UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Organización de Computadores

Laboratorio 1: Instrucciones MIPS y Programación en Lenguaje Ensamblador

Nombre: Agustín Vera Rojas

Sección: L-1

Profesor: Viktor Tapia

Ayudante: Nicolas Henríquez

Introducción

En el presente informe se documentará las distintas actividades realizadas para el primer laboratorio correspondiente a la asignatura Organización de Computadores. Este informe presenta la siguiente estructura: introducción, marco teórico, desarrollo y resultados, conclusiones del laboratorio y se finalizará con las referencias utilizadas para realizar este trabajo. Para la realización del laboratorio se utilizará el software MARS, para realizar las distintas actividades que requieren programación en lenguaje ensamblador.

Para la realización de la actividad se declaran los siguientes objetivos:

- Predecir funcionamiento de un programa MIPS.
- Traducir programas escritos en un lenguaje de alto nivel a MIPS.
- Escribir programas MIPS que usan instrucciones aritméticas, de salto y memoria.
- Usar MARS (un IDE para MIPS) para escribir, ensamblar y depurar programas MIPS.
- Implementar algoritmos en MIPS para resolver problemas matemáticos de baja complejidad.

Principalmente se tiene como problema de este laboratorio resolver las diferentes actividades planteadas en el enunciado del laboratorio 1. Las actividades por resolver se separan en dos partes, la primera es predecir el funcionamiento de programas MIPS, donde dado un código se debe explicar su funcionamiento. La segunda parte corresponde a escribir programas MIPS para resolver problemas matemáticos de baja complejidad.

Marco Teórico

Con el fin de brindar un contexto sobre los diferentes temas en las distintas actividades que se abordaran en este laboratorio, se propone un marco teórico para que el lector se familiarice con los diferentes conceptos que se abordaran en este laboratorio.

MARS: “MARS, Mips Assembly and Runtime Simulator, ensamblará y simulará la ejecución de programas en lenguaje ensamblador MIPS” (MissouriState, 2014).

Lenguaje Ensamblador: El lenguaje ensamblador, o ASM, es un lenguaje de programación de bajo nivel que permite escribir programas que interactúan con el hardware del ordenador (Romero, 2022).

Instrucciones aritméticas: Son todas aquellas instrucciones que permiten realizar operaciones matemáticas, tales como: sumar, multiplicar, restar, etc. (Tapia, 2023).

Instrucciones de salto: Son todas aquellas instrucciones que permiten realizar distintos saltos en la ejecución de instrucciones de un programa, alteran el flujo en el cual se ejecuta el programa (Tapia, 2023).

Instrucciones de memoria: Son todas aquellas instrucciones que permiten acceder a la memoria principal, con el fin de poder leer información de la memoria o para escribir información en un sector especifico dentro de la memoria (Tapia, 2023).

Multiplicación: “Una multiplicación es una operación matemática que consiste en encontrar el resultado de multiplicar una cifra por otra. Multiplicar consiste en añadir o sumar un número varias veces, por ejemplo, la operación 2 x 3 equivale sumar tres veces el número 2, en ambas el resultado es 6” (Significados.com, 2023).

Syscall: “Una llamada al sistema o system call es un método utilizado por los programas de aplicación para comunicarse con el núcleo del sistema. En los sistemas operativos modernos, esto es necesario cuando una aplicación o proceso de usuario necesita transmitir a o leer información del hardware, de otros procesos o del propio núcleo” (IONOS, 2022).

Subrutinas: “Una subrutina es un conjunto de instrucciones que realizan una tarea concreta. Es un pequeño programa que puede incorporarse en un programa y accederse a él con una sentencia GOSUB, o puede ser externo al programa y accederse a él con una sentencia CALL. Los procesos comunes a menudo se guardan como subrutinas externas. Este método permite al programador acceder a ellos desde muchos programas distintos sin tener que volver a escribirlos” (IBM, 2021).

Desarrollo de la Solución

Inicialmente, para realizar una correcta solución para el laboratorio, se aprendió a utilizar el software MARS para solucionar las tareas que requerían soluciones con lenguaje ensamblador. Este aprendizaje se complemento con todo el material aprendido y brindado en las clases de teoría, en conjunto con la documentación del programa.

Con la finalidad de explicar el desarrollo de la solución, se explicarán las soluciones planteadas para las actividades “predecir el funcionamiento de programas MIPS” y “escribir programas MIPS” de forma separada.

1. Predecir Funcionamiento de Programas MIPS

Para dar solución a esta actividad se debió aplicar el conocimiento obtenido de las clases de teoría. Por cada ejercicio planteado en esta actividad se realizó la documentación de código brindado y se obtuvieron los valores finales de los registros. El desarrollo de esta actividad se verá reflejado en la sección de resultados.

2. Escribir Programas MIPS

Para el desarrollo de esta actividad, se realizó el primer ejercicio sin la utilización de subrutinas, además no se utilizó .data ni .text, la solución se presenta de esa manera, se realiza el loop correspondiente para simular el ciclo while y realizar las sumas correspondientes, para almacenar el resultado en un registro especifico. Se presenta un fragmento del cómo se llega a la solución para este ejercicio.

```
While:
        beq $s2, $s1, EndWhile
        add $s0, $s0, $s1
        addi $s1, $s1, 1
        j While
```

Figura 1: Ciclo While para ejercicio 1.

Para la realización del ejercicio dos se utilizó la traducción de una parte del código brindada en la actividad, la cual realiza la creación del arreglo, obtiene las direcciones necesarias de este y además provee la cantidad de números en el arreglo. Para realizar la suma de todos los números pares dentro del arreglo, se realiza un ciclo for para recorrer el arreglo accediendo a distintas memorias, con el fin obtener elementos, y luego sumar los elementos pares en un registro correspondiente. Se presenta un fragmento de la solución de este ejercicio.

```
For:
        beq $s2, $s1, EndFor
        sll $t1, $s2, 2
        add $t1, $t1, $s0
        lw $t2, 0($t1)
        andi $t3, $t2, 1
        beq $t3, $0, SumaPar
        addi $s2, $s2, 1
        j For
SumaPar:
        add $t0, $t0, $t2
        addi $s2, $s2, 1
        j For
```

Figura 2: Ciclo For para el ejercicio 2.

Para la realización del último ejercicio de este apartado, se debía tener en cuenta que no se podía hacer uso de instrucciones multiplicación, división y desplazamiento. Para calcular la multiplicación de dos enteros mediante la implementación de subrutinas, se realizaron distintas subrutinas alrededor del código con el fin de abarcar todos los casos que presenta la multiplicación. La idea general para resolver esta problemática fue sumar el operando1 consigo mismo tantas veces indique el número correspondiente al operando2, con ello, se obtiene un resultado y se almacena en un registro correspondiente. Se presenta un breve fragmento de la solución para esta actividad.

```
Operacion:
        beq $s1, $0, MultiplicarPorCero     # Si el operando 2 es
        addi $t1, $s0, 1                   # Suma 1 al operando
        beq $t1, $t0, MultiplicarPorUno     # Si el operando 1 es
        jr $ra

MultiplicarPorCero:
        # Como el operando 2 es igual a 0, la multiplicacion es
        addi $s2, $0, 0      # $s2 = $0 + 0 --> $s2 = 0
        j Exit

MultiplicarPorUno:
        # Como el operando 1 es igual a 1, la multiplicacion es
        addi $s2, $s1, 0     # $s2 = $s1 + 0 --> $s2 = operando
        j Exit

Multiplicacion:
        # Se verfica si el operando 1 es negativo, en caso de no
        bltz $s1, MultiplicacionConSegundoOperandoNegativo
        beq $t0, $s1, Exit     # Si el iterador es igual al operan
        add $s2, $s2, $s0      # $s2 = $s2 + $s0 --> realiza la mu
        addi $t0, $t0, 1       # avanza el iterador --> $t0 = $t0
        j Multiplicacion
```

Figura 3: Extracto de subrutinas para el ejercicio 3.

Resultados

Luego de realizar el desarrollo a las diferentes actividades planteadas, se procede a evaluar las soluciones propuestas en el apartado anterior. Para ello, se procederá a mostrar las diferentes actividades por separado.

1. Predecir Funcionamiento de Programas MIPS

Para este apartado se mostrarán imágenes del desarrollo “a mano” de cada actividad con sus respectivas respuestas.

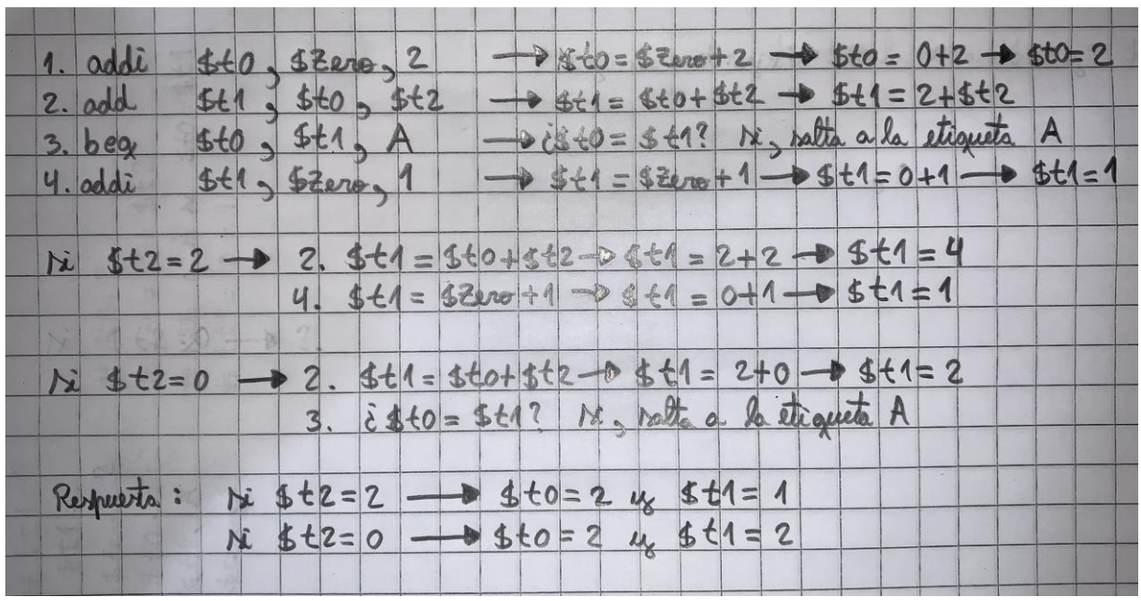

Figura 4: Documentación y respuesta ejercicio 1.

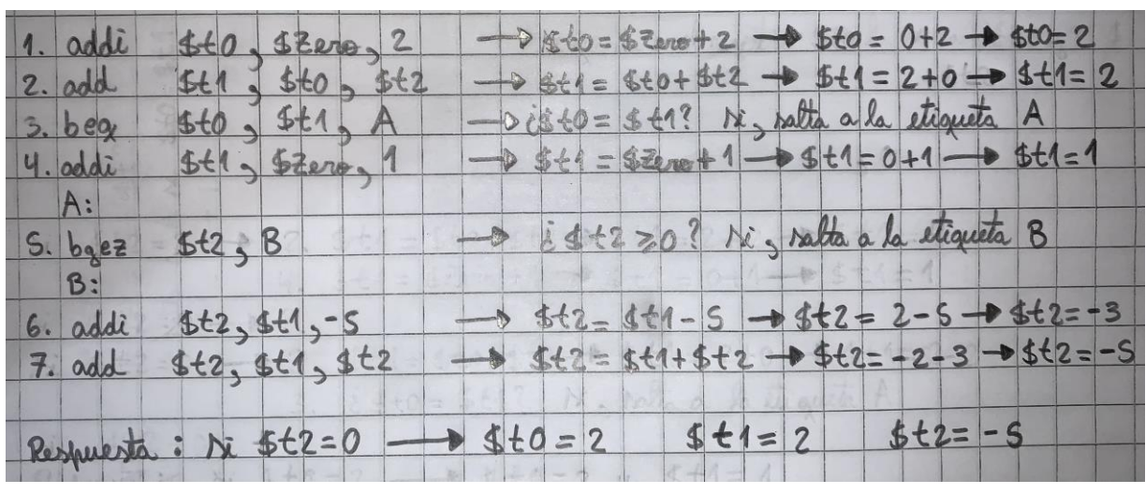

Figura 5: Documentación y respuesta ejercicio 2.

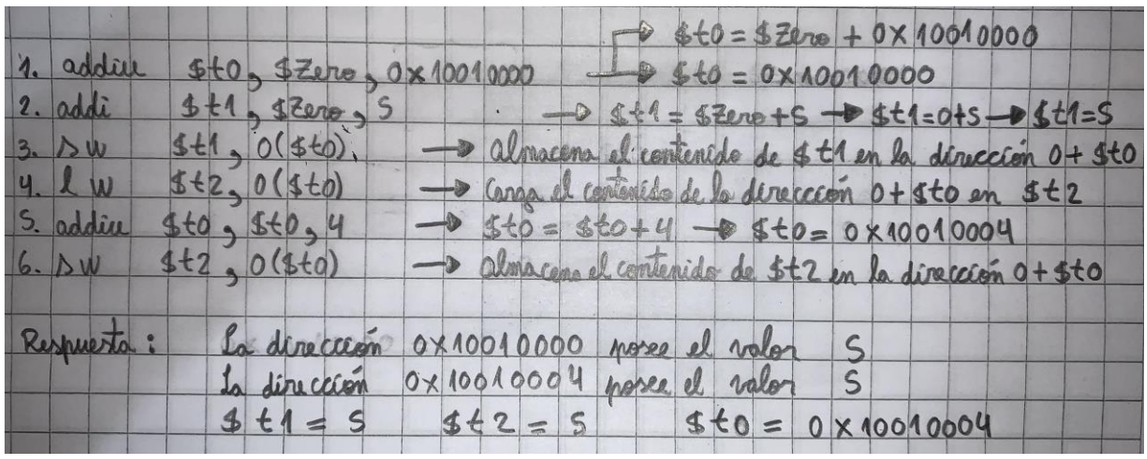

Figura 6: Documentación y respuesta ejercicio 3.

2. Escribir Programas MIPS

Para este apartado se mostrarán distintas tablas con los distintos resultados de las actividades, con diferentes evaluaciones a estas.

Variable	Registro	Valor inicial	Valor final	Valor final Hexa
a	$s0	0	45	0000002d
z	$s1	1	10	0000000a

Tabla 1: Distintos valores obtenidos en ejercicio 1.

Arreglo	Variable	Registro	Valor inicial	Valor final Hexa
[10, 22, 15, 40]	evensum	$t0	0	00000048
	i	$s2	0	00000004
	arrlen	$s1	4	00000004
[10, 14, 16, 40, 51]	evensum	$t0	0	00000050
	i	$s2	0	00000005
	arrlen	$s1	5	00000005

Tabla 2: Distintos valores obtenidos en ejercicio 2.

Variable	Registro	Valor inicial	Valor final Hexa
Operando 1	$s0	-10	f f f f f f f 6
Operando 2	$s1	-5	f f f f f f f b
Resultado	$s2	0	00000032
Operando 1	$s0	7	00000007
Operando 2	$s1	-8	f f f f f f f 8
Resultado	$s2	0	f f f f f f c 8

Tabla 3: Distintos valores obtenidos en ejercicio 3.

Conclusiones

En base a los objetivos planteados, se puede comentar que se logró cumplir todos los objetivos planteados en el laboratorio: se logra predecir el funcionamiento de distintos programas en MIPS, realizando ejercicios a mano alzada, se logra traducir distintos programas escritos en un lenguaje de alto nivel a MIPS a través del IDE MARS, utilizando distintos tipos de instrucciones según correspondió, además se realizan distintos algoritmos para resolver distintos problemas matemáticos de bajo nivel.

Se tuvieron ciertas complicaciones a la hora de realizar el ejercicio tres del apartado “escribir programas MIPS”, debido a que se tuvo que aprender que eran las subrutinas, y a la vez se aplicaron para resolver, además se tuvo que pensar en cómo abordar la actividad debido a la serie de restricciones que esta tenía.

Finalmente, se puede concluir que se realiza un trabajo adecuado, se realizan todas las actividades planteadas y se muestra un desglose de estas a través de presente informe en cada apartado que este posee, agregando que estas actividades proveen una serie de conocimientos al realizarlas, conocimiento que posiblemente será empleado en las próximas entregas del laboratorio.

Referencias

IBM. (28 de 2 de 2021). *Introducción a InfoSphere DataStage BASIC*. Obtenido de https://www.ibm.com/docs/es/iis/11.5?topic=basic-subroutines

IONOS, D. G. (6 de 5 de 2022). *System calls: ¿qué son y para qué se emplean?* Obtenido de https://www.ionos.es/digitalguide/servidores/know-how/que-son-las-system-calls-de-linux/

MissouriState. (8 de 2014). *MARS - Mips Assembly and Runtime Simulator*. Obtenido de http://courses.missouristate.edu/kenvollmar/mars/Help/MarsHelpIntro.html

Romero, I. (3 de 12 de 2022). *Profesional review*. Obtenido de https://www.profesionalreview.com/2022/12/03/ensamblador-vs-codigo-maquina/

Significados.com. (10 de 3 de 2023). *Multiplicación*. Obtenido de https://www.significados.com/multiplicacion/

Tapia, V. (2023). Lenguaje Ensamblador y Máquina. *Organización de Computadores – Segundo Semestre 2023*.