Estructura de los computadores

Practica 1,2,3

Francisco Joaquín Murcia Gómez 48734281H

Grupo 3

Índice

Practica 1

Ejemplos

1. La ventana Registers

1.1. Probad a modificar el contenido de algún registro. Notad que podéis escribir en decimal o hexadecimal y que no podéis cambiar $0, $31 ni $pc.
$0,$31 y $pc no se pueden modificar

1.2. Probad a escribir valores negativos en los registros.
El -4 →0xFFFFFFFc (esta en complemento a2)

1.3. ¿Cuál es el mayor positivo que puede contener un registro del MIPS?
0x7FFFFFFF

1.4. ¿Cuál es el mayor negativo que puede contener un registro del MIPS?
0x80000000

2. El primer programa – análisis

2.1. ¿Cómo se codifica la instrucción addi $10,$8,5?

addi	$8	$10	k
00100	01000	01010	0000000000000101

0x210A0005

4. La ventana text segment

4.1. ¿En qué dirección se almacena cada instrucción del programa?
En adres

4.2. ¿Qué vale el PC?
0x00400000, $pc apunta a adres

5. El cicló de instrucción

5.1. Dad el siguiente valor inicial a $8 = 0x7FFFFFFF y ejecutad de nuevo el programa paso a paso fijándoos en la ventana Mars Messages. ¿Qué ha ocurrido?
Que hay overflow, 0x7FFFFFFF es el valor más grande y al sumarle 5 se sale del rango de representacion

5.2. ¿Cuál es el valor más grande que podrá contener $8 para que no se aborte el programa?
0x7FFFFFFA

6. Usos alternativos de addi

6.1. Modificad el programa para dar un valor inicial al registro $8 utilizando addi.

```
.text 0x00400000
addi $8,$0,12
```

$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x0000000c
$t1	9	0x00000000
$t2	10	0x00000000

6.2. Añadid una instrucción para que el resultado final se encuentre en $12

```
.text 0x00400000
addi $8,$0,12
addi $12,$8,0
```

$t0	8	0x0000000c
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x0000000c

6.3. ¿Se podría utilizar la instrucción addi para hacer una resta?

Si, addi $X,$Y,-k

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

addi $8,$8,-1

001000 01000 01000 1111111111111110

Entregas

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

mips1.asm

```
1  .text 0x00400000
2  addi $12,$0,5
3  addi $10,$0,8
4  addi $13,$12,10
5  addi $10,$10,-4
6  addi $14,$13,-30
7  addi $15,$10,0
```

$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	0xfffffff1
$t7	15	0x00000004
$s0	16	0x00000000
$s1	17	0x00000000

2. **¿Se podría escribir el mismo código utilizando la instrucción addiu? Haz la prueba.**

   Si, se podría:

mips1.asm

```
1  .text 0x00400000
2  addiu $12,$0,5
3  addiu $10,$0,8
4  addiu $13,$12,10
5  addiu $10,$10,-4
6  addiu $14,$13,-30
7  addiu $15,$10,0
```

$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	0xfffffff1
$t7	15	0x00000004
$s0	16	0x00000000
$s1	17	0x00000000

3. **¿Cuál es el código de operación de la instrucción addiu?**

   001001

4. **Codifica en binario la instrucción addiu $v0, $zero, 1.**

addiu	$zero	$v0	k
001001	00000	00010	0000000000000001

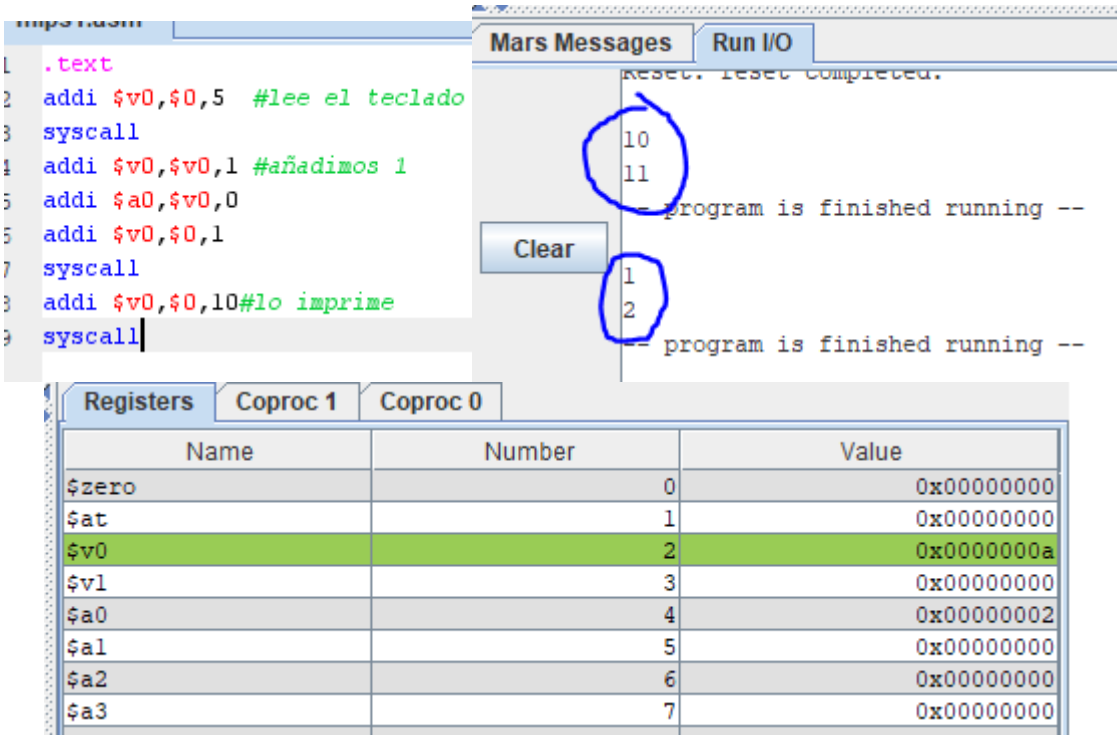
00100100000000100000000000000001

Practica 2

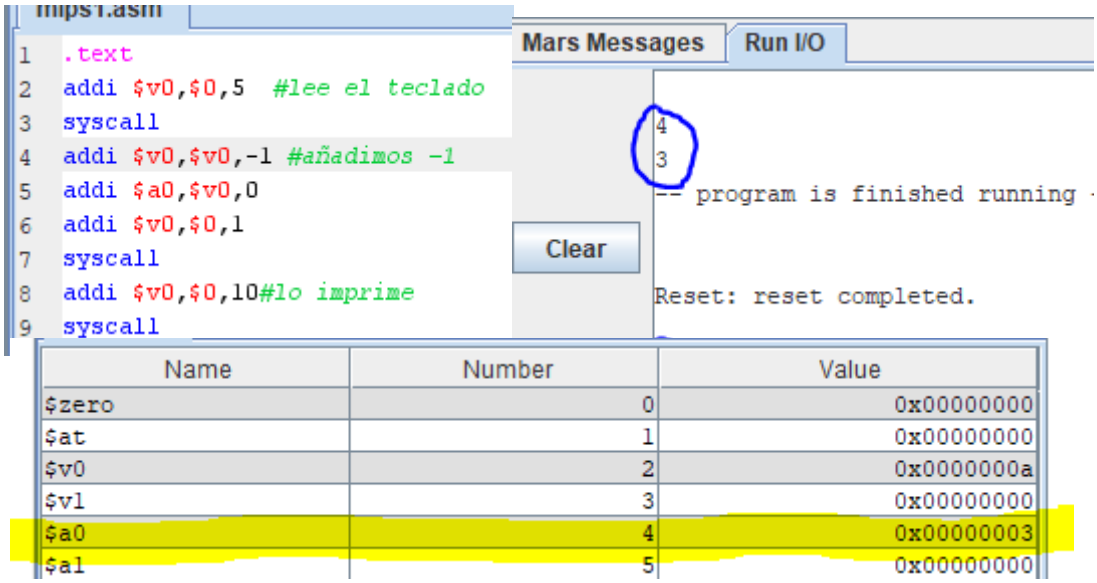
Ejemplos

1. Input y Output

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



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



2. El conjunto de instrucciones: Aritmética de enteros

2.1. ¿Qué hace cada línea de código de partida?

mips1.asm*

```
1
2    .text
3    addiu $t0, $zero, 25
4   # pone el valor 25 al registro $t0 (0+25=25)
5    addiu $t1, $zero, 5
6     # pone el valor 5 al registro $t (0+5=5)
7    sub $t2,$t0,$t1
8    # $t0 - $t1 =$t2 = 20
9    addi $v0, $zero, 10 #Salir
10   syscall
```

2.2. ¿En qué dirección de memoria se almacena la instrucción sub?

Text Segment

Bkpt	Address	Code	Basic
	0x00400000	0x24080019	addiu $8,$0,25
	0x00400004	0x24090005	addiu $9,$0,5
	0x00400008	0x01095022	sub $10,$8,$9
	0x0040000c	0x2002000a	addi $2,$0,10
	0x00400010	0x0000000c	syscall

2.3. Ensambla y ejecuta el código. ¿Cuál es el valor final del registro $t2?

$a3	7	0
$t0	8	25
$t1	9	5
$t2	10	20
$t3	11	0

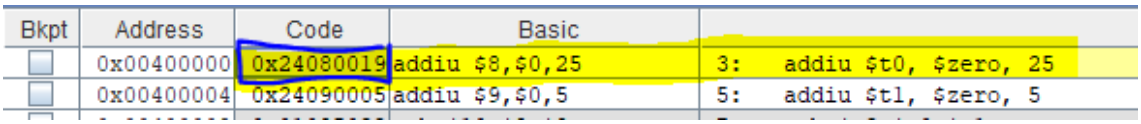
3. Introducción al formato de instrucciones del MIPS: código máquina.

3.1. Observa el código de partida Aritmética de enteros del apartado anterior. ¿Cómo se codifica la primera instrucción? Hacedlo a mano (el código de operación de la instrucción addiu es 0x09)

addiu	$zero	$t0	k
001001	00000	01000	0000000000011001

00100100000010000000000000011001

3.2. Confirmad con el programa ensamblado que el código máquina es el mismo.



Bkpt	Address	Code	Basic	
	0x00400000	0x24080019	addiu $8,$0,25	3: addiu $t0, $zero, 25
	0x00400004	0x24090005	addiu $9,$0,5	5: addiu $t1, $zero, 5

En binario seria: 00100100000010000000000000011001

3.3. ¿Cómo se codifica la última instrucción de resta del código de partida Aritmética de enteros que acabamos de escribir? Hacedlo a manos (el campo función de la resta es 0x22)

Operación	Rs	Rt	Rd	shamt	Función
000000	01000	01001	01010	00000	100010

000000 01000 01001 01010 00000 10001º

3.4. Notad que hay 64 instrucciones distintas con formato tipo R. ¿Por qué?

Son 64 porque el campo de la función es de 6 bits 6^2=64.

3.7. A la vista del código escrito, ¿es necesario que forme parte del repertorio de instrucciones la instrucción subi?

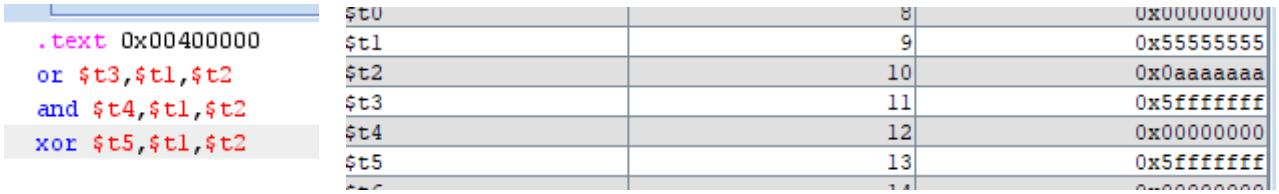
No, se implementa con addi

4. .Repertorio de instrucciones: instrucciones lógicas

4.1. ¿Podríamos utilizar la instrucción lógica ori para dar un valor inicial a un registro en lugar de la instrucción addi?

Si

4.7. Escribe el código que haga la operación lógica OR de $t1 y $t2 y lo guarde en $t3, la operación lógica AND de $t1 y $t2 y lo guarde en $t4, y la operación lógica XOR de $t1 y $t2 y lo guarde en $t5. Escribe en la ventana de registros, tras ensamblarlo, los siguientes valores para los registros $t1=0x55555555 y $t2= 0xAAAAAAA. Ejecuta el código y estudia los resultados



```
.text 0x00400000
or $t3,$t1,$t2
and $t4,$t1,$t2
xor $t5,$t1,$t2
```

$t0	8	0x00000000
$t1	9	0x55555555
$t2	10	0x0aaaaaaa
$t3	11	0x5fffffff
$t4	12	0x00000000
$t5	13	0x5fffffff

4.8. 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.

```
1  .text 0x00400000
2  or $t2,$t1,$t0
3  ori $t2,$t2,0x000000F0
4  andi $t2,$t2,0x0000cFFF
```

$t0	8	0x00000000
$t1	9	0x0000face
$t2	10	0x0000cafe
$t3	11	0x00000000

Entregas

1. **Modifica el código del último ejercicio del apartado 4 para que aparezca en la pantalla el contenido del registro $t2=0000CAFE.**

```
.text 0x00400000
or $t2,$t1,$t0
ori $t2,$t2,0x000000f0
andi $t2,$t2,0x0000cfff
ori $a0,$t2,0
ori $v0,$0,34
syscall
```

```
Reset: reset completed

Reset: reset completed

0x0000cafe
-- program is finished
```

$t0	8	0x00000000
$t1	9	0x0000face
$t2	10	0x0000cafe
$t3	11	0x00000000

2. **Escribe el código que lee un valor entero por teclado y escribe el mismo valor en binario por la consola.**

```
.text 0x00400000
addi $v0,$0,5
syscall
#lee teclado
addi $a0,$v0,0
addi $v0,$0,35
#funcion para que
#aparezca en binario
syscall
#imprime
```

```
-- program is finished running (dropped off bottom)

6
00000000000000000000000000000110
-- program is finished running (dropped off bottom)
```

Aparecen tantos 0 porque está en 64 bits

$v1	3	0x00000000
$a0	4	0x00000006
$a1	5	0x00000000

Practica 3

Ejemplos

1. **Desbordamiento e instrucciones insignes**
   - **1.1. ¿Cuál es el mayor valor positivo que puede contener un registro del MIPS?.**

     0x7FFFFFFF
   - **1.2. Escribe un programa de una instrucción que sume $t0=$t1+$t2 y ensámblalo**

     or $t0,$t1,$t2
   - **1.4. Ensambla y ejecuta el código fijándote en la ventana Mars Missatges y explica lo que ha pasado.**

     Overflow,, 0x7FFFFFFF es el máximo numero positivo
   - **1.5. Sustituye add por addu y vuelve a hacer la prueba. Explica el resultado.**

     addu $t0,$t1,$t2 la operación se hace sin signo

2. **Pseudoinstrucciones**

   - **2.1. Comprueba como se traducen las siguientes pseudoinstrucciones al ensamblar el programa. Mira la columna de la izquierda del código nombrado Basic.**

     Addiu $9,$0,4
     Addu $10,$0,$9
     Nor $1,$10,$0
     Addi $1,$0,1
     Sub $12,$9,$1

3. **Constantes grandes**
   - **3.1. ¿Qué hace la instrucción lui? Buscad en la web o en la ayuda de las instrucciones básicas del MARS.**

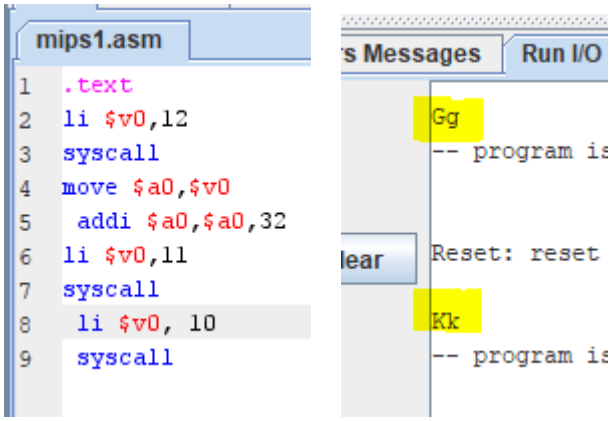
     Asigna datos a un registro
   - **3.2. ¿Cómo haríais $t0=0x10000000?**

     lui $t0,0x1
   - **3.3. ¿Cómo haríais $t1=0x10001000?**

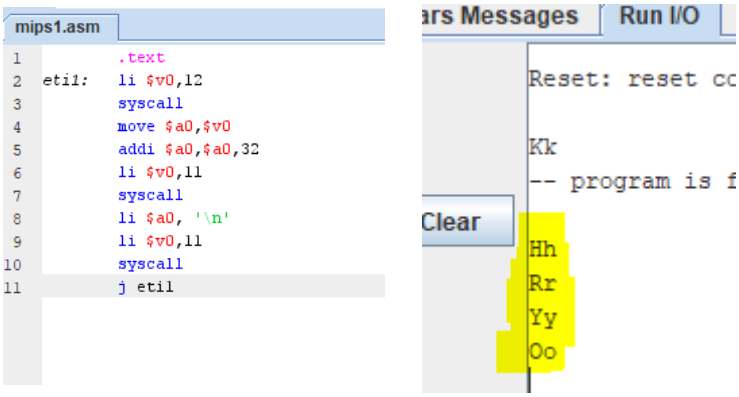
     lui $t0,0x100001

Entregas

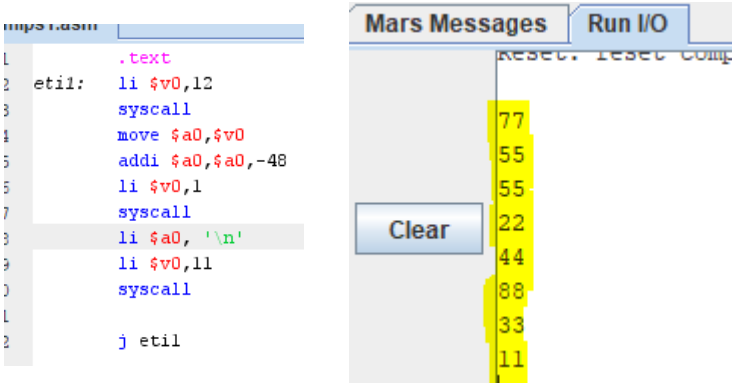
1. **Escribe un programa que lea del teclado una letra en mayúscula y la escríba en minúscula en la consola.**



2. **Itera el código que acabas de escribir**



3. **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.**



el numero de la izquierda es tipo carácter y el de la derecha numero

En el ejemplo del 4:

$v0	2	0x00000034
$v1	3	0x00000000
$a0	4	0x00000034
$a1	5	0x00000000

$a0 toma el valor de 4 en ascii(52)

$v1	3	0x00000000
$a0	4	0x00000004
$a1	5	0x00000000

Se le resta 48 para obtener el numero 4