

MEMORIA

Índice

1. PRÁCTICA 1 (p.2-4)
2. PRÁCTICA 2 (p.5)
3. PRÁCTICA 3 (p.6-8)
4. Ejercicios optativos (p.9-10)

DATOS

- Nombre: Raúl Beltrán Marco
- Email: rbm61@alu.ua.es/raulbeltmarc@gmail.com
- Grupo: lunes 12/02/19

PRÁCTICA 1

Instrucciones y Registros

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

Usando Mars obtenemos los siguientes resultados en sus respectivas casillas:

The screenshot shows the Mars MIPS simulator interface. On the left, the assembly code for 'Practica1.asm' is displayed:

```
1 #Práctica 1
2 .text 0x00400000
3 addi $t4, $zero, 5 #Introducimos el valor de 5 en el registro 12
4 addi $t2, $zero, 8 #Introducimos el valor de 8 en el registro 10
5
6 addi $t5, $t4, 10
7 addi $t2, $t2, -4
8 addi $t6, $t5, -30
9 addi $t7, $t2, 0
10
```

On the right, the 'Registers' window shows the state of the MIPS registers:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000004
\$t3	11	0x00000000
\$t4	12	0x00000005
\$t5	13	0x0000000f
\$t6	14	0xffffffff
\$t7	15	0x00000004
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400018
hi		0x00000000
lo		0x00000000

Obteniendo: $\$t7 = \$t2 = 4$ | $\$t6 = -15$ | $\$t4 = 5$ | $\$t5 = 15$

Al realizar el mismo código, pero con la orden addiu, podemos observar que obtenemos el mismo resultado anteriormente mostrado:

The screenshot shows a MIPS assembler interface. On the left, the assembly code is displayed in a window titled 'Practica1_2.asm'. The code is as follows:

```
1 #Práctica 1
2 .text 0x00400000
3 addiu $t4, $zero, 5 #Introducimos el valor de 5 en el registro 12
4 addiu $t2, $zero, 8 #Introducimos el valor de 8 en el registro 10
5
6 addiu $t5,$t4,10
7 addiu $t2,$t2,-4
8 addiu $t6,$t5,-30
9 addiu $t7,$t2,0
10
```

On the right, the 'Registers' window shows the state of the MIPS registers. The registers are organized into columns: Name, Number, and Value. The registers are listed from \$zero to \$lo. The values are as follows:

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	4
\$t3	11	0
\$t4	12	5
\$t5	13	15
\$t6	14	-15
\$t7	15	4
\$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
\$ra	31	0
pc		4194328
hi		0
lo		0

El código de operación resultante después de realizarlo con addiu es:

The screenshot shows a MIPS assembler interface. On the left, the assembly code is displayed in a window titled 'Text Segment'. The code is as follows:

```
1 #Práctica 1
2 .text 0x00400000
3 addiu $t4, $zero, 5 #Introducimos el valor de 5 en el registro 12
4 addiu $t2, $zero, 8 #Introducimos el valor de 8 en el registro 10
5
6 addiu $t5,$t4,10
7 addiu $t2,$t2,-4
8 addiu $t6,$t5,-30
9 addiu $t7,$t2,0
```

On the right, the 'Text Segment' window shows the resulting machine code. The code is organized into columns: Bkpt, Address, Code, Basic, and Source. The code is as follows:

Bkpt	Address	Code	Basic	Source
	0x00400000	0x200c0005	addi \$12,\$0,5	3: addi \$t4, \$zero, 5 #Introd...
	0x00400004	0x200a0008	addi \$10,\$0,8	4: addi \$t2, \$zero, 8 #Introd...
	0x00400008	0x218d000a	addi \$13,\$12,10	6: addi \$t5,\$t4,10
	0x0040000c	0x214afffc	addi \$10,\$10,-4	7: addi \$t2,\$t2,-4
	0x00400010	0x21aeffe2	addi \$14,\$13,-30	8: addi \$t6,\$t5,-30
	0x00400014	0x214f0000	addi \$15,\$10,0	9: addi \$t7,\$t2,0

Podemos ver claramente que el código corresponde con el orden con el cual los hemos realizado:

$\$12=5 \rightarrow 0x200c0005$

$\$10=8 \rightarrow 0x200a0008$

$\$13=\$12 + 10 \rightarrow 0x218d000a$

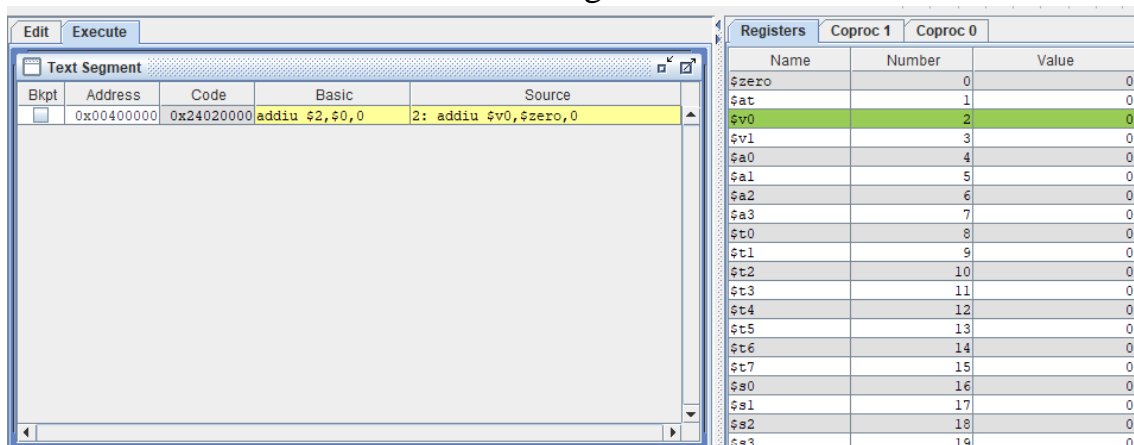
$\$10=\$10 - 4 \rightarrow 0x214afffc$

$\$14=\$13 - 30 \rightarrow 0x21aeffe2$

$\$15=\$10 \rightarrow 0x214f0000$

Para codificar en binario: `addiu $v0, $zero, 1`

Hacemos uso de Mars obteniendo el siguiente resultado:



Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$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

Hacemos la conversión a decimal obteniendo:

$0x24020000 \rightarrow 0010\ 0100\ 0010\ 0000\ 0000$

Aritmética de Enteros, operaciones lógicas y Entrada/Salida

Realizando los respectivos cambios obtenemos el resultado deseado:

Mars Messages	Run I/O
	0x0000cafe -- program is finished running (dropped off bottom) --

[illegible]

PRÁCTICA 3

Aritmética de Enteros, operaciones lógicas y pseudoinstrucciones

Escribe un programa que lea del teclado una letra en mayúscula y la escriba en minúscula en la consola.

The screenshot shows the Mars MIPS simulator interface. The top tab is 'Práctica3_1.asm'. The assembly code is as follows:

```
1 .text
2
3     li $v0, 12 #Leer el caracter
4     syscall
5     move $t0, $v0 #Mover el valor de v0 a t0
6     li $a0, '\n' #Salto de linea
7     li $v0, 11
8     syscall
9     move $a0, $t0
10    addi $a0, $a0, 32
11    li $t0, 11 #Imprimir el caracter
12    syscall
13    li $v0, 10 #Acabar el proceso
14    syscall
15
```

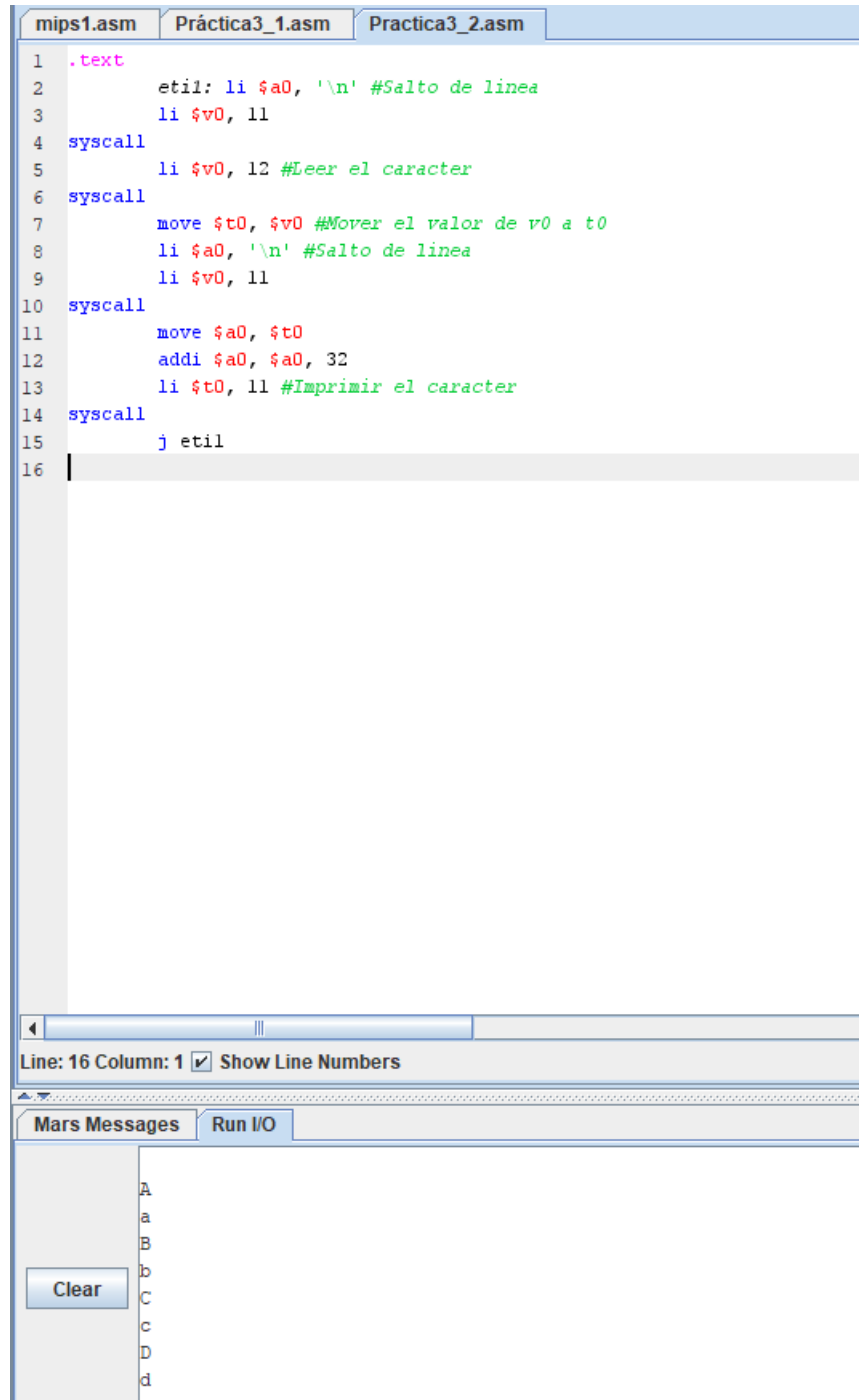
Below the code editor, the status bar shows 'Line: 15 Column: 1' and a checked 'Show Line Numbers' option. The bottom panel is divided into 'Mars Messages' and 'Run I/O'. The 'Run I/O' tab is active, showing the output of the program:

```
A
a
-- program is finished running --
Z
z
-- program is finished running --
```

A 'Clear' button is located to the left of the output text.

Itera el código que acabas de escribir.

Para iterarlo hacemos uso de las etiquetas y el bucle de tipo J:



The screenshot shows the MARS MIPS simulator interface. At the top, there are three tabs: 'mips1.asm', 'Práctica3_1.asm', and 'Practica3_2.asm'. The 'Práctica3_1.asm' tab is selected. The main window displays assembly code with line numbers from 1 to 16. The code is as follows:

```
1 .text
2     etil: li $a0, '\n' #Salto de linea
3         li $v0, 11
4     syscall
5         li $v0, 12 #Leer el caracter
6     syscall
7         move $t0, $v0 #Mover el valor de v0 a t0
8         li $a0, '\n' #Salto de linea
9         li $v0, 11
10    syscall
11    move $a0, $t0
12    addi $a0, $a0, 32
13    li $t0, 11 #Imprimir el caracter
14    syscall
15    j etil
16
```

Below the code editor, there is a status bar showing 'Line: 16 Column: 1' and a checkbox for 'Show Line Numbers' which is checked. At the bottom, there are two tabs: 'Mars Messages' and 'Run I/O'. The 'Mars Messages' tab is selected, and it shows a list of messages: A, a, B, b, C, c, D, d. There is a 'Clear' button next to the list.

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

The screenshot shows the MARS MIPS simulator interface. At the top, there are tabs for 'mips1.asm', 'Práctica3_1.asm', 'Practica3_2.asm', and 'Practica3_3.asm'. The main window displays assembly code for 'Practica3_1.asm' with line numbers 1 through 16. The code is as follows:

```
1 .text
2     etil: li $a0, '\n' #Salto de linea
3         li $v0, 11
4     syscall
5         li $v0, 12 #Leer el caracter
6     syscall
7         move $t0, $v0 #Mover el valor de v0 a t0
8         li $a0, '\n' #Salto de linea
9         li $v0, 11
10    syscall
11    move $a0, $t0
12    addi $a0, $a0, -48
13    li $v0, 1 #Imprimir entero
14    syscall
15    j etil
16
```

Below the code editor, there is a status bar showing 'Line: 16 Column: 1' and a checked box for 'Show Line Numbers'. At the bottom, there are tabs for 'Mars Messages' and 'Run I/O'. The 'Run I/O' tab is active, showing a list of input/output operations. The list contains the following entries:

- 0
- 0
- 4
- 4
- 5
- 5
- 9
- 9

There is a 'Clear' button to the left of the list.

Ejercicios optativos

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

```
.text
addi $v0,$0,5 #Función 5, leer el número
syscall #Valor leído en $v0

addi $a0,$v0,0 #Movemos el valor leído a $a0
addi $a0, $a0, 1
addi $v0,$0,1 #Función 1, imprimir el número
syscall #Escribimos en consola $a0

addi $v0,$0,10 #Función 10, exit
syscall #Acaba el programa
```

4
5
-- program is finished running --

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

```
.text
addi $v0,$0,5 #Función 5, leer el número
syscall #Valor leído en $v0

addi $a0,$v0,0 #Movemos el valor leído a $a0
addi $a0, $a0, -1
addi $v0,$0,1 #Función 1, imprimir el número
syscall #Escribimos en consola $a0

addi $v0,$0,10 #Función 10, exit
syscall #Acaba el programa
```

6
5
-- program is finished running --

Ejemplo de cómo hacer que lea un número introducido y luego lo devuelva después de usar un salto de línea ‘\n’

```
1 # Programa ECO
2 .text
3     li $v0,12 #Función 12. Read character
4     syscall #Carácter leído en $v0
5     move $t0, $v0 #Guardamos el valor en $t0 para no machacarlo
6     li $a0, '\n'
7     li $v0, 11
8     syscall
9     move $a0,$t0 #Carácter a escribir en $a0
10    li $v0,11 #Función 11. Print character
11    syscall
12    li $v0, 10 #Función 10. Acaba programa
13    syscall
```

ne: 14 Column: 1 ☒ Show Line Numbers

Mars Messages Run I/O

3
3
-- program is finished running --

Clear

L
L
-- program is finished running --

Ejecuta el código y comprueba el valor correcto de las etiquetas

```
1 # Pruebas etiquetas
2 .text
3     eti1: addi $v0,$0,5
4     syscall
5     eti2: addi $a0,$v0,15
6     addi $v0,$0,1
7     syscall
8
9     li $a0, '\n'
10    li $v0,11
11    syscall
12
13    la $t1, eti1 #Carga en $t1 la dirección de eti1
14    la $t2, eti2 #Carga en $t2 la dirección de eti2
15    move $a0,$t1
16    li $v0,34
17    syscall
18    li $a0, '\n'
19    li $v0,11
20    syscall
21    move $a0,$t2
22    li $v0,34
23    syscall
24    li $v0,10 #Función 10 Exit syscall
25    syscall
```

e: 26 Column: 1 ☒ Show Line Numbers

Mars Messages Run I/O

5
20
0x00400000
0x00400008
-- program is finished running --

¿Qué hace el código? → Lee un número y le suma 15, además muestra en hexadecimal el contenido de los registros \$t1 y \$t2

¿En qué instrucciones básicas se ha traducido la pseudoinstrucción la? → En lui y ori