



ALT SEVİYE PROGRAMLAMA ÖDEV RAPORU

ÖĞRENCİ ADI: Ertuğrul ŞENTÜRK

ÖĞRENCİ NO: 18011028

ÖĞRENCİ MAIL: mdesenturk@gmail.com

l1118028@std.yildiz.edu.tr

DÖNEM: 2

GRUP NO: 2

DERSİN EĞİTMENİ: Arş.Gör. Furkan ÇAKMAK

ÖDEV 1:

```
void SteganografiBul(int n, int resim_org, int resim_ste, int steganografi_adres) {
```

```
    char no [12] = " - 18011028";  
    int no_adres= (int)no;
```

```
    __asm {
```

```
        MOV EBX, steganografi_adres
```

```
        MOV ECX, n
```

```
        MOV ESI, resim_org
```

```
        MOV EDI, resim_ste
```

```
L3:     MOV AX, WORD PTR[ESI]
```

```
        MOV DX, WORD PTR[EDI]
```

```
        CMP AX, DX
```

```
        JE L2
```

```
        CMP AX, DX
```

```
        JB L1
```

```
L1:     ADD DX, 256
```

```
        SUB DX, AX
```

```
L2:     MOV BYTE PTR[EBX], DL
```

```
        INC EBX
```

```
L2:     INC EDI
```

```
        INC EDI
```

```
        INC ESI
```

```
        INC ESI
```

```
        LOOP L3
```

```
        MOV ECX, 11
```

```
        MOV ESI, no_adres
```

```
L4:     MOV AL, BYTE PTR[ESI]
```

```
        MOV BYTE PTR[EBX], AL
```

```
        INC EBX
```

```
        INC ESI
```

```
        LOOP L4
```

```
        MOV BYTE PTR[EBX], 0
```

```
    }
```

Öğrenci numarası karakter dizisi olarak no değişkeninde saklandı. Bu dizinin adresi de diğer diziler gibi DW lük integer değişkende saklandı.

Dizinin eleman sayısı loop counterı olan ECX'de, dizi adresleri ise sırası ile DW boyutundaki registerlara atandı.

ESI ve EDI adresleri word boyutunda veriyi işaret ettiklerinden; orjinal resim elemanı-> AX'e, düzenlenmiş resim elemanı->DX'e atandı.

AX ve DX değerleri aynıysa gizli yazıyı saklayan karakter dizisine eleman eklememek için L2 bölümüne atlandı.

Elemanların aynı olmaması durumunda AX ve DX yeniden kontrol edilerek eğer AX daha büyükse AX+gizlenen karakter değeri 256'yı aşmış olduğu anlamına geldiğinden DX'e 256 eklenerek negatif değerlerin oluşması önlemlendi.

DX den AX çıkartıldı ve sonuç 0-256 arasında olmak zorunda olduğundan DH her durumda 0 çıktı.

DL kısmı saklanılan karakterin ASCII değerini içerdiği için gizli metnin saklandığı adrese kopyalandı.

EBX char değeri sakladığından 1'er, EDI ve ESI word değeri sakladığından 2'şer arttırıldı. ECX zaten en başta eleman sayısına atandığından loop yapıldı.

Son olarak 11 karakterli öğrenci no kısmı ekleneceğinden CX 11'e atandı. ESI ise öğrenci noyu saklayan dizinin adresine atandı. LOOP ile ilk bulunan dizinin sonuna eklendi.

Yazının bittiğini belirtmek için en son karakter olarak 0 (NULL ASCII değeri) eklendi.

ÖDEV 2:

1	myss	SEGMENT PARA	STACK 'yigin'	STACK SEGMENT TANIMI
2		DW 20	DUP(?)	
3	myss	ENDS		DATA SEGMENT TANIMI
4				
5	myds	SEGMENT PARA	'stack'	
6	n	DW	(?)	n = dizinin boyutu
7	temp	DW	(?)	temp = input alınan sayıların word değerini saklayan geçici eleman
8	string	DB	5 DUP(?)	string = word olarak alınan sayının stringe çevrildiğinde saklanacağı dizi
9	a	DW	(0)	a, b, c, Üçgenin kenarları a kontrol işlemi için 0 a atandı.
10	b	DW	(?)	d_in = input olarak alınan karakterleri aski tablosundaki '0' dan çıkartıp sayısal değer olarak saklayan dizi.
11	c	DW	(?)	dizi = verilen dizinin elemanlarının sayısal değerlerini saklayan dizi
12	d_in	DW	10 DUP(?)	eleman, exp1, err, err2, err3 kullanıcıya verilen output mesajları
13	dizi	DW	100 DUP(?)	min = kenarların toplam değerini saklayan değişken
14	eleman	DB	'Lutfen Dizinin eleman sayisini giriniz.',13,10,'\$'	en küçüğü aradığımızdan maximum word değere atandı.
15	exp1	DB	'Dizinin elemanlarini giriniz.',13,10,'\$'	ran_min = Input için alt sınır, ran_max = Input için üst sınır
16	err	DB	'Verilen dizide ucgen olusturabilecek eleman yok !',13,10,'\$'	
17	err2	DB	13,10,'Lutfen 0 ila 9 arasinda rakamlar giriniz!',13,10,'\$'	
18	err3	DB	'Sayi limit asimi! Tekrar giris yapin',13,10,'\$'	
19	min	DW	0FFFFh	
20	ran_min	DW	(?)	
21	ran_max	DW	(?)	
22	myds	ENDS		
26	GETINPUT	MACRO		
27		LOCAL M0, M1, M2, M3, M4, M5, M6, M7, M8, M9		GETINPUT: Kullanıcıdan 0 ila 9 arası karakterlerden oluşan bir string alıp hata durumunda kullanıcıya uyarı veren makro
28		MOV BL,10		BL ye 10 atadık bu sayede 10un katı olan değerleri alabileceğiz.
29	M0:	LEA SI,d_in		SI içine rakamları geçici olarak dizide tutacak dizinin adresi atandı.
30		XOR CX,CX		CX değeri eleman sayısını tutacağından sıfırlandı. temp değeri kullanılmak üzere sıfırlandı.
31		MOV temp,0		AH değerini input almak için 1 e eşitledik. Interrupt ile kullanıcıdan AL ye input alındı.
32	M1:	MOV AH,1		
33		INT 21h		
34		CMP CX,0		CX in 0 olduğu durumda yani ilk eleman için enter girilirse tekrar input almak için kod M0 labelına gönderildi.
35		JNE M2		
36		CMP AL,0Dh		
37		JE M0		
38	M2:	CMP AL,0Dh		
39		JE M4		
40		CMP AL,30h		
41		JB M3		
42		CMP AL,39h		
43		JA M3		
44		SUB AL,30h		
45		MOV [SI],AL		Alınan değerden 0'ın ASCII tablosundaki değeri çıkartılarak sayı değeri elde edildi.
46		INC SI		
47		INC CX		Alınan değerler d_in dizisine SI registeri ile kaydedildi. CX değeri rakam sayısını tutmak için, SI sonraki elemanı göstermek için arttırıldı.
48		JMP M1		
49	M3:	MOV AH,9		Yeni rakam almak için M1 labeline atlandı.
50		LEA DX,err2		
51		INT 21h		
52		JMP M0		Yanlış karakter girilmesi halinde inputun tamamen yenilenmesi için kod M0 a yollandı.
53	M4:	LEA SI,d_in		
54	M5:	MOV AL,[SI]		M4 labeli input alımının bittiği label bu adımdan sonra hesaplamalar için SI yeniden dizinin ilk elemanına alındı.
55		XOR AH,AH		
56		PUSH CX		
57	M6:	CMP CX,1		Bir while loopu ile kaydedilen elemanlar BL ye kayıtlı 10 sayısı ile basamak numarasını tutan CX adedince çarpıldı.
58		JBE M7		
59		MUL BL		
60		DEC CX		Bir dış for loopu ile de AX'de kaydedilen basamak değerleri temp değişkeninde toplandı.
61		JMP M6		
62	M7:	POP CX		Her loopta CX 0'lanacağından CX değeri stackda korundu.
63		ADD temp,AX		
64		INC SI		
65		LOOP M5		Hesaplanan tamsayı değeri belirlenen minimum ve maximum değerleri karşılaştırıldı
66		MOV AX,temp		aralıkta değilse kullanıcıya hata mesajı vermek için M8 labeline yönlendirilip ardından M0 labeline tekrar gönderip kullanıcıdan yeniden input alındı.
67		CMP AX,ran_min		
68		JL M8		
69		CMP AX,ran_max		Eğer input sorunsuz alınmış ise M9 labeli ile makrodan çıkıldı.
70		JLE M9		
71	M8:	MOV AH,9		
72		LEA DX,err3		
73		INT 21h		
74		JMP M0		
75	M9:			
76	ENDM			

80	WORDTOS MACRO sayi,string	WORDTOS: Word değerini stringe çevirip karakter olarak yazdıran makro
81	LOCAL W0,W1	
82	XOR CX,CX	
83	LEA DI,string	Makro üzerinde stack ve data bölümleri kullanılması gerekmediğinden tanımlanmadı.
84	ADD DI,4	
85	MOV AX,sayi	Makroya input olarak bir word boyutunda sayı ve string adresi gönderildi.
86	MOV BL,10	
87	W0: DIV BL	CX değeri içinde hexadecimal olarak alınan sayının decimal boyutu saklamak üzere sıfırlandı. AX sayının değerine atandı. DI iteratörü içine string dizisinin adresi atandı. Dizinin son elemanına erişmek için DI 4 arttırıldı.
88	INC CX	
89	ADD AH,30h	
90	MOV BYTE PTR[DI],AH	
91	DEC DI	AX BL ye atanan 10 değerine bölündü. AH da oluşan kalan değerine '0' karakterinin ASCII değeri eklenerek char değeri hesaplandı. CX değeri boyutu hesaplamak üzere arttırıldı.
92	XOR AH,AH	
93	CMP AX,0	
94	JNE W0	Hesaplanan değer son rakamdan ilk rakama doğru olacağından diziye sondan başa kaydedildi. AX 0 oluncaya kadar bölme işlemi loop edildi böylece sayının tüm rakamları diziye eklenmiş oldu.
95	INC DI	
96	MOV AH,2	
97	W1: MOV DL,BYTE PTR[DI]	
98	INC DI	Loop sonunda DI iteratörü dizinin ilk elemanından 1 önceki elemanı gösterdiğinden 1 arttırılarak dizinin ilk elemanını göstermesi sağlandı. Hesaplanan decimal sayının rakam adedi CX de saklandığından loop edilerek her karakter AH,2 komutu ile yazdırıldı.
99	INT 21h	
100	LOOP W1	
101	ENDM	
102	mycs SEGMENT PARA 'code'	CODE SEGMENT TANIMI
103	ASSUME SS:myss,DS:myds,CS:mycs	
104	UCGEN PROC FAR	EXE tipi program için temel tanımlamalar yapıldı.
105	PUSH DS	
106	XOR AX,AX	
107	PUSH AX	
108	MOV AX,myds	
109	MOV DS,AX	Eleman sayısını isteyen text yazdırıldı.
110	MOV AH,9	
111	LEA DX,eleman	
112	INT 21h	
113	MOV ran_min,1	Eleman sayısı alt sınırı 1 üst sınırı 100 olacak şekilde ayarlandı.
114	MOV ran_max,100	
115	GETINPUT	Eleman sayısı GETINPUT makrosu ile belirlenen sınırlar arasında alındı.
116	MOV AX,temp	
117	MOV n,AX	
118	MOV AH,9	Eleman isteyen text yazdırıldı. Alt sınır 0, üst sınır 1000 olacak şekilde değiştirildi.
119	LEA DX,expl	
120	INT 21h	
121	MOV ran_min,0	
122	MOV ran_max,1000	Alınan dizi eleman sayısı CX e atandı, makro üzerinde değiştirilebileceğinden stack a yollandı.
123	MOV CX,n	
124	LEA DI,dizi	
125	L0: PUSH CX	L0 loopu ile GETINPUT makrosunu kullanarak tüm elemanlar dizi'ye kaydedildi.
126	GETINPUT	
127	POP CX	
128	MOV AX,temp	
129	MOV WORD PTR [DI],AX	
130	INC DI	
131	INC DI	
132	LOOP L0	
133	LEA SI,dizi	
134	MOV CX,n	
135	CMP CX,0	
136	JE L7	
137	L1: MOV DI,SI	İÇİÇE 3 FOR LOOPU AÇIKLAMASI
138	PUSH CX	L0 labeli ile tanımlanan diziye eleman alan loop bittikten sonra SI üzerine kaydedilen dizinin adresi alındı. CX üzerinde dizinin eleman sayısı alındı. CX 0 olduğunda loopa girmek için kontrol oluşturuldu.
139	DEC CX	
140	CMP CX,0	
141	JE L6	
142	L2: INC DI	Dizinin tüm elemanlarını 3 erli kontrol edebilmek için içiçe 3 for döngüsü kullanıldı. Döngülerin her birinde register olarak sırasıyla SI DI ve BX kullanıldı. DI nin değeri SI nin değerinden BX in değeri DI nin değerinden kopyalanarak loop sonrası değişen DI ve SI değerleri eski değerlere stack aracılığı olmadan doğrudan atanabildi.
143	INC DI	
144	MOV BX,DI	Her loopun bir dış döngünün 1 sonrasındaki elemanı alarak o döngüden 1 sefer daha az olacak şekilde çalışması için her iç looptan önce CX stack a atılarak saklandı ve loop bitince geri çağırıldı.
145	PUSH CX	
146	DEC CX	
147	CMP CX,0	En iç döngüde a+b>c a+c>b b+c>a şartının sağlanması durumu kontrol edildi.
148	JE L5	
149	L3: INC BX	Eğer bu şart sağlanıyorsa a+b+c ilk değeri maxime aayarlanmış min elemanı ile karşılaştırıldı.
150	INC BX	
151	MOV AX,WORD PTR [SI]	
152	ADD AX,WORD PTR [DI]	
153	CMP AX,WORD PTR [BX]	Toplamin sonucu daha düşük olması durumunda a b c değerleri değışkene kaydedildi.
154	JBE L4	
155	MOV AX,WORD PTR [SI]	Eğer bu şartlardan biri bile sağlanmazsa o döngü L4 e gönderilerek işlemler atlandı.
156	ADD AX,WORD PTR [BX]	
157	CMP AX,WORD PTR [DI]	
158	JBE L4	CX 0 olduğu durumlarda loopa girilmemesi için L5 L6 L7 labelları oluşturuldu.
159	MOV AX,WORD PTR [DI]	
160	ADD AX,WORD PTR [BX]	Elemanlar word boyutunda saklandığından SI DI ve BX her döngüde 2 şer arttırıldı ve WORD PTR kullanıldı.
161	CMP AX,WORD PTR [SI]	
162	JBE L4	
163	ADD AX,WORD PTR [SI]	
164	CMP AX,min	
165	JAE L4	
166	MOV min,AX	

```

167      MOV AX,WORD PTR [SI]
168      MOV a,AX
169      MOV AX,WORD PTR [DI]
170      MOV b,AX
171      MOV AX,WORD PTR [BX]
172      MOV c,AX
173      L4: LOOP L3
174      L5: POP CX
175      LOOP L2
176  L6: INC SI
177      INC SI
178      POP CX
179      LOOP L1
180  L7: MOV AX,a
181      CMP AX,0
182      JNE L8
183      MOV AH,9
184      LEA DX,err
185      INT 21h
186      JMP L9
187  L8: MOV AH,2
188      MOV DL,'('
189      INT 21h
190      WORDTOS a,string
191      MOV DL','
192      INT 21h
193      WORDTOS b,string
194      MOV DL','
195      INT 21h
196      WORDTOS c,string
197      MOV DL,')'
198      INT 21h
199  L9: RETF
200  UCGEN ENDP
201  mycs ENDS
202  END   UCGEN

```

L7 labelinden sonra for loopları bitmiş oldu.

a kenarının ilk değeri 0 olarak atanmıştı 0 bir üçgen kenarı olamayacağından a değeri AX e atanarak 0 ile karşılatırıldı. a nın sıfır olması durumunda verilen kenarlarla üçgen oluşturulamaz hata mesajı yazdırıldı ve işlem L9 ile program sonuna gönderildi.

Çıktı karakter Karakter istendiğinden AH değerine 2 atandı.

Üçgen oluşturulabilmesi halinde a b c değerlerinde istedimiz üçgen kenarı değerlerini saklamış olduk ancak bu değerler hexadecimal word olarak saklandığından bu değerleri yazdırabilmek için WORDTOS makrosunu kullandık.

Çıktı istenilen şekilde yazdırılıp program sonlandırıldı.