Facultad de Informática
Universidad Complutense de Madrid

PROBLEMAS DE FUNDAMENTOS DE COMPUTADORES

TEMA 5

1. Para el siguiente código indicar cuál es el estado del procesador tras la ejecución de cada una de las instrucciones:

```
add x3, x0, 34
and x4, x3, x1
or x5, x1, x2
srl x5, x1, x2
sw x5, 12(x3)
```

Si el estado previo es:

PC=0x00000000

Banco de registros:

Registro	Contenido
x1	0x 0000 10FF
x2	0x FFFF FFFF
X3	0x 1111 1104

2. Escribir los dos siguientes fragmentos de código en ensamblador del RISC-V:

Código A:

```
if (g > h)
    g = g + 1;
else
    h = h – 1;
```

Código B:

```
if (g <= h)
    g = 0;
else
    h = 0;
```

3. Suponiendo que x1 contiene la variable ***i***, x2 contiene la variable ***total*** y x3 contiene la dirección base del array ***vals***, y que el array está inicializado antes de usarlo:

a. Escribe el siguiente código en ensamblador RISC-V.
b. Si el Tiempo de ciclo es de 0,5 ns y cada instrucción tarda un ciclo en ejecutarse, calcula el tiempo de ejecución de los dos códigos del apartado a).

```
int i;
int total;
int vals[200];
total=0
for (i=0; i<200;i=i+1){
    vals[i]=i*7;
     total = total +    vals[i]
    }
```

4. Tenemos que realizar un programa en ensamblador que lea un mensaje contenido en cuatro palabras de memoria a partir de la posición etiquetada por MENS y extraiga cada carácter (byte) de estas palabras y los coloque en una palabra a partir de la dirección SAL. Cada palabra tendrá entonces tres bytes con ceros y el último con el carácter extraído. A continuación se muestra la estructura del programa con la inicialización de variables de entrada y la reserva de espacio para las de salida.

```
        .globl start
        .data
MENS: .word 0x2146656C, 0x697A204E, 0x61766964, 0x61642120
        .bss
SAL:    .space 64
        .text
start:
     Código a realizar

          FIN: B .
          .end
```

- **a.** Realizar el programa
- **b.** Sabiendo que los caracteres están representados en ASCII descifrar el mensaje
- **c.** Suponiendo que se carga el programa en la dirección de memoria 0x00003000 mostrar el mapa de memoria del programa a partir de esta dirección una vez que se ha finalizado la ejecución del programa.

5. Tenemos que realizar un programa en ensamblador RISC-V que lea un vector de 8 enteros positivos (ocupan 32 bits) almacenado en memoria a partir de la posición A y lo rescriba en orden inverso en otro vector que empiece en la posición B (B [0] es A[7], B[1] es A[6] y así sucesivamente). B está inicializado a 0.

La estructura del programa es la siguiente:

```
      .globl start
      .data
A:    .word 7, 3, 25, 4, 75, 2, 1, 1
B:     .word 0, 0, 0, 0, 0, 0, 0, 0
      .text
start:
Código a realizer
      FIN: B .
     .end
```

- **a.** Realizar el programa.
- **b.** Si la dirección de comienzo de la zona de datos (.data) es 0x0000C000, y las zonas de datos y de instrucciones en memoria son consecutivas indicar en qué dirección de memoria se encuentra la primera instrucción del programa y la etiqueta FIN.

6. Tenemos que realizar un programa en ensamblador RISC-V que busque el valor máximo de un vector de enteros positivos (cada uno ocupa 32 bits) **A[i]** de 8 elementos almacenado en memoria y coloque el resultado en la variable **max** también en memoria. La estructura del programa es la siguiente:

```
        .globl start
        .data
A:      .word 7, 3, 25, 4, 75, 2, 1, 1
max:    .word 0
        .text
start:
Código a realizar
        FIN: B .
        .end
```

a. Codificar el programa en ensamblador RISC-V.

b. Si la dirección de comienzo de la zona de datos (.data) es 0x0000C000, y las zonas de datos y de instrucciones en memoria son consecutivas indicar en qué dirección de memoria se encuentra la primera instrucción del programa.

c. Queremos reescribir el programa que calcula el máximo como una función, sabiendo que el parámetro de dirección de comienzo del array se situará en a0, y que el valor que devuelve la función conteniendo el máximo también se situará en a0. Codificar la función y el programa principal que llama a la función y que tiene que traer el parámetro desde memoria a a0 y actualizar el valor devuelto por la función en la variable **max** en memoria.

7. La function *strcpy* copia la cadena de caracteres *src* a la cadena de caracteres *dst*. La cadena de caracteres acaba cuando aparece un cero en ella. Las dos cadenas de caracteres están en memoria:

```
void strcpy(char dst[], char src[]= {
        int i = 0;
        do {
            dst[i]= src[i]
        } while (src[i++]);
}
```

a. Usando las instrucciones **lbu** y **sb** implementar la function strcpy en ensamblador RISC-V. Usa x4 para contener la variable i. x2 contiene la dirección base del array *dst*. x1 contiene la dirección base del array *src*.

b. Muestra el estado de la pila antes, durante y después de la llamada a la función *strcpy*, suponiendo que sp=0xBEFF F000.

8. Tenemos que realizar un programa en ensamblador RISC-V que haga lo siguiente:

A partir de la posición etiquetada como DATOS se encuentra un array de números con valores entre el 0 y el 10, cuyo último número es 0xFF. El programa recorrerá este array y almacenará en otro array, que comenzará en la posición etiquetada como RES, el resultado de sumar cada elemento del array con el siguiente. Finalizará el almacenamiento colocando el valor 0xFF al final del array.

A continuación se muestra la estructura del programa con la inicialización de variables de entrada y la reserva de espacio para las de salida.

```
        .globl start
        .data
DATOS: .word 0x2, 0x3, 0x4, 0x4, 0x1, 0x1, 0x4, 0x7, 0x2, 0x1, 0x5, 0xFF
        .bss
```

```
RES:   .space 44
       .text
start:
    Código a realizar
        FIN: B .
        .end
```

a) Escribir el código del programa.
b) Suponiendo que se carga el programa en la dirección de memoria 0x00003000 indicar dónde se encuentra la primera instrucción del programa, y cuál es la dirección del array RES.

NOTA: Para los datos del ejemplo los valores que se almacenarán a partir de RES serán:
0x5 0x7 0x8 0x5 0x2 0x5 0xB 0x9 0x3 0x6 0xFF

9. Tenemos que realizar un programa en ensamblador que lea un array en el que están contenidas las notas de prácticas, primer parcial y examen final de 5 alumnos. Las notas están almacenadas a partir de la posición etiquetada por NOTAS, cada nota ocupa una palabra de 4 bytes. Entre las tres notas de un alumno y las del siguiente se encuentra una palabra de 4 bytes que contiene 0xf.

Se desea guardar en otro array almacenado a partir de la posición etiquetada por FINALES las notas totales (resultado de sumar las tres notas) de los cinco alumnos.

A continuación se muestra la estructura del programa con la inicialización de variables de entrada y la reserva de espacio para las de salida (no se ha dado un valor concreto para esta reserva de espacio).

```
        .globl start
        .data
NOTAS: .word 0x2, 0x3, 0x4, 0xf, 0x1, 0x1, 0x4, 0xf, 0x2, 0x1, 0x5, 0xf, 0x1, 0x0, 0x4, 0xf,
0x0, 0x3, 0x0, 0xf
        .bss
FINALES:    .space XXXXX
        .text
start:
    Código a realizar

        FIN: B .
        .end
```

a. ¿Cuál es el valor qué hay que colocar junto a la directiva .space sustituyendo al valor XXXXX?
b. Escribir el código del programa.
c. Reescribir todo el programa para utilizar una rutina que sume tres valores que se pasan en los registros a0, a1, y a2 y devuelve el resultado de la suma en el registro a0. Indicar si es necesario preservar el valor de algún registro en la pila y por qué.