Guía Práctica 1 - ASM Risc V

1. Escribir el siguiente código en el RARS.

```
.text
        lui t0, 0x12345
        lui t1, 201
        lui t2, 0xABCDE
```

La directiva *.text* indica que parte de lo escrito se cargará en el segmento de código (Text Segment). Grabar (ctrl+s), ensamblar (F3) y responder lo siguiente:

a) ¿Qué diferencia se visualiza entre las instrucciones del código en Source y Basic?
b) ¿Cuál es la dirección de comienzo del programa y que longitud tiene cada instrucción?
c) Escribir el código objeto (Code) de cada instrucción en binario
d) ¿Qué valores inicialmente tienen los registros *t0, t1, t2* y *pc*?
e) ¿En qué dirección comienza el segmento de datos?
f) Presionar F7 para ejecutar la primera instrucción. ¿Qué valor toma *t0*?
g) ¿En cuánto y por qué cambia el valor del registro *pc*? ¿Cuál es su función?
h) Seguir ejecutando el programa (F7) y verificar los valores en *t1, t2* y *pc*.

2. Escriba el siguiente código en el RARS.

```
.text
        lui a1, 0x10010
        lw t0, 0(a1)
        lw t1, 4(a1)
        lw t2, 8(a1)
```

Grabar (ctrl+s), ensamblar (F3) y realizar:

a) Cargar manualmente los valores de las siguientes palabras a partir de la dirección *0x10010000* (segmento de datos), *0x12345678*, *1000*, *0x12AB34CD*.
b) ¿Cuál es la función de *lui a1,0x1001*?
c) ¿Qué hace la instrucción *lw t1,4(a1)*? Describir cuáles son sus operandos. Si *a1* fuera *0x10010004*, ¿se obtendría el mismo resultado al hacer *lw t1,0(a1)*? ¿Por qué?
d) Ejecutar paso a paso el programa y responda qué valores carga en *t0, t1* y *t2*.

3. Escribir el siguiente código en el RARS.

```
.data
valor: .word 0

.text
        lui t0, 0x12345
        lui t1, 0x345
        lui t2, 0x5
        sw t0, valor, t6
        la a7, valor
        sw t1, 4(a7)
        sw t2, 8(a7)
```

La directiva *.data* identifica los elementos que estarán en el segmento de datos (solamente a los efectos de esta práctica lo utilizaremos) y la directiva *.word* indica que estamos referenciando una palabra (4 bytes).
Grabar, ensamblar y responder lo siguiente:

a) ¿Qué diferencia se visualiza entre las instrucciones del código en Source y Basic?
b) Pasar el código generado en la instrucción de la línea 4 (en Code), indicando los campos que la conforman.
c) Hacer un seguimiento paso a paso y verificar qué registros intervienen en las operaciones.
d) ¿Qué valor tenía el registro *t6* antes de ejecutar el programa y en cuánto quedó?
e) Verificar qué valor se almacenó en memoria y en qué dirección
f) Editar el programa y cambiar la instrucción Basic por las que surgieron en Source.
Probar de utilizar los nombres de los registros y los números de los registros (por ejemplo si la instrucción es *lui t0, 0x12345*, reemplazarla por *lui x5, 0x12345*).
g) Verificar el funcionamiento del programa con los cambios realizados en el punto f)

4. Escribir el siguiente código en el RARS, luego ejecutar el programa y responder:

```
.text
        lui t0, 0x12345
        sw t0, (sp)
        lui a1, 0x10010
        lw t1, (sp)
        sw t1, (a1)
        li a7, 10
        ecall
```

a) ¿Qué hace el programa?
b) ¿En dónde se almacena el valor de *t0*?
Verifique mostrando en la ventana de ejecución el contenido de la memoria de *sp*. ¿Se modifica el valor de *sp* luego de la línea 3? Si no es así, ¿Que modifica la línea?
c) ¿Por qué se asigna *0x1001* a *a1*? (ver ejercicio 2)
d) ¿Qué acción hace *sw t0,(sp)* y cuál es la dirección en donde se almacenará el dato? ¿Qué valor había antes en esa dirección y qué valor hay después de ejecutar la instrucción?
e) Al ejecutar *lw t1,(sp)* ¿Qué valor toma *t1*?
f) Con *sw t1,(a1)* ¿En qué dirección se almacena el valor de *t1*?

5. Copiar y responder

```
.data
elemento:  .word 0x4E1C532D
.text
        lw t1,elemento
```

a) ¿Qué instrucciones se generan?
b) En las instrucciones generadas, ¿cuáles son los registros intervinientes?
c) ¿Por qué valor es reemplazada la etiqueta elemento?
d) ¿Cuál es la ventaja de utilizar etiquetas en lugar de direcciones?
e) Verificar el valor almacenado en la dirección de memoria apuntada por elemento.

f) Al ejecutar el programa, ¿qué valor queda en *t1*?

6. Copiar y ejecutar paso a paso cada uno de los códigos y responder:

```
.data
valor:  .word 0x805215C9
.text
        la s0,valor
        lbu a1, 0(s0)
        lbu a2, 1(s0)
        lbu a3, 2(s0)
        lbu a4, 3(s0)

.data
valor:  .word 0x805215C9
.text
        la s0,valor
        lb a1, 0(s0)
        lb a2, 1(s0)
        lb a3, 2(s0)
        lb a4, 3(s0)
```

a) ¿Cómo se almacena el valor en el segmento de datos? ¿La arquitectura es big endian o little endian?
b) ¿Qué valores se almacenan en cada caso en *a1, a2, a3* y *a4*? ¿Por qué?

7. Copiar y ejecutar paso a paso cada uno de los códigos y responder:

```
.data
valor:  .word 0x805215C9
.text
        la s0, valor
        lhu a1, 0(s0)
        lhu a2, 2(s0)
---
.data
valor:  .word 0x805215C9
.text
        la s0, valor
        lh a1, 0(s0)
        lh a2, 2(s0)
```

a) Analizar las instrucciones generadas. Identificar los registros intervinientes para poder extraer un valor de la memoria.
b) ¿Qué valores se almacenan en cada caso en a1 y a2? ¿Por qué?

8. Realizar un programa que asigne las palabras 0xABCD0000 en la primera dirección del segmento de datos y 0x12340000 en la siguiente.

9. Considerando el enunciado anterior agregar las líneas necesarias para intercambiar los valores en la memoria