Laboratorio 1: Operaciones básicas en Ensamblador MIPS

Reinaldo Pacheco Parra (26 horas)
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, Santiago, Chile
reinaldo.pacheco@usach.cl

***Resumen*—En el siguiente Laboratorio se documentará el procedimiento realizado para resolver una problemática mediante el uso de MARS Mips. Para ello, se debió realizar una investigación del funcionamiento del simulador mencionado con el fin de resolver las operaciones básicas Suma, Resta, Multiplicación y Division mediante un código con un menú que contiene todas las operaciones, aplicando el uso de direcciones de memoria, funcionamiento de registros y construcción de operaciones en base a otras. Además se aplican temas tales como la recursividad y los saltos incondicionales que se aplicaron en la resolución el problema planteado.**

***Palabras claves*—MARS, MIPS, Saltos, Registros.**

I. Introducción

En el siguiente laboratorio se realizan operaciones aritméticas básicas (suma, resta, multiplicación y división) en lenguaje ensamblador utilizando el simulador MARS (MIPS Assembler and Runtime Simulator). El programa tiene como objetivo representar el resultado de manera especial, separando y ordenando en forma inversa cada dígito. Esto se realiza para comprender mejor las inscripciones de civilizaciones antiguas. Para resolver el problema, el usuario carga valores aleatorios en los operandos 1 y 2 en la sección de datos. Luego, selecciona la operación aritmética deseada a través de la consola. El programa almacena el resultado en un registro y lo muestra en pantalla con el formato mencionado. Este laboratorio busca comprender los conceptos básicos del simulador MARS, incluyendo el acceso y comprensión de la memoria, el uso de instrucciones de salto, el almacenamiento de datos y la capacidad de utilizar operaciones básicas para resolver problemas más complejos.

II. Antecedentes

Mars MIPS es un entorno de desarrollo integrado (IDE) para el lenguaje de programación MIPS, que se utiliza en la arquitectura de procesadores MIPS. Este IDE permite escribir, depurar y ejecutar programas en un entorno gráfico de usuario, lo que lo hace útil tanto para estudiantes como para profesionales que trabajan con este simulador. [1]

Las operaciones que se pueden realizar en MIPS son importantes para el desarrollo de programas. El amplio conjunto de operaciones disponibles permite realizar cálculos, transferir y almacenar datos en direcciones de memoria. En la tabla I, titulada ”Métodos y operaciones básicas del conjunto de instrucciones MIPS de Mars”, se presentan algunas de estas operaciones.

Además, el manejo de direcciones de memoria es fundamental en la programación en MIPS. Cada instrucción trabaja con una dirección de memoria que indica la ubicación espécifica de los datos que se deben procesar. Por lo tanto, es importante que el programador maneje adecuadamente e indique correctamente las direcciones de memoria a utilizar para evitar errores en el programa. [2]

Por último, se presenta la descripción de algunos conceptos importantes que se deben comprender, ya que fueron utilizados en la construcción del laboratorio, estos son:

Recursividad: La recursividad es una técnica en la que una función se llama a sí misma para resolver un problema o realizar una tarea que se vuelve a repetir. Para evitar que la función se ejecute infinitamente, es necesario establecer un caso base.[3]

Salto incondicional: En MIPS, un salto incondicional j ”jump”permite ir directamente a una etiqueta específica del código sin depender de que se deba cumplir alguna condición especifica. La sintaxis es: **j label** con j como la instrucción y label la etiqueta a la cual se quiere saltar. Esta instrucción es muy útil para realizar ciclos o salir fácilmente de alguna función.

II-A. Tablas

Método/Operación	Función
`Temporales t0-t7`	Espacios de almacenamiento en el procesador
`add`	Suma dos registros y almacena el resultado en un registro destino
`addi`	Suma un valor inmediato a un registro y almacena el resultado en un registro destino
`sub`	Resta dos registros y almacena el resultado en un registro destino
`lw`	Carga un valor de la memoria en un registro
`sw`	Almacena un valor de un registro en la memoria
`beq`	Compara dos registros y salta a una etiqueta de destino si son iguales
`bne`	Compara dos registros y salta a una etiqueta de destino si son diferentes
`j`	Salta a una etiqueta de destino sin ninguna condición
`jr`	Salta a la dirección de retorno almacenada en un registro

Tabla I: Métodos y operaciones básicas del conjunto de instrucciones MIPS de Mars [4]

II-B. Algoritmos

Se presenta el algoritmo que permite construir la multiplicación de dos números mediante sumas iteradas.

Algoritmo 1 Multiplicación mediante Sumas Iteradas

Entrada: Dos números enteros a y b.
Salida: El producto $c = a \times b$.
$c \leftarrow 0$
if $b < 0$ **then**
    $a \leftarrow -a$
    $b \leftarrow -b$
end if
for $i = 1$ to b **do**
    $c \leftarrow c + a$
end for
c

La construcción de este algoritmo es importante, ya que muestra al usuario una forma de utilizar la multiplicación sin las operaciones restringidas. Además permite evidenciar que es posible el desarrollo de operaciones que estén prohibidas o no existan, en base a otras, con el objetivo de utilizarlas durante la creación del programa.

II-C. Fórmulas

Las fórmulas de las operaciones que se realizan en este laboratorio mediante el programa MIPS y que fueron creadas con el editor son las siguientes

Suma:

$$a + b = c \tag{1}$$

Resta:

$$a - b = c \tag{2}$$

Multiplicación:

$$a \times b = c \tag{3}$$

División:

$$\frac{a}{b} = c \tag{4}$$

Donde a y b son los valores que se suman, restan, multiplican o dividen, y c es el resultado obtenido. Se pueden cambiar los valores de a y b según las necesidades del usuario.

III. Materiales y métodos

III-A. Materiales

Se debe descargar el simulador MARS Mips 4.5 de su página oficial para abrir los archivos con extensión .asm. Además, el dispositivo que se encarga de almacenar, procesar y ejecutar los códigos es un Notebook NITRO 5 con procesador Intel® Core ™ i5-10300H CPU @ 2.50 GHz 16 GB de RAM (15,8GB utilizables), Sistema operativo de 64 bits, procesador x64 con una tarjeta gráfica NVIDIA® GeForce® GTX 1650 y la versión 22H2 de Windows 11.

III-B. Métodos

El usuario accede al simulador MARS Mips, carga desde el segmento el operando 1 en la dirección de memoria **0x100100a0** y el operando 2 en la dirección **0x100100c0** ,luego selecciona la operación aritmética que desea realizar, recibe el resultado en la dirección **0x100100e0** y se muestra por pantalla en el nuevo formato.

Las operaciones a realizar en el programa son las siguientes:

Suma: Si se selecciona la primera opción se realiza la operación suma. Esta se realiza mediante la instrucción **add** la cual toma ambos operandos y guarda en un tercer registro el resultado de la adición entre ellos.

Resta: Si se selecciona la segunda opción, se realiza la operación resta. Esta se realiza mediante la instrucción **sub** la cual toma ambos operandos y guarda en un tercer registro el resultado de la diferencia entre ellos.

Multiplicación: Si se selecciona la tercera opción, se realiza la operación multiplicación. Esta se realiza mediante sumas iteradas a través de las instrucciones **add** y **addi**. Sumando el operando 1 la cantidad de veces que indique el operando 2.

División: Si se selecciona la tercera opción, se realiza la operación división. Esta se realiza mediante restas iteradas a través de las instrucciones **subi** y **subi**. Calculando la resta del operando 2 frente al operando 1 hasta que el resultado de la resta sea menor que el divisor. Si este es igual a 0, corresponde a una división entera, en el caso contrario, se debe realizar el mismo procedimiento con el resto, multiplicandolo por 10 hasta obtener el resultado decimal. Entregando el numero entero junto a su parte decimal.

Para el resultado de cada operación se debe realizar la inversión de sus dígitos. Guardando el resultado en la dirección de memoria anteriormente mencionada.

IV. Resultados

Se muestran los resultados y el análisis de resultados.

En la Figura 1 se muestra una imagen de la selección de operación, la cual permite escoger entre Suma, Resta, Multiplicación y División entre los operandos ingresados.

Además se muestra el segmento de datos, donde se ingresa el operando 1 en el registro 0x100100a0, el operando 2 en el registro 0x100100c0, y luego de realizar la operacion, se guarda el resultado invertido en el registro 0x100100e0. (ver Figura 2)

En el caso de seleccionar la opción 1, se muestra por consola el resultado de la suma entre el operando 1 y el operando 2. (ver Figura 3) Así mismo, si se selecciona la opción 2, se muestra por consola el resultado de la resta entre los operandos (ver Figura 4) Si la operación seleccionada por el usuario es la opción 3, se muestra por consola el resultado de la multiplicación entre el operando 1 y el operando 2 (ver Figura 5) Finalmente, si la opción que se escoge es la 5, se muestra por consola el resultado de la división entre los operandos (ver Figura 6)

Durante el análisis de los resultados, se observó que las cuatro operaciones construidas lograron almacenar correctamente el resultado en el registro correspondiente. Además, se muestra el resultado a través de la consola para que pueda ser leído por el usuario.

Otro de los aspectos a evaluar es que la precisión de las operaciones realizadas, los valores obtenidos fueron los esperados, es decir, las operaciones funcionan correctamente y sirven para calcular resultados entre los operandos que escoja el usuario.

Un aspecto relevante a tener en cuenta es que las operaciones de multiplicación y división presentaron un tiempo levemente superior en comparación con las operaciones de suma y resta. Esto puede deberse a la cantidad de iteraciones que el programa realiza durante el cálculo de las sumas y restas necesarias para la construcción de las operaciones de multiplicación y división.

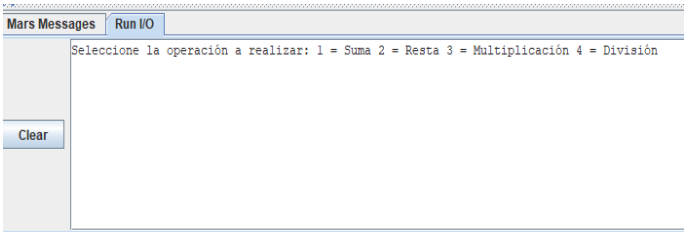

Figura 1: Selección de operación

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	1701602643	1869177699	1814062446	1886330977	1667330661
0x10010020	540701281	540876849	1634563411	1025520160	1936020000
0x10010040	1667853424	-211197087	540287086	1766072381	1769171318
0x10010060	0	0	0	0	0
0x10010080	0	0	0	0	0
0x100100a0	1	2	3	4	0
0x100100c0	4	5	6	0	0
0x100100e0	5	7	9	4	0
0x10010100	0	0	0	0	0
0x10010120	0	0	0	0	0

Figura 2: (Ejemplo de guardado en el Data Segment)

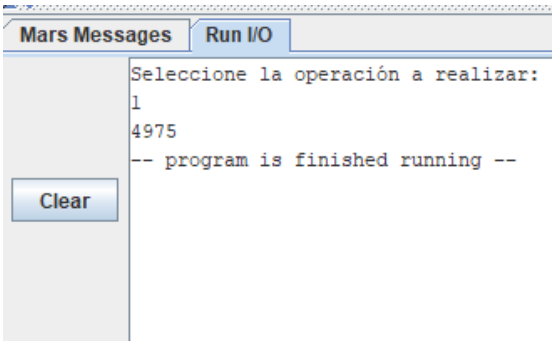

Figura 3: (Resultado por consola para la suma)

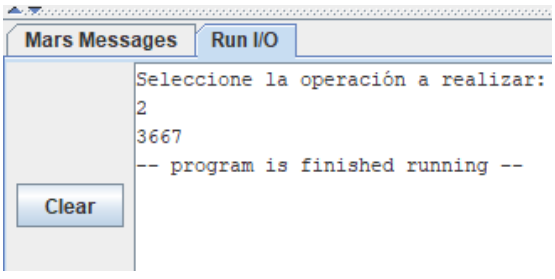

Figura 4: (Resultado por consola para la resta)

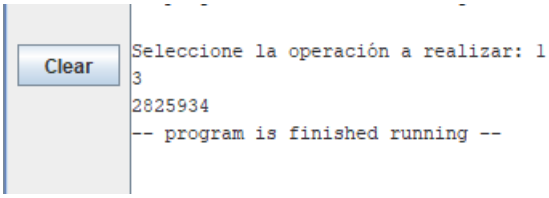

Figura 5: (Resultado por consola para la multiplicación)

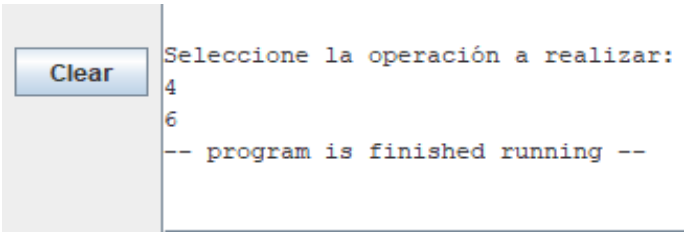

Figura 6: Resultado por consola para la división

V. Conclusiones

En conclusión, la realización de este laboratorio permitió abordar de buena forma los conceptos básicos del simulador MARS Mips, ya que gracias a la elaboración de los códigos se permitió desarrollar el pensamiento lógico de algunos ejercicios, como el problema de no poder utilizar las funciones mul, div ni derivados, por lo que se tuvieron que elaborar operaciones un poco más complejas en base a operaciones básicas conocidas. También se permitió entender el funcionamiento de los registros de memoria y como estos almacenan los datos necesarios para realizar la operacion y guardar el resultado de forma correcta. Si bien se cumplió la mayoría de los objetivos propuestos al comienzo tales como comprender operaciones con MIPS, uso de instrucciones de salto y construcción de todas las operaciones, se dificultó la elaboración de la parte decimal al momento de dividir los operandos, si bien, se guarda y entrega correctamente la división entera, no se logró identificar un método para guardar la parte decimal restante.

Referencias

[1] C. W. Kann, “Introduction to mips assembly language programming,” [Online] https://cupola.gettysburg.edu/cgi/viewcontent.cgi?article=1001&context=oer, 2003, visitada el 26 de Abril del 2023.

[2] A. H. Cerezo, “Arquitectura mips trabajo 2,” [Online]https://www.infor.uva.es/~bastida/OC/TRABAJO1_MIPS.pdf, 2018, visitada el 22 de Mayo del 2023.

[3] K. R. Irvine, “Assembly language for x86 processors,” [Online] http://www.nlpir.org/wordpress/wp-content/uploads/2019/03/Assembly.Language.For_.x86.Processors.Kip_.R..Irvine..6ed.Prentice.Hall_.2011www.xuexi111.com_.pdf, 2010, visitada el 26 de Mayo del 2023.

[4] K. L. Algara, “Mars mips assembler and runtime simulator,” [Online] https://usermanual.wiki/Document/MarsManualUsuario.686917485/view, 2012, visitada el 22 de Mayo del 2023.