School of Engineering and Information Technology

Department of Electrical and Computer Engineering

Computer Architecture

LAB 1. MIPS ASSEMBLER SIMULATOR

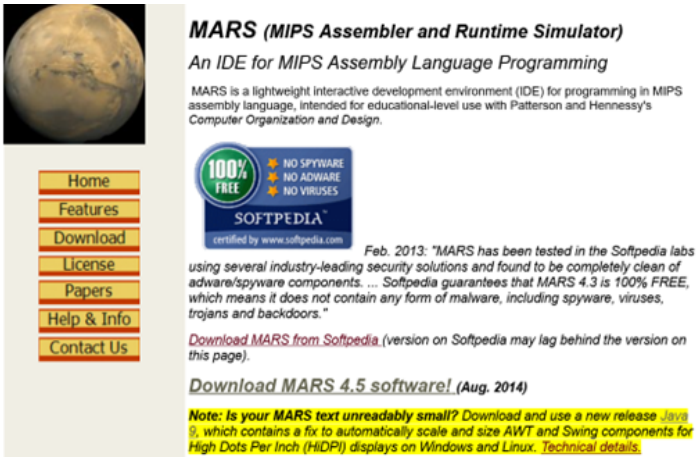
Equipo 5:

Héctor Javier Pequeño Chairez A01246364

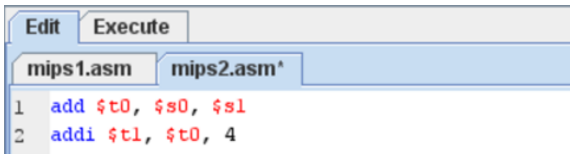
Gabriela Jazmín Álvarez Espinoza A00825719

Objective: Students will download and test a MIPS assembler simulator to identify how registers and memory locations look like. In addition, students will be able to practice with MIPS instruction sets seen in class.

1) ***Go to the web page: https://courses.missouristate.edu/KenVollmar/mars/download.htm and download the simulator.***



2) ***Open the simulator and identify your available section to write code in MIPS format. You must write your code before modifying any registers or memory locations.***



Comprobación de funcionamiento

Nuestra ejecución inicia con el siguiente fragmento de código:

Laboratorio1

```
1  add $t0,$s0,$sl
2  addi $tl,$t0,3
```

Codigo de prueba.

En donde $t0 = $s0 + $s1, en este caso le daremos a $s0 el valor de 2 y dejaremos en 0 el valor de $s1, por lo que la primera línea nos debería guardar en $t0 el valor de 2.

$t0	8	2

Comprobación de la operación, valor hexadecimal en tercera columna.

Después ejecutamos la segunda instrucción que se encarga de guardar en $t1 la suma del valor que se encuentra en el registro $t0 y la constante 3, por lo que el valor de $t0 debería ser el siguiente: $t1 = $t0 + 3 = 2+ 3 = 5.

$t0	8	2
$t1	9	5
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	2

Registros con valores actualizados.

Como bien podemos observar, todos los valores explicados anteriormente se encuentran respectivamente en la tercera columna del registro correspondiente y $t1 con un valor de 5 decimal.

Exercises

a) Write a code to set $t0 to ($s0 plus $s1). Use 5 and 4 values for $s0 and $s1 respectively.

Laboratorio1

```
1   add $t0,$s0,$s1
```

Fragmento de código utilizado para ejecutar la suma.

Una vez escrito nuestro código anterior, damos los valores a los registros $s0 y $s1, para ejecutar la suma y guardar su valor en $t0, en este caso $t0 = $s0 + $s1

$s0	16	5
$s1	17	4

Registros con los valores en formato decimal.

$t0	8	9

Registro temporal con el valor de la suma guardada 9 = 5 + 4.

Codigo:

add $t0,$s0,$s1

b) Use the value $ t0 and add the constant 3. Save the result in $t1. (Before this, you will need to write the new instruction to the above code and reassign the values to $ s0 and $ s1).

Laboratorio1*

```
1   add $t0,$s0,$s1
2   addi $t1,$t0,3
```

Código actualizado con la nueva instrucción para realizar la operación $t0 + 3 y guardarlo en el registro temporal $t1.

$s0	16	5
$s1	17	4

Registros con valores correspondientes.

$t1	9	12

Registro temporal 1 con el resultado guardado.

Debido a que el anterior código nos da un resultado almacenado en $t0 de 9, ahora que sumamos la literal 3, el registro guarda el valor correspondiente a $t1 = 9 + 3 = 12.

Codigo:

```
add $t0,$s0,$s1
addi $t1,$t0,3
```

c) Store contents of $t1 into memory location 0x10010020. Please use register $s3 to save the base address 0x10010000. (Before this, you will need to write the new instruction to the above code and reassign the values to the register used).

Primero debido a que nuestros valores los utilizamos en decimal, debemos convertir el valor de la memoria base de hexadecimal a decimal.

$$0x10010000 \rightarrow 268,500,992$$

$$0x20 \rightarrow 32$$

El valor anterior lo asignamos al registro $s3.

$s3	19	268500992

Valor añadido a el registro $s3.

Agregamos la instrucción que nos permitirá agregar el valor a la memoria indicada por el valor hexadecimal dado anteriormente + su offset de 32 decimal.

Laboratorio1

```
1  add $t0,$s0,$s1
2  addi $t1,$t0,3
3  sw $t1,32($s3)
```

Codigo actualizado.

Ensamblamos y ejecutamos código..

Data Segment

Address	Value (+0)
0x10010000	0
0x10010020	12

Valores en memoria actualizada.

Como podemos observar, en la posición de memoria 0x10010020 + 0, tenemos el resultado que guardamos en $t1, en el anterior ejercicio.

Codigo:

add $t0,$s0,$s1

addi $t1,$t0,3

sw $t1,32($s3)

d) Write 0x00000011 value to memory location: 0x10010028.

Iniciamos convirtiendo el 0x11 a valor decimal y obtenemos que

$$0x00000011 \rightarrow 17.$$

En este caso utilizaremos $t2 para asignar nuestro valor de 17 decimal y seguiremos utilizando el registro $s3 con la dirección base en memoria. Lo cual nos deja la siguiente instrucción.

```
6  # Parte d)
7  sw $t2, 40($s3)
```

Código actualizado. El valor 40 se obtiene de convertir el valor 28 a decimal.

En la siguiente imagen como se podrá observar, en el campo 0x10010020 + 8 = 0x10010028, el cual hace referencia hacia nuestra dirección en memoria que deseamos modificar, el valor 17 que es equivalente al 0x11.

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0	0	0
0x10010020	12	0	17

Valores en formato decimal.

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000000
0x10010020	0x0000000c	0x00000000	0x00000011

Valores en formato Hexadecimal.

Codigo:

Parte d)

sw $t2, 40($s3)

Now load $t2 with the value previously stored.

Ahora será necesario utilizar la instrucción lw, para traer de memoria un valor. Debido a que en el ejercicio pasado utilizamos $t2, para guardar el 0x11, modificamos el código anterior para que se guarde en el $t3 y se refleje el movimiento en $t2.

```
6   # Parte d)
7   sw $t3, 40($s3)
8
9   # Parte d.2)
10  lw $t2, 40($s3)
```

Codigo actualizado.

Como podemos observar en la siguiente imagen, ahora el valor de $t2 tiene el valor de 17, que corresponde a 0x11, al igual que el espacio en memoria de donde se extrajo (0x10010028).

$t1	9	12
$t2	10	17
$t3	11	17

Valor de los temporales, $t2 obtuvo el valor de extraerlo de memoria, mientras que $t3, fue el temporal con el que se cargó la memoria.

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0	0	0
0x10010020	12	0	17

Aquí se puede observar los valores en memoria.

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000000
0x10010020	0x0000000c	0x00000000	0x00000011

$t1	9	0x0000000c
$t2	10	0x00000011
$t3	11	0x00000011

Para complementar, se muestran los valores en hexadecimal.

Código acumulado hasta este punto:

```
# Parte a) - c)
add $t0,$s0,$s1
addi $t1,$t0,3
sw $t1,32($s3)
# Parte d)
sw $t3, 40($s3)
lw $t2, 40($s3)
```

e) Set $t0 to ($t2 minus $t1)

En este caso tenemos que guardar en $t0 el valor de $t2 - $t1, lo cual corresponde a 17-12 en valor decimal, por lo que buscaríamos que el valor final de $t0 fuera 5. Para esto usaremos la instrucción sub de la siguiente manera.

```
# Parte e)
sub $t0, $t2, $t1
```

Se muestra cómo utilizar la instrucción de resta para que se realice la operación $t0 = $t2 - $t1.

En este caso podemos ver el resultado guardado en los registros.

$t0	8	5
$t1	9	12
$t2	10	17

Valores registros.

En este caso podemos observar el resultado deseado guardado en el registro $t0 y los valores correspondientes a $t2 y $t1.

Codigo:

Parte e)

sub $t0, $t2, $t1 # t0 = t2 - t1 = 17 - 12 = 5, t0 = 5

f) Using your above code, try to address memory locations using negative numbers (for offsets) to compute memory addresses.

Ahora aumentaremos la dirección base en 80 unidades decimales, para que al momento de restarlas de nuestradireccion con número negativos, sigamos utilizando los mismos espacios de memoria, pero ahora realizando una resta en el offset.

Dirección base = 268,500,992 + 80 $\rightarrow$ *268,501,072*

Diferencia = Dirección deseada - Dirección Base.

Para acceder al espacio de memoria 0x10010028, necesitamos obtener la cantidad a restar para acceder a ella, entonces:

Para acceder a la primera dirección de memoria.

Diferencia =0x10010050 - 0x10010028 = *48 decimal.*

Para la segunda dirección calculamos de nuevo esta diferencia.

Diferencia =0x10010050 - 0x10010028 = *40 decimal.*

Una vez calculadas las restas modificamos el código.

Esto nos deja con el siguiente código:

```
# Parte a) - c)
add $t0, $s0, $s1 # t0 = s0 + s1 = 5 + 4, t0 = 9
addi $t1, $t0, 3  # t1 = 9 + 3, t1 = 12
sw $t1, -48($s3)  # memoria [268,500,992 + 32 =  268,501,024] = 12

# Parte d)
sw $t3, -40($s3)  # memoria [268,500,992 + 40 = 268,501,032] = 17 | 0x11

# Parte d.2)
lw $t2, -40($s3)  # t2 = memoria [268,500,992 + 40 = 268,501,032] = 17

# Parte e)
sub $t0, $t2, $t1 # t0 = t2 - t1 = 17 - 12 = 5, t0 = 5
```

Codigo actualizado.

A continuación se muestran los espacios en memoria con sus valores respectivos y los registros con los mismos valores de los cálculos obtenidos anteriormente.

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0	0	0
0x10010020	12	0	17

Los valores en memoria coinciden con los calculados anteriormente y en sus respectivas posiciones.

$t0	8	5
$t1	9	12
$t2	10	17
$t3	11	17
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	4
$s2	18	0
$s3	19	268501072

Al igual, los registros contienen los mismos valores obtenidos anteriormente, donde $t0 guarda el valor de la resta de 17-12 = 5.

Código (Los comentarios fueron utilizados para guiarnos en el cálculo de la compensaci[on de la dirección para usar el offset con valores negativos):

```
# Parte a) - c)
add $t0, $s0, $s1 # t0 = s0 + s1 = 5 + 4, t0 = 9
addi $t1, $t0, 3  # t1 = 9 + 3, t1 = 12
sw $t1, -48($s3)  # memoria [268,500,992 + 32 =  268,501,024] = 12
# Parte d)
sw $t3, -40($s3)  # memoria [268,500,992 + 40 = 268,501,032] = 17 | 0x11
# Parte d.2)
lw $t2, -40($s3)  # t2 = memoria [268,500,992 + 40 = 268,501,032] = 17
# Parte e)
sub $t0, $t2, $t1 # t0 = t2 - t1 = 17 - 12 = 5, t0 = 5
```

g) Write an immediate instruction for the next operation:

Register t0 = 17-19

Debido a que no se puede ejecutar una instrucción del tipo subi $t0, 17,19. Primero deberemos cargar el valor 17 a algún registro, en este caso decidimos cargarlo en el registro $t4, para después restarle el valor de 19.

```
# Parte f)
subi $t0, $t4, 19
```

Resta considerar el $t4 para almacenar el valor 17.

En la siguiente imagen podemos observar el resultado almacenado en $t0 con su signo correcto y su magnitud. Donde $t0 = 17 - 19 = -2.

$t0	8	-2
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	17

Registros con los valores correctos, para realizar la resta.

Codigo:

```
# Parte f)
subi $t0, $t4, 19
```

Código Final de la práctica.

```
# Parte a) - c)
add $t0, $s0, $s1 # t0 = s0 + s1 = 5 + 4, t0 = 9
addi $t1, $t0, 3  # t1 = 9 + 3, t1 = 12
sw $t1, -48($s3)  # memoria [268,500,992 + 32 =  268,501,024] = 12

# Parte d)
sw $t3, -40($s3)  # memoria [268,500,992 + 40 = 268,501,032] = 17 | 0x11

# Parte d.2)
lw $t2, -40($s3)  # t2 = memoria [268,500,992 + 40 = 268,501,032] = 17

# Parte e)
sub $t0, $t2, $t1 # t0 = t2 - t1 = 17 - 12 = 5, t0 = 5

# Parte f)
subi $t0, $t4, 19
```