Alejandro Martínez Cedillo – 74393989Y

***Práctica 1 Estructura de los Computadores***

*a) 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.***

He cambiado la posición 8 de memoria por un valor de 10 en decimal o escrito en notación Hexadecimal **0x0000000a.**

*b) Probad a escribir valores negativos en los registros.*

En este caso he cambiado la posición 5 de memoria por un valor de -5 en decimal o escrito en notación Hexadecimal **0xffffffffb**.

*c) ¿Cuál es el mayor positivo que puede contener un registro del MIPS?*

El mayor número positivo que se puede introducir en los registros del CPU de MIPS es el **0x7fffffff** en notación Hexadecimal o **2147483647** en decimal.

*d) ¿Cuál es el mayor negativo que puede contener un registro del MIPS?*

El menor número que se puede introducir en los registros del CPU de MIPS es el **0x80000000** en notación Hexadecimal o **-2147483648** en decimal.

*e) ¿Cómo se codifica la instrucción* ***addi $10, $8, 5****? Escribid el código resultante en Hexadecimal.*

En notación Hexadecimal la anterior instrucción se traduce como **0x210a0005**.

*f) ¿En qué dirección se almacena cada instrucción del programa?*

La instrucción inicial dada es la **0x00400000** por lo tanto la dirección en la que se va a almacenar cada instrucción es esa sumándole 4 cada vez que pasemos a la siguiente, por ejemplo:

-La primera instrucción se encuentra almacenada en la dirección **0x00400000** y la segunda instrucción se le suma 4 yendo a la dirección **0x00400004***.*

*g) Comprobad la codificación de las instrucciones en código máquina.*

*Hexadecimal 🡪 Binario*

***Instrucción 1:*** *0x21090019 🡪 0010 0001 0000 1001 0000 0000 0001 1001*

***Instrucción 2:*** *0x210a0005 🡪 0010 0001 0000 1010 0000 0000 0000 0101*

|  |  |  |  |
| --- | --- | --- | --- |
| **Código Op (6bits)** | **Rs (5bits)** | **Rt (5bits)** | **K (16bits)** |
| 0010 00 | 01 000 | 0 1001 | 0000 0000 0001 1001 |
| 0010 00 | 01 000 | 0 1010 | 0000 0000 0000 0101 |

Como podemos observar, ambas instrucciones comparten el mismo Código Op, es decir que ambas están usando el comando “*addi*”.

*h) (En la ventana registros) ¿Qué vale el PC?*

En la ventana registros el valor del **PC** es de **0x00400008** en notación Hexadecimal o **4194312** en notación decimal.

Como ya he dicho en el apartado B la siguiente dirección es la actual más 4, por lo tanto, podemos deducir que el **PC** es la siguiente dirección en memoria.

*i) Modificad el programa para dar un valor inicial al registro $8 utilizando* ***addi****.*

Al espacio 8 de memoria se le asigna un valor de **10,** mediante la sentencia **addi $8, $0, 10.**

*j) Añadid una instrucción para que el valor final se encuentre en $12*

*Se le ha copiado el valor final de $8 a $12 mediante esta sentencia* ***addi $12, $8, 0.***

*k) ¿Se podría utilizar la instrucción addi para hacer una resta?*

Se podría ya que al hacer una suma entre un número positivo y otro negativo sería automáticamente una resta.

*l) ¿Cómo se escribe la instrucción que hace* ***$8 = $8 -1****? ¿Cómo quedaría el resultado en binario?*

Para completar esta tarea utilizaremos **addi $8, $8, -1**.

Como el resultado es **0x00000009,** al pasarse a binario es **0000 0000 0000 0000 0000 0000 0000 1001.**

*m) Reescribid el programa anterior utilizando el convenio de registros y vuelve a ejecutarlo.*

El código final del programa ha sido este:

**addi $t0, $zero, 10 #Añadimos a la posición 8 un 10 ya que se lo sumamos a la Zero**

**addi $t4, $t0, 0 #Copiamos el resultado de la posición 8 a la 12**

**addi $t0, $t0, -1 #En la posición 8 le restamos 1 a su valor**

*n) Volved a escribir el programa cambiando addi por addiu y dad como valor inicial de $t0 el positivo más grade y ejecutadlo observando el contenido de $t0 en hexadecimal y en decimal. ¿Qué ha ocurrido?*

Al inicio del programa el valor de todos los registros es 0, tras este paso ocurren dos pasos intermedios en la posición $1 o $at, en la cual el número positivo más grande no aparece, pero luego sí.

En notación Hexadecimal se puede ver claramente por qué ocurre eso, en un inicio pasa de valer 0 a **0x7fff0000** y en el siguiente paso cambia a **0x7fffffff,** en decimal el cambio es pasar de 0 a **2.147.418.112** y luego **2.147.483.647**.

Tras eso copiamos el resultado de $8 a $12 y en el siguiente paso le restamos uno a $8, quedando los valores así:

**$1 🡪 0x7fffffff / 2.147.483.647**

**$8 🡪 0x7ffffffe / 2.147.483.646**

**$12 🡪 0x7fffffff / 2.147.483.647**

*o) Si el programador considera que está operando con números naturales, ¿el resultado que hay en $t0 sería correcto? ¿Cuál sería su valor en decimal?*

No, no es correcto, como estamos trabajando solo con números naturales el ‘-1’ no debería tomarlo como uno negativo y debería restarlo en vez de sumarlo.

Al mismo tiempo no podría sumarlo porque ya es el número entero más grande, por lo tanto el valor **2.147.483.648** al ser imposible de existir en este sistema, no puede ser mostrado.

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

.text 0x00400000 #Ponemos esta dirección de memoria como la inicial

addi $t4, $zero, 5 #Sumamos 5 a la posición $zero y lo introducimos en la $t4

addi $t2, $zero, 8 #Sumamos 8 a la posición $zero y lo introducimos en la $t2

addi $t5, $t4, 10 #Sumamos 10 a la posición $t4 y lo introducimos en la $t5

addi $t2, $t2, -4 #Sumamos '-4' a la posición $t2

addi $t6, $t5, -30 #Sumamos '-30' a la posición $t5 y lo introducimos en la $t6

addi $t7, $t2, 0 #Copiamos la información de la $t2 y lo introducimos en la $t7

Mediante el siguiente código los resultados cuadran y son correctos.

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

Su codificación es en notación Hexadecimal **0x3c017fff,** que pasándolo a binario queda de la siguiente forma:

|  |  |  |  |
| --- | --- | --- | --- |
| **Código Op (6bits)** | **Rs (5bits)** | **Rt (5bits)** | **K (16bits)** |
| 0011 10 | 11 000 | 0 0001 | 0111 1111 1111 1111 |

Su código de instrucción es **0011 10**

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

En binario es de la siguiente manera.

|  |  |  |  |
| --- | --- | --- | --- |
| **Código Op (6bits)** | **Rs (5bits)** | **Rt (5bits)** | **K (16bits)** |
| 0010 01 | 00 000 | 0 0010 | 0000 0000 0000 0001 |