**Laboratorio 1 “TicTacToe en MIPS”**

**organización de computadores**

|  |  |
| --- | --- |
| Nombre: | Matias Coronado |
| Profesor: | Leonel Medina |
| Ayudante: | Ricardo Álvarez |
| Fecha de Entrega: | 23 de noviembre |

Santiago de Chile

2 - 2018

Tabla de contenido

[Capítulo 1. Introducción 1](#_Toc530751940)

[1.1 Enunciado del problema 1](#_Toc530751941)

[1.2 Motivación 1](#_Toc530751942)

[1.3 Objetivos 1](#_Toc530751943)

[1.3.1 Objetivo general 1](#_Toc530751944)

[1.3.2 Objetivos específicos 1](#_Toc530751945)

[1.4 Herramientas 2](#_Toc530751946)

[1.5 Estructura del informe 2](#_Toc530751947)

[Capítulo 2. Marco teórico 3](#_Toc530751948)

[Capítulo 3. Desarrollo 4](#_Toc530751949)

[Capítulo 4. Experimentos a realizar 7](#_Toc530751950)

[4.1 Resultados 7](#_Toc530751951)

[4.2 Análisis de resultados 8](#_Toc530751952)

[Capítulo 5. Conclusión 9](#_Toc530751953)

[Capítulo 6. Referencias 11](#_Toc530751954)

Índice de Figuras

Figura 3.1 Combinación ganadora ……………………………………… 5

Figura 4.4 Entrada prueba A ……………………………………………. 6

Figura 4.1 Salida “jugadas invalidas” prueba A ………………………... 6

Figura 4.3 Salida “Etapas” prueba A……………………………………. 6

Figura 4.2 Salida “Resultados” prueba A ………………………………. 6

Índice de tablas

**No se encuentran elementos de tabla de ilustraciones.**

# Introducción

## Enunciado del problema

Como primer laboratorio del curso organización de computadores, se pidió conformar un programa que sea capaz de manejar un camino de datos monociclo tipo MIPS, con el fin de leer instrucciones de este tipo. Además, se solicitó que el programa pueda manear un conjunto de instrucciones relacionadas al juego “Gato”, en donde se deberá indicar tanto las jugadas invalidas de los jugadores, como el estado final de la partida, además de un breve análisis de los elementos del camino de datos que fueron utilizados.

## Motivación

Si es que se llega a implementar correctamente el programa, teniendo en consideración tanto las restricciones como las funcionalidades solicitadas, será posible obtener un conocimiento más amplio sobre lo que camino de datos conlleva, para pulir y comprender de una mejor manera el contenido pasado en catedra. Se tiene como ventaja, el conocimiento previo que se tiene del juego TicTacToe (o mejor conocido “Gato”), la cual permitirá crear funciones acordes al comportamiento del juego.

## Objetivos

### Objetivo general

* Simular camino de datos para un procesador monociclo, determinando cuales unidades funcionales de este son utilizadas dentro de una secuencia de instrucciones tipo MIPS.

### Objetivos específicos

* Obtener un análisis de profundidad sobre el funcionamiento del camino de datos.
* Implementar correctamente el juego Gato, tomando en consideración los posibles errores que se puede presentar dentro del juego mismo.
* Mejorar capacidad de desarrollar en C, tanto en el sentido de simplicidad de funciones, como en el correcto manejo de memoria.

## Herramientas

Se procederá a trabajar con el lenguaje de programación C, utilizando el paradigma imperativo. Los archivos del programa serán creados a partir del IDE NetBeans 8.2, mientras que estos serán compilados por los elementos obtenidos a partir del implementador de compiladores MINGW 5.3.

## Estructura del informe

Este informe contempla 5 capítulos, el primero entregara una breve introducción al problema y su contexto, indicando la motivación que se tendrá para su desarrollo, además de los principales objetivos a abarcar y las herramientas que se utilizaran para este, el segundo capítulo se entregara un marco teórico que contenga de todas las explicaciones pertinentes sobre la terminología que se utilizara dentro del informe, el tercero indicara todos los elementos que se tomaron en consideración para la creación del programa, dentro del cuarto se presentaran las pruebas pertinentes a cada funcionalidad implementada, además de los posibles casos de entradas que puede abarcar el programa, finalmente, en el último capítulo se entregara una breve conclusión de lo trabajado.

# Marco teórico

Dentro de este capítulo se explicarán los principales conceptos y relaciones que se deben saber para entender el problema. Primero que nada, se debe saber que MIPS es una arquitectura diseñara de tal forma que se pueda leer un conjunto de instrucciones, en donde su funcionamiento se basa principalmente en el uso de registros, el cual es una estructura que se encarga de almacenar las variables de las instrucciones, las cuales, se pueden dividir en 3 tipos, tipo I (carga o almacenamiento de datos tal como SW y LW, y operaciones aritméticas con números inmediatos, tales como ADDI o SUBI), tipo J (Salto, como JUMP) y tipo R (Operaciones aritméticas, como por ejemplo ADD).

Por otro lado, MIPS hace uso de líneas y unidades de control, la primera hace uso de “señales de control” para direccionar las operaciones de las instrucciones dentro del programa, mientras que la segunda corresponde a las estructuras internas que compone a la lectura de las instrucciones, estas son, Instruction fetch (IF), la cual le indica al programa cual instrucción se debe leer además de identificar los componentes de esta, tal como el tipo de instrucción, o de cual operación se trata, Instruction decode (ID), se leen los registros solicitados, Execute (EX), donde se proceden a realizar las operaciones aritméticas, tal como el cálculo de direcciones de memoria, o de una simple suma, Memory Access (MEM), donde se accede a la memoria principal, con el fin de leer o escribir datos, y por último, Write back (WB), el cual se encarga de escribir los datos obtenidos dentro de los registros del programa.

El recorrido de una instrucción, por los elementos descritos anteriormente, se denomina “Camino de datos”, en donde la cantidad de estos que se pueden ejecutar dentro de un mismo ciclo esta normado por el “procesador”, el cual es una entidad que se encarga de supervisar las instrucciones del programa. Para el caso de este problema, se utilizará un procesador monociclo, el cual puede realizar solo un camino de datos por ciclo.

Todo lo descrito anteriormente se debe implementar utilizando el lenguaje C, el cual corresponde a uno de mediano nivel, el cual posee estructuras típicas de los lenguajes de alto nivel, pero compartiendo elementos de bajo nivel (como es el uso de memoria).

# Desarrollo

Para darle forma a la solución del problema, se decidió implementar una especie de lector MIPS dentro del programa, en el cual, sus componentes estarán subdivididos en diferentes funciones, tales como IF, ID, EX, MEM y WB. Estas se relacionarían con una instrucción que pasara por cada componente, con el fin de simular el camino de datos. Los componentes ID y MEM, deberán hacer uso de una estructura externa, que les permita almacenar tanto el valor de los registros (para el caso ID), y el de los datos de los arreglos en memoria (para MEM). Para en análisis del juego gato, se utilizará una estructura externa, que utilice los elementos de la instrucción, con el fin de saber cómo se va desarrollando el juego. De la observación anterior, se desprenden las siguientes estructuras:

* **Instrucción**: Estructura donde se almacenan los elementos relevantes de cada instrucción, tal como: Tipo, Operación, Registros a utilizar, valor de los registros. Se debe tomar en consideración que para cada una de estas existirán los registros R1, R2 y R3, los cuales pueden variar de valor a razón del tipo de instrucción que se le almacene.
* **Memoria de datos**: En esta se guardará los valores de los arreglos relacionados a cada registro, por medio de una matriz de enteros, en donde cada columna representará los elementos de cada registro.
* **Registros**: Esta estructura almacenara los valores de los registros dentro de una lista de enteros.
* **Lista de instrucciones**: Dentro de esta se almacenarán todas las instrucciones que se leyeron a partir del archivo de entrada, sin tomar en consideración los comentarios de este.
* **Gato**: Dentro de esta estructura, se guardarán todos los valores del juego que hayan sido captados dentro de cada instrucción, tanto como los registros relacionados con los jugadores, como las jugadas realizadas por cada uno.

El programa iniciara consultándole al usuario que ingrese el nombre del archivo de entrada, del cual, se le procederán a obtener las instrucciones MIPS que posea, para almacenarlo dentro de la estructura lista de instrucciones, la cual pasara por un ciclo while, normado por un ProgramCounter (encargado de indicar cual instrucción debe leer el programa), el cual aumentara hasta llegar al tope de la lista de instrucciones. Cada instrucción pasará por las funciones de las unidades de control, de donde se utilizará un contador para ver si la función hace uso de la unidad. Al final de cada ciclo, se le entregara la información de la instrucción a la estructura gato, para que esta almacene los elementos del juego que se encuentran adentrados en esta.

Por otra parte, para el análisis del estado del juego del gato, se separó las jugadas de cada jugador en arreglos diferentes, de los cuales se proceden a buscar cuales de ellos poseen los elementos de una combinación “ganadora”, de la forma representada en la figura 3.1.

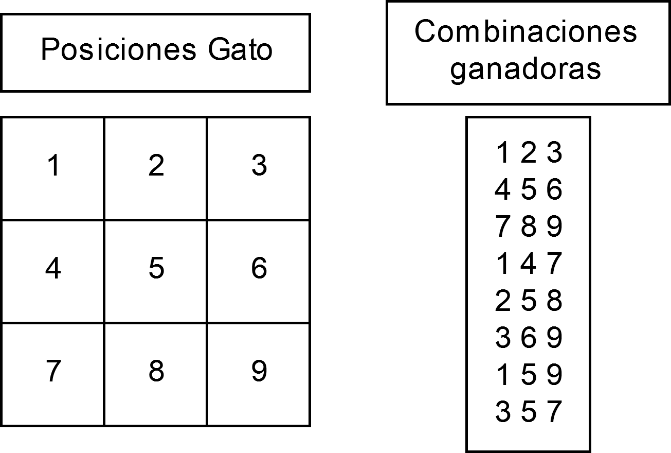


Figura 3.1 Combinación ganadora

Dependiendo de cuál de los jugadores tenga la combinación ganadora, se procederá a indicar cuál de estos gano. Si ninguno posee lo descrito anteriormente, se dirá que existe un empate. En similar analogía, si la suma de los jugadores no es igual a la cantidad de posiciones gato, se dirá que el tablero se encuentra incompleto.

Se destaca que en cada jugada del usuario debe existir una verificación de si esta es válida, tanto en el sentido de que no borre alguna jugada adversaria, o como que este no juegue dentro de un bloque que haya sido utilizado, de lo anterior, esto se realiza verificando constantemente los arreglos de jugadas de ambos jugadores, en busca de incongruencias al momento de eliminar o guardar una jugada.

# Experimentos a realizar

## Resultados

Con el fin de probar el correcto funcionamiento del programa, se decidió compilar y ejecutar el programa, para paso siguiente, probar una entrada, la cual intentara abarcar toda la gama de las posibles entradas que pueda llegar a tener el programa.

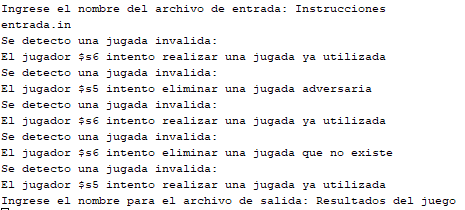
Prueba A:

Figura 4.1 Salida “jugadas invalidas” prueba A

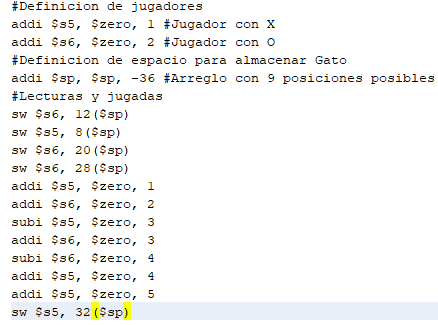
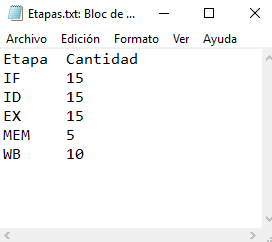
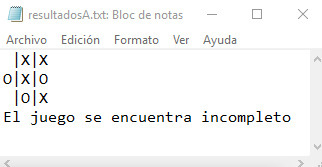


Figura 4.2 Salida “Resultados” prueba A

Figura 4.3 Salida “Etapas” prueba A

Figura 4.4 Entrada prueba A

## Análisis de resultados

A juzgar por los resultados obtenidos, se puede afirmar que el programa funciona correctamente, a raíz de los siguientes puntos:

* El programa pudo identificar que existían comentarios dentro del archivo de instrucciones, removiendo a estos dentro de la lectura de instrucciones.
* El programa pudo manejar, y reconocer las instrucciones ingresadas, como se puede notar en la figura 4.1.
* Se identificaron con éxito las jugadas invalidas de los usuarios, indicando además cuál de estos lo cometió.
* El archivo etapas pudo ilustrar el paso de las instrucciones por las diferentes etapas.
* El programa pudo identificar el estado final del gato, además de la posición de las jugadas de cada usuario.
* Se puedo simular correctamente el manejo de memoria, y registros, en relación con MIPS, en relación a que el conjunto final de salidas fueron los esperados.

# Conclusión

A modo de conclusión, se pudo alcanzar todos los objetivos propuestos al inicio del informe en donde se logró implementar tanto un MIPS correcto, como un juego gato que use elementos de este. En el proceso, se pudo comprender el camino de datos de las instrucciones, al tener que implementar las estructuras que conforman estos, lo cual, contrajo una mayor claridad de cómo funciona por debajo MIPS. Gracias al juego del gato se pudo observar de una manera más grafica el resultado de la interpretación de las instrucciones MIPS.

De lo anterior, se logró implementar exitosamente todos los elementos en el lenguaje C, en cuanto al manejo de memoria, y el orden que se le dio a tales elementos. De esto, se consiguió que el programa pudiera reconocer el nombre del archivo ingresado por el usuario, además de poder generar archivos de las correspondientes salidas del juego.

Se espera que en el próximo laboratorio (que puede ser de procesador multiciclo) se puedan utilizar elementos implementados dentro de este (monociclo), en donde se tiene la expectativa de que se podrá realizar sin problemas.

# Referencias

M. Martin. (1998). MIPS Instruction reference. 13/10, de University of Idao Sitio web: <http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html>

J.M. Mendías. (2003). Diseño de la ruta de datos y la unidad de control. 15/10, de

Universidad Complutense de Madrid Sitio web: www.fdi.ucm.es/profesor/mendias/512/docs/tema16.pdf