ALT SEVİYE PROGRAMLAMA

- ÖDEV 1 -

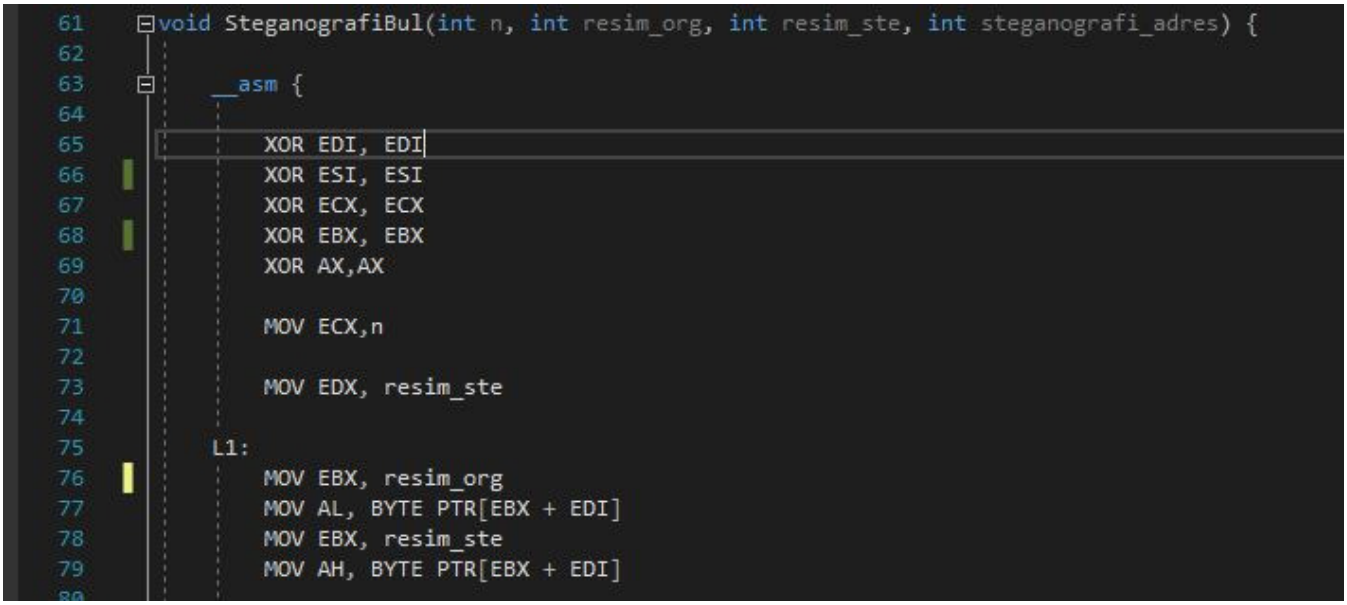
GRUP 1

İrem ATILGAN

17061036

05.12.2019

1. SORU



```
61  void SteganografiBul(int n, int resim_org, int resim_ste, int steganografi_adres) {
62
63      __asm {
64
65          XOR EDI, EDI
66          XOR ESI, ESI
67          XOR ECX, ECX
68          XOR EBX, EBX
69          XOR AX,AX
70
71          MOV ECX,n
72
73          MOV EDX, resim_ste
74
75      L1:
76          MOV EBX, resim_org
77          MOV AL, BYTE PTR[EBX + EDI]
78          MOV EBX, resim_ste
79          MOV AH, BYTE PTR[EBX + EDI]
```

Satır 65-73:

```
//int tipi 4 byte olduğu için 32 bit registerlar (EBX,ECX,EDX) kullanılır.
Öncelikle bu registerlar sıfırlanır. Bununla birlikte işlemler 16 ve 8
bitlik register'larda yapılacağı için AX de sıfırlanır.

//Dizinin eleman sayısı ECX'e aktarılır.

//Steganografik resmin başlangıç adresi EDX'e atılır.
```

Satır 75-79:

```
//Orijinal resmin başlangıç adresi EBX'e atılır

//EDI orijinal/steganografik resim dizisi içinde ilerlemek için index
olarak kullanılır

//orijinal dizinin DI'ncı indisindeki değeri AL register'ına kopyalar

//Steganografik resmin başlangıç adresi EBX'e atılır

//orijinal dizinin DI'ncı indisindeki değeri AH register'ına kopyalar
```

```
86          CMP AH,AL
87          JZ L2
88
89
90          SUB AH,AL
91          JG POSITIVE
92          ADD AH,256
93
94      POSITIVE:
95
96          MOV EDX, steganografi_adres
97          MOV BYTE PTR [EDX+ESI], AH
98          INC ESI
99
100
101     L2:
102         ADD EDI,2
103         LOOP L1
```

Satır 86-103:

//CMP ile iki dizinin elemanı kıyaslanır.

//Elemanlar eşitse L2 etiketine atlar ve diziyi gezmeye devam edilir.

//Elemanlar eşit değilse SUB instruction'ı ile farklarını alır.

//Eğer ilk sayı ikinci sayıdan küçükse sign flag 1 olacaktır. Burada CMP yerine SUB intruction'ının kullanılmasının nedeni sonucun ne çıktığına bağlı olmaksızın çıkarma işlemi yapılmasının gerekliliğidir.

//Oluşan fark eğer pozitifse POSITIVE etiketine atlar. POSITIVE etiketinde ise EDX'e harfleri yerleştireceğimiz dizinin başlangıç adresi olan steganografik_adres atılır. Daha sonra adrese, bulunduğumuz index değeri eklenir ve böylece kaldığımız dizi hücresine bulduğumuz harfin ASCII karşılığı değeri konulur.

//Eğer çıkartma işleminin sonucu negatifse işlemin sonucunun tutulduğu AH register'ına 256 eklenir (Resim 0-255 arasında değerler barındırdığı için modu alınmış olur). Bu işlemden sonra artık harfleri yazacağımız dizi için uygun duruma getirilmiş olur ve POSITIVE etiketinden devam eder.

//L2 Etiketine eğer iki dizinin elemanı birbirine eşitse direkt olarak, değilse bir dizi işlemler sonucunda ulaşılır ve dizinin tüm elemanları gezilene kadar döngü devam eder.

```
100
101         MOV BYTE PTR[EDX + ESI], ' '
102         MOV BYTE PTR[EDX + ESI + 1], '-'
103         MOV BYTE PTR[EDX + ESI + 2], ' '
104         MOV BYTE PTR[EDX + ESI + 3], '1'
105         MOV BYTE PTR[EDX + ESI + 4], '7'
106         MOV BYTE PTR[EDX + ESI + 5], '0'
107         MOV BYTE PTR[EDX + ESI + 6], '6'
108         MOV BYTE PTR[EDX + ESI + 7], '1'
109         MOV BYTE PTR[EDX + ESI + 8], '0'
110         MOV BYTE PTR[EDX + ESI + 9], '3'
111         MOV BYTE PTR[EDX + ESI + 10], '6'
112         MOV BYTE PTR [EDX + ESI + 11], 0h
113
```

Satır 101-112:

//Ödevde istenilen öğrenci numarası teker teker harflerin tutulduğu dizinin sonuna eklenir.

//Dizinin bittiği yere kadar yazdırması için dizinin son elemanından sonraki hücreye NULL eklenir.

2. SORU

```
STEK SEGMENT PARA STACK 'STACK'
    DW 300 DUP(?)
STEK ENDS

DSEG SEGMENT PARA 'DSEGMENT'
    kenarDizi DW 100 DUP(?) ;Dizi elemanları 0-1000 aralığında değerler alabileceğini için
                            ;data segmentte word tipinde elemanlar içeren bir dizi tanımlanmıştır
    kenarSayisi DB ?    ;Dizi en fazla 100 eleman alacağı için kenar sayısı byte tipinde tanımlanmıştır
    CR EQU 13           ;ENTER tuşunun ASCII karşılığı tanımlanmıştır


    ;Mesajlar
    parenthesis1 DB '(',0
    parenthesis2 DB ')',0
    comma DB ',',0
    arrow DB '->',0
    MSG1 DB 13,10,"Ucgen olmaya aday kenar sayisini giriniz (n) : ",0
    MSG2 DB 13,10,"Kenar Girisi = ",13,10,0
    HATA DB "Girdiginiz Sayi TAM SAYI Degildir !! Lutfen Tekrar Giris Yapiniz..",13,10,13,10,0
    YETERSIZ DB 13,10,"Yetersiz Kenar Sayisi = ",0
    KENARYAZ DB "Kenarlar = ",0
    MSGNO DB "Verilen dizide ucgen olusturabilecek eleman yok !",13,10,0

DSEG ENDS

CSG SEGMENT PARA 'CODE'
    ASSUME SS:STEK,DS:DSEG,CS:CSG
MAIN PROC FAR

    ;Sistemin geri dönebilmesi için AX ve DS değerleri stack'e atılır
    PUSH DS
    XOR AX,AX
    PUSH AX
```

```

KENARAL:
    ;Ucgen olmaya aday kenar sayisi girisi yapilir

    MOV AX,OFFSET MSG1 ;AX'e MSG1 mesajinin baslangic adresi konulur
    CALL PUT_STR ;String yazdirma fonksiyonu

    CALL GETN ;Input alimi (Eleman sayisi)
    MOV kenarSayisi,AL  ;kenar sayisi (input) 1 byte olacagi icin degiskene AL'den atama yapilir
    CMP kenarSayisi,3   ;kenar sayisi 3'ten az mi diye kontrol edilir
    JAE DEVAM           ;3 ve daha fazla ise devam edilir

    MOV AX,OFFSET YETERSIZ
    CALL PUT_STR            ;3'ten az kenar varsa hata mesaji yazılır
    XOR AX,AX
    MOV AL,kenarSayisi
    CALL PUTN               ;Kullanici tarafindan girilien kenar sayisi yazdirilir
    JMP KENARAL             ;Yeniden kullanicidan kenar alınır

DEVAM:

    ;Dizi elemanlari kullanicidan almaya baslar
    MOV AX,OFFSET MSG2
    CALL PUT_STR

    XOR CX,CX
    MOV CL,kenarSayisi ;Dongude kullanmak uzere CL register'ına kenar sayisi aktarilir
    XOR SI,SI       ;Dizide indis olarak kullanmak icin SI sifirlanir

```

```
INPUT_ARRAY:

    XOR AX,AX
    MOV AX,OFFSET arrow
    CALL PUT_STR

    CALL GETN               ;GETN fonksiyonuyla dizi elemani input olarak alinir
    MOV kenarDizi[SI],AX    ;alinan input dizinin bir hucresine aktarilir
    ADD SI,2                ;Dizi word tipinde oldugundan, gelecek hucreye gecmek icin SI 2 arttirilir

    LOOP INPUT_ARRAY


    ;Inputlar alindiktan sonra dizi oncelikle kucukten buyuge olacak sekilde siralanir
    ;Bunun icin bubble sort kullanilir

    ;Sort islemi icin SI ve DI sifirlanir
    XOR SI,SI
    XOR DI,DI

    XOR CX,CX
    MOV CL,kenarSayisi  ;Donguye baslamak icin CL register'ına kenar sayisi aktarilir
    DEC CL              ;Dizinin son elemanina gidilmesi gerekmedigi icin CL register'ındaki
                        ;deger bir azaltılır

SORT1:
    PUSH CX             ;CX ic ice dongude tekrar kullanilacagi icin eski degerini kaybetmemesi
                        ;icin stack'e kopyalanir
    MOV DI,SI           ;SI'daki deger DI'ya kopyalanir
```

```
SORT2:
    ;Bubble sortta tum dizi gezilerek dizinin bir elemani ile dizinin
    ;bir sonraki elemani kiyaslanir (N elemanli bir dizi bastan sonra N-1 kez gezilir)
    ;Bir sonraki elemana ulasabilmek icin DI arttirilir. Dizi word tipinde oldugu icin de 2 eklenir
    ADD DI,2
    MOV AX,kenarDizi[SI]
    CMP AX,kenarDizi[DI]    ;Dizi elemaniyla dizideki bir sonraki eleman kiyaslanir
    JB SORT_FIN             ;Ilk eleman kucukse diziyi gezmeye devam eder (kucukten buyuge
                            ;siralama yapildigi icin)

    ;Eger bir sonraki kenar uzunlugu daha buyukse dizideki yerlerini degis tokus et
    XCHG AX,kenarDizi[DI]
    MOV kenarDizi[SI],AX

SORT_FIN:

    LOOP SORT2  ;Ic dongu
    POP CX      ;Ic dongu bittiginde stack'te tutulan dis donguye ait CX degeri stack'ten cekilir
    ADD SI,2    ;Dis dongunun index'i 2 (dizi word oldugu icin) arttirilir
    LOOP SORT1  ;Dis Dongu

    ;Bubble Sort islemi bittikten sonra dizideki kenar uzunluklarindan olusturulacak
    ;en az buyukluge sahip cevre bulunmaya calisilir
    ;Oncelikle kenar uzunluklarinin ucgen olma sartini karsilayip karsilamadigi kontrol edilmelidir
    ;Eger uc kenardan en kisa ilk ikisinin toplami, en buyuk uzunluga sahip kenardan daha buyukse
    ;Ucgen olma sartini karsilar
    ;Dizi elemanlarini teker teker gezip dizinin bir sonraki elemaniyla toplar ve
    ;ondan sonraki elemanla kiyaslarsak ucgen olma
    ;sartini karsilayip karsilamadigini anlayabilir, hatta elimizdeki kenarlar kucukten buyuge siralandigi icin
    ;ucgenlik sartini karsilayan ilk 3 elemanin en az cevreye sahip olacagini iddia edebiliriz

    MOV CL,kenarSayisi  ;Kenar sayisi byte tipinde oldugu icin deger CX'e degil CL'ye kopyalanir
    XOR SI,SI
    SUB CL,2            ;Dizinin son iki elemanina gidilmesine gerek olmadigi icin CL kenar sayisi - 2 degerini alir

```

```
CONTROL:
    MOV AX,kenarDizi[SI]    ;Dizi elemani AX'e atanir
    ADD AX,kenarDizi[SI+2]  ;AX'e dizinin sonraki elemani eklenir
    CMP AX,kenarDizi[SI+4]  ;Toplam 3. degerle kiyaslanir
    JA CEVRE                ;Eger ilk iki elemanin toplami  ucuncu elemandan daha buyukse minimum cevreye
                            ;sahip kenarlar bulunmus demektir
    ADD SI,2                ;Dizi word tipinde oldugu icin diger elemana gecebilmek icin segment index (SI)'ya
                            ;2 eklenir
    LOOP CONTROL            ;Daha buyuk degilse dongu CL sart saglanana veya 0 olana dek devam eder
    JMP NO                  ;Dongu tamamlanmissa ucgen olusturma sartini karsilayabilen 3 kenar bulunamamis demektir
                            ;NO Etiketine atla

CEVRE:

    ;Ucgen olma sarti karsilayan, minimum cevreye sahip ucgenin kenarlari yazdirilir
    MOV AX,OFFSET KENARYAZ
    CALL PUT_STR
    MOV AX,OFFSET parenthesis1
    CALL PUT_STR
    MOV AX,kenarDizi[SI]
    CALL PUTN
    MOV AX,OFFSET comma
    CALL PUT_STR
    MOV AX,kenarDizi[SI+2]
    CALL PUTN
    MOV AX,OFFSET comma
    CALL PUT_STR
    MOV AX,kenarDizi[SI+4]
    CALL PUTN
    MOV AX,OFFSET parenthesis2
    CALL PUT_STR


    JMP L1 ;MAIN'in sonuna gidilir
```

```
NO:
    MOV AX,OFFSET MSGNO     ;Ucgen olusturulabilecek eleman olmadigi yazdirilir
    CALL PUT_STR
L1:

    RETF                    ;Ana yordam FAR oldugu icin RETF ile donulur
                            ;Baslangicta PUSH ettigimiz DS ve AX degerleri cekilir
MAIN ENDP

;GETC FONKSIYONU

GETC PROC NEAR
    MOV AH,1H   ;AH = 1 ve INT 21H yazılırsa
    INT 21H     ;Keyboard'dan bir karakter okur ve ekranda gosterir
                ;Okunan karakter AX'e atanir
    RET         ;Fonksiyon NEAR tipinde oldugu icin RET yazilir
GETC ENDP       ;PROC biter

;PUTC FONKSIYONU

PUTC PROC NEAR
    ;AX ve DX'in degeri degisecegi icin mevcut degerleri stack'e kopyalar
    PUSH AX
    PUSH DX

    MOV DL,AL

    MOV AH,2    ;AH = 2 ise ve interrupt handler 0x21 call edilirse
    INT 21H     ;Ekrana bir karakter yazar

    ;Stack'ten push edilen degerler cekilir
    POP DX
    POP AX

    RET         ;Fonksiyon geri donebilmek icin stack'ten IP ve CS'yi cekerek MAIN'e geri doner
PUTC ENDP
```

```
;GETN FONKSIYONU

GETN PROC NEAR

    ;BX,CX ve DX degerleri degisecegi icin stack'e kopyalanir
    PUSH BX
    PUSH CX
    PUSH DX

GETN_START:

    MOV DX,1    ;DX register'ı birler basamagini temsil etmesi amaciyla 1 degerini alir

    ;BX ve CX sifirlanir
    XOR BX,BX
    XOR CX,CX

NEW:
    CALL GETC       ;karakteri okumak icin GETC fonksiyonu cagirilir
    CMP AL,CR       ;CR->ENTER tuşuna basildi mi diye kontrol edilir
    JE FIN_READ     ;Basildiysa okumayi bitir
    JMP CTRL_NUM    ;Basilmadiysa CTRL_NUM etiketine atla
HATALI:
    MOV AX, OFFSET HATA
    CALL PUT_STR
    JMP GETN_START
CTRL_NUM:
    CMP AL,'0'      ;Girilen karakterin ASCII kodunu '0'ın ASCII koduyla kiyaslar
    JB HATALI       ;Daha kucukse hata mesaji verir
    CMP AL,'9'      ;Girilen karakterin ASCII kodunu '9'un ASCII koduyla kiyaslar
    JA HATALI       ;Daha buyukse hata mesaji verir

    SUB AL,'0'      ;Girilen karakter '0' dan cikarilarak karakter numerik hale getirilir
    MOV BL,AL
    MOV AX,10

    ;Birler basamagindan baslanarak her karakter girildiginde
    ;onceki deger (CX) = onceki deger (CX) * 10 (AX) + yeni deger (BX) islemi yapilarak
    ;sayi numeriklestirilir

    PUSH DX         ;DX değişmesin diye yığına at
    MUL CX          ; DX : AX <- AX*CX
    POP DX          ;DX'in eski değeri yigindan geri alinir
    MOV CX,AX       ; CX <- AX
    ADD CX,BX       ; CX <- CX + BX
    JMP NEW         ;Yeniden karakter okuma işlemine atlanir
FIN_READ:
    MOV AX,CX       ;Okuma bittiyse elde edilen sayi AX'e kopyalanir
    ;Basta stack'e atilan register'lar stack'ten geri cekilir
    POP DX
    POP CX
    POP DX

    RET
GETN ENDP
```

```
;PUTN FONKSIYONU
PUTN PROC NEAR
    ;CX ve DX register'lari stack'e kopyalanir
    PUSH CX
    PUSH DX
    ;DX sifirlanir ve stack'e kopyalanir
    XOR DX,DX
    PUSH DX

    ;Sayimiz surekli 10'a bolunerek kalan DX yazdirilir, bolum olan AX
    ;ise 0 olana dek 10 a bolunmeye devam eder

    MOV CX,10
CALC_DIGITS:
    DIV CX              ;Ekrana yazdirilmasi gereken sayi neyse CX'e bolunur (DX:AX = AX/CX=
                        ;DX kalan, AX bolum olur
    ADD DX,'0'          ;DX'e '0' karakterinin ASCII kodu eklenir
    PUSH DX             ;Elde edilen basamagin ASCII degeri stack'e kopyalanir
    XOR DX,DX
    CMP AX,0            ;AX 0 ise yazdirilacak sayi kalmamistir, LOOP_YAZDIR etiketine gider
    JNE CALC_DIGITS     ;AX 0 degilse basamaklari gezmeye devam et
LOOP_YAZDIR:
    POP AX              ;AX <- DX (Stack'e kopyalanan DX degerleri loop icinde POP ile AX'e konulur)
    CMP AX,0            ;AX degeri 0 ise yazdirilacak sayinin sonuna gelinmistir
    JE END_YAZDIR       ;Dogruysa yazdirmayi bitir
    CALL PUTC           ;Degilse elimizdeki karakteri ekrana yazdirmak icin PUTC
                        ;fonksiyonunu cagir
    JMP LOOP_YAZDIR     ;DX'in 0 oldugu durum stack'ten cekilene dek dongu devam eder
END_YAZDIR:
    ;Fonksiyonun basinda stack'e atilan CX ve DX register degerleri
    ;sirayla stack'ten geri cekilir
    POP DX
    POP CX
    RET
PUTN ENDP   ;FONKSIYON SONU
```

```
;PUT_STR FONKSIYONU
PUT_STR PROC NEAR
    PUSH BX                 ;BX register degeri stack'e kopyalanir
    MOV BX,AX               ;Yazdirilmasi istenen string'in offset degeri AX'tedir
                            ;Bu deger BX'e yerlestirilir
    MOV AL,BYTE PTR [BX]    ;OFFSET'teki deger byte olarak AL register'ına aktarilir
LOOP1:
    CMP AL,0                ;Okunan deger 0 ise string'in sonuna gelinmistir
    JE FIN                  ;Eger oyleyse FIN etiketine atla
    CALL PUTC               ;Karakter karakter yazdirma yapilacagi icin PUTC fonksiyonu cagirilir
    INC BX                  ;Adres degeri (BX) 1 arttirilir
    MOV AL, BYTE PTR [BX]   ;Adresin icindeki deger AL'ye aktarilir
    JMP LOOP1               ;AL 0 olana dek dongu devam eder
FIN:
    POP BX                  ;Fonksiyonun basinda stack'e PUSH edilen BX register'ı stack'ten cekilir
    RET
PUT_STR ENDP                ;FONKSIYON SONU


CSG ENDS                    ;CODE SEGMENT BITER
    END MAIN
```