**Práctica 3**

**Ejercicio 1: Las llamadas al sistema y las funciones de los programas**

* ¿Cuáles son las dos últimas instrucciones del programa principal?

Carga el inmediato 10 en $v0, es decir, para realizar la llamada de *exit* para acabar el proceso y, en la línea siguiente, utiliza la llamada al sistema (*syscall*) para efectuar la operación.

* ¿Cuál es la última instrucción de la función (Mult)?

La última instrucción es: **beqz $a1, MultRet.**

Lo que hace esta instrucción es comparar si $a1 = 0, y si lo es, salta a la función MultRet.

* Busque las cuatro llamadas al sistema utilizadas en el programa. ¿Qué hace cada una?
  + **Llamada 1:**

**li $v0, 5**

**syscall**

**Explicación:** Carga el inmediato 5 en $v0, es decir, para leer un entero (*read\_int*) y utiliza la llamada al sistema (*syscall*) para efectuar la operación.

* + **Llamada 2:**

**li $v0, 5**

**syscall**

**Explicación:** Carga el inmediato 5 en $v0, es decir, para leer otro entero (*read\_int*) y utiliza la llamada al sistema (*syscall*) para efectuar la operación.

* + **Llamada 3:**

**li $v0, 1**

**syscall**

**Explicación:** Carga el inmediato 1 en $v0, es decir, para imprimir el valor de un entero (*print\_int)* y utiliza la llamada al sistema (*syscall*) para efectuar la operación.

* + **Llamada 4:**

**li $v0,10**

**syscall**

**Explicación:** Carga el inmediato 10 en $v0, es decir, para acabar el proceso (*exit)* y utiliza la llamada al sistema (*syscall*) para efectuar la operación.

* Busque un bucle dentro de la función. ¿Cuántas veces se ejecuta este bucle? Se ejecuta tantas veces como el valor de $a1.
* ¿Qué hace la función exactamente?

1. Suma a $v0 el valor de salida $a0.
2. Reduce el contador $a1 en una unidad.
3. Compara si $a1 = 0; Si no lo es, empieza el bucle de nuevo; Si lo es, acaba de el bucle.

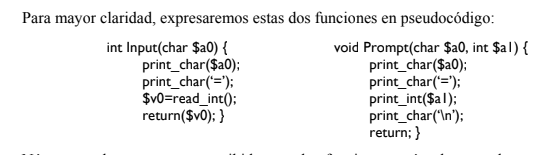
* Carga el programa y ejecútalo. Nota que la entrada/salida por la consola es muy pobre. OK.
* ¿Sabe ejecutar el programa completo? Al ejecutarlo, tenga en cuenta que el programa pide la entrada de dos números por el teclado y luego imprime un resultado. Ahora bien, no habrá ningún mensaje que indique que se está esperando una entrada del teclado. Sí. Para ejecutar el programa completo, se debe pulsar la tecla **F5 (Go)**.
* ¿Sabe hacer una ejecución paso a paso? Sí. Para hacer una ejecución paso a paso, se deberá pulsar la tecla **F10 (Step)**.

**Técnica experimental:** uso de los breakpoints. Es muy útil para detener el programa en un punto donde conviene inspeccionar los registros o la memoria sin tener que ir paso a paso desde el principio. Simplemente se le indica al simulador la dirección de la instrucción donde ha de detenerse la ejecución. Utilice la técnica anterior para detener la ejecución dentro de Mult y observar el valor de la dirección de retorno contenida en el registre $ra. Deberá indicar como punto de ruptura del flujo de ejecución la dirección de la instrucción jr $ra.

* ¿Cuál es el valor de la dirección de retorno?
* ¿A qué instrucción del programa apunta? Apunta a **break $1**.

**Ejercicio 2: Creación de funciones**

Se debe implementar:



.data 0x10000000

.globl \_\_start

.text 0x00400000

\_\_start:

la $a0,'M'

jal Input

move $a2,$v0

la $a0,'C'

jal Input

move $a1,$v0

move $a0,$a2

jal Mult

move $a1,$v0

jal Prompt

li $v0,10

syscall

Mult: li $v0, 0

beqz $a1, MultRet

MultFor: add $v0, $v0, $a0

addi $a1, $a1, -1

bne $a1, $zero, MultFor

MultRet: jr $ra

Input:

li $v0,11

syscall

li $a0,'='

li $v0,11

syscall

li $v0,5

syscall

jr $ra

Prompt:

li $a0,'R'

li $v0,11

syscall

li $a0,'='

li $v0,11

syscall

move $a0,$a1

li $v0,1

syscall

jr $ra

**Ejercicio 3: Instrucciones condicionales**

.data 0x10000000

.globl \_\_start

.text 0x00400000

\_\_start:

la $a0, 'M'

jal Input

move $a2, $v0

la $a0, 'Q'

jal Input

move $a1, $v0

move $a0, $a2

IF: bltz $a1, THEN

ElSE: jal Mult

THEN: sub $a1, $zero, $a1

sub $a0, $zero, $a0

jal Mult

END:

move $a1, $v0

jal Prompt

li $v0, 10

syscall

Mult: li $v0, 0

beqz $a1, MultRet

MultFor: add $v0, $v0, $a0

addi $a1, $a1, -1

bne $a1, $zero, MultFor

MultRet: jal END

Input:

li $v0, 11

syscall

li $a0, '='

li $v0, 11

syscall

li $v0, 5

syscall

jr $ra

Prompt:

li $a0, 'R'

li $v0, 11

syscall

li $a0, '='

li $v0, 11

syscall

move $a0, $a1

li $v0, 1

syscall

jr $ra