

EXAMEN DE PRÁCTICAS
Convocatoria extraordinaria
ARQUITECTURA DE COMPUTADORES

1º Robótica Software, URJC

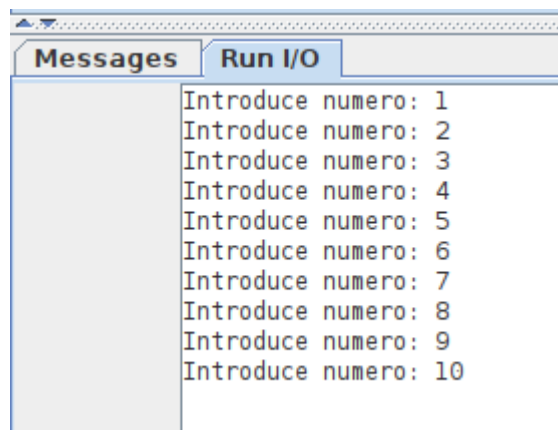
Fuenlabrada (Examen remoto). 13 de Julio de 2020

AVISO: Asegúrate que tus programas **cumplen** con los siguientes criterios. Si no se cumple alguno de ellos la **nota máxima** de tu examen será de **2 puntos**

- **Cumplimiento de especificaciones.** Se deben cumplir las especificaciones indicadas en el enunciado: nombres de funciones, nombres de archivos, 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

Se quiere implementar en ensamblador del RISC-V las siguientes **funciones**, cuyas **especificaciones** son las siguientes:

- *int sum10(pint):* Calcula la **suma** de un conjunto de **10 números** situados en la memoria de forma consecutiva. Tiene un **único parámetro de entrada**, pint, con la dirección del primer número del conjunto. La función recorre la memoria a partir de esa dirección, sumando los 10 números y devuelve su **suma** como salida (un parámetro de salida)
- *int usersuma10(pint):* Pide al usuario **10 números** y los **almacena** consecutivamente en memoria, a partir de la dirección indicada por su **parámetro de entrada** pint. A continuación calcula su suma llamando a la función sum10(). Se devuelve como parámetro de salida esta suma. Este es un ejemplo de lo que ocurre en la consola al llamar a esta función. El usuario ha introducido los números 1,2,3,4,5,6,7,8,9 y 10

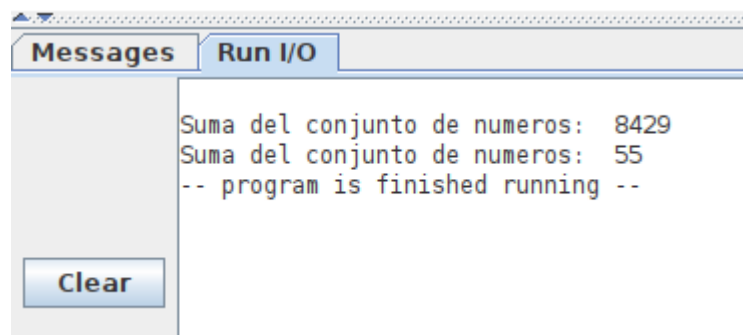


```
Messages Run I/O
Introduce numero: 1
Introduce numero: 2
Introduce numero: 3
Introduce numero: 4
Introduce numero: 5
Introduce numero: 6
Introduce numero: 7
Introduce numero: 8
Introduce numero: 9
Introduce numero: 10
```

Para comprobar que las funciones anteriores **funcionan correctamente**, se definen los siguientes **programas principales**:

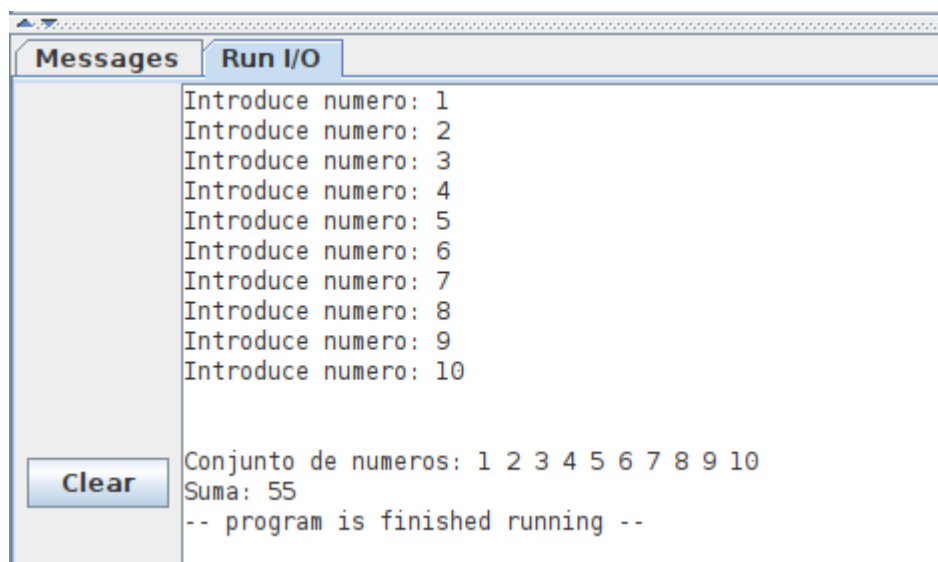
- *test-sum10.s*: Programa para comprobar la función `sum10()`. Realizará la suma de los siguientes conjuntos de números, situados en la memoria y definidos en **tiempo de compilación**:
 - 100, 50, 288, 99, 3000, 1528, 927, 318, 99, 2020
 - 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Esta función deberá calcular la suma de cada conjunto, llamando a la función `sum10()` e imprimiendo en la consola el resultado. Se deberá implementar con un **bucle**. Esto permitirá ampliarlo fácilmente en el futuro para calcular la suma de más conjuntos, y no sólo dos. Al ejecutarlo, la salida en la consola deberá ser:



```
Messages Run I/O
Suma del conjunto de numeros: 8429
Suma del conjunto de numeros: 55
-- program is finished running --
Clear
```

- *test-usersuma10.s*. Programa para comprobar la función `usersuma10()`. Primero llama a la **función `usersuma10()`** para pedir al usuario los números y que queden **almacenados en memoria**. Después recorrerá este conjunto de números **imprimiéndolos** en la consola, separados por un espacio, para comprobar que la función `usersuma10()` los había almacenado correctamente. Por último se imprime su suma. Este sería el resultado que aparece en consola al **ejecutarlo**, cuando el usuario ha introducido los números 1,2,3,4,5,6,7,8,9 y 10.



```
Messages Run I/O
Introduce numero: 1
Introduce numero: 2
Introduce numero: 3
Introduce numero: 4
Introduce numero: 5
Introduce numero: 6
Introduce numero: 7
Introduce numero: 8
Introduce numero: 9
Introduce numero: 10
Conjunto de numeros: 1 2 3 4 5 6 7 8 9 10
Suma: 55
-- program is finished running --
Clear
```

Una parte de esa información se ha impreso en la función `usersuma()` y la otra en el programa de prueba

Se pide:

1. Implementar la función `sum10()` en el fichero **sum10.s** (2.5 puntos)
2. Implementar el programa de prueba **test_sum10.s** (2.5 puntos)
3. Implementar la función `usersuma10()` en el fichero **usersuma10.s** (2.5 puntos)
4. Implementar el programa de prueba **test-usersuma10.s** (2.5 puntos)

NOTA:

Sólo hay que entregar los 4 ficheros pedidos. Cualquier constante necesaria se debe definir dentro de cada fichero (y no cargándola de ficheros externos adicionales)

NOTA 2:

Los programas pedidos son cortos y fáciles de implementar, siempre y cuando tengas muy claro qué es lo que se te pide. Dedica tiempo a pensar y a organizar las ideas. Luego el código fluirá muy rápidamente

NOTA 3:

¡Y recuerda! Tienes que implementar lo pedido **sin violar** el convenio ABI del RISC-V. Violarlo es lo mismo que si no te funcionase, aunque el resultado de la consola fuese correcto