YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ALT SEVİYE PROGRAMLAMA 2. ÖDEV RAPORU

DERS YÜRÜTÜCÜSÜ: FURKAN ÇAKMAK

20011042

ABDULKADİR TÜRE

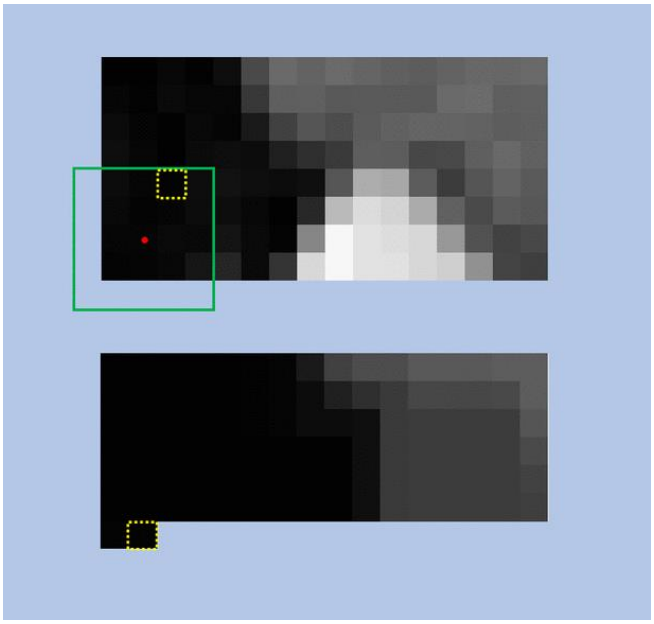
abdulkadir.ture@std.yildiz.edu.tr

Erosion İşlemi:

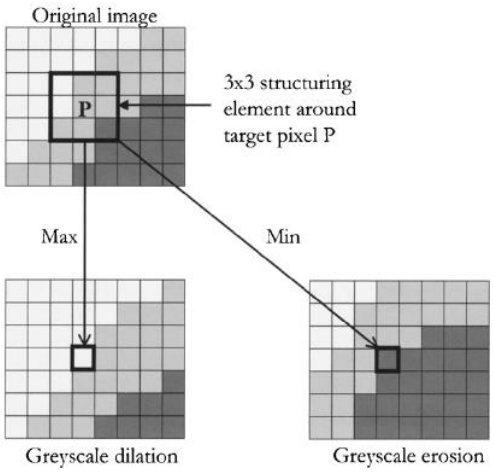
Bu işlemde https://en.wikipedia.org/wiki/Erosion_(morphology) sayfasındaki Grayscale erosion metodu referans alınarak kodlama yapılmıştır.

Bu yönteme göre;

Bir hücrenin değeri komşuluk matrisi (kernel matrisi) içerisindeki değerlerden en küçüğü bulunarak hesaplanır.



Burada üst kısımda bulunan ana görüntü üzerinde kernel matris gezdirilmiş ve iki matrisin de kesiştiği hücrelerde minimum değere sahip olan hücrenin değeri, yeni matriste ilgili hücrenin yeni değeri olmuştur.



Assembly Kod Açıklaması:

```
1     __asm {
2         PUSH ECX
3         PUSH ESI
4         PUSH EAX
5
6         XOR EAX, EAX //Bu blok bana stackte fotograf icin istedigim alanı acar.
7         MOV ECX, n
8         SHR ECX, 1
9   L5 :  PUSH EAX
10        LOOP L5
11
12        PUSH EBX
13        PUSH EDX
14        PUSH EDI
15
```

1 – 15 satırları arasında öncelikle __asm blok içerisinde kullanacağım değerlerin korunması için push işlemini gerçekleştiriyorum. Sonrasında **MOV ECX, n** komutu sayesinde fonksiyonuma gelen 512*512 değerine sahip n değişkenini ECX içerisine atıyorum. **SHR ECX,1** komutu sayesinde bu değeri yarıya bölüyorum. Bu sayede DWORD olarak tanımlı stack içerisinde fotoğraf dizim için yer açacak olan for döngüsünün kaç kez döneceğini hesaplamış oluyorum. Sonrasında **PUSH EAX** komutuyla stack içerisinde fotoğraf için yer ayırıyorum.

16. ve 32. Satırlar arasında tüm fotoğraf matrisini gezmesi için iç içe 2 for yapısı ve kernel matrisini gezmesi için de iç içe 2 for yapısı kullanıyorum. İçteki 2 for **filter_size** değişkeni kadar döneceğinden

```
15
16          MOV ECX, 512
17  disD1:
18          PUSH ECX
19          //EN DIS DONGU KOMUTLARI
20          //EN DIS DNGU KOMUTLARI
21          MOV ECX, 512
22  disD2 :
23          PUSH ECX
24          //DONGU KODLARI BASLANGİC
25          MOV ECX, filter_size //k hesaplandı ECX içinde.
26          MOV EAX, 255  //max veya min degeri yeni elemanda
27          //DONGU KODLARI BİTİS
28  disD3 :
29          PUSH ECX
30          //DONGU KODLARI
31          MOV ECX, filter_size
32          //DONGU KODLARI
33  dis4 :
34          //EN IC DONGU KODLARI baslangic
```

MOV ECX, filter_size komutunu kullanıyorum.

26. Satırda EAX register değerini 255 olarak atama yapıyorum. EAX register değeri kernel matrisinde bulunacak olan minimum değeri taşıyacak. Bu nedenle içteki 2 for yapısı her tekrarlandığında EAX değeri 255 olarak atanıyor.

```
33  dis4 :
34          //EN IC DONGU KODLARI baslangic
35          PUSH EAX
36          MOV EAX, [ESP + 12] //EAX içinde i
37          ADD EAX, [ESP + 4] //EAX = İ+K
38          SUB EAX, 2
39          MOV EBX, filter_size
40          SHR EBX, 1 //EBX içinde filter_size
41          SUB EAX, EBX //EAX = İ+K-size
42          //Taşma kontrolleri baslangic
43          CMP EAX, 0
44          JL PASS
45          CMP EAX, 512
46          JNL PASS
47          MOV EAX, [ESP + 8]
48          ADD EAX, ECX
49          SUB EAX, 2
50          SUB EAX, EBX //EAX = J+L-SİZE
51          CMP EAX, 0
52          JL PASS
53          CMP EAX, 512
54          JNL PASS
55          //Taşma kontrolleri bitis
56          SHL EAX, 1 //EAX = 2*(J-SİZE+L)
57          PUSH EAX
58          MOV EAX, [ESP + 16]
59          ADD EAX, [ESP + 8]
60          SUB EAX, 2
61          SUB EAX, EBX
62          MOV EBX, 1024
63          MUL EBX
64          POP EBX
65          ADD EAX, EBX
66          MOV EBX, resim_org
67          ADD EBX, EAX
68          POP EAX //filtre_Size*3 hucre alan
69          CMP AX, WORD PTR[EBX]
70          JNA PASS1
71          MOV AX, WORD PTR[EBX]
72          JMP PASS1
73  ARA1 :  JMP disD1
74  ARA3 :  JMP disD2
75          //EN IC DONGU KODLARI bitis
76  PASS :  POP EAX
```

En içteki döngüde en dışta bulunan döngü değişkenleri stack üzerinden çekiliyor ve kernel matris için indis değerleri hesaplanıyor. 35. Satırdaki **PUSH EAX** işlemi minimum değeri korumak için kullanılıyor.

EAX		ESP
ECX	4	k
ECX	8	j
ECX	12	i
..		
..		

Visual Studio 2022 ortamında **MOV EBP,ESP** komutu hata verdiğinden stack içerisindeki değerlere **ESP** ile erişim sağlıyorum.

Ayrıca ECX döngü değişkenlerini i, j,k,l şeklinde indis değeri olarak kullanabilmek için **SUB EAX, 2** komutunu gerekli yerlerde kullanıyorum.

Örnek;

0.	1.	2.	3.	4.

ECX = 5

Döngü değişkeni olarak ECX-1 kullanılır ve indis değerlerine erişim sağlanır.

(Döngü değişkeni ECX üzerinden hesaplandıktan sonra EBX registerına atanır.)

Kernel matris üzerinde gezecek olan 2 indis değerim bulunduktan sonra

1. İndis = İ+K-size
2. İndis = J+L-size
   (L içten 1. For değişkeni, K içten 2. For değişkeni)

Bu indis değerleri için 43 , 45 , 51 , 53. Satırlarda fotoğraf dışına taşma kontrolü yapılmaktadır. Taşma işlemi olması durumunda PASS etiketine zıplanarak içteki döngü sonlandırılmaktadır. Taşma işlemi olmadığı durumda ilgili (K,L) indisinin hafızada nerede bulunduğu hesaplanmaktadır.

Hesaplama Yöntemi;

1. İndis değeri 512*2 = **1024** değeri ile çarpılır.
2. İndis değeri **2** ile çarpılır.
   (Her iki değerin de 2 ile çarpılma sebebi dizinin **WORD** tipinde olmasından dolayıdır.)

```
76  PASS :  POP EAX
77  PASS1 : LOOP dis4
78          POP ECX
79          LOOP disD3
80          //Max veya min degeri stackteki di
81          PUSH EAX
82          MOV EAX, [ESP + 8]
83          DEC EAX
84          MOV EBX, 1024
85          MUL EBX
86          MOV EDI, EAX
87          MOV EAX, [ESP + 4]
88          DEC EAX
89          SHL EAX, 1
90          ADD EDI, EAX //EDI DEGERİ MATRİSİ
91          POP EAX
92          ADD EDI, 20 // BU en basta push e
93          MOV WORD PTR[ESP + EDI], AX
94          //Max veya min degeri stackteki di
95          POP ECX
96          LOOP ARA3
97          POP ECX
98          LOOP ARA1
99
100         POP EDI
101         POP EDX
102         POP EBX
```

Sonrasında bu iki değer toplanır ve dizi adresi **MOV EBX, resim_org** komutuyla EBX registerına atanır. Hesaplama ile bulunan **EAX** register değeri ile **EBX** register değeri toplanır ve matristeki ilgili hücre adresine erişilir. Sonrasında Stack içerisinde minimum eleman tutması için atana EAX register pop edilir. İlgili hücre ile AX register değeri karşılaştırılır ve düşük olan AX register yeni değeri olur. Sonrasında en içteki döngü tamamlanır.

EAX		ESP
ECX	4	j
ECX	8	i
..		
..		

*Burada ARA1, ARA3 etiketleri for döngülerinin jmp atlama sınırı aşıldığından bir ara durak olarak yazılmıştır. For döngüsü öncelikle bu etiketlere atlama işlemi gerçekleştirir. Sonrasında döngü etiketine atlama işlemi gerçekleştirir.

İç içe 4 for döngüsü bitimlerinde ECX döngü değişkeni pop edilerek döngülerin istenilen tekrarda dönmesi sağlanır.

En içteki 2 döngü 80. Satırda sonlanmaktadır. Bu satırda EAX değişkeni (AX) içerisinde kernel matrisinin alanında bulunan minimum değer bulunmaktadır.

81-90. satırlar arasında EAX çeşitli işlemler için kullanılacağından korunması için PUSH edilmiştir.

Sonrasında yukarında anlatılan hesaplama yöntemiyle değer ataması yapılacak olan hücrenin indisi hesaplanır.

```
 99
100         POP EDI
101         POP EDX
102         POP EBX
103
104         MOV ESI, resim_org
105         MOV ECX, n
106         SHR ECX, 1
107 D:      POP EAX
108         MOV WORD PTR[ESI], AX
109         ADD ESI, 2
110         ROL EAX, 16
111         MOV WORD PTR[ESI], AX
112         ADD ESI, 2
113         LOOP D
114
115         POP EAX
116         POP ESI
117         POP ECX
118     }
```

Hesaplanan indis değeri EDI registerında tutulur. **ADD EDI,20** komutu stack içerisinde depolanacak olan fotoğraf dizisinin başlangıç adresine gitmek için kullanılır. Sonrasında Push edilerek korunan minimum AX değeri stack içerisinden alınır. Ardından

MOV WORD PTR [ESP+EDI],AX komutuyla stack içerisindeki diziye AX register değeri atılır.

ECX	ESP	
ECX	4	
EDI	8	
EDX	12	
EBX	16	
..	20	DİZİ ALANI BAŞLANGICI
..	24	
..	28	
EAX		
ESI		
ECX		

100-102. satırlar arasında program başlangıcında PUSH edilen register değerleri POP edilir. Sonrasında **MOV ESI, resim_org** komutuyla resim_org dizisinin adresi ESI içerisine atılır.

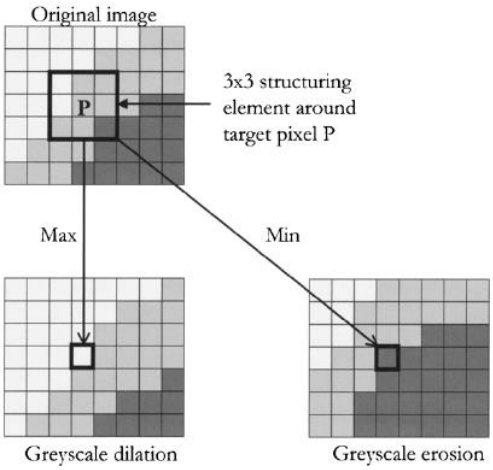
MOV ECX, n komutuyla n değeri ECX içerisine alınır ve **SHR ECX,1** komutuyla 2'ye bölünür. Bölünme sebebi POP işlemi her defasında DWORD değer çeker ve bu değer içerisinde fotoğraf dizisinin 2 değeri bulunmaktadır.

Döngü içersine girildiğinde POP EAX olarak stack içerisinden fotoğraf elemanı çekilir. EAX içerisinde yüksek anlamlı 16 bit içerisinde fotoğrafın 1 elemanı, düşük anlamlı 16 bit içerisinde fotoğrafın 1 elemanı bulunmaktadır.

EAX	
	AX
00000000XXXXXXXX	00000000YYYYYYYY

Öncelikle AX register içerisindeki değer **MOV WORD PTR [ESI], AX** komutuyla resim_org dizisine yerleştirilir. Sonrasında **ROL EAX,16** komutuyla EAX içerisindeki yüksek anlamlı 16 bit ile düşük anlamlı 16 bit yer değiştirir. Bu sayede EAX register içerisindeki diğer fotoğraf değerine erişilmiş olur ve **ADD ESI,2** yapıldıktan sonra **MOV WORD PTR [ESI], AX** komutuyla bu değer de diziye yazılır. Döngü işlemi tüm değerler yazıldıktan sonra bitirilir ve program başlangıcında PUSH edilen registerlar POP edilerek __asm blok sonlandırılır.

Delation İşlemi:



Bu işlemde https://en.wikipedia.org/wiki/Dilation_(morphology) sayfasındaki Grayscale dilation metodu referans alınarak kodlama yapılmıştır.
Bu yönteme göre;
Bir hücrenin değeri komşuluk matrisi (kernel matrisi) içerisindeki değerlerden en büyüğü bulunarak hesaplanır.

Erosion işleminden farklı olarak burada dıştan 2. Döngü içerisinde **MOV EAX,255** yapılmak yerine **XOR EAX,EAX** yapılarak maksimum değer değişkeni sıfırlanmıştır. En içteki döngüde ise karşılaştırma yapan **CMP AX,WORD PTR [EBX]** komutundan sonraki komut **JA PASS1** olarak değiştirilmiştir. Bunun dışında kalan tüm işlemler tamamıyla aynıdır.