

EXAMEN DE PRÁCTICAS
Convocatoria extraordinaria
ARQUITECTURA DE COMPUTADORES

3o, IST, ITT, IT, 4o IT-ADE y IT-AEROESPACIAL, URJC

2 de Julio de 2021

AVISO: Asegúrate que tus programas **cumplen** con los siguientes criterios. Si no se cumple alguno de ellos la **nota será de 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, funcionalidad pedida, 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

Como parte de una aplicación de dibujo 2D, necesitamos trabajar en el RISC-V con **vectores 2D**, formados por dos componentes (x,y). Cada componente es una palabra (32-bits), de tipo entero. Por tanto necesitamos dos palabras para definir cada vector

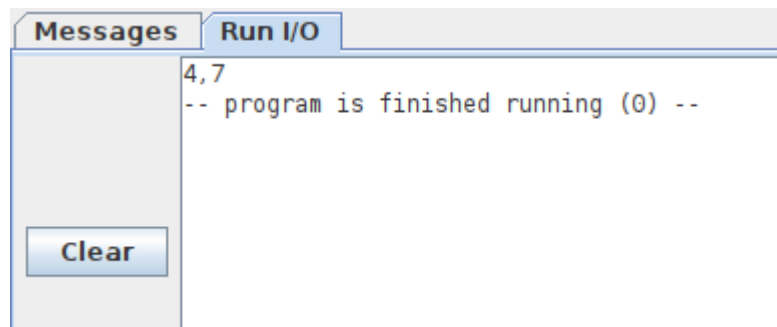
PARTE I

Queremos realizar la **suma de dos vectores**. Para ello debemos implementar la función **sumav()** siguiendo las especificaciones dadas por nuestro ingeniero jefe. Además, tendremos que implementar el **programa principal sumav_test.s para probar** esta función, siguiendo también las especificaciones indicadas

Las **ESPECIFICACIONES** son las siguientes:

- *(int wx, int wy) sumav(int ux, int uy, int vx, int vy):* Función que realiza la **suma de dos vectores** y devuelve como resultado las dos componentes del vector suma. Tiene **4 parámetros de entrada**: Las componentes x e y del primer vector (ux, uy), y las componentes x e y del segundo vector (vx,vy). Devuelve **2 parámetros de salida**: las dos componentes del vector suma (wx, wy). El vector suma se obtiene sumando de manera independiente cada una de sus componentes: (ux+vx, uy + vy). Así, la suma de los vectores (1,2) y (10,20) da como resultado (11,22)
- *sumav_test.s:* Programa para comprobar que la función de suma de vectores está funcionando. Para ello se realizará la suma de los dos vectores constantes (1,2) y (3,5), cuyo resultado es el vector (4,7). Ambos vectores deberán estar situados en el segmento de datos, uno a continuación del otro. En total ocupan 4 palabras consecutivas. En la primera posición se encuentra la componente x

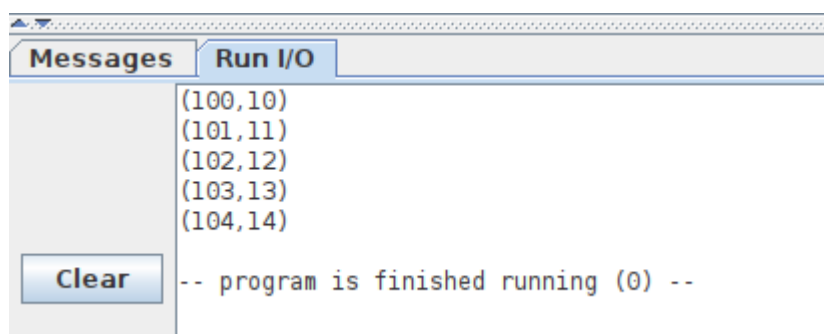
del primer vector (que vale 1). En la segunda posición la componente y del primer vector (que vale 2). En la tercera posición la componente x del segundo vector (que vale 3) y en la cuarta posición la componente y del segundo vector (que vale 5). El programa principal deberá leer los vectores del **segmento de datos** y **llamar a la función sumav()** con los parámetros adecuados para realizar la suma de los vectores. El **resultado de la suma se imprimirá en la consola** (no hay que almacenarlo en ningún lugar de la memoria). Primero se imprime la componente x, después el carácter ',' (coma) y por último la componente y de la suma. Al ejecutarlo, la salida será la mostrada en este pantallazo:



```
Messages Run I/O
4,7
-- program is finished running (0) --
Clear
```

Aunque el programa se probará con los vectores constantes mencionados, si en la memoria se cambian los valores por otros vectores, la suma se debe realizar también correctamente

PARTE II. Como parte de otro proyecto independiente, también relacionado con vectores 2D, **otro ingeniero** ha implementado la función `printv()` en el fichero **printv.s**, siguiendo las especificaciones indicadas por el jefe de proyecto. Esta función imprime en la consola el vector pasado como argumento. La función es correcta y funciona. Para probar esta función el mismo ingeniero ha creado el programa principal **printv_test.s**, que imprime en la consola cinco vectores de prueba. La salida de este programa principal es la siguiente:



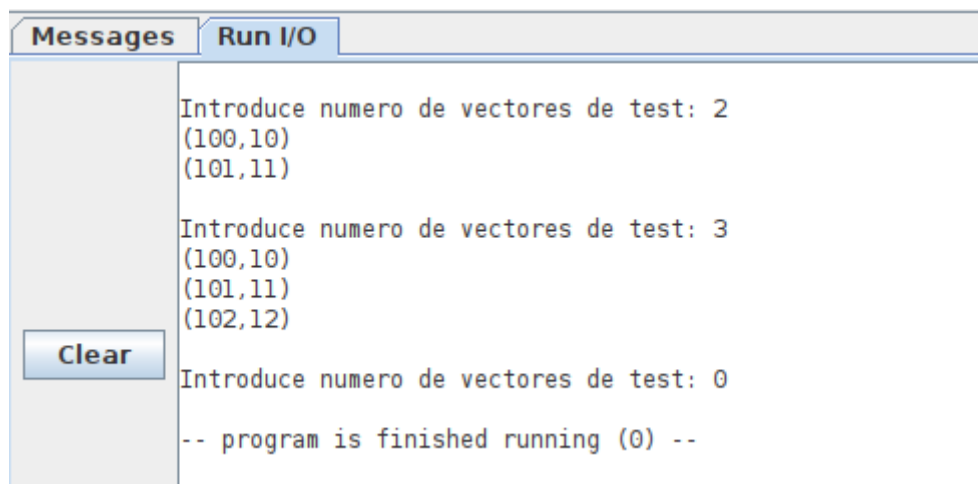
```
Messages Run I/O
(100,10)
(101,11)
(102,12)
(103,13)
(104,14)
-- program is finished running (0) --
Clear
```

Tanto el programa principal como la función **YA están implementadas**, y funcionan correctamente. El jefe de proyecto nos pide que **convirtamos** este programa principal en la **función printv_test()**, que hace exactamente lo mismo, pero al ser una función la podemos llamar desde otros programas. También nos pide que desarrollemos el **programa principal main.s**, que llama a la función `printv_test()` para comprobar que funciona bien. Las **ESPECIFICACIONES** son las siguientes:

- `void printv_test(n)`. Esta función tiene un parámetro de entrada que indica el

número de vectores de test a imprimir en la consola. No devuelve ningún valor como salida. Debemos modificar el programa principal para convertirlo en una función, pero NO PODEMOS MODIFICAR el código indicado en los comentarios. Es decir, que hay que convertirlo en una función modificando SOLO el código inicial y final, PERO NO el que está indicado en el medio

- Programa *main.s*. Programa de prueba de la función `printv_test(n)`. Debe pedir al usuario que introduzca el número de vectores de test a imprimir. Este número se utiliza como parámetro para llamar a la función `printv_test(n)`. Este proceso se repite en un bucle hasta que el usuario introduzca el valor 0, en cuyo caso el programa termina (sin imprimir nada adicional). En esta figura se muestra un pantallazo de su funcionamiento. El usuario ha introducido primero el valor 2, luego el 3 y finalmente el 0. La salida en estas condiciones es la siguiente:



```
Messages Run I/O
Introduce numero de vectores de test: 2
(100,10)
(101,11)
Introduce numero de vectores de test: 3
(100,10)
(101,11)
(102,12)
Introduce numero de vectores de test: 0
-- program is finished running (0) --
```

Se pide:

PARTE I: (5 puntos)

1. Implementar la función **sumav()** en el fichero **sumav.s**. (2.5 puntos)
2. Implementar el programa de prueba **sumav_test.s**. Define todas las constantes que necesites dentro de ese fichero (No uses ningún fichero externo) (2.5 puntos)

PARTE II (5 puntos):

3. Implementar la función **printv_test()** en el fichero **printv_test.s**. En este fichero ya se encuentra el programa original. Deberás modificarlo para convertirlo en una función. Lo debes hacer SIN MODIFICAR el código indicado. Define todas las constantes que necesites dentro de este fichero (No uses ningún fichero externo). (2.5 puntos)
4. Implementar el programa principal **main.s** (Define todas las constantes que necesites en el propio fichero, sin usar ningún fichero externo) (2.5 puntos)

NOTA:

Todos tus programas deben funcionar sólo con los ficheros proporcionados en las plantillas (no puedes añadir ninguno adicional). Los **ficheros a entregar** son: **sumav.s**, **sumav_test.s**, **printv_test.s** y **main.s** (Envíalos por aula virtual, sin comprimir en un ZIP)