Boletín de problemas RISC V

Problema 1.
Suponiendo que el registro x7 contiene la dirección 0x10000000 y que en dicha dirección de memoria se ubica el dato de tamaño palabra 0x1020d040, indique la palabra almacenada en la dirección 0x10000004 tras ejecutar los siguientes pares de instrucciones:

a)
```
lb x6, 0(x7)
sw x6, 4(x7)
```

b)
```
lh x6, 0(x7)
sw x6, 4(x7)
```

c)
```
lhu x6, 0(x7)
sw x6, 4(x7)
```

Problema 2.
Para el trozo de código que se muestra indique el contenido final de t0, t1 y t2.

```
li   t0, 4
slli t1, t0, 3
addi t2, t1, -5
add  t0, t2, t0
```

Problema 3.
Para el trozo de código que se muestra y suponiendo los valores de a0 y memoria que se dan, indique el contenido final de t0 y de la memoria.

a0 = 0x2000
Memoria:
[0x2000] = 5
[0x2004] = 8
[0x2008] = 12

Código:

```
lw   t0, 4(a0)
addi t0, t0, 2
sw   t0, 8(a0)
lw   t0, 8(a0)
```

Problema 4.
Explique qué hace este código

```
lw t0, 0(a0)
lw t1, 4(a0)
sw t0, 4(a0)
sw t1, 0(a0)
```

Problema 5.
Siempre que no de error de ensamblado, indique el contenido del registro x10 después de ejecutar las siguientes parejas de instrucciones e indique una pseudoinstrucción equivalente:

```
a)lui x10,0x12345
  addi x10,x10,0x678
b)lui x10,0x12345
  addi x10,x10,0x876
c)lui x10,1
  addi x10,x10,-0x544
d)lui x10,0xffed8
  addi x10,x10,-0x679
```

Problema 6.
Proponga una secuencia de instrucciones nativas de la ISA RISC-V, equivalentes a las siguientes pseudoinstrucciones:

```
li x10, 0xcaba56ff
li x10, 0x1abc
li x10, 0x56789abc
li x10, 0x1d3
li x10, 0x800
```

Problema 7.
Proponga una secuencia de instrucciones nativas de la ISA RISC-V, equivalentes a las siguientes pseudoinstrucciones:

```
mv t0, t1
li t0, 12
j etiqueta
ret
neg t0, t1
not t0, t1
call funcion
nop (no operación)
```

Problema 8.
Analice el siguiente código y diga la función realizada.

```
    .data
X: .word 7
Y: .word 8
Z: .word 0

.text
la t0, X
lw t1, 0(t0)
lw t2, 4(t0)
add a0, t1, t2
sw a0, 8(t0)
```

Problema 9.
Escriba un programa en lenguaje ensamblador del RISC-V que implemente el siguiente bloque de código.
Use la sección .data para asignar el valor inicial a las variables.

```
int x = 10, y = 5;
if (x >= y) {
        x = x + 2;
        y = y - 2;
        }
```

Problema 10.
Escriba un programa en lenguaje ensamblador del RISC-V que implemente el siguiente bloque de código.
Use la sección .data para asignar el valor inicial a las variables.

```
int x = 5, y = 10;
if (x >= y) {
        x = x + 2;
        y = y + 2;
        }
else {
        x = x - 2;
        y = y - 2;
        }
```

Problema 11.
Escriba un programa en lenguaje ensamblador del RISC-V que implemente el siguiente bloque de código. Use la sección .data para reservar espacio en memoria para las variables.

```
int a, b;
a = 81;
b = 18;
do {
    a = a - b;
    } while (a > 0);
```

Problema 12.
Escriba un programa en lenguaje ensamblador del RISC-V que implemente el siguiente bloque de código. Use la sección .data para reservar espacio en memoria para las variables.

```
int n, fprev, f, i;
n = 5;
fprev = 1;
f = 1;
i = 2;
while (i <= n) {
                faux = f;
                f = f + fprev;
                fprev = faux;
                i = i + 1;
                }
```

Problema 13.
Escriba un programa en lenguaje ensamblador del RISC-V que implemente el siguiente bloque de código.

```
int f = 2, n = 5;
int i;
for (i = 2; i <= n; i++)
        f = f + f;
```

Problema 14.
Una tabla de 100 números enteros está almacenada a partir de la dirección 0x0. Escriba un programa que almacene en x5 el número de datos positivos que hay en dicha tabla, en x6 el número de datos negativos y en x7 el número de elementos iguales a cero.

Problema 15.
Escriba una subrutina que busque el menor de una tabla de números enteros. La subrutina recibirá como argumento de entrada la dirección de comienzo de la tabla y el número de elementos de esta y devolverá el valor del elemento menor de la tabla y el número de orden donde se encuentra.
Compruebe el correcto funcionamiento de la subrutina con una tabla de 16 números enteros definida en tiempo de compilación en la sección de datos.

Problema 16.
Escriba una subrutina que traslade una tabla de 20 datos tamaño word. La subrutina recibirá como argumentos (en este orden) la dirección de comienzo de la tabla y la dirección a la que se la quiere trasladar.
Compruebe el correcto funcionamiento de la subrutina anterior con una tabla de 16 números enteros almacenada en la sección de datos.

Problema 17.
Escriba una subrutina que invierta el orden de datos de una tabla datos tamaño word de longitud arbitraria. La subrutina recibirá como argumentos (en este orden) la dirección de comienzo de la tabla y el número de elementos.
Compruebe el correcto funcionamiento de la subrutina anterior con una tabla de 16 números enteros almacenada en la sección de datos.

Problema 18.
Escriba una subrutina que calcule el valor absoluto de su argumento. Utilice esta subrutina para calcular mediante otra subrutina la suma de los valores absolutos de los elementos de una tabla. Esta segunda subrutina recibirá como argumentos (en este orden) la dirección de comienzo de la tabla y el número de elementos.

Problema 19.
Escriba un programa en ensamblador del RISC-V que llame a una subrutina swap encargada de intercambiar el contenido de dos posiciones de memoria. La subrutina recibirá como parámetros de entrada las posiciones de memoria correspondiente a las variables a y b y deberá preservar el contenido de todos los registros que obligue el estándar de llamadas estudiado en clase.

Problema 20.
Utilizando la subrutina del problema anterior realice otra subrutina llamada swapTable que intercambie dos tablas que se encuentran en memoria. Esta subrutina recibe como argumentos el tamaño de las tablas en a0, y la dirección de comienzo de las dos tablas en a1 y a2.

Problema 21.
Escriba un programa para el ensamblador del RISC-V que cuente el número de 0 de un vector de longitud arbitraria. Emplee para ello una subrutina llamada cuenta_ceros que reciba como parámetros de entrada toda la información necesaria para llevar a cabo la tarea.

Problema 22.
Realice una subrutina llamada prod_escalar que calcule el producto escalar de dos vectores. Para ello, se utilizará otra subrutina (Muls) que debe realizar la multiplicación de números enteros con signo.
Compruebe el correcto funcionamiento de la subrutina prod_escalar con dos vectores de 10 elementos almacenados en la sección de datos.

Problema 23.
Realice la subrutina Muls del programa anterior usando una subrutina llamada mul que multiplica dos números enteros positivos.