ESTRUCTURA DE LOS COMPUTADORES

MEMORIA DE PRÁCTICAS 1-2-3

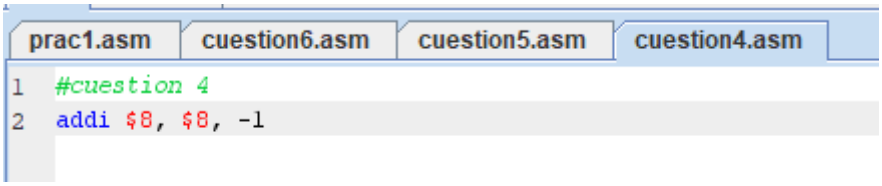
Luis Girona Pérez

19/02/2020

PRÁCTICA 1

CUESTIÓN 4

Com s'escriu la instrucció que fa $8=$8-1?



La instrucción addi suma elementos, por lo que haría lo siguiente, al segundo parámetro ($8), le sumamos un -1.

El resultado de esta operación es guardado en el primer parámetro que aparece, $8.

Com en quedaria la codificació en binari

0010 0001 0000 1000 1111 1111 1111 1111

001000	01000	01000	1111 1111 1111 1111
Operación	RS	RT	K

CUESTIÓN 5

prac1.asm | cuestion6.asm | cuestion5.asm | cuestion4.asm

```
1  .text 0x00400000
2  #addi $8,$0,1
3  #addi $9,$8,25
4  #addi $10,$8,5
5  #addi $12,$10,0
6  #Convenio de registros
7  addi $t0,$zero,1
8  addi $t1,$t0,25
9  addi $t2,$t0,5
10 addi $t4,$t2,0
```

Como vemos en la imagen, el texto comentado sería el código anterior, mientras que el de abajo es con el convenio de registros.

CUESTIÓN 6

Escriu el codi que faça les següents accions utilitzant el conveni de registres i utilitzant la instrucció addi:

$12=5

$10= 8

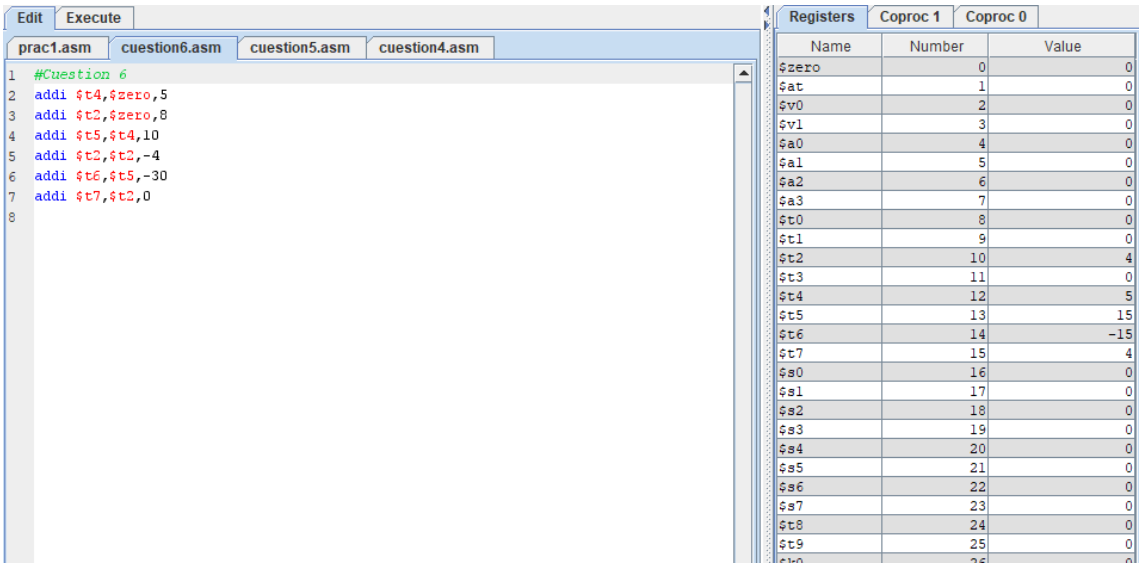
$13=$12 + 10

$10=$10 - 4

$14=$13 – 30

$15=$10

Assembleu i executeu el programa i comproveu que el resultat final és $t7=$t2=4, $t6=-15, $t4=5, $t5=15.



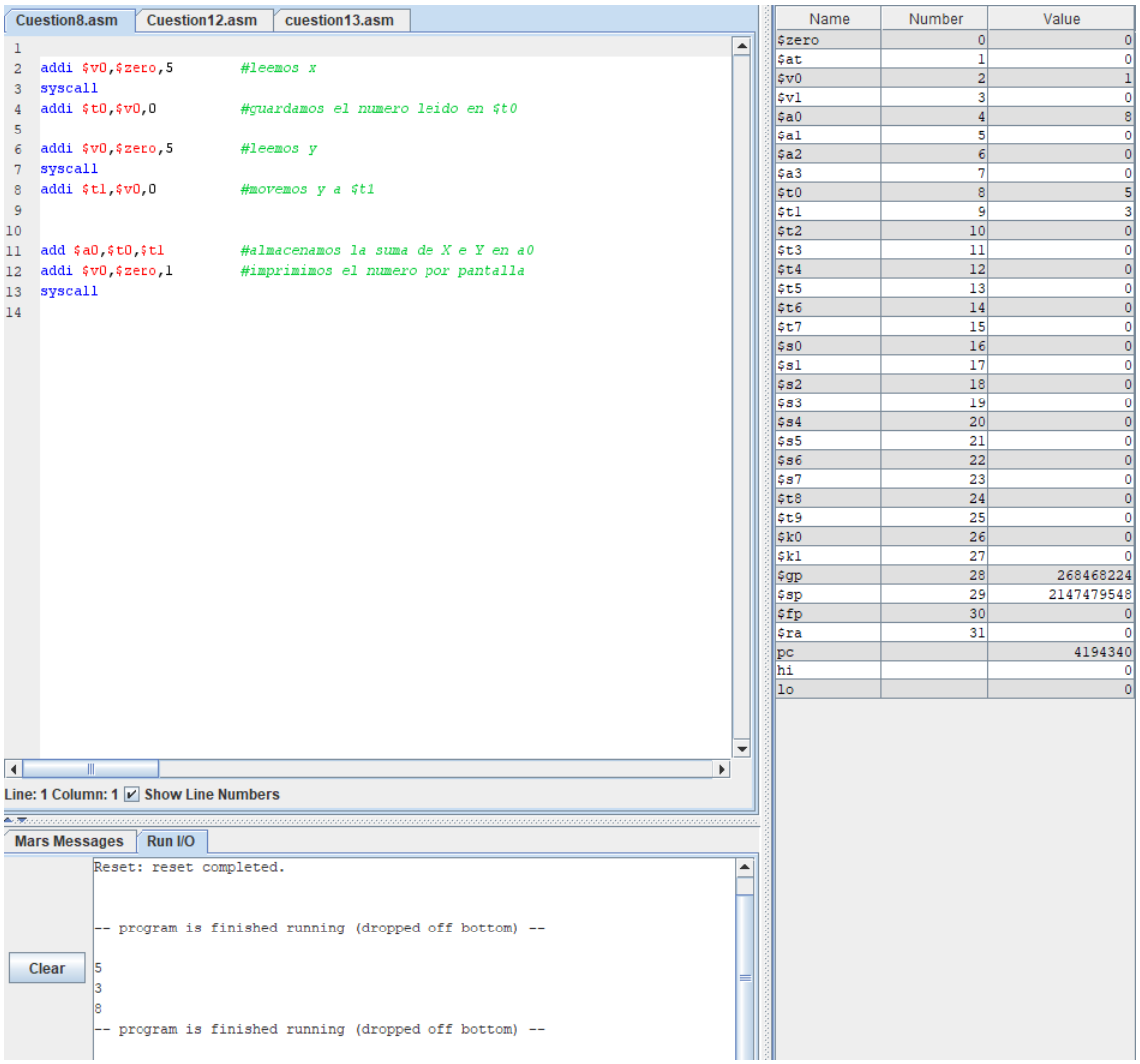
Dado el “esquema” del código, utilizamos las instruccions de suma pertinentes y el convenio de registros. Nos quedaria un código como el de arriba.

Como vemos, podemos comprobar el resultado es el deseado.

PRÁCTICA 2

Cuestión 8

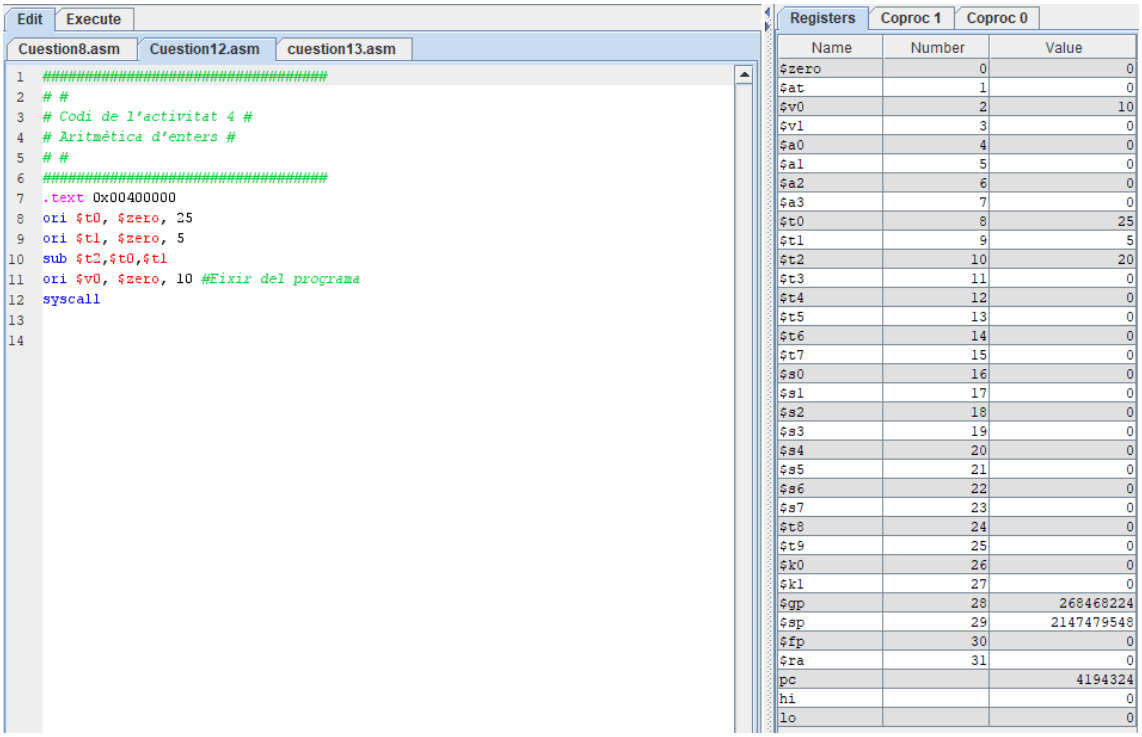
Feu el codi que llig dos nombres x i y i obté per pantalla el valor de la suma x+y.



Ponemos $v0 con valor 5 para leer un entero (X), lo guardamos en otra variable y volvemos a leer otro entero (Y). Utilizamos la instrucción addi para sumar los enteros X e Y, y la almacenamos en la variable $a0. Por último, imprimimos el numero sumado por pantalla.

Cuestion 12

Reescriviu el codi de partida Aritmètica d'enters de l'activitat 4 canviant addi o addiu per ori:



Quin és el valor del codi d'operació de la instrucció ori?

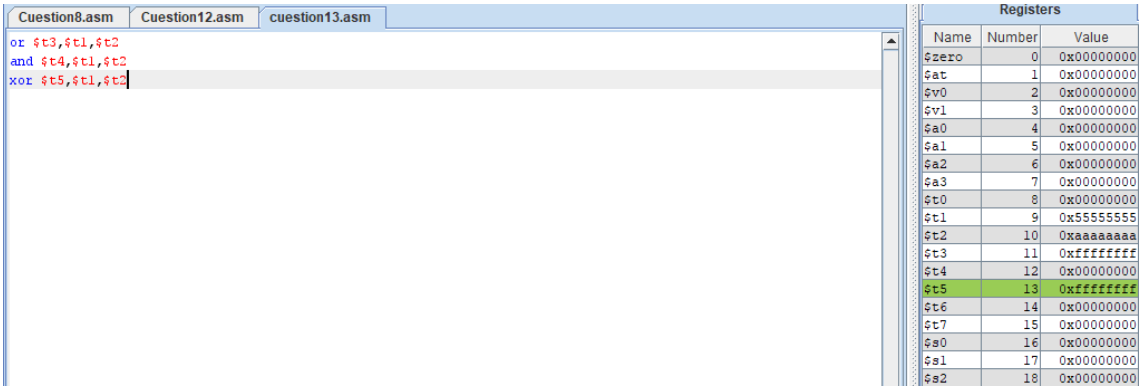
El código de la instrucción en hexadecimal: 0x34080019

Código de la instrucción en binario: 0011 0100 0000 1000 0000 0000 0001 1001

El código de operación son los primeros 6 bits, es decir 001101

Cuestión 13

Escriviu el codi que faça l'operació lògica OR de $t1 i $t2 i el guardeu en $t3, l'operació lògica AND de $t1 i $t2 i la guardeu en $t4 i l'operació lògica XOR de $t1 i $t2 i la guardeu en $t5. Escriviu en la finestra de registres, després d'assemblat, els següents valors per als registres $t1=0x55555555 i $t2= 0xAAAAAAA. Executa el codi i estudia els resultats.



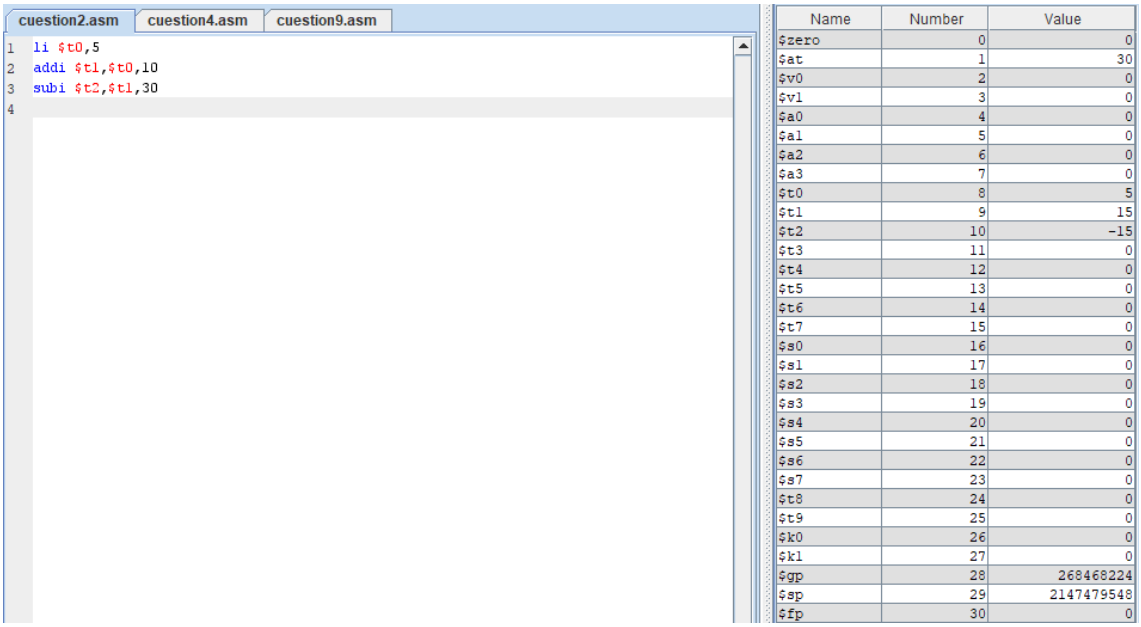
Como vemos en la OR, nos hace una suma de 5($t1) y 10($t2), la puerta AND al no ser iguales obtenemos un resultado de 0, y por último, en la puerta XOR al ser distintos $t1 y $t2, obtenemos la suma de estos, en caso de que tuvieran el mismo valor, el resultado seria 0.

Práctica 3

Cuestión 2

Escriviu el codi (3 línies entre instruccions i pseudoinstruccions) que fa aquestes accions:

$t0=5 $t1=$t0+10 $t2=$t1-30



cuestion2.asm | cuestion4.asm | cuestion9.asm

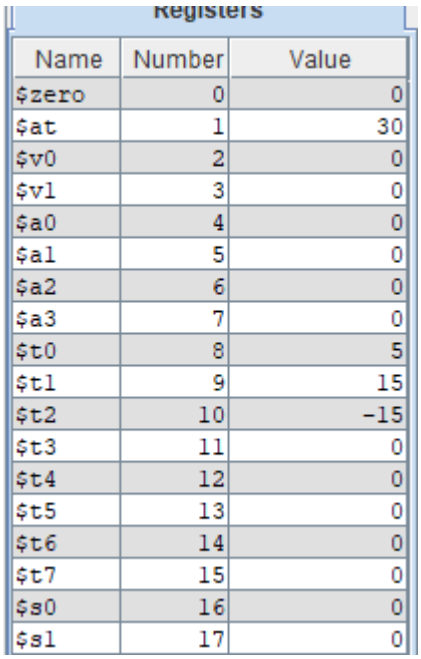
```
1  li $t0,5
2  addi $t1,$t0,10
3  subi $t2,$t1,30
4
```

Name	Number	Value
$zero	0	0
$at	1	30
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	5
$t1	9	15
$t2	10	-15
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0

Utilizamos la pseudo-instrucción li para cargar un inmediato en $t0.

Y para las otras dos líneas utilizamos dos instrucciones para suma y resta.

Assembleu i executeu el programa. Comproveu que el resultat final ($t0=5, $t1=15, $t2=-30) és correcte.



Registers

Name	Number	Value
$zero	0	0
$at	1	30
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	5
$t1	9	15
$t2	10	-15
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0

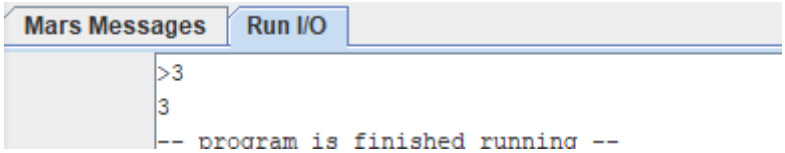
Como vemos los resultados obtenidos son los mismos salvo en $t2, que según la instrucción, subi $t2, $t1,30, tendríamos que restar 30 a $t1, con valor de 15, lo que nos daría -15. Ese es el resultado que nos da en el MARS aunque en el enunciado nos indica -30.

Cuestión 4

Amb la finalitat de facilitar la claredat en l'execució dels programes, modifiqueu el codi de la qüestió 3 perquè mostre '>' o '?' abans de llegir el caràcter del teclat com una manera de demanar-lo.

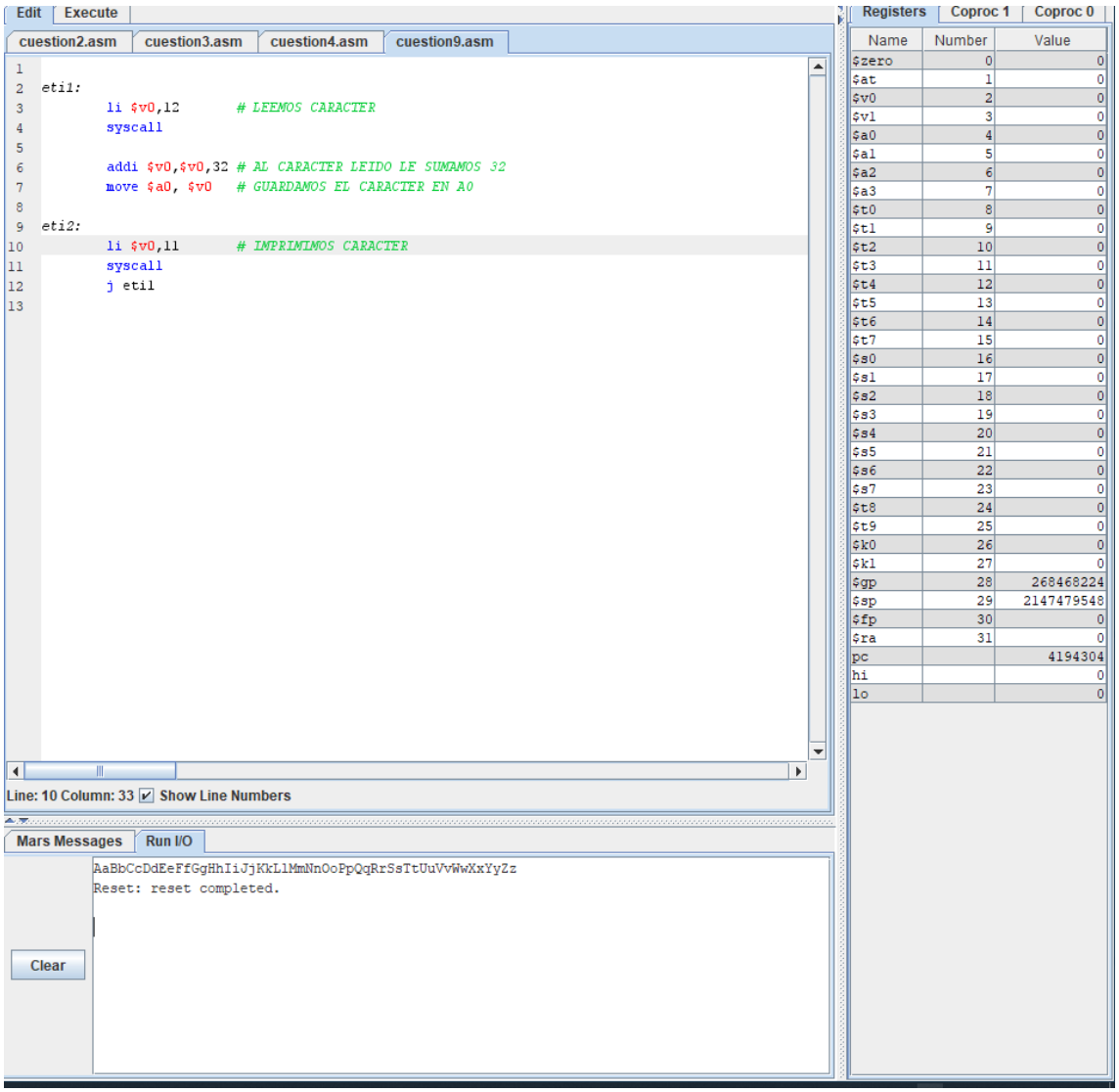
```
1
2  li $a0,'>'      #imprimimos el caracter >
3  li $v0,11
4  syscall
5
6
7  li $v0,12       # Funció 12. Read character
8  syscall         # Caràcter llegit en $v0
9
10 move $t0,$v0    # guardamos el caracter en la variable t0
11 li $a0,'\n'     # guardamos en a0 el caracter de salto de linea
12 li $v0,11       # imprimimos el salto de linea
13 syscall
14 move $a0,$t0    # guardamos en a0 el caracter que queremos imprimir, almacenado en t0
15
16
17 li $v0,11       # Funció 11. Print character
18 syscall
19 li $v0, 10      #Funció 10. Acaba programa
20 syscall
```

Usamos li para cargar en la variable $a0 el carácter '>', después usamos el syscall 11 para imprimir el carácter en la consola. Una vez hecho esto nos mostrará el carácter, seguido de la introducción del dato que pedimos en el siguiente syscall.



Cuestion 9

Itereu el codi que acabeu d'escriure. És a dir, utilitzeu la instrucció j perquè s'execute indefinidament



Lo que hacemos aquí es leer un carácter con valor de $v0 en 12. El carácter introducido es guardado en $v0, a esta variable le sumamos 32, ya que es la diferencia entre letras mayúsculas y minúsculas en la tabla ASCII. Imprimimos el carácter y hacemos un j eti1 lo que nos haría volver al principio del código y volveríamos a repetir esta operación.