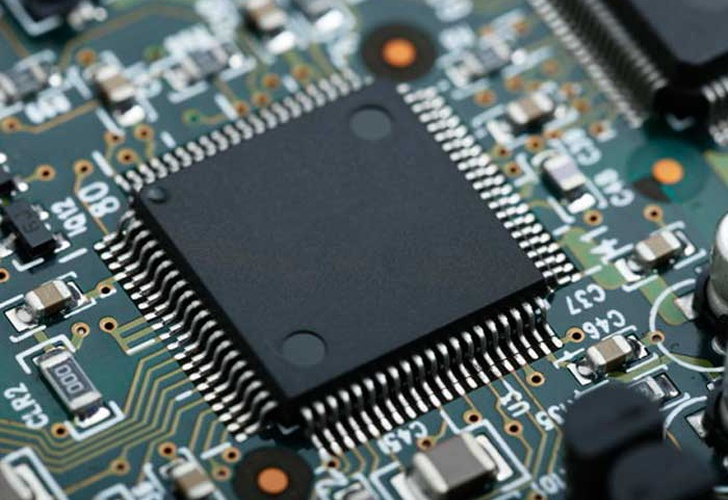
Estructura de los computadores

Luis Girona Pérez

08/04/2020

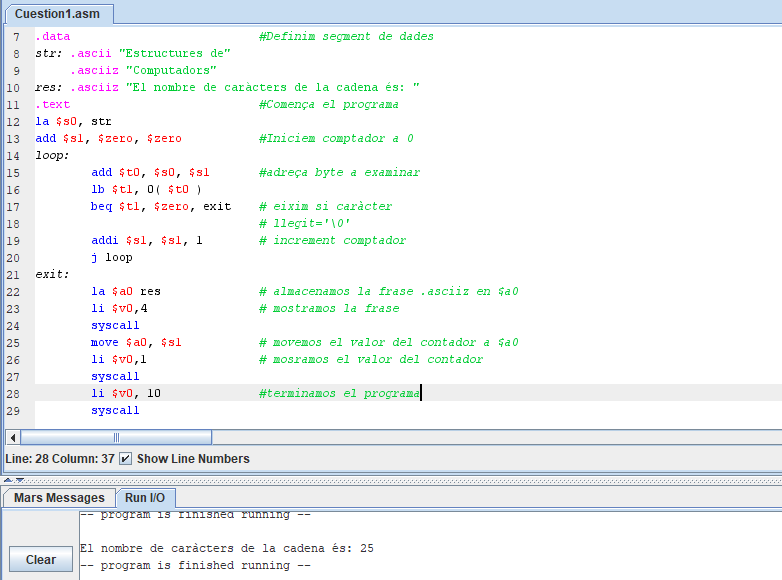
**MEMORIA DE PRÁCTICAS 7-8-9**



**PRÁCTICA 7**

**Cuestión 1**

* **Modifica el código de la actividad 1 para que muestre por pantalla el mensaje “El número de caracteres de la cadena es: “y a continuación el resultado.**

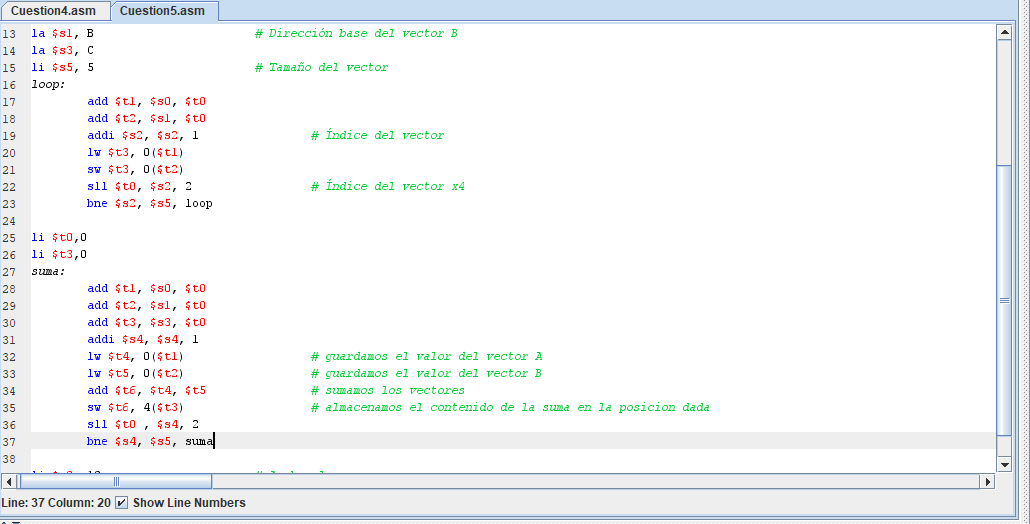


Lo que hemos hecho aquí ha sido añadir un .asciiz llamado res con la frase que queremos mostrar. En la función exit he añadido un par de líneas que son las siguientes:

* + la $ao, res : para almacenar en el registro la frase a mostrar
  + li $v0, 4 : mostramos la frase
  + move $a0,$s1 : movemos el valor del contador de caracteres a $a0
  + li $v0, 1 : mostramos el valor del contador después de la frase

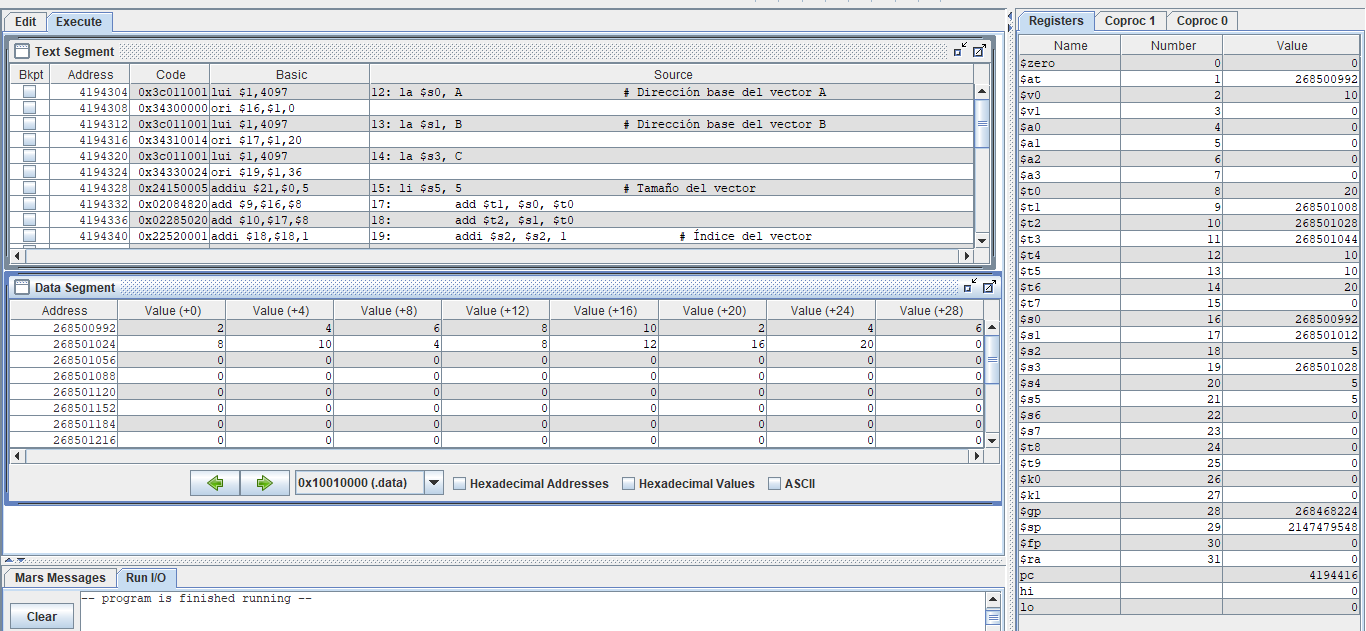
**Cuestión 5**

➢ **Completa el programa de la actividad 2 para que se rellene el vector C con la suma de los elementos del vector A y del B (C[i]=A[i]+B[i] i).**



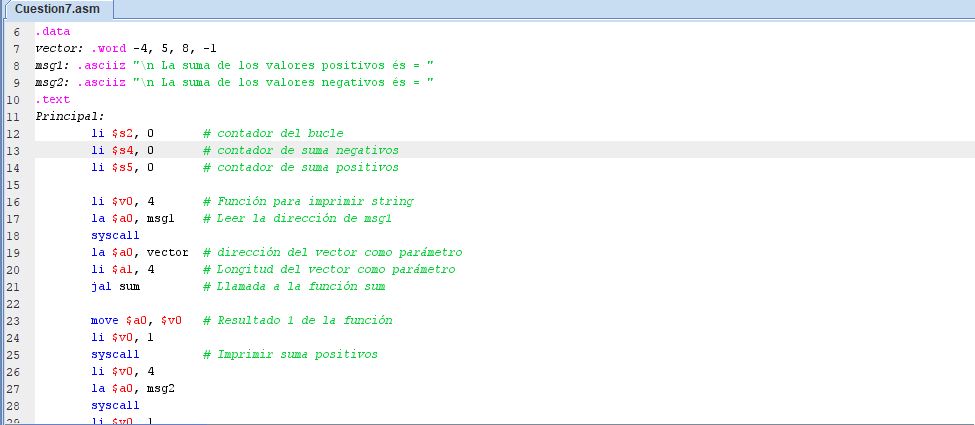
Primero he almacenado la dirección del vector C en $s3. Una vez finaliza el bucle loop he puesto a 0 los registros $t0 y $t3.

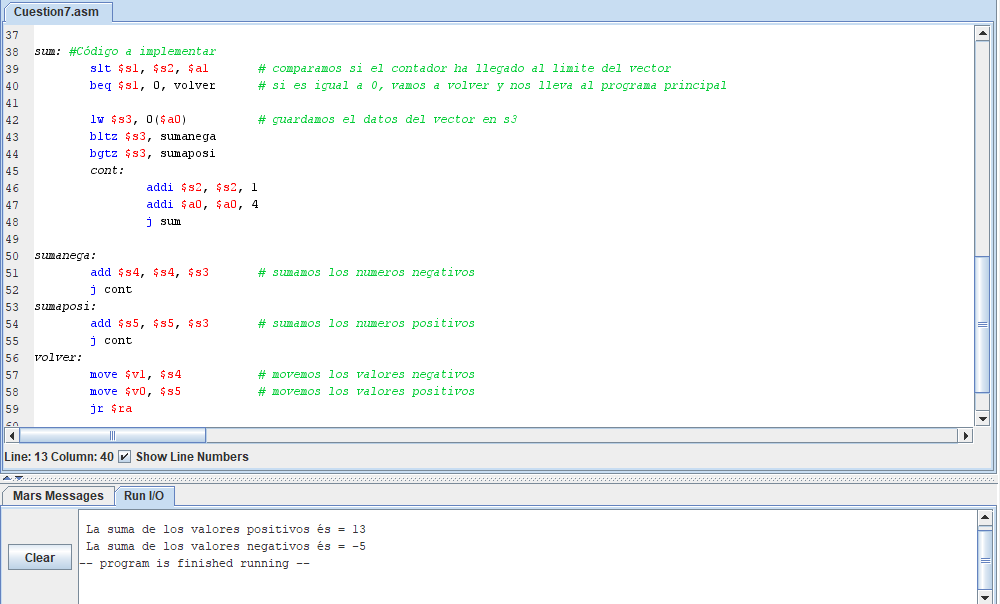
He creado el bucle suma en el cual con Load Word extraigo los datos de los vectores y los sumos, almacenando el resultado en el registro $t6. Una vez hecho la suma utilizo store Word para almacenarla en el vector C.



**Cuestión 7**

➢ **Analiza el programa de la cuestión 7 i escribe el código de la función sum que calcula la suma de los valores positivos y negativos del vector, dirección del cual se pasa como parámetro en $a0 y la longitud en $a1. La función devuelve en $v0 la suma de los valores positivos y en $v1 la suma de los negativos. Recuerda que en la función tienes que utilizar los registros $t j .**





Al principio del programa principal inicializamos los contadores de bucle y de las sumas.

En sum usamos la instrucción slt para comprobar si el contador del bucle ha llegado al límite, que es el tamaño del vector. Con beq comprobamos si es igual a 0, en caso de que lo fuera iríamos a la etiqueta volver, el cual nos devolvería al programa principal.

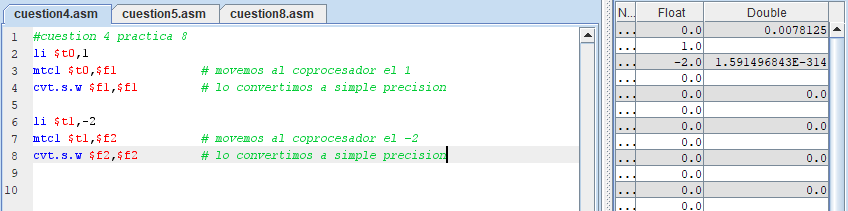
Con lw nos movemos por el vector y cargamos el valor en $s3, el cual según sea mayor o menos que 0 nos moveríamos a la función sumaposi o sumanega, dentro de ellas haríamos la suma y nos devolvería a la etiqueta cont, que esta dentro de suma para continuar con los cálculos.

Dentro de cont aumentamos en 1 el contador del bucle y sumamos 4 a la posición del vector, para pasar al siguiente valor, por último volveríamos a sum, para realizar de nuevo todas estas operaciones. Una vez terminada las operaciones y llegado al límite del bucle saltaríamos a volver, en el cual moveríamos las sumas de los negativos y positivos a $v0 y $v1, para ser mostrados en el programa principal.

**PRÁCTICA 8**

**Cuestión 4.**

**➢ Escribe el código que haba que el contenido de $f1 sea el valor 1 en coma flotante y el de $f2 el valor -2 en coma flotante utilizando las instrucciones de conversión de tipo.**

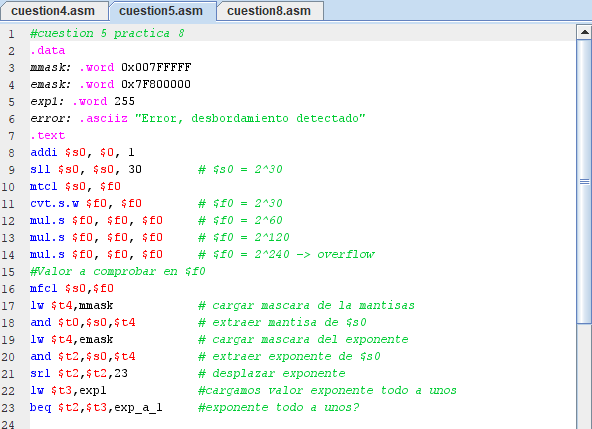


Con li en t0 metemos el valor 1, con mtc1 movemos el valor de t0 al coprocesador, almacenándolo en el registro f1. Por ultimo con cvt.s.w convertimos el valor que tenemos en f1 de Word a simple precisión.

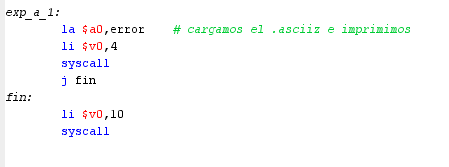
Haríamos este mismo proceso con el -2.

**Cuestión 5**

**➢ Completa el código de la actividad 6 para que muestre en consola un mensaje de error por desbordamiento. Comprueba que funciona correctamente.**

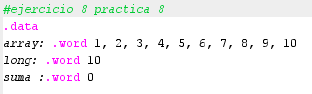


Lo que hacemos aquí es añadir la etiqueta exp\_a\_1 el cual llama el beq que tenemos en la última línea de la imagen de arriba. En ella se detecta si hay error de desbordamiento y en la etiqueta a la que llama cargamos el texto que queremos mostrar y lo imprimimos por pantalla.

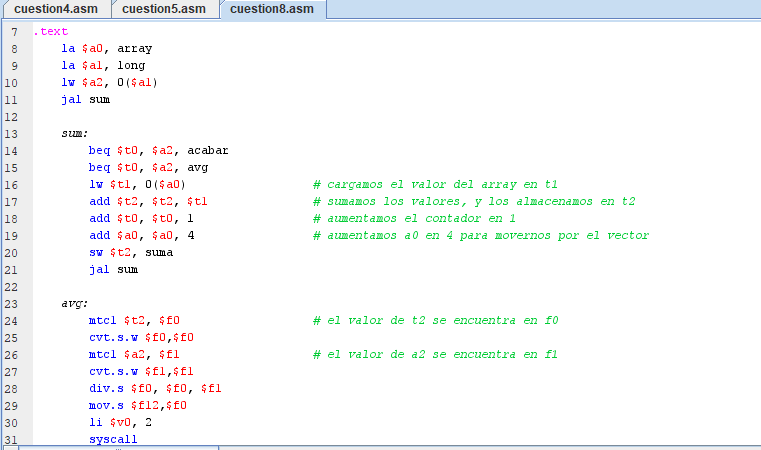


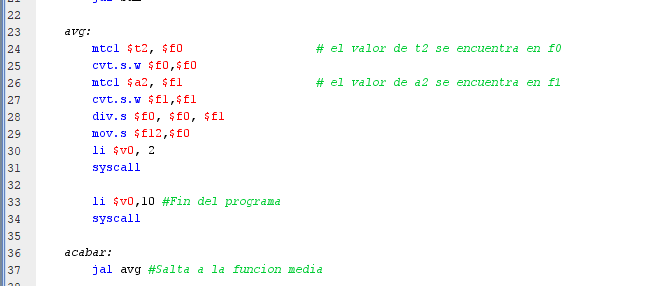
**Cuestión 8**

**A partir de la siguiente declaración de un vector de 10 elementos:**



**➢ Haz el código que suma los elementos del vector y calcula el valor medio en coma flotante. Muestra el resultado por la consola.**





En sum lo primero que hacemos es comprobar si hemos llegado al limite del bucle. Guardamos el valor del vector de esta iteración en t1. Sumamos el valor del vector y lo almacenamos en t2, después aumentamos el contador y también aumentamos a0 en 4, para pasar al siguiente valor del vector.

En avg lo que hacemos es calcular la media de estos numero, movemos t2 al registro f0 del coprocesador y lo convertimos a simple precisión.

**PRÁCTICA 9**

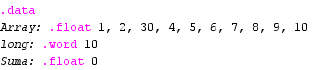
**Cuestión 1.**

**➢ ¿Cuál es la razón por la que el registro base de las instrucciones lwc1 y swc1 pertenecen al banco de registros de enteros y no de la FPU?**

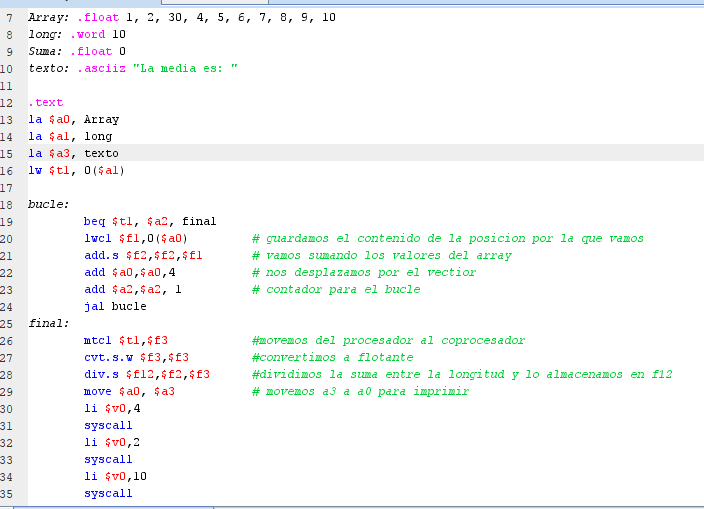
Porque esas dos instrucciones acceden a la memoria y la memoria se da en números enteros no en coma flotante.

**Cuestión 4.**

**A partir de la siguiente declaración de un vector de 10 elementos:**

****

**➢ Haz el código que suma los elementos del vector y calcula el valor medio. Muestra el resultado por la consola.**



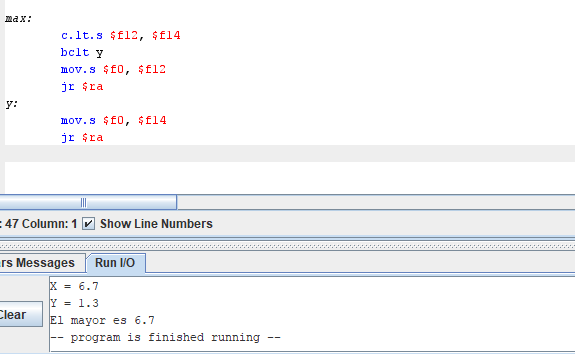
En bucle comprobamos si t1 es igual a a2, sino lo es, en la siguiente línea con lwc1 guardamos el contenido de la posición del vector de esta iteración en f1. Utilizamos add.s para sumar los valores del array e ir almacenándolos en f2. Con add sumamos 4 a a0 para desplazarnos al siguiente valor del vector y también aumentamos en 1 el contador del bucle.

Si t1 y a2 son iguales saltaríamos a final, en él moveríamos al coprocesador el valor almacenado en t1 y lo convertimos a simple precisión.

Con div.s dividimos la suma de los valores del vector entre la longitud del vector y lo almacenamos en f12.

**Cuestión 6**

➢ **Implementar la función max que nos devuelve el valor mayor de dos números en coma flotante. Los argumentos se pasan según convenio en $f12 y $f14 y el resultado se devuelve en $f0. Utilizad el siguiente código de partida:**



Con c.lt.s, si f12 es menor que f14 pone el flag del coprocesador a true sino lo es, se pondría a false. La siguiente línea bclt y, saltaría a la etiqueta y si el flag del coprocesador esta puesto a true. En caso de saltar a la etiqueta y asignaríamos el valor de f14 a f0 y volveríamos al programa principal.

Mov.s asigna el valor de f12 a f0 y con jr $ra volveríamos a la línea donde llaman a max.