Informe laboratorio N°1 - Implementación de operaciones ariméticas en MIPS

Benjamín Zúñiga Jofré
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, Santiago, Chile
benjamin.zuniga.j@usach.cl

***Resumen*—En este informe se verá cómo es posible implementar operaciones aritméticas como multiplicación y división en el lenguaje ensamblador MIPS, por medio de otras operaciones más básicas que dispone el lenguaje, aparte se verá cómo acceder a direcciones de memoria específicas y trabajar con el contenido de estas y como guardar contenido en estas también.**

***Palabras claves*—MIPS, MARS, lenguaje ensamblador y algoritmo.**

I. Introducción

Este laboratorio N°1, perteneciente a la asignatura Arquitectura de Computadores, tiene como objetivo lograr implementar operaciones aritméticas básicas, tales como suma, resta, multiplicación y división en el lenguaje ensamblador MIPS por medio del IDE MARS, con algunas particularidades y restricciones, como que los operandos de cada operación se ingresan en una dirección de memoria específica y con sus dígitos al revés, es decir, si se quiere operar el 21, este está guardado como 12, el signo de cada operando se encuentra en otras direcciones de memoria específicas, el resultado de la operación debe ser guardado en una dirección de memoria en particular y en el mismo formato, es decir, sus dígitos están al revés, y no se puede hacer uso de los operadores mul y div, entre otros, obligando a crear un algoritmo que sea equivalente a los operadores mencionados para implementar la multiplicación y división.

II. Antecedentes

II-A. Palabras claves

1 - MIPS: ”Las siglas MIPS (*Microprocessor without Interlocked Pipelines Stages*, o lo que es lo mismo, microprocesador sin enclavamiento de estados de tuberías) hacen referencia a la gama de microprocesadores desarrollados por MIPS TEchnologies, de arquitectura RISC y registros tipo propósito general de clasificación registro-registro, en los que la mayoría de las instrucciones no acceden a memoria(salvo las instrucciones de carga/descarga y las instrucciones de los procesadores presentan dos operandos, el fuente y el resultado.”[1]

2 - MARS: ”MARS (MIPS Assembler and Runtime Simulator) es un entorno de desarrollo interactivo (IDE) para programación en lenguaje ensamblador MIPS, destinado a uso a nivel educativo.”[2]

3 - Lenguaje ensamblador: ”Lenguaje ensamblador es un lenguaje de muy bajo nivel, legible por humanos y programable, donde cada instrucción de lenguaje ensamblador corresponde a una instrucción de código de máquina de computadoras. Los programas de lenguaje de ensamblaje se traducen directamente en instrucciones de código de máquina, y cada instrucción de ensamblaje se traduce en una sola instrucción de código de máquina.”[3]

4 - Algoritmo: ”Se puede entender por algoritmo, cualquier procedimiento computacional bien definido que parte de un estado inicial y un valor o un conjunto de valores de entrada, a los cuales se les aplica una secuencia de pasos computacionales finitos, produciendo una salida o solución. Se puede considerar al algoritmo como una herramienta para resolver un cálculo computacional bien especificado”[4]

III. Materiales y Métodos

III-A. Materiales

Para desarrollar la actividad fue utilizado el lenguaje ensamblador MIPS, por medio del IDE MARS en su versión 4.5. Por otra parte utilizado fue un ordenador portátil que cuenta con los siguientes componentes: procesador Intel Core i5-10300H, una tarjeta gráfica Nvidia Geforce GTX 1650, memoría RAM de 8GB a 2933 MHz, una unidad de estado sólido Intel de 512 GB y cuenta con el sistema operativo Windows 11 de 64 Bits.

III-B. Métodos

Para entender mejor los métodos utilizados estos serán separados y explicados uno a uno en el orden que fueron realizados.

III-B1. ”Voltear”los operandos: Lo primero realizado fue el procedimiento de ”voltear”los operandos, los cuales al ser ingresados mediante el segmento de datos, se deben cargar y almacenar temporalmente en registros para poder aplicarles una operación en particular, la cuál debe ser escogida posteriormente por consola. Es importante notar cada número debe estar descompuesto en los dígitos lo forma y estos están al revés.[en] el segmento de datos, es decir, el 1357 debe estar en el segmento de datos como 7531, donde el 7 estará en la primera ”posición”del segmento de datos, el 5 en la segunda, el 3 en la tercera y el 1 en la cuarta, es por esto que para ”voltear.[el] número se puede ir recorriendo el segmento de datos en orden y multiplicando

cada dígito por su unidad correspondiente, que pueden ser expresadas como potencias de base 10 y el exponente será la posición menos 1, es decir, al cargar el dígito en la primera posición este debe ser multiplicado por 10^0, el dígito en la segunda posición por 10^1, así sucesivamente hasta la séptima posición. Para realizar lo anterior se realizó el siguiente proceso: se carga el dígito en el registro y este se debería multiplicar por la potencia de 10 correspondiente a la posición del dígito en el segmento de datos, pero como no se puede usar la operación mul, la solución a esto fue llevar a cabo un ciclo de sumas de la potencia de 10 en el registro t2, donde la cantidad de veces que se ejecuta el ciclo está dada por el dígito, es decir, si en la posición 3 se encuentra un 3 se suma 3 veces 100 en el registro t2 mediante la operación addi, para garantizar que el ciclo se efectúe la veces que corresponde cada vez que se suma en t2 la potencia de 10 se resta 1, mediante la operación subi, al contenido t1, registro donde originalmente se carga el dígito del operando, por lo cuando t1 es 0 se salta a calcular el siguiente dígito, y finalmente todo queda almacenado en el registro t2 y posteriormente el número es movido al registro t3 para realizar el mismo proceso con el siguiente operando, quedando este guardado en el registro t2.

III-B2. Escojer la operación: El siguiente proceso a realizar es escoger la operación, la cual debe ser ingresada por consola, para esto la solución encontrada es imprimir un string el cual vincula cada operación, es decir, suma, resta, multiplicación y división, con 1, 2, 3 y 4 respectivamente, pidiendo aspi que sea ingresado el número en vez del nombre o signo de la operación, para ser este número almacenado en registro a0 y luego dependiendo el contenido en a0 saltar a realizarse la operación correspondiente.

III-B3. Suma: Para realizar la suma se utilizó la operación add entre el contenido del registro t3 y t2, que contienen el primer y segundo operando respectivamente y se almacena en el registro a0 para luego imprimirlo, con el único detalle que al estar los signos de los operandos en otros registros primero se verifica que signo le corresponde a cada operando, en caso de que el registro que corresponde al signo de cada operando contenga un 1, que equivale a que el operando es negativo, se actualiza el registro mediante la operación sub, restando a 0 el contenido del registro y dejando así el número negativo, luego se realiza la suma y finalmente se verifica si el resultado es menor que 0, lo cual hace que se almacene un 1 en s3, en caso contrario se almacena un 0 y se salta a imprimir el resultado.

III-B4. Resta: Para realizar la resta se utilizó la operación sub, entre el contenido del registro t3 y t2, que contienen el primer y segundo operando respectivamente y se almacena en el registro a0 para luego imprimirlo y al igual que en la suma se verifican los registros que contienen signos de cada operando y se vuelve negativo el operando que corresponda, luego se realiza la resta y finalmente se verifica si el resultado es menor que 0, lo cual hace que se almacene un 1 en s3, en caso contrario se almacena un 0 y se salta a imprimir el resultado..

III-B5. Multiplicación: La manera en que se implemento la multiplicación, frente a la prohibición de usar la opreación mul, es al igual que en el ”volteo”de los operandos cuando se multiplicaban por potencias de 10, fue realizar un ciclo de sumas , tantas veces sea el valor del segundo operando y la manera en la que se implementó fue copiar el valor en el registro t3 (primer operando) en el registro t4 y luego actualizar el contenido de t3 con el mismo contenido de t3 más el contenido de t4, luego restar 1 al contenido de t2, y así vuelve a empezar el ciclo, que se repite hasta que en t2 haya un 0, por otra parte con respecto al signo de cada operadno, estos no se tomaron en consideración y se trabajó con ambos operandos positivos independiente de lo que indique el registro correspondiente al signo de cada operando, los cuales son son commparados directamente para almacenar el signo del resultado, almacenando un 1 en el registro s3, que indica que el resultado es negativo, que en caso que sean distintos los valores en s1, s2, ya que indicaría que se está multiplicando un número negativo con uno positivo lo cual da como resultado un número negativo, y en caso de que el contenido de s1 y s3 sea igual provoca que se almacene un 0 en el registro s3, indicando que el resultado es positivo.

III-B6. División: Para lograr implementar la división, frente a la prohibición del uso de div, se diseñó un algoritmo el cual consiste en un ciclo de restas y funciona para calcular tanto la parte entera de la división como los cuatro primeros decimales, solo que estan implementadas en etiquetas por separado para hacer el cálculo de está operación, y el procediento contruido es: restarle al primer operando ubicado en t3, el segundo operando ubicado en t2 y almacenar este resultado en t4, luego comprobar si el es negativo, es decir, menor que 0, en caso de que el resultado no sea negativo,se actualiza el valor en el registro t3, con el resultado de la anterior resta, y se adiciona un 1 al valor en el registro a0, guardondose en el mismo registro a0, y se vuevle a empezar el ciclo, en caso de que el resultado de la resta sea menor que 0, se salta a un paso intermedio, el cual se multiplica por 10, el valor en t3 para volver a repetir el ciclo hasta 4 veces más, siendo 5 en total, donde la primera vez calcula la parte entera dela división y las otras 4 cada decimal, considerar que cada uno de estos ciclos almacena su resultado en registros distintos, facilitando posteriormente el almacenamiento del valor, por último al igual que en la multiplicación no se consideran los signos de cada operando, y se comparan entre sí para obtener el de resultado.

III-B7. Resultados de las operaciones: Para mostrar los resultados, existen dos procedimientos, uno para la suma, resta y multiplicación, y uno solo para división, esto debido que las primeras 3 operaciones el resultado es solo un entero y la división entrega una parte entera más 4 decimales, ambos imprimen un string indicando que el siguiente valor posterior al string es el resultado y también mueven la parte entera al registro t0 antes de saltar a guardar el resultado. El primer procedimiento funciona imprimiendo el número almacenado en a0, que contiene el resultado independiente de la operación realizada, luego verifica si el resultado es negativo y en caso de serlo lo convierte en positivo para saltar a guardar el número, en caso de no ser negativo el resultado salta a guardar el número. El segundo procedimiento es un poco más extenso, ya que la solución

implementada primero verifica si el resultado de la división es negativo, en caso de serlo imprime hace negativa la parte entera, seguido de la parte entera de la división, luego se imprime una coma seguida de los cuatro decimales en orden, aprovechando que la implementación de la división guarda los decimales por separado, a medida se van imprimiendo se guardan en el segmento de datos correspondiente, con el mismo formato que se encontraban los operandos, es decir, con sus dígitos separados y al revés, en caso de ser negativa y por último se salta a guardar la parte entera de la división.

III-B8. Guardar el resultado: El último procedimiento realizado es guardar el resultado de la operación, en caso de la división, la parte entera del resultado, resultado el cual debe ser almacenado en un segmento de datos en específico y en el mismo formato que están los operandos, es decir,con sus dígitos separados y al revés y por último se salta a guardar la parte entera de la división. La solución realizada se basa en ciclos de restas, estilo pseudo-división, dividiendo en cada ciclo el resultado por una potencia de 10, correspondiente a la posición en el segmento de datos, y siendo almacenado el valor de esta división entre el resultado y la potencia de 10 correspondiente, cada ciclo funciona igual que la implementación de la división, solo que se restan valores fijos en cada iteración, el procedimiento se termina cuando se resta de a 1 y el registro t0 contiene un 0.

IV. Resultados

Los resultados obtenidos luego de poner a prueba los procedimientos anteriormente mencionados en III-B, fueron satisfactorios y buenos a excepción de el caso cuando se divide un negativo con un positivo, o viceversa y la parte entera resulta ser 0, ya que no se mostrará el signo menos por pantalla. El resto de procedimientos funcionan bien, al menos con los casos puestos a prueba, ya que el ”voltear”los operandos funcionó para todos los casos probados. El escoger la operación también funcionó siempre, ya que siempre se saltó a la operación escogida. Cada operación funcionó de buena manera, entregando el resultado que correspondía, el cual fue comprobado con calculadora, lo único que cabe resaltar es que al usarse ciclos tanto en la multiplicación y división, estas pueden tardar más tiempo que la suma y resta, proporcionalmente al tamaño de los operandos. El imprimir resultados también funcionó siempre, indicando por pantalla el resultado. Finalmente el guardar resultados también funciona correctamente, almacenando el valor obtenido de buena manera y en el formato correspondiente. A continuación se adjuntan imágenes de algunos procedimientos:

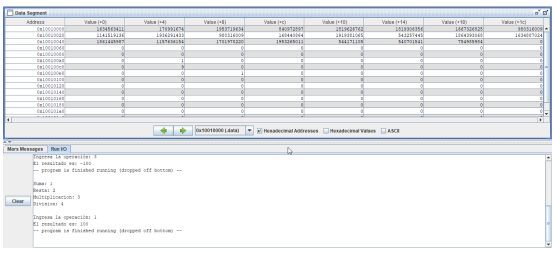

Figura 1: Imagen 1.

En la imagen anterior se realiza una suma de 10 y 90, y se puede observar como por pantalla muestra que el resultado es 100 y en el segmento de datos se guarda 001.

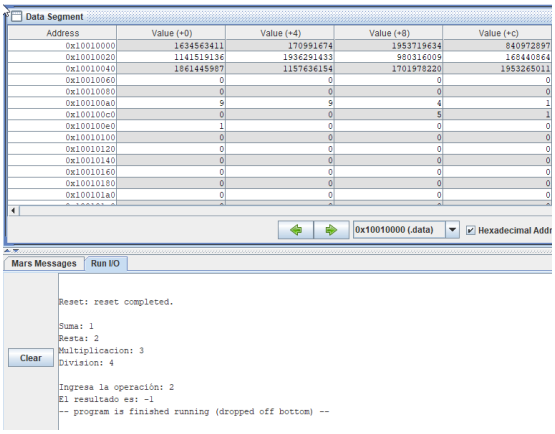

Figura 2: Imagen 2.

En la imagen anterior el procedimiento realizado fue restar 1500 a 1499, y se puede observar como por pantalla se muestra que el resultado es -1 y en el segmento de datos es almacenado un 1.

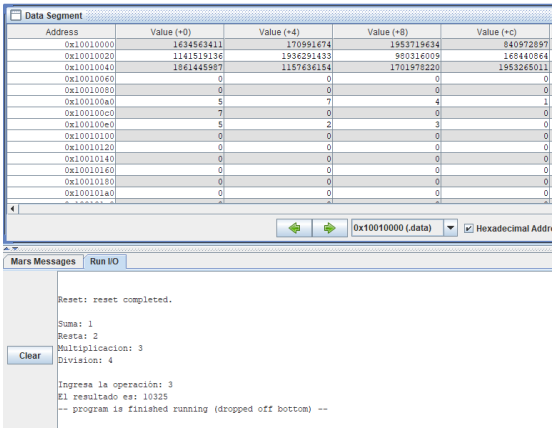

Figura 3: Imagen 3.

En la imagen anterior el procedimiento realizado fue multiplicar 1475 por 7, y se puede observar como por pantalla se muestra que el resultado es 10325 y en el segmento de datos es almacenado un 52301.

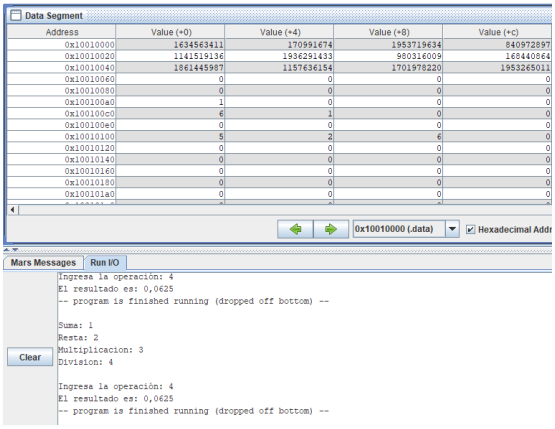

Figura 4: Imagen 4.

En la imagen anterior el procedimiento realizado fue dividir 1 entre 16, y se puede observar como por pantalla se muestra que el resultado es 0,0625 y en el segmento de datos de la parte entera es almacenado un 0 y en el de la parte decimal es almacenado un 5260.

V. Conclusiones

A pesar que programar en un lenguaje de bajo nivel como lo es el lenguaje ensamblador MIPS es una tarea difícil en la mayoría de casos, esto no fue un impedimento para poder realizar la actividad propuesta como laboratorio N°1. Por otra parte, la mayoría de los procedimientos que se requieren para completar los objetivos propuestos en este laboratorio están implementados y funcionan tal y como se planeaba desde un inicio, cumpliendo su objetivo y realizando lo que le corresponde. Concluyendo así que el laboratorio fue realizado de buena manera, debido a que desafíos y problemas propuestos fueron superados y ayudaron a poner en práctica los conocimientos adquiridos en las clases de teoría de la asignatura Arquitectura de Computadores.

Referencias

[1] C. T. G. Alberto Hernández Cerezo and R. A. Iglesias. (2010) Arquitectura mips. [Online]. Available: https://www.infor.uva.es/~bastida/OC/TRABAJO1_MIPS.pdf

[2] M. S. University. (2014) Mars - mips assembly and runtime simulator. [Online]. Available: http://courses.missouristate.edu/kenvollmar/mars/Help/MarsHelpIntro.html

[3] C. W. Kann. (2015) ¿qué es el lenguaje ensamblador? [Online]. Available: https://espanol.libretexts.org/Ingenieria/Implementacin_de_una_CPU_de_una_direccin_en_Logisim_(Kann)/02%3A_Lenguaje_de_la_Asamblea/2.01%3A_Qu_es_el_lenguaje_ensamblador%3F

[4] R. L. R. Thomas H. Cormen, Charles E. Leiserson and C. Stein, *Introduction to Algorithms*, 4th ed. MIT Press, 2022.