



Práctica 1 - solución año 2018/19

Estructura De Los Computadores (Universidad de Alicante)



Escanea para abrir en Studocu

Práctica 1 – Instrucciones y registros

1. La ventana *Registers*

➤ 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.

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00003d50
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000

➤ Probad a escribir valores negativos en los registros

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00003d50
\$v1	3	0xffffffffd3
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000

Podemos ver que se escribe su codificación hexadecimal de su forma binaria en complemento a 2

➤ ¿Cuál es el mayor positivo que puede contener un registro del MIPS? Basta con que lo digas en hexadecimal.

El mayor positivo que se puede representar en decimal es el 2147483648 (que viene de dividir 2^{32} entre 2, el total de números con representación en complemento a 2 positivos y negativos con 32 bits), en hexadecimal es 7FFFFFFF_H.

➤ ¿Cuál es el mayor negativo que puede contener un registro del MIPS? Basta con que lo digas en hexadecimal.

El mayor negativo que se puede representar es el 80000000_H.

2. El primer programa – análisis

➤ ¿Cómo se codifica la instrucción `addi $10,$8,5`? Escribid el código resultante en hexadecimal.

0x210a0005 en hexadecimal

0010 0001 0000 1010 0000 0000 0000 0101 en C2

3. Ensamblado

`Assemble: operation completed successfully.`

4. La ventana *text segment*

➤ ¿En qué dirección se almacena cada instrucción del programa?

00400000_H y 00400004_H

➤ Comprobad la codificación de las instrucciones en código máquina.

21090019_H y 210a0005_H

➤ (En la ventana de registros) ¿Qué vale el PC?

00400000_H

5. El ciclo de instrucción

➤ 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?

`Error in C:\Users\Guillermo\Desktop\Universidah\Estructura de los computadores\Práctica\mips2.asm line 3: Runtime exception at 0x0040000c: arithmetic overflow`

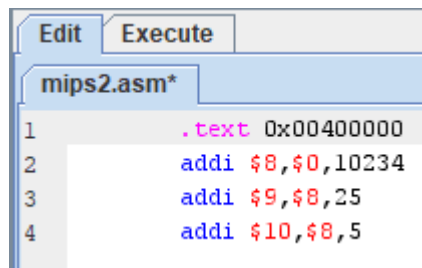
`Step: execution terminated with errors.`

➤ ¿Cuál es el valor más grande que podrá contener `$8` para que no se aborte el programa?

7FFFFFFD_H

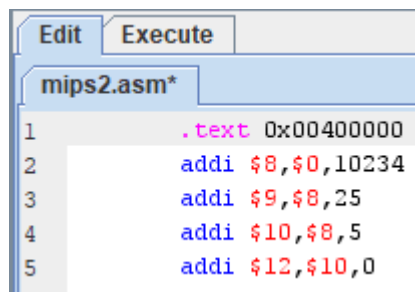
6. Usos alternativos de addi

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



```
1      .text 0x00400000
2      addi $8,$0,10234
3      addi $9,$8,25
4      addi $10,$8,5
```

- Añadid una instrucción para que el resultado final se encuentre en \$12



```
1      .text 0x00400000
2      addi $8,$0,10234
3      addi $9,$8,25
4      addi $10,$8,5
5      addi $12,$10,0
```

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

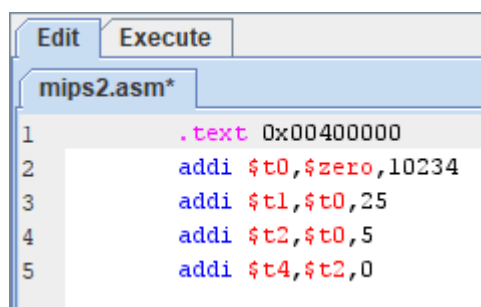
Escribiendo un numero negativo se realiza una resta.

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

```
addi $8,$8,-1
```

7. Ayudas a la programación. Nombres alternativos de los registros.

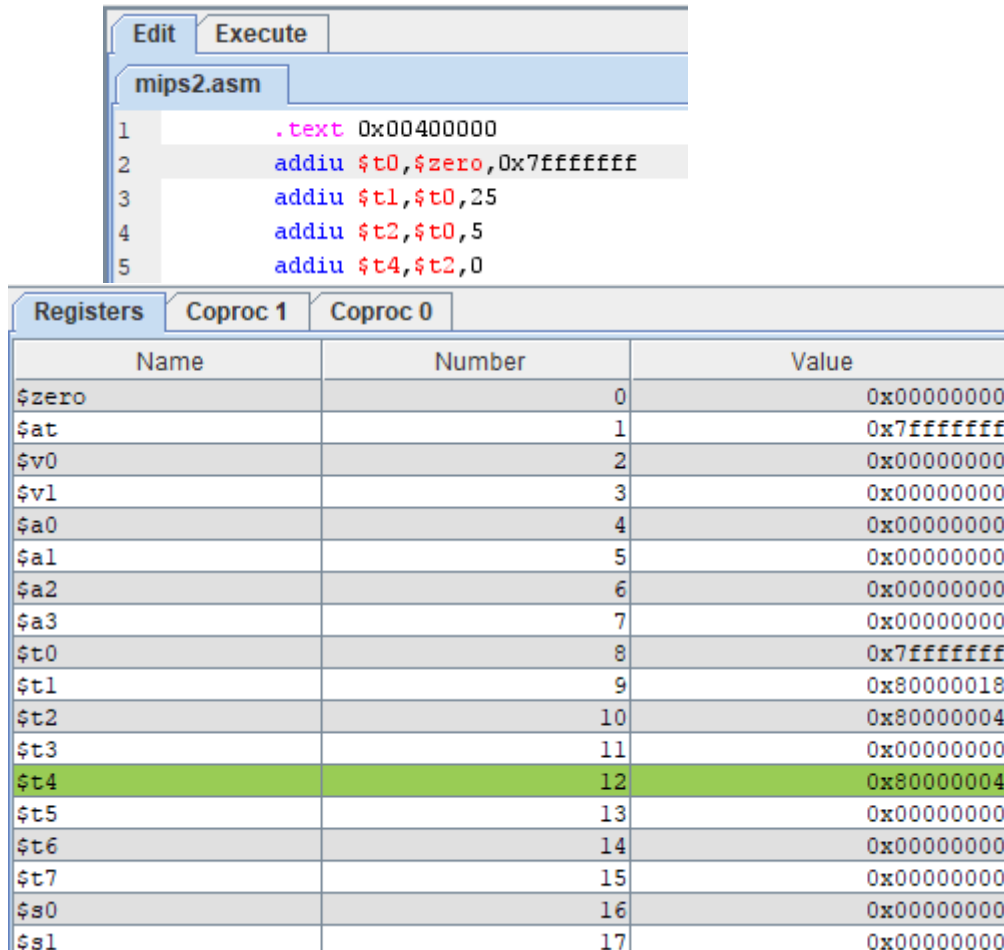
- Reescribid el programa anterior utilizando el convenio de registros y vuelve a ejecutarlo.



```
1      .text 0x00400000
2      addi $t0,$zero,10234
3      addi $t1,$t0,25
4      addi $t2,$t0,5
5      addi $t4,$t2,0
```

8. Más sobre la suma inmediata.

➤ Volved a escribir el programa cambiando *addi* or *addiu* y dad como valor inicial de \$t0 el positivo más grande posible (\$t0 = 0x7FFFFFFF) y ejecutadlo observando el contenido de \$t0 en hexadecimal y en decimal. ¿Qué ha ocurrido?



The screenshot shows a MIPS assembler interface. The top part is a code editor with tabs for 'Edit' and 'Execute'. The file is named 'mips2.asm'. The code contains five lines of assembly instructions:

```
1 .text 0x00400000
2 addiu $t0,$zero,0x7fffffff
3 addiu $t1,$t0,25
4 addiu $t2,$t0,5
5 addiu $t4,$t2,0
```

The bottom part is a register window with tabs for 'Registers', 'Coproc 1', and 'Coproc 0'. It displays a table of registers with their names, numbers, and values.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x7fffffff
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x7fffffff
\$t1	9	0x80000018
\$t2	10	0x80000004
\$t3	11	0x00000000
\$t4	12	0x80000004
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000

Los números de los registros \$t1, \$t2 y \$t3 pasan del valor máximo, pues la instrucción *addiu* no ha tenido en cuenta el signo.

➤ Si el programador considera que está operando con número naturales, el resultado que hay en \$t0 ¿sería correcto? ¿Cuál sería su valor en decimal?

Sí lo sería, y su valor decimal sería 2147483647₁₀

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

\$12=5

addi \$t4,\$zero,5

\$10= 8

addi \$t2,\$zero,8

\$13=\$12 + 10

```

        addi $t5,$t4,10
$10=$10 - 4
        addi $t2,$t2,-4
$14=$13 - 30
        addi $t6,$t5,-30
$15=$10
        addi $t7,$t2,0

```

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

Registers	Coproc 1	Coproc 0	
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	0x7ffffc	
\$fp	30	0x00000000	
\$ra	31	0x00000000	
pc		0x00400018	
hi		0x00000000	
lo		0x00000000	

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

Sí se puede.

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

0x24000000

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

0010 0100 0000 0010 0000 0000 0000 0001