YILDIZ TEKNİK ÜNİVERSİTESİ

Elektrik- Elektronik Fakültesi

Bilgisayar Mühendisliği Bölümü

BLM 2021

ALT SEVİYE PROGRAMLAMA GR:1

Dr. Öğretim Üyesi Erkan Uslu

Ödev2- ASM Dilinde Dilation ve Erasion İşlemlerinin Gerçeklenmesi

İsim: Doğukan Baş

E-mail:dogukan.bas@std.yildiz.edu.tr

No: 21011003

Contents

Dilation İşlemi

Her bir elemana filtre, eleman filtrenin merkezine gelecek şekilde uygulanmıştır. Eğer filtre matrisin dışına çıkıyorsa bu elemanlar aynen bırakılmıştır. Filtrenin tüm elemanları fotoğrafın içindeyse, bu elemanlardan en büyüğü alınıp merkezdeki elemanın yerine konulur. Burda dikkat edilmesi gereken bir nokta, bu eleman değiştirme işlemi aynı matris üzerinden işlem bitmeden yapılırsa sonraki elemanları etkileyeceğidir. Bunu engellemek adına ekstra bir dizi veya stack kullanabiliriz. Ben stack kullanarak bu işlemi gerçekleştirdim.

Pseudo Kod

```
for i in range(n):
    x=i/512
    y=i%512
    max_value=0

    if x-filter_size/2 < 0 or x+filter_size/2 >= 512 or y+filter_size/2 < 0 or
y+filter_size/2 >= 512:
        continue

    for j in range(-filter_size/2, filter_size/2+1):
        for k in range(-filter_size/2, filter_size/2+1):

            max_value = max(max_value, resim_org[(x+j)*512 + y+k])
    PUSH max_value
for i in range(n,0): //resmi tersten geziniyoruz ,en üstteki eleman = son eleman
      POP max_value
      resim[i]= max_value
```

Assembly Kodu

```
MOV ECX, n
 MOV ESI, resim_org
 XOR EBX, EBX
 L1 :
        MOV EDI, 512
        MOV EAX, EBX
        XOR EDX, EDX
        DIV EDI
        ; EAX[i][j] eax i   edx j
        MOV EDI, filter_size
        SHR EDI, 1
        ADD EAX, EDI
        CMP EAX, 512
        JAE SONUC
        SUB EAX, EDI
        CMP EDI, EAX
        JAE SONUC
        ADD EDX, EDI
        CMP EDX, 512
        JAE SONUC
        SUB EDX, EDI
        CMP EDI, EDX
        JAE SONUC
        JMP DEVAM
        TEMP : JMP L1
 DEVAM :
        XOR EDX, EDX
        XOR EAX, EAX
        PUSH ECX
        XOR ECX, ECX
        MOV CL, 9
        SHL EDI, CL
        PUSH EBX
        SUB EBX, EDI
        SHR EDI, CL
        MOV ECX, filter_size
        SUTUN :
                PUSH ECX
                MOV ECX, filter_size
                PUSH EBX
                SUB EBX, EDI

                SATIR :
                        SHL EBX, 1
```

```
                        MOV AX, WORD PTR[EBX + ESI]
                        SHR EBX, 1
                        CMP DX, AX
                        JAE DEGISME
                        MOV DX, AX
                        DEGISME :
                        ADD EBX, 1
                    LOOP SATIR
                    POP EBX
                    POP ECX
                    ADD EBX, 512
                LOOP SUTUN
                POP EBX
                POP ECX
                PUSH DX

                jmp EXIT

SONUC :
                SHL EBX, 1
                MOV AX, WORD PTR[EBX + ESI]
                PUSH AX
                SHR EBX, 1

EXIT :
                ADD EBX, 1
                LOOP TEMP

MOV ECX, n
XOR EAX, EAX
MOV EBX, resim_org
ADD EBX, n
ADD EBX, n

L2 :
                sub EBX, 2
                pop(AX)
                MOV WORD PTR[EBX], ax
loop l2
```

Kodun Açıklaması

1) Öncelikle ECX registerında resmin boyutunun toplam büyüklüğü(N*M) olan n değeri saklanır çünkü toplam n adet eleman vardır ve bu döngü n defa dönecektir, ESI registerında resim_org isimli dizinin adresi atılır ve dizi elemanlarına erişimde kullanılacak olan EBX registerı 0 olarak belirlenir.
2) Daha sonra en dıştaki döngüye ait etiket olan L1: etiketine girilir. Bu döngü bir önceki maddede de belirtildiği üzere n defa döner.
3) Döngüye ilk girdiğimizde ilgili elemanın matris yapısındaki satır ve sütun değerleri belirlenir. Elemanın bulunduğu satır = indis/512 , sütun = indis%512 şeklinde hesaplanır.
4) Daha sonra EDI ifadesine filter_size değeri atılır. Bu değer SHR komutu ile 2'ye bölünür filter_size/2 değeri EDI'da elde edilir . Burdan sonra DEVAM etiketine kadar olan kısımdaki 4 kontrol de matrisin dışına taşma durumlarıdır. Eğer bu elemanın filter_size/2 kadar sağında,solunda,üstünde veya altında bulunan elemanlardan herhangi bir tanesi bile matris dışında ise SONUC etiketine gidilir. Bu etikette ise stack'e eleman aynen pushlanır. Bu elemanlar değiştirilmemiş olur.
5) Devam etiketine geldiğimizde ise EDX ve EAX registerları sıfırlandıktan hemen sonra dıştaki döngüye ait olan ECX registerı kaybedilmemek adına stack'e atılır. Burada ECX kullanılacaktır, Dolayısıyla sıfırlanır ve CL değerine 9 atanır. EDI registerında en son filter_size/2 vardı. $2^9 = 512$ değeriyle SHL komutu yardımıyla çarpılır. Daha sonra kaybedilmemek adına EBX registerı da stack'e atılır. Bunun sebebim bu registerın dıştaki döngüde de kullanılmasıdır.
6) Döngü $i - \frac{filtersize}{2} * 512 - \frac{filtersize}{2}$ 'den başlatılmalıdır. Çünkü bu eleman matris olarak düşünüldüğünde bu filtrenin en sol üstünde kalan elemandır . EBX registerında i değerini saklıyorduk. EBX- EDI bize $i - \frac{filtersize}{2} * 512$ değerini verir. (bu işlemden hemen sonra EDI tekrar 2^9 ile çarpılarak filter_size/2 haline geri getirilir)
7) ECX registerına filter_size atılır. Bu içerideki sütun ve satır döngülerinin her biri filter_size kadar dönecektir. Sutun döngüsüne girdikten sonra, her bir satır için döngünün , filtrenin o satırdaki ilk elemanından başlaması için SUB EBX,EDI işlemi yapılır . Bu işlem EBX iadesinden filtersize/2 kadar soldaki elemanı verir. Daha sonra SATIR labelına gelinir.
8) Burada EBX iki ile çarpılır çünkü resimin elemanları word tipindedir. İlgili eleman AX registerına çekilir. Daha sonra AX register DX'ten büyük mü diye kontrol yapılır. Burada Dx bizim en büyük değeri tutan elemanımızdır.Dolayısıyla başlangıçta 0 olarak başlatılır . AX>DX durumunda DX registerına AX atılır. İçerideki döngünün sonuna gelindiğinde EBX 1 artırılır. Bi dışarıdaki döngüde ise EBX 512 artırılır. EBX'in bir artırılması demek, matris olarak düşünüldüğünde bir sağındaki elemana geçmesidir. 512 artırılması ise bir alt satırda aynı sütunda bulunan elemana geçilmesi demektir. Bu şekilde filtrenin içinde bulunan tüm elemanlar gezilir ve en büyüğü bulunur.
9) Bulunan en büyük eleman stack'e atılır daha sonra EXIT registerına gidliir . Burada ebx 1 artırılır. Burada önemli hususlardan bir tanesi EBX'in az önce bahsettiğimiz EBX'ten farklı olduğudur. Burdaki EBX ilk başta tüm elemanları gezdiğimiz döngüye aittir. Filtrenin merkezine gelecek elemanı belirlememiz için kullanılır.
10) En dıştaki döngü de bittiği zaman artık stackte elemanlarımızı ters sıralı bir şekilde elde ederiz. Bunları bu şekilde stackten çekip sırayla fotoğrafın ilk elemanından başlayıp yazdırcak olsak fotoğrafı ters görürdük. Bunun sebebi stack'e son atılan elemanın ilk çekilmesidir. Son atılan

eleman da fotoğrafın en sonundaki eleman olduğundan böyle yapsaydık fotoğraf ters oluşturulurdu.

11) Dolayısıyla burada en son elemandan başlayarak başa doğru gidilir ve stackten çekilen eleman resim dizisi üzerine yazılır .

Çıktılar:

Orjinal fotoğraf

filter_size=3

filter_size=5

filter_size=7

Erosion İşlemi

Her bir elemana filtre, eleman filtrenin merkezine gelecek şekilde uygulanmıştır. Eğer filtre matrisin dışına çıkıyorsa bu elemanlar aynen bırakılmıştır. Filtrenin tüm elemanları fotoğrafın içindeyse, bu elemanlardan en küçüğü alınıp merkezdeki elemanın yerine konulur. Burda dikkat edilmesi gereken bir nokta, bu eleman değiştirme işlemi aynı matris üzerinden işlem bitmeden yapılırsa sonraki elemanları etkileyeceğidir. Bunu engellemek adına ekstra bir dizi veya stack kullanabiliriz. Ben stack kullanarak bu işlemi gerçekleştirdim.

Pseudo Kod

```
for i in range(n):
    x=i/512
    y=i%512
    min_value=0

    if x-filter_size/2 < 0 or x+filter_size/2 >= 512 or y+filter_size/2 < 0 or
y+filter_size/2 >= 512:
        continue

    for j in range(-filter_size/2, filter_size/2+1):
        for k in range(-filter_size/2, filter_size/2+1):

            min_value = min(min_value, resim_org[(x+j)*512 + y+k])
    PUSH min_value
for i in range(n,0): //resmi tersten geziniyoruz ,en üstteki eleman = son eleman
    POP min_value
    resim[i]= min_value
```

Assembly Kodu

```
      MOV ECX, n
        MOV ESI, resim_org
        XOR EBX, EBX
        L1 :
             MOV EDI, 512
            MOV EAX, EBX
            XOR EDX, EDX
            DIV EDI
            ; EAX[i][j] eax i   edx j
            MOV EDI, filter_size
            SHR EDI, 1

            ADD EAX, EDI
            CMP EAX, 512
            JAE SONUC
            SUB EAX, EDI

            CMP EDI, EAX
            JAE SONUC

            ADD EDX, EDI
            CMP EDX, 512
            JAE SONUC
            SUB EDX, EDI

            CMP EDI, EDX
            JAE SONUC
            JMP DEVAM
            TEMP : JMP L1
      DEVAM :
            MOV EDX,255
            XOR EAX, EAX
            PUSH ECX

            XOR ECX, ECX
            MOV CL, 9
            SHL EDI, CL
            PUSH EBX
            SUB EBX, EDI
            SHR EDI, CL
            MOV ECX, filter_size
        SUTUN :
```

```
            PUSH ECX
            MOV ECX, filter_size
            PUSH EBX
            SUB EBX, EDI
           SATIR :
            SHL EBX, 1
            MOV AX, WORD PTR[EBX + ESI]
            SHR EBX, 1
            CMP AX, DX
            JAE DEGISME
            MOV DX, AX
            DEGISME :
            ADD EBX, 1
            LOOP SATIR
            POP EBX
            POP ECX
            ADD EBX, 512
            LOOP SUTUN
            POP EBX
            POP ECX
            PUSH DX
            jmp EXIT
         SONUC :
            SHL EBX, 1
            MOV AX, WORD PTR[EBX + ESI]
            PUSH AX
            SHR EBX, 1
         EXIT :
            ADD EBX, 1
            LOOP TEMP
            MOV ECX, n
            XOR EAX, EAX
            MOV EBX, resim_org
            ADD EBX, n
            ADD EBX, n
        L2 :
         sub EBX, 2
            pop(AX)
            MOV WORD PTR[EBX], ax
            loop l2
```

Kodun Açıklaması

1) Öncelikle ECX registerında resmin boyutunun toplam büyüklüğü(N*M) olan n değeri saklanır çünkü toplam n adet eleman vardır ve bu döngü n defa dönecektir, ESI registerında resim_org isimli dizinin adresi atılır ve dizi elemanlarına erişimde kullanılacak olan EBX registerı 0 olarak belirlenir.
2) Daha sonra en dıştaki döngüye ait etiket olan L1: etiketine girilir. Bu döngü bir önceki maddede de belirtildiği üzere n defa döner.
3) Döngüye ilk girdiğimizde ilgili elemanın matris yapısındaki satır ve sütun değerleri belirlenir. Elemanın bulunduğu satır = indis/512 , sütun = indis%512 şeklinde hesaplanır.
4) Daha sonra EDI ifadesine filter_size değeri atılır. Bu değer SHR komutu ile 2'ye bölünür filter_size/2 değeri EDI'da elde edilir . Burdan sonra DEVAM etiketine kadar olan kısımdaki 4 kontrol de matrisin dışına taşma durumlarıdır. Eğer bu elemanın filter_size/2 kadar sağında,solunda,üstünde veya altında bulunan elemanlardan herhangi bir tanesi bile matris dışında ise SONUC etiketine gidilir. Bu etikette ise stack'e eleman aynen pushlanır. Bu elemanlar değiştirilmemiş olur.
5) Devam etiketine geldiğimizde ise EDX ve EAX registerları sıfırlandıktan hemen sonra dıştaki döngüye ait olan ECX registerı kaybedilmemek adına stack'e atılır. Burada ECX kullanılacaktır, Dolayısıyla sıfırlanır ve CL değerine 9 atanır. EDI registerında en son filter_size/2 vardı. $2^9 = 512$ değeriyle SHL komutu yardımıyla çarpılır. Daha sonra kaybedilmemek adına EBX registerı da stack'e atılır. Bunun sebebim bu registerın dıştaki döngüde de kullanılmasıdır.
6) Döngü $i - \frac{filtersize}{2} * 512 - \frac{filtersize}{2}$ 'den başlatılmalıdır. Çünkü bu eleman matris olarak düşünüldüğünde bu filtrenin en sol üstünde kalan elemandır . EBX registerında i değerini saklıyorduk. EBX- EDI bize $i - \frac{filtersize}{2} * 512$ değerini verir. (bu işlemden hemen sonra EDI tekrar 2^9 ile çarpılarak filter_size/2 haline geri getirilir)
7) ECX registerına filter_size atılır. Bu içerideki sütun ve satır döngülerinin her biri filter_size kadar dönecektir. Sutun döngüsüne girdikten sonra, her bir satır için döngünün , filtrenin o satırdaki ilk elemanından başlaması için SUB EBX,EDI işlemi yapılır . Bu işlem EBX iadesinden filtersize/2 kadar soldaki elemanı verir. Daha sonra SATIR labelına gelinir.
8) Burada EBX iki ile çarpılır çünkü resimin elemanları word tipindedir. İlgili eleman AX registerına çekilir. Daha sonra AX register DX'ten küçük mü diye kontrol yapılır. Burada Dx bizim en küçük değeri tutan elemanımızdır.Dolayısıyla başlangıçta 255 (olabilecek en büyük değer)olarak başlatılır . AX<DX durumunda DX registerına AX atılır. İçerideki döngünün sonuna gelindiğinde EBX 1 artırılır. Bir dışarıdaki döngüde ise EBX 512 artırılır. EBX'in bir artırılması demek, matris olarak düşünüldüğünde bir sağındaki elemana geçmesidir. 512 artırılması ise bir alt satırda aynı sütunda bulunan elemana geçilmesi demektir. Bu şekilde filtrenin içinde bulunan tüm elemanlar gezilir ve en küçüğü bulunur.
9) Bulunan en küçük eleman stack'e atılır daha sonra EXIT registerına gidliir . Burada ebx 1 artırılır. Burada önemli hususlardan bir tanesi EBX'in az önce bahsettiğimiz EBX'ten farklı olduğudur. Burdaki EBX ilk başta tüm elemanları gezdiğimiz döngüye aittir. Filtrenin merkezine gelecek elemanı belirlememiz için kullanılır.
10) En dıştaki döngü de bittiği zaman artık stackte elemanlarımızı ters sıralı bir şekilde elde ederiz. Bunları bu şekilde stackten çekip sırayla fotoğrafın ilk elemanından başlayıp yazdırcak olsak fotoğrafı ters görürdük. Bunun sebebi stack'e son atılan elemanın ilk çekilmesidir. Son atılan

eleman da fotoğrafın en sonundaki eleman olduğundan böyle yapsaydık fotoğraf ters oluşturulurdu.

11) Dolayısıyla burada en son elemandan başlayarak başa doğru gidilir ve stackten çekilen eleman resim dizisi üzerine yazılır .

Çıktılar:

Orjinal fotoğraf

filter_size=3

filter_size=5

filter_size=7