

# **YILDIZ TEKNİK ÜNİVERSİTESİ**

## **BİLGİSAYAR MÜHENDİSLİĞİ**



**BLM 2021 – ALT SEVİYE PROGRAMLAMA DERSİ**

**ÖDEV 2 – MORFOLOJİK İŞLEMLER**

**ÖĞR. GÖR. FURKAN ÇAKMAK GR:2**

**15 OCAK 2023**

**SENA ALAY**

**20011047**

**TEL NO: 0532 487 84 17**

[sena.alay@std.yildiz.edu.tr](mailto:sena.alay@std.yildiz.edu.tr)

[alaysena@gmail.com](mailto:alaysena@gmail.com)

# *İÇERİK LİSTESİ*

**1- DİLATION İŞLEMİ ALGORİTMAM**

**2- DİLATION İŞLEMİ ASM KODLARIM VE ADIM ADIM**

**AÇIKLAMALARI**

**3- DİLATION İŞLEMİ EKRAN ÇIKTILARIM (3X3, 5X5, 7X7 FİLTRE İLE)**

**4- EROSION İŞLEMİ ALGORİTMAM**

**5- EROSION İŞLEMİ ASM KODLARIM VE ADIM ADIM**

**AÇIKLAMALARI**

**6- EROSION EKRAN ÇIKTILARIM (3X3, 5X5, 7X7 FİLTRE İLE)**

**7- KAZANIMLAR**

**8- YARARLANILAN KAYNAKLAR**

## DİLATION İŞLEMİ ALGORİTMAM

**Dilation işlemi için yazdığım algoritmanın c kodu:**

```
1 void dilate(int n, int filter_size, short* resim_org) {
2     short* resimadres = resim_org;
3     int i, j, k, l;
4     int max;
5
6     for (i = filter_size; i < n - filter_size; i++) {
7         for (j = filter_size; j < n - filter_size; j++) {
8             max = 0;
9             for (k = -filter_size; k <= filter_size; k++) {
10                for (l = -filter_size; l <= filter_size; l++) {
11                    if (resimadres[(i + k)*n + (j + l)] > max)
12                        max = resimadres[(i + k)*n + (j + l)];
13                }
14            }
15            resimadres[i*n + j] = max;
16        }
17    }
18 }
```

## DİLATION İŞLEMİ ASM KODUM:

### 1.ADIM:

```
_asm{
    xor ecx, ecx
    dec ecx

    FIND_SQRT :
        inc ecx
        mov eax, ecx
        mul eax
        cmp eax, n
        jne FIND_SQRT

    mov n, ecx
```

**İşleme önce n sayısının karekökünü alarak başlıyorum.**

**Bunun için bir while loop kullanıyorum.**

**Her seferinde kontrol ede ede ilerliyorum.**

**Find\_sqrt'in içinden çıktığında mov n, ecx satırı işlendiğinde**

**N'in içinde n'in karekökü oluyor.**

## 2.ADIM:

```
; for (i = filter_size; i < n - filter_size; i++) {  
mov esi, filter_size; i = filtre_size  
mov edi, n  
sub edi, filter_size
```

ilk for karşılaştırmasının içini ayarlamak için

önce esi registerına, filter\_size'ı alıyorum.

Daha önce karekökünü aldığım N'i de edi registerının

içine alıyorum.

Sub edi, filter\_size satırı işlendiğinde

Döngünün döneceği değer, n-2filter\_size, edi'nın içinde

Oluşmuş oluyor.

## 3.ADIM:

```
for1 : cmp esi, edi  
      ja endfor1  
      ; for (j = filter_size; j < n - filter_size; j++) {  
  
      push eax  
      push ebx  
      push ecx  
      push edi  
      push esi  
      push ebp  
  
      mov ebx, filter_size  
      mov edi, n; ebxte n var  
      sub edi, filter_size; ikinci loop donguyu ayarlama yine n - 2 tane filtresize kadar donucek
```

İlk döngünün içi, eğer esi'ya aldığım filter size n-filtre\_sizedan küçükse direk 1. Döngünün bitişine zıplıyor ja endfor1 komutu ile.

Ondan sonra ilk for'da kullandığım registerların değerini korumak için gerekli olan pushları yapıyorum.

Ondan sonra da 2. For döngüsünün içini ayarlıyorum

Filtre\_size'ı ebx'e alıyorum. İlk döngümün indisi esi idi. İkinci döngümün indisi ebx.

Döngünün yeteri kadar dönmesi için gerekli değer sub edi, filter\_size satırı işlendiğinde ayarlanmış oluyor.

## 4.ADIM:

```
for2 : cmp ebx, edi
      ja endfor2

      push eax
      push ebx
      push ecx
      push edi
      push esi
      push ebp

      // for (k = -filter_size; k <= filter_size; k++) {
      mov ecx, filter_size      // k = filtre_size
      neg ecx                  // ARTIK K = -FILTRESIZE
      mov edi, filter_size

      xor eax, eax             // maxı tanımlama kısmı. max için eax registerını kullanıyorum
```

*İkinci döngünün içi.*

*Eğer döngü şartı sağlanmadıysa ja endfor2 ile ikinci döngüden çıkıyor.*

*Daha sonra 2. Döngüde kullanılan registerların değeri bozulmasın diye gerekli pushları yapıyorum.*

*Daha sonra 3.döngü şartım için gerekli düzenlemeleri yapıyorum.*

*3.döngü şartım (for k=-filtre\_size; k<=filter\_size; k++ ) idi. Üçüncü döngümün indisi ecx.*

*Filter\_Size'i ecx'e alıyorum. Neg ecx yaptığımda ecx'te artık -filtre\_Size var.*

*Ondan sonra da algoritmamda her pixel'i sırasıyla karşılaştırma yapacağım max değişkeni için eax'i kullanıyorum. Xor ile max'ı sıfırlıyorum.*

## 5.ADIM:

```
for3:
  cmp ecx, edi      //ucuncu for sartının ic kismini ayarlama
  jge endfor3

  push ebx
  push ecx
  push edx
  push edi
  push esi
  push ebp

  // for (l = -filter_size; l <= filter_size; l++) {
  mov edx, filter_size
  neg edx
  mov edi, filter_size
```

*Üçüncü döngü şartım sağlanmadıysa jge ile endfor3 yaparak döngüden çıkıyorum.*

*Daha sonra döngüde kullandığım registerların değeri bozulmasın diye değerleri stack'e pushlıyorum.*

*Daha sonra da en son döngü şartımı ayarlıyorum. Filter\_Size'ı edx'e alıyorum. Neg edx yapınca edx'te artık -filtre\_Size oluyor. Son döngümün indisi de edx.*

## 6.ADIM:

```
for4:  cmp edx, edi
      jge endfor4
      // if (resimadres[(i + k) * n + (j + l)] > max)

      push ebx
      push ecx
      push edx
      push edi
      push esi
      push ebp

      push eax

      add esi, ecx
      mov eax, n
      mul esi

      add eax, ebx
      add eax, edx

      mov esi, eax
      pop eax

      shl esi, 1

      //(esi + ecx) * n + ebp + edx
      mov ebx, yeni_resim
      mov cx, WORD PTR [ebx + esi]
      cmp cx, ax
      jg maxiguncelle
      jmp next_loop
```

**En son döngünün içi.**

**Eğer gerekli şart sağlanmıyorsa jge endfor4 ile döngüden çıkıyorum.**

**Daha sonra kullandığım registerların değerleri değişmesin diye stacke pushluyorum.**

**Daha sonra if(resimadres[i+k]\*n + (j+l) > max) karşılaştırmasını yapabilmek için**

**İlk döngünün indisi = i yani esi,**

**İkinci döngünün indisi = j yani ebx,**

**Üçüncü döngünün indisi = k yani ecx,**

**Dördüncü döngünün indisi = l yani edx**

**Add esi, ecx yapınca i+k toplamı esida oluşuyor.**

**Mov eax, n yapınca ve mul esi yapınca çarpım sonucu eax'te oluşuyor.**

**Daha sonra ( i+k ) \* n in sonucu eaxteydi. Bu toplamın üzerine j+lyi sırayla toplayabilmek için add eax, ebx**

**Ve add eax, edx yapıyorum. Önce j sonra da lyi eklemiş oluyorum. Eax'te istediğim değer oluşuyor.**

*Endfor4'ün içi. Önce stacke attığım değerleri pop ile çekip üçüncü döngünün indisi olan ecx'i artırıp jmp for3 ile üçüncü döngüye geliyorum.*

## 8.ADIM:

```
endfor3 :  
  
// resimadres[i * n + j] = max;  
push eax  
push edx  
  
mov eax, esi  
mul n  
add eax, ebx  
  
shl eax, 1  
mov esi, eax  
  
pop edx  
pop eax  
  
mov ebx, resim_org  
mov WORD PTR [ebx + esi], ax  
  
pop ebp  
pop esi  
pop edi  
pop ecx  
pop ebx  
pop eax  
  
inc ebx  
jmp for2
```

Bulunan max değeri resme yazarak dilation işlemini yaptığım yer. Resime yazabilmek için Word ptr ile cast ediyorum önce.

Gereken popları yapıp, döngümün indisi olan ebxi artırıp jmp for2 ile 2. Döngüye geçiyorum

## 9.ADIM:

```
endfor2 :  
    pop ebp  
    pop esi  
    pop edi  
    pop ecx  
    pop ebx  
    pop eax  
  
    inc esi  
    jmp for1  
  
endfor1 : // end of the function
```

Endfor2 ile 2. Döngüden çıkmışsam attığım değerleri stackten pop ile çekiyorum.

İlk döngümün indisi olan esi'yi artırıp jmp for1 ile ilk döngüme atlıyorum.

Endfor1: işlemin tamamlanıp resmin oluşturulduğu ve asm kodunun bittiği yer.



***DİLATED.PGM - 1 (3X3 FİLTRE)***



***LENA.PGM ORİJİNAL HALİ VS 3X3 FİLTRE İLE  
DİLATION İŞLEMİ SONRASI***



***DİLATED.PGM – 2 (5X5 FİLTRE)***



***LENA.PGM ORİJİNAL HALİ VS 5X5 FİLTRE İLE  
DİLATION İŞLEMİ SONRASI***



***DİLATED.PGM - 3 (7X7 FİLTRE)***



***LENA.PGM ORİJİNAL HALİ VS 7X7 FİLTRE İLE  
DİLATION İŞLEMİ SONRASI***



## EROSİON İŞLEMİ ALGORİTMAM

```
17
20 void erosion(int n, int filter_size, short* resim_org) {
21     int i, j, k, l;
22     short min;
23     for (i = 0; i < n - filter_size + 1; i++) {
24         for (j = 0; j < n - filter_size + 1; j++) {
25             min = resim_org[i * n + j];
26             for (k = 0; k < filter_size; k++) {
27                 for (l = 0; l < filter_size; l++) {
28                     if (resim_org[(i + k) * n + (j + l)] < min) {
29                         min = resim_org[(i + k) * n + (j + l)];
30                     }
31                 }
32             }
33             resim_org[i * n + j] = min;
34         }
35     }
36 }
```

## EROSİON İŞLEMİ ASM KODUM

### 1.ADIM:

Algoritmalar benzer. Dilation'da maxı 0 tanımlıyorduk, erosion2da mini 255 tanımlayıp karşılaştırmalar yapacağız.

\_\_asm {

xor ecx, ecx  
dec ecx

FIND\_SQRT :

inc ecx  
mov eax, ecx  
mul eax  
cmp eax, n  
jne FIND\_SQRT  
  
mov n, ecx

*İşleme önce n sayısının karekökünü alarak başlıyorum.*

*Bunun için bir while loop kullanıyorum.*

*Her seferinde kontrol ede ede ilerliyorum.*

*Find\_sqrt'in içinden çıktığında mov n, ecx satırı işlendiğinde*

*N'in içinde n'in karekökü oluyor.*

## 2.ADIM:

ilk for karşılaştırmasının içini ayarlamak için

önce esi registerına, filter\_size'ı alıyorum.

Daha önce karekökünü aldığım N'i de edi registerının

içine alıyorum.

Sub edi, filter\_size satırı işlendiğinde

Döngünün döneceği değer, n-2filter\_size, edi'nın içinde

Oluşmuş oluyor.

```
// for (i = filter_size; i < n - filter_size; i++) {
    mov esi, filter_size //i = filtre_size
    mov edi, n
    sub edi, filter_size

for1 :
    cmp esi, edi
    ja endfor1
    // for (j = filter_size; j < n - filter_size; j++) {

        push eax
        push ebx
        push ecx
        push edi
        push esi
        push ebp

        mov ebx, filter_size
        mov edi, n // edida n var
        sub edi, filter_size // loopdonguyu ayarladigim yer n - 2 filtresize kadar donucek
```

**FOR1:**

İlk döngünün içi, eğer esi'ya aldığım filter size n-filtre\_sizedan küçükse direk 1. Döngünün bitişine zıplıyor ja endfor1 komutu ile.

Ondan sonra ilk for'da kullandığım registerların değerini korumak için gerekli olan pushları yapıyorum.

Ondan sonra da 2. For döngüsünün içini ayarlıyorum

Filtre\_size'ı ebx'e alıyorum. İlk döngümün indisi esi idi. İkinci döngümün indisi ebx.

Döngünün yeteri kadar dönmesi için gerekli değer sub edi, filter\_size satırı işlendiğinde ayarlanmış oluyor.

## 3.ADIM:

```
for2:
    cmp ebx, edi
    ja endfor2

    push eax
    push ebx
    push ecx
    push edi
    push esi
    push ebp

    // for (k = -filter_size; k <= filter_size; k++) {
    mov ecx, filter_size    // k = filtre_size
    neg ecx                // ARTIK K = -FILTRESIZE
    mov edi, filter_size

    mov eax, 255 // mini tanımladığım yer. her seferinde 255 ile karsilastirip küçükse degistirecegim.
```

*İkinci döngünün içi.*

*Eğer döngü şartı sağlanmadıysa ja endfor2 ile ikinci döngüden çıkıyor.*

*Daha sonra 2. Döngüde kullanılan registerların değeri bozulmasın diye gerekli pushları yapıyorum.*

*Daha sonra 3.döngü şartım için gerekli düzenlemeleri yapıyorum.*

*3.döngü şartım (for k=-filtre\_size; k<=filter\_size; k++ ) idi. Üçüncü döngümün indisi ecx.*

*Filter\_Size'i ecx'e alıyorum. Neg ecx yaptığımda ecx'te artık -filtre\_Size var.*

*Ondan sonra da algoritmamda her pixel'i sırasıyla karşılaştırma yapacağım min değişkeni için eax'i kullanıyorum. Min'e 255 koyuyorum. Minden küçük olan her değeri sırasıyla değiştireceğim.*

## 4.ADIM:

```
for3 :
    cmp ecx, edi    // 3.for sartinin ayarlanmasi
    jge endfor3

    push ebx
    push ecx
    push edx
    push edi
    push esi
    push ebp

    //for (l = -filter_size; l <= filter_size; l++) {
    mov edx, filter_size
    neg edx
    mov edi, filter_size
```

*Üçüncü döngü şartım sağlanmadıysa jge ile endfor3 yaparak döngüden çıkıyorum.*

*Daha sonra döngüde kullandığım registerların değeri bozulmasın diye değerleri stack'e pushlıyorum.*

*Daha sonra da en son döngü şartımı ayarlıyorum.*

*Filter\_Size'ı edx'e alıyorum.*

*Neg edx yapınca edx'te artık -filtre\_Size oluyor. Son döngümün indisi de edx.*

## 5.ADIM:

```
for4:  cmp edx, edi
      jge endfor4
      // if (resimadres[(i + k) * n + (j + l)] < MIN)

      push ebx
      push ecx
      push edx
      push edi
      push esi
      push ebp

      push eax

      add esi, ecx
      mov eax, n
      mul esi

      add eax, ebx
      add eax, edx

      mov esi, eax
      pop eax

      shl esi, 1

      //(esi + ecx) * n + ebp + edx
      mov ebx, yeni_resim
      mov cx, WORD PTR[ebx + esi]
      cmp cx, ax
      jl miniguncelle
      jmp next_loop
```

*En son döngümün içi.*

*Eğer gerekli şart sağlanmıyorsa jge endfor4 ile döngüden çıkıyorum.*

*Daha sonra kullandığım registerların değerleri değişmesin diye stacke pushluyorum.*

*Daha sonra if(resimadres(i+k)\*n + (j+l) > max) karşılaştırmasını yapabilmek için*

*İlk döngümün indisi = i yani esi,*

*İkinci döngümün indisi = j yani ebx,*

*Üçüncü döngümün indisi = k yani ecx,*

*Dördüncü döngümün indisi = l yani edx*

*Add esi, ecx yapınca i+k toplamı esida oluşuyor.*

*Mov eax, n yapınca ve mul esi yapınca çarpım sonucu eax'te oluşuyor.*

*Daha sonra (i+k)\*n in sonucu eaxteydi. Bu toplamın üzerine j+lyi sırayla toplayabilmek için add eax, ebx*



Ve add eax, edx yapıyorum. Önce j sonra da lyi eklemiş oluyorum. Eax'te istediğim değer oluşuyor.

Eaxin bundan önceki değerini kaybetmemek için bu işlemlerden önce pushladım stacke.

Daha sonra eaxte oluşturduğum sonucu esi'ya aldım.

Eax'i stackten çektim. Esi'yi shl ile 2 ile çarptım.

Daha sonra oluşturduğum yeni\_resim dizisinden okuduğum değeri ebx' e alıyorum.

Word ptr ile cast işlemi yapıyorum.

Cmp cx ,ax ile eğer cxtaki değer küçükse

min değeri güncellemek için jl minigüncelle ile zıplıyorum.

Sonra da diğer döngüye geçmek için jmp next\_loop yapıyorum.

## 6.ADIM:

```
miniguncelle :  
    mov ax, cx  
  
next_loop :  
    pop ebp  
    pop esi  
    pop edi  
    pop edx  
    pop ecx  
    pop ebx  
  
    inc edx  
    jmp for4  
  
endfor4 :  
    pop ebp  
    pop esi  
    pop edi  
    pop edx  
    pop ecx  
    pop ebx  
  
    inc ecx  
    jmp for3
```

***MIN'i güncellediğim kısım.***

***Next\_loop'ta döngüye girmeden pushladığım değerleri***

***önce pop ile stackten çekiyorum.***

***Döngü indisim olan edx'i artıyorum.***

***Jmp for4 ile tekrar dördüncü döngümdeyim.***

***Endfor4'ün içi.***

***Önce stacke attığım değerleri pop ile çekip üçüncü döngünün indisi olan ecx'i artırıp***

***jmp for3 ile üçüncü döngüye geliyorum.***



## 7.ADIM:

```
endfor3:
    // resimadres[i * n + j] = MİN;
    push eax
    push edx

    mov eax, esi
    mul n
    add eax, ebx

    shl eax, 1
    mov esi, eax

    pop edx
    pop eax

    mov ebx, resim_org
    mov WORD PTR[ebx + esi], ax

    pop ebp
    pop esi
    pop edi
    pop ecx
    pop ebx
    pop eax

    inc ebx
    jmp for2
```

Bulunan MİN değeri resme yazarak EROSION işlemini yaptığım yer.

Resime yazabilmek için Word ptr ile cast ediyorum önce.

Gereken popları yapıp, döngümün indisi olan ebxi artırıp jmp for2 ile 2. Döngüye geçiyorum

## 8.ADIM:

```
endfor2:
    pop ebp
    pop esi
    pop edi
    pop ecx
    pop ebx
    pop eax

    inc esi
    jmp for1

endfor1: // end of the function
```

Endfor2 ile 2. Döngüden çıkmışsam attığım değerleri stackten pop ile çekiyorum.

İlk döngünün indisi olan esi'yi artırıp jmp for1 ile ilk döngüme atlıyorum.

Endfor1: işlemin tamamlanıp resmin oluşturulduğu ve asm kodunun bittiği yer.

***ERODED.PGM – 1 (3X3 FİLTRE)***



***LENA.PGM ORİJİNAL HALİ VS 3X3 FİLTRE İLE  
EROSİON İŞLEMİ SONRASI***



***ERODED.PGM – 2 (5X5 FİLTRE)***



***LENA.PGM ORİJİNAL HALİ VS 5X5 FİLTRE İLE  
EROSİON İŞLEMİ SONRASI***



***ERODED.PGM – 3 (7X7 FİLTRE)***



***LENA.PGM ORİJİNAL HALİ VS 7X7 FİLTRE İLE  
EROSİON İŞLEMİ SONRASI***



## KAZANIMLAR

*C dili içerisinde inline Assembly yazabilme kabiliyeti kazandım. Algoritma kurma konusunda pratik yapmış oldum. Morfolojik işlem yapabilmeyi, Mac bilgisayarına virtual machine ile Windows kurup Visual Studio ile çalışmayı öğrendim.*

## YARARLANILAN KAYNAKLAR

- <https://buzztech.in/erosion-and-dilation-in-digital-image-processing/>
- <https://www.geeksforgeeks.org/erosion-and-dilation-morphological-transformations-in-opencv-in-cpp/>
- <https://www.ncbi.nlm.nih.gov/books/NBK546156/box/ch3.box16/?report=objectonly>
- [https://en.wikipedia.org/wiki/Dilation\\_\(morphology\)](https://en.wikipedia.org/wiki/Dilation_(morphology))
- [https://en.wikipedia.org/wiki/Erosion\\_\(morphology\)](https://en.wikipedia.org/wiki/Erosion_(morphology))
- <https://www.youtube.com/watch?v=OTVtGdYY3YE>
- <https://www.youtube.com/watch?v=xO3ED27rMHs>
- <https://www.youtube.com/watch?v=fmyE7DialYQ>
- <https://www.youtube.com/watch?v=fmyE7DialYQ>
- <https://www.youtube.com/watch?v=rP1KZb3lICY>