

Adresleme Kipleri

- * Anlık (immediate) adresleme
- * yazmaç (register) Adresleme
- * Dogrudan (Direct) adresleme
- * yazmaç dolaylı (Register indirect) adresleme
- * Baz görelî (Base Relative) adresleme
- * Dogrudan indisli (Direct Index) Adresleme
- * Dizi (string) Adresleme
- * iskele (Port) adresleme

Anlık (immediate) adresleme

mov reg, idata

Mov mem, idata

Mov AL, 16

Mov CX, 0ABCDH

adresleme denilince otla belleğin içerişine birşeyler koyma, belleğin içerişine yeri koyma registerin içerişine yeri koyma bu şekilde düşünübilirsiniz

Yeri Atma Yeri Otturma bu şekilde düşünübilirsiniz

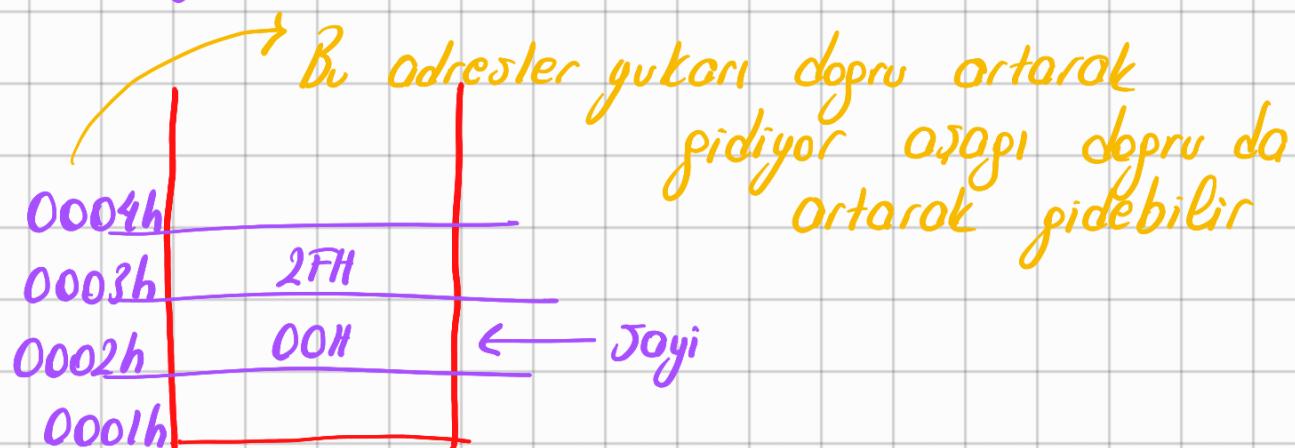
Yazmaq (Register adresleme)

Mov rep, rep

Mov DS, AX

Mov DL, CH

Dogrudan (Direct) adresleme



Jöyi → word ise

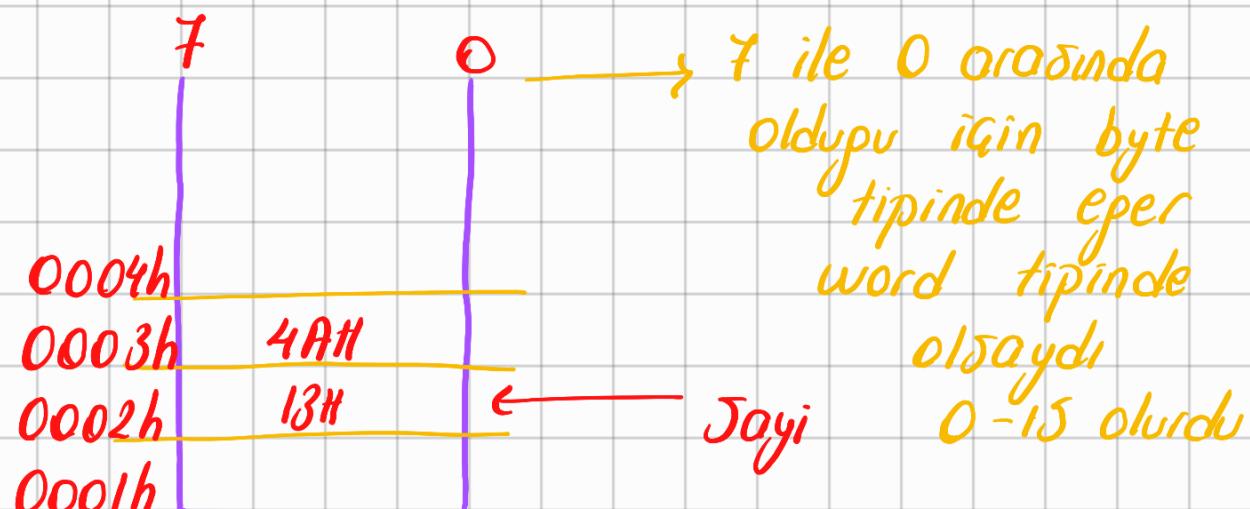
$Mov Ax, sayi \Rightarrow Ax : 2FOOh$

$sayi \rightarrow$ Byte ise

$Mov AL, sayi \Rightarrow AL : 00h$

Yazmaç Dolaylı (Register Indirect) adresleme

yani bizim registerlerimiz var ve biz bu register'lar üzerinden bir değerleri adresliyor oluyoruz



$LEA SI, sayi \Rightarrow SI : 0002h$

$Mov Ax, [SI] \Rightarrow Ax : 4A13h$

$Mov DI, OFFSET Jayi \Rightarrow DI : 0003h$

$Mov Ax, [DI] \Rightarrow Ax : 4A14h$

OFFSET Kodu mnemonic değil pseudo kodlardan derleyicide bir anlam ifade ediyor

Base göreli (Base Relative) adresleme

7	0
59h	0018h
4Ch	0017h
42h	0016h
4Dh	0015h
45h	0014h
53h	0018h
53h	0012h
41h	0011h
	0010h

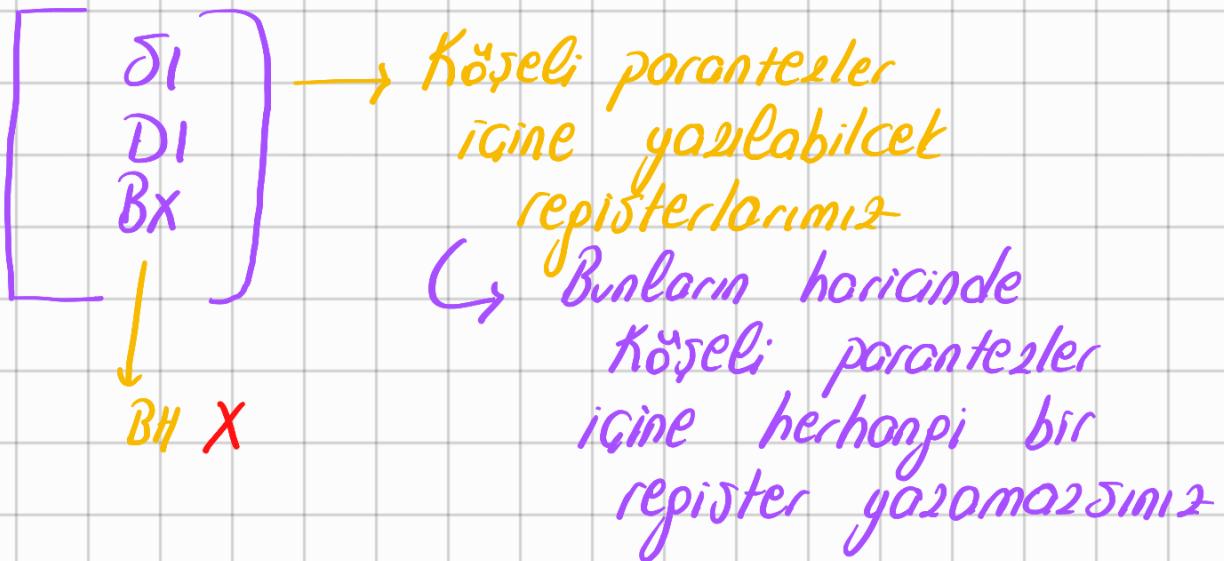
$m\sigma p$

$m\sigma p db 'ASSEMBLY'$

$LEA Bx, m\sigma p \Rightarrow Bx : 0010h$

$Mov AL, [Bx+4] \Rightarrow AL : 4Dh$

$Mov Ax, [Bx+5] \Rightarrow Ax : 4C42h$



Doprudan indisli (Direct Index) adresleme

yüksel servisli dillerdeki dizi erişimlerine benziyor

Dizilerin tipi çok önemli
 eğer dizi byte tanımıysa 1 arttı-
 rırsın dizi word tanımıysa 2
 arttırsın dizi double word
 tanımıysa 4 byte 4 byte arttırsın

7	0
59h	0018h
4Ch	0017h
42h	0016h
40h	0015h
48h	0014h
53h	0013h
53h	0012h
41h	0011h
	0010h

dizi DB 41h, 53h, 58h, 45h, 40h, 42h, 4ch, 59h

XOR SI, SI

Mov AL, dizi[SI] \Rightarrow AL: ? 41h

INC SI

Mov AL, dizi[SI] \Rightarrow AL: ? 53h

dizi2 DW 5341h, 4553h, 424h, 594Ch

XOR SI, SI

Mov AX, dizi2[SI] \Rightarrow AX: ? 5341h

INC SI

Mov AX, dizi2[SI] \Rightarrow AX: ? 5353h

ÖRnek 1

2 dizinin yerini farklı adreslemeyle depostirelim

d1, d2 : byte tanımlı

n: dizilerin boyutu

Boyle bir sey gördüğümüzde okumizo bunu CX icine atmak gelmeli

Mov CX, n

XOR SI, SI \rightarrow SI = 0 (Burdaki değer bir soyi)

LEA DI, d1 \rightarrow DI \rightarrow 1FA2h

(Burdaki değer ise bir adresdir)

don: Mov AL, d1[SI]

XCHG AL, [DI]

Mov d1[SI], AL

X XCHG [DI], di[SI]
X Xchq DADDR, DADDR

INC DI
INC SI
Loop don

XOR SI, SI
MOV AL, [SI] X

LEA AX, di ✓

LEA DI, dici
MOV AL, dici[DI] X

Örnek 2

BYTE
0-255

1962 > 255 o yüzden WORD
2001

A şirketinde çalışanların doğum yılı di dizisinde, B şirketinde çalışanların yaşıları d2 dizisinde tutulmaktadır.

Önce A şirketinde çalışan ol kişinin, sonra B şirketinde çalışan b2 kişinin bu bilgileri boyutu ol + bi olan d3 dizisine aktarmak için gerekli ASM kodunu yazalım

LEA DI, d1
LEA DJ, d2
XOR BX, BX

Mov DADDR, DADDR
Mov d3[Bx], [DI]

Mov CX, 01
L1: mov AX, [DI] } Burda di dizisi var
mov d3[Bx], AX }
→ Burda d3 dizisi
ADD DI, 2 ; word olduğu var
ADD BX, 2 ; iki 2'ser 2'ser arttırırız
LOOP L1

Mov CX, b1
L2: mov AL, [SI]
CBW
mov d3[Bx], AX
ADD BX, 2
INC SI
LOOP L2

Sözdə (Pseudo) Komutlar

Bunlar derleyiciye neyin ne şekilde yapılmasını söyleyen komutlar

Matine kodu karsılığı yoktur
Object kod karsılıkları yoktur
Bayraklar üzerinde etkisi yoktur

Debug ortamında kullanılmıyorlar

IST dosya düzeni

Program keşir düşesi

Program kesim ismini

Yeri tanımlamaları için kullanılır.

Derleyici tarafından anlamlandırılırlar.

Sözdə komutlar Segment / Ends

Segment tanımından kullanıyoruz
bu komutları

→ Segmenti açma
parantezi

Kesim_ismi Segment { Segenekler }



Kesim_ismi

Ends

→ Segmenti
Kapatma
parantezi

Segenekler

a) hizolama (Alignment) Tipi

tanımlanır keşimlerin hangi
sınırından başlayacağını belirler

BYTE 11

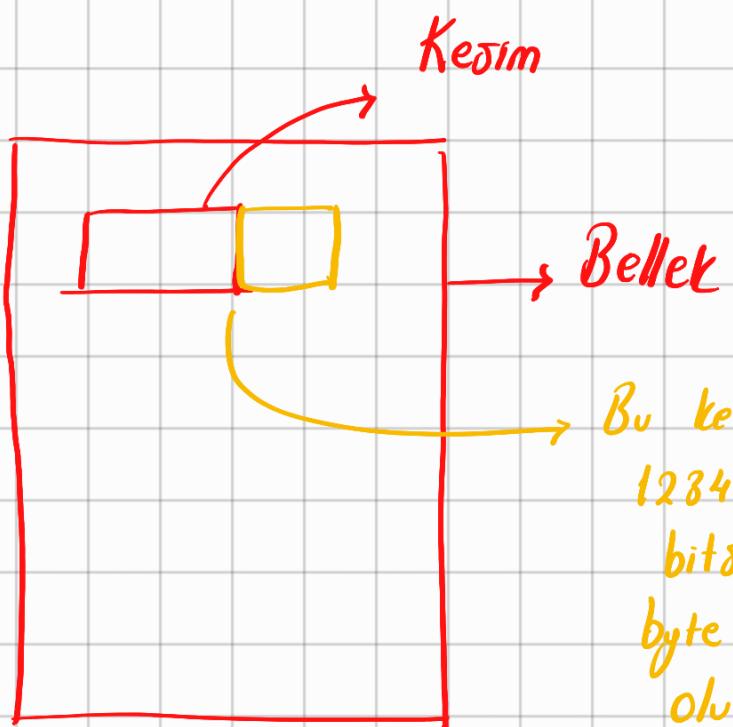
WORD 12

PARA 116 → Mesela 16'ya bölünebilen
adreslere yerlesir

PAGE 1256

Segmentlerin tamami bellepin icerisinde
yer aliyor

Data segment re Code Segment 'de
mesela bellepin icerisinde



→ Bir sayinin 0'a
bolunmesi icin
h karsiliginin
sonu 0'la
bitmeli 256'ile
birlikte

1284 h adresiyle
bitdin eger
byte ide 1285h
olur word ide
1286 h olur Para
ide 1240h olur
Page ide 1300h
olur yani o adresden
buyuk o sayiya
bolunebilen ilk

tom bölünebilmesi
için sonu 00h
ile bitmeli

adres

b-) Birleştirme (Combine) tipi

Link işleminden sonra aynı içim
verilmiş kesimlerin birleştirilmesi
için kullanılır

PUBLIC aynı içimli kesimler birbirinin
devamı olur

COMMON kesimler birbirinin üzerine
yazılır

STACK LIFO sisteminde gelir

AT #####; Belirlenen fiziksel adresen
başlar

(fiziksel adreslerimiz 8086'da 20 bitti)

Mesela biz kodumuzu yazarken 5 farklı
kod doğasından yazdık eger bunların kesim-
leri aynı birleştirme tipinde verilisse
sayet bunları birleştirebiliyor sunuz

Bir segment maximum 64K'lık bir boyutta
olabiliyor

mesela bir kod yazardın 256 B çıktı bi da
yazardın 112B çıktı sen bunları birleştirir-
sen tek bir 64k'lik yere de sokabilişsin

C-1 Junif (class) tipi

Birleştirme tipi Public olarak tanımlanmış
keşimlerin link işlemi sırasında birbirinin
peşi sıra gelmesi için kullanılır

Tırnak içinde gelir

Birlestirmeye karar verdiğimizi düşünelim

Class adını istediğimizi gibi verebilirsiniz

Burayı
isteğim
gibi isimlendire
bilirim.
Junif(class)
tipi

Sintaks örneği

Stack

SEGMENT PARA STACK 'yipin'

Ends

Stack

Anahtar sözcüğü

Combine tipi
olarak stack
kelimesini
kullanmışım

'olignment', paragraf
16'ya tam bölü-
nebilen bir sayidan
başladığını bu segmen-
tin görünüyorum

Bu işi onla-

Code39 SEGMENT PARA 'cod'

ma geliyor stock
bunun first in
look out set-
linde GOLIS-
masını yapıly-
cak

Code39 ENDS

↳ Combine tipi kullanmadığımız
pördük bunu kullanmadı-
mızda bunun stock olmadığını
ve standart bir segment tanımı
yapıldığını poraproftan bıslayın
bir adresten itibaren yerleştiril-
dığını görünüz

data39 SEGMENT PARA 'veri'

data39 ENDS

test SEGMENT BYTE AT 0120AH '+'

test ENDS

Sözde Komutlar ASSUME, DB, DW, DD, DQ, DT

Bizim tanımladığımız segmentler var
bu segmentlerin hangisinin hangi
segment olduğunu söyleyen
ifade ASSUME'dır

Bunlar EXE
tipi progra-
ram için

ASSUME SS:stack\$P DS:data\$P CS:code\$P geçerli

Segment atamaları için kullanılır

COM tipinde ise

ASSUME SS:my\$P, CS:my\$P, DS:my\$P

exe tipinde 3 tanesi segment var CS, DS, SS

Com tipinde ise tek bir segment var

exe'nin uztatılmış hali executable

com'un uztatılmış hali compact

ufak küçük bir program yaziyorsanız
com'u kullanarak bu işi gerçekles-
tirebilirsiniz

Assume komutu şu işe yarıyor sizin tanı-
madığınız segmentlerin isimleri ile segmentleri

eslestirmek icin kullaniliyor

DB (Define Byte) \rightarrow int sayi = 25;

sayi1 DB 25

sayi2 DB 1EH

sayi3 DB ?

sayi4 DB 1, 2, 3Fh, 0A4h, 5

strb DB 'Assembly dersi'

int sayi1 = 25;

? Koymak ilk degerinin ne oldugunu
öncemi yok

int sayi3;

Bu komut data segment tarafindan
tanimlayipimiz degistirlerin tiplerinin
byte tipinde olmasini sagliyor

Byte tipinde
bir dizisi
tanımlamak
icin böyle
yapariz

String kelimenin
her elemanı byte
tipinde olacak
şekilde yine
bir diziyi ifade
etmiş oluruz

Dw (Define WORD)

Bu basindaki 0 sadece

sayi1 Dw 0ABCDh

posta icin mesela ABCDh olsa

bu variable olabilir bunun hexa decimal

bir sey olabilmesi icin önde

F

0

0-F Arası Ø koymuyoruz ki bunun

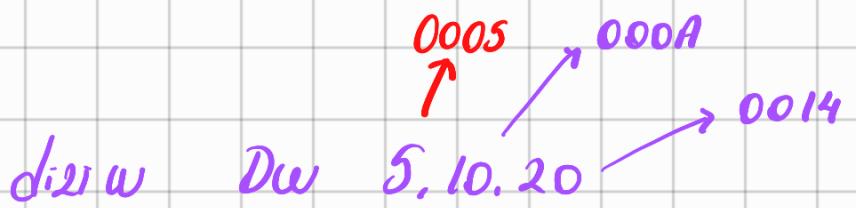
olması byte bir sayi oldugu

tipinde Oldugunu gunu onlara

üstesine A ile yaramaz

Yazılıyor 0-15 yabilirim
oldaydı bu word tipinde
olduğunu gösterirdi

0250h	
0251h	?
0252h	CD
0258h	AB



0518h	
0517h	
0516h	00
0515h	14
0514h	00
0518h	0A
0512h	00
0511h	05
0510h	

→ yüksek onlamlı kısmı buraya geldi
 → düşük onlamlı kısmı buraya
 gelirken
 dizi w

DD (Define Double Word) → wordun 2 katı (4 byte)

DQ (Define Quadro Word) → wordun 4 katı (8 byte)

DT (Define Ten bytes) → (10 byte)

Sözdə Komutlar DUP, PTR
dizinin tipi

DUP (Duplication factor)

diziB2 DB 15 DUP()

Bizim oslində dizileri yüksətən onlamlı tanımlamak tərif içün kullanıdığımız pseudo kodlardan

byte tipinde her bir elemeni
0 olan 15 elemənlə bir
dizi tanımlıyorum bu
dizinin adı diziB2

Mb1 DB 3 DUP(4 DUP(8))
3x4 matris(8)

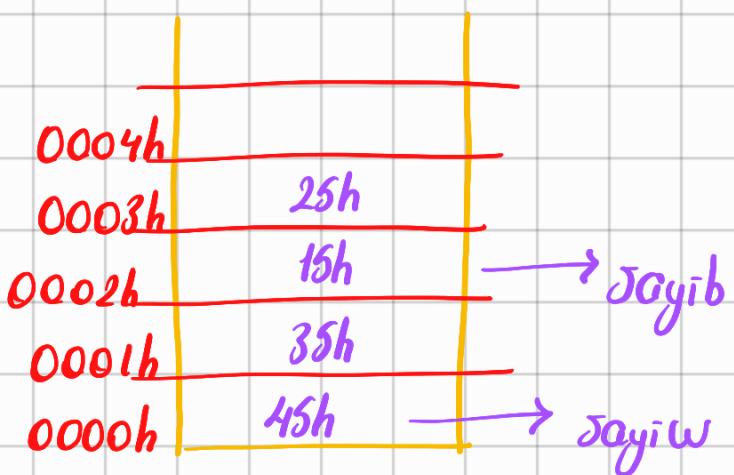
diziW DW 10 DUP(?) → 20 byte

her bir elemeni word
tanımlı olan 10 tane
eleməndən oluşan ilk
değerlerinin benim içim
önemi olmayan bir
dizi

PTR (pointer)

tip dönüştümü (casting)

Sayıb DB 15h
Sayıw DW 3545h



Mov AX, WORD PTR Sayıb \Rightarrow Ax : ? 2615h

Mov AH, BYTE PTR Sayıw \Rightarrow AH : ? 45h

Mov AL, BYTE PTR Sayıw+1 \Rightarrow AL : ? 35h

Sözdə Komutlar - Label

(Assembly dili
case sensetive
bir dil deşil)

0012h				Labels
0013h	35h	→ Jayiw ← yeniB	Byte tipinde tanımladığımız için label 35h'i gösterir	(Byte tipinde tanımladığımız için label 35h'i gösterir)
0014h	25h			
0015h	45h	→ JayiBb ← yeniW		
0016h	55h	→ JayiBC		

Bir de isteklerimiz var bir de etiketlerimiz var etiketlere burda Label ismi veriyoruz

✓ YeniB Label BYTE → tanımlandığı
35h Jayiw Dw 2535h yerden sonraki
✓ YeniW LABEL WORD 1 bytelik yeri
5545h JayiBb Db 45h olur

Jayı BC DB 55h → tanımlan

dığı yerden
sonraki 1 bytelik
değeri olur

Mov Ax, YeniW

Ax : 5545h

Mov BL, YeniB

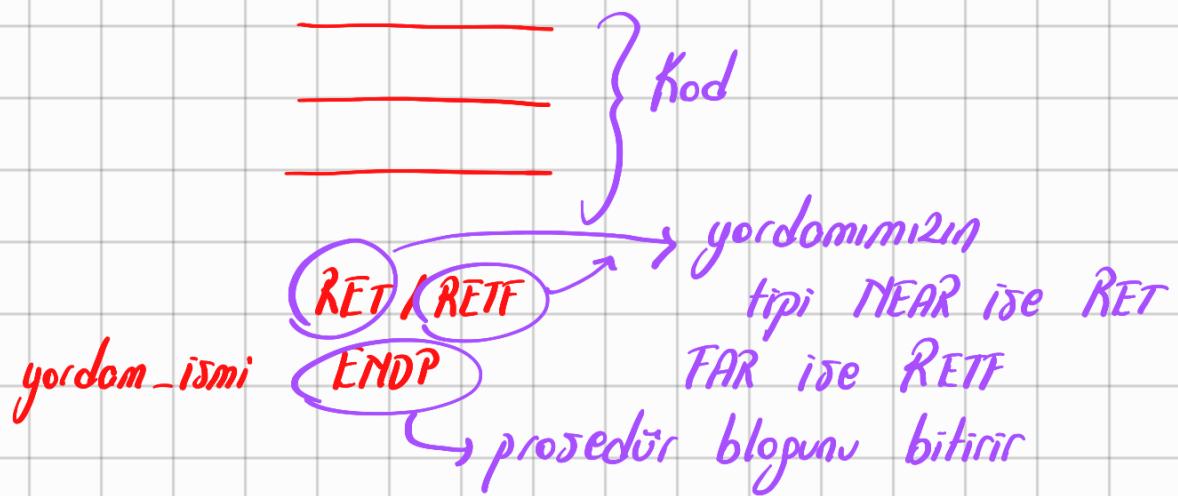
BL ← 35h

Sözdə komutlar PROC / EndP, SEG, END

Proc (Procedure) / EndP

→ yordamımızın
tipi NEAR yado

yordam_ismi PROC {NEAR,FAR} FAR olabilir



prosedürün türkçe karşılığı yordam'dır

OFFSET

LEA aynı işler

Mov Ax, OFFSET Tablo

Ax: ?
0713h

7	0712h	
	0713h	3ch
	0714h	FFh
	0715h	89h

→ Tablo

END

END baslangic_noktasi

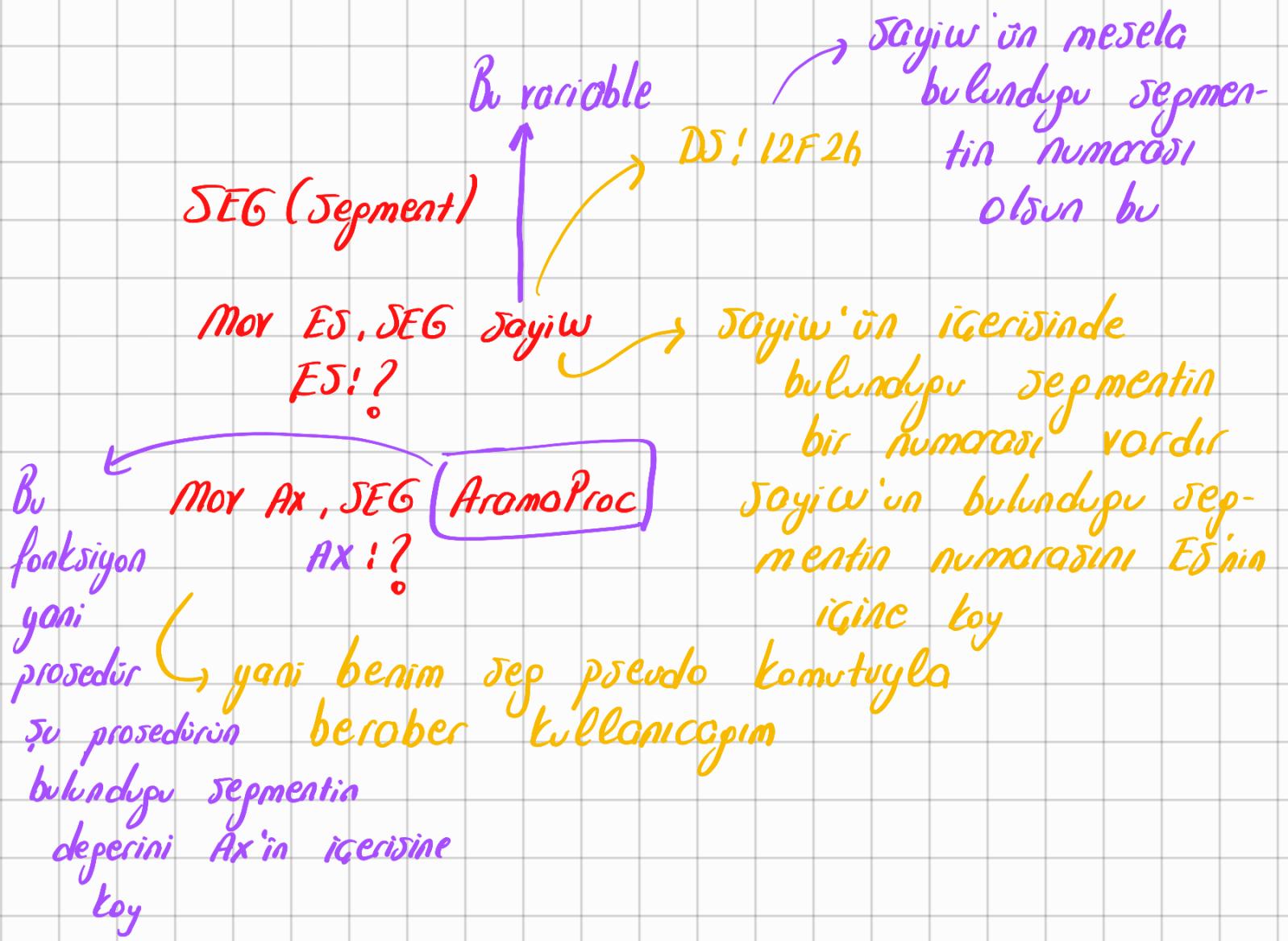
(

END Komutu ise program
ortak birçok farklı yordam-
dan birçok farklı fonksiyondan
oluştugunu düşünün farklı farklı
fonsiyonlar var her fonsiyon

komutunuz için ne seg. olsun
komutunuz programın en sonunda
do END yazıyorsunuz ve bu program
diyelim ki 5 farklı modulenden oluşuyor
diğer burada hangi modulenden başla-
mamı istiyorsanız buraya o prosedürün
adını yazıyorsunuz



END proc-name



Sözde Komutlar - Length, TYPE, SIZE

LENGTH

Dup'ten önceki sayının Jana
ne olduğu Jana döndürüyor

Tablo Dw 15 Dup(?)
Mır Ax, LENGTH Tablo
Ax : ? 15 → yani 000Fh

TYPE

depisten tipinin tag BYTE
oldugunu bulmak icin
sizeof()

Tablo DW 15 Dup(0)
Mır Ax, TYPE Tablo
Ax : ? 0002h
tipini döndürmüşt oluyor

Sayı DD 3412ABCDh

Mır Ax, TYPE Sayı
Ax : ?
→ 0004h

SİZE

Length * type
 $2 * 20 \rightarrow 40$
Tablo DW (20) Dup(?)

Mov AX, SIZE Tablo

AX : ? h
0028 h

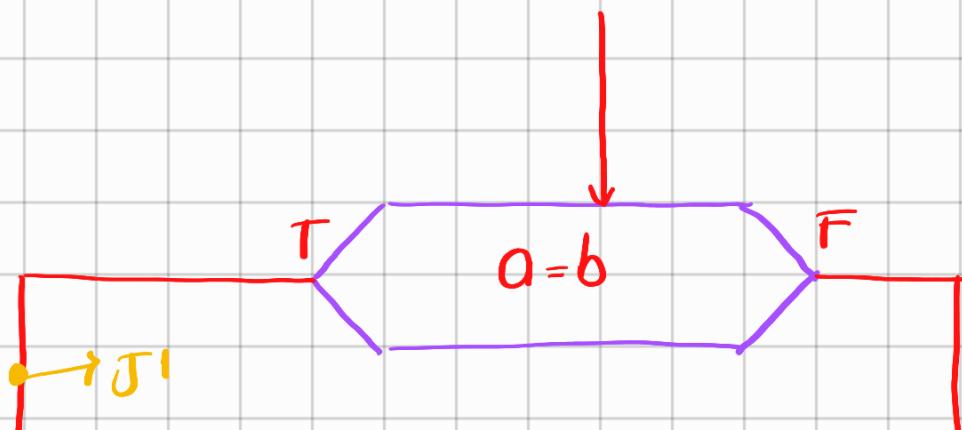
ÖR

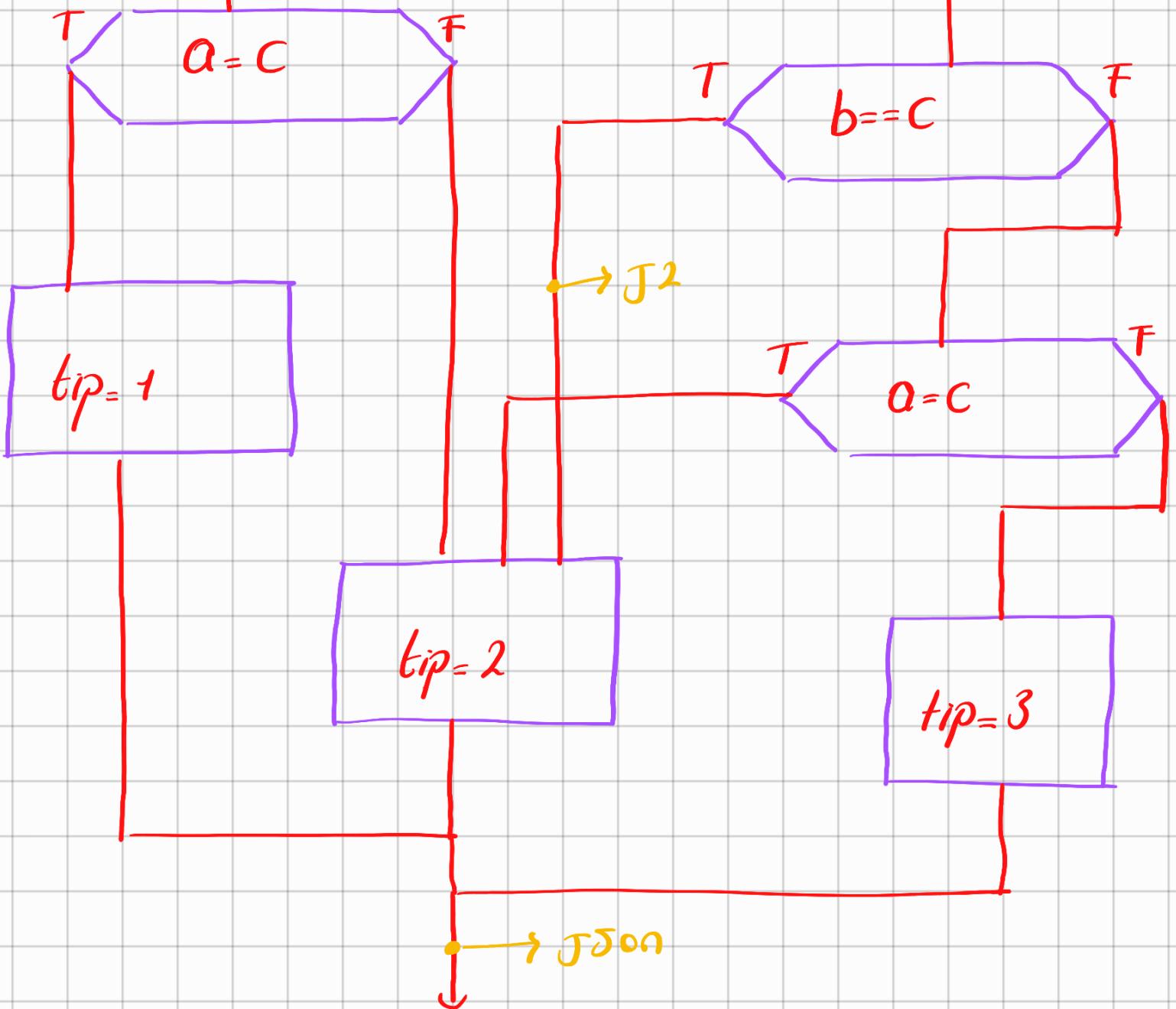
3 Kenar bilinen üçgenin tipini bulunuz

tip1 = 1 \Rightarrow eştkenar

tip2 = 2 \Rightarrow ikizkenar

tip3 \Rightarrow Gesitkenar





DS {

dosasp	SEGMENT PARA 'veri'
tip	D8
a	DB 12
b	DB 13
c	DB 12
dosasp	ENDS

Stacksp SEGMENT PARA STACK 's'

DW 15 DUP(?)

Stack ENDS

code\$P SEGMENT PARA 'kos'

ASSUME CS:code\$P DS:datos\$P SS:stack\$P

ANA PROC FAR

Push DS

XOR AX, AX

Push Ax

Mov Ax, datos\$P

Mov DS, Ax

Mov AL, A j 1.kenar

Mov BL, B j 2.kenar

Mov CL, C j 3.kenar

CMP AL, BL

JE J1

CMP BL, CL

JE J2

CMP AL, CL

JE J2

Mov tip, 3

Jmp JSON

J1: CMP AL, CL

JNE J2

Mov tip1

Jmp JSON

J2: Mov tip, 2

JSON: RETF

ANA ENDP
codesg ENDS
END ANA



