Práctico 4 : Programación RISC-V

Ejercicios de entrada/salida

1. Escribir un programa que lea dos números enteros desde el teclado, los sume y muestre el resultado por pantalla.
2. Escribir un programa que lea un número entero desde el teclado y calcule su tabla de multiplicar del 1 al 10, mostrando cada resultado.

Ejercicios de Manipulación de arreglos

3. Escribir un programa que recorra un arreglo de 10 enteros (llamado TABLA) y cuente cuántos son mayores que un valor X. Guardar el resultado en una dirección de memoria etiquetada como CANT. Además, generar otro arreglo llamado RES, donde cada elemento sea 1 si el valor correspondiente en TABLA es mayor que X, o 0 si es menor o igual.
4. Escribir un programa que busque el valor mínimo en un arreglo de N enteros de 32 bits y lo almacene en la dirección de memoria MINIMO.
5. A partir del ejercicio anterior, implementar un algoritmo que ordene el arreglo de menor a mayor. El arreglo ordenado debe almacenarse en uno nuevo llamado RES.
6. Escribir un programa que, dado un arreglo de 10 bytes, genere dos nuevos arreglos: uno con los números pares (PAR) y otro con los números impares (IMPAR).
7. Escribir un programa que sume todos los valores de un arreglo de 10 elementos de tipo half-word, que tengan un 1 en el bit 3.
8. Escribir un programa que, dados dos arreglos A y B de 15 enteros, reemplace en A, aquellos elementos que tengan un 1 en los bits 3 y 12, por los valores correspondientes del arreglo B.
9. Escribir un programa que, dada una matriz de 3x3 enteros (word), calcule el valor máximo de cada fila y lo almacene en un arreglo llamado MAX.

Ejercicios con subrutinas

10. Implementar una subrutina que, dada una matriz de enteros de dimensiones n×m almacenada por filas, devuelva el valor del elemento ubicado en la posición (i, j).
11. Escribir un programa que calcule el factorial de un número:
    a. Directamente desde el programa principal, sin usar subrutinas.
    b. Utilizando una subrutina llamada FACT.
12. Escribir un programa que multiplique dos números NUM1 y NUM2 y almacene el resultado en RES:
    a. Desde el programa principal, sin subrutinas.
    b. Llamando a una subrutina MUL, pasando los parámetros por valor (usando registros) y devolviendo el resultado por valor (también en un registro).

c. Llamando a una subrutina MUL, pasando los parámetros por referencia (usando registros) y devolviendo el resultado por valor (en un registro).

13. Escribir una subrutina DIV que calcule la división de dos números positivos. Los parámetros deben pasarse por valor a través de la pila y el resultado debe devolverse también por la pila, por valor.
14. Escribir una subrutina RESTO que calcule el resto de dividir dos números positivos. Los parámetros deben pasarse por valor usando registros y el resultado debe devolverse por referencia usando un registro.
15. Escribir un programa que sume dos números de 32 bits almacenados en memoria:
    a. Resolviendo todo en el programa principal, sin subrutinas.
    b. Llamando a una subrutina SUM32, que reciba los parámetros por valor a través de la pila y devuelva el resultado también por referencia usando la pila.

Comparación con lenguaje C

16. Dado el siguiente programa en C, usar el conversor https://godbolt.org/ para pasarlo a RISC-V. Comparar ambos códigos.
    - ¿Cuál es la principal diferencia que encuentra?
    - ¿Cómo es la equivalencia entre las instrucciones?
    - ¿Cómo es el orden de las instrucciones?

```c
#include <stdio.h>

int main () {
        char l, palabra[21];
        int i;

        printf("Teclee una palabra de menos de 20 letras:");
        scanf("%s", palabra);
        i = 0;
        while(palabra[i++] != "\0") ;
                l = i-1;
                printf("%s tiene %d letras\n", palabra, l);
                printf("%s escrita al revés es: ", palabra);
                i = l;

        while (i > 0)
                printf("%c", palabra[--i]);
        return 0;
}
```

Ejercicios con Estructuras y punteros

17. Definir una estructura Persona con los campos: nombre (cadena de caracteres), edad (entero) y altura (entero).
    a. Implementar una función crear_persona que reciba:
       - Dirección donde se almacenará la estructura

- Dirección de la cadena con el nombre
- Edad
- Altura

Esta función debe cargar los datos recibidos en la estructura Persona.

b. Implementar una función imprimir_persona que reciba la dirección de una estructura Persona e imprima sus campos: nombre, edad y altura.
c. Escribir un programa principal (main) que utilice ambas funciones para crear una persona, asignar datos y mostrarlos en pantalla.

Pista: Usar las instrucciones lw y sw para acceder a los datos en memoria. Como RISC-V no tiene instrucciones específicas para cadenas, deberás manipular los bytes directamente.

Ejercicios con recursividad

18. Escribir un programa principal que calcule la suma de los primeros N números naturales utilizando una función recursiva. La función debe llamarse suma_n(n) y recibir un entero n como parámetro. Debe devolver la suma de 1 + 2 + ... + n.

    Ejemplo: Si el usuario ingresa n = 5, la función debe devolver 15 (porque 1+2+3+4+5 = 15).

19. Escribir un programa principal que lea una cadena ingresada por el usuario y calcule su longitud. Debe llamar a una función len(pcad) que reciba un puntero a la cadena y devuelva su longitud. Implementar esta función de forma recursiva.