BLM 2021 ÖDEV RAPORU

Dersin Adı: Alt Seviye Programlama

Öğretim Üyesi: Dr. Öğretim Üyesi Erkan USLU

Öğrencinin Adı ve Soyadı: Seyyid İbrahim GÜLEÇ

Öğrencinin Numarası: 16011609

Soru 1 Açıklama:

İlk soruda istenen lena.pgm dosyasını sağa veya sola çeviren fonksiyonların içine inline assembly yazılarak resimleri döndürme işlemini tamamlamadır.

- Resmi sola döndürmek için kullandığım algoritma :

Ilk döngü
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16

. ilk döngü ilk aşama
4 2 3 16
5 6 7 8
9 10 11 12
1 14 15 13

ilk döngü ikinci aşama
4 8 3 16
5 6 7 15
2 10 11 12
1 14 9 13

ilk döngü son aşama
4 8 12 16
3 6 7 15
2 10 11 14
1 5 9 13

İkinci döngü
4 8 12 16
3 6 7 15
2 10 11 14
1 5 9 13

ikinci döngü döndü hali
4 8 12 16
3 7 11 15
2 6 10 14
1 5 9 13

- Resmi sola döndürmek için kullandığım algoritmanın C++ kodu

```
void rotateMatrix(int mat[][N])
{
    // Consider all squares one by one
    for (int x = 0; x < N / 2; x++)
    {
        // Consider elements in group of 4 in
        // current square
        for (int y = x; y < N-x-1; y++)
        {
            // store current cell in temp variable
            int temp = mat[x][y];

            // move values from right to top
            mat[x][y] = mat[y][N-1-x];
```

```
            // move values from bottom to right
            mat[y][N-1-x] = mat[N-1-x][N-1-y];

            // move values from left to bottom
            mat[N-1-x][N-1-y] = mat[N-1-y][x];

            // assign temp to left
            mat[N-1-y][x] = temp;
        }
    }
}
```

- Resmi döndürmek için yazdığım assembly kodu:

```
        XOR EDX,EDX;Bölmeden önce EDX yazmacını 0'lanır
        MOV AX,n; AX yazmaçına n degeri atılır
        MOV BX,2;  BX yazmaçına 2 değeri atılır
        DIV BX;  AX/BX bölme işlemi n/2 elde edilir
        XOR ECX, ECX;32 bitlik yazmaçlar kullandığım için 0'lama işlemi yaptım
        XOR ESI, ESI ;32 bitlik yazmaçlar kullandığım için 0'lama işlemi yaptım
        MOV CX,AX;  CX yazmaçına n/2 atılır çünkü dış döngü n/2 defa dönücek
        XOR EDI, EDI
        MOV EDI, resim;  resmin adresi EDI yazmaçına atılır
        XOR EAX, EAX; EAX yazmaçında x değeri tutulucağı için x=0 ataması yapılır
        XOR EBX, EBX;32 bitlik yazmaçlar kullandığım için 0'lama işlemi yapılır
        XOR EDX, EDX;32 bitlik yazmaçlar kullandığım için 0'lama işlemi yapılır
    L2: PUSH CX;  iç döngüde CX yazmaçı değişeceği için CX degeri stacke atılır
        XOR EBX, EBX;
        MOV EBX, EAX;  y = x ataması yapılır, y degeri EBX yazmacında tutulur
        XOR ECX, ECX
        MOV CX, n;  iç döngü y < n - x - 1 olduğu sürece dönücek bu yüzden CX yazmacına
önce n degeri atılır
        SUB CX, AX;  daha sonra n den x çıkartılır
        DEC CX;  ve n - x olan yazmaç 1 azaltılır ve n - x - 1 degeri oluşturulur daha sonra
aşağıda compare işlemi yapılmak üzere
    L1: PUSH CX;  stacke atılır
        PUSH EAX;  EAX yazmacında x değeri tutuluyordu üzerinde işlem yapılacak olduğu
için stacke atılır
        XOR ECX, ECX;
        MOV CX, n; CX yazmacına n degeri atılır çünkü matris programda
dizi gibi tutulduğu için mat[x][y] gibi bir değere ulaşmak için dizide
dizi[(n * x) + y] degerine gitmeliyiz;
        MUL ECX;  n * x işlemi yapılır değerin DX yazmacına taşmasını önlemek için 32 bitlik
yazmaç kullanılır
        ADD EAX, EBX;  (n * x) degerine y degeri eklenir
        ADD EAX, EAX;  dizi word dizisi olduğu için 2 ile çarpma yerine toplama işlemi
kullandım [(n * x) + y + (n * x) + y] degeri hesaplanır
        MOV CX, WORD PTR[EDI+EAX];  CX yazmacına mat[x][y] degeri atılır
        POP EAX;  x degeri stackten geri çekilir ve EAX yazmacında yine x degeri olur
        PUSH EAX;  x degeri üzerinde yeniden işlem yapılacağı için x tekrardan stacke atılır
        XOR EAX, EAX;  EAX yazmacı 0 lanır
        XOR ESI, ESI;  ESI yazmacı 0 lanır
        MOV AX, n;  EAX yazmacına n degeri atılır
        MOV SI, n;  ESI yazmacına n degeri atılır
        DEC SI;  SI yazmacında n - 1 degeri elde edilir
        SUB SI, BX;  SI yazmacında n - 1 - y değeri elde edilir
```

```
MUL ESI; n * (n - 1 - y)  işlemi yapılır çünkü matris dizi şeklinde
MOV ESI, EAX; ESI yazmacına çarpma sonucu elde edilen deger atılır
POP EAX; EAX yazmacına x degeri geri çekilir
ADD ESI, EAX; ESI yazmacına x degeri eklenir ve mat[n-1-y][x] degeri için indis degerinin yarısı elde edilir çünkü degerler Word şeklinde
ADD ESI, ESI; ESI toplama yapılarak 2 ile çarpılmış hali elde edilir ve mat[n-1-y][x] degerinin indis degeri edilmek üzeredir
MOV DX, WORD PTR[EDI+ESI]; DX yazmacına mat[n-1- y][x] degeri atılır
MOV WORD PTR[EDI+ESI], CX; CX yazmacında mat[x][y] degeri vardı bu deger mat[n-1-y][x] degerine atılır
MOV CX, DX; ve CX yazmacına mat[n-1-y][x] degeri atılır
PUSH EAX; x degeri üzerinde işlem yapılcak diye stacke atılır
XOR ESI, ESI; ESI degeri 0'lanır
MOV SI, n; ESI yazmacına n degeri atılır
DEC ESI; ESI yazmacında n - 1 değeri elde edilir
SUB ESI, EAX; ESI yazmacında n - 1 - x degeri elde edilir
XOR EAX, EAX; EAX yazmacı 0 lanır
MOV AX, n; n * (n - 1 - x ) degerini elde etmek için AX yazmacına n degeri atılır
MUL ESI; Çarpma işlemi yapılır ve n * ( n - 1 - x) değeri elde edilir
MOV ESI, EAX; EAX de elde edilen n * (n - 1 - x ) degeri ESI yazmacına atılır
XOR EAX, EAX; EAX yazmacı 0'lanır
MOV AX, n; AX yazmacına n degeri atılır
ADD ESI, EAX; ESI yazmacına n degeri eklenir ve n * (n - 1 - x) + n degeri elde edilir
DEC ESI; ESI yazmacı 1 azaltılır ve n * (n - 1 - x) + n -1 degeri elde edilir
SUB ESI, EBX; ESI yazmacından y degeri çıkartılır ve n * (n - 1 - x) + n -1 -y degeri elde edlir
ADD ESI, ESI; Word dizisi olduğu için kendi ile toplanır yani iki ile çarpılır
MOV DX, WORD PTR[EDI+ESI]; DX yazmacına mat[n-1-x][n-1-y] degeri atılır
MOV WORD PTR[EDI+ESI], CX; CX yazmacında mat[n-1-y][x] degeri vardı bu deger mat[n-1-x][n-1-y] degerine atılır
MOV CX, DX; ve CX yazmacına mat[n-1-x][n-1-y] degeri atılır
POP EAX; x degeri stackten geri çekilir
PUSH EAX; x degeri stacke geri atılır
XOR EAX, EAX; x degeri 0 lanır
MOV AX, n; EAX yazmacına n degeri atılır
MUL EBX; n * y işlemi yapılır
MOV ESI, EAX; ESI yazmacına elde edilen n*y degeri atılır
XOR EAX, EAX; EAX yazmacı 0 lanır
MOV AX, n; AX yazmacına n degeri atılır
ADD ESI, EAX; (n * y) + n degeri elde edilir
DEC ESI; (n * y) + n - 1 degeri elde edilir
POP EAX; x degeri stackten geri çekilir
SUB ESI, EAX; (n * y) + n - 1 - x degeri elde edilir
ADD ESI, ESI; Word dizisi olduğu için 2 ile çarpılır yani kendi ile toplanır
MOV DX, WORD PTR[EDI+ESI]; mat[y][n-1-x] degeri DX yazmacına atılır
MOV WORD PTR[EDI+ESI], CX; mat[n-1-y][n-1-x] degeri mat [y][n-1-x] degerine atılır
MOV CX, DX; CX yazmacında mat[y][n-1-x] degeri tutulur
PUSH EAX; x degeri stacke atılır
PUSH EBX; y degeri stacke atılır
XOR EBX, EBX; EBX yazmacı 0 lanır
MOV BX, n; BX yazmacına n degeri atılır
MUL EBX; (n * x) degeri elde edilir
POP EBX; y degeri stackten geri çekilir
ADD EAX, EBX; (n * x) + y degeri elde edilir
```

```
        MOV ESI, EAX; ESI yazmacına (n * x) + y degeri atılır
        ADD ESI, ESI; Word dizisi olduğu için 2 ile çarpılır yani kendi ile toplanır
        POP EAX; x degeri stackten geri çekilir
        MOV WORD PTR[EDI+ESI], CX; mat[x][y] degerine mat[y][n-1-x] degeri atılır
        INC EBX; y degeri döngü sonunda bir artar
        POP CX; Döngü kontrolü için n - x - 1 degeri stackten geri çekilir
        CMP BX, CX; y < n - x - 1 kontrolü yapılır
        JB L1; küçükse L1 etiketine geri döner ve iç döngü yeniden döner
        POP CX; iç döngüden çıktıktan sonra n/2 degeri stackten geri çekilir
        INC EAX; x degeri 1 artar
        CMP AX,CX; x < n/2 kontrolü yapılır
        JB L2; Küçükse L2 etiketine geri döner büyükse döngü biter ve döndürme işlemi
tamamlanmış olur.
```

• Resmi sağa döndürmek için yazdığım assembly kodu:

```
        XOR EDX,EDX; Bölme işlemi öncesi EDX yazmacı 0'lanır
        MOV AX,n; AX yazmacına n değeri atılır
        MOV BX,2; BX yazmacına 2 değeri atılır
        DIV BX; AX/BX işlemi yapılır işlem sonucunda AX yazmacında n/2 degeri elde edilir
        XOR ECX, ECX
        XOR ESI, ESI
        MOV CX,AX; CX yazmacına n/2 degeri atılır
        XOR EDI, EDI
        MOV EDI, resim; EDI yazmacına resmin adresi atılır
        XOR EAX, EAX; X degeri EAX yazmacında tutulur x = 0 ' dan başlar
        XOR EBX, EBX
        XOR EDX, EDX
    L2: PUSH CX; Dış döngüdeki kontrolde kullanılıcak olan CX degeri stacke atılır
        XOR EBX, EBX
        MOV EBX, EAX; Y degeri EBX yazmacında tutulur y = x ataması yapılır
        XOR ECX, ECX; ECX yazmacı 0'lanır
        MOV CX, n; CX yazmacına n degeri atılır
        SUB CX, AX; CX yazmacından x degeri çıkartılır ve n - x degeri elde edilir
        DEC CX; CX yazmacındaki değer 1 azaltılır ve n - x - 1 degeri elde edilir
    L1: PUSH CX; İç döngüde kullanılıcak olan CX degeri stacke atılır
        PUSH EAX; X degeri stacke atılır
        XOR ECX, ECX; ECX yazmacı 0 lanır
        MOV CX, n; CX yazmacına n degeri atılır
        MUL ECX; n * x işlemi yapılır sonuc DX'e yazmacına taşmasın diye CX yerine ECX
yazmacı kullanılır ve sonuc EAX yazmacında oluşur
        ADD EAX, EBX; (n * x) + y işlemi yapılır
        ADD EAX, EAX; Word dizisi olduğu için değer kendi ile toplanır yani iki ile çarpılmış
olur
        MOV CX, WORD PTR[EDI+EAX]; CX yazmacına mat[x][y] degeri atılır
        POP EAX; X degeri stackten geri çekilir

        PUSH EAX; X degeri stacke geri atılır
        MOV EAX,EBX; EAX yazmacına y degeri atılır
        XOR ESI, ESI; ESI yazmacı 0 lanır
        MOV SI, n; SI yazmacına n degeri atılır
        MUL ESI; (n * y) değeri hesaplanır sonuc EAX yazmacında oluşur
        MOV ESI, EAX; (n * y) degeri ESI yazmacına atılır
        XOR EAX, EAX; EAX yazmacı 0'lanır
```

```
MOV AX, n; AX yazmacına n degeri atılır
ADD ESI, EAX; (n * y) + n degeri elde edilir
DEC ESI; (n * y) + n - 1 degeri elde edilir
POP EAX; X degeri stackten geri çekilir
SUB ESI, EAX; (n * y) + n - 1 - x degeri elde edilir
ADD ESI, ESI; Word dizisi olduğu için ESI içindeki değer kendi ile toplanır yani iki ile çarpılmış olur
MOV DX, WORD PTR[EDI+ESI]; DX yazmacına mat[y][n-1-x] degeri atılır
MOV WORD PTR[EDI+ESI], CX; mat[y][n-1-x] degerine mat[x][y] degeri atılır
MOV CX, DX; CX yazmacına mat[y][n-1-x] degeri atılır

PUSH EAX; x degeri stacke atılır
XOR ESI, ESI; ESI yazmacı 0'lanır
MOV SI, n; SI yazmacına n değeri atılır
DEC ESI; n - 1 değeri elde edilir
SUB ESI, EAX; n - 1 - x değeri elde edilir
XOR EAX, EAX; EAX yazmacı 0'lanır
MOV AX, n; AX yazmacına n değeri atılır
MUL ESI; n * (n - 1 - x) değeri elde edilir
MOV ESI, EAX; ESI yazmacına EAX de oluşmuş olan n *(n - 1 - x) değeri atılır
XOR EAX, EAX; EAX yazmacı 0'lanır
MOV AX, n; AX yazmacına n değeri atılır
ADD ESI, EAX; n * (n - 1 - x) + n değeri elde edilir
DEC ESI; n * (n - 1 - x) + n -1 değeri elde edilir
SUB ESI, EBX; n * (n - 1 - x) + n - 1 - y değeri elde edilir
ADD ESI, ESI; Word dizisi olduğu için değer kendi ile toplanır
MOV DX, WORD PTR[EDI+ESI]; DX yazmacına mat[n-1-x][n-1-y] değeri atılır
MOV WORD PTR[EDI+ESI], CX; mat[n-1-x][n-1-y] değerine mat[y][n-1-x] değeri atılır
MOV CX, DX; CX yazmacına mat[n-1-x][n-1-y] değeri atılır
POP EAX; x değeri stackten geri çekilir

PUSH EAX; x değeri stacke geri atılır
XOR ESI, ESI; ESI yazmacı 0'lanır
MOV SI, n; SI yazmacına n değeri atılır
DEC ESI; n - 1 değeri elde edilir
SUB ESI, EBX; n - 1 - y değeri elde edilir
XOR EAX, EAX; EAX yazmacı 0'lanır
MOV AX, n; AX yazmacına n değeri atılır
MUL ESI; EAX yazmacında n * (n - 1 - y) değeri elde edilir
MOV ESI, EAX; n * (n - 1 - y ) değeri ESI yazmacına atılır
POP EAX; X değeri stackten geri çekilir
ADD ESI, EAX; n * (n - 1 - y ) + x değeri elde edilir
ADD ESI, ESI; Word dizisi olduğu için değer kendi ile toplanır yani iki ile çarpılır
MOV DX, WORD PTR[EDI+ESI]; DX yazmacına mat[n-1-y][x] değeri atılır
MOV WORD PTR[EDI+ESI], mat[n-1-y][x] değerine mat[n-1-x][n-1-y] değeri atılır
MOV CX, DX; CX yazmacına mat[n-1-y][x] değeri atılır


PUSH EAX; X değeri stacke atılır
PUSH EBX; Y değeri stacke atılır
XOR EBX, EBX; EBX yazmacı 0'lanır
MOV BX, n; BX yazmacına n değeri atılır
MUL EBX; EAX yazmacında (n*x) değeri elde edilir
```

```
POP EBX; Y değeri stackten geri çekilir
ADD EAX,EBX; (n * x) + y değeri elde edilir
ADD EAX,EAX; Word dizisi olduğu için oluşan değer kendi ile toplanır yani iki ile çarpılmış olur
MOV WORD PTR[EDI+EAX], CX; mat[x][y] değerine mat[n-1-y][x] değeri atılır
POP EAX; X değeri stackten geri çekilir



INC EBX; Y değeri 1 arttırılır
POP CX; n - x -1 değeri stackten geri çekilir
CMP BX, CX; y < n -x -1 kontrolü yapılır
JB L1; küçükse L1 etiketine geri döner ve iç döngü bi daha döner
POP CX; n/2 değeri stackten geri çekilir
INC EAX; X değeri 1 arttırılır
CMP AX,CX; x < n/2 kontrolü yapılır
JB L2; küçükse L2 etiketine geri döner ve dış döngü bi daha döner büyükse işlem tamamlanmıştır resim sağa dönmüştür.
```

- **SORU 2 AÇIKLAMA:**

```
    PAGE 60,80
    TITLE ornek____

STACKSG SEGMENT PARA STACK 'STACK'
    DW 32 DUP(?)
STACKSG ENDS

DATASG  SEGMENT PARA 'DATA'
CR      EQU 13
LF      EQU 10
MSG1  DB 'Dizinin eleman sayisini giriniz: ', 0
MSG2  DB CR, LF, 'Eleman giriniz :', 0
HATA   DB CR, LF, 'Dikkat sayi vermediniz yeniden giris yapiniz.!!!!', 0
HATA2 DB CR, LF, 'Dikkat girdiginiz sayi [-128,127] araasinda degil!!', 0
SIRALI DB CR, LF, 'Siralanmis dizi :', 0
ILKHAL        DB CR, LF, 'Girdiginiz dizi :', 0
BOSLUK        DB CR, LF, 44, 0
FIRST  DW ?
LAST   DW ?
ESAYISI       DB ?
DIZI    DB 100 DUP(?)
DATASG ENDS

CODESG  SEGMENT PARA 'CODE'
    ASSUME CS:CODESG, DS:DATASG, SS:STACKSG

ANA    PROC FAR
       PUSH DS
       XOR AX,AX
       PUSH AX

       MOV AX, DATASG
       MOV DS, AX
```

```
;-------------------------------
;         OKUMA ISLEMLERI   |
;-------------------------------
          MOV AX, OFFSET MSG1;  AX yazmacına MSG1 değeri atılır
          CALL PUT_STR; MSG1(Dizinin eleman bayisini giriniz :) değeri ekrana yazdırılır
          CALL GETN; Kullanıcıdan alınana değer AX yazmacına atılır
          MOV ESAYISI, AL; Kullanıcıdan alınan değer ESAYISI değişkenine atılır
          MOV CX, AX; Eleman sayısı kadar kullanıcıdan değer okunacağı için CX yazmacına AX
değeri atılır
          LEA SI, DIZI; Dizinin adresi SI yazmacına atılır

L1:       MOV AX, OFFSET MSG2; MSG2 AX yazmacına atılır
          CALL PUT_STR; MSG2(Eleman giriniz :) ekrana yazdırılır
          CALL GETN; Girilen değerler AX yazmacına atılır
          CMP AX, 127; Değer 127 den büyükse Hata verir ve değer yeniden girilir
          JG H1
          CMP AX, -128; Değer -128 den küçükse hata verir ve değer yeniden girilir
          JL H1
          MOV [SI], AL; Girilen değer dizinin ilgili adresine yazılır
          INC SI; dizinin adresi 1 artar (BYTE dizisi)
          LOOP L1; CX 0'dan büyük olduğu sürece döngü tekrar döner
          JMP HATA_ATLA; Döngüden çıkınca hata kısmına atlayabilmek için atlama yapılır
;----------------------
;           HATA        |
;----------------------
H1:       MOV AX, OFFSET HATA2; Hata mesajını AX yazmacına atar
          CALL PUT_STR; Hata mesajını ekrana yazdırır
          JMP L1; Tekrar döngüye döner

;-----------------------------------
;         DIZIYI EKRANA YAZDIRMA | Dizinin sıralanmadan önceki halini ekrana yazdırır
;-----------------------------------
HATA_ATLA:
          XOR CX, CX;  Döngüden önce CX yazmacı 0'lanır
          MOV CL, ESAYISI;  CL yazmacına Eleman sayısı atılır
          MOV AX, OFFSET ILKHAL ; (Girdiginiz dizi:) mesajı AX yazmacına atılır
          CALL PUT_STR; Mesaj ekrana yazdırılır
          LEA SI, DIZI; Dizinin adresi SI yazmacına atılır
L2:       MOV AL, [SI]; Dizinin ilgili elemanı AL yazmacına atılır
          CBW ; Convert Byte to Word kullanılır değerler AX şeklinde ekrana yazıldığı için negatif
sayı yerine pozitif sayı yazılmasını engellemek için
          CALL PUTN; AX yazmacındaki sayı ekrana yazdırılır
          CALL PUTC; Bir tane boşluk bırakır
          INC SI; SI yazmacının değeri bir artar
          LOOP L2; CX > 0 olduğu sürece döngü devam eder


;-----------------------
;         QUICKSORT |
;-----------------------
          XOR AX, AX ; AX yazmacı 0'lanır
          MOV AL, ESAYISI; AL yazmacına Eleman Sayısı atılır
          DEC AX; Eleman Sayısı - 1 elde edilerek dizinin son elemanını elde etmek amaçlanır

          LEA SI, DIZI; FIRST değişkenini dizinin başlangıç adresinden başladığı için aşağıda FIRST
değişkenine SI atanır
          MOV DI, SI; DI yazmaçına önce dizinin başlangıç adresi atılır
          ADD DI, AX; daha sonra AX eklenerek yani Eleman sayısı - 1 eklenerek dizinin son
elemanının adresi elde edilir
          MOV FIRST, SI; Dizinin ilk elemanının adresi
```

```
        MOV LAST, DI; Dizinin son elemanının adresi
        CALL QUICK; Quick yordamı çağrılıyor
;_____________________________________
;       DIZIYI EKRANA YAZDIRMA | Dizinin sıralanmadan sonraki halini ekrana yazdırır
;-----------------------------------
        XOR CX, CX;  Döngüden önce CX yazmacı 0'lanır
        MOV CL, ESAYISI;  CL yazmacına Eleman sayısı atılır
        MOV AX, OFFSET SIRALI ; (Girdiginiz dizi:) mesajı AX yazmacına atılır
        CALL PUT_STR; Mesaj ekrana yazdırılır
        LEA SI, DIZI; Dizinin adresi SI yazmacına atılır
L3:     MOV AL, [SI]; Dizinin ilgili elemanı AL yazmacına atılır
        CBW ; Convert Byte to Word kullanılır değerler AX şeklinde ekrana yazıldığı için negatif
sayı yerine pozitif sayı yazılmasını engellemek için
        CALL PUTN; AX yazmacındaki sayı ekrana yazdırılır
        CALL PUTC; Bir tane boşluk bırakır
        INC SI; SI yazmacının değeri bir artar
        LOOP L3; CX > 0 olduğu sürece döngü devam eder


        RETF
ANA     ENDP

GETC    PROC NEAR
;_____________________________________________________________
;  Klavyeden basılan karakteri AL yazmacına alır ve ekranda gösterir.   |
;  İşlem sonucunda AL etkilenir                                         |
;_____________________________________________________________
        MOV AH, 1h
        INT 21H
        RET
GETC    ENDP

PUTC    PROC NEAR
;_____________________________________________________________________________
;  AL yazmacındaki değeri ekranda gösterir. DL ve AH değişiyor. AX ve DX yazmaçlarının.  |
;  değerlerini korumak için PUSH/POP yapılıyor                                           |
;_____________________________________________________________________________
        PUSH AX
        PUSH DX
        MOV DL, AL
        MOV AH, 2
        INT 21H
        POP DX
        POP AX
        RET
PUTC    ENDP

GETN    PROC NEAR
;_____________________________________________________________________________________
;  Klavyeden basılan sayıyı okur, sonucu AX yazmacı üzerinden döndürür.
;  DX: sayının işaretli olup olmadığını belirler. 1 (+), -1 (-) demek
;  BL:  Hane bilgisini tutar
;  CX: okunan sayının işlenmesi sırasındaki ara değeri tutar
;  AL: klavyeden okunan karakteri tutar (ASCII)
;  AX: zaten dönüş değeri olarak değişmek durumundadır. Ancak diğer yazmaçları değerlerini
;  korumak zorundadır.
;_____________________________________________________________________________
        PUSH BX
        PUSH CX
        PUSH DX
```

```
GETN_START:
        MOV DX, 1
        XOR BX, BX
        XOR CX, CX
NEW:
        CALL GETC
        CMP AL, CR
        JE FIN_READ
        CMP AL, '-'
        JNE CTRL_NUM
NEGATIVE:
        MOV DX, -1
        JMP NEW
CTRL_NUM:
        CMP AL, '0'
        JB error
        CMP AL, '9'
        JA error
        SUB AL, '0'
        MOV BL, AL
        MOV AX, 10
        PUSH DX
        MUL CX
        POP DX
        MOV CX, AX
        ADD CX, BX
        JMP NEW
ERROR:
        MOV AX, OFFSET HATA
        CALL PUT_STR
        JMP GETN_START
FIN_READ:
        MOV AX, CX
        CMP DX, 1
        JE FIN_GETN
        NEG AX
FIN_GETN:
        POP DX
        POP CX
        POP BX
        RET
GETN  ENDP

PUTN  PROC NEAR
;-------------------------------------------------------------------
;  AX de bulunan sayıyı onluk tabanda hane hane yazdırır.
;  CX: haneleri 10'a bölerek bulacağız, CX = 10 olucak
;  DX: 32 bölmede işleme dahil olacak. Sonucu etkilemesin diye 0 olmalı
;-------------------------------------------------------------------
        PUSH CX
        PUSH DX
        XOR DX, DX
        PUSH DX


        MOV CX, 10
        CMP AX, 0
        JGE CALC_DIGITS
        NEG AX
        PUSH AX
```

```
        MOV AL, '-'
        CALL PUTC
        POP AX
CALC_DIGITS:
        DIV CX
        ADD DX, '0'
        PUSH DX
        XOR DX, DX
        CMP AX, 0
        JNE CALC_DIGITS
DISP_LOOP:
        POP AX
        CMP AX, 0
        JE END_DISP_LOOP
        CALL PUTC
        JMP DISP_LOOP
END_DISP_LOOP:
        POP DX
        POP CX
        RET
PUTN  ENDP

PUT_STR     PROC NEAR
;--------------------------------------------------------------------
;  AX de adresi verilen sonunda 0 olan dizgeyi karaktere karakter yazdırır.
;  BX dizgeye indis olarak kullanılır. Önceki değeri saklamalıdır
;--------------------------------------------------------------------
        PUSH BX
        MOV BX, AX
        MOV AL, BYTE PTR[BX]
PUT_LOOP:
        CMP AL, 0
        JE PUT_FIN
        CALL PUTC
        INC BX
        MOV AL, BYTE PTR[BX]
        JMP PUT_LOOP
PUT_FIN:
        POP BX
        RET
PUT_STR     ENDP

QUICK  PROC NEAR
;--------------------------------------------------------------------
;  Bir diziyi quick sort algoritması ile sıralar
;  Recursive bir yordamdır
;  FIRST ve LAST değişkenleri ile çalışır
;  Bu değerler data segmentte tanımlanmalıdır
;  Daha sonra yordamı çağırmadan önce FIRST değişkenine dizinin başlangıç adresi
;  LAST değişkenine de Dizinin son elemanının adresi atanmalıdır
;--------------------------------------------------------------------
        MOV SI, FIRST; SI yazmacına FIRST değeri atılır —- i indisini temsil eder
        MOV DI, LAST; DI yazmacına LAST değeri atılır   —- j indisini temsil eder
        MOV BX, SI; Pivot indisini tutan yazmaçtır
        CMP SI, DI;  FIRST ile LAST karşılaştırılır
        JAE SON; FIRST LAST'a eşit ise veya daha büyükse  döngüye girmeden yordam sonlanır
```

```
DONGU1:
;-------------------------------------------
;  while(dizi[i]<=dizi[pivot]&&i<last)
;           i++;
          MOV CL, [SI];
          MOV CH, [BX];
          CMP CL, CH
          JG ATLA1
          MOV DX, LAST
          CMP SI, DX
          JAE ATLA1
          INC SI
          JMP DONGU1
;-------------------------------------------
; -------------------------------------------
;  while(dizi[j]>dizi[pivot])
;           j--;

ATLA1:MOV CL, [DI]
          MOV CH, [BX]
          CMP CL, CH
          JLE ATLA2
          DEC DI
          JMP ATLA1
;-------------------------------------------
;-------------------------------------------
; if(i<j)
; {
;     temp=dizi[i];
;     dizi[i]=dizi[j];
;     dizi[j]=temp;
; }

ATLA2:CMP SI, DI
          JAE ATLA3
          MOV CL, [SI]
          MOV CH, [DI]
          MOV [SI], CH
          MOV [DI], CL
;-------------------------------------------
ATLA3:CMP SI, DI
          JB DONGU1 ; i < j olduğu sürece döngü devam eder
;-------------------------------------------
;     temp=dizi[pivot];
;     dizi[pivot]=dizi[j];
;     dizi[j]=temp;

          MOV CL, [BX]
          MOV CH, [DI]
          MOV [BX], CH
          MOV [DI], CL
;-------------------------------------------
;-------------------------------------------
; quicksort(dizi,first,j-1);

          PUSH DI
          MOV DX, LAST
          PUSH DX
          DEC DI
          MOV LAST, DI
```

```
        CALL QUICK
;-------------------------------------------
;-------------------------------------------
; quicksort(dizi,j+1,last);

        POP DX
        MOV LAST, DX
        POP DI
        INC DI
        MOV FIRST, DI
        CALL QUICK
;-------------------------------------------

SON:    RET
QUICK ENDP

CODESG  ENDS
      END ANA
```