

05.12.2019

YILDIZ TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



BLM2021 ALT SEVİYE PROGRAMLAMA 1.ÖDEV RAPORU

AHMET SAİD SAĞLAM

17011501

SORU 2 :

n elemanlı bir dizide değerleri 0 ile 1000 arasında değişen üçgen kenarı olmaya aday uzunlukları kullanıcıdan komut satırı aracılığıyla alan, bu dizideki kenar uzunluklarından, **en az uzunluktaki çevreye sahip olacak üçgeni oluşturan** kenarları ekrana yazdıran EXE tipinde assembly programını yazınız.

ÇÖZÜM :

```
001 Stacksg SEGMENT PARA STACK 'STACK'
002 DW 32 DUP(?)
003 Stacksg ENDS
004 Datasg SEGMENT PARA 'DATA'
005 Ucgendizi DW 100 DUP(0)
006 ELEMAN DW ?
007 MSG1 DB 'ELEMEN SAYISINI VERİNİZ:',0
008 MSG2 DB 'ELEMENLERİ GİRİNİZ:',0
009 MSG3 DB 'UYGUN KENAR YOK',0
010 MSG4 DB 'UYGUN KENARLAR:',0
011 MSG5 DB 'ÖNEMLİ NOT:LÜTFEN 3 BASAMAKLI SAYI GİRDİĞİNİZDE ENTER TUSUNA BASMAYINIZ!',0
012 Datasg ENDS
013 Codesg SEGMENT PARA 'CODE'
014 ASSUME CS:Codesg,DS:Datasg,SS:Stacksg
```

Kodun ilk kısmında öncelikle EXE tipi program yazmak için gerekli STACK SEGMENT tanımı yapılmaktadır. Daha sonra DATA SEGMENT kısmında ilk olarak programda bize gerekecek olan 100 elemanlı Ucgendizi isiminde WORD tipinde tüm elemanları 0 olan bir dizi tanımlanmıştır. Bu diziye kullanıcıdan alınan üçgen kenarı olmaya aday elemanlar atılacaktır. Dizinin WORD tipinde tanımlanmasının nedeni kullanıcının girebileceği değerlerin 0 ile 1000 arasında olmasından kaynaklıdır. Dizinin tüm elemanlarının 0 tanımlanmasının sebebi ise daha sonradan girilen değerlerin 100 elemanın hepsini doldurmaması durumunda sıralama işleminde bozulmaya yol açmamasıdır.

Sonraki adımda ELEMAN isimli, Ucgendizi'nin eleman sayısını tutacak olan WORD tipinde bir değişken tanıımıdır. WORD tipinde olmasının nedeni CX yazmacıyla yapacağımız işlemler sırasında tip uyumsuzluğu vermemesi amacıyla. 7. satırdan 11. satıra kadar olan kısımda ekrana yazdırılacak olan mesajlar tanımlanmıştır. Daha sonra ise CODE SEGMENT kısmındaki gerekli tanımlama ve işlemlerden sonra MAIN etiketiyle ana program yazılmaya başlanmıştır.

```

016 MAIN      PROC FAR
017             PUSH DS
018             XOR AX,AX
019             PUSH AX
020             MOV AX,DataSeg
021             MOV DS,AX
022             MOV AX,OFFSET MSG1
023             CALL PUT_STR
024             MOV DL,10
025             MOV AH,02H
026             INT 21H
027             MOV DL,13
028             MOV AH,02H
029             INT 21H
030             XOR AX,AX           ;Registerlar 0 lanir
031             XOR BX,BX
032             XOR CX,CX
033             XOR DX,DX
034             XOR SI,SI

```

22. satırda OFFSET sözde komutu yardımıyla MSG1 isimli değişkenin adresi AX yazmacına aktarılmıştır ve hemen sonraki satırdaki CALL komutu aracılığıyla PUT_STR yordamı çağırılarak AX yazmacındaki adresten itibaren bellekteki değişkenler NULL ASCII ile karşılaşılan dek ekrana yazdırılmıştır. Bu yazdırma işlemi PUT_STR yordamıyla alakalıdır ve raporun ilerleyen kısımlarında nasıl çalıştığı açıklanmıştır.

24. ve 29. satırlar arasında DL yazmacındaki ASCII değeri aracılığıyla ve AH yazmacına 02H değeri atanarak INT 21H komutuyla kesme yapılarak ekranda düzenli bir şekilde yeni satıra geçilmesi sağlanmıştır. Daha sonraki kısımda ise kullanılacak yazmaçlar sıfırlanmıştır.

```

036 ElemanSayi: MOV AH,00H
037             INT 16H
038             CMP AL,0DH
039             JE CİK
040             MOV AH,0EH
041             INT 10H
042             SUB AL,30H
043             XOR AH,AH
044             MOV BX,10
045             XCHG AX,CX
046             MUL BX
047             ADD AX,CX
048             XCHG AX,CX
049             JMP ElemanSayi

```

36. satırda dizinin eleman sayısını almak amacıyla yani kullanıcının kaç adet kenar değeri gireceğini öğrenmek için AH yazmacına 00H değeri atanarak ve INT 16H kesmesiyle klavyeden alınan ASCII değeri AL yazmacına atılmış ve hemen sonraki adımda CMP komutu vesilesiyle bu değer Enter tuşunun ASCII değeri olup olmadığı karşılaştırılmıştır çünkü eğer kullanıcı Enter tuşuna

bastıysa kullanıcıdan eleman alma işlemi sonlanmak isteniyor demektir. Bu durumda CIK etiketine dallanma gerçekleşir. Aksi durumda AL yazmacında bulunan kullanıcıdan alınan değerden SUB komutu yardımıyla 30H çıkarılır ve yine AL yazmacında tutulur. Bu işlemin sebebi alınan ASCII karakterin unsigned sayı değerine çevrilmesi gerekliliğindendir. Bu işlem kodun ilerleyen kısımlarında ihtiyaç oluşmasıyla birlikte tekrarlanacaktır. Daha sonra AH yazmacı, AL yazmacındaki değer ile AX yazmacı vasıtasıyla işlem yapılacağından sıfırlanmıştır. Hemen sonraki satırda BX yazmacına bir sonraki işlemde kullanılmak üzere 10 değeri atanmıştır.

45-48 satırları arasındaki işlemler kodun daha sonraki kısımlarında da kullanılmıştır. Öncelikle AX ve CX yazmaçlarının değerleri XCHG komutu yardımıyla yer değiştirilmiştir. Sonrasında MUL komutu yardımıyla AX yazmacında bulunan ama esasında CX yazmacına ait olan değer, BX yazmacıyla yani 10 ile çarpılmıştır. Hemen sonraki satırda da bu çarpım CX yazmacında bulunan ama esasında AL yazmacına ait olan ve kullanıcıdan alınan değer ile toplanmıştır. En son adımda ise AX ve CX yazmaçlarındaki değerler yeniden yer değiştirilerek CX yazmacında olması gereken değer yeniden atanmıştır. Tüm bu işlem bloğunun amacı kullanıcının değerleri giriş sırasına ve 10'luk sayı sistemine göre basamakları belirlemek ve nihai eleman sayısına ulaşmaktır.

49. satırda JMP komutu ile Elemansayi etiketine dönmüştür ve bu komut sayesinde kullanıcı Enter tuşuna basana dek bu işlem bloğu kendini tekrar edecektir.

Bu kısımda ödev dokümanında belirtildiği üzere kullanıcının dizinin boyutunu maksimum 100 vereceği kabul edilerek tam sayı, karakter veya girilen sayının büyüklüğü kontrol edilmemiştir. Programın kullanımı esnasında kullanıcının bu hususa dikkat etmesi gerekmektedir.

```

051 CIK:      MOV DL,10
052          MOV AH,02H
053          INT 21H
054          MOV DL,13
055          MOV AH,02H
056          INT 21H
057          MOV ELEMEN,CX
058          MOV AX,OFFSET MSG2
059          CALL PUT_STR
060          MOV DL,10
061          MOV AH,02H
062          INT 21H
063          MOV DL,13
064          MOV AH,02H
065          INT 21H
066          MOV AX, OFFSET MSG5
067          CALL PUT_STR
068          MOV DL,10
069          MOV AH,02H
070          INT 21H
071          MOV DL,13
072          MOV AH,02H
073          INT 21H
074          XOR AX,AX
075          XOR DX,DX
076          XOR BX,BX

```

CIK etiketinde 51-56, 60-65 ve 68-73 satırları arasındaki işlemler daha önce de belirtildiği üzere ekranda yeni satıra geçme ile alakalı düzenleme komutlarıdır. 57. satırda CX yazmacındaki eleman sayısı ELEMEN isimli değişkene atılmıştır. 58-59 ve 66-67 satırlarında ekrana gerekli mesajlar yazdırılmıştır. 74-76 satırları arasında da daha sonra kullanılacak yazmaçlar sıfırlanmıştır.

```

079 Basadon:  MOV AH,00H
080          INT 16H
081          CMP AL,0DH
082          JE Enter
083          JMP Karakter
084
085 Karakter:  CMP AL,47
086          JA Buyuk
087          JMP Basadon
088
089 Buyuk:    CMP AL,58
090          JB Kucuk
091          JMP Basadon
092
093 Kucuk:    MOV AH,0EH
094          INT 10H
095          SUB AL,30H
096          XOR AH,AH      ;Sadece al den alınan degeri kontrol eddebilmek icin ah 0 lanir
097          PUSH CX
098          MOV CX,10
099          XCHG AX,DX
100          PUSH BX      ;MUL isleminde DX yazmacinin degeri degistigi icin kısa sureligine BX
101          MOV BX,DX
102          MUL CX
103          MOV DX,BX
104          POP BX
105          ADD AX,DX
106          XCHG AX,DX
107          POP CX
108          INC BX
109          CMP BX,3      ;Basamak sayisi kontrolunu yapar
110          JE Enter
111          JMP Basadon

```

Basadon etiketi ile başlayan satırda kullanıcıdan kesme yardımıyla bir değer alınır ve bu değerın CMP komutu yardımıyla Enter tuşu olup olmadığı kontrol edilir. Eğer ki Enter ise Enter isimli etikete dallanma gerçekleşir. Aksi durumda Karakter isimli etikete dallanma olur. Karakter ve Büyük isimli etiketlerde kullanıcının klavyeden değer girerken rakamlar hariç herhangi bir tuşa basması durumunda Basadon etiketine dallanma yapılarak sanki o tuşa hiç basılmamışçasına bir etki oluşturulur. Böylece kullanıcı pozitif tam sayılar hariç hiçbir değeri giremez. Bu kontrol basılan tuşun ASCII değerleri karşılaştırılarak yapılır. Eğer tüm şartlar sağlanıyorsa Kucuk isimli etikete geçilir.

93-94 satırlarında kullanıcının girdiği değer ekrana yazdırılır. 95. satırdan itibaren daha önce AX, BX ve CX yazmaçlarıyla yapılan ASCII değeri unsigned değere çevirme ve basamak kontrolü işlemleri aynı şekilde yapılır ancak burada bazı farklar bulunmaktadır. İlk olarak burada CX ve BX yazmaçlarındaki değere ihtiyaç duyduğumuz için önce o değerler yığına atılır. CX yazmacına 10 değeri atılarak çarpma işleminde kullanılır. BX yazmacının kullanılma nedeni ise MUL işleminde AX yazmacıyla beraber DX yazmacının da değeri değiştiğinden kısa süreliğine DX yazmacının yerini alması gerekliliğindendir.

CX yazmacındaki orijinal değere döngü sayısını belirlemek adına ihtiyaç duyulurken BX yazmacındaki değere ise ödev dokümanında belirtildiği üzere kullanıcının 1000'e kadar sayı girebilme izninin kontrolü sırasında ihtiyaç duyulmaktadır. Bu kontrol şu şekilde işlemektedir : kullanıcı her tuşa bastığında BX yazmacının değeri 1 arttırılmıştır. Eğer yazmacın değeri 3 olursa Enter isimli etikete dallanma olur ve kullanıcıdan başka bir değer için tuşa basılması istenir. Böylece kullanıcı 3 basamaklıdan daha fazla basamağa sahip hiçbir sayı değerini giremez. Burada dikkat edilmesi gereken bir husus vardır. Eğer kullanıcı 3 basamaklı bir değer girerse son hamlede Enter tuşuna basmamalıdır çünkü program otomatik olarak 3. basamaktan sonra bir sonraki değeri almaya hazır konuma gelecektir.

```

113 Enter:      MOV  Ucgendizi[SI],DX
114             MOV  DL,10
115             MOV  AH,02H
116             INT  21H
117             MOV  DL,13
118             MOV  AH,02H
119             INT  21H
120             XOR  DX,DX
121             XOR  BX,BX
122             ADD  SI,2
123             LOOP Basadon

```

Enter isimli etikette daha önceki işlemlerde elde ettiğimiz ve DX yazmacında bulunan, kullanıcının girdiği üçgen kenarı olmaya aday uzunluk değeri Ucgendizi isimli dizinin ilk kontrolde, SI yazmacı yardımıyla ilk elemanına atılır ve sonrasında dizi WORD tipinde olduğu için SI 2 arttırılır. Böylece sonraki değer dizinin bir sonraki gözüne yerleştirilebilir hale gelir. 114-119 satırları ekranda yeni satıra geçmek amacıyla yazılmıştır. 120. ve 121. satırlarda ise sonraki adımda kullanılacak yazmaçlar sıfırlanır ve nihayetinde LOOP komutu ile Basadon etiketine dizinin eleman sayısı kadar dönülür. Böylece eleman sayısı kadar değer kullanıcıdan alınmış olunur.

```

125             MOV  CX,ELEMAN                      ;BUBBLE SORT
126             DEC  CX
127             XOR  AX,AX
128 L2:          PUSH CX
129             XOR  SI,SI
130             MOV  CX,ELEMAN
131             DEC  CX
132 L1:          MOV  AX,Ucgendizi[SI]
133             CMP  AX,Ucgendizi[SI+2]
134             JBE  Next
135             XCHG AX,Ucgendizi[SI+2]
136             MOV  Ucgendizi[SI],AX
137 Next:        ADD  SI,2
138             LOOP L1
139             POP  CX
140             LOOP L2

```

Bu blokta dizinin elemanları, SI yazmacı yardımıyla dizinin elemanlarına erişerek, CX yazmacı yardımıyla eleman sayısı kadar çift döngünün içinde ve AX yazmacı sayesinde Bubble Sort algoritmasına göre küçükten büyüğe sıralanmıştır. Bu sıralamanın nedeni en küçük çevreye sahip üçgeni yakalarken kullanılan algorithmada gerekli olmasındandır. Burada bir detay bulunmaktadır. Dizinin elemanları 0 lar dahil olmak üzere 100 tanedir ama sıralamada sadece

eleman sayısı kadar döngüye girilmiştir. Bu sayede Ucgendizi dizisinin içinde sadece kullanıcı tarafından verilen elemanlar kendi arasında sıralanmıştır. Üçgeni bulan algoritmada da bu husus göz önünde bulundurulmuştur.

```

142      XOR     DI,DI
143      XOR     DI,DI
144      XOR     DI,DI
145      XOR     DI,DI
146      XOR     DI,DI
147      ADD     DI,DI
148      MOV     DL,2
149      MOV     CX,ELEMAN
150      SUB     CX,2
151
152      L4:     PUSH    CX
153             MOV     CX,ELEMAN
154             SUB     CX,2
155             MOV     AX,SI
156             MOV     DI,DL
157             XOR     AH,AH
158             SUB     CX,AX
159             MOV     BX,Ucgendizi[SI]
160             ADD     DI,BX
161             CMP     DI,2
162             JAE     CIKIS
163             LOOP   L3
164             ADD     SI,2
165             MOV     DI,SI
166             ADD     DI,2
167             POP     CX
168             LOOP   L4
169
170      MOV     AX,OFFSET MSG3
171      CALL    PUTSTR
172      JMP     BITIS
173

```

142-146 satırlarında üçgeni bulmakta kullanılacak olan yazmaçlar sıfırlanır ve hemen akabinde DI yazmacına 2 eklenir. Bunun nedeni DI yazmacının SI yazmacının gösterdiği dizi elemanından bir sonrakini göstermesi isteğidir. Sonrasında DL yazmacına bölmede kullanmak amacıyla 2 değeri atanmıştır. Döngüde kullanılacak olan CX yazmacına eleman sayısının 2 eksiği bir değer atanır. Bunun sebebi ilk iki elemanı zaten en başta elimizde bulundurmamız ve kontrole hiç sokmamızdır. 152. Satırdan itibaren dış döngünün içinde iç döngüdeki çevrim sayısını bulmak amacıyla yazmaçlar kullanılarak işlemler yapılmıştır. Böylece her kontrolden sonra bir eleman ilerleyen dizide bir sonraki kontrolü etkileyecek bir taşma oluşmaz. Bu kontroller şu şekilde işlemektedir:

Dizinin ilk elemanından başlamak üzere bir eleman BX yazmacına atılır ve kendisinden bir sonraki elemandan başlayarak son eleman hariç diğer tüm elemanlarla küçüklük-büyüklük sırasına göre toplanır ve toplanan elemandan bir sonraki eleman ile karşılaştırılır. Dizinin elemanlarının küçükten büyüğe sıralanması burada faydalı olmuştur. Bu karşılaştırmadan sonra eğer toplam 3. elemandan büyükse CIKIS etiketine dallanma olur ve üçgen oluşmuş demektir.

Aksi halde döngü kendini tekrar eder. İç döngünün tamamlanmasının ardından dış döngü sayesinde aynı işlem 2. eleman ile ondan sonrakilerin toplamı ve karşılaştırılması şeklinde, uygun üçgenin bulunamaması durumunda sondan üçüncü elemana kadar devam eder. Eğer CIKIS etiketine dallanma gerçekleşmez ise uygun kenar değerleri girilmemiş demektir. Bu durumda AX yazmacı ve PUT_STR yordamı sayesinde ekrana uygun mesaj yazdırılır ve BITIS etiketine dallanma gerçekleşir ve de program sonlandırılır.

```
177 CIKIS:      XOR AX,AX
178             MOV AX,OFFSET MSG4
179             CALL PUT_STR
180             MOV DL,10                      ;\n
181             MOV AH,02H                     ;
182             INT 21H                         ;
183             MOV DL,13                      ;
184             MOV AH,02H                     ;
185             INT 21H                         ;
186             XOR AX,AX
187             MOV AX,Ucgendizi[SI]
188             CALL PUTN
189             MOV DL,10                      ;\n
190             MOV AH,02H                     ;
191             INT 21H                         ;
192             MOV DL,13                      ;
193             MOV AH,02H                     ;
194             INT 21H                         ;
195             MOV AX,Ucgendizi[DI-2]
196             CALL PUTN
197             MOV DL,10                      ;\n
198             MOV AH,02H                     ;
199             INT 21H                         ;
200             MOV DL,13                      ;
201             MOV AH,02H                     ;
202             INT 21H                         ;
203             MOV AX,Ucgendizi[DI]
204             CALL PUTN
205
206
207 BITIS:      RETF
208 MAIN       ENDP
```

Uygun kenarların bulunması durumunda ise kenarlar PUTN yordamı yardımıyla ekrana yazdırılır ve program biter.

```

211 PUTC
212
213
214
215
216
217
218
219
220 PUTC
PROC NEAR
PUSH AX
PUSH DX
MOV DL,AL
MOV AH,2
INT 21H
POP DX
POP AX
RET
ENDP

```

PUTC YORDAMI

Ekrana kesme kullanarak karakter yazdırmaya yarar. INT 21H kesmesi sayesinde AH yazmacında 2 değeri olduğu sürece, DL yazmacındaki değer ekrana yazdırılır. PUTC yordamı içinde AL yazmacındaki değer DL yazmacına aktarılır ardından da kesme kullanılarak ekrana karakter yazdırılır. Ek olarak AX ve DX yazmaçlarındaki değer bozulmaması amacıyla yığına atılmıştır. Bu işlem diğer kullanılan iki yordamda da , yordamın içinde kullanılan yazmaçlara bağlı olarak tekrarlanır.

```

223 PUTN
224
225
226
227
228
229 CALC_DIGITS:
230
231
232
233
234 DISP_LOOP:
235
236
237
238
239
240 END_DISP_LOOP:
241
242
243 PUTN
PROC NEAR
PUSH CX
PUSH DX
XOR DX,DX
PUSH DX
MOV CX,10
DIV CX
ADD DX,'0'
PUSH DX
XOR DX,DX
CMP AX,0
JNE CALC_DIGITS
POP AX
CMP AX,0
JE END_DISP_LOOP
CALL PUTC
JMP DISP_LOOP
END_DISP_LOOP: POP DX
POP CX
RET
ENDP

```

PUTN YORDAMI

AX yazmacında bulunan sayıyı PUTC yordamını da kullanarak ekrana onluk tabanda hane hane yazdırmaya yarar. İlk olarak CALC_DIGITS etiketi içinde AX yazmacında bulunan sayıyı CX yazmacı sayesinde 10'a böler ve kalan değerinin (DX yazmacında saklanır) ASCII değerini bularak onu yığına atar

(230-231.satırlar). Sonrasında bölüm değerini 0 ile kıyaslar. Eğer 0 ise bölme işlemi tamamlanmış demektir ve rakamları karakter karakter yazdırmak üzere DISP_LOOP etiketine geçer. Değil ise bölüm 0 olana kadar işlem tekrarlanır. DISP_LOOP etiketinde yığındaki ASCII değerleri AX yazmacına alınır ve PUTC yordamı yardımıyla sonda 0 ASCII değeri görene kadar ekrana sıra ile yazdırılır ve yordam bu şekilde tamamlanır.

```
247 PUT_STR      PROC NEAR
248              PUSH BX
249              MOV BX,AX
250              MOV AL,BYTE PTR [BX]
251 PUT_LOOP:     CMP AL,0
252              JE PUT_FIN
253              CALL PUTC
254              INC BX
255              MOV AL,BYTE PTR [BX]
256              JMP PUT_LOOP
257 PUT_FIN:      POP BX
258              RET
259 PUT_STR      ENDP
```

PUT_STR YORDAMI

AX yazmacında adresi verilen ve sonunda 0 olan karakter dizisini ekrana karakter karakter yazdırır. Sondaki sıfır değeri karakter dizisinin sonuna geldiğini anlamaya yarar ve yordamı sonlandırmak üzere dallanma yapılır.

```
263 Codesg      ENDS
264              END MAIN
```

CODE segmentle birlikte kod sona erer.

ÖRNEK ÇIKTILAR

Üçgen Oluşturan Kenar Değerleri Girilmesi ve Program Çıktısı

```
DOS
BOX
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
Run File [17011501.EXE]:
List File [INUL.MAP]:
Libraries [LIB]:

C:\>

C:\>debug 17011501.exe
-g
ELEMEN SAYISINI VERİNİZ:
9
ELEMENLERİ GİRİNİZ:
ÖNEMLİ NOT:LUTFEN 3 BASAMAKLI SAYI GİRDİĞİNİZDE ENTER TUSUNA BASMAYINIZ!
4
45
17
36
2
5
993
51
17
UYGUN KENARLAR:
2
4
5
```

Üçgen Oluşturmayan Kenar Değerleri Girilmesi ve Program Çıktısı

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
HWQSC  EXE      1,354 27-11-2019 12:48
HWQSC  OBJ      965 27-11-2019 12:47
LINK   EXE     64,982 31-07-1987  1:00
MAKE   BAT       57 05-02-2014  0:16
MAKE   EXE     17,024 16-10-1985 23:00
MASM   EXE    111,611 29-03-1995 11:52
RUN    BAT       52 14-02-2012 12:09
  18 File(s)      260,852 Bytes.
   2 Dir(s)      262,111,744 Bytes free.

C:\>debug 17011501.exe
-g
ELEMEN SAYISINI VERİNİZ:
6
ELEMENLARI GIRINIZ:
ONEMLI NOT:LUTFEN 3 BASAMAKLI SAYI GIRDIGINIZDE ENTER TUSUNA BASMAYINIZ!
10
500
300
958
40
67
UYGUN KENAR YOK
Program terminated normally
-
```