1306130076 – Emre KARACA
1306130078 – Mustafa YÜNSEL
1306160090 – Özgün Doğuş CAN
1306170046 – Zeynep GÖK

BİLGİSAYAR MİMARİSİ PROJESİ

DNA Karşılaştırma

Bu projede, bize verilmiş olan 5 nükleotit zincirini bilgisayarın belleğine yerleştirip birbirlerinin karşılığı olup olmadıklarını kontrol eden bir MIPS kodu yazmamız gerekiyordu. Bunu yapmak için yazdığımız kod kabaca şunu yapıyor:

- İlk zinciri alıp, diğer dörtlü ile karşılaştırıyor,
- İkinci zinciri, sonraki diğer üç zincir ile,
- Üçüncü zinciri, sonraki diğer iki zincir ile,
- Dördüncü zinciri ise sonuncuyla karşılaştırıyor.

Böylece tüm eşlemeleri kontrol etmiş oluyor, eğer bir eşleme bulunursa o eşlemeyi kaydedip duruyor, bulamazsa programın sonunda 0 ve 0'ı iki register'a kaydederek duruyor.

Bunu yapmak için, öncelikle bize verilmiş olan 5 zinciri ASM kodunda .data segmentini kullanarak belleğe kaydettik.

```
.data

chain1: .word 'A', 'T', 'G', 'A', 'T', 'G', 'A', 'T', 'G', 'C'
chain2: .word 'T', 'C', 'G', 'C', 'G', 'C', 'T', 'A', 'G', 'C'
chain3: .word 'C', 'G', 'T', 'C', 'G', 'T', 'A', 'A', 'A', 'C'
chain4: .word 'T', 'A', 'T', 'T', 'T', 'A', 'C', 'G', 'A', 'A'
chain5: .word 'T', 'A', 'C', 'T', 'A', 'C', 'T', 'A', 'C', 'G'
```

Daha sonra .text segmentinde karşılaştırmak için gerekli kodu yazdık. Bunun için öncelikle *main* label'ında gerekli hazırlıklar yapılıyor. s1, t0 ve t1 registerlarına, ilk zincirin adresi kaydediliyor. t0 ve t1'deki değerler tek bir seferde karşılaştırma yapılan zincirlerin elemanlarını işaret ediyor olup her zaman değişecekken s1'deki değer hep sabit kalıp ilk zincirin ilk elemanının adresine işaret

edecek. t1'in değeri 40 arttırılıyor çünkü her bir zincir 10 uzunluğunda ve zincirin her bir elemanı 4 bit büyüklüğünde, bu sebeple bir sonraki zincirin adresine gitmek için 40 artırmak gerekiyor.

Daha sonra index olarak kullanacağımız 3 register belirledik. Bunlardan t7, bir zincirdeki elemanların sırasına işaret edip 1 ile 10 arasında değişecek, t8 ve t9 ise karşılaştıran zincirlerin kaçıncı olduklarını tutup 0 ile 4 arasında değişecek. Daha sonra kod t2 ve t3'e ilk ve ikinci zincirin ilk elemanlarını yazıp *nucleot* label'ına geçiyor.

Bu label, o anda kontrol edilen nükleotitlerden soldakinin yani t2'deki tutulanın hangisi olduğunu buluyor. Eğer A ise *nucA* label'ına, T ise *nucT* label'ına gibi, gerekli label'a yönlendirme yapıp, zincirde bozukluk varsa yani A T G C nükleotitleri haricinde bir şey varsa programı bitiriyor.

Gerekli label'a gidildikten sonra t3'tekinin, doğru karşılık veren nükleotit olup olmadığı kontrol ediliyor. Örneğin *nucG* label'ı, t3'te C harfinin olup olmadığına bakıyor. Eğer karşılığı varsa *next* label'ına, yoksa *rightchange* label'ına gidiliyor.

next label'ında, öncelikle t7 register'ındaki değeri 10'la karşılaştırılıyor, yani karşılaştırılan zincirlerin sonuna gelinip gelinmediği kontrol ediliyor. Eğer sonuna gelindiyse, herhangi bir bozukluk olmadan tüm karşılaştırmalar yapılmış ve iki zincir birbirini tamamlıyor demektir. Bu yüzden *save* label'ına gidiliyor.

save label'ında ise, seçtiğimiz register'lar olan s4 ve s5'e, birbirinin karşılığı olan zincir çifti yazılıyor ve program bitiriliyor.

Eğer zincirin sonuna gelinmediyse *next* label'ında öncelikle index olan t7 artırılıyor, sonra adresleri işaret eden t0 ve t1 registerları 4 artırılıyor, böylelikle zincirdeki bir sonraki nükleotite geçiliyor. Sonrasında t2 ve t3'e bu adreslerdeki değerler yazılıyor ve *nucleot* label'ına tekrar gidilerek bir döngü oluşturulmuş oluyor.

rightchange label'ında, karşılaştırmadaki sağ zincir bir sonrakine geçiyor. Bunun için ilk önce t9 register'ı aracılığıyla elimizdeki zincirlerin sonuna gelinip gelinmediği kontrol ediliyor, eğer sona gelindiyse leftchange label'ına geçiliyor. Sona gelinmediyse, sağ zinciri değiştirmek için gerekli işlemler yapılıyor. Zincirin sayısı 40la çarpılarak ilk adresi tuttuğumuz s1 register'ına ekleniyor, böylece bir sonraki zincire geçilmiş oluyor. Karşılaştırmada solda kalan zincir için de aynı işlem yapılıyor ki zincirin ilk elemanına dönülmüş olsun. Daha sonra ilk elemanların değerleri t2 ve t3'e okunduktan sonra tekrar *nucleot* label'ına gidiliyor.

Eğer ki en son beşinci zincirle karşılaştırma yapıldıysa, *leftchange* label'ına gidiliyor. Burada da aynı şekilde öncelikle dördüncü zincire gelinip gelinmediğine bakılıyor, eğer öyleyse herhangi bir

eşlenme bulunamadan program bitmiş oluyor. Sona gelinmediyse, yine aynı şekilde index'ler güncellenip, 40la çarpılarak yeni adresler belirleniyor ve t0 ve t1'e kaydedilip, t2 ve t3'e de bu adreslerdeki değerler kaydediliyor, *nucleot* label'ı tekrar çağırılıyor.

Tüm bu işlemlerin birleşiminde, elimizdeki 5 nükleotit zinciri karşılaştırılmış oluyor.

ASM kodunun kendisi ve verilen zincirlerle program çalıştırıldığında elde edilen program çıktısı:

main.asm

```
.text

main:
        la $s1, chain1
        la $t0, chain1
        la $t1, chain1
        addi $t1, $t1, 40
        addi $t7, $t7, 1
        addi $t8, $t8, 0
        addi $t9, $t9, 1
        lw $t2, 0($t0)
        lw $t3, 0($t1)
        b nucleot

next:
        beq $t7, 10, save
        addi $t7, $t7, 1
        addi $t0, $t0, 4
        addi $t1, $t1, 4
        lw $t2, 0($t0)
        lw $t3, 0($t1)
        b nucleot

rightchange:
        beq $t9, 4, leftchange
        addi $t9, $t9, 1
        addi $t7, $0, 1
        mul $t5,$t9, 40
        add $t1, $s1, $t5
        mul $t5,$t8,40
        add $t0, $s1, $t5
        lw $t2, 0($t0)
        lw $t3, 0($t1)
        b nucleot

leftchange:
        beq $t8, 3, finish
        addi $t8, $t8, 1
        addi $t9, $t8, 1
        mul $t5,$t8,40
        add $t0, $s1, $t5
        mul $t5, $t9, 40
        add $t1, $s1, $t5
        lw $t2, 0($t0)
        lw $t3, 0($t1)
        b nucleot
```

```
nucleot:
        beq $t2, 65, nucA
        beq $t2, 84, nucT
        beq $t2, 67, nucC
        beq $t2, 71, nucG
        b finish

nucA:
        beq $t3, 84, next
        b rightchange

nucT:
        beq $t3, 65, next
        b rightchange

nucC:
        beq $t3, 71, next
        b rightchange

nucG:
        beq $t3, 67, next
        b rightchange

save:
        addi $s4, $t8, 1
        addi $s5, $t9, 1
        jr $ra

finish:
        addi $s4, $s4, 0
        addi $s5, $s5, 0
        jr $ra

.data

chain1: .word 'A', 'T', 'G', 'A', 'T', 'G', 'A', 'T', 'G', 'C'
chain2: .word 'T', 'C', 'G', 'C', 'G', 'C', 'T', 'A', 'G', 'C'
chain3: .word 'C', 'G', 'T', 'C', 'G', 'T', 'A', 'A', 'A', 'C'
chain4: .word 'T', 'A', 'T', 'T', 'T', 'A', 'C', 'G', 'A', 'A'
chain5: .word 'T', 'A', 'C', 'T', 'A', 'C', 'T', 'A', 'C', 'G'
```

```
PC       = 4194336
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 10
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 2
R5  [a1] = 2147483048
R6  [a2] = 2147483060
R7  [a3] = 0
R8  [t0] = 268501028
R9  [t1] = 268501188
R10 [t2] = 67
R11 [t3] = 71
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 10
R16 [s0] = 0
R17 [s1] = 268500992
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 1
R21 [s5] = 5
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 4
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 268468224
R29 [sp] = 2147483044
R30 [s8] = 0
R31 [ra] = 4194328
```