ALİ ASAF POLAT

16011079

*Yaptığım açıklamalar kod dosyamda da yazılıdır.

*Soru2’nin exe dosyası DosBox üzerinden çalıştığım için açılmamaktadır. Dosbox ortamında çalışacaktır.

```
void sagaDondur(short N, int resim) {
	//KODUNUZU BURADAN BASLAYARAK YAZINIZ
	__asm {
		//İlk işim asal köşegene göre transpozesini almak.
		//Sonrasında sağa veya sola çevirme durumlarına göre islem uygulamak.
		//Bu kısımda saga cevirme anlatılacaktır.
		//Saga cevirme: Transpozesini al.Sonrasında sütun sayısının yarısından itibaren simetri al.

		XOR ESI, ESI		//ESI :Dış döngünün i değeri.
		XOR EBX, EBX		//EBX :İç  döngünün j değeri.
		XOR EAX, EAX		//EAX :İşlemlerimi yaptığım register.
		XOR ECX, ECX		//ECX :Döngü değerlerimi tuttuğum register.
		XOR EDI, EDI		//EDI :Resimi atadığım register .


		MOV CX, N		//Döngü değerimi atadım.

		Lopp1 :

			XOR EAX, EAX		//AX registerı akümülator olarak kullanılacak.Sıfırladım.
			PUSH CX			//İçteki for için CX kullanacagım.Dış döngümü kaybetmemek icin yığına atama yaptım.
			MOV EAX, ESI		//Dış döngümün değeri N-i kadar olacak.
			SHR EAX, 1		//SI yi her turda 2 arttırdığım için ikiye böldüm.Amacım i değerini elde etmek.
			MOV CX, N		//CX registerına N değeri atandı.
			SUB ECX, EAX		//Şimdi N-i değeri gerçeklenmiş oldu.İç döngümün değeri artık belli.
```

```
XOR EBX, EBX          //İç döngümün değeri her turda i ye eşit olacak.
                      //Transpoze alan algoritmada j=i den başlıyorduk hatırlarsak.
MOV EBX, ESI          //j=i gerçeklendi.

Lopp2 :

XOR EAX, EAX          //EAX registerını her bir dönme için kullanacağım.Bu yüzden içinde değer bırakmıyorum.
XOR EDX, EDX          //EDX registerını her bir döngü için kullanacağım.İçinde değer bırakmıyorum.

MOV AX, N             //N*ESI+EBX BULUNDUGUM YERI GOSTERIYOR.Matris olarak düşünürsek N*i+j bulundugum yerdir.
MUL ESI               //N*ESI gerçeklendi.
ADD EAX, EBX          //N*ESI + EBX gerçeklendi.

PUSH CX               //EAX üzerinden işlem yapmaya devam edeceğim.
MOV ECX,EAX           //Bu sebeple ECX registerına EAX i alıp daha sonra tekrardan EAX e atayacağım.
XOR EAX,EAX           //EAX değerini kullanacağımdan içinde değer bırakmıyorum.
PUSH BX               //EBX registerı hatırladığımız gibi iç döngü elemanıydı.Ben EBX registarını
                      //yer değiştireceğim değişkenlerde hedef adresi gösteren olarak ayarlayacağım.
                      //Swap işlemi bittikten sonra iç döngü değerimin değişmesini istemediğim için yığına atıyorum.

MOV AX, N             //N*EBX+ESI GITMEK ISTEDIGIM YERI GOSTERIYOR.
MUL EBX               //Mantıken arr[i][j]=arr[j][i] yapacağım.Dolayısıyla gitmek istediğim adres N*EBX+ESI olmalı.
```

ADD EAX, ESI //N*EBX+ESI gerçeklendi.

MOV EBX,EAX //Hedef adresim EBX registera atandı.

MOV EAX,ECX //BULUNDUGUM YER ECX DEN GERI ALINDI.Daha önce saklamak için ECX'e almıştım hatırlarsak.

//**SWAP İŞLEMLERİ BAŞLIYOR.**

//EAX:Bulunduğum yeri işaret ediyor.

//EBX:Bulunduğum yer ile swap yapmak istediğim kısmı işaret ediyor.

XOR EDX, EDX //DX İ TMP OLARAK KULLANACAGIM.

MOV EDI, resim //EDI registerina resimi her atadığımda en başını gösteriyor.Ben sadece başından istediğim kadar ilerleyip

//swap işlemini gerçeklemek istiyorum.Yani EDI yı sadece resime ulaşmak için kullanıyorum.

ADD EDI, EBX //Resimdeki hedef adresime ulaşıyorum.

MOV DX, WORD PTR[EDI]//Hedef adresdeki değeri DX:tmp olarak alıyorum.

PUSH DX //Hedefteki değeri saklıyorum daha sonra bulunduğum yere atayacağım.

XOR DX, DX //DX registerı kullanacağım.İçinde değer bırakmıyorum.

XOR EDI, EDI

MOV EDI, resim //Resim EDI ya alındı.

ADD EDI, EAX //Bulunduğum yer EAX de saklanıyordu.Başlangıçtan EAX kadar ilerledim

MOV DX, WORD PTR[EDI]//Bulunduğum yerdeki değeri tmp olarak aldım.DİKKAT!:Daha önceden hedefi alıp yığına atmıştım.

XOR EDI, EDI

MOV EDI, resim //Resmin en başına gidiyorum.

ADD EDI, EBX //Hedef kadar ilerliyorum.

MOV WORD PTR[EDI], DX//Önceki adımda bulunduğum yerden aldığım DX değerini atıyorum.Swapın ilk adımı gerçekleşti.

POP DX //Hedeften alıp yığına attığım Tmp değerini çekiyorum.

XOR EDI, EDI

MOV EDI, resim //Resmin başlangıcından bulunduğum yer kadar ilerliyorum

ADD EDI, EAX

MOV WORD PTR[EDI], DX//Yığından çektiğim hedef değerini bulunduğum yere koyup swapı tamamlıyorum.

XOR ECX,ECX //CX değerini sıfırlıyorum.

XOR EBX,EBX //EBX değerini sıfırlıyorum.

POP BX //Yığına attığım iç döngü değişkenini çekiyorum böylelikle ara işlemlerde döngü değerim korunmuş oluyor.

POP CX //CX i EAX değerini saklanmak için yığına atmıştım.Çekiyorum.

ADD EBX, 2 //İç döngü değerimi 2 arttırıyorum.

LOOP Lopp2

POP CX //Dış döngü değerini çekiyorum.

ADD ESI, 2 //Dış döngü indisimi 2 artırıyorum.

```
                LOOP Lopp1
```

//TRANSPOZE TAMAM.

//Bu kısımda sağa çevirmek istediğimden matrisin ortasındaki sütününa göre simetri alacağım.

```
                //Kullanacağım registerları önceki değerleri (kalmış ise) sıfırlıyorum.

                XOR EAX,EAX             //EAX :Bulundugum yer olacak.

                XOR EBX,EBX             //EBX :İç döngü değerim olacak.İlerleyen kısımda ise hedef değerim olacak.
                XOR ECX,ECX             //ECX :Döngü değerim olacak.
                XOR EDX, EDX        //EDX :Tmp olarak kullanacağım.
                XOR EDI,EDI         //EDI :Resimin başlangıç adresini tutacak.
                XOR ESI,ESI         //ESI :Dış döngü değerim olacak.

                MOV CX,N            //Dış döngü değeri atandı.

        Lup1:
                PUSH CX                 //Dış döngü değerini kaybetmemek için yığına atıldı.
                MOV CX,N            //Matrisin ortadaki sütununa göre simetri alacaktım
                SHR CX,1            //N değerinin yarısı alınarak gerçeklendi.
                XOR EBX,EBX             //İç döngü her bittiğinde EBX değeri(indis) sıfırlanıyor.
        Lup2:
                XOR EAX,EAX             //EAX' bulunduğum yeri atacağım yani EAX=N*ESI+EBX(N*i+j)
                MOV AX,N
                MUL ESI                 //ESI*N gerçeklendi.
```

```
ADD EAX,EBX          //ESI*N+EBX gerçeklendi.**BURASI BULUNDUGUM YER**

PUSH CX              //EAX registerı işlemler için kullanılacağından ECX'i EAX değerini saklamak için kullanacağım.
                     //Döngü değerimi kaybetmemek için yığına atıyorum.

MOV ECX,EAX          //EAX üzerinden işlem yapacağım için ECX değerinde saklıyorum.

XOR EAX,EAX          //Artık EAX registerını sıfırlayabilirim.

MOV AX,N             //Önce EBX değeri belirlenecek.Sütunlar arası yer değişimi olacak yani arr[i][j]=arr[i][n-j-1]
                     //Buradaki n-j-1 değeri baktığımız zaman iç döngü değerine karşılık geliyor o halde EBX i güncellememem gerekli.
SHL EAX, 1           //Her elemanım 2 byte olduğundan N değerini 2 ile çarpıyorum aslında.
SUB EAX,2            //Haliyle -1 değerim de -2 olacak.
SUB EAX,EBX          //Şimdi ise -j değerini gerçekledim.
PUSH BX              //Bu işlemleri yaparken iç döngü değerimi kaybetmek istemiyorum.Bunu yığına atıyorum.
XOR EBX,EBX
MOV EBX,EAX          //Artık [n-j-1] değerim belli ve bunu EBX atıyorum.Hedef değerini hesaplarken bu indis kullanılacak.

XOR EAX,EAX          //EAX kullanılacak.Sıfırlıyorum.

MOV AX,N             //Hedef değerimi belirliyorum.
MUL ESI              //Dış döngü değerimle çarpıyorum. [i].
ADD EAX,EBX          //[n-j-1] değerimle topluyorum.
MOV EBX,EAX          //**ARTIK HEDEF ADRESİM BELLI**.
```

```
XOR EAX,EAX

MOV EAX,ECX          //Bulunduğum yeri ECX'e atmıştım o değeri geri çekiyorum.
XOR EDX, EDX    //DX Tmp olarak olarak kullanılacak.

MOV EDI, resim    //Resimin başlangıcı alındı.
ADD EDI, EBX    //Hedef adresime ulaştım.
MOV DX, WORD PTR[EDI]//Hedefteki değeri Tmp olarak aldım.
PUSH DX          //Ve yığına attım.
XOR DX, DX    //Tekrar tmp olarak kullanılcak.Sıfırlandı.
XOR EDI, EDI
MOV EDI, resim    //Resimin başlangıç adresi alındı.
ADD EDI, EAX    //Bulunduğum yer kadar ilerledim.
MOV DX, WORD PTR[EDI]//Bulunduğum yerdeki değer Tmp olarak alındı.

XOR EDI, EDI
MOV EDI, resim    //Resimin başlangıç adresi alındı.
ADD EDI, EBX    //Hedef değerime gidiyorum.
MOV WORD PTR[EDI], DX//Bulundugum yerden aldıgım degeri hedef değere atıyorum ve Swapın ilk adımı gerçekleşiyor.

POP DX          //Yığına attığım hedefdeki değer çekiliyor.

XOR EDI, EDI
MOV EDI, resim    //Resmin en başına gidiyorum
ADD EDI, EAX    //Bulundugum yer kadar ilerliyorum
MOV WORD PTR[EDI], DX//Bulundugum yere hedefteki değer atılıp Swap tamamlanıyor.

XOR ECX, ECX
XOR EBX, EBX
```

```
            POP BX                          //İç döngü indisini korumak için
yığına atmıştım.Çekiyorum.
            POP CX                          //ECX, EAX değerini korumak için
kullanılmıştı.İç döngü değerimi çekiyorum.
                ADD EBX, 2          //İç döngü indisini artırıyorum.

                LOOP Lup2

                POP CX                  //Dış döngü değerini çekiyorum.
                ADD ESI, 2          //Dış döngü indisini artırıyorum.

                LOOP Lup1









        }
        //KODUNUZU YAZMAYI BURADA BITIRINIZ
}

void solaDondur(short N, int resim) {
        //KODUNUZU BURADAN BASLAYARAK YAZINIZ
        __asm {
            //İlk işim asal köşegene göre transpozesini almak.
            //Sonrasında sağa veya sola çevirme durumlarına göre islem
uygulamak.
            //Bu kısımda sola cevirme anlatılacaktır.Transpoze alana kadar sağa
çevirmeyle aynı işlemler uygulandı.
            //Sola cevirme: Transpozesini al.Sonrasında satır sayısının yarısından
itibaren simetri al.
```

```
XOR ESI, ESI          //ESI :Dış döngünün i değeri.
XOR EBX, EBX          //EBX :İç  döngünün j değeri.
XOR EAX, EAX          //EAX :İşlemlerimi yaptığım register.
XOR ECX, ECX          //ECX :Döngü değerlerimi tuttuğum register.
XOR EDI, EDI          //EDI :Resimi atadığım register .


MOV CX, N             //Döngü değerimi atadım.


Lupp1 :


XOR EAX, EAX          //AX registerı akümülator olarak kullanılacak.Sıfırladım.
    PUSH CX                         //İçteki for için CX kullanacagım.Dış döngümü kaybetmemek icin yığına atama yaptım.
    MOV EAX, ESI      //Dış döngümün değeri N-i kadar olacak.
    SHR EAX, 1        //SI yi her turda 2 arttırdığım için ikiye böldüm.Amacım i değerini elde etmek.
    MOV CX, N         //CX registerına N değeri atandı.
    SUB ECX, EAX      //Şimdi N-i değeri gerçeklenmiş oldu.İç döngümün değeri artık belli.


    XOR EBX, EBX      //İç döngümün değeri her turda i ye eşit olacak.
                      //Transpoze alan algoritmada j=i den başlıyorduk hatırlarsak.
    MOV EBX, ESI      //j=i gerçeklendi.


    Lupp2 :


XOR EAX, EAX          //EAX registerını her bir dönme için kullanacağım.Bu yüzden içinde değer bırakmıyorum.
    XOR EDX, EDX      //EDX registerını her bir döngü için kullanacağım.İçinde değer bırakmıyorum.
```

MOV AX, N //N*ESI+EBX BULUNDUGUM YERI GOSTERIYOR.Matris olarak düşünürsek N*i+j bulundugum yerdir.

MUL ESI //N*ESI gerçeklendi.

ADD EAX, EBX //N*ESI + EBX gerçeklendi.

PUSH CX //EAX üzerinden işlem yapmaya devam edeceğim.

MOV ECX, EAX //Bu sebeple ECX registerına EAX i alıp daha sonra tekrardan EAX e atayacağım.

XOR EAX, EAX //EAX değerini kullanacağımdan içinde değer bırakmıyorum.

PUSH BX //EBX registerı hatırladığımız gibi iç döngü elemanıydı.Ben EBX registarını

//yer değiştireceğim değişkenlerde hedef adresi gösteren olarak ayarlayacağım.

//Swap işlemi bittikten sonra iç döngü değerimin değişmesini istemediğim için yığına atıyorum.

MOV AX, N //N*EBX+ESI GITMEK ISTEDIGIM YERI GOSTERIYOR.

MUL EBX //Mantıken arr[i][j]=arr[j][i] yapacağım.Dolayısıyla gitmek istediğim adres N*EBX+ESI olmalı.

ADD EAX, ESI //N*EBX+ESI gerçeklendi.

MOV EBX, EAX //Hedef adresim EBX registera atandı.

MOV EAX, ECX //BULUNDUGUM YER ECX DEN GERI ALINDI.Daha önce saklamak için ECX'e almıştım hatırlarsak.

//**SWAP İŞLEMLERİ BAŞLIYOR.**

//EAX:Bulunduğum yeri işaret ediyor.

//EBX:Bulunduğum yer ile swap yapmak istediğim kısmı işaret ediyor.

XOR EDX, EDX //DX İ TMP OLARAK KULLANACAGIM.

MOV EDI, resim //EDI registerina resimi her atadığımda en başını gösteriyor.Ben sadece başından istediğim kadar ilerleyip

//swap işlemini gerçeklemek istiyorum.Yani EDI yı sadece resime ulaşmak için kullanıyorum.

ADD EDI, EBX //Resimdeki hedef adresime ulaşıyorum.

MOV DX, WORD PTR[EDI]//Hedef adresdeki değeri DX:tmp olarak alıyorum.

PUSH DX //Hedefteki değeri saklıyorum daha sonra bulunduğum yere atayacağım.

XOR DX, DX //DX registerı kullanacağım.İçinde değer bırakmıyorum.

XOR EDI, EDI

MOV EDI, resim //Resim EDI ya alındı.

ADD EDI, EAX //Bulunduğum yer EAX de saklanıyordu.Başlangıçtan EAX kadar ilerledim

MOV DX, WORD PTR[EDI]//Bulunduğum yerdeki değeri tmp olarak aldım.DİKKAT!:Daha önceden hedefi alıp yığına atmıştım.

XOR EDI, EDI

MOV EDI, resim //Resmin en başına gidiyorum.

ADD EDI, EBX //Hedef kadar ilerliyorum.

MOV WORD PTR[EDI], DX//Önceki adımda bulunduğum yerden aldığım DX değerini atıyorum.Swapın ilk adımı gerçekleşti.

POP DX //Hedeften alıp yığına attığım Tmp değerini çekiyorum.

XOR EDI, EDI

MOV EDI, resim //Resmin başlangıcından bulunduğum yer kadar ilerliyorum

ADD EDI, EAX

MOV WORD PTR[EDI], DX//Yığından çektiğim hedef değerini bulunduğum yere koyup swapı tamamlıyorum.

XOR ECX, ECX //CX değerini sıfırlıyorum.

XOR EBX, EBX //EBX değerini sıfırlıyorum.

POP BX //Yığına attığım iç döngü değişkenini çekiyorum böylelikle ara işlemlerde döngü değerim korunmuş oluyor.

POP CX //CX i EAX değerini saklanmak için yığına atmıştım.Çekiyorum.

ADD EBX, 2 //İç döngü değerimi 2 arttırıyorum.

LOOP Lupp2

POP CX //Dış döngü değerini çekiyorum.

ADD ESI, 2 //Dış döngü indisimi 2 artırıyorum.

LOOP Lupp1

//TRANSPOZE TAMAM.

//Bu kısımda sola çevirmek istediğimden matrisin ortasındaki satıra göre simetri alacağım.

//Kullanacağım registerları önceki değerleri (kalmış ise) sıfırlıyorum.

XOR EAX, EAX //EAX :Bulundugum yer olacak.

XOR EBX, EBX //EBX :İç döngü değerim olacak.İlerleyen kısımda ise hedef değerim olacak.

```
XOR ECX, ECX            //ECX :Döngü değerim olacak.
XOR EDX, EDX            //EDX :Tmp olarak kullanacağım.
XOR EDI, EDI            //EDI :Resimin başlangıç adresini tutacak.
XOR ESI, ESI        //ESI :Dış döngü değerim olacak.

MOV CX, N               //Bu kısımda satır sayısının yarısına kadar gideceğim.
SHR CX,1                //Çünkü satırların ortasına göre simetri alıyorum.

Lup1 :
    PUSH CX                     //Dış döngü değerimi kaybetmek istemiyorum.Yığına atıldı.
MOV CX, N               //İç döngü değerim atandı.Tüm sütunlar gezilecek N atandı.
XOR EBX, EBX            //İç döngü değerim dış döngü her döndüğünde sıfırlanacak.
Lup2 :
XOR EAX, EAX            //Bulunduğum yeri belirleyeceğim.
MOV AX, N               //N*ESI+EBX (N*i+j) bulunduğum yerdir.
MUL ESI                 //N*ESI gerçeklendi
ADD EAX, EBX            //N*ESI+EBX gerçeklendi.**BURASI BULUNDUGUM YER**

    PUSH CX                     //EAX'deki değeri saklamam gerekecek.Döngü değerimi saklıyorum.

MOV ECX, EAX            //EAX işlem yapacağım için buradaki değerimi ECX de sakladım.İşlemler bittikten sonra tekrar EAX'e atılacak.

XOR EAX, EAX
```

```
MOV AX, N          //Satır üzerinde işlem yapacağım yani arr[i][j]=arr[n-i-1][j] olacak.
SHL EAX, 1         //Burada word tipinde çalıştığım için aslında 2*N yapmalıyım.
SUB EAX, 2         //-1 ise -2 değerinde olacak.(word)
SUB EAX, ESI       //-i değerini gerçekledim
PUSH BX            //EBX hedef değerimi gösterecek.Döngüdeki indis değerini kaybetmemesi için yığına atıyorum.
PUSH SI            //ESI değişen indis değerini yani(n-i-1) i gösterecek.Dış döngü değerini kaybetmek istemiyorum.

XOR ESI, ESI
MOV ESI, EAX       //ESI = (n-i-1) gerçeklendi.
XOR EAX, EAX

MOV AX, N          //Hedef değerimi hesaplayacağım.
MUL ESI            //Satır sayısıyla boyutu çarpıyorum.DİKKAT!:Değişen satır sayısıyla işlem yapıldı.[i][j] değil [n-i-1][j] yani.
ADD EAX, EBX       //N*ESI+EBX gerçeklendi.
MOV EBX, EAX       //EBX hedef değerim.***ARTIK HEDEF DEGERİ BELLI**
POP SI             //SI ile işim bitti dış döngü değerimi korumuş oldum.
XOR EAX, EAX

MOV EAX, ECX       //Bulunduğum yeri ECX de korumuştum.Çekiyorum.
XOR EDX, EDX       //DX Tmp olarak kullanılacak.

MOV EDI, resim     //Resmin başlangıcı alındı.
ADD EDI, EBX       //Hedef adresine gidildi.
MOV DX, WORD PTR[EDI]//Hedefteki değer Tmp olarak alındı..
PUSH DX            //Hedefteki değeri yığında saklıyorum.
XOR DX, DX
```

```
XOR EDI, EDI
MOV EDI, resim          //Resmin başlangıç adresine gidildi.
ADD EDI, EAX            //Bulundugum yere ilerledim.
MOV DX, WORD PTR[EDI]//Bulunduğum yerdeki değer Tmp olarak alındı.

XOR EDI, EDI
MOV EDI, resim          //Resmin başlangıç adresine gidildi.
ADD EDI, EBX            //Hedef konuma ilerledim
MOV WORD PTR[EDI], DX//Bulunduğum yerden aldığım değeri hedef değerine yazdım swapın ilk adımı gerçekleşti.

POP DX                  //Hedeften aldığım tmp değeri çekiyorum

XOR EDI, EDI
MOV EDI, resim          //Resmin başlangıç adresi alındı.
ADD EDI, EAX            //Bulundugum adrese gidildi.
MOV WORD PTR[EDI], DX//Hedeften aldığım değer atıldı.

XOR ECX, ECX
XOR EBX, EBX
POP BX                  //İç döngü indisim bozulmadan yığından alındı.
POP CX                  //ECX de EAX korunmuştu. Döngü değerimi geri çekiyorum.
ADD EBX, 2              //İç döngü indisini artırıyorum

LOOP Lup2

POP CX                  //Dış döngü değerim çekiliyor.
ADD ESI, 2              //Dış döngü indisim artırıldı.

LOOP Lup1
```

```
        }
        //KODUNUZU YAZMAYI BURADA BITIRINIZ

}
```

SORU2:

```
PAGE 60,80
TITLE QUICK-SORT PROGRAM

STACKSG SEGMENT PARA STACK 'STACK'
 DW 20 DUP (?)
STACKSG ENDS

DATASG SEGMENT PARA 'DATA'
        CR              EQU 13
        LF              EQU 10
        ISMIM DB 'ALI ASAF POLAT 16011079   ','$'
        MSG0 DB 'Dizi boyutunu belirleyiniz.',0
        MSG1 DB 'Dizi elemanini giriniz.',0

        HATA  DB 'Girdiginiz eleman -128,127 arasinda degil veya tamsayi degil!
Yeni giris:',0
        EKSI    DB '-','$'
        SUNUM1  DB '  Girdiginiz degerler: ','$'
        SUNUM2  DB '  Siralanmis hali: ','$'
        DIZI    DB 100 DUP(?)
        BOYUT           DB ?


DATASG ENDS
```

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

BASLA PROC FAR

                        PUSH DS                 ;DONUS ICIN GEREKLI
                        XOR AX,AX               ;OLAN DEGERLER
                        PUSH AX                 ;YIGINDA SAKLANIYOR.

                        MOV AX,DATASG   ;DATASG TANIMLI OLAN
                        MOV DS,AX               ;KISMA ERISMEK ICIN.

                        LEA DX,ISMIM            ;EKRANA ISMIMI BASMAK ICIN KESME.
                        MOV AH,09H
                        INT 21H

                        XOR AX,AX
                        XOR DX,DX
                        XOR CX,CX

                        MOV AX,OFFSET MSG0      ;DIZININ BOYUTUNU ISTEYEN MESAJ
                        CALL PUT_STR            ;YAZDIRILDI.
                        CALL GETN

                        MOV CL,AL               ;DIZININ BOYUTU CL'YE ATANDI.
                        MOV BOYUT,AL            ;AYRICA BOYUT DEGISKENINE
YERLESTIRILDI.
                        XOR AX,AX
                        XOR SI,SI               ;DİZİNİN BASLANGIC DEGERINE
GITTIM.
        DİZİ_LOOP:
```

```
                MOV AX,OFFSET MSG1
                CALL PUT_STR      ;MESAJ1'I GOSTER.
                XOR AX,AX         ;AX SIFIRLANDI.
                CALL GETN         ;DİZİ ELEMANINI OKU.
                MOV DIZI[SI],AL;ALINAN DEGER DIZININ ILGILI OFFSETINE
YAZILDI.
                INC SI
                XOR AX,AX         ;SAGLAMA ICIN.

                LOOP DİZİ_LOOP    ;ARTIK TUM ELEMANLARIM DIZIDE.



                LEA DX,SUNUM1     ;DOSBOX TERMINALDE
                MOV AH,09H        ;GIRILEN DEGERLER
                INT 21H                 ;YAZDIRMAK ICIN KESME



                XOR SI,SI         ;KULLANACAGIM REGISTERLARI
SIFIRRLIYORUM.
                XOR CX,CX
                XOR AX,AX

                XOR SI,SI         ;DİZİYİ EKRANA YAZDIRMAK ICIN.
                MOV CL,BOYUT      ;DIZININ BOYUTUNU ATIYORUM.

        YAZDIR:
```

```
                MOV AL,DIZI[SI]     ;YAZDIRMAK ISTEDIGIM ELEMANI
ATIYORUM.
                CMP AL,0            ;EGER SAYI NEGATIF ISE EKRANA '-'
BASTIRACAGIM
                JGE EKSIDEGER1      ;NEGATIF DEGILSE ZIPLATIYORUM
                LEA DX,EKSI ;EKRANA EKSI BASMAK ICIN KESME.
                MOV AH,09H
                INT 21H

                MOV AH,255 ;NEGATIF DEGERI 255 DEN ÇIKARTIP 1
EKLEDIGIMDE
                SUB AH,AL   ;POZITIF HALINI ALMIS OLUYORUM.EKRANA ZATEN
'-' BASILMIŞTI.
                MOV AL,AH   ;ÇIKARMA SONUCUNU AL'YE ATIP AH'I
SIFIRLIYORUM.
                XOR AH,AH
                INC AL              ;+1 EKLEME GERCEKLENDI.
EKSIDEGER1:
                CALL PUTN           ;YAZDIRACAK YORDAM CAGIRILDI.
                XOR AX,AX           ;ONCEKI DEGERIMI TEMIZLIYORUM.

                INC SI              ;BIR SONRAKI DIZI ELEMANINA GIDIYORUM.
                LOOP YAZDIR

                MOV BL,0            ;LOW DEGERIMI
                MOV BH,BOYUT        ;HİGH DEGERİM
                DEC BH                      ;QUICKSORTA GIDERKEN HIGH-1
DEGERIMI ATIYORUM.
                CALL QUICKSORT   ;REGISTERLAR UZERINDEN DEGER
GONDERMEK ISTIYORUM.



                LEA DX,SUNUM2           ;DOSBOX TERMINALDE 'SIRALANMIS
HALI'
```

```
                MOV AH,09H              ;YAZDIRMAK ICIN KULLANDIGIM
KESME
                INT 21H




                XOR SI,SI               ;KULLANACAGIM REGISTERLARI
SIFIRLIYORUM
                XOR CX,CX
                XOR AX,AX

        ;**QUICKSORT UYGULANDIKTAN SONRA TEKRAR EKRANA
YAZDIRIYORUM**

                MOV CL,BOYUT            ;DIZININ BOYUTUNU ATIYORUM.

    YAZDIR1:
                MOV AL,DIZI[SI]         ;YAZDIRMAK ISTEDIGIM ELEMANI
ATIYORUM.
                CMP AL,0
                JGE EKSIDEGER           ;EGER NEGATIF DEGILSE
ZIPLATIYORUM
                LEA DX,EKSI             ;EKRANA EKSI BASMAK ICIN KESME.
                MOV AH,09H
                INT 21H


                MOV AH,255              ;SAYI NEGATIFSE 255 DEN ÇIKARIP 1
EKLEDIGIMDE
                SUB AH,AL               ;POZITIF HALINI ELDE ETMIS
OLUYORUM.
                MOV AL,AH               ;ZATEN '-' YAZDIRILMIŞTI.
                XOR AH,AH               ;AH DEGERIMI
SIFIRLIYORUM.KARISIKLIK ONLENIYOR.
                INC AL                          ;+1 EKLEME GERCEKLENDI.
```

```
EKSIDEGER:

                CALL PUTN       ;YAZDIRACAK YORDAM CAGIRILDI.
                XOR AX,AX       ;ONCEKI DEGERIMI TEMIZLIYORUM.

                INC SI          ;BIR SONRAKI DIZI ELEMANINA GIDIYORUM.
                LOOP YAZDIR1

                ;*****************************DIZI ALMA KISMI BİTTİ SORT
KISMI BASLIYOR*****************************

        RETF
BASLA ENDP

QUICKSORT PROC NEAR

                CMP BL,BH       ;LOW VE HIGH DEGERLERINI
KARSILASTIRIYORUM
                JGE SON             ;EGER LOW BUYUK VEYA HIGH
DEGERINE ESITSE SON'A.
                PUSH BX             ;BH VE BL DEGERLERIMI
KAYBETMEMEK ICIN YIGINA ATIYORUM.

                CALL PARTITION
                XOR CX,CX
                MOV CL,AL       ;PARTITIONDAN DONEN AL DEGERINI CL YE
ATTIM.

                PUSH CX             ;CL DEGERINI DIGER QUICKSORTA
VEREBILMEK ICIN YIGINDA SAKLIYORUM.
```

```
                DEC CL                  ;ILK QUICKSORTA PI-1 DEGERI
GIDECEKTI.

                MOV BH,CL       ;HIGH DEGERIME PI-1 DEGERINI ATTIM.

                CALL QUICKSORT  ;QUICKSORT YORDAMINI CAGIRDIM.

                POP CX                  ;2.QUIKSORTTAN ONCE CL DEGERIMI
ALDIM.
                POP BX                  ;LOW VE HIGH DEGERLERIM GERI
ALINDI

                INC CL                  ;IKINCI QUICKSORTA PI+1 DEGERI
GIDECEKTI.
                MOV BL,CL       ;LOW DEGERIM PI+1 DEGERINI ALDI.
                CALL QUICKSORT  ;QUICKSORT YORDAMI TEKRAR ÇAĞIRILDI


SON:
                RET                     ;RET ILE BITIRIYORUM.

QUICKSORT ENDP

PARTITION PROC NEAR
                PUSH BX
                PUSH CX
                PUSH DX

                ;BL LOW
                ;BH HIGH
                ;DH PIVOT

                ;DI ->J
```

```
                ;SI ->İ
                XOR AX,AX       ;DI YA DOGRUDAN BH ATAMIYORUM

                MOV AL,BH       ;AL UZERINDEN ATACAGIM
                MOV DI,AX       ;HIGH DEGERIM DI YA ATANDI
                MOV DH,DIZI[DI] ;PIVOT DEGERIM ARTIK DH.
(PIVOT=ARR[HIGH])

                XOR AX,AX       ;BL DEGERINI SI'YA DOGRUDAN
ATAMADIGIM
                MOV AL,BL       ;ICIN AX REGISTERI UZERINDEN ATACAGIM.
                DEC AL                  ;BASLANGIC INDISI LOW-1=İ OLACAK
                CMP AL,0FFH             ;BURADAKI AL DEGERIM -1
OLUYORSA
                JNE FF_DEGIL    ;SI DEGERINI DE FFFF YAPMAK ISTIYORUM.
                MOV AH,AL

FF_DEGIL:    MOV SI,AX          ;SI DEGERIM INDEX DEGERI YANİ =i

                XOR CX,CX       ;DONGU DEGISKENI ATANACAK.
                MOV CL,BH       ;DONGU DEGERIM HIGH ATANDI.AMA HIGH-
LOW KERE DONMELI.
                SUB CL,BL       ;DONGU DEGERIM HIGH-LOW
GERCEKLENDI.



                XOR DI,DI       ;DI'YI KULLANACAGIM.SIFIRLADIM.
                XOR AX,AX       ;DONGUNUN İLK DEGERI OLAN LOW
DEGERINI
                MOV AL,BL       ;DI YA ATMALIYIM BUNUN ICIN AX
UZERINDEN
                MOV DI,AX       ;DI REGISTERINA GECIRDIM.

PART_LOOP:
```

```
                CMP DIZI[DI],DH     ;PIVOT DEGERIMLE DIZININ SIRADAKI
ELEMANI CMP.
                JG PART_SON             ;BUYUK ISE ISLEM YOK.
                INC SI              ;DEGIL ISE SI DEGERIM SI+1.
                XOR AX,AX
                MOV AL,DIZI[SI] ;SWAP ISLEMI YAPILACAK.
                MOV AH,DIZI[DI] ;DEGERLER ARTIK AX REGISTERDA
                MOV DIZI[SI],AH
                MOV DIZI[DI],AL ;SWAP GERCEKLESTI.
PART_SON:
                INC DI              ;DONGU INDISIMI ARTIRIYORUM(BYTE)
                LOOP PART_LOOP

                INC SI              ;LOOPTAN SONRAKI SWAP ISLEMI ICIN SI+1
                XOR DI,DI
                XOR AX,AX           ;DI YA DOGRUDAN BH ATAMIYORUM

                MOV AL,BH           ;AL UZERINDEN ATACAGIM
                MOV DI,AX           ;DI=BH BUNU ARR[HIGH]'A ERISMEK ICIN
KULLANACAGIM

                XOR AX,AX
                MOV AL,DIZI[SI]     ;SWAP ISLEMINI YAPACAGIM.
                MOV AH,DIZI[DI]     ;DEGERLERIM ARTIK AX UZERINDE.
                MOV DIZI[SI],AH
                MOV DIZI[DI],AL     ;SWAP ISLEMIM GERCEKLESTI.

                MOV AX,SI           ;RETURN DEGERI ICIN AX
DONDURUYORUM.


                POP DX                  ;YUKARIDA PUSHLADIGIM SEKILDE
TERS
```

```
                POP CX                  ;SIRAYLA TEKRAR POP ISLEMINI
GERCEKLESTIRIYORUM.
                POP BX


                RET                     ;YORDAMDAN CIKIYORUM.
PARTITION ENDP


;***DIZIYE ELEMAN ALMA YAZDIRMA KISMI BASLIYOR.***


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


PUTC PROC NEAR
                PUSH AX                 ;AX DX YAZMACLARININ
DEGERLERINI
                PUSH DX                 ;KORUMAK ICIN PUSH ISLEMI
YAPILIR.
                MOV DL,AL
                MOV AH,2
                INT 21H                 ;GEREKLI VERIYI YAZDIRMAK
ICIN YAPILAN ISLEMLER.
                POP DX
                POP AX
                RET
PUTC ENDP


PUT_STR PROC NEAR                       ;AX DE ADRESI VERILEN SONUNDA 0
OLAN
                                        ;DIZGEYI KARAKTER KARAKTER
YAZDIRIR.
```

```
                PUSH BX                 ;BX DIZGE INDISI OLARAK
KULLANILACAK
                MOV BX,AX               ;ADRESI BX E AL.
                MOV AL,BYTE PTR[BX]     ;AL'DE ILK KARAKTER VAR.
PUT_LOOP:   CMP AL,0
                JE PUT_FIN              ;0 GELDI ISE DIZGE SONA ERDI
DEMEK.
                CALL PUTC               ;AL'DEKI DEGERI EKRANA YAZAR.
                INC BX                  ;BİR SONRAKI KARAKTERE
GEC.
                MOV AL,BYTE PTR[BX]
                JMP PUT_LOOP            ;YAZDIRMAYA DEVAM ET.
PUT_FIN:
                POP BX                  ;BX'I GERI CEKIYORUM.
                RET
PUT_STR ENDP

GETN PROC NEAR                          ;KLAVYEDEN BASILAN DEGERI
OKUR.SONUC AX'DE DONER.
; DX:SAYININ ISARETLI OLUĞ OLMADIGINI BELIRLER.1(+) -1(-).
; BL:HANE BILGISINI TUTAR.
; CX:OKUNAN SAYININ ISLENMESI SIRASINDA ARADEGERI TUTAR.

; AL:KLAVYEDEN OKUNAN KARAKTERI TUTAR
; AX DONUS DEGERI OLARAK DEGISMEK DURUMUNDA FAKAT DIGER DEGERLER
; KORUNMALIDIR.

                PUSH BX
                PUSH CX
                PUSH DX
GETN_START:
                MOV DX,1                ;SAYININ POZİTİF OLDUGUNU
DUSUNELIM.
                XOR BX,BX               ;OKUMA YAPMADI HANE 0 OLUR.
```

```
                XOR CX,CX               ;ARA TOPLAM DEGERI 0 DIR.
NEW:
                CALL GETC               ;KLAVYEDEN ILK DEGERI AL'YE OKU.

                CMP AL,CR
                JE FIN_READ             ;ENTER TUSUNA BASILMIS ISE
OKUMA BITER.
                CMP AL,'-'              ;AL '-' GELDIYSE.
                JNE CTRL_NUM            ;GELEN SAYI 0-9 ARASI MI?
NEGATIVE:
                MOV DX,-1               ;- BASILDIYSA DX=-1
                JMP NEW
CTRL_NUM:
                CMP AL,'0'              ;SAYININ 0-9 ARASINDA OLDUGUNU
KONTROL ET.
                JB HATALI
                CMP AL,'9'
                JA HATALI               ;DEGIL ISE HATA MESAJI.
                SUB AL,'0'              ;RAKAM ALINDI HANEYI TOPLAMA
DAHIL EDIYOR.
                MOV BL,AL               ;BL YE OKUNAN DEGERI KOY.
                MOV AX,10               ;HANEYI EKLERKEN *10 YAPAR.
                PUSH DX                         ;MUL KOMUTUNDAN
ETKILENMEMESI ICIN.
                MUL CX
                POP DX
                MOV CX,AX               ;CX'DEKI ARADEGER 10 ILE
CARPILDI.
                ADD CX,BX               ;OKUNAN HANE ARA DEGERE
EKLENDI.
                JMP NEW                         ;KLAVYEDEN YENI BASILAN
DEGER ALINDI.
HATALI:
                MOV AX,OFFSET HATA
                CALL PUT_STR            ;HATA MESAJINI GOSTERIR.
```

```
                JMP GETN_START          ;O ANA KADAR YAPILANLARI UNUT
YENIDEN SAYI ALMAYA BASLA.

FIN_READ:
                MOV AX,CX               ;SONUC AX UZERINDEN DONECEK
                CMP DX,1                ;ISARETE GORE SAYI AYARLANACAK
                JE FIN_GETN
                NEG AX                          ;AX=-AX.

FIN_GETN:
                CMP AX,127              ;EGER SAYI 127 DEN BUYUKSE HATA
MESAJI ALACAK.
                JG  HATALI

                CMP AX,-128                     ;EGER SAYI -128DEN KUCUKSE
HATA MESAJI ALACAK.
                JL  HATALI

                POP DX
                POP CX
                POP BX

                RET
GETN ENDP

PUTN PROC NEAR                          ;AX DE BULUNAN SAYIYI EKRANA
YAZDIRIR.

                PUSH CX
                PUSH DX                         ;DX BOLUMDEN ETKILENMESIN
DIYE.
                XOR DX,DX
                PUSH DX                         ;KAC HANE ALACAGIMIZI
BILMEDIGIMIZDEN
```

```
                                        ;YIGINA 0 KOYUP ONU ALANA
KADAR DEVAM EDELIM.

                MOV CX,10
                CMP AX,0
                JGE CALC_DIGITS
                NEG AX                  ;SAYI NEG İSE POZITIF YAPILIR
                PUSH AX                         ;AX I SAKLA
                MOV AL,'-'              ;ISARETI EKRANA YAZDIR.
                CALL PUTC
                POP AX                          ;AX I GERI AL
CALC_DIGITS:
                DIV CX                          ;BOLUM=AX KALAN=DX
                ADD DX,'0'              ;KALAN DEGERI ASCII OLARAK BUL.
                PUSH DX
                XOR DX,DX
                CMP AX,0                ;BOLEN 0 ISE ISLEM BITTI DEMEKTIR.
                JNE CALC_DIGITS         ;ISLEMI TEKRARLA.

DISP_LOOP:
                ;YAZILACAK TUM HANELER YIĞINDA.SIRA SIRA ALALIM.
                POP AX
                CMP AX,0                ;AX=0 OLDUGUNDA SONA ERDI
DEMEKTIR.
                JE END_DISP_LOOP
                CALL PUTC               ;AL'DEKI DEGERI YAZ.
                JMP DISP_LOOP           ;ISLEMI TEKRARLA
END_DISP_LOOP:

                POP DX                          ;YIGINDAN DEGERLERIMI
ALIYORUM.
                POP CX
                RET
```

```
PUTN ENDP

CODESG ENDS
END BASLA
```