UNIVERSIDAD DE SANTIAGO DE CHILE

Pre-Laboratorio 1: Instrucciones MIPS y Simulación en MARS

Arquitectura de Computadores 2021-2

Nombre: John Serrano Carrasco

Sección: 13309-0-A-1

Fecha: 30 de septiembre de 2021

I. Preguntas acerca de MARS:

1) ¿Cómo se ensambla un programa MIPS?

Una vez escrito el programa correctamente, se debe seleccionar la opción “Assemble the current file and clear breakpoints” y de esa forma el programa será ensamblado.

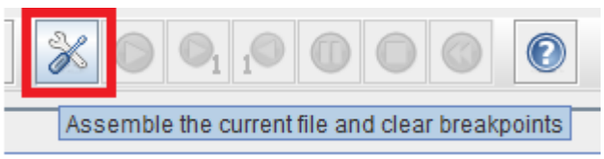


Figura N°1: Marcada con rojo, la opción para ensamblar un programa MIPS.

2) ¿Cómo se ejecuta un programa en MIPS?

Para ejecutar un programa en MIPS, primero se debe ensamblar el programa y luego, hay dos opciones: La primera es apretar el botón que esta al lado del botón para ensamblar, el cual dice “Run the current program”. Esta opción ejecutará el programa en su totalidad. La segunda opción es apretar el botón que esta al lado derecho de la última opción mencionada, el cual dice “Run one step at a time”. Lo que hará esta opción es ejecutar el programa línea por línea (Se ejecuta una línea y luego se debe apretar el botón de nuevo para poder ejecutar la siguiente línea y así sucesivamente hasta finalizar el programa).

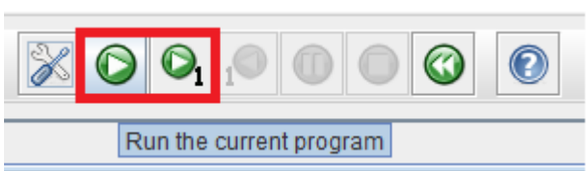
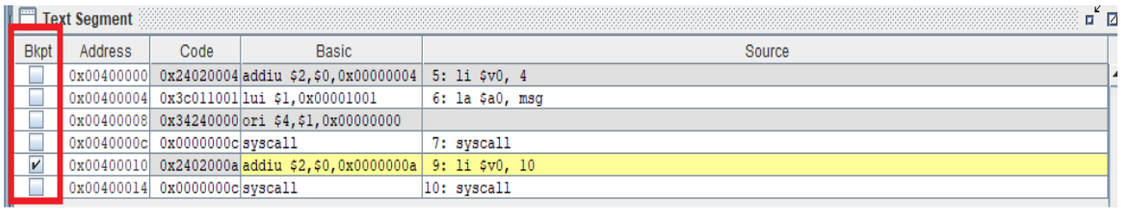


Figura N°2: Marcadas con rojo, ambas opciones anteriormente mencionadas.

3) ¿Cómo detendrías la ejecución en cierta línea que no es necesariamente la última?

Se deben utilizar breakpoints. Los breakpoints aparecen una vez que el código este ensamblado, en la parte izquierda del menú de MARS y permiten elegir la línea en la que se quiere detener el código tras ejecutarlo.



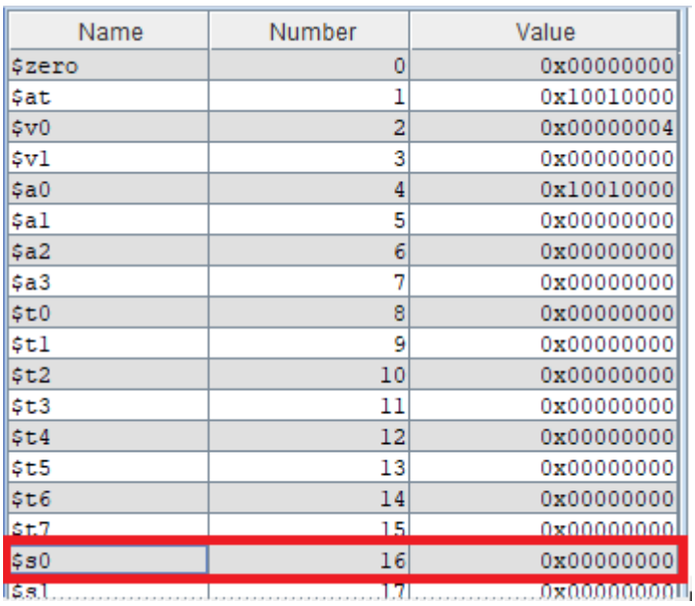
Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x24020004	addiu $2,$0,0x00000004	5: li $v0, 4
☐	0x00400004	0x3c011001	lui $1,0x00001001	6: la $a0, msg
☐	0x00400008	0x34240000	ori $4,$1,0x00000000	
☐	0x0040000c	0x0000000c	syscall	7: syscall
☑	0x00400010	0x2402000a	addiu $2,$0,0x0000000a	9: li $v0, 10
☐	0x00400014	0x0000000c	syscall	10: syscall

Figura N°3: En este ejemplo, se utiliza el breakpoint correspondiente a la línea 9 para detener la ejecución del código en esa línea.

4) ¿Dónde encontrarías el valor actualmente almacenado en el registro $s0?

En el menú de la derecha de MARS, aparecen varios registros junto con un número de identificación y el valor del registro. El registro $s0 corresponde al registro numero 16 y por lo tanto ahí se puede ver el valor actualmente almacenado.

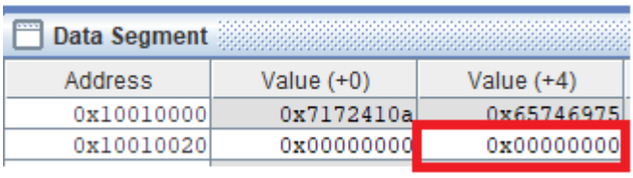


Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x00000004
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000

Figura N°4: Tabla de registros donde el registro $s0 está marcado. Su valor actual es 0x00000000

5) ¿Dónde encontrarías el valor actualmente almacenado en la dirección de memoria 0x10010024?

El valor almacenado en la dirección se puede encontrar en el Data Segment tras haber ensamblado el código. Se debe seleccionar la opción “0x10010000 (.data) para poder encontrarla fácilmente. El valor se encuentra donde Adress es 0x10010020 y Value es (+4).



Data Segment

Address	Value (+0)	Value (+4)
0x10010000	0x7172410a	0x65746975
0x10010020	0x00000000	0x00000000

Figura N°5: Valor de la dirección de memoria x10010024 en el Data Segment.

II. Preguntas sobre predecir el funcionamiento de programas MIPS:

Para cada programa, responde la pregunta sobre el resultado. No utilices software en esta parte, sino que desarrolla "a mano" escribiendo el resultado en papel.

Nota: Estuve experimentando con MARS y algunas de las operaciones básicas de MIPS, para poder conocer mejor como funciona su código y algunas de estas operaciones. Las siguientes son mis explicaciones sobre lo que sucede en el programa hasta llegar a la respuesta de la pregunta, basándome en lo que experimenté antes de realizar los siguientes programas a mano.

1. ¿Cuál es el valor en los registros $t1 y $t0 al terminar el programa?

```
addi $t0, $zero, 4
add $t1, $t0, $t0
```

Similar a una de las máquinas vistas en los cursos anteriores, MIPS tiene una estructura muy parecida con sus instrucciones: operaciones – lugar a guardar el resultado – operador 1 – operador 2.

Para el caso de addi, se suma lo que esta guardado en $zero, que corresponde a 0 y 4. Por lo tanto, en $t0 se guarda el número 0x00000004.

Luego con la instrucción add, se toma lo que esta guardado en $t0 con $t0 para así obtener 4 + 4 = 8. Este 0x00000008 se guarda en t$1 y el programa finaliza.

Por lo tanto, en el registro $t1 queda guardado el número **0x00000008** y en el registro $t0 queda guardado el número **0x00000004**.

2. ¿Cuál es el valor en los registros $t1 y $t0 al terminar el programa?

```
addi $t0, $zero, 2
addi $t1, $zero, 2
beq $t0, $t1, A
addi $t1, $zero, 1
```

Inmediatamente se puede notar algo: Gracias a MARS, sabemos de antemano que no existe ningún registro "A", ni tampoco opera con números hexadecimales, por lo que ese "A" que se encuentra en la línea 3, no esta definido. Por lo tanto, el programa solo debería ejecutar las primeras dos líneas antes de detenerse en la tercera.

La primera instrucción, suma el contenido de $zero, el cual es 0, con 2, guardando en $t0 el valor 0x00000002.

La segunda instrucción, realiza el mismo procedimiento, pero esta vez guarda el resultado en $t1, quedando este con el valor 0x00000002.

Por lo tanto, al terminar (prematuramente) el programa, $t1 y $t0 quedan con el valor **0x00000002.**

3. ¿Cuál es el valor en los registros $t2, $t1 y $t0 y las direcciones de memoria 0x10010000 y 0x10010004 al terminar el programa?

```
addiu $t0, $zero, 0x10010000
addi $t1, $zero, 5
sw $t1, 0($t0)
lw $t2, 0($t0)
addiu $t0, $t0, 4
sw $t2, 0($t0)
```

Nota: Para este problema, se considera que los registros y memorias mencionadas en el enunciado parten con el valor 0x00000000.

Primero, se parte sumando el contenido de $zero con el número 0x10010000, lo cual nos da x10010000 ya que $zero tiene a 0x00000000. Se guarda el resultado en $t0.

Luego se sume el contenido de $zero con 5, por lo que en $t1 se guarda el valor 0x00000005.

Para la siguiente línea se realiza la suma 0 + el contenido de $t0, lo que da como resultado x10010000. La instrucción nos dice que en esta memoria se debe guardar el contenido de $t1, el cual es 0x00000005 y por lo tanto, la memoria queda con el ultimo valor mencionado anteriormente.

Continuando, la siguiente línea nos dice la operación inversa a la anterior y pues esta vez debemos guardar en $t2 el contenido de la memoria 0x10010000, por lo que $t2 queda con el valor 0x00000005.

Para la quinta línea, se suma el contenido de $t0 con el número 4, quedando guardado el valor 0x10010004 en el registro $t0.

Finalmente, en la última línea del programa, se guarda en la memoria 0x10010004 el contenido de $t2, por lo que esta memoria queda con el valor 0x00000005.

En conclusión, los registros y memorias mencionadas en el enunciado quedan con los siguientes valores:

Registro $t2	0x00000005
Registro $t1	0x00000005
Registro $t0	0x10010004
Memoria 0x10010000	0x00000005
Memoria 0x10010004	0x00000005