PRÁCTICA 6
LENGUAJE MÁQUINA PROCESADOR RISC -V. Subrutinas II

Objetivos:

- Familiarizarse con la arquitectura del RISC-V
- Definir cadenas de caracteres en memoria
- Realizar llamadas anidadas de subrutinas
- Uso de la pila para guardar el retorno en llamadas anidadas

Desarrollo:

Parte 1) Realizar en ensamblador RISC-V una subrutina que calcule el número de caracteres de una cadena de caracteres acabada con el carácter nulo. Esta subrutina la denominaremos **LongitudCadena** y tendrá como argumento de entrada en **a0** la dirección de comienzo de la cadena de caracteres. A su vez devolverá como parámetro de salida en **a0** el número de caracteres de la cadena.

Parte 2) Realizar en ensamblador RISC-V una subrutina que invierta los caracteres de una cadena de caracteres. La llamaremos **InvertirCadena**. Esta función tendrá como argumento de entrada en **a0** la dirección de la cadena de caracteres a invertir y deberá llamar a LongitudCadena para obtener el número de caracteres. Además deberá imprimir por consola la cadena de caracteres invertida.

Parte 3) Completar el siguiente programa principal para que usando la subrutina anterior invierta los caracteres de las dos cadenas dadas.

```
      .data
cadena1: .string "123 45"   #acaba la cadena con el carácter nulo (0x00)
cadena2: .string "adios"

      .text
main:
la a0, cadena1
      [...]

      .data
cadena1: .string “123 45”
cadena2: .string “adios”
      .text
main:
      la a0, cadena1              #Carga dir. mem cadena1 en a0
      jal invertir_cadena

      la a0, cadena2              #Carga dir.  mem cadena2 en a0
      jal invertir_cadena

      li a7, 10
      ecall
```

```
invertir_cadena:
        add t0, zero, a0          #Cargamos el valor de a0 cade1
        add t1, zero, a0          #Cargamos el valor de a0 cade2
        addi sp, sp, -4           #Sacamos de la pila para guardar
        sw ra,0(sp)               #Almacenamos retorno para no perderlo
        jal longitud_cadena       #Salto a subrutinas
        lw ra, 0(sp)              #Recuperamos retorno
        addi sp, sp, 4            #Recuperamos el estado de la pila

        add t2, a0, t0            #Cargo la dir de memoria final de la cadena + 1
        addi t2, t2, -1           #Lo elimino
bu:     bge t0, t2, imprimirca    #Bucle que salta cuando dir. inicial >= dir. final
                                  #Útil para darle la vuelta a los strings
```

lb t3, 0(t0) lb t4, 0(t2)

sb t4, 0(t0) sb t3, 0(t2)

```
        #MUY IMPORTANTE
        lb t3, 0(t0)              #Al principio de cadena t0
        lb t4, 0(t2)              #Al final de cadena t2
        sb t4, 0(t0)
        sb t3, 0(t2)
```

Supongamos "123 45". En t0 tenemos un puntero que apunta al inicio del String, sease "1", y en t2 tenemos un puntero que apunta al final del String, sease "5". Usamos los loads para coger el 1 en t3 y el 5 en t4. Con los stores tomamos el valor de t4 que es el que está almacenando al final y lo guardamos en t0, la cadena de strings en orden, luego tomamos t3 y lo dejamos al final. Después de la primera ejecución quedaría como "523 41".

Luego pasamos al siguiente, tomamos el 2 y el 4 y les damos la vuelta de la misma forma, quedando "543 21"

Por último nos queda hacer el cambio del 3 y el espacio, siendo como final "54 321"

Sale de la condición bge t0, t2, imprimirca ya que dir. inicial >= dir. final

#MUY IMPORTANTE

```
        addi t0,t0,1              #Pasamos al siguiente del inicio
        addi t2,t2,-1             #Pasamos al anterior del final
        j bu                      #Salto al bucle

imprimirca:
        addi a0, t1, 0
        li a7, 4
        ecall                     #Impresión después del cambio

        ret
```

```
longitud_cadena:
        add t2, zero, zero          #Metemos 0 en t2 para inicializar
longitud_loop:
        lb t3, 0(a0)                #Cargamos carácter encontrado en a0
        addi a0, a0, 1              #Avanza al siguiente el puntero
        beq t3, zero, end           #Si el carácter que encuentra es nulo (acabado) salta al final
        addi, t2, t2, 1             #Aumenta en 1 la longitud de la cadena
        j longitude_loop            #Vuelta al inicio de la subrutina para repetir bucle
end:    add a0, t2, zero            #Metemos el valor en a0 para devolverlo

end_longitud:
        jr ra
```