

ARQUITECTURA DE COMPUTADORES

EXAMEN DE PRÁCTICAS

- **Grados:** IST, ITT, IT, IT-ADE, IT-AEROESPACIAL, URJC
- **Fecha:** 29-Junio-2022
- **Convocatoria Extraordinaria** (PRESENCIAL)

AVISO

Asegúrate que tus programas cumplen con los siguientes criterios. Si no se cumple alguno de ellos **la nota será 0** en ese apartado

- **Cumplimiento de especificaciones.** Se deben cumplir las especificaciones indicadas en el enunciado: nombres de funciones, nombres de archivos, parámetros de las funciones, constantes, funcionalidad, etc. Compruébalo antes de entregar el examen
- **Respetar el convenio.** Resuelve las preguntas sin violar el convenio del uso de registros (ABI del RISC-V)
- **Sin errores en tiempo de ejecución (Runtime errors).** Tus programas no deben generar excepciones al ejecutarse
- **Sin errores al ensamblar.** Los ficheros entregados NO deben dar errores al ensamblarlos. Si una función la has dejado a medio hacer, asegúrate que al menos se ensambla sin errores

Programa 1 (5 ptos)

El siguiente programa calcula el término de fibonacci indicado en la constante N, y lo imprime en la consola (Está en el fichero `fibonacci.s`)

```
#-----  
#-- Sucesion de Fibonacci  
#-- Calcular el término n  
#-----  
#-- fib(0) = 0  
#-- fib(1) = 1  
#-- fib(2) = 1  
#-- fib(3) = 2  
#-- fib(4) = 3  
#-- fib(5) = 5  
#-- fib(6) = 8  
#-- fib(n) = fib(n-1) + fib(n-2)  
  
#-- Servicios del sistema operativo  
    .eqv PRINT_INT      1  
    .eqv PRINT_STRING  4  
    .eqv EXIT           10  
  
#-- Termina de fibonacci a Calcular: fib(N)  
#-- (Debe ser > 1)
```

```
.eqv N 6

.data
msg1: .string "\nFibonacci de "
msg2: .string " ---> "

.text

#-- Meter en t6 el termino a calcular
li t6, N

#-- El registro t5 se usa para indicar el numero del termino
#-- de fibonacci actual (n)
li t5, 1 #-- Termino final a calcular

#-- Los registros t0 y t1 contienen fib(n) y fib(n+1)
li t0, 0
li t1, 1

bucle:
#-- Comprobar si hemos calculado el termino pedido
beq t5,t6, fin

#-- Calcular el siguiente termino:  $t2 = t1 + t0$ 
add t2, t1, t0

#-- Actualizar los valores recordados
mv t0, t1 #--  $t0 = t1$  (termino anterior)
mv t1, t2 #--  $t1 = t2$  (nuevo termino)

#-- Incrementar el termino actual
addi t5, t5, 1

#-- Repetir bucle
b bucle

#-- Mostrar el termino calculado
#-- El termino esta almacenado en t2
fin:
#-- Imprimir mensaje en consola
la a0, msg1
li a7, PRINT_STRING
ecall

#-- Imprimir n
mv a0, t6
li a7, PRINT_INT
ecall

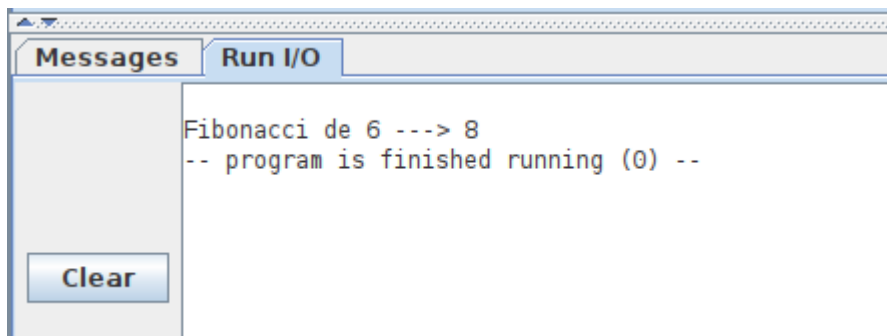
#-- Mensaje 2
la a0, msg2
li a7, PRINT_STRING
```

```
ecall

#-- Imprimir el numero de fibonacci
mv a0, t2
li a7, PRINT_INT
ecall

#-- Terminar
li a7, EXIT
ecall
```

Tras su ejecución esta es la salida que obtenemos en la consola:



Se ha calculado el valor del término 6 de fibonacci (N=6). Cambiando la constante, se imprime su término correspondiente

Queremos separar este programa en **dos ficheros**, uno el **programa principal** y otro con la **función fib(n)** que calcula el término n de fibonacci. La función fib() tiene un argumento de entrada, n, que es el término a calcular, y devuelve el valor de ese término

Se pide:

a) (2.5) Implementa la función de fibonacci en el fichero **01_fib.s**

b) (2.5) Implementa el programa principal en el fichero **01_main.s**. Para ello, modifica el programa original para que llame a la función de fibonacci para calcular el término N. La salida en consola de este programa debe ser exactamente la misma que el programa original

NOTA: Este programa sólo consta de esos dos ficheros indicados. Debe funcionar correctamente sólo con esos dos ficheros. No puedes añadir ninguno más

Programa 2 (5 Ptos)

La función *linea(car)* imprime en la consola una línea de **5 caracteres**. Tiene un parámetro de entrada, *car*, que indica el carácter a usar para dibujar la línea. No tienen ningún parámetro de salida

La implementación de esta función se encuentra en el fichero **02_linea.s**, cuyo contenido es:

```
#-----
#-- Funcion linea(car)
#--
#-- Dibujar una linea de 5 caracteres
```

```

#-- ENTRADAS:
#--   a0: caracter a usar para dibujar la linea
#--
#-- SALIDAS: Ninguna
#-----

        .globl linea

#-- Servicios del sistema operativo
.equv PRINT_CHAR    11

#-- Numero de caracteres en la linea
.equv N 5

        .text

#-- Punto de entrada de la funcion
linea:

#-- Contador de caracteres en la linea
li t0, 0

#-- Comprobar si la linea se ha completado
bucle:
    li t1, N
    beq t0,t1,fin

#-- Imprimir el caracter
li a7, PRINT_CHAR
ecall

#-- Incrementar el contador de caracteres
addi t0,t0,1

    b bucle

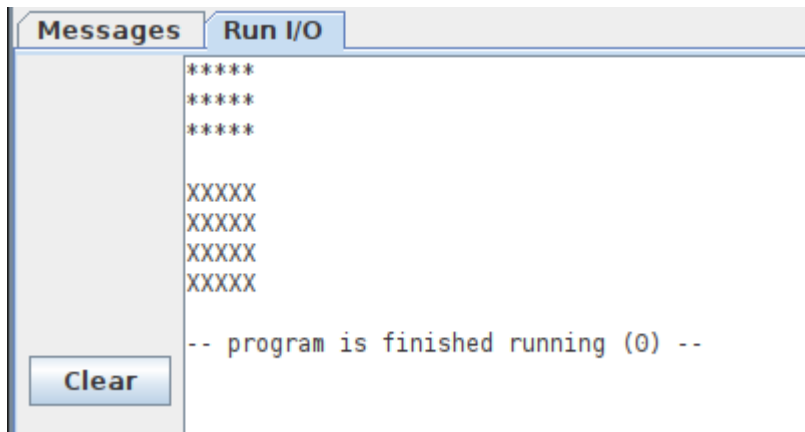
fin:
#-- Imprimir un salto de linea
li a0, '\n'
li a7, PRINT_CHAR
ecall

#-- Retornar
ret

```

La función *bloque(car, lineas)* imprime varias líneas de caracteres, una debajo de la otra. Tiene dos parámetros de entrada: el carácter a usar para dibujar las líneas, y el número de líneas a dibujar. No devuelve ningún valor. La función *bloque()* llama a la función *linea()* para dibujar cada una de las líneas

Además se quiere hacer un programa principal cuya salida en consola sea la siguiente:

A screenshot of a terminal window with two tabs: 'Messages' and 'Run I/O'. The 'Run I/O' tab is active, displaying the output of a program. The output consists of three lines of asterisks, followed by four lines of 'X's, and a final line indicating the program has finished. A 'Clear' button is visible in the bottom left corner of the terminal window.

```
*****
*****
*****

XXXXX
XXXXX
XXXXX
XXXXX

-- program is finished running (0) --
```

El programa imprime en la consola dos bloques, llamando a la función `bloque()`. El primero está formado por **3 líneas** con asteriscos, y el segundo por **4 líneas** con el carácter 'X'. Ambos bloques están separados por un salto de línea

Se pide:

- a) (2.5 ptos) Implementar la función *bloque* en el fichero `02_bloque.s`
- b) (2.5 ptos) Implementar el **programa principal** en el fichero `02_main.s`

NOTA: Este programa está formado únicamente por tres ficheros: `02_linea.s`, `02_bloque.s` y `02_main.s`. Debe funcionar sin añadir ninguno más