Organización de Computadoras
Guía de Ejercicios Nº 1 - ASM MIPS

1. Escribir el siguiente código en el Mars.

```
.text
    lui $t0,0x1234
    lui $t1,1000
    lui $t2,65535
```

La directiva **.text** indica que parte de lo escrito se cargará en el segmento de código (Text Segment).

Grabar, ensamblar 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 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 primer 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:

```
.text
    lui $at,0x1001
    lw $t0,0($at)
    lw $t1,4($at)
    lw $t2,8($at)
```

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 $at,0x1001`**?
c) ¿Qué hace la instrucción **`lw $t1,4($at)`**? Describir cuáles son sus operandos. Si $at fuera 0x10010004, ¿se obtendría el mismo resultado al hacer **`lw $t1,0($at)`**? ¿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 Mars.

```
.data
valor: .word
.text
    lui $t0,9
    sw $t0,valor
```

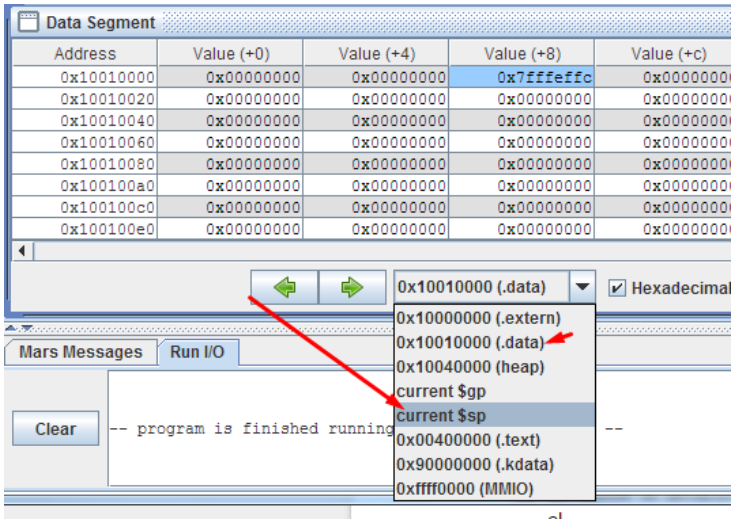
La directiva **.data** identifica los elementos que estarán el el segmento de datos (solamente a los efectos de esta práctica lo utilizaremos) y la directiva **.word** indica que estamos referenciando una palabra (4 bytes).
Grabar, ensamblar y responder lo siguiente:

a) ¿Qué diferencia se visualiza entre las instrucciones del código en Source y Basic?
b) Pasar el código generado en la 3 instrucción (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 $at 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 es la instrucción es **lui $1, 0x1001**, reemplazarla por **lui $at, 0x1001**).
g) Verificar el funcionamiento del programa con los cambios realizados en el punto f)

4. Escribir el siguiente código en el Mars y luego ejecutar el

```
.text
    lui  $t0,0x1234
    sw   $t0,($sp)
    lui  $at,0x1001
    sw $sp,8($at)
    lw $v0,8($at)
    lw $v1,($v0)
```

programa y responder:



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 (figura a la derecha). ¿Se modifica el valor de $sp luego de la segunda línea? Si no es así, ¿Qué modifica la línea?
c) ¿Por qué se asigna 0x1001 a $at? (ver ejercicio 2)
d) ¿Qué acción hace **`sw $sp,8($at)`** 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 $v0,8($at)`** ¿Qué valor toma $v0?
f) Y con **`lw $v1,($v0)`** ¿Qué valor se carga en $v1?
g) ¿Cómo se podría haber logrado el mismo resultado que tuvo $v1 reemplazando las 4 últimas líneas por una sola? Verifíquelo.

5. Copiar y responder

```
.data
elemento: .word 0x54EF128A
.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
dato1: .word 0x801215f9
.text
        lbu $t0,dato1
        lbu $t1,dato1+1
        lbu $t2,dato1+2
        lbu $t3,dato1+3
```

```
.data
dato1: .word 0x801215f9
.text
        lb $t0,dato1
        lb $t1,dato1+1
        lb $t2,dato1+2
        lb $t3,dato1+3
```

a) ¿Cómo se almacena el valor de dato1 en el segmento de datos? ¿La arquitectura es *big endian* o *little endian*?
b) ¿Qué valores se almacenan en cada caso en $t0, $t1, $t2 y $t3? ¿Por qué?

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

```
.data
dato1: .word 0x801215f9
.text
        lhu $t0,dato1
        lhu $t1,dato1+2
```

```
.data
dato1: .word 0x801215f9
.text
        lh $t0,dato1
        lh $t1,dato1+2
```

a) Analizar las instrucciones generadas tanto el ejercicio 5 como en éste. Identificar los registros intervinientes para poder extraer un valor de la memoria.
b) ¿Qué valores se almacenan en cada caso en $t0 y $t1? ¿Por qué?

8. Realizar un programa que asigne las palabras 0xabcd0000 en la primer 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