### Laboratorio 2

**Ensamblador de MIPS y subrutinas**

*Arquitectura de Ordenadores*

En este laboratorio se estudiará la codificación de instrucciones en código máquina, el protocolo de llamada a subrutinas y el uso de *syscalls*.

*Ejercicio 1 (3 puntos)*

Copie el siguiente código. Este programa utiliza llamadas a subrutinas para calcular el número de valores en un array mayores a uno dado. Ejecutando el programa sin *breakpoints*, los registros $s0 y $s1 deberían tener respectivamente el número de valores mayores a 5 del array *X* y el número de valores mayores a 50 de *Y*. Modifique los valores y ejecute el programa varias veces hasta que esté seguro de entender cómo funciona. Después conteste a las preguntas.

.data

X: .word 1,2,6,8,9,3,5,10

Y: .word 10,20,35,11,99,30,5,100

.text

.globl main

main:

la $a0, X

li $a1, 5

li $a2, 8

jal CountGreaterThan

addi $s0, $v0, 0

la $a0, Y

li $a1, 50

li $a2, 8

jal CountGreaterThan

addi $s1, $v0, 0

# End

li $v0, 10

syscall

# Contar numero de valores mayores que el argumento

# a0 – direccion del array

# a1 – numero para comparar

# a2 – numbero de elementos del array

# Return – numero contado

# Usa - t1, t2, t3

CountGreaterThan:

# Guardar $ra

addi $sp, $sp, -4

sw $ra, 0($sp)

# Inicializar

addi $t1, $a0, 0

addi $t2, $a2, 0

li $t3, 0

# Bucle

loopGreaterThan:

lw $a0, 0($t1)

jal IsGreaterThan

beq $v0, $zero, updateAddress

addi $t3, $t3, 1

updateAddress:

addi $t1, $t1, 4

addi $t2, $t2, -1

bne $t2, $zero, loopGreaterThan

# Recuperar $ra

lw $ra, 0($sp)

addi $sp, $sp, 4

# Return

addi $v0, $t3, 0

jr $ra

# Comprobar si a0 > a1

# a0 - numero

# a1 - numero

# Return - 1 si es mayor

# Usa - t0

IsGreaterThan:

addi $t0, $a0, -1

slt $v0, $t0, $a1

xori $v0, $v0, 1 # Not(a <= b) === a > b

jr $ra

1. La comprobación de “mayor que” no necesita ejecutarse en una subrutina y no se tiene que implementar con un XOR. Se puede programar utilizando la instrucción slt (*set if less than*) y colocando cuidadosa-mente los registros y las condiciones bne/beq. Explique cómo implementaría esta condición.
2. Explique por qué es necesario guardar $ra al principio de la subrutina CountGreaterThan. Localice la zona de memoria donde se guarda en MARS e incluya una captura de pantalla con el valor de $ra guardado. Puede poner un breakpoint en medio de CountGreaterThan para parar el programa.

Compruébelo siguiendo estos pasos:

1. Comente las líneas marcadas en azul en el código (es decir, añada un símbolo # delante para que no se ejecuten).
2. Ponga un breakpoint antes de la línea jr $ra al final de CountGreaterThan.
3. Ejecute hasta el *breakpoint* ().
4. Ejecute paso a paso ().
5. ¿A qué instrucción vuelve el programa? Explique lo que ha pasado. ¿Es esto lo esperado?

*Ejercicio 2 (3 puntos)*

Modifique el programa anterior para cumplir estas condiciones. Cada modificación es independiente del resto.

1. Escriba un código en lenguaje de alto nivel que sea equivalente a las funciones main y CountGreaterThan.
2. Implemente en el main las llamadas al sistema necesarias para leer el número para comparar de teclado y para imprimir los resultados en la pantalla.
3. En la subrutina CountGreaterThan se quiere utilizar los registros $s1, $s2 y $s3 en lugar de $t1, $t2 y $t3. Los registros $sX son registros que debe ser preservados en llamadas a funciones. ¿Qué tendría que cambiar en esa función?

*Ejercicio 3 (2 puntos)*

Introduzca el siguiente código en MARS y responda las siguientes preguntas:

li $t0, 16

li $t1, 0x100234

lw $t2, 4($t1)

bne $t2, $t0, different

addi $s0, $t0, -2

j endIf

different:

add $s0, $t0, $t2

endIf:

sw $s0, 4($t1)

1. ¿Qué instrucciones reales se corresponden a la instrucción li $t0, 16? ¿Y a li $t1, 0x100234?
2. Explique qué hacen estas dos instrucciones y por qué la misma pseudoinstrucción (li) se traduce de dos formas diferentes.

*Ejercicio 4 (2 puntos)*

Sobre el código anterior, en la pestaña **Execute**, ventana “**Text Segment**”, observe la columna “**Code**”. Esta columna da la instrucción máquina completamente ensamblada correspondiente al código fuente (en hexadecimal).

1. Para las siguientes instrucciones, rellene los campos correspondientes en las siguientes tablas e indique en la primera fila qué función desempeña cada campo y a qué parte de la instrucción se refiere, siguiendo el ejemplo.
2. Para las instrucciones bne y j explique qué quiere decir el campo de más a la derecha. Relacione las etiquetas different y endIf con el valor de ese campo.

***Ejemplo:*** addi $s0, $t0, -2

|  |  |  |  |
| --- | --- | --- | --- |
| ***Opcode (addi)*** | ***rs ($t0)*** | ***rt ($s0)*** | ***Inmediato (-2)*** |
| ***001000*** | ***01000*** | ***10000*** | ***1111111111111110*** |

add $s0, $t0, $t2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

sw $s0, 4($t1)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

bne $t2, $t0, different

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

j endIf

|  |  |
| --- | --- |
|  |  |
|  |  |