Practica 2 MIPS

Valentino Laveggi - 6to INFO

Cuestiones

1.1) En la posición de memoria “res” se carga el valor 1.

1.2) En la posición de memoria “res” se carga el valor 0.

1.3) La comparación que se ha evaluado es: dato1<dato2.

1.4)

```
            .data
dato1:      .word 20
dato2:      .word 20
res:        .space 1

            .text
main:       lw $t0,dato1($0) # cargar dato1 en t0
            lw $t1,dato2($0) # cargar dato2 en t1
            seq  $t2,$t0, $t1 # poner a 1 $t2 si t0=t1
            sb $t2,res($0) # almacenar $t2 en res
```

1.5)

```
            .data
dato1:      .word 21
dato2:      .word 20
res:        .space 1

            .text
main:       lw $t0,dato1($0) # cargar dato1 en t0
            lw $t1,dato2($0) # cargar dato2 en t1
            sge $t3,$t0,$t1 # poner a 1 $t2 si t0>=t1
            sle $t2,$t0,$t1 # poner a 1 $t2 si t0<=t1
            and $t2,$t3,$t2
            sb $t2,res($0) # almacenar $t2 en res
```

1.6) En la posición de memoria “res” se carga el valor 1.

1.7) En la posición de memoria “res” se carga el valor 0.

1.8) En la posición de memoria “res” se carga el valor 1.

1.9) La comparación que se ha evaluado es: dato1<=dato2.

1.10)

```
            .data
dato1:      .word 40
dato2:      .word 40
res:        .space 1

            .text
main:       lw $t0,dato1($0) # cargar dato1 en t0
            lw $t1, dato2($0) # cargar dato2 en t1
            sne $t2, $t1, $t0 # poner a 1 $t2 si t0!=t1
            sb $t2,res($0) # almacenar $t2 en res
```

1.11)

```
            .data
dato1:      .word 50
dato2:      .word 40
res:        .space 1

            .text
main:       lw $t0,dato1($0) # cargar dato1 en t0
            lw $t1, dato2($0) # cargar dato2 en t1
            slt $t2, $t1, $t0 # poner a 1 $t2 si t0>t1
            bne $t0,$t1,fineval # si t0<>t1 salta a fineval
            ori $t2,$0,1 # poner a 1 t3 si t0=t1

fineval:    sb $t2,res($0) # almacenar $t2 en res
```

1.12)

```
            .data
```

```
dato1:     .word 30
dato2:     .word 30
res:       .space 1

           .text
main:      lw $t0,dato1($0) # cargar dato1 en t0
           lw $t1, dato2($0) # cargar dato2 en t1
           sle $t2, $t0, $t1 # poner a 1 $t2 si t0>=t1
           sb $t2,res($0) # almacenar $t2 en
```

1.13) En la posición de memoria “res” se carga el valor 1.

1.14) En la posición de memoria “res” se carga el valor 0.

1.15) En la posición de memoria “res” se carga el valor 0.

1.16) En la posición de memoria “res” se carga el valor 0.

1.17) La comparación compuesta que se ha evaluado es: (dato1<>0) AND (dato2<>0).

1.18)

```
           .data
dato1:     .word 10
dato2:     .word 10
res:       .space 1

           .text
main:      lw $t8,dato1($0)
           lw $t9,dato2($0)
           and $t0,$t0,$0
           and $t1,$t1,$0
           beq $t8,$0,igual
           ori $t0,$0,1

igual:     beq $t9,$t8,fineval
           ori $t1,$0,1

fineval:   and $t0,$t0,$t1
           sb $t0,res($0)
```

1.19) En la posición de memoria “res” se carga el valor 1.

1.20) En la posición de memoria “res” se carga el valor 0.

1.21) En la posición de memoria “res” se carga el valor 0.

1.22) La comparación compuesta que se ha evaluado es: (dato1<>0) AND (dato2<dato1).

1.23)

```
            .data
dato1:      .word 0
dato2:      .word 20
res:        .space 1

            .text
main:       lw $t8,dato1($0)
            lw $t9,dato2($0)
            and $t1,$t1,$0
            and $t0,$t0,$0
            beq $t8,$t9,igual
            ori $t0,$0,1

igual:      slt $t1,$t8,$t9
            bge $t8,$t9,fineval
            ori $t3,$0,1

fineval:    and $t0,$t0,$t1
            and $t3,$t0,$t3
            sb $t0,res($0)
```

1.24)

```
            .data
dato1:      .word 0
dato2:      .word 20
res:        .space 1

            .text
```

```
main:      lw $t8,dato1($0)
           lw $t9,dato2($0)
           and $t1,$t1,$0
           and $t0,$t0,$0
           beq $t8,$t9,igual
           ori $t0,$0,1

igual:     sle $t1,$t8,$t9

fineval:   and $t0,$t0,$t1
           sb $t0,res($0)
```

1.25) En la posición de memoria “res” se carga el valor 0.

1.26) En la posición de memoria “res” se carga el valor 1.

1.27) En la posición de memoria “res” se carga el valor 1.

1.28) En la posición de memoria “res” se carga el valor 0.

1.29) La comparación compuesta que se ha evaluado es: (dato1<dato2) OR (dato2==0).

1.30)

```
           .data
dato1:     .word 0
dato2:     .word 10
res:       .space 1

           .text
main:      lw $t8,dato1($0)
           lw $t9,dato2($0)
           and $t0,$t0,$0
           and $t1,$t1,$0
           sge $t0,$t9,$t8
           sge $t1,$0,$t8

fineval:   or $t0,$t0,$t1
           sb $t0,res($0)
```

1.31)

```
            .data
dato1:      .word 0
dato2:      .word 10
res:        .space 1

            .text
main:       lw $t8,dato1($0)
            lw $t9,dato2($0)
            and $t0,$t0,$0
            and $t1,$t1,$0
            sle $t0,$t8,$t9
            sle $t1,$t8,$0

fineval:    or $t0,$t0,$t1
            sb $t0,res($0)
```

Actividades

1)

```
            .data
vector:     .byte 0,1,1,1,0
res:        .space 1

            .text
main:       lb $t1, vector($0)
            lb $t2, vector+1($0)
            lb $t3, vector+2($0)
            lb $t4, vector+3($0)
            lb $t5, vector+4($0)

            and $t6, $t1, $t5
            or $t7, $t2, $t4
            or $t8, $t1, $t2
            and $t8, $t8, $t3

            sb $t1, res($0)
```

```
            sb $t2, res+1($0)
            sb $t3, res+2($0)
```

2)

```
            .data
enteros:    .byte 2,-4,-6
            .space 1
bool:

            .text
main:       lb $t0, enteros($0)
            lb $t1, enteros+1($0)
            lb $t2, enteros+2($0)

            sge $t3,$t0,$0
            sge $t4,$t1,$0
            sge $t5,$t2,$0

            sb $t3, bool($0)
            sb $t4, bool+1($0)
            sb $t5, bool+2($0)
```

3)

```
            .data
enteros: .byte 1,-4,-5,2
            .space 1
bool:

            .text
main:       lb $t0, enteros($0)
            lb $t1, enteros+1($0)
            lb $t2, enteros+2($0)
            lb $t3, enteros+3($0)

            slt $t8,$t0,$0
            slt $t9,$t1,$0
            and $t9,$t8,$t9
            slt $t8,$t2,$0
            and $t9,$t8,$t9
```

```
            slt $t8,$t3,$0
            and $t9,$t8,$t9

            sb $t9, bool($0)
```

Cuestiones

2.1) La instrucción que evalúa la condición y controla el flujo del programa es beq, ya que según el resultado de la comparación realiza un salto.

2.2) El conjunto de instrucciones que implementan la estructura if-then, son las que se encuentran en las etiquetas “si” y “entonces”, ya que, en “si” se encuentra la instrucción beq que realiza una comparación, la cual, de ser cierta, realiza un salto omitiendo “si”, pero en cambio si no lo es, el programa realiza las instrucciones de “si”, funcionando como la estructura then.

2.3) Luego de ejecutar el programa la variable res almacena el valor 71.

2.4) Si dato2 = 0, luego de la ejecución del programa la variable res almacena el valor 0.

2.5)

```
            .data
dato1:      .word 40
dato2:      .word 30
res:        .space 4

            .text
main:       lw $t0,dato1($0)
            lw $t1,dato2($0)
            and $t2,$t2,$0

Si:         ble $t1,$0,finsi

entonces:   div $t0,$t1
```

```
            mflo $t2

finsi:      add $t3,$t0,$t1
            add $t2,$t3,$t2
            sw $t2,res($0)
```

2.6)

```
VARIABLES
      ENTERO: dato1=40; dato2=30; res
INICO
      Si ((dato2!=0) AND (dato1!=0)) res=dato1/dato2;
      res = res + dato1 + dato2;
FIN
```

2.7) Las instrucciones que evalúan la condición y controlan el flujo del programa son las que se encuentran en la etiqueta “si”, las cuales constan de las instrucción beg, en las que, si se cumple la condición de alguna de ellas el programa realizará un salto, de modo tal que comparándolo con el pseudocódigo, funcionan como un if el cual posee dos condiciones en AND.

2.8) El conjunto de instrucciones que implementan la estructura if-then, son las que se encuentran en las etiquetas “si” y “entonces”, ya que, en “si” se encuentran dos instrucciones beq las cuales cada una realiza una comparación, de ser cierta alguna de ellas, realiza un salto omitiendo “si”, pero en cambio si ninguna cumple la condición, el programa realiza las instrucciones de “si”, funcionando como la estructura then.

2.9) Luego de ejecutar el programa la variable res almacena el valor 71.

2.10) Si dato1 = 0, entonces en res se almacenará dato2, en cambio si dato2 = 0, se almacenará dato1.

2.11)

```
            .data
dato1:      .word 0
dato2:      .word 0
res:        .space 4
```

```
            .text
main:       lw $t0,dato1($0)
            lw $t1,dato2($0)
            and $t2,$t2,$0

Si:         ble $t1,$0,finsi
            blt $t0,$0,finsi

entonces:   div $t0,$t1
            mflo $t2

finsi:      add $t3,$t0,$t1
            add $t2,$t3,$t2
            sw $t2,res($0)
```

2.12)

```
VARIABLES
      ENTERO: dato1=40; dato2=30; res
INICO
      Si (dato1<dato2)
      Entonces
            res=dato1;
      Sino
            res=dato2;
FIN
```

2.13) En res se almacena dato1, puesto que se cumple que dato1<dato2, por lo que inicialmente en res se almacena 30. En cambio, si dato1=35, en res se almacenará 35, pues sigue siendo menor a 40.

2.14) La pseudoinstrucción bge, implementa las instrucciones slt y beq.

2.15)

```
            .data
dato1:      .word 45
dato2:      .word 40
```

```
res:        .space 4

            .text
main:       lw $t0,dato1($0) #cargar dato1 en $t0
            lw $t1,dato2($0) #cargar dato2 en $t1

Si:         blt $t0,$t1, sino #si $t0>=$t1 ir a sino

entonces:   sub $t2,$t0,$t1
            sw $t2,res($0) #almacenar $t0 en res
            j finsi #ir a finsi

sino:       sub $t2,$t1,$t0
            sw $t2,res($0)#almacenar $t1 en res

finsi:
```

2.16)

```
VARIABLES
      ENTERO: dato1=40; dato2=30; dato3=-1; res
INICO
      Si ((dato3<dato1) OR (dato3>dato2)
      Entonces
            res=1;
      Sino
            res=0;
FIN
```

2.17) Luego de ejecutar el programa se almacena en res el valor 1. Igualmente, cuando dato1=40 y dato2=30 el valor almacenado en res será 1.

2.18)

```
            .data
dato1:      .word 10
dato2:      .word 30
dato3:      .word 20
res:        .space 4
```

```
            .text
main:       lw $t1,dato1($0)
            lw $t2,dato2($0)
            lw $t3,dato3($0)

Si:         blt $t3,$t1, sino
            bgt $t3,$t2, sino

entonces:   addi $t4,$0,1
            j finsi

sino:       and $t4,$0,$0

finsi:      sw $t4,res($0)
```

2.19) En este programa se puede observar un contador que almacena la cantidad de caracteres que posee la etiqueta “cadena” (en este caso “hola”), para ello, inicialmente almacena los valores de la memoria en el registro, donde $t0 = cadena. Luego mediante andi asignamos el valor 0 en $t2.

A continuación de ello, nos encontramos en la etiqueta “mientras”, lugar donde sucede la magia. Al inicio se encuentra lb, el cual simula el contador de un for/while, almacenando bytes de $t0 en $t1, de modo tal que el ciclo termina cuando $t1==0 mediante beq, el cual realiza un salto saliendo del ciclo cuando esto suceda.

Ahora, dentro del ciclo, se encuentra el sumador de $t2, el cual será el que almacenaremos en “n” indicándonos la cantidad de caracteres en “cadena”. Mientras que a su vez, se encuentra el sumador de $t0, el cual sirve para el contador mencionado anteriormente. Y por último la instrucción j, que reinicie el ciclo.

2.20) Luego de ejecutar el programa se almacena en n la cantidad de caracteres que posee la etiqueta “cadena”, en este caso: 4.

2.21)

```
            .data
tira1:      .asciiz "hola"
tira2:      .asciiz "adios"
            .align 2
n:          .space 4
```

```
            .text

main:       la $t0,tira1
            la $t1,tira2
            andi $t4,$t4, 0

mientras:   lb $t2,0($t0)
            lb $t3,0($t1)
            beq $t2,$0,finmientras
            beq $t3,$0,finmientras
            addi $t4,$t4, 1
            addi $t0,$t0, 1
            addi $t1,$t1, 1
            j mientras

finmientras: sw $t4,n($0)
```

2.22) En este programa se puede observar un contador el cual realiza una suma de todos los valores del array “vector” para almacenarlo en res. Para ello, el programa inicia almacenando los valores de la memoria en el registro, en $t2 almacena contador, en $t3 almacena 0, ya que será donde se almacenará la suma; en $t0 almacena el contador, y en $t1 el largo de vector.

Posteriormente se encuentra la etiqueta “para”, la cual simula un for. En ella se encuentra inicialmente el condicional, mediante un bgt, el cual indica que si $t0 (contador) supera a $t1 realice un salto fuera del ciclo. Si no se cumple la condición, en “para” se almacena en $t4 el valor de vector en la posición indicada por $t2, luego, en $t3, le suma el valor de $t4. A continuación, se le suma a $t2 4, para de este modo poder pasar al siguiente valor de “vector”, y se le suma uno al contador $t1, Para finalmente pasar a la instrucción j y repetir el ciclo.

Una vez finalizado el valor de $t3 se almacena en la posición de memoria res.

2.23) Luego de ejecutar el código, en res se almacena la suma de todos los valores de “vector”, en este caso: 41.

2.24)

```
            .data
```

```
vector1:   .word 6,7,8,9,10,-1,34,23
vector2:   .space 4

           .text
main:      la $t2,vector1
           and $t3,$0,$t3
           li $t0,0
           li $t1,7

para:      bgt $t0,$t1,finpara
           lw $t4,0($t2)
           add $t4,$t4,1
           sw $t4,vector2($t3)
           addi $t3,$t3, 4
           addi $t2,$t2, 4
           addi $t0,$t0, 1
           j para

finpara:
```

Actividades

4)

```
           .data
dato1:     .word 1
dato2:     .word 10
dato3:     .word 50
dato4:     .word 70
dato5:     .word 34
res:       .space 1

           .text
main:      lw $t0,dato1($0)
           lw $t1,dato2($0)
           lw $t2,dato3($0)
           lw $t3,dato4($0)
           lw $t4,dato5($0)
           and $t5,$0,$t5
```

```
            and $t6,$0,$t6
            bge $t0,$t1,mayor

menor:      sle $t7,$t4,$t1
            sge $t5,$t4,$t0
            and $t5,$t5,$t7
            j sig

mayor:      sge $t7,$t4,$t1
            sle $t5,$t4,$t0
            and $t5,$t5,$t7

sig:        bge $t2,$t3,mayor1

menor1:     sle $t8,$t4,$t3
            sge $t6,$t4,$t2
            and $t6,$t6,$t8
            j fin

mayor1:     sge $t8,$t4,$t3
            sle $t6,$t4,$t2
            and $t6,$t6,$t8

fin:        or $t9,$t5,$t6
            sw $t9,res($0)
```

5-

```
            .data
vector1:    .word 6,0,8,9,0,0,34,0
total:      .space 4

            .text
main:       la $t2,vector1
            and $t3,$0,$t3
            and $t5,$0,$t5
            li $t0,0
            li $t1,7

para:       bgt $t0,$t1,finpara
            lw $t4,0($t2)
```

```
            seq $t3,$t4,0
            addi $t2,$t2, 4
            addi $t0,$t0, 1
            add $t5,$t5,$t3
            j para

finpara:    sw $t5,total($0)
```

6-

```
            .data
vector1:    .word 6,56,8,5,2,9,6,7
rango1:     .word 9
rango2:     .word 5
res:        .space 4

            .text
main:       la $t2,vector1
            lw $t6,rango1
            lw $t7,rango2
            li $t0,0
            li $t1,7
            and $t8,$0,$t8

para:       bgt $t0,$t1,finpara
            lw $t4,0($t2)
            addi $t2,$t2, 4
            addi $t0,$t0, 1
            and $t3,$0,$t3
            and $t5,$0,$t5

            bge $t6,$t7,mayor

menor:      sle $t3,$t4,$t7
            sge $t5,$t4,$t6
            and $t5,$t5,$t3
            add $t8,$t8,$t5
            j para

mayor:      sge $t3,$t4,$t7
            sle $t5,$t4,$t6
            and $t5,$t5,$t3
```

```
            add $t8,$t8,$t5
            j para

finpara:    sw $t8,res($0)
```

7-

```
            .data
caracteres: .ascii "Cadena de caracteres"
caracter: .ascii "a"
            .align 2
total:      .space 1

            .text
main:       la $t0,caracteres
            lb $t1,caracter
            and $t5,$t5,$0

while:      lb $t2,0($t0)
            beq $t2,$0,fin
            addi $t0,$t0, 1
            seq $t4,$t1,$t2
            add $t5,$t5,$t4
            j while

fin:        add $t5,$t5,-1
            sb $t5,total($0)
```