Instituto Politécnico Superior “General San Martín”

Tecnicatura en Informática Profesional y Personal

INSTALACIÓN Y REEMPLAZO DE COMPONENTES INTERNOS

Docente: Alejandro Rodríguez Costello

Trabajo de MIPS 2

Estudiante Santino Zanin

Fecha de entrega: 15/09/2022

Consignas 1:

1) En res se carga el valor 1.
2) Al cambiar los valores de $t0 y $t1, ahora en res se carga el valor 0.
3) Mediante la instrucción de comparación slt, se evalúa si el valor de $t0 es menor que el valor de $t1
4) Para ello se reemplaza la instrucción slt por seq.
5)

```
.data
dato1: .word 30
dato2: .word 40
res: .space 1
.text
main: lw $t0,dato1($0) # cargar dato1 en t0
lw $t1,dato2($0) # cargar dato2 en t1
sge $t2, $t0, $t1
sge $t3, $t1, $t0 # poner a 1 $t2 si t0<t1
and $t4, $t2, $t3
sb $t2,res($0) # almacenar $t2 en res
```

6) En la posición de memoria res se carga el valor 1
7) Cambiando los valores de dato1 y dato2 por 50 y 20, el valor que se carga en res es 0.
8) Si dato1 y dato2 tienen valores iguales, el valor que se carga en res es 1.
9) Se hacen dos comparaciones entre dato1 y dato2, primero con el comando slt, el cual compara si dato1 es menor que dato2, y luego con el comando bne, que compara si los dos valores son iguales.
10)

```
.data
dato1: .word 30
dato2: .word 40
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
sb $t2,res($0) # almacenar $t2 en res
```

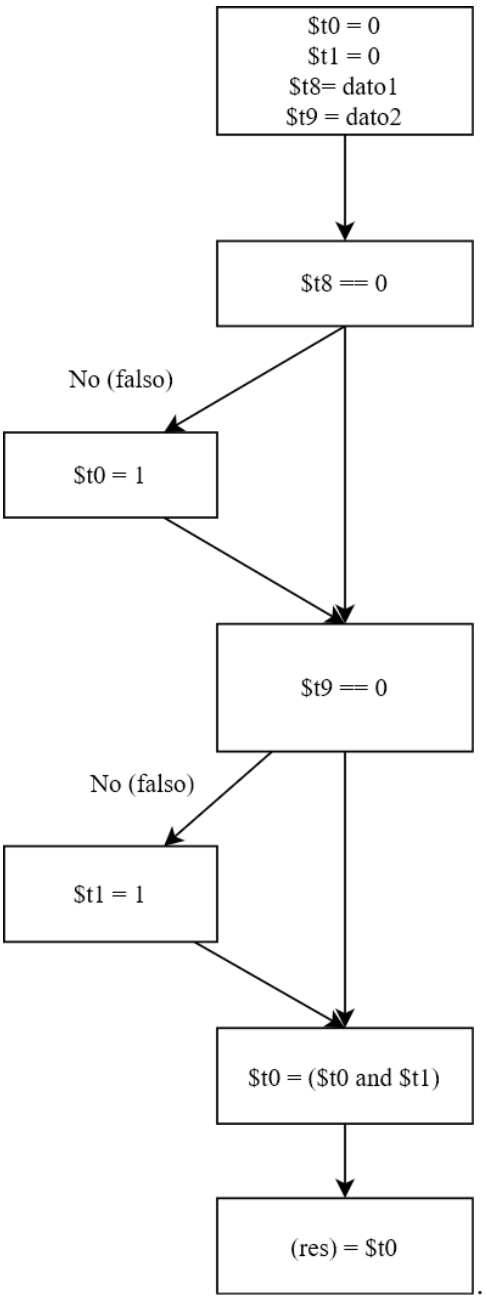
11)

```
.data
dato1: .word 30
dato2: .word 40
res: .space 1
.text
main: lw $t0,dato1($0)
lw $t1, dato2($0)
sgt $t2, $t0, $t1
bne $t0,$t1,fineval
ori $t2,$0,1
fineval: sb $t2,res($0)
```

12)

```
.data
dato1: .word 30
dato2: .word 40
res: .space 1
.text
main: lw $t0,dato1($0) # cargar dato1 en t0
lw $t1, dato2($0) # cargar dato2 en t1
sge $t2, $t0, $t1 # poner a 1 $t2 si t0<t1
sb $t2,res($0) # almacenar $t2 en res
```

13) En el espacio de memoria res se almacena el valor 1



14) El valor que se carga en res es 0.
15) De nuevo, el valor que se carga en res ahora es 0.
16) Ahora también el valor que se carga en res es 0.
17) En la comparación entre dato1 y dato2 primero se utiliza la instrucción beq para verificar si ellos son iguales a cero, luego se comparan los resultados con el operador and.
18)

```
.data
dato1: .word 40
dato2: .word -50
```

```
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 $t8,$t9,fineval    # Compara dato1 y dato2
ori $t1,$0,1
fineval: and $t0,$t0,$t1
sb $t0,res($0)
```

19) Se almacena el valor 1 en el espacio de memoria, ya que dato2 es menor a dato1.
20) En este caso se almacena 0 en res, ya que dato1 es menor a dato2.
21) De nuevo, se almacena el valor 0 en res.
22) Primero se verifica si dato1 es desigual a cero con la instrucción beq, luego si dato2 es menor a dato1 con la instrucción slt, y por último se comparan los resultados con and.
23)

```
.data
dato1: .word 30
dato2: .word -50
res: .space 1
.text
main: lw $t8,dato1($0)
lw $t9,dato2($0)
and $t1,$t1,$0
and $t0,$t0,$0
bne $t8,$t9,igual
ori $t0,$0,1
igual: sge $t1,$t9,$t8
fineval: and $t0,$t0,$t1
sb $t0,res($0)
```

24)

```
.data
dato1: .word 30
```

```
dato2: .word -50
res: .space 1
.text
main: lw $t8,dato1($0)
lw $t9,dato2($0)
and $t1,$t1,$0
and $t0,$t0,$0
bne $t8,$t9,igual
ori $t0,$0,1
igual: sle $t1,$t8,$t9
fineval: and $t0,$t0,$t1
sb $t0,res($0)
```

25) Se guarda el valor 0 en res.
26) En este caso, se guarda 1 en res.
27) En este también se guarda el valor 1 en res.
28) Ahora se almacena el valor 0 en res.
29) Primero se verifica que dato1 sea menor a dato2, luego que dato2 sea igual a 0, por último se comparan estos resultados con or *(dato1 < dato2 || dato2 == 0)*.
30)

```
.data
dato1: .word 30
dato2: .word -20
res: .space 1
.text
main: lw $t8,dato1($0)
lw $t9,dato2($0)
and $t0,$t0,$0
and $t1,$t1,$0
sge $t0,$t8,$t9
ble $t9,$0,fineval
ori $t1,$0,1
fineval: or $t0,$t0,$t1
sb $t0,res($0)
```

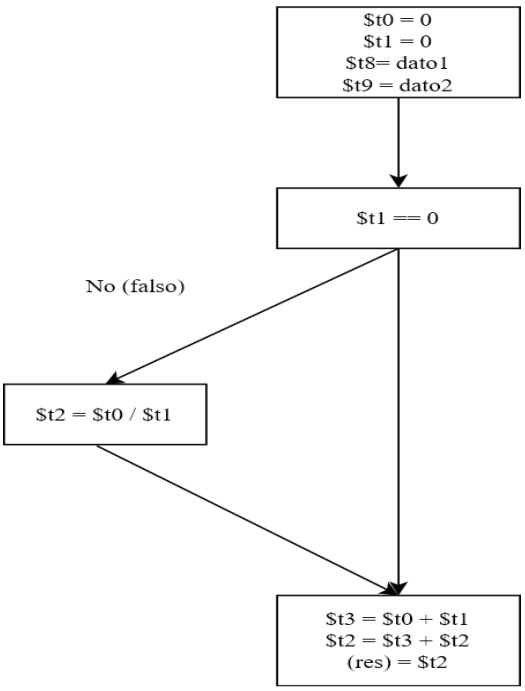
31)

```
.data
dato1: .word 30
```

```
dato2: .word -20
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 $t9,$0,fineval
fineval: or $t0,$t0,$t1
sb $t0,res($0)
```

Consignas 2:

1) Esta instrucción es beq, la cual compara dos registros, en caso de ser iguales, saltan a la dirección finsi. En el caso del pseudocódigo, se compara el registro $t1 con el entero 0, en caso de ser iguales salta hacia fuera del condicional if.
2) Las instrucciones que se implementan en la estructura if-then son beq y div, una utilizada para el condicional, y la otra cumpliendo la función a realizar si el condicional no se cumple.
3) En res se almacenará el valor entero 71.
4) En este caso, se almacenará el entero 40.



5)

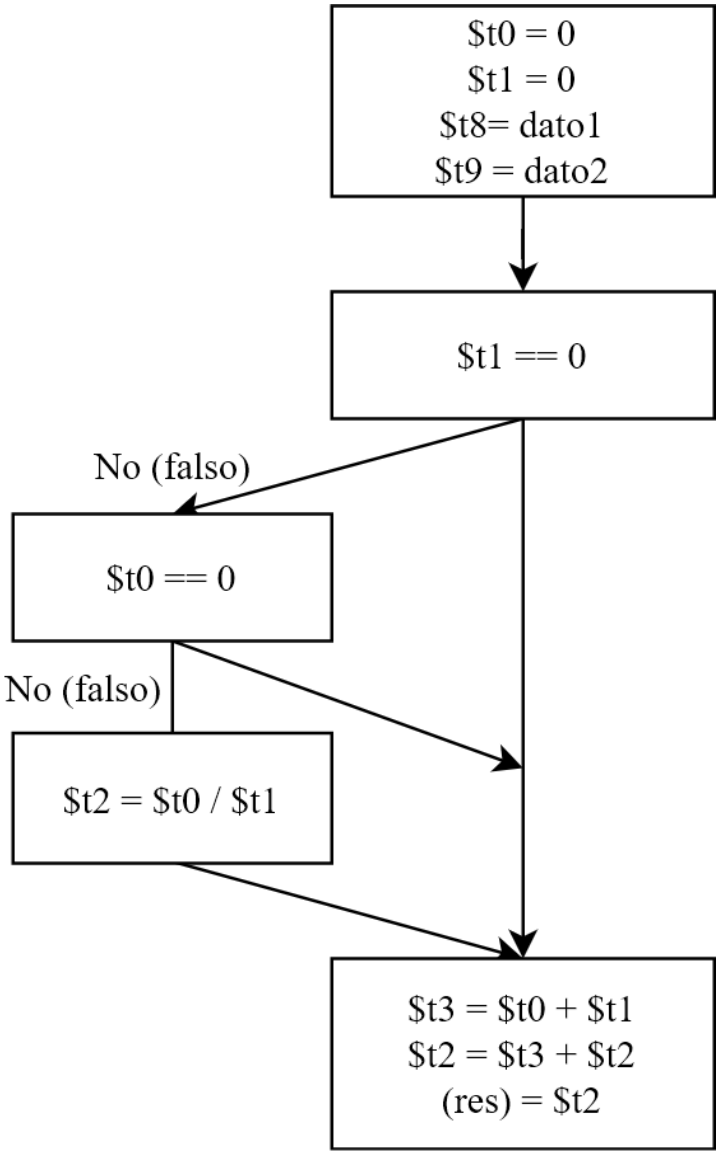
```
.data
dato1: .word 40
dato2: .word 30
res: .space 4
.text
main: lw $t0,dato1($0)
lw $t1,dato2($0)
and $t2,$t2,$0
Si: bgt $t1,$0,finsi
entonces: div $t0,$t1
mflo $t2
finsi: add $t3,$t0,$t1
add $t2,$t3,$t2
sw $t2,res($0)
```

6)

```
VARIABLES
      ENTERO: dato1 = 40, dato2 = 30, res;
INICIO
      Si (dato2 == 0 || dato1 == 0) res = dato1 / dato2;
       res = res + dato1 + dato2;
FIN
```

7) La instrucción que evalúa la condición es beq, utilizada dos veces para evaluar si dato1 o dato2 son iguales. En el pseudocódigo se utiliza el operador or y el operador de igualdad (==). El de igualdad se utiliza para evaluar si dato1 o dato2 son iguales a cero, los resultados de estas evaluaciones luego se comparan con el operador or.
8) La instrucción que se implementa para la estructura condicional if-then es beq, la cual funciona comparando dos valores y determinando si uno de ellos es menor que el otro (esto depende del orden en el que se ingresen).

Diagrama de flujo:



9) El valor almacenado en la variable res es 71.
10) Si cambiamos el valor de dato1 para que resulte dato1 = 0, en la variable res se

almacenará el valor 30. Y en el caso de que dato2 = 0, res guardará el valor 40.

11)

```
dato1: .word 40
dato2: .word 30
res: .space 4
.text
main: lw $t0,dato1($0) #cargar dato1 en $t0
lw $t1,dato2($0) #cargar dato2 en $t1
and $t2,$t2,$0 #t2=0
blt $t1,$0, salto #si $t1 > 0
ble $t0,$0 salto #si %$t0 <= 0
div $t0,$t1
mflo $t2 #almacenar LO en $t2
salto: add $t3,$t0,$t1 #$t3=$t0+$t1
add $t2,$t3,$t2 #$t2=$t3+$t2
sw $t2,res($0) #almacenar en res $t2
```

12)

```
VARIABLES
      ENTERO: dato1 = 40, dato2 = 30, res;
INICIO
      Si (dato1 >= dato2) res = dato2;
       Sino res = dato1;
FIN
```

13) Al ejecutar el programa, res almacenarán 30. En el caso de que dato1 = 35, res almacenará 35, ya que este valor sigue siendo menor al de dato2.

14) Primero se llama a la instrucción slt, que compara el valor de dos registros temporales ($8 y $9) de esta forma: $8 < $9, el resultado de esta comparación se almacena en el registro $1. Luego se utiliza la instrucción beq, la cual compara el registro $1 y $0 de forma $1 == $0, en caso de ser verdadero salta hacia la etiqueta 4.

15)

```
.data
dato1: .word 30
```

```
dato2: .word 40
res: .space 4
.text
main: lw $t0,dato1($0) #cargar dato1 en $t0
lw $t1,dato2($0) #cargar dato2 en $t1
and $t2,$t2,$0
blt $t0,$t1, sino #si $t0 < $t1 ir a sino
sub $t2, $t0, $t1
j saltar #ir a finsi
sino: sub $t2, $t1, $t0
saltar: sw $t2,res($0) #almacenar $t2 en res
```

16)

```
VARIABLES
      ENTERO: dato1 = 40, dato2 = 30, dato3 = -1, res;
INICIO
      Si (dato3 < dato1) res = 1;
       Sino
           Si (dato3 <= dato2) res = 0;
FIN
```

17) El valor que se almacenará en res es 1, ya que el valor de dato3 es menor al de dato1. Si dato1 = 40 y dato2 = 30, res seguiría almacenando el mismo valor, ya que dato3 almacena un valor menor que los dos.

18)

```
dato2: .word 40
dato3: .word -1
res: .space 4
.text
main: lw $t1,dato1($0) #cargar dato1 en $t1
lw $t2,dato2($0) #cargar dato2 en $t2
```

```
lw $t3,dato3($0) #cargar dato3 en $t3

bge $t3,$t1, then1 #si $t3 < $t1 ir a then1
j finsi
then1:      ble $t3,$t2, then2 #si $t3 <= $t2 ir a then2
j finsi
then2:      addi $t4, $0, 1    #carga el valor 1 en $t4
finsi:      sw $t4,res($0)
```

19) Lo primero que sucede es la carga de la cadena en un registro temporal, además de la limpieza de $t2 para su uso como contador. Luego inicia el bucle mediante el uso de la etiqueta *mientras*, la cual almacena un byte de la cadena en $t1, para después comparar el valor de $t1 y $t0, si son diferentes vuelve a la etiqueta *mientras*, pero si no lo son, aumenta en 1 a $t2 y $t0, permitiendo el recorrido por la cadena. Cuando $t1 y $t0 sean iguales, salta a la etiqueta finmientras, la cual almacena el contenido de $t2 en el espacio de memoria n.

20) El valor que se almacena en el espacio de memoria n es 4, ya que el contador aumentó hasta este número para igualar la cantidad de caracteres de la cadena.

21)

```
.data
tira1: .asciiz "hola"
tira2: .asciiz "adios"
.align 2
n: .space 4
.text
main: la $t0, tira1
la $t1, tira2
andi $t2,$t2, 0
while: lb $t3,0($t0)
lb $t4,0($t1)

beq $t3, $0, fin
beq $t4, $0, fin
addi $t2,$t2,1
addi $t3,$t3,1
addi $t4,$t4,1
j while
```

```
fin: sw $t2,n($0)
```

22) Al ejecutar el programa, primero se carga un vector de 6 elementos enteros en $t2, $t3 se limpia, se inicia $t0 y $t1, este último guardando la longitud del vector. Luego se inicia la primera etiqueta, *para*, dentro de la cual se desarrollará el bucle. La primera instrucción dentro de *para* es bgt, que en el caso de que $t0 sea mayor a $t1, terminará el bucle. La siguiente instrucción es lw, la cual guarda la dirección del vector en $t4, para luego con la instrucción add, guardar la suma de $t4 y $t3 en $t3, además de aumentar $t2 en 4, permitiendo el recorrido por el vector. Por último se aumenta el contador de $t0 en 1 y se repite el bucle. Con el bucle terminado mediante el salto a la etiqueta finpara, se almacena en res el resultado de la suma de todos los elementos del vector.

23) Al ejecutar el código anterior, res almacenará el valor 41.

24)

```
.data
v1: .word 6,7,8,9,10,-1,34,23
v2: .space 32
.text
main: la $t0,v1
la $t1,v2

li $t2,0
li $t3,8
para: bgt $t2,$t3,finpara #si $t2>$t3 saltar finpara
lw $t4,0($t2) #carga elemento vector en $t4
addi $t4, $t4, 1

sw $t4, 0($t1)
addi $t0,$t0,4 #$t0=$t0+4
addi $t1,$t1,1 #$t1=$t1+1
addi $t2,$t2,1 #$t2=$t2+1
finpara:
```