**Estructura de los computadores**

Practica 1,2,3

Francisco Joaquín Murcia Gómez 48734281H

Grupo 3

**Índice**

**Practica 1**…………………………………………………………………………3-5

Instrucciones y registros

Ejemplos …………………………………………………………………3-4

Entregas ……………………………………………………………………...5

**Practica 2**……………………………………………………………………………6

Aritmética de enteros (1), operaciones lógicas y Entrada/Salida

Ejemplos …………………………………………………………………6-9

Entregas ………………………………………………………………….10

**Practica 3** …………………………………………….

Aritmética de enteros (2) y pseudoinstrucciones

Ejemplos …………………………………………….

Entregas …………………………………………….

**Practica 1**

**Ejemplos**

1. **La ventana Registers**
   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. **Probad a escribir valores negativos en los registros.**

El -4 🡪0xFFFFFFFc (esta en complemento a2)

* 1. **¿Cuál es el mayor positivo que puede contener un registro del MIPS?**

0x7FFFFFFF

* 1. **¿Cuál es el mayor negativo que puede contener un registro del MIPS?**

0x80000000

1. **El primer programa – análisis**
   1. **¿Cómo se codifica la instrucción addi $10,$8,5?**

0x210A0005

1. G
2. **La ventana text segment**
   1. **¿En qué dirección se almacena cada instrucción del programa?**

En adres

* 1. **¿Qué vale el PC?**

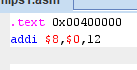
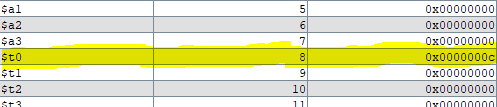
0x00400000, $pc apunta a adres

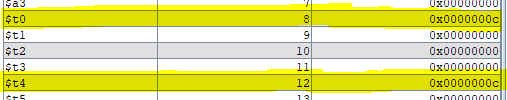
1. **El cicló de instrucción**
   1. **Dad el siguiente valor inicial a $8 = 0x7FFFFFFF y ejecutad de nuevo el programa paso a paso fijándoos en la ventana Mars Messages. ¿Qué ha ocurrido?**

Que hay overflow, 0x7FFFFFFF es el valor más grande y al sumarle 5 se sale del rango de representacion

* 1. **¿Cuál es el valor más grande que podrá contener $8 para que no se aborte el programa?**

0x7FFFFFFA

1. **Usos alternativos de addi**
   1. **Modificad el programa para dar un valor inicial al registro $8 utilizando addi.**
   2. **Añadid una instrucción para que el resultado final se encuentre en $12**



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

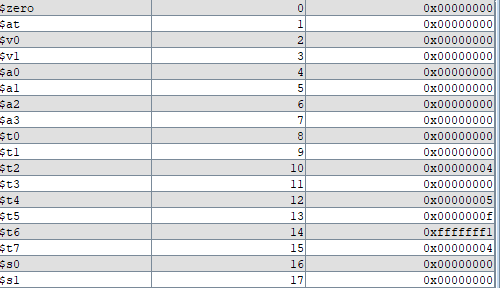
Si, addi $X,$Y,-k

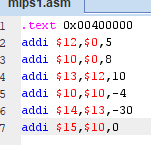
* 1. **¿Cómo se escribe la instrucción que hace $8 = $8-1? ¿Cómo quedaría su codificación en binario?**

addi $8,$8,-1

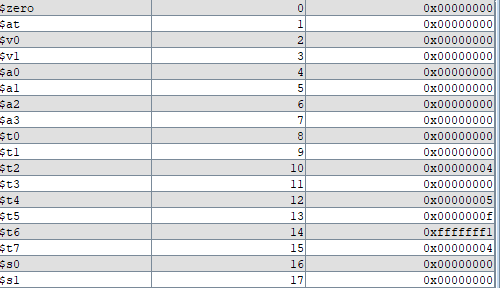
001000 01000 01000 1111111111111110

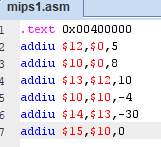
**Entregas**

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



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

Si, se podría:



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

001001

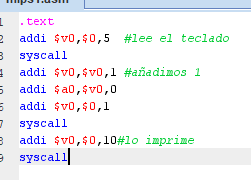
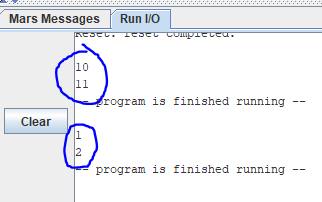
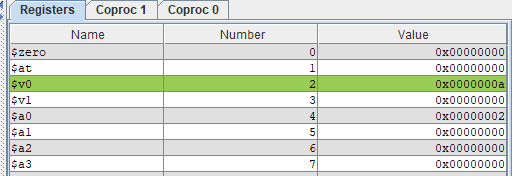
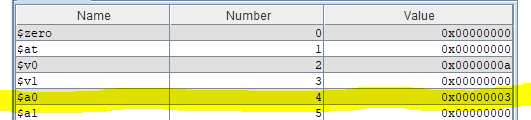
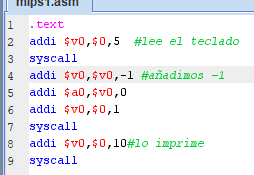
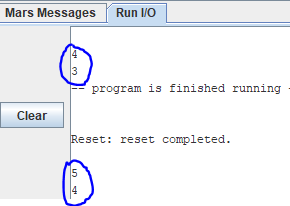
1. **Codifica en binario la instrucción addiu $v0, $zero, 1.**

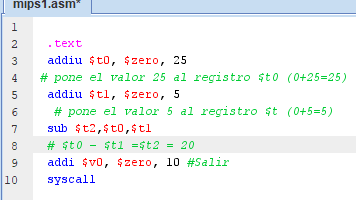
|  |  |  |  |
| --- | --- | --- | --- |
| addiu | $zero | $v0 | k |
| 001001 | 00000 | 00010 | 0000000000000001 |

00100100000000100000000000000001

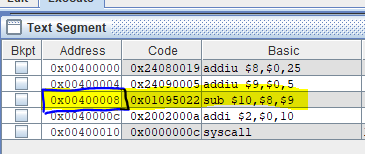
**Practica 2**

**Ejemplos**

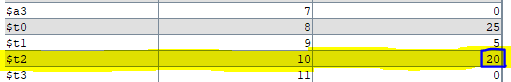
1. **Input y Output**
   1. **Haz un código que lee un valor x de teclado y escribe x+1 en la consola.**
   2.  **Haz un código que lee un valor x de teclado y escribe x-1 en la consola.**
2. **El conjunto de instrucciones: Aritmética de enteros**
   1. **¿Qué hace cada línea de código de partida?**



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



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



1. **Introducción al formato de instrucciones del MIPS: código máquina.**
   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

* 1. **Confirmad con el programa ensamblado que el código máquina es el mismo.**



En binario seria: 00100100000010000000000000011001

* 1. **¿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

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

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

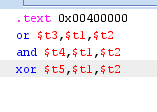
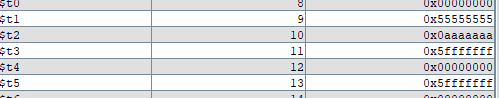
* 1. **A la vista del código escrito, ¿es necesario que forme parte del repertorio de instrucciones la instrucción subi?**

No, se implementa con addi

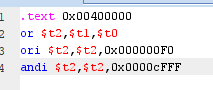
1. **.Repertorio de instrucciones: instrucciones lógicas**
   1. **¿Podríamos utilizar la instrucción lógica ori para dar un valor inicial a un registro en lugar de la instrucción addi?**

Si

* 1. **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**

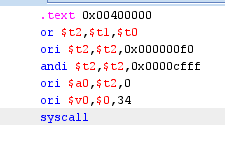
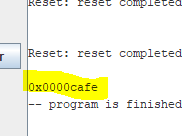


* 1. **Supón que $t1=0x0000FACE, utilizando únicamente las instrucciones lógicas de la tabla anterior, escribe el código que reordene los bits de $t1 de manera que en $t2 aparezca el valor 0x0000CAFE. Ensambla y escribe en la ventana de registros $t1=0x0000FACE. Ejecuta y comprueba que el código es correcto.**



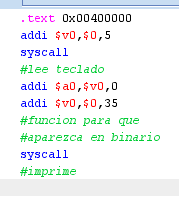
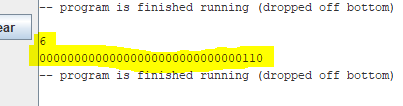


**Entregas**

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



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



Aparecen tantos 0 porque está en 64 bits



**Practica 3**

**Ejemplos**

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

0x7FFFFFFF

* 1. **Escribe un programa de una instrucción que sume $t0=$t1+$t2 y ensámblalo**

or $t0,$t1,$t2

* 1. **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. **Sustituye add por addu y vuelve a hacer la prueba. Explica el resultado.**

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

1. **Pseudoinstrucciones**
   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

1. **Constantes grandes**
   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

* 1. **¿Cómo haríais $t0=0x10000000?**

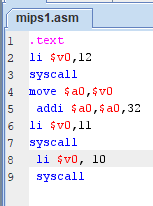
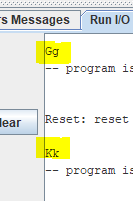
lui $t0,0x1

* 1. **¿Cómo haríais $t1=0x10001000?**

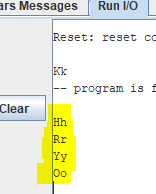
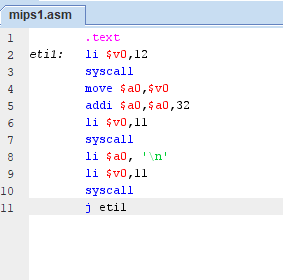
lui $t0,0x100001

**Entregas**

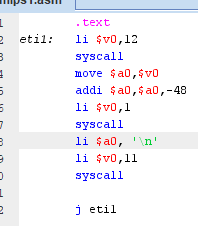
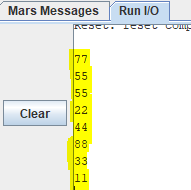
1. **Escribe un programa que lea del teclado una letra en mayúscula y la escríba en minúscula en la consola.**



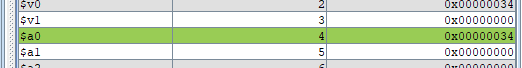
1. **Itera el código que acabas de escribir**



1. **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:

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



Se le resta 48 para obtener el numero 4