ENTREGA 1 - INSTRUCCIONES Y REGISTROS

Universitat d'Alacant

Nombre: Iván Soler Sánchez

Grupo: ARA

Fecha: 02/02/2024

ÍNDICE

PRÁCTICA 1

Objetivos

El objetivo principal de esta práctica es introducirnos a esta nueva asignatura y a su vez nos servirá de toma de contacto con el material que estaremos usando en las próximas prácticas y con el lenguaje ensamblador.

Realizaremos las cuestiones C2, C4 y C6.

C2

- ¿Cómo se codifica la instrucción *addi $10,$8,5*? Escribid el código resultante en hexadecimal

La instrucción addi está formada por 3 elementos, el primero ($10) es el registro en el cual vamos a almacenar el resultado de la suma, el segundo ($8) es el primero de los dos números que vamos a sumar, en este caso se trata del contenido que haya en el registro $8, y en tercer y último lugar, tenemos la cantidad que queremos sumar al anterior número, en este caso 5.

prac1.asm*

```
1  ######################################
2  #                                    #
3  #                                    #
4  #                                    #
5  #  Primer programa                   #
6  #                                    #
7  #                                    #
8  #                                    #
9  ######################################
10
11 .text 0x00400000
12 addi $10,$8,5
13
```

8	0x00000000

Hemos sumado 5 al contenido del registro $8 (0), y lo hemos almacenado en el registro 10.

10	0x00000005

Code
0x210a0005

El resultado es 5, que en hexadecimal es 00000005.

C4

- ¿Cómo se escribe la instrucción que hace $8 = $8-1?
- ¿Cómo quedaría su codificación en binario?

Bkpt	Address	Code	Basic
☐	0x00400000	0x2108ffff	addi $8,$8,0xffffffff

```
.text 0x00400000
addi $8, $8, -1
```

Estamos sumando el valor de $8 (0) y -1, cuyo resultado es -1, que en complemento A2 (3 bits) quedaría de la siguiente manera:

1 → 001, lo pasamos a complemento A1 invirtiendo los 1 por los 0 = 110, por último, lo pasamos a complemento A2 sumándole 1, 110+1=111

Su codificación en hexadecimal es 0x2108ffff, en binario: 00100001000010001111111111111111

C6

- Escribe el código que haga las siguientes acciones utilizando el convenio de registros y utilizando la instrucción addi:

  $12=5
  $10= 8
  $13=$12 + 10
  $10=$10 - 4
  $14=$13 – 30
  $15=$10

- Ensamblad y ejecutad el programa y comprobad que el resultado final es $t7 = $t2 = 4, $t6=-15, $t4=5,$t5=15.

El código sería el siguiente:

```
.text 0x00400000
addi $t4, $zero, 5
addi $t2, $zero, 8
addi $t5, $t4, 10
addi $t2, $t2, -4
addi $t6, $t5, -30
addi $t7, $t2, 0
```

$t2	10	0x00000004
$t3	11	0x00000000
$t4	12	0x00000005
$t5	13	0x0000000f
$t6	14	0xfffffff1
$t7	15	0x00000004

$t2 = 4 // $t7 = 4
$t6 = 0xffffff1 → $t5 = 15, le sumamos -30 y por ende = -15
$t4 = 5
$t5 = 15 (como hemos visto en el cálculo del valor almacenado en $t6)

PRÁCTICA 2

Objetivos

En la práctica anterior ya tuvimos nuestra primera toma de contacto con el ensamblador, en esta práctica empezaremos a implementar funciones de entrada y de salida, empezaremos a conocer instrucciones lógicas y la codificación de instrucciones.

Realizaremos las cuestiones C2, C7 y C14.

C2

Cuestión 2.

- Haz un código que lee un valor *x* de teclado y escribe *x-1* en la consola.

```
1  .text 0x00400000
2
3  addi $v0,$0,5
4  syscall
5
6  subi $a0, $v0, 1
7  addi $v0,$0, 1
8  syscall
9
10 addi $v0, $0, 10
11 syscall
```

Primero estamos llamando a la función 5, que se encarga de leer el número que nosotros pasemos por teclado, seguidamente restamos 1 a lo que habíamos leído y lo pasamos a $a0, debido a que queremos que este valor sea mostrado en la consola, y por último, lo mostramos y salimos del programa con la función 10.

C7

Cuestión 7.

- Escribe el código que hace estas acciones haciendo uso de las instrucciones estudiadas:

  $t0=5
  $t1=$t0+10
  $t2=$t0+$t1
  $t3=$t1-30

- Ensambla y ejecútalo y comprueba que el contenido de los registros es correcto.
- A la vista del código escrito, ¿es necesario que forme parte del repertorio de instrucciones la instrucción `subi`?

En primer lugar, guarda en $t0 5, en segundo lugar suma el valor de $t0 a 10 y lo guarda en $t1, seguidamente suma el valor de $t0 y $t1 y lo guarda en $t2, y por último, resta 30 al valor de $t1 y lo guarda en $t3.

```
1  .text 0x00400000
2
3  addi $t0, $0, 5
4  addi $t1, $t0, 10
5  add $t2, $t0, $t1
6  subi $t3, $t1, 30
7
8  addi $v0, $zero, 10 #Salir del programa
9  syscall
```

$t0	8	0x00000005
$t1	9	0x0000000f
$t2	10	0x00000014
$t3	11	0xfffffff1

En el registro $t0 tenemos 5, correcto.
En el registro $t1 tenemos f (15), correcto.
En el registro $t2 tenemos 20, correcto.
Y por último, en el registro $t3 tenemos -15, correcto.

No es necesario la instrucción subi ya que podríamos emplear la instrucción addi y sumar un número negativo por ejemplo.

C14

Cuestión 14.

- Supón que $t1=0x0000FACE, utilizando únicamente las instrucciones lógicas de la tabla anterior, escribe el código que reordene los bits de *$t1* de manera que en *$t2* aparezca el valor 0x0000CAFE. Ensambla y escribe en la ventana de registros $t1=0x0000FACE. Ejecuta y comprueba que el código es correcto.

Pasando los números a decimal para aclarar la operación tenemos que:
0x0000FACE = 64206 // 0x0000CAFE = 51966, por ende tenemos que pasarlos a binario y ver bit por bit con la tabla de verdad de la XOR que bit corresponde, pasar ese número a decimal y pasarlo a la instrucción xor, dicho número es el 12336.

$t1	9	0x0000face
$t2	10	0x0000cafe

```
1  .text 0x00400000
2
3  xor $t2, $t1, 12336
4
5  addi $v0, $zero, 10 #Salir del programa
6  syscall
```

PRÁCTICA 3

Objetivos

El objetivo principal de esta práctica es operar teniendo en cuenta el desbordamiento y así entender su significado, conocer algunas de las pseudoinstrucciones del MIPS. En la práctica anterior vimos las funciones de sistema y entrada de enteros, en esta práctica veremos cómo realizar una tarea similar pero con caracteres.

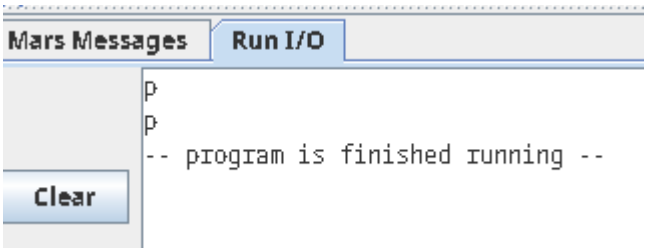
Realizaremos las cuestiones C3, C6 y C10.

C3

Cuestión 3.

- El carácter '\n' es el de nueva línea y provoca que la salida por consola desde el programa comience en una línea nueva. Modifica el código anterior para que al leer un carácter muestre la salida en una línea diferente.

```
1  .text
2
3  li $v0, 12
4  syscall
5  move $t0, $v0
6
7
8  li $a0, '\n'
9  li $v0,11
10 syscall
11
12 move $a0, $t0
13 li $v0,11
14 syscall
15
16 li $v0,10
17 syscall
18
```



C6

Cuestión 6.

- Escribe el código que imprime por consola el carácter correspondiente al valor ASCII leído en decimal desde el teclado (ayudado por la tabla ASCII anterior).

c3.asm | c7.asm

```
1  .text
2
3  li $a0, '>'
4  li $v0,11
5  syscall
6
7  li $v0, 12
8  syscall
9  move $t0, $v0
10
11
12 li $a0, '\n'
13 li $v0,11
14 syscall
15
16 move $a0, $t0
17 li $v0,34
18 syscall
19
20 li $v0,10
21 syscall
22
```

```
>f
0x00000066
-- program is finished running --
```

Para ello simplemente tendremos que usar la función li $v0, 34 que imprime por consola el valor hexadecimal del carácter que le es pasado.

C10

Cuestión 10.

- Convierte caracteres numéricos. Escribe el código que lea del teclado un carácter numérico (del '0' al '9') y lo convierta en un valor numérico (del 0 al 9) y lo escriba por pantalla. Itera el código.

```
1  .text
2
3  etiq1:
4  li $a0, '>'
5  li $v0, 11
6  syscall
7
8  li $v0,12
9  syscall
10
11
12
13 eitq2:
14 subi $a0, $v0, 48
15 li $v0, 1
16 syscall
17 j etiq1
```

```
>11>22>33>44>55>66>77>88>99>00>
```

PRÁCTICA 4

Objetivos

El objetivo principal de esta práctica es entender el funcionamiento de las multiplicaciones y divisiones enteras mediante desplazamientos y de las propias funciones de multiplicación y división de enteros del MIPS.

Realizaremos las cuestiones C2, C7, C9

C2

Cuestión 2.

- Las instrucciones de desplazamiento siguen el formado tipo R. ¿Cuál es la codificación en hexadecimal de la instrucción `sll $t2,$t1,3`? Y de la instrucción `srl $t2,$t1,7`. ¿Cuál es el valor de cada campo? Ayúdate ensamblado un código de prueba.

En hexadecimal se codifican de la siguiente manera:

Bkpt	Address	Code	Basic	
☐	0x00400000	0x000950c0	sll $10,$9,0x00000003	3: sll $t2, $t1, 3
☐	0x00400004	0x000951c2	srl $10,$9,0x00000007	4: srl $t2, $t1, 7

```
1   .text
2
3   sll $t2, $tl, 3
4   srl $t2, $tl, 7
```

pasado a binario sería:
sll = 00000000000010010101000011000000
srl = 00000000000010010101000111000010

El formato tipo R se codifica de la siguiente manera:

Codi op (6 bits)	*Rs (5 bits)*	*Rt (5 bits)*	*Rd (5 bits)*	*Shamt (5 bits)*	*Funció (6 bits)*
A-L	$s1	$s2	$t1	-	suma
0	17	18	9	0	32
0000 00	10001	10010	0 1001	0 0000	10 0000

Por lo tanto, podemos decir que:

ssl) op 0000 00, Rs 00000, Rt 01001, Rd 01010, Shamt 00011 y funció 000000

srl) op 0000 00, Rs 00000, Rt 01001, Rd 01010, Shamt 00111 y funció 000010

C7

Cuestión 7.

- Modifícalo ahora para tener una nueva función mult10 que multiplique por 10. Comprobar que el resultado es correcto.

```
1  .text
2  li $a0, '>'
3  li $v0,11
4  syscall
5  li $v0,5
6  syscall
7  move $a0, $v0
8  jal mult10
9  move $a0, $v0
10 jal imprim
11 li $v0,10
12 syscall
13
14 imprim:
15         addi $v0,$0,1 #función imprim
16         syscall #Escribe el valor en $a0
17         li $a0, '\n' #Salto de línia
18         li $v0,11
19         syscall
20         jr $ra
21
22 mult10:
23         sll $t0, $a0, 3
24         sll $t1, $a0, 1
25         add $v0, $t0, $t1
26         jr $ra
27
```

Cuando realizamos un desplazamiento de por ejemplo dos posiciones, lo que hacemos es multiplicar el número que estemos desplazando por 2^2 (4), es decir multiplicamos el número por 2 elevado a la cantidad del desplazamiento. El problema es que no hay ninguna potencia de 2 que nos de 10, por lo que tenemos que buscar una combinación de dos potencias, en este caso yo he elegido 2^3 y 2^1, primero realizamos el desplazamiento de 3, lo añadimos al registro temporal 0, realizamos el desplazamiento de 1, lo añadimos al registro temporal 1 y los sumamos, almacenando la suma en el registro $v0.

C9

Cuestión 9.

- Modifica el código de tal manera que ahora lo que lea sea una cantidad de hora y muestre por consola la cantidad de segundos.

El procedimiento que tenemos que seguir es algo parecido al del ejercicio anterior, en este caso tenemos que encontrar la combinación que nos lleve a multiplicar por 3600, yo he utilizado la siguiente:

$2^{11} + 2^{10} + 2^9 + 2^4$

c9.asm* mips1.asm*

```
1  .text
2  li $a0, '>'
3  li $v0,11
4  syscall
5
6  li $v0,5
7  syscall
8
9  move $a0, $v0
10
11 jal horasToSegs
12 move $a0, $v0
13 jal imprim
14
15 li $v0,10
16 syscall
17
18 imprim:
19         addi $v0,$0,1 #función imprim
20         syscall #Escribe el valor en $a0
21         li $a0, '\n' #Salto de línia
22         li $v0,11
23         syscall
24         jr $ra
25
26 horasToSegs:
27         sll $t0, $a0, 11
28         sll $t1, $a0, 10
29         sll $t2, $a0, 9
30         sll $t3, $a0, 4
31         add $v0, $t0, $t1
32         add $v0, $t2, $v0
33         add $v0, $t3, $v0
34         jr $ra
35
```