# Bilgisayar Mimarisi Proje Ödevi

1306160048 Barış YARAR

1306170024 Mert AKYAZI

Öncelikle zincirleri .asciiz türünde oluşturduk. Asciiz olma sebebi string'in sonuna geldiğimizde 0 kontrolü ile anlayabilmemiz.

Diffmsg, eğer kıyasladığımız zincirler farklıysa konsola "stringler farklı" yazdırıyoruz.

Samemsg, eğer kıyasladığımız zincirler aynıysa (eşlenikse) k0 ve k1 registerlarına o zincirlerin idlerini yazdırıyoruz. Zincir1 için 1, zincir2 için 2...

10 byte'lık tek bir buffer kullanmak yeterli, bir zincir seçip o zincirin eşleniğini alıp buffer'a kaydediyoruz. Zincirler de 10 byte'lık olduğu için 10 byte'lık buffer yeterli.

A,T,G,C karakterlerinin ascii karşılıklarını kullanarak zincirlerdeki elemanlarla kıyaslıyoruz, bu nedenle .word tipinde ascii değerleri kaydedildi.

```
1  .data
2  zincir1: .asciiz "ATGATGATGC"
3  zincir2: .asciiz "TCGCGCTAGC"
4  zincir3: .asciiz "CGTCGTAAAC"
5  zincir4: .asciiz "TATTTACGAA"
6  zincir5: .asciiz "TACTACTACG"
7
8
9  diffmsg:    .asciiz "\nstringler farklı\n\n"
10 samemsg:    .asciiz "\nstringler aynı, k0 ve k1 registerında saklanıyor\n\n"
11
12 buffer: .space 10   #buffer bizim eşleniğimizi tutacak
13
14 A:   .word   65 #A
15 T:   .word   84 #T
16 G:   .word   71 #G
17 C:   .word   67 #C
18
```

Kodumuzda 2 adet fonksiyon bulunuyor. İlki _function_eslenik eşlenik almamızı sağlıyor. Yani zincirin elemanlarını tek tek okuyup zincirin ilk elemanı:

-A ise buffer'ın ilk elemanına T,

-T ise buffer'ın ilk elemanına A,

-G ise buffer'ın ilk elemanına C,

-C ise buffer'ın ilk elemanına G

Atamasını yapıp buffer'ın adresini tutan $t5 register'ını ve zincirin adresini tutan $t0 register'ına 1 ekleyerek bir sonraki byte'ı okuyor. Burada char 1 byte olduğu için adresi 1 artırdığımızda zincirin bir sonraki elemanının adresine erişip o değeri okumuş oluyoruz.

Fonksiyonun sonunda ise kullandığımız register'ları sıfırlıyoruz ve buffer'ı konsola yazdırıyoruz. (Register sıfırlama işlemi yapılmasa da olur ama okuduğum kaynakta register'ları kullanan fonksiyon boşaltmaktan sorumludur yazıyordu. Zaten aynı register'a değer atarken overwrite ediyor.)

```
.text
#EŞLENİK ALMA BAŞLANGICI
#eşlenik nasıl alınır: zincirin elemanlarını tek tek okur ve A ise bufferda karşılı
_function_eslenik:
        lw      $t1, A
        lw      $t2, T
        lw      $t3, G
        lw      $t4, C
        move    $t0, $a0    #zincir1'in adresi t0'da
        la      $t5, buffer #buffer adresi t5'te
        lb      $s1, ($t5)  #bufferın ilk elemanı
    loop:# 0. eleman A mı, T mi, G mi, C mi?
        lb      $s0, ($t0)  #zincirin ilk elemanı
        beq     $s0, $t1, elemanA   #dizinin elemanı A ise elemanA dallan
        beq     $s0, $t2, elemanT   #dizinin elemanı T ise elemanT dallan
        beq     $s0, $t3, elemanG   #dizinin elemanı G ise elemanG dallan
        beq     $s0, $t4, elemanC   #dizinin elemanı C ise elemanC dallan
        blez    $s0, print  #stringin sonuna geldiysek buffer yazdırmaya dallan
    loop2:
        addi    $t0, $t0, 1
        j       loop
    elemanA:
        move    $s1, $t2    #buffer elemanına T at
        sb      $s1, ($t5)  #buffer adresindeki değeri belleğe store et
        addi    $t5, $t5, 1 #buffer adresini 1 artır (char 1 byte)
        j       loop2
    elemanT:
        move    $s1, $t1
        sb      $s1, ($t5)
        addi    $t5, $t5, 1
        j       loop2
    elemanG:
        move    $s1, $t4
        sb      $s1, ($t5)
        addi    $t5, $t5, 1
        j       loop2
    elemanC:
        move    $s1, $t3
        sb      $s1, ($t5)
        addi    $t5, $t5, 1
        j       loop2
    print:
        #registerları boşalt..
        move    $s0, $zero
        move    $s1, $zero
        move    $t0, $zero
        move    $t1, $zero
        move    $t2, $zero
        move    $t3, $zero
        move    $t4, $zero
        move    $t5, $zero
        #..ve buffer'ı yazdır
        li      $v0, 4
        la      $a0, buffer
        syscall
        jr $ra

#EŞLENİK ALMA BİTİŞİ
```

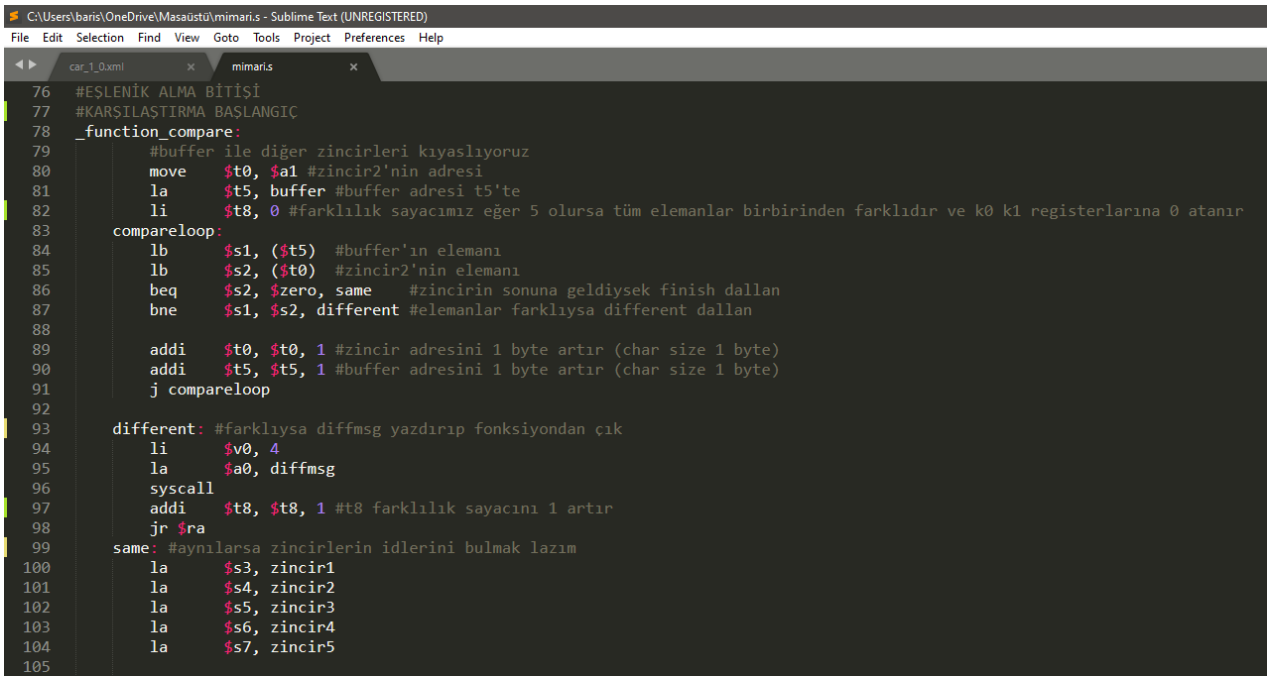
Karşılaştırma fonksiyonu tek input alıyor ve input'u buffer ile karşılaştırıyor.

Karşılaştırma kısmında da $t5 buffer adresini, $t0 zincirin adresini tutuyor. Bu adresleri 1 artırıp bir sonraki char okunuyor ve karşılaştırma yapılıyor.

(Yorum satırında zincir2 yazma sebebi 1. adımda zincir1 ve zincir2 kıyaslanıyor. Birazdan bahsedeceğim..)

Eğer buffer ve zincir farklıysa konsola diffmsg "stringler farklı" yazdırıyoruz. Eğer stringler aynıysa bu stringlerin idlerini bulmamız gerekiyor. $s3'ten $s7'ye kadarki register'lara zincirlerimizin adreslerini load ediyoruz.

$t8 register'ı farklılık sayacı. Eğer zincirler farklıysa bu değişkeni 1 artırıyoruz. 10 karşılaştırma sonunda tüm zincirler farklıysa $k0 ve $k1 register'larına 0 yazmak için kullandım.




Adres hesaplaması yapmadan önce unutmamamız gereken bir nokta var. Eşlenik bulma fonksiyonunda buffer'ı yazdırırken $a0'a buffer'ı atamıştık. Bu nedenle karşılaştırma fonksiyonunu çağırmadan önce $t9 register'ına kıyaslamak istediğimiz zincirin adresini koyuyoruz.

Karşılaştırma fonksiyonu $a1'i tek başına input olarak alıyor. Karşılaştırma işlemi buffer ile bu input arasında oluyor ve $t9 register'ı, buffer'ın oluşmasını sağlayan zincirin adresini tutuyor. Fonksiyonu çağırmadan önce $t9 register'ına zincir adresini ekliyoruz.

$k0 register'ına buffer'ı oluşturan zincirin id'si kaydediliyor. $k1 register'ına fonksiyona input olarak verilen $a1 zincirinin id'si kaydediliyor. Burada id değişkenleri kullanmak yerine eğer buffer'ı oluşturan zincir zincir1 ise $k0 register'ına 1 yazdırdık. Aynı mantık $k1 için de geçerli.

```
 97      same: #aynılarsa zincirlerin idlerini bulmak lazım
 98          la      $s3, zincir1
 99          la      $s4, zincir2
100          la      $s5, zincir3
101          la      $s6, zincir4
102          la      $s7, zincir5
103
104          #t9 register'ı bizim a0 ile gelen inputun adresini tutuyor
105          beq     $t9, $s3, findid1   #t9 zincir1 ise
106          beq     $t9, $s4, findid2   #t9 zincir2 ise
107          beq     $t9, $s5, findid3   #t9 zincir3 ise
108          beq     $t9, $s6, findid4   #t9 zincir4 ise
109          beq     $t9, $s7, findid5   #t9 zincir5 ise
110
111      nextid:
112          #a1 bizim 2. inputumuz
113          beq     $a1, $s3, findid_1  #a1 zincir1 ise
114          beq     $a1, $s4, findid_2  #a1 zincir2 ise
115          beq     $a1, $s5, findid_3  #a1 zincir3 ise
116          beq     $a1, $s6, findid_4  #a1 zincir4 ise
117          beq     $a1, $s7, findid_5  #a1 zincir5 ise
118
119      #1. inputumuz kaçıncı zincirse o id k0 registerına kaydedilir
120      findid1:
121          li  $k0, 1
122          j nextid
123      findid2:
124          li  $k0, 2
125          j nextid
126      findid3:
127          li  $k0, 3
128          j nextid
129      findid4:
130          li  $k0, 4
131          j nextid
132      findid5:
133          li  $k0, 5
134          j nextid
135
```

Karşılaştırma fonksiyonu sonunda ise samemsg ile konsola “stringler aynı,  k0 ve k1 registerlarında saklanıyor” yazdırdık. Sonra da $t7 register’ını 1 artırdık. $t7 register’ı bizim bool değişkenimiz rolünü üstleniyor. Eğer 2 zincir birbiriyle aynı ise $t7 register’ını kullanarak aramayı sonlandırıp programı durduruyoruz.

```
    #2. inputumuz kaçıncı zincirse o id k1 registerına kaydedilir
    findid_1:
        li      $k1, 1
        j return
    findid_2:
        li      $k1, 2
        j return
    findid_3:
        li      $k1, 3
        j return
    findid_4:
        li      $k1, 4
        j return
    findid_5:
        li      $k1, 5
        j return

    return:
        li      $v0, 4
        la      $a0, samemsg
        syscall
        addi $t7, $t7, 1 #bool değişkenimizi 1 artırdık
        jr $ra
```

Main içerisinde sırasıyla:

1-2 1-3 1-4 1-5 2-3 2-4 2-5 3-4 3-5 4-5

Kıyaslamaları yapıldı. Bunun için:

1. Adım: Zincir1 $a0 register'ına load edildi ve eşlenik fonksiyonuna input olarak verildi. Eşlenik fonksiyonu zincir1'in eşleniğini aldı ve buffer'a kaydetti.
2. Adım: $t9 register'ına zincir1'in adresi load edildi. Eğer karşılaştırma sonucunda aynı çıkarlarsa id'lerini bulmak için zincir1'in adresine ihtiyacımız var.
3. Adım: Zincir2 $a1 register'ına load edildi ve karşılaştırma fonksiyonuna input olarak verildi.
4. Adım: Eğer $t7 register'ı 1 ise (yani herhangi 2 zincir birbiriyle eşlenikse) aramayı durdurup exit ile programdan çıkıyoruz.

Ve bu işlemler zincir 1-2'den zincir 4-5'e kadar yapıldı.

$t7 register'ına başlangıç değeri olarak 0 atayıp taramayı başlattık.

```
162  main:
163      #zincir1 vs zincir2
164      li      $t7, 0 #bizim için bool değişkeni, eğer kıyaslama yaptıktan sonra zincirler aynıysa exite atlamamızı sağlayacak
165      la      $a0, zincir1
166      jal     _function_eslenik
167      #eşlenik hesaplandığında $a0'da buffer var!
168      #Bu nedenle zincir1'i t9 register'ında tutuyoruz ki idsini hesaplayalım.
169      la      $t9, zincir1
170      la      $a1, zincir2
171      jal     _function_compare
172      beq     $t7, 1, exit
173
174      #zincir1 vs zincir3
175      la      $a0, zincir1
176      jal     _function_eslenik
177      la      $t9, zincir1
178      la      $a1, zincir3
179      jal     _function_compare
180      beq     $t7, 1, exit
181
182      #zincir1 vs zincir4
183      la      $a0, zincir1
184      jal     _function_eslenik
185      la      $t9, zincir1
186      la      $a1, zincir4
187      jal     _function_compare
188      beq     $t7, 1, exit
189
190      #zincir1 vs zincir5
191      la      $a0, zincir1
192      jal     _function_eslenik
193      la      $t9, zincir1
194      la      $a1, zincir5
195      jal     _function_compare
196      beq     $t7, 1, exit
197
198      #zincir2 vs zincir3
```

## Çıktılar

1. ve 5. zincirler birbirinin tamamlayanı olduğu için program 1-5 karşılaştırmasını yaptıktan sonra durdu. $k0 ve $k1 register'larına 1 ve 5 değerlerini yazdı.

```
R12 [t4] = 0
R13 [t5] = 268501133
R14 [t6] = 0
R15 [t7] = 1
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 268500992
R20 [s4] = 268501003
R21 [s5] = 268501014
R22 [s6] = 268501025
R23 [s7] = 268501036
R24 [t8] = 0
R26 [k0] = 1
R27 [k1] = 5
R29 [sp] = 2147478520
R30 [s8] = 0
R31 [ra] = 4194908

SPIM Version 9.1.21 of Jan
Copyright 1990-2017 by Jam
```

```
[00400320] 0c10003a

Console

TACTACTACG
stringler farklı

TACTACTACG
stringler farklı

TACTACTACG
stringler farklı

TACTACTACG
stringler aynı, k0 ve k1 registerında saklanıyor
```

Çıktı 2: Bu senaryoda ise zincir5'in son karakterine G yerine C yazdık.

```
1  .data
2  zincir1: .asciiz "ATGATGATGC"
3  zincir2: .asciiz "TCGCGCTAGC"
4  zincir3: .asciiz "CGTCGTAAAC"
5  zincir4: .asciiz "TATTTACGAA"
6  zincir5: .asciiz "TACTACTACC" #sonu G
7
```

Zincirlerin hepsi birbirinden farklı olduğu için $k0 ve $k1 register'larında 0 ve 0 yazıyor.