Guía Práctica 2 - ASM Risc V

1. Escribir la instrucción para asignar el valor 0x512 al registro **t0** y ejecutarla.
    a) Copiar el código generado y pasarlo a binario.
    b) Agrupar los bits de acuerdo al tipo de instrucción e identificar los operandos y elementos.

2. Escribir las instrucciones necesarias para almacenar el valor 0x12345678 en el registro **s1**
    a) Ejecutar paso a paso lo escrito y verificar que funcione correctamente.

3. Copiar el siguiente código:

```
.text
    li a0,0x111117FF
    li a1,0x11111CAB
```

y compararlo con

```
    lui a2,0x11111
    ori a2,a2,0x7FF
    lui a3,0x11112
    ori a3,a3,-0x355
```

y con

```
    lui a4,0x11111
    addi a4,a4,0x7FF
    lui a5,0x11112
    addi a5,a5,-0x355
```

    a) ¿Qué código se genera? ¿A qué conclusión llega? Investigue por qué.
    b) ¿Qué resultado almacenan los registros cuando lo ejecuta paso a paso?
    c) Por qué hay que agregar el valor con el signo **-**. Note que para llegar al valor **a1** tiene que estar incrementado en 1. ¿Siempre?
    d) Qué valor resulta de hacer complemento a 2 del valor 0xCAB?

4. Escribir las instrucciones necesarias para restarle 5 a 20, usando la instrucción **sub** con la instrucción **addi**.
    a) Identifique los códigos generados en ambos casos. ¿Cómo se representan los valores, en particular si hay negativos?

5. Realizar un programa que cargue el valor 0x234 en el registro **t0** y lo almacene en la primera posición del segmento de datos (0x10010000)

6. ¿Tiene sentido esta instrucción **add t0,t0,t0** ? ¿Qué hace?

7. Indique para qué sirve esta instrucción: **add t0, t0, zero**

8. Codificar en ASM las siguientes expresiones aritméticas de C++ (considerar y que ningún valor será 0)

```
    Int a, b, c, d, e;
```

a) a = b;
b) a = b + c;
c) a = a + 1;
d) a = c + 2;
e) a = b + c + d + e;
f) a = b - c;
g) a = c + (b - d);
h) a = (b + c) - (d + e);
i) a = b * c;
j) a = b / c;
k) a = 3 * e;
l) a = (b - c) * (d - e);
m) a = b * c * d;
n) a = (b + c) * (d / e);

9. Copiar el siguiente programa. Antes de ejecutarlo analizar cada instrucción y tratar de calcular el resultado de cada línea. Luego verificar los valores obtenidos ejecutando paso a paso el código

```
.text
    ori t0, zero, 0x465
    ori t1, zero, 0x0FF
    ori t2, zero, 0x123
    and a0, t0, t2
    andi a1, t1, 0x35
    or a2, t1, t0
    ori a3, t1, 1
    addi a4, t1, 1
    xor a5, t0, t0
    xori a6, t0, 0x371
    xori a7, a6, 0x371
    not s1, a0
```

10. Copiar el siguiente programa. Antes de ejecutarlo analizar cada instrucción y tratar de calcular el resultado. Luego verificar los valores obtenidos ejecutando paso a paso el código.

```
.text
    ori t0, zero, 476
    ori t1, zero, 0xFF
    lui t2, 1
    ori t3, zero, 4
    lui t4 0xC3010
    slli, s0,t0, 16
    srai si, t1, 1
    srl s2, t4, t3
    sra s3, t4, t3
    slli s4, t3, 1
    add s4, s4, t3
```

sll s5, t0. s4

11. Escribir un algoritmo para multiplicar por 17 un número que está en **a0** sin utilizar las instrucciones **mul** y otro algoritmo que lo multiplique por 24. Dejar los resultados en **a1**.

12. Poner en 0 los bits 10, 11 y 15 de **t0**

13. Setear en 1 los 4 bits menos significativos de **t1**