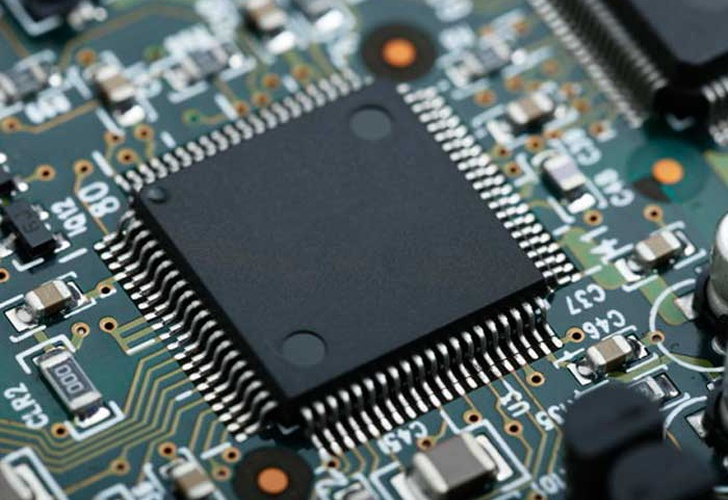
Estructura de los computadores

Luis Girona Pérez

12/03/2020

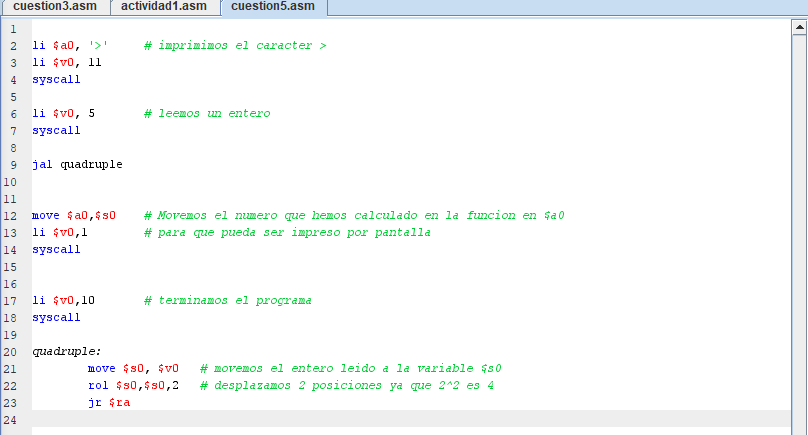
**MEMORIA DE PRÁCTICAS 4-5-6**



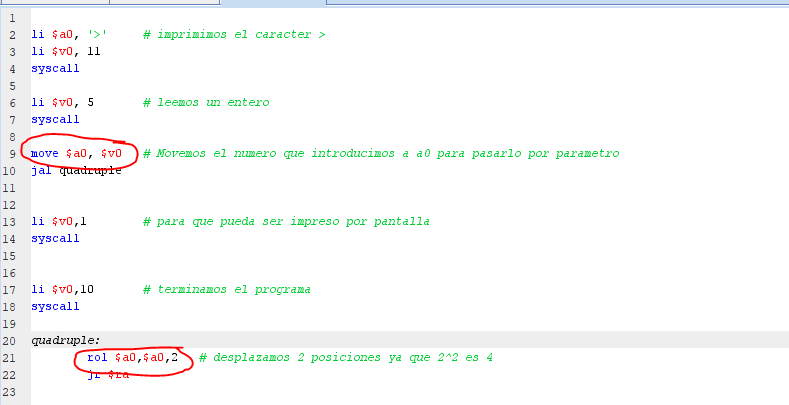
PRÁCTICA 4

**CUESTIÓN 5**

* **Observeu l’últim codi escrit de la qüestió 4, s’ajusta al conveni MIPS d’utilització de registres? Què hauríeu de modificar? Reescriviu el codi perquè s’ajuste al conveni. D’ara endavant feu servir el conveni d’utilització de registres.**



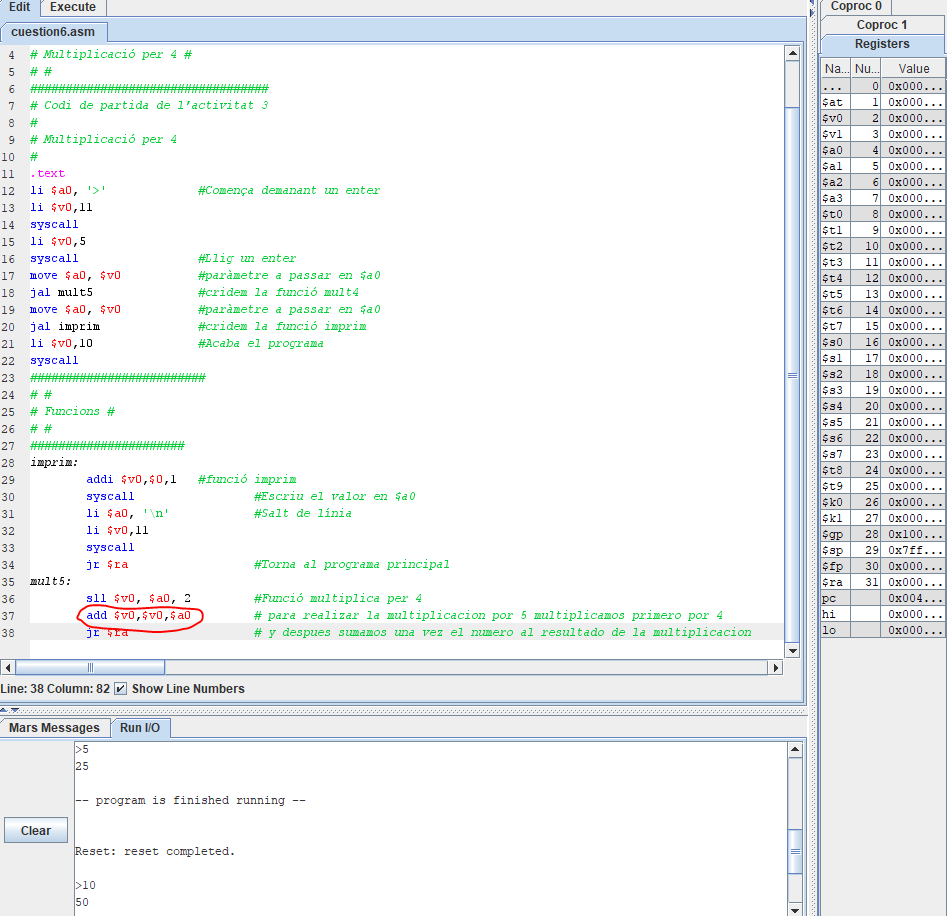
Este código es el que tenemos que modificar para cumplir el convenio. Tendríamos que cambiar las variables de paso de parámetros en la función de cuádruple, ya que utilizamos las variables s y son para el programa principal. Por lo que el programa nos quedaría tal que así.



He señalado los cambios que he realizado. El move modificado lo he hecho para tener el numero introducido en $a0 y pasarlo por parámetro a la función. El segundo cambio ha sido cambiar los $s utilizados dentro de la función (los cuales solo se usan en el programa principal) por $a0, para tenerlo guardado en una variable que se pase por parámetro y poder imprimirlo.

**CUESTION 6**

* **Modifiqueu el codi de l’activitat 3 perquè ara hi haja una funció mult5 que multiplique per 5 i mostreu el resultat per consola. (Fixeu-vos que al valor obtingut en la multiplicació per 4 haureu de sumar-li una vegada el valor original, x\*5=x\*4+x). Comproveu que el resultat és correcte.**



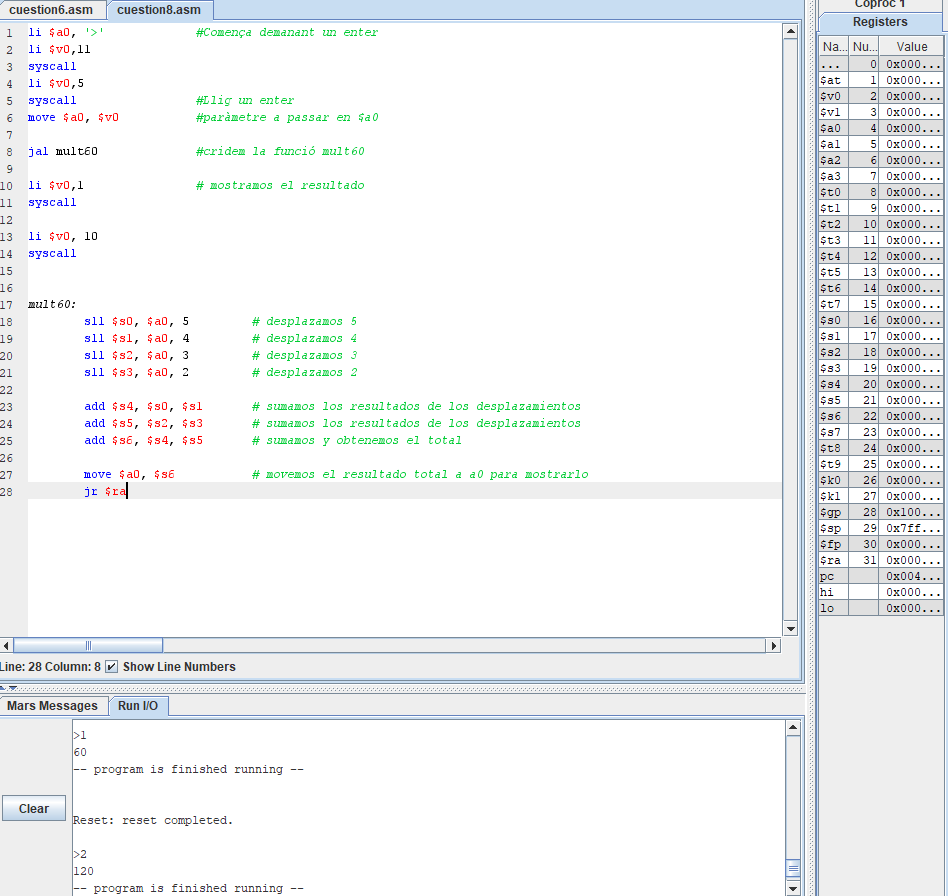
Para obtener la multiplicación por 5, lo único que he tenido que hacer ha sido añadir una línea de código con la instrucción add, lo que hace es sumar el número que hemos introducido en la consola con la multiplicación por 4 que hemos hecho en la línea de arriba, obteniendo así la multiplicación x5.

**CUESTION 8**

* **Escriviu una funció que multiplique per 60. Escriviu el programa principal que lliga una quantitat de minuts i retorne per consola la quantitat en segons.**

Tenemos que multiplicar el número que introduzcamos por 60. Al pasarlo a binario obtenemos el número 111100. Podemos decir que multiplicar por 60 es lo mismo que multiplicar el número por , por lo que la multiplicación la realizaremos mediante desplazamientos.

He traducido la operación descrita en el párrafo anterior como 4 desplazamientos a la izquierda de 5, 4, 3 y 2 bits. Después he sumado los resultados mediante 3 add, lo muevo a $a0 y lo muestro por pantalla.

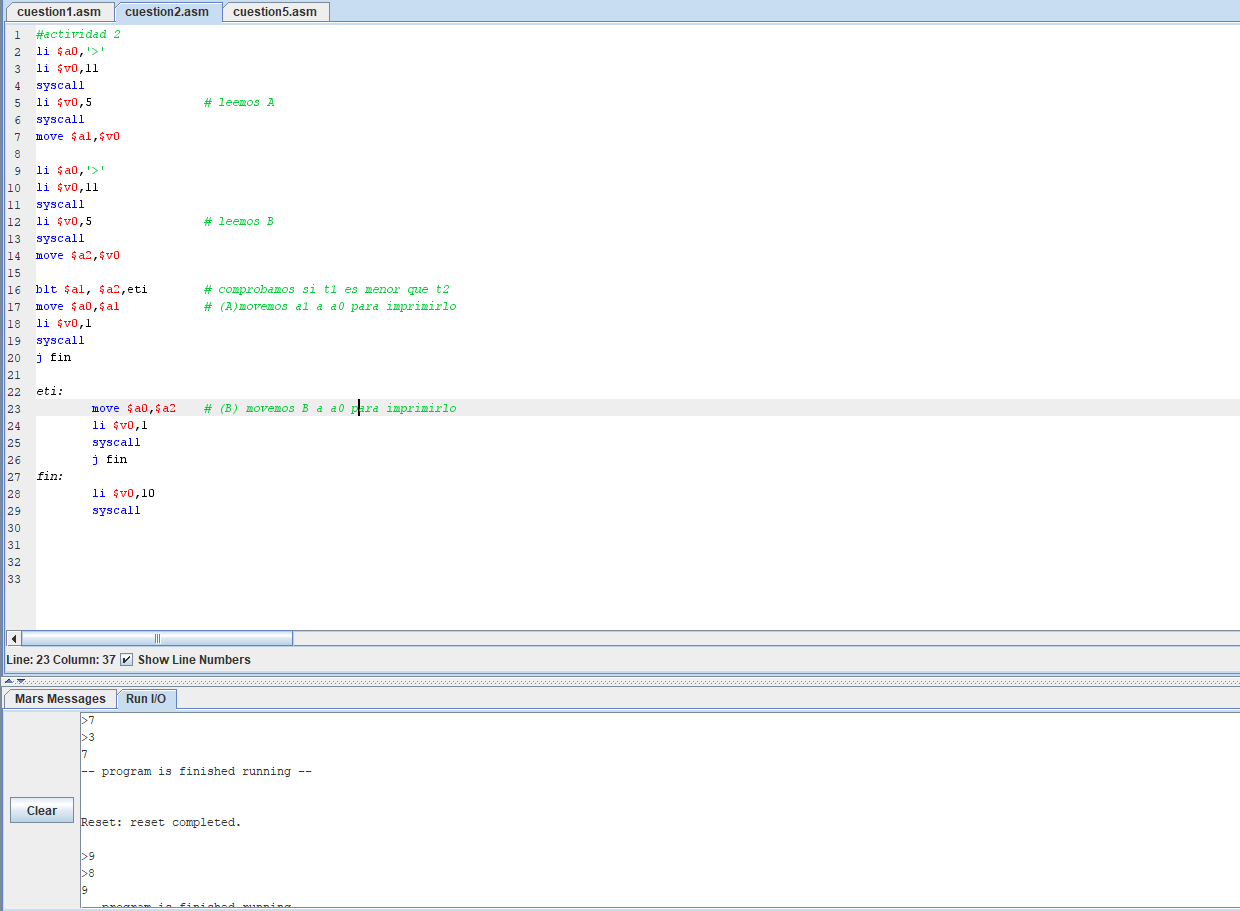


PRÁCTICA 5

**CUESTIÓN 2**

* **Escriviu el programa que llig dos enters del teclat i escriu en la consola el més gran. El programa ha de tenir l’estructura següent:**

**Assembla i prova el programa amb diferents valors d’A i B**

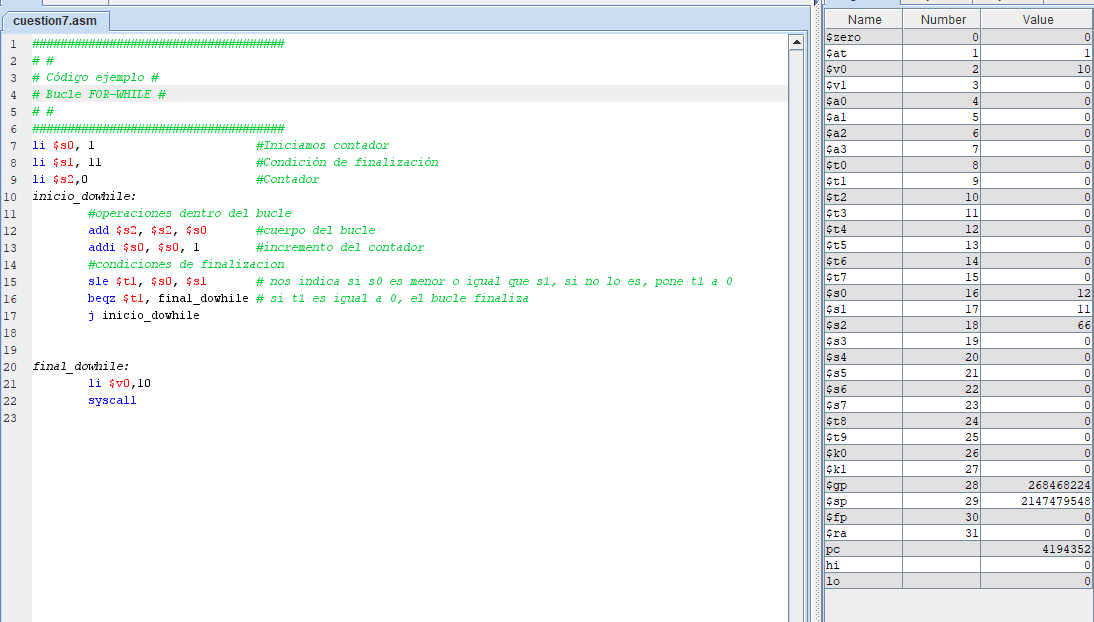


Lo que he hecho ha sido primero, pedir dos números con el syscall 5, después he utilizado blt para que salte a “eti” cuando el número A sea menor que B. Dentro de eti movemos el valor de B a $a0 para poder imprimirlo por pantalla y después saltamos a la etiqueta para finalizar el programa.

En caso de que B sea menor que A no saltaríamos a eti y seguiríamos con la instrucción siguiente, que sería mover el valor de A a $a0 para imprimir por pantalla y finalizaríamos el programa.

**CUESTIÓN 7**

* **En exemples com l’anterior, en què sabem que almenys el bucle s’executa una vegada, és més eficient posar la condició d’eixida del bucle al final. Reescriviu el codi anterior amb la condició d’eixida al final del bucle (es tracta d’un bucle del tipus do-while).**



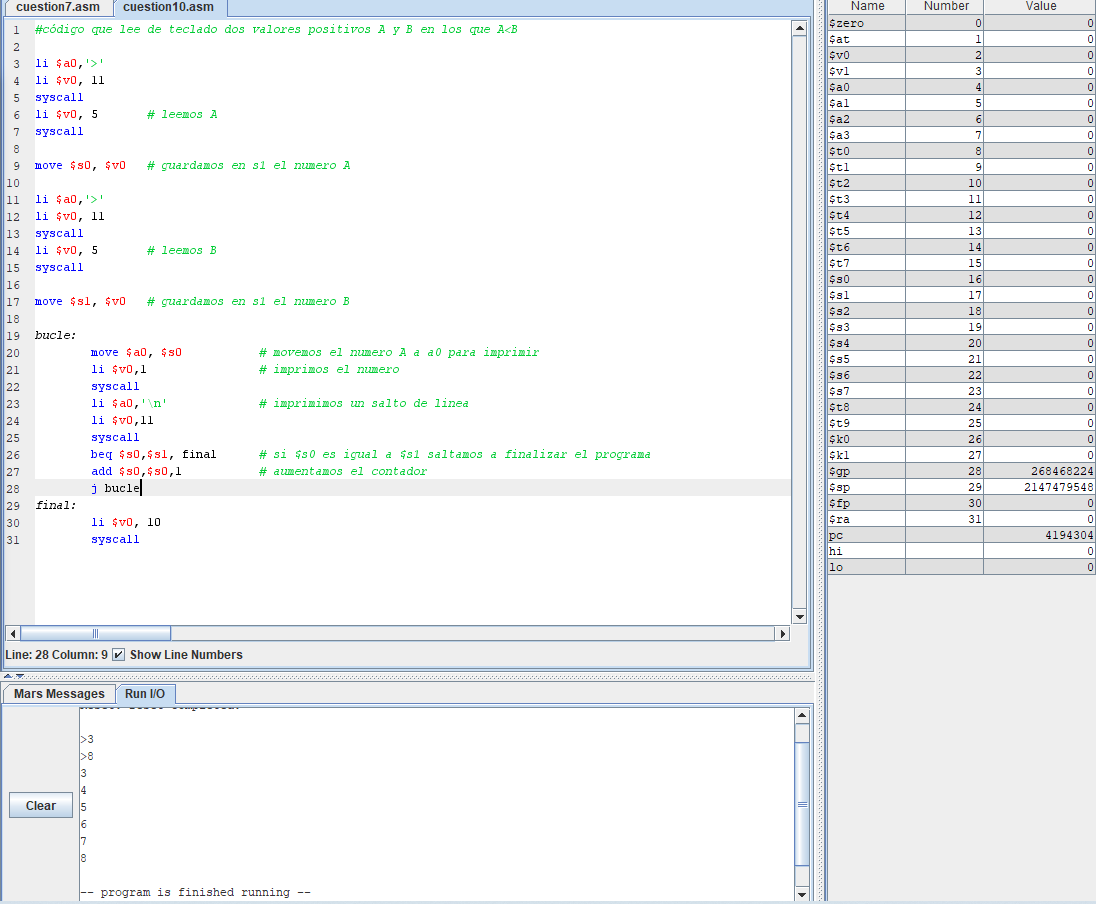
Lo que he hecho aquí ha sido modificar el cogido del ejemplo anterior de tal manera que haga primero las operaciones y después que haga las comprobaciones. Por lo que pasaríamos de un for a un do-while.

Al inicio del bucle realiza la operación e incrementamos el contador, después comprueba si el contador y el límite del contador son iguales. En caso de que lo fueran el registro $t1 se pondría a 0, en caso contrario a 1.

Por último, se comprueba si $t1 es igual a 0, es decir si el contador ha llegado a su límite o no. Si no ha llegado al límite, saltará de nuevo a la etiqueta inicio\_dowhile y volverá a ejecutar lo mismo que antes. En caso de que hubiera llegado irá a la etiqueta final\_dowhile y finalizará el programa.

**CUESTIÓN 10**

* **Feu el codi que llig de teclat dos valors positius A i B en els quals A<B. El programa ha d’escriure per consola els valors compresos entre ells inclosos ella mateixa. Es a dir, si A=3 i B=6, escriu en la consola 3 4 5 6 (podeu escriure, per exemple, un salt de línia després de cadascun dels valors a mostrar).**

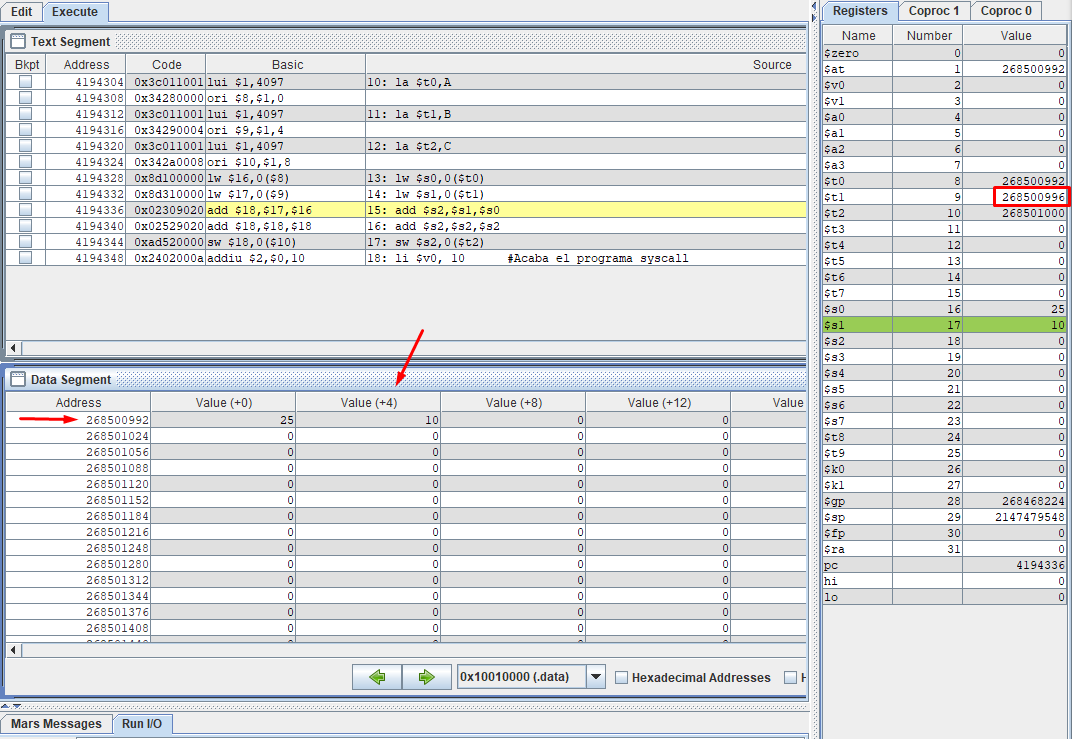


Solicitamos los dos números y dentro de un bucle movemos el número A (que lo utilizamos como contador) a $a0 para imprimirlo nada más comenzar, después tenemos un salto de línea. Ahora es cuando con la instrucción beq comprobamos si tenemos que terminar el bucle o no, miramos si $s0(contador) es igual a $s1(límite del contador), si lo es salta a final y termina el programa. Si no lo es, pasa a la siguiente instrucción que aumenta en 1 el contador y utiliza la instrucción de salto para volver al principio de bucle.

PRÁTICA 6

**CUESTIÓN 3**

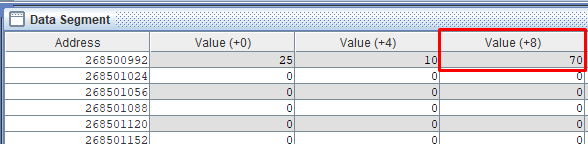
* **Quin valor té el registre $t1 quan s’executa la instrucció lw $s1,0($t1)?**



Como vemos en la imagen, $t1 tiene como valor 268500996. Podemos comprobar que ese valor coincide con la dirección del valor que tenemos en $s1. Ya que, si vemos, la flecha de la izquierda señala a 268500992 y la segunda flecha a +4, lo que coincidiría al hacer la suma con la dirección del valor 10 almacenado en $s1.

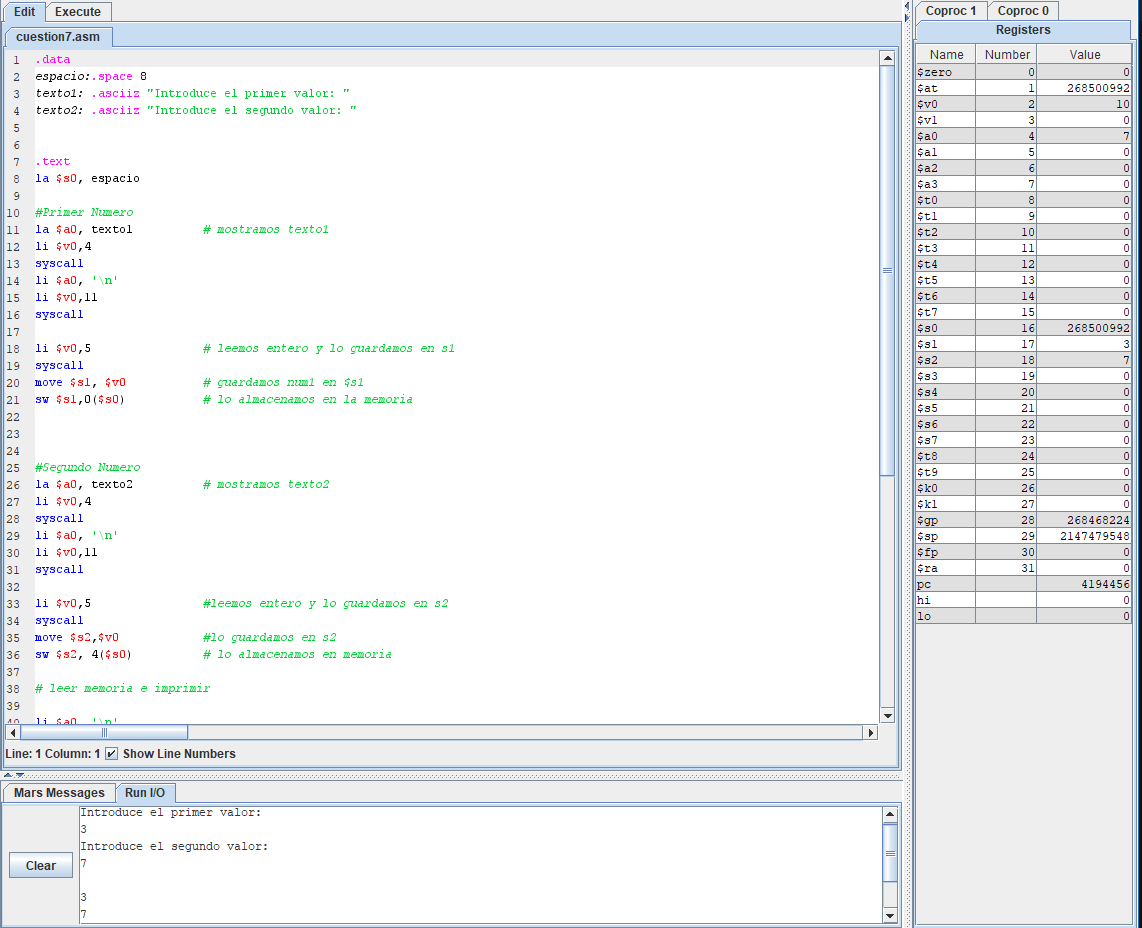
* **En quina adreça s’emmagatzema el resultat?**

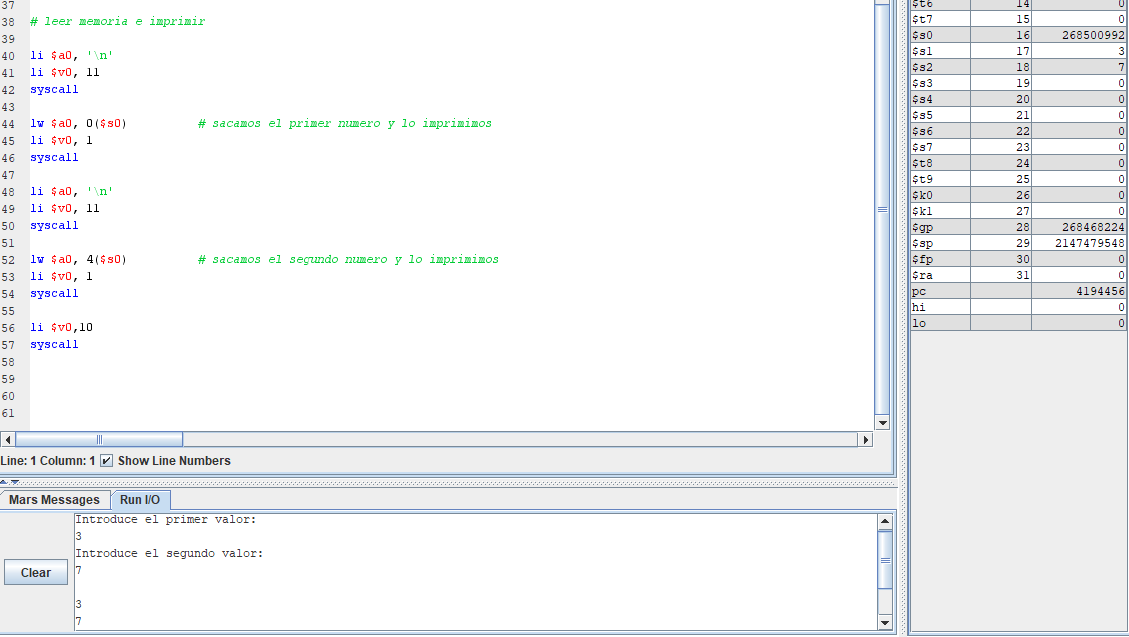
El resultado final del programa se almacena en la dirección 268501000



**CUESTIÓN 7**

* **Feu el codi que llig dos enters del teclat. Amb aquesta finalitat heu de mostrar dos missatges en la consola: un primer que demane a l’usuari que introduïsca un valor i una vegada llegit, que mostre un altre missatge demanant el segon valor. Les dades s’emmagatzemaran en posicions consecutives de la memòria; per això, prèviament haureu reservat espai en el segment de dades amb la directiva .space. A continuació el programa llegirà els valors guardats en la memòria i els mostrarà en la pantalla.**





Reservamos un espacio de memoria de 8. Declaramos dos .asciiz que serán los textos que mostraremos más a delante. Guardamos la dirección del espacio em $s0. Introducimos el primer numero y lo guardamos en la variable $s1, y mediante un sw lo almacenamos en memoria con sw $s1, 0($s0), siendo 0 el comienzo del espacio y $s0 la dirección del espacio.

Con el segundo numero hacemos exactamente lo mismo salvo a la hora de almacenarlo en memoria, ya que utilizaremos sw $s2, 4($s0), aumentamos 4 para pasar a la siguiente palabra.

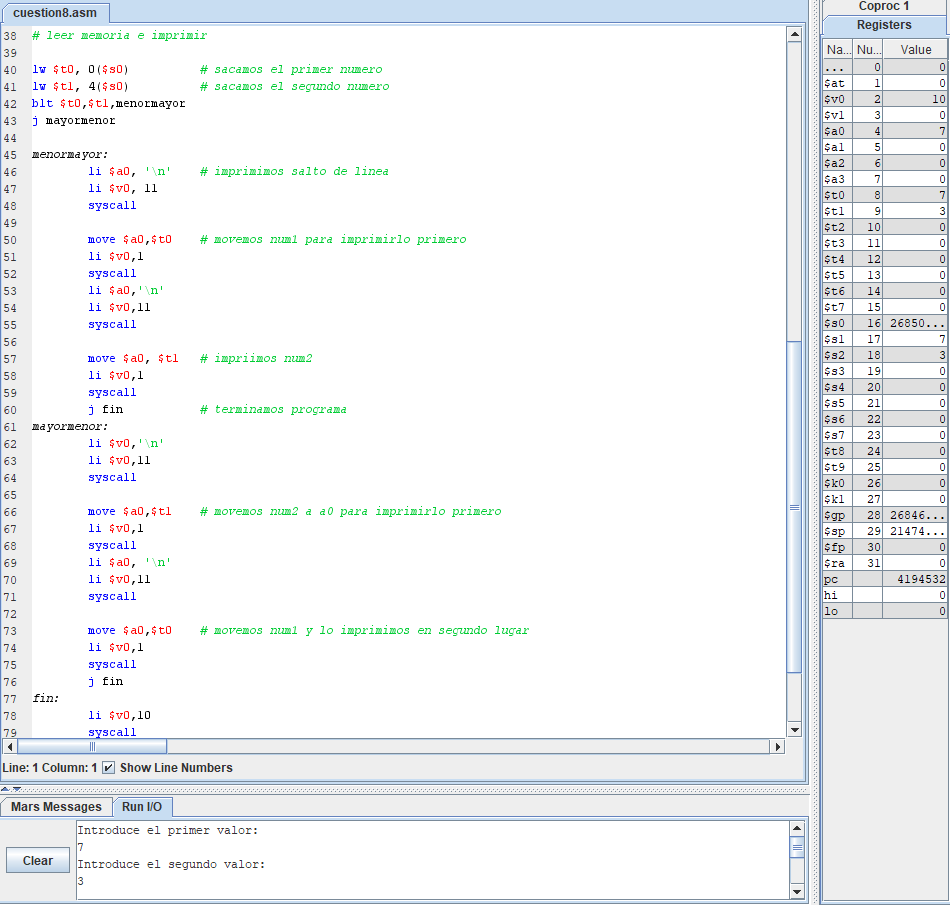
Ahora pasaríamos a leer los datos de la memoria e imprimirlos por pantalla. Para esto utilizaremos la instrucción lw $a0, 0($s0), cargamos lo que haya almacenado en la posición 0($s0) en el registro $a0 y con su syscall correspondiente lo imprimimos.

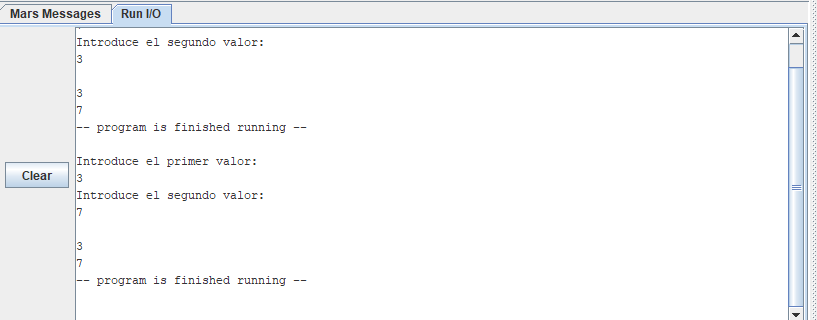
Con el segundo número sería igual cambiando la dirección de memoria por 4($s0).

**CUESTIÓN 8**

* **Modifiqueu el programa de la qüestió 7 perquè mostre en la pantalla les dues dades guardades en la memòria ordenades de menor a major valor.**

El código es el mismo que en la cuestión anterior salvo en la parte de imprimir, que es la siguiente:





Lo que hacemos es cargar los números con la instrucción lw en un registro, $t0 y $t1 respectivamente. Utilizamos la instrucción blt para que salte a la etiqueta menormayor si $t0 es menor que $t1. En ese caso iríamos a la función menormayor e imprimiríamos primero $t0 y luego $t1.

En caso de que el mayor fuera $t0, no realizaría el salto a la etiqueta menormayor, sino que pasaría a la siguiente instrucción que sería un salto a la etiqueta mayormenor, en la que imprimiríamos primero el registro $t1 y luego $t0 y finalizaría el programa.