

parametre aktarma yöntemleri

3 adet yöntem var

- yaz瑪G üzerinden parametre aktarımı
- yığın (stack) üzerinden parametre aktarımı
- EXTRN / PUBLIC komutları ile parametre aktarımı

her yordamın değil programımızın bir tane stack'i oluyor

Sonki yüksek seviyeli dillerde global değişken tanımlamak gibi extrn / public komutları sayesinde data segment içindeki dataları diğer yordamların kullanabilmesine için vericeğimiz bir yelpiz pörücez

Örnek-1 Stack üzerinden parametre aktarma

Ardışık 3 elemanı üçgen kenarı olan bir dizide n tane üçgenin kenarları tutulmaktadır. Üçgen kenarını stack üzerinden olan ve bu üçgenin olonının karesini Ax register'ı üzerinden döndüren bir horici yordam yordımıyla en büyük üçgenin olonının karesini bulan Exe tipi ASM programını yazınız.

1.üçgen 2.üçgen 3.üçgen

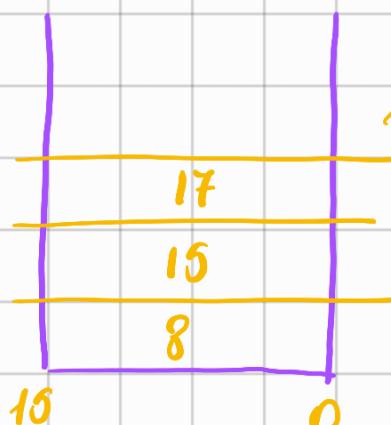
8

15

17

ardisit gelen 3 tane eleman bir üçgenin kenarini belirttiyo

bildipimiz bir sey var burda verilen 3 tane kenar muhakkak bir üçgen olusturuyor



→ stackiminin
içine attık

Bizim bir stackpointerimiz (SP) vardi yani bizim stack'in içindede nerde olduğumuzu gösteren bir SP'mız vardi.

SP'nin degeri otomatik olarak stack'e bir sey atadipimizde ya da stack'ten bir sey çektiğimizde kendisi degistiriyordu SP'yi biz kontrol etmiyoruz Bir de base pointer'imiz vardi (BP) bu base pointerin degerini stack'ten bir sey yazmak ve okumak için kullanıyoruz. stack pointer ile base pointer'in degerini esitledikten sonra mov BP, SP diyeret esitledikten sonra biz ortak base pointeri stack'in içeriğine istedipimiz gibi erişim yapmak

İçin kullanabiliriz

Stack word tipinde bir organizasyona sahip olduğunu içi mesela düşünelim stack pointer'in gösterdiği en son değer sun 17 stack pointer'i base pointer'a eşitledikten sonra base pointer'in gösterdiği değer de 17 BP+2'nin gösterdiği değer artık 15 BP+4'ün gösterdiği değer artık 8 yani karşı tarafta gönderdiğimiz seye böyle göreceli bir şekilde erişebiliyoruz

v'lu olan formulu

$$v = \frac{a+b+c}{2}$$

$$A(\triangle ABC) = \sqrt{v(v-a)(v-b)(v-c)}$$

Alanının karesi denmesinin sebebi bu karekökle uygulanıyor

Koda sunu söylemiş oluyoruz olan bul'ı
extra Alan_bul_ifor burda bulmazsun
myss Segment para stack 's' demis oluyoruz

Dw 20 DuP(?)

myss ENDS

myss Segment para 'd'

kenarlar Dw 6, 8, 5, 9, 4, 8, 2, 2, 3

n Dw 3

enbykalan Dw 0

myds ends

myCS segment para 't'

Assume CS:myCS, DS:myDS, SS:mySS

Ana proc Far

Push DS

XOR Ax, Ax

Push Ax

Mov Ax, myDS

Mov DS, AX

Mov Cx, n

XOR SI, SI

L1: Push kenarlar [SI] → ügenin birinci kenarını stack'e attıktır

Push kenarlar [SI+2] → ügenin 2. kenarını stack'e attıktır

Push kenarlar [SI+4] → ügenin 3. kenarını stack'e attıktır

call Alan_Bul

Pop BX
pop BX
pop BX

Cmp Ax, enbykalan

JB kucuk

Mov enbykolan, Ax

luat: ADD Sl. b

Loop L1

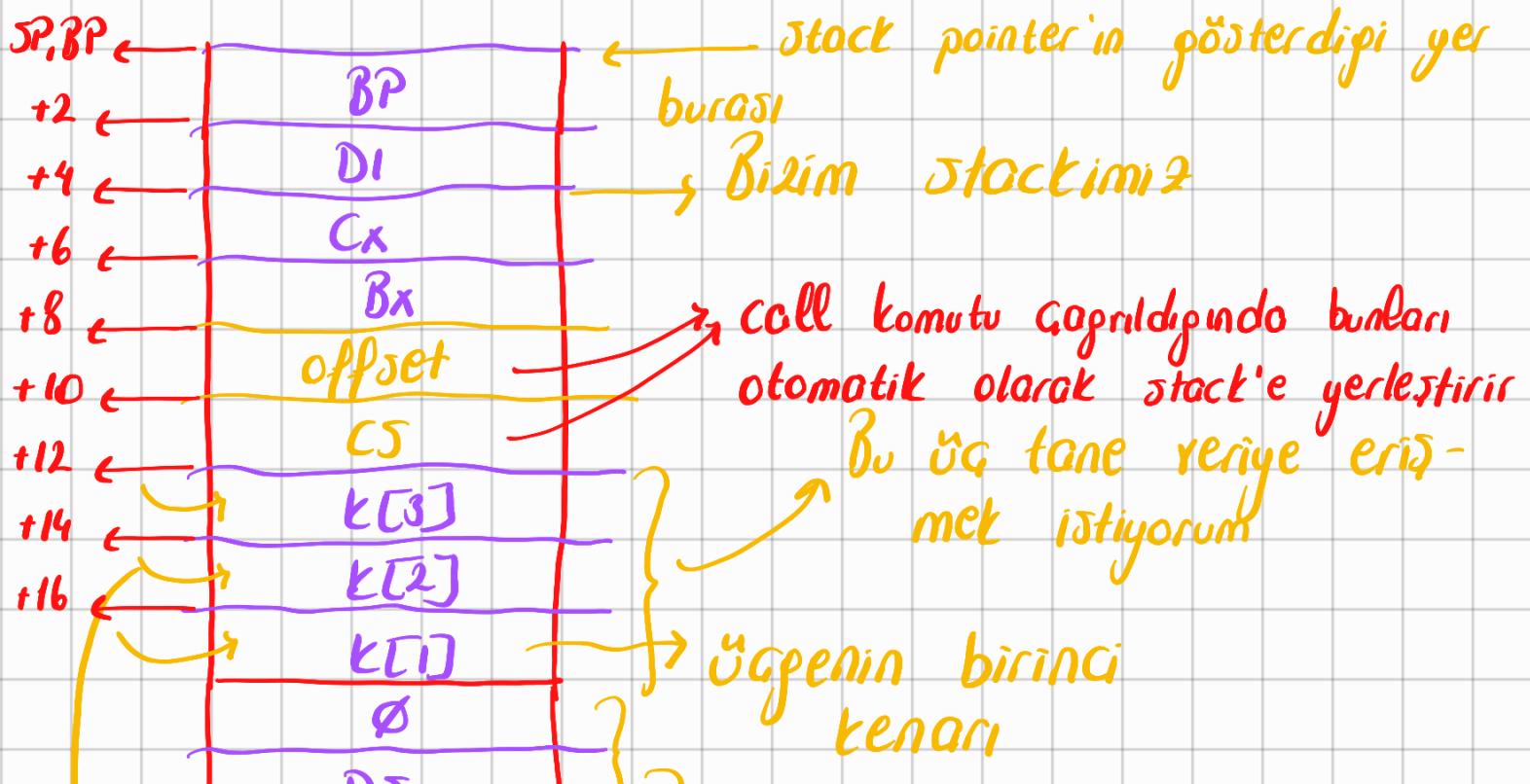
self

Ana endP

mycs ends

end ana

harici yordamlarınızın içinde kullandığınız
registerlerin değerlerini kullanmaya başlamadan
Önce stack'in içine oturmanız ordin dan
bunları tetrordon program bitmeden hemen
önce stack'ten çekmeniz beklenir.



15

0

Ben üşpenin 3
tanıe kenarına
erişmek istiyorum

Nasıl erişicem?
Üstüne büssürüp bir
sey koymam pop
yapamam

BP+12 ile k'nın
3. elemanına BP+14
ile k'nın 2. elemanına
BP+16 ile k'nın 1. ele-
manına ulaşınız

Stack'ımızın içinde
exe yazdığınıza
dolayı standart
olarak bunları yerles-
tiriyoruz

→ Ardından call harici
yordam yapmadığımız
an do ne oluyordu öncelikle
bu code segmentimizi (CS)
Stack'ımızın içeriğine bu
call komutu otomatik
olarak atıyor

→ Ardından bide offset
değerini atıyordu

→ Ardından yordama yordam içinde
kullandığımız registerleri
Stack'ımızın içine
atıyoruz

Public alan_bul
mycode Segment para 'kod'

Assume CS: mycode

ALAN_BUL Proc Far

Push Bx
Push Cx
Push DI
Push BP

Mov BP, SP
XOR Ax, Ax
ADD Ax, [BP+12]
ADD Ax, [BP+14]
ADD Ax, [BP+16]

JHR Ax, 1
Mov Bx, Ax
Sub Bx, [BP+12]
Mov Cx, Ax
Sub Cx, [BP+14]
Mov DI, Ax
Sub DI, [BP+16]

MUL Bx j Ax * Bx → Dx : Ax
MUL Cx j Ax * Cx → Dx : Ax
MUL DI

Pop BP
Pop DI
Pop Cx
Pop Bx

retf
Alan_bul EndP

mycode ends

end

Public Alan_Bul
mycode segment para 'cod'

Assume CS:mycode

Alan_Bul Proc For

Push BX

Push CX

Push DI

Push BP

Push DX

Mov BP, SP

Xor AX, AX

Add AX, [BP+14]

Add AX, [BP+16]

Add AX, [BP+18]

Shr AX, 1

Mov BX, AX

Sub BX, [BX+14]

Mov CX, AX

Sub CX, [BP+16]

Mov DL, AX

Mor Dİ. İXA
JuB Dİ, [BP+18]

mul Bx j Ax * Bx → Dx !Ax
mul Cx j Ax * Cx → Dx !Ax
mul DI

Pop Dx
Pop BP
Pop DI
Pop CX
Pop BX

retf
Alan_Bul endP

mycode ends

end

→ retf komutu stackteki offset ve CS'yi
alıyor ordin dönmesi gereken yerin
hangi kod segment ve hangi offset olduğunu
anlıyor ve oraya gidiyor

→ yani offset ve CS'yi kolduran
retf komutudur

→ eğer near tipi bir yardım olısaydı
orda sadece offset olacaktı

ve ret komutıyla o offseti ordan kaldırılmış olacaktı

→ Ardından sırasıyla k3, k2 ve k1'e geldiniz bunu kim kaldıracak burda 2 türlü yaklaşım sergileyebilirsiniz

→ 1- gönderdiğiniz yordam kullanmadığınız bir registerde pop içinde geçer
pop Bx
pop Bx
pop Bx

→ 3 tane değeri böyle verebiliriz

→ eger k3, k2 ve k1 geçemezsen yani bunları temizlemeğen program dondurduğunda doğru doto segmente peri döneneyiz

→ ilk yöntemimiz şöyle oldu yani gönderdiğimiz ana yordam içinde bunları temizlicez

→ harici yordamı yaratan kişinin bunu bilmesi lazım

↳ Bir diğer yöntem de ju
bize eger ti

Stackten toz byte'lik veri
temizleceğim retf'dan sonra
onu yazıyorum mesela retf 6
yani 6 byte'lik veri temizle
dediğimiz onda stackiminin
içindeki o 6 byte'lik verimizi
retf temizlemiş oluyor

↳ 2 farklı yaklaşım var 2'sinden
birini tercih edebilirsiniz

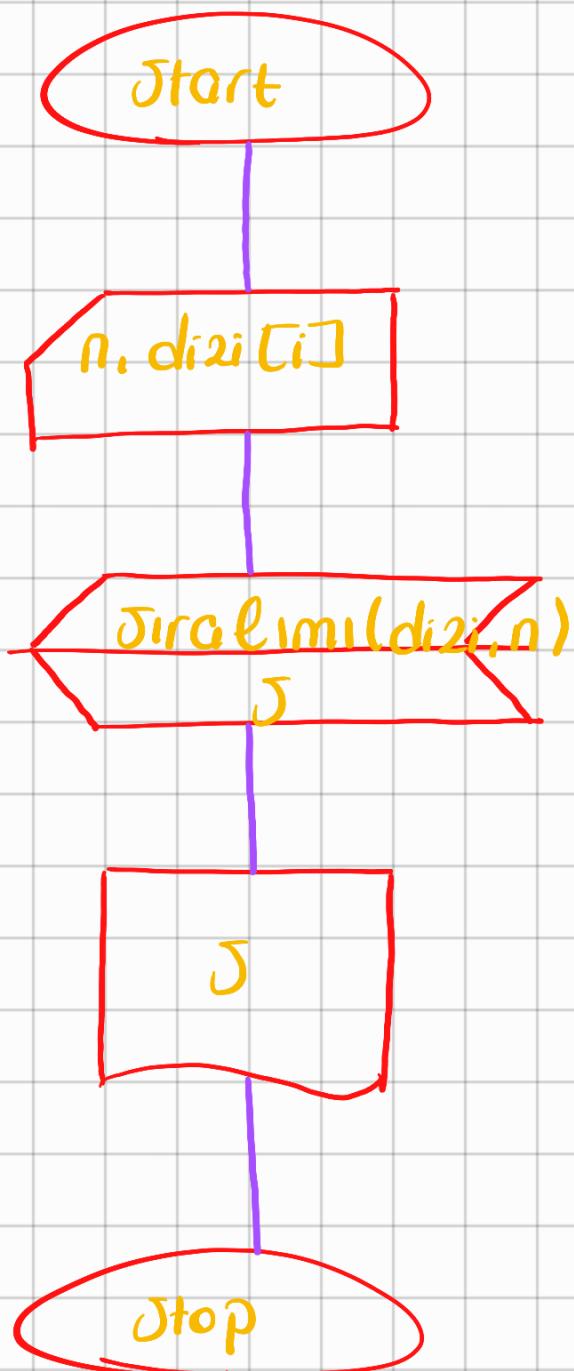
horici yordama gittipimizde horpi register-
leri kullanıyorsak onları saklamamız
lazım

↳ Güntük bu registerler ona yordan içinde
kullanılıyo mu kullanılmıyo mu bilmeyiz

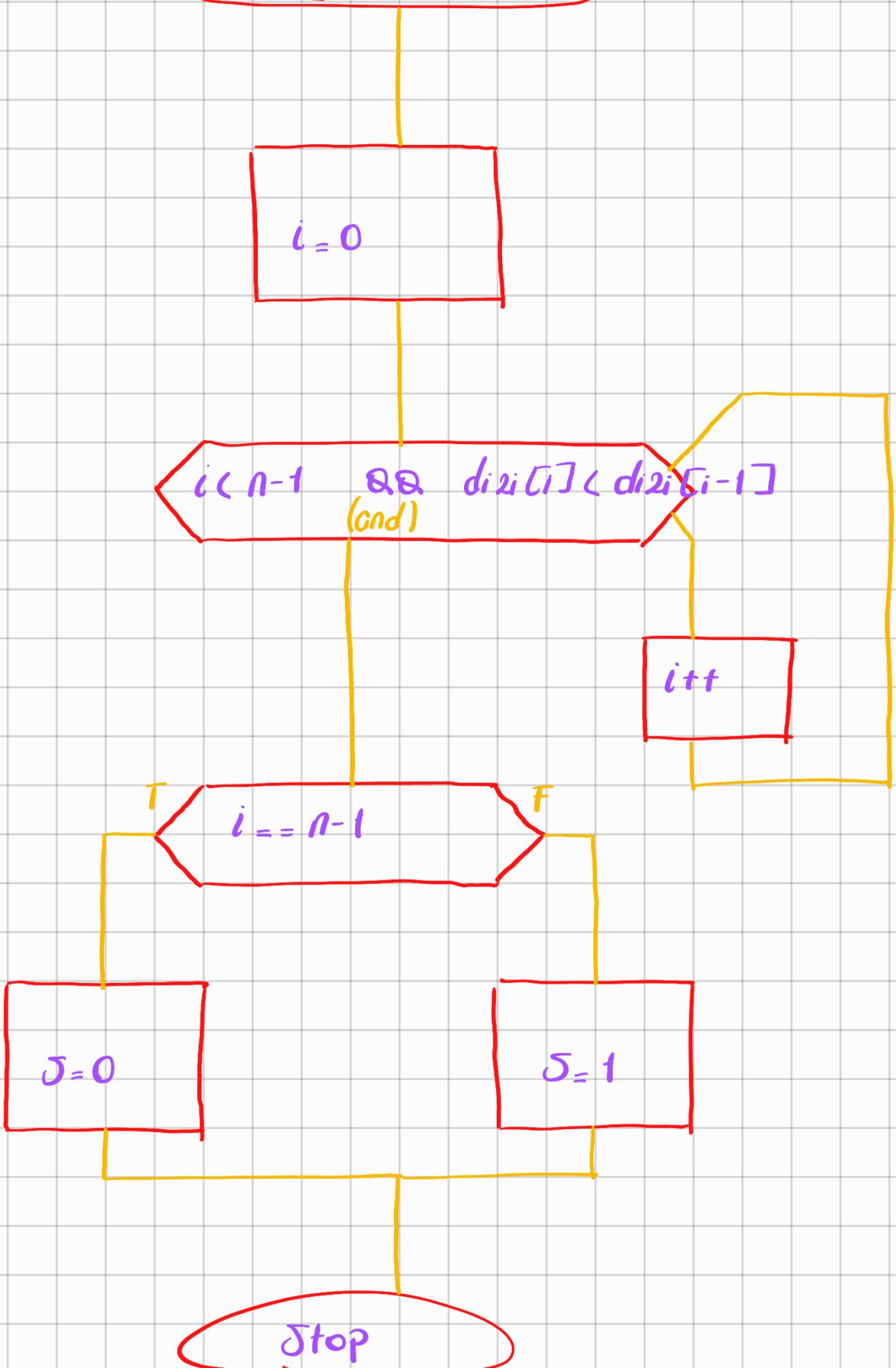
↳ mümkünse sonuclar Ax registeri üzerinden
dönüşün

Örnek 2 - extrn / Public ile parametre oktarma

Yerilen bir dizinin köküten büyüğe sıralı olup olmadığını bulan harici yordamı yazınız. Harici yordam verilere extrn ve public komutları ile erişmeli dir.



Sıralımı (int, int)



Stop

extrn Siralimi : Far

Public dizi .n

myss segment para stack 'SSS'

Dw 20 Dup(?)

myss ends

myds segment para 'd'

N Dw 7
J DB 0
dizi DB 12, 14, 16, 18, 20, 22, 24

myds ends

mycs segment para 'k'

Assume CS: mycs, DS: myds, SS: myss

Ana proc far

Push DS

XOR AX, AX

Push AX

Mov AX, myds

mov ds, Ax
call Jiroli
cmp AL, 0
jz Jiroli
mov S, 1

Jiroli : ret
Ana endp
mycs ends
end ana

Public Jiroli

extrn dizi : Byte , n : word

mycode segment para 'test'

Assume CS: mycode

Jiroli proc far

push SI
push CX
xor AX, AX
xor SI, SI
mov CX, n
dec CX
cmp SI, CX
JAE Jiroli
mov BH, dizi[SI]

mov ah, al
cmp ah, [si+1]
jge si+1+
inc si
jmp don

si+1+ : mov al

si+1 : pop cx
pop si

ret

si+1+ endp
mycode ends
end

