ALT SEVİYE PROGRAMLAMA ODEV-1 RAPOR

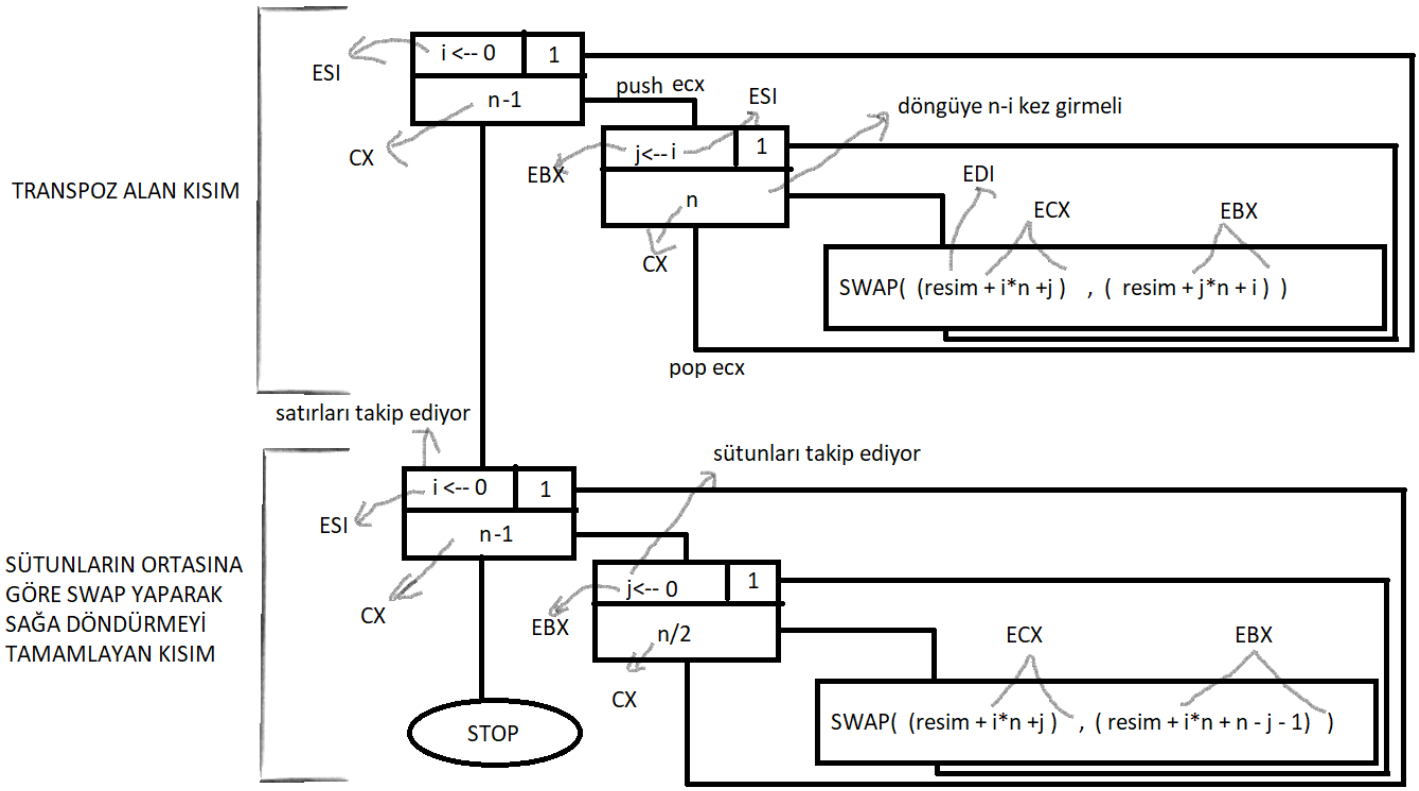
17011907 – AYŞE HİLAL DOĞAN

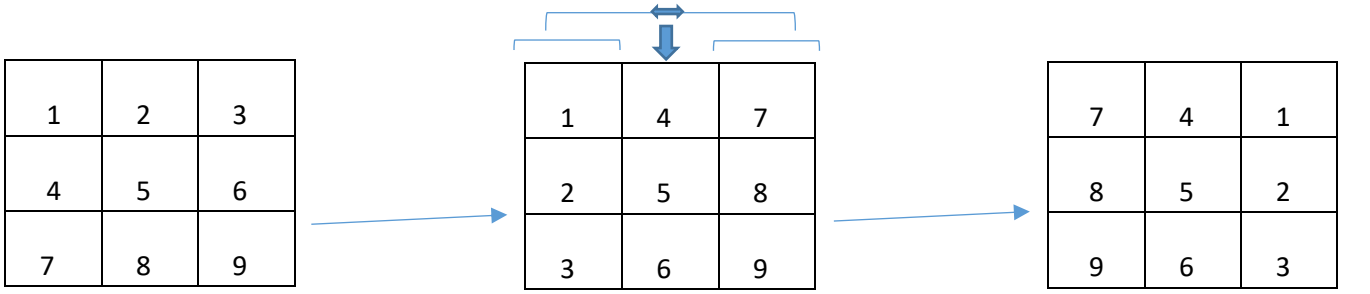
SORU1 –

Ekte verilen lena.zip sıkıştırılmış dokümanının main.cpp dosyasında SagaDondur() ve SolaDondur() "C" fonksiyonlarının içerisinde inline assembly kodu yazarak resimlerin seçilen tipe göre sağa ve sola döndürülmesini sağlayınız.

Sağa döndürme ve sola döndürme fonksiyonlarını hazırlarken; her ikisi için de önce transpoze aldım, daha sonra sağa ya da sola döndürmeye uygun şekilde indisleri hesaplayarak swap işlemini gerçekleştirdim.

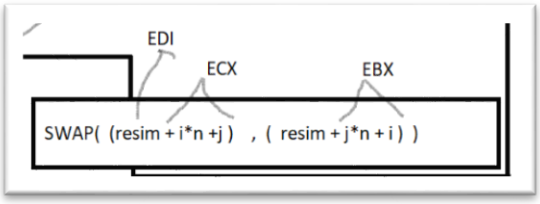
SAĞA DÖNDÜRME :





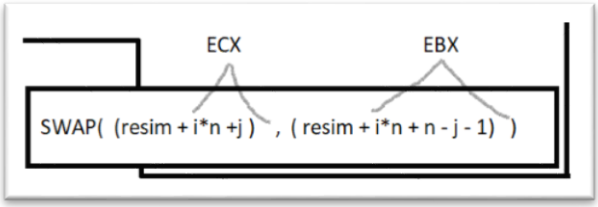
RESİM MATRİSİ TRANSPOZU ALINMIŞ RESİM SAĞA DÖNDÜRÜLMÜŞ RESİM

```
        void sagaDondur(short n, int resim) {
        //KODUNUZU BURADAN BASLAYARAK YAZINIZ
        __asm {
        //TRANSPOZ ALAN KISIM
        XOR ESI,ESI                ; indis olarak kullanacağım icin sıfırlıyorum ( i )
        XOR ECX,ECX                ; CX'e n'i atacağım.ECX'i sıfırlıyorum
        MOV CX,n                   ; 1.loop'un donme degeri (n-1)'i CX'e vereceğim. Once n verdim
        DEC CX                     ; CX = n-1 oldu
        MOV EDI,resim              ; resimin tutuldugu dizinin adresi EDI'da
                                   ; LOOP'a giriş
L1:     PUSH ECX                   ; Içteki loop için ECX yazmacını kullanacağım. Stack'e atiyorum
        MOV EBX,ESI                ; İçteki loopun indisi j , i 'den başlayacak. EBX' te j indisi var.
        MOV CX,n                   ; İçteki loopun dönme degerini ayarlayacağım.CX yazmacında tutacağım
        SUB ECX,ESI                ; İçteki loop'a n-i kez girmeli. CX'i i kadar azaltıyorum.
                                     Fakat ECX üzerinden çıkarıyorum. Registerlar uyumlu olmalı zaten
                                     ECX 'in yüksek anlamlı kısmını sıfırlamıştım.
            ; İçteki loop'a giriş
        L2:
ECX'te      XOR EAX,EAX            ; Swap içinde kullanacağım değerleri öncelikle EAX üzerinde
n*i+j                                belirleyeğim. EAX'i sıfırlıyorum.
oluşturma                          ; 1.belirleyeceğim değer ( i*n+j ) olacak
            MOV AX,n               ; AX'e n verdim
            MUL ESI                ; n*i ' yi gerçekleştiriyorum. ESI ' da i var. EAX' te n var.
            ADD EAX,EBX            ; EAX'te n*i var. EBX'te j var. (n*i) + j ;
            PUSH ECX               ; ECX'i bir takım işlemler için kullanacağım.Değerini bozmamak için
                                     Stack'e atıyorum.
            MOV ECX,EAX            ; ECX ' te EAX'teki değeri tutacağım. ; ECX'te ( n*i+j ) var
            SHL ECX,1              ; Dizi word boyutunda olduğu için belirlediğim değeri 2
                                     ile çarpıyorum
            XOR EAX,EAX            ; EAX'teki değeri ECX'e vermiştim. EAX'I başka şey için kullanacağım
EBX'te      MOV AX,n               ; AX'e n verdim
n*j+i       MUL EBX                ; n*j ' yi gerçekleştiriyorum. EBX ' de j var. EAX' te n var.
oluşturma   ADD EAX,ESI            ; EAX'te n*j var. ESI'da i var. (n*j) + i ;
            PUSH EBX               ; EBX j indisini tutuyor. Başka işlem için kullanacağım ve bu değeri
                                     de bozmamam lazım. Stack'e atıyorum.
            MOV EBX,EAX            ; EBX ' te EAX'teki değeri tutacağım. ; EBX'te ( n*j+i ) var
            SHL EBX,1              ; Dizi word boyutunda olduğu için belirlediğim değeri 2
                                     ile çarpıyorum
            ; 3 adımda [EDI+ECX] ve [EDI+EBX] in yerlerini değiştiriyorum
            ; Swap işleminde temp olarak AX'i kullanacağım
```




```
            MOV AX,WORD PTR[EDI+ECX]       ; AX'e (EDI + 2*(n*i+j)) adresindeki değeri veriyorum.
                                             ECX ' te 2*(n*i+j) var. Dizinin tipi word olduğu için
                                             Değere WORD PTR ile ulaşıyorum
SWAP        XCHG WORD PTR[EDI+EBX],AX      ; (EDI + 2*(n*j+i)) adresindeki değer ile AX'in
                                             değerlerini değiştiriyorum. AX' te ise
                                             (EDI + 2*(n*i+j)) adresindeki değer vardı.
                                           ; AX'te WORD PTR[EDI+EBX],
                                             WORD PTR[EDI+EBX]'te WORD PTR[EDI+ECX] var
            MOV WORD PTR[EDI+ECX],AX       ; WORD PTR[EDI+ECX]'e WORD PTR[EDI+EBX]'teki değeri ver

            POP EBX                ; EBX yazmacını stackten çek
            POP ECX                ; ECX yazmacını stackten çek
            INC EBX                ; indis değerini artir ( j )
            LOOP L2
```

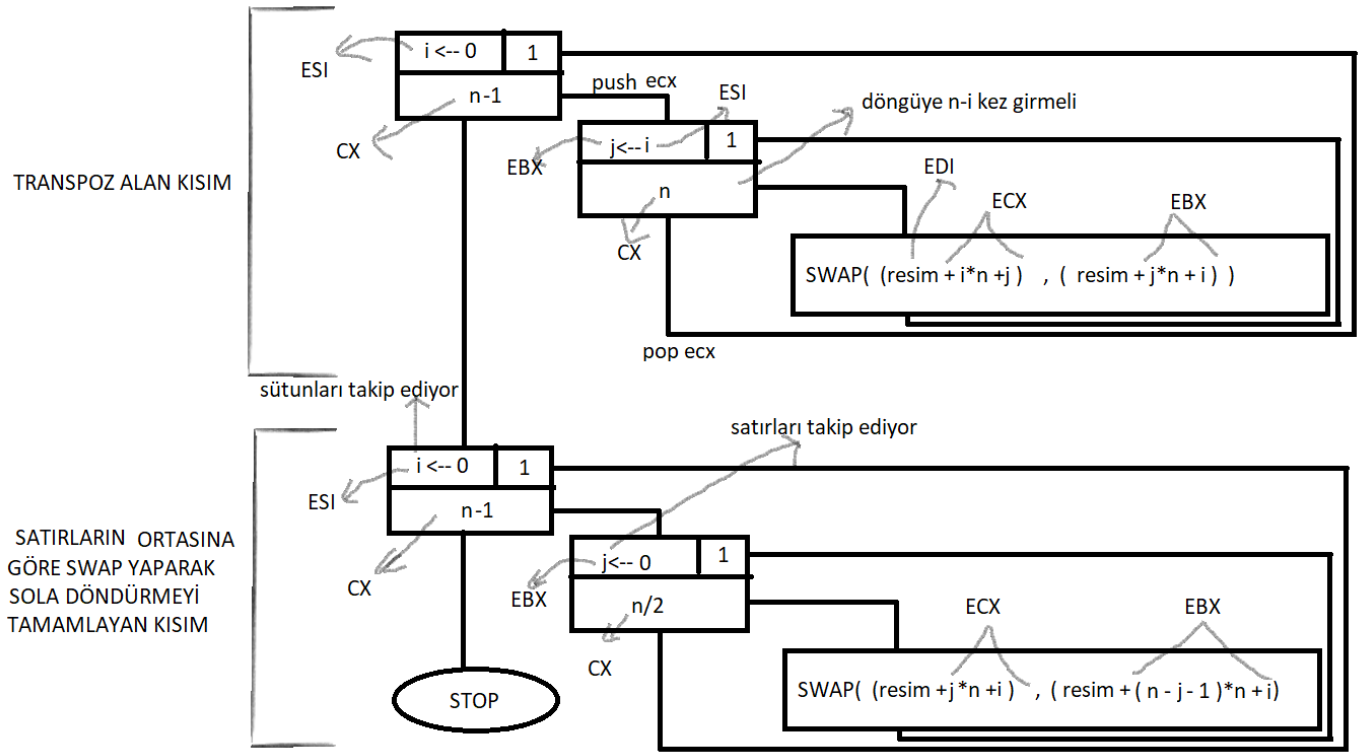
```
        INC ESI         ; Dış loopun indisini artir ( i )
        POP ECX         ; Dış loop için kullandığım ECX 'i stackten çek
        LOOP L1

        //sütunların ortasına göre swap yaparak saga dondurme işlemini tamamla

        XOR ESI, ESI            ; ESI' yı indis olarak kullacağım. ESI' yı sıfırlıyorum
        XOR ECX, ECX            ; CX donme değeri olacak. ECX'I sıfırla
        MOV CX,n                ; 1.loop'un donme degeri n CX'te.

  L3:   PUSH ECX                ; Içteki loop için ECX yazmacını kullanacağım. Stack'e atiyorum
        XOR EBX,EBX             ; Içteki loop indisini (j) sıfırlıyorum.
        MOV CX, n               ; Iç loopun donme değeri için CX kullanacağım. CX = n
        SHR CX,1                ; CX = n/2 (sütunların ortasına kadar gideceği için)

        L4:     XOR EAX, EAX    ; Swap içinde kullanacağım değerleri öncelikle EAX üzerinde
                                  belirleyeğim. EAX'i sıfırlıyorum.

ECX'te
n*i+j
oluşturma
                MOV AX, n       ; AX'e n verdim
                MUL ESI         ; n*i ' yi gerçekleştiriyorum. ESI ' da i var. EAX' te n var.
                ADD EAX,EBX     ; EAX'te n*i var. EBX'te j var. (n*i) + j ;
                PUSH ECX        ; ECX'i bir takım işlemler için kullanacağım. Değerini bozmamak için
                                  stack'e atıyorum.
                MOV ECX,EAX     ; ECX ' te EAX'teki değeri tutacağım. ; ECX'te ( n*i+j ) var
                SHL ECX,1       ; Dizi word boyutunda olduğu için belirlediğim değeri 2
                                  ile çarpıyorum

                XOR EAX, EAX    ; EAX'I başka bir şey için kullanacağım.Eski değerini sıfırlamalıyım.
                MOV AX, n       ; AX'e n verdim
EBX'te          MUL ESI         ; n*i ' yi gerçekleştiriyorum. ESI ' da i var. EAX' te n var.
n*i-n-j-1                       ; SWAP için n*i + n-j-1 'i hesaplamam gerekiyor. Başta n*i'yi EAX'te
oluşturma                         tutup n-j-1 i hesaplarken DX yazmacından yardım alıyorum
                XOR EDX,EDX     ; DX'e n vereceğim. EDX'I sıfırlıyorum
                MOV DX,n        ; DX = n
                ADD EAX,EDX     ; ( n*i ) + n  ; EAX' te n*i , EDX' te n vardı
                SUB EAX,EBX     ; ( n*i + n ) - j  ; EAX 'te n*i+n , EBX' te j vardı
                DEC EAX         ; ( n*i + n-j ) - 1  ; EAX 'te n*i+n-j vardı
                                ; Şuan EAX' te  ( n*i + n-j-1 ) var
                PUSH EBX        ; EBX j indisini tutuyor. Başka işlem için kullanacağım ve bu değeri
                                  de bozmamam lazım. Stack'e atıyorum.
                MOV EBX, EAX    ; EBX ' te EAX'te oluşturduğum değeri tutacağım.
                SHL EBX, 1      ; Dizi word boyutunda olduğu için belirlediğim değeri 2
                                 ile çarpıyorum
        ; 3 adımda [EDI+ECX] ve [EDI+EBX] in yerlerini değiştiriyorum
        ; Swap işleminde temp olarak AX'i kullanacağım
```

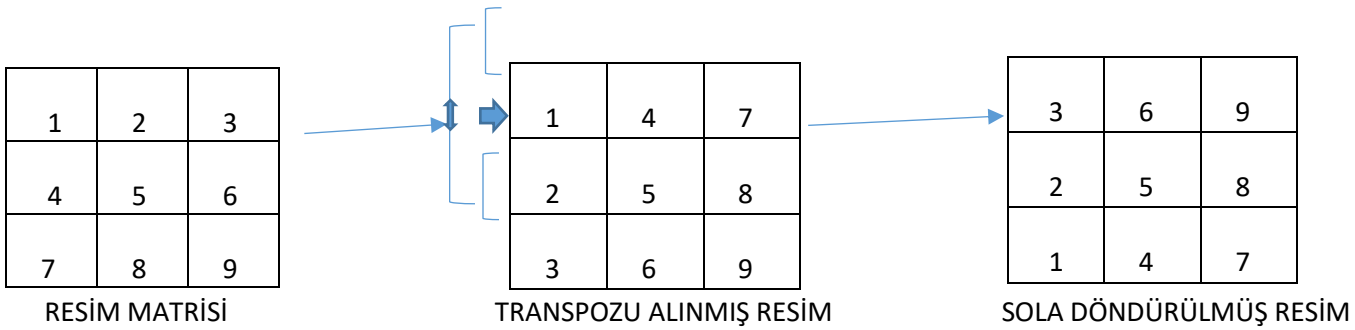



```
                MOV AX,WORD PTR[EDI+ECX]        ; AX'e (EDI + 2*(n*i+j)) adresindeki değeri veriyorum.
                                                  ECX ' te 2*(n*i+j) var. Dizinin tipi word olduğu için
                                                  Değere WORD PTR ile ulaşıyorum
SWAP            XCHG WORD PTR[EDI+EBX],AX       ; (EDI + 2*(n*i + n-j-1) ) adresindeki değer ile AX'in
                                                  değerlerini değiştiriyorum. AX' te ise
                                                  (EDI + 2*(n*i+j)) adresindeki değer vardı.
                                                ; AX'te WORD PTR[EDI+EBX],
                                                  WORD PTR[EDI+EBX]'te WORD PTR[EDI+ECX] var
                MOV WORD PTR[EDI+ECX],AX        ; WORD PTR[EDI+ECX]'e WORD PTR[EDI+EBX]'teki değeri ver
```

```
            POP EBX             ; EBX yazmacını stackten çek
            POP ECX             ; ECX yazmacını stackten çek
            INC EBX             ; indis değerini artir ( j )
            LOOP L4
        INC ESI       ; Dış loopun indisini artir ( i )
        POP ECX       ; Dış loop için kullandığım ECX 'i stackten çek
        LOOP L3
        }
        //KODUNUZU YAZMAYI BURADA BITIRINIZ
}
```

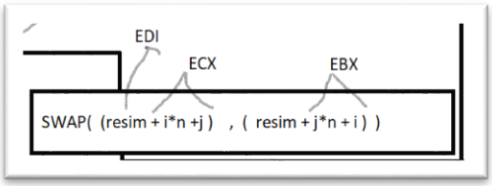
SOLA DÖNDÜRME:





RESİM MATRİSİ TRANSPOZU ALINMIŞ RESİM SOLA DÖNDÜRÜLMÜŞ RESİM

```
        void solaDondur(short n, int resim) {
        //KODUNUZU BURADAN BASLAYARAK YAZINIZ
        __asm {
        //TRANSPOZ ALAN KISIM
        XOR ESI,ESI                 ; indis olarak kullanacağım icin sıfırlıyorum ( i )
        XOR ECX,ECX                 ; CX'e n'i atacağım.ECX'i sıfırlıyorum
        MOV CX,n                    ; 1.loop'un donme degeri (n-1)'i CX'e vereceğim. Once n
                                      verdim
        DEC CX                      ; CX = n-1 oldu
        MOV EDI,resim               ; resimin tutuldugu dizinin adresi EDI'da
                                    ; LOOP'a giriş
L1:     PUSH ECX                    ; Içteki loop için ECX yazmacını kullanacağım. Stack'e
                                      atiyorum
        MOV EBX,ESI                 ; İçteki loopun indisi j , i 'den başlayacak. EBX' te j
                                      indisi var.
        MOV CX,n                    ; İçteki loopun dönme degerini ayarlayacağım.CX
                                      yazmacında tutacağım
        SUB ECX,ESI                 ; İçteki loop'a n-i kez girmeli. CX'i i kadar
                                      azaltıyorum.
                                      Fakat ECX üzerinden çıkarıyorum. Registerlar uyumlu
                                      olmalı zaten ECX 'in yüksek anlamlı kısmını
                                      sıfırlamıştım.
                                    ; İçteki loop'a giriş
        L2:     XOR EAX,EAX         ; Swap içinde kullanacağım değerleri öncelikle EAX
                                      üzerinde belirleyeceğim. EAX'i sıfırlıyorum.
                                    ; 1.belirleyeceğim değer ( i*n+j ) olacak
                MOV AX,n            ; AX'e n verdim
ECX'te          MUL ESI             ; n*i ' yi gerçekleştiriyorum. ESI ' da i var. EAX' te
n*i+j                                 n var.
oluşturma       ADD EAX,EBX         ; EAX'te n*i var. EBX'te j var. (n*i) + j ;
                PUSH ECX            ; ECX'i bir takım işlemler için kullanacağım.Değerini
                                      bozmamak için Stack'e atıyorum.
                MOV ECX,EAX         ; ECX ' te EAX'teki değeri tutacağım. ; ECX'te ( n*i+j) var
                SHL ECX,1           ; Dizi word boyutunda olduğu için belirlediğim değeri 2
                                      ile çarpıyorum
                XOR EAX,EAX         ; EAX'teki değeri ECX'e vermiştim. EAX'I başka şey için kullanacağım
EBX'te          MOV AX,n            ; AX'e n verdim
n*j+i           MUL EBX             ; n*j ' yi gerçekleştiriyorum. EBX ' de j var. EAX' te n var.
oluşturma       ADD EAX,ESI         ; EAX'te n*j var. ESI'da i var. (n*j) + i ;
                PUSH EBX            ; EBX j indisini tutuyor. Başka işlem için kullanacağım ve bu değeri
                                      de bozmamam lazım. Stack'e atıyorum.
                MOV EBX,EAX         ; EBX ' te EAX'teki değeri tutacağım. ; EBX'te ( n*j+i ) var
                SHL EBX,1           ; Dizi word boyutunda olduğu için belirlediğim değeri 2
                                      ile çarpıyorum
                ; 3 adımda [EDI+ECX] ve [EDI+EBX] in yerlerini değiştiriyorum
                ; Swap işleminde temp olarak AX'i kullanacağım
```

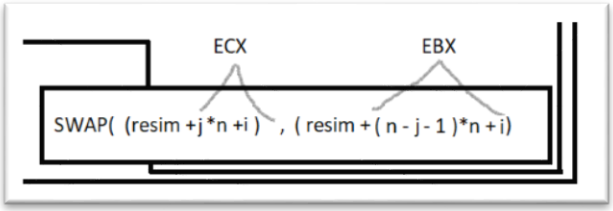



```
                MOV AX,WORD PTR[EDI+ECX]    ; AX'e (EDI + 2*(n*i+j)) adresindeki değeri veriyorum.
                                              ECX ' te 2*(n*i+j) var. Dizinin tipi word olduğu için
                                              Değere WORD PTR ile ulaşıyorum
SWAP            XCHG WORD PTR[EDI+EBX],AX   ; (EDI + 2*(n*j+i)) adresindeki değer ile AX'in
                                              değerlerini değiştiriyorum. AX' te ise
                                              (EDI + 2*(n*i+j)) adresindeki değer vardı.
                                            ; AX'te WORD PTR[EDI+EBX],
                                              WORD PTR[EDI+EBX]'te WORD PTR[EDI+ECX] var
                MOV WORD PTR[EDI+ECX],AX    ; WORD PTR[EDI+ECX]'e WORD PTR[EDI+EBX]'teki değeri ver
```

```
            POP EBX             ; EBX yazmacını stackten çek
            POP ECX             ; ECX yazmacını stackten çek
            INC EBX             ; indis değerini artir ( j )
            LOOP L2
        INC ESI       ; Dış loopun indisini artir ( i )
        POP ECX       ; Dış loop için kullandığım ECX 'i stackten çek
        LOOP L1
```

//satırların ortasına göre swap yaparak sola dondurme işlemini tamamla

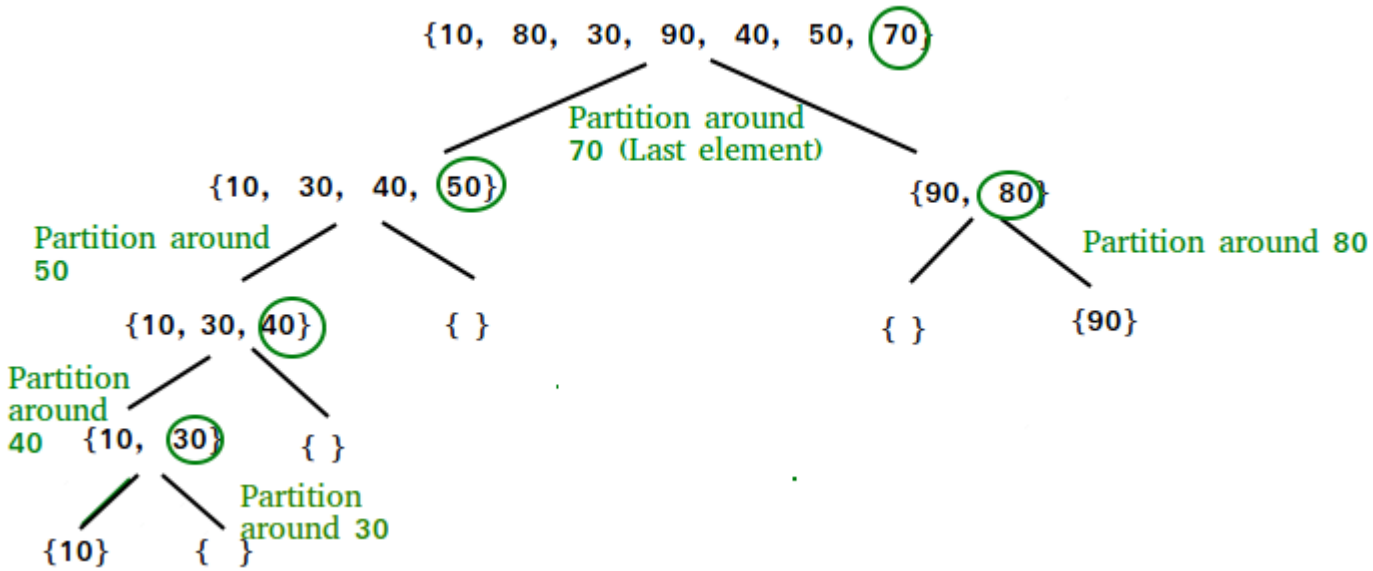
```
        XOR ESI, ESI          ; ESI' yı indis olarak kullacağım. ESI' yı sıfırlıyorum
        XOR ECX, ECX          ; CX donme değeri olacak. ECX'I sıfırla
        MOV CX,n              ; 1.loop'un donme degeri n CX'te.

  L3:   PUSH ECX              ; Içteki loop için ECX yazmacını kullanacağım. Stack'e atiyorum
        XOR EBX,EBX           ; Içteki loop indisini (j) sıfırlıyorum.
        MOV CX, n             ; Iç loopun donme değeri için CX kullanacağım. CX = n
        SHR CX,1              ; CX = n/2

        L4 :  XOR EAX, EAX    ; Swap içinde kullanacağım değerleri öncelikle EAX üzerinde
                              belirleyeğim. EAX'i sıfırlıyorum.

ECX'te
n*j+i
oluşturma
              MOV AX, n       ; AX'e n verdim
              MUL EBX         ; n*j ' yi gerçekleştiriyorum. EBX ' te j var. EAX' te n var.
              ADD EAX, ESI    ; EAX'te n*j var. ESI'da i var. (n*j) + i ;
              PUSH ECX        ; ECX'i bir takım işlemler için kullanacağım. Değerini bozmamak için
                              stack'e atıyorum.
              MOV ECX, EAX    ; ECX ' te EAX'teki değeri tutacağım. ; ECX'te ( n*j + i ) var
              SHL ECX, 1      ; Dizi word boyutunda olduğu için belirlediğim değeri 2
                              ile çarpıyorum

              ; ( (n-j-1)*n + i ) yi adim adim hesaplayip, bu değeri EBX'e vereceğim

              XOR EAX, EAX    ; EAX'I başka bir şey için kullanacağım.Eski değerini sıfırlamalıyım.
              MOV AX, n       ; AX'e n verdim

EBX'te
(n-j-1)*n+i
oluşturma
              SUB EAX,EBX     ; EAX'te n, EBX'te j vardı. EAX'te n-j oluştu
              DEC EAX         ; EAX'te n-j-1 oluştu
              XOR EDX,EDX     ; DX'e n vereceğim. EDX'I sıfırlıyorum.
              MOV DX,n        ; DX=n
              MUL EDX         ; EAX'te n-j-1 vardı, EDX'te n var. EAX'te (n-j-1)*n oluştu
              ADD EAX,ESI     ; ESI'da i var, EAX' te (n-j-1)*n + i oluştu
              PUSH EBX        ; EBX j indisini tutuyor. Başka işlem için kullanacağım ve bu değeri
                              de bozmamam lazım. Stack'e atıyorum.
              MOV EBX, EAX    ; EBX ' te EAX'te oluşturduğum değeri tutacağım.
                              ; EBX' te ( (n-j-1)*n + i )var
              SHL EBX, 1      ; Dizi word boyutunda olduğu için belirlediğim değeri 2
                               ile çarpıyorum

              ; 3 adımda [EDI+ECX] ve [EDI+EBX] in yerlerini değiştiriyorum
              ; Swap işleminde temp olarak AX'i kullanacağım
```



```
            MOV AX,WORD PTR[EDI+ECX]          ; AX'e (EDI + 2*(n*j+i)) adresindeki değeri veriyorum.
                                                ECX ' te 2*(n*j+i) var. Dizinin tipi word olduğu için
                                                Değere WORD PTR ile ulaşıyorum
SWAP        XCHG WORD PTR[EDI+EBX],AX           ; (EDI + 2*((n-j-1)*n + i ) adresindeki değer ile AX'in
                                                  değerlerini değiştiriyorum. AX' te ise
                                                  (EDI + 2*(n*j+i)) adresindeki değer vardı.
                                                ; AX'te WORD PTR[EDI+EBX],
                                                  WORD PTR[EDI+EBX]'te WORD PTR[EDI+ECX] var
            MOV WORD PTR[EDI+ECX],AX            ; WORD PTR[EDI+ECX]'e WORD PTR[EDI+EBX]'teki değeri ver

            POP EBX             ; EBX yazmacını stackten çek
            POP ECX             ; ECX yazmacını stackten çek
            INC EBX             ; indis değerini artir ( j )
            LOOP L4
        INC ESI       ; Dış loopun indisini artir ( i )
        POP ECX       ; Dış loop için kullandığım döngü değerini stackten çek
        LOOP L3
        }
        //KODUNUZU YAZMAYI BURADA BITIRINIZ
}
```

SORU2 –

Byte tipinde bir dizinin boyutunu ve pozitif ve negatif sayılardan oluşan değerlerini kullanıcıdan komut satırı aracılığıyla alan ve bu diziyi QuickSort yöntemiyle sıraladıktan sonra dizinin sıralanmış halini ekrana yazdıran EXE tipinde assembly programını yazınız.



Quicksort'ta sıralama mantığı;

Bir pivot seçip, o pivot'tan büyükler ve küçükler olmak üzere dizi ikiye bölünür. Bölünen dizilerde tekrar pivot ayarlanır. Tekrar dizi bölünür.. Bu şekilde adım adım dizinin sonuna kadar gidilir. Bu şekilde swap yapılarak dizi sıralanır.

```c
nt main()

    int dizi[100],n,i;
    printf("Eleman sayisini giriniz:");
    scanf("%d",&n);
    printf("\nSayilari girin:");

    for(i=0;i<n;i++)
        scanf("%d",&dizi[i]);

    quick_sort(dizi,0,n-1);
    printf("\nArray after sorting:");

    for(i=0;i<n;i++)
        printf("%d ",dizi[i]);

    return 0;
```

```c
void quick_sort(int dizi[],int low,int high)
{
    int j;
    if(low<high)
    {
        j=partition(dizi,low,high);
        quick_sort(dizi,low,j-1);
        quick_sort(dizi,j+1,high);

    }
}
```


```c
void quick_sort(int dizi[],int low,int high)
{
    int j;
    if(low<high)
    {
        j=partition(dizi,low,high);
        quick_sort(dizi,low,j-1);
        quick_sort(dizi,j+1,high);

    }
}
```


Stack segment tanımı

Data segment tanımı

Ekranda vereceğim mesajlar

Kullanacağım değişkenler

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


DATASG SEGMENT PARA 'DATA'
CR EQU 13
LF EQU 10
MSG1 DB 'Dizinin boyutunu belirleyiniz: ',0
MSG2 DB CR,LF, 'Dizinin elemanlarini giriniz: ',0
MSG3 DB CR,LF, 'Sayi giriniz: ',0
BASARILI DB CR,LF, 'dizi elemanlari verildi... ',0
MSG4 DB CR,LF, 'Dizinin siralanmis hali: ',0
MSG5 DB CR,LF, ' ',0
HATAMSJ DB CR,LF, 'Girdiginiz deger -128 ve 127 araliginda olmalidir, tekrar giriniz: ',0
HATA DB CR, LF, 'Dikkat !!! Sayi vermediniz yeniden giris yapiniz.!!! ',0

BOYUT DB ?
DIZI  DB 100 DUP(?)
DATASG ENDS
```

```
CODESG SEGMENT PARA 'CODE'                    ; Code segment tanımı
    ASSUME CS:CODESG, DS:DATASG, SS:STACKSG
ANA     PROC FAR            ;Ana kodun başlangıcı
        PUSH DS
        XOR AX,AX           Dönüş için gerekli olan değerler yığında saklanıyor
        PUSH AX
        MOV AX, DATASG      DATASG ismiyle tanımlı kesim alanına erişebilmek için gerekli tanımlar
        MOV DS,AX
      ; İşlemin gerçeklestirildiği kod bloğu
        MOV AX,OFFSET MSG1      Ekrana mesaj veren kısım ( Mesaj dizinin boyutunu istiyor ) ; Mesajı AX'e kaydediyor
        CALL PUT_STR            AX'teki mesajı ekrana yazdırıyor
        CALL GETN               ;Girilen sayıyı AX'e alır
        MOV BOYUT,AL            ;Boyut değişkenine AL'deki sayıyı verir. Boyut değişkeni AL'yi aşacak boyutta değildir
        MOV CX, AX              ;CX' e n veriyorum. Diziyi alırken lazım olacak
        XOR SI,SI               ;Diziyi alırken kullanacağım indis değerini sıfırlıyorum
        MOV AX,OFFSET MSG2      Ekrana mesaj veren kısım( Kullanıcının dizi elemanlarını girmesi için ) ;Mesajı AX'e kaydediyor
        CALL PUT_STR            AX'teki mesajı ekrana yazdırıyor
DIZIYIAL:                       ;Diziyi alacağım
        MOV AX,OFFSET MSG3      Ekrana mesaj veren kısım( Dizinin elemanlarını tek tek istiyor ); Mesajı AX'e alıyor
        CALL PUT_STR            AX'teki mesajı ekrana yazdırıyor
        CALL GETN           ; Alinan sayi AX de
        MOV BX,AX           ; Aldığım sayıyı BX'e alıyorum
      ;SAYI DOGRU ARALIKTA MI?
        CMP AX,-128         ; sayıyı AX'e alıyorum.Çünkü aralık dışında bir sayı girdiğinde AH'ye taşar.AL kullanırsam taşan sayıyı
                              göremem.Hata farkedilmez.
        JL  yanlis          ; Sayı -128 den küçükse hata mesajı vermek ve tekrar sayı girmesi için yönlendiriyorum
        CMP AX,127          ; Sayı 127'den büyük mü kontrol
        JG yanlis           ; Sayı 127 den büyükse tekrar sayı girmesi için yönlendiriyorum
        MOV DIZI[SI],BL     ; Buraya geldiyse doğru aralıkta sayı girilmiştir. Sayıyı diziye alıyorum
        JMP dogru           ; Yeni eleman alması için yönlendiriyorum
yanlis: MOV AX,OFFSET HATAMSJ   Ekrana hata mesajı veren ve tekrar sayı girmesini isteyen kısım
        CALL PUT_STR            AX'teki mesajı ekrana yazdırıyor
        JMP DIZIYIAL        ; Sayıyı yanlış girdiyse buraya geliyor. İndisi arttırmadan tekrar sayı girmesini istiyor
dogru:  INC SI              ; Sayı doğru girildiyse indisi artırıyor bir sonraki elemanı alacak
        LOOP DIZIYIAL
```

```
MOV AX, OFFSET BASARILI   Dizi verilen n sayısı kadar başarılı şekilde alınınca kullanıcıya bildiren mesajı ekrana yazan yer
CALL PUT_STR              AX'teki mesajı ekrana yazdırıyor
XOR SI,SI                 ; low icin SI kullaniyorum
XOR AX,AX                 ; AX'i sıfırlıyorum
MOV AL,BOYUT              ; Boyutu( n ) AL'ye alıyorum
MOV DI,AX                 ;high indisini DI'da tutuyorum   Quicksortu çağırırken high indisine ( n-1 )  yollamam gerekiyor, onu ayarlıyorum
```

```
quick_sort(dizi,0,n-1);
```

```
DEC DI                    ; DI = high-1
CALL QUICKSORT            ;Fonksiyonu çağırıyorum
MOV AX, OFFSET MSG4       Fonksiyondan döndüğünde dizi sıralanmış oluyor. Bunu kullanıcıya bildiren mesajı AX'e alıyor
CALL PUT_STR              AX'teki mesajı ekrana yazdırıyor
XOR CX,CX                 ; CX indisini kullanacağım, sıfırlıyorum
MOV CL,BOYUT              ; n'i CL'ye alıyorum
XOR SI,SI                 ;Loop için kullanacağım indisi sıfırlıyorum
```

DIZIYAZ:

```
MOV AX, OFFSET MSG5       Boşluk bırakmak için mesaj veren kısım
CALL PUT_STR
XOR AX,AX                 ; AX yazmacını sıfırla                                                                  Loop içinde
MOV AL,DIZI[SI]           ; Ekrana yazdıran PUTN fonksiyonu AL'deki mesajı yazdırdığı için yazdıracağım elemanı AL'ye alıyor   dizi eleman-
CALL PUTN                 ; Elemanları ekrana yazdırıyor                                                          larını alıyor
INC SI                    ; Dizi indisini arttırıyor
LOOP DIZIYAZ
RETF
ANA   ENDP                ; Ana fonksiyonu bitir
```

;QUICKSORT FONKSIYONU

```
void quick_sort(int dizi[],int low,int high)
{
    int j;
    if(low<high)
    {
        j=partition(dizi,low,high);
        quick_sort(dizi,low,j-1);
        quick_sort(dizi,j+1,high);

    }
}
```

QUICKSORT PROC NEAR

CMP SI,DI ; low ve high indislerini karşılaştırıyorum

JGE **CIKIS** ; Eğer low<high değilse Quicksortu bitiriyorum

PUSH SI ; İçeride low indisini bozmamak için her ihtimale karşı stack'e atıyorum

PUSH DI ; İçeride high indisini bozmamak için her ihtimale karşı stack'e atıyorum

CALL PARTITION ; Sonuç BX üzerinden dönüyor

MOV DX,BX ;BX'teki sonucu DX'e veriyorum

POP SI ;Partition fonksiyonundan sonra low indis değerini korumak için stack'e atıyorum

```
quick_sort(dizi,low,j-1);
```

PUSH DI ; Quicksort'u çağırırken low indisine j-1 vereceğim fakat şuanki high indis değerini korumam lazım.Stack'e atıyorum

MOV DI,DX ; DI = high = j

PUSH DI ; DI'nın j halindeki değerini koruyorum ; Quicksortu tekrar çağırırken SI' yı belirlerken bu değer lazım olacak

DEC DI ; DI = high = j - 1

CALL QUICKSORT ; DI = j - 1 ; SI sabit

POP SI ; Daha önce koruma altına aldığım j sayısını stack'ten çekiyorum

```
quick_sort(dizi,j+1,high);
```

INC SI ; low indisi olarak j+1 ' i yollamam gerektiği için j = j + 1

POP DI ; DI'nın koruduğum değerini geri alıyorum

CALL QUICKSORT ; DI sabit ; SI=j+1

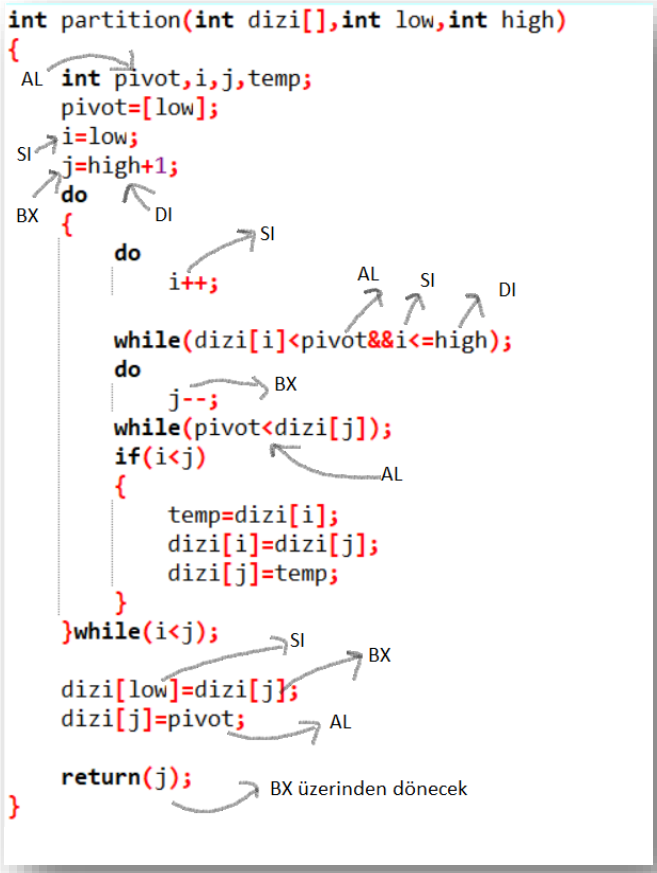
CIKIS:

POP SI ; Stack'teki SI'yı geri çekiyorum

RET

QUICKSORT ENDP ;Quicksort fonksiyonunu bitir

;PARTITION FONKSIYONU



PARTITION PROC NEAR **;** Partition fonksiyonu başlangıç

PUSH DI ;DI'nın (high indis değeri) değişmemesi için stack'e atıyorum

XOR AX,AX ;AX yazmacını sıfırla

MOV AL,DIZI[SI] ;AL:pivot SI:low olan indis degeri

PUSH SI ;SI'nın low için kullandığım değerini korumaya aldım. SI' yı (i) için kullanacağım. Sıfırlıyorum

MOV BX,DI ;BX'i j olarak kullanacağım. j = high + 1 olması gerekiyor

INC BX ; j = high + 1

doowhile:

do1: INC SI ; do 1 içinde olduğu müddetçe -> i ++

CMP DIZI[SI],AL ; dizi [i] < pivot ? ; AL'de pivot değeri vardı

JGE **do2** **;** do1 koşulunu sağlamıyorsa do2'ye geç

CMP SI,DI ; i <= high ? ; DI' da high indisi var

JG **do2** **;** do1 koşulunu sağlamıyorsa do2'ye geç

JMP do1 ; do1' e geri dön

do2: DEC BX ; do2 içinde olduğu müddetçe -> j --

CMP AL,DIZI[BX] ; pivot < dizi [j] ? ; AL'de pivot var, BX'te j

JGE **kontrol** **;** do2 koşulunu sağlamıyorsa kontrol'e (IF STATEMENT) geç

JMP **do2** ; koşulu sağlıyorsa do2 ' ye devam et

kontrol:

CMP SI,BX ; i < j ? ; SI: i , BX: j

JGE **devam** ;koşulu sağlamıyorsa aşağıdaki swap işlemini yapmadan devam et..

PUSH AX ; AX yazmacini swap icin kullanacagim, degerinin bozulmamasi icin stack'e atiyorum

; DIZI[i] ve DIZI[j] yi swap yapmak istiyorum fakat XCHG direk bunu yapamiyor, bir degiskenle bunu saglayacagim

```
if(i<j)
{
    temp=dizi[i];
    dizi[i]=dizi[j];
    dizi[j]=temp;
}
```

SWAP

MOV AL,DIZI[SI] ; AL=temp gibi kullaniyorum, gecici degisken

XCHG DIZI[BX],AL ; DIZI[i] ve DIZI[j] değerlerini swap

MOV DIZI[SI],AL ; CL'de en son DIZI[j] degeri kaldi, bunu ait oldugu yere (DIZI[i]) veriyorum

POP AX ; İşim bittiği için stack'ten geri çekiyorum

devam:

CMP SI,BX ; doowhile için şartımı kontrol ediyorum ; i < j ?

JGE **bitir** **;** şartı sağlamıyorsa doowhile ' dan çık

JMP **doowhile**

bitir:

POP SI ;Fonksiyon içinde SI'yı I olarak kullanmıştım. Şimdi asıl değerine (low) geri dönüyorum

PUSH CX ;CX'I temp gibi kullanıyorum, geçici değişken, değerini korumak için stack'e atıyorum

```
dizi[low]=dizi[j];
dizi[j]=pivot;

return(j);
```

SWAP

```
MOV CL,DIZI[BX]   ; DIZI[ j ] yi CL' ye alıyorum
MOV DIZI[SI],CL   ; CL' deki değeri ( DIZI[ j ] ) DIZI[ low ] ' a veriyorum.
MOV DIZI[BX],AL   ; AL : pivot  ; DIZI[ j ] = pivot
POP CX            ;CX' in koruduğum değerini stack'ten çekiyorum
;return j icin BX yazmacini kullaniyorum
POP DI            ; DI'nın koruduğum değerini stack'ten çekiyorum
RET
```

PARTITION ENDP ; Partition fonksiyonunu bitiriyorum

```
GETC PROC NEAR
MOV AH,1h
INT 21H
RET
GETC ENDP
```

Klavyeden basılan karakteri AL yazmacına alır ve ekranda gösterir

```
PUTC PROC NEAR
PUSH AX
PUSH DX
MOV DL, AL
MOV AH,2
INT 21H
POP DX
POP AX
RET
PUTC ENDP
```

AL yazmacındaki değeri ekranda gösterir. DL ve AH değişiyor. AX ve DX yazmaçlarının değerlerini korumak için push/pop yapılır

```
GETN PROC NEAR
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 DX
RET
GETN ENDP
```

Klavyeden basılan sayıyı okur, sonucu AX üzerinden döndürür

DX:sayının işaretli olup olmadığını belirler.

BL: hane bilgisini tutar

CX: okunan sayının işlenmesi sırasındaki ara değeri tutar

AL: klavyeden okunan karakteri tutar (ASCII)

AX zaten dönüş değeridir. Ancak diğer değişkenler korunmalıdır.

```
PUTN PROC NEAR
PUSH CX
PUSH DX
XOR DX,DX
PUSH DX
MOV CL,10
CMP AL,0
JGE CALC_DIGITS
NEG AL
PUSH AX
MOV AL, '-'
CALL PUTC
POP AX
CALC_DIGITS:
DIV CL
ADD AH, '0'
MOV DL,AH
PUSH DX
XOR AH,AH
CMP AL,0
JNE CALC_DIGITS
DISP_LOOP:
POP AX
CMP AL,0
JE END_DISP_LOOP
CALL PUTC
JMP DISP_LOOP
END_DISP_LOOP:
POP DX
POP CX
RET
PUTN ENDP
```

AX'de bulunan sayıyı onluk tabanda yazdırır.

CX: haneleri ona bölerek bulacağız. CX=10 olacak

DX: 32 bölmede işleme dahil olacak. Sonucu etkilemesin diye 0 olmalı

```
PUT_STR PROC NEAR
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
```

AX'de adresi verilen sonunda 0 alan dizgeyi karakter karakter yazdırır.

BX: dizgeye indis olarak kullanılır, önceki değeri saklanmalıdır

CODESG ENDS ; CODESG bittiği yer

END ANA ;Programın başlangıç notası