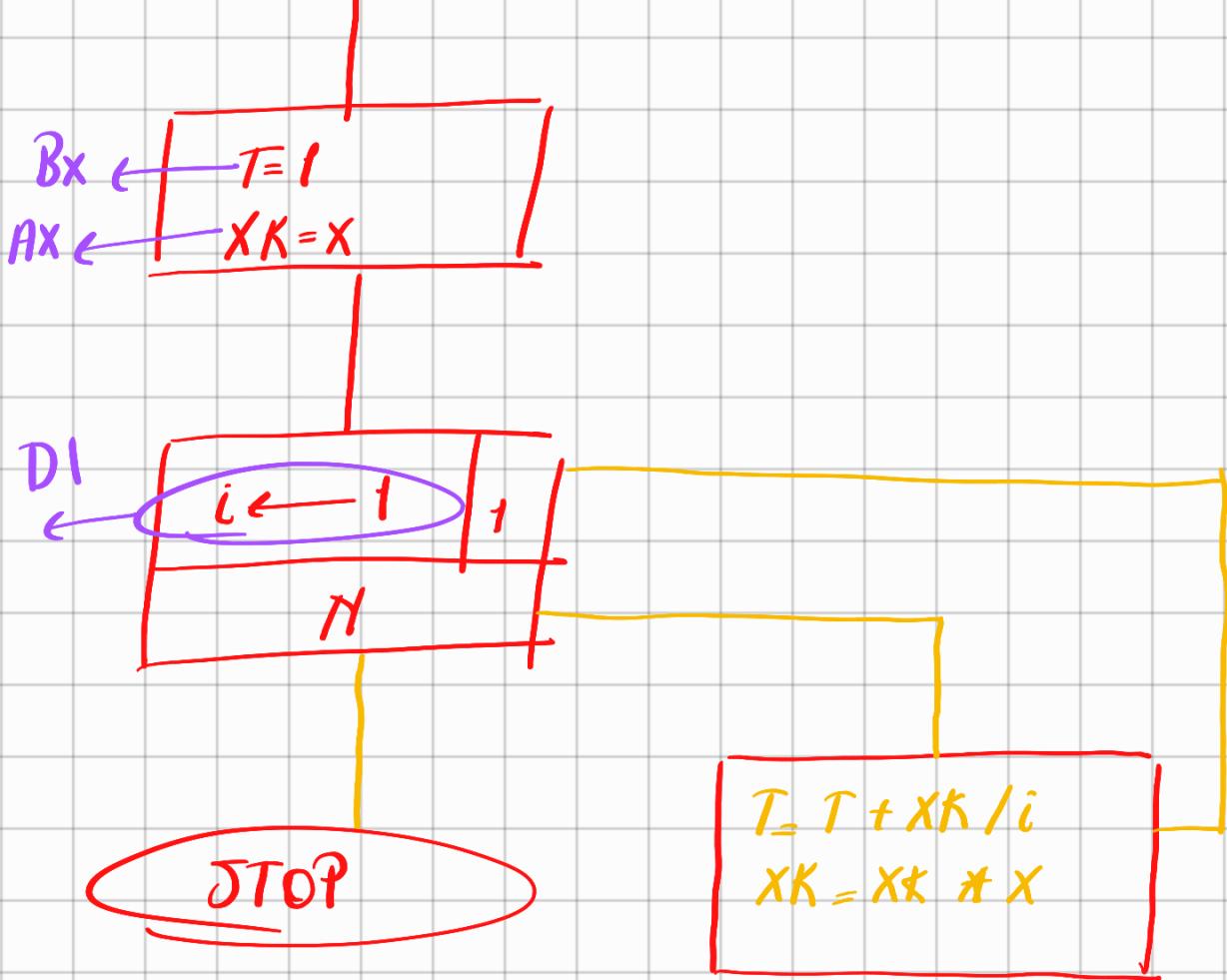
$x = [-5], n = [1-6]$  olmak üzere

$$T = 1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \dots - \frac{x^n}{n}$$

T = ?  
o



1 byte en fazla 255  
2 byte en fazla 65535 değerini  
olar



Mov Cx, 6 ; Döngü sayısını burda tutuyoruz

Mov SI, 5 ; X deplikenini burda tutuyoruz

Mov BX, 1 ; toplamı burda tutuyoruz

Mov AX, SI ; AX registerine X değerini atıyoruz

Mov DX, 0 ; bölme yaparken DX'i kullanırız  
kullanmak için bıçımı 8 set ederiz

L1: Push AX

DIV DI

ADD BX AX

Bölme yaparken AX deplicesi için  
Burda AX'in değerini korumak için  
Stack'e oturuz

POP AX

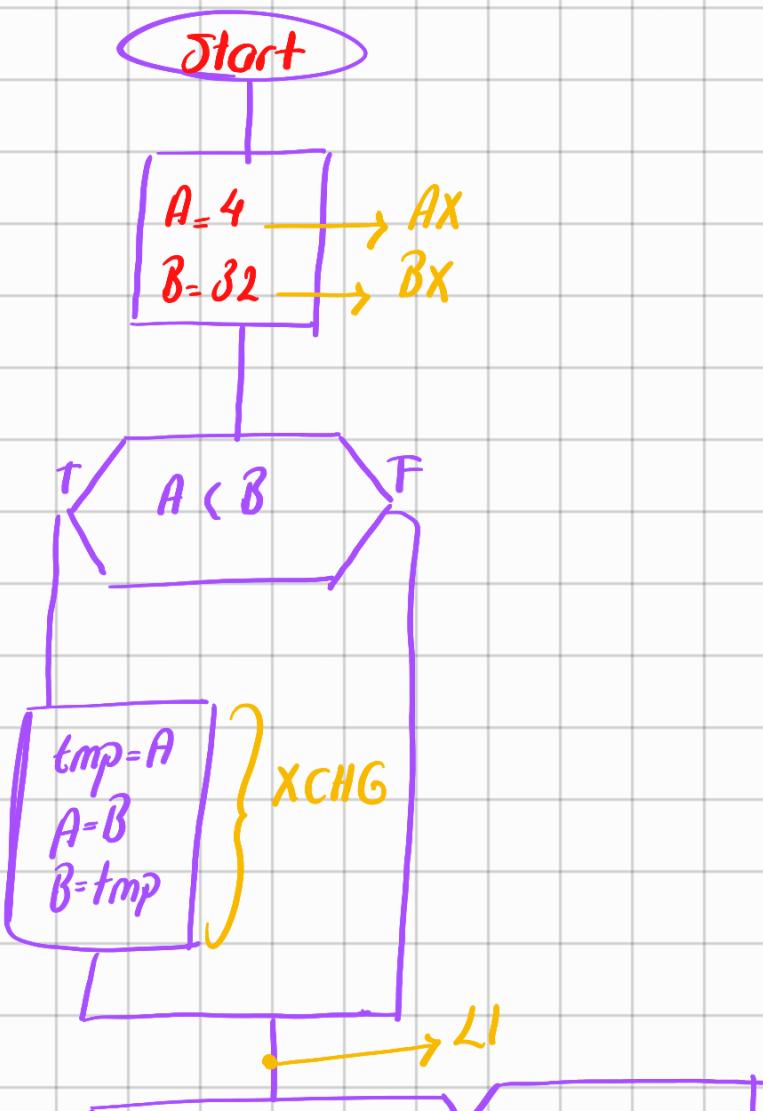
MUL DI  $\rightarrow$  *Görmeden dolayı buuya 0 set ediyor*  
 $DX:AX \rightarrow DI * AX$

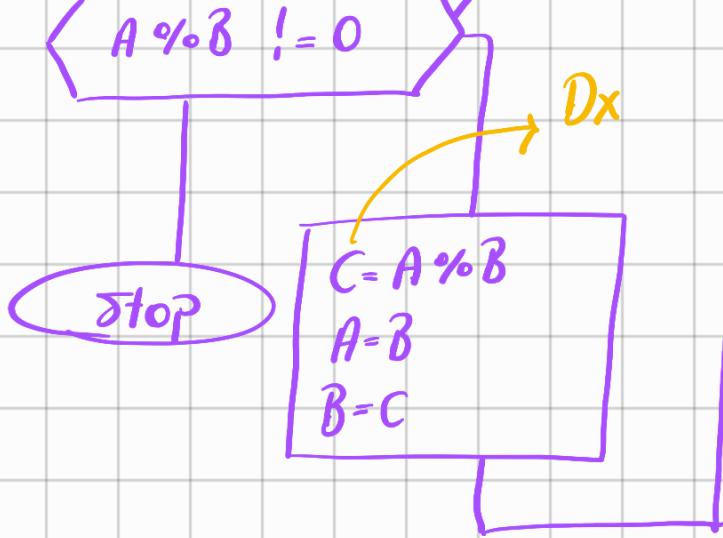
INC DI *j indexi 1 arttıyoruz*

LOOP LI

JORU

$A \text{ ve } B = [0-500] \rightarrow EB08(A,B) = ?$





Mov AX, 4

Mov BX, 32

Cmp AX, BX

JAE L1

XCHG AX, BX

L1: Mov DX, 0

Div BX

Cmp DX, 0

JF L2

Mov AX, BX

Mov BX, DX

Jmp L1

L2: Sonuc

BX'te

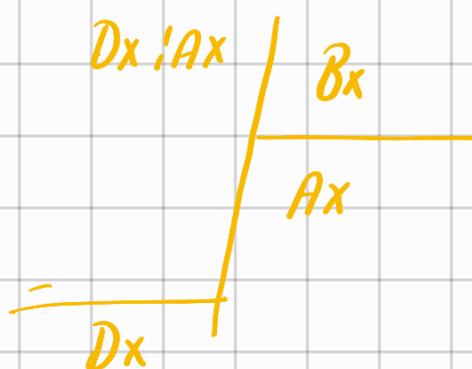
→ Sonuc BX'te  
olur

Bölme yapıcıımız için Dx'i 0'ladık

Böyle işlemlerde  
ilk operand AL veya

AX olsun BX olmasın

Güntü daha hızlı olacaktır



9 tane Bayrakımız var  
bu bayratları biz iste-  
dipimiz gibi depostire  
biliyoruz clear ede-  
biliyoruz set edebili-  
yorum

*yorum*

CLC clear CF carry flag'i 0'lar

CMC Complement CF 0'sa 1'ise 0

STC set CF carry flag'i 1 yapar

CLD Clear DF

STD Set DF

STI Set IF

CLI Clear IF

LAHF Load AH with flag Bayraklarımızın değerlerini AH'in  
igue koyan flag

SAHF Store AH in flag AH'in içindeki bayrak değerlerini  
git flagin igue yaz

## *Mantıksal Komutlar*

NOT

Mov AX, 00FFh

NOT AX; AX ← OFFFOOh

↓  
FFFOOh

OR

AND

XOR

Operand 1 ve operand 2'nin orunu  
oluyor sonuc operand 1'de  
oluşuyor

Mov AL, 12h  
OR AL, 80h

Mov AX, 0      } Bunlar aynı  
Sub AX, AX      } işlemi yapıyor  
XOR AX, AX      } fakat biri diperi-  
                    } ne göre daha  
                    } 02 yer kaplıyor  
                    } ve daha hızlı  
Bu daha      } hızla çalışıyor  
hızla      } çalışıyor

TEST op1, op2 → Test komutu op1'in değerini  
değiştirmez

Flags ← PSW ← op1 ⊕ op2 → op1'in değerini  
And op1, op2      } korumak isti-  
                    } yorsak bu  
                    } işlemi yaparız

Op1 ← op1 ⊕ op2

