UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Laboratorio N 1

Integrantes:	Valentina Ligueño
Curso:	Organización de Computadores
Seccion	0-L-2
Profesor(a):	Leonel Medina Daza
Ayudante:	Ricardo Alvarez Zambrano

23 de Noviembre de 2018

Tabla de contenidos

1 Introducción **1**

1.1 Contexto 1

1.2 Problema 1

1.3 Motivación 1

1.4 Objetivos 1

1.4.1 Objetivo general 1

1.4.2 Objetivos específicos 2

1.5 Propuesta de solución 2

1.6 Herramientas 2

1.7 Estructura del informe 2

2 Marco teórico **3**

3 Desarollo **5**

4 Experimentos a realizar **7**

4.1 Resultados obtenidos 9

4.2 Análisis de resultados 10

5 Conclusiones **11**

Índice de figuras

1 Tablero inicial del juego El Gato. 3
2 Ejemplo de partida ganada 4
3 Ejemplo de archivo de entrada. 5
4 Archivo prueba formato 1 7
5 Archivo prueba formato 2 8
6 Archivo prueba formato 3 8
7 Experimento 1, archivos de salida. 9
8 Experimento 2, archivos de salida. 9
9 Experimento 3, archivos de salida. 9

1. Introducción

1.1. Contexto

A los estudiantes que cursando el curso de Organización de computadores se les solicita la implementación de un proyecto de laboratorio, el cual está enfocado en llevar a la práctica lo visto en clases, esto a través de la programación y la capacidad de resolver problemas bajo un cierto intervalo de tiempo.

1.2. Problema

Se solicita simular el famoso juego El gato, a través del lenguaje de programación C. Para esto se va a entregar un archivo de entrada que tendrá una serie instrucciones escritas en lenguaje MIPS. Se debe generar entonces, un programa que sea capaz de leer estas instrucciones, y logre simular jugadas del gato. Finalmente el programa entrega el ganador del juego y cuántas veces se pasó por cada etapa del camino de datos (IF, ID, EX, MEM, WB).

1.3. Motivación

El laboratorio está motivado por las ganas de interiorizar y entender los contenidos vistos en las clases de teoría, ya que se trabajará directamente con las instrucciones MIPS y más específicamente con los procesos internos que ocurren con las líneas de instrucciones de este lenguaje y como actúa cada instrucción en las etapas del camino de datos.

1.4. Objetivos

1.4.1. Objetivo general

El objetivo principal es resolver el problema planteado, con todo lo solicitado y con sus respectivas especificaciones, si es posible sin errores y que pueda resistir a diversos archivos de prueba que serán utilizados al momento de la evaluación.

1.4.2. Objetivos específicos

Como objetivos específicos se tiene primero lograr el correcto manejo de datos y memoria, para así lograr la máxima eficiencia del algoritmo, realizar la correcta lectura de los archivos de entrada entregando así los correctos datos en los archivos de salida, y finalmente entender los conceptos con los que se está trabajando y relacionarlo con la materia vista en clases.

1.5. Propuesta de solución

La propuesta a solución es tomar el archivo de entrada, y guardar todos sus datos en una estructura, que contendrá el nombre de la instrucción y sus respectivos parámetros, luego se van a pasar esas instrucciones por un algoritmo que simula las etapas del camino de datos para generar finalmente las jugadas y con esto el tablero final, dicho tablero se pasará por una función que busque que jugador ganador y finalmente se crearán los archivos de salida respectivos.

1.6. Herramientas

1. Lenguaje de programación ANSI C.

2. Buen editor de texto y conocimientos sobre el funcionamiento de MIPS.

1.7. Estructura del informe

El informe se va a estructurar de la siguiente manera: Primero se dará un marco teórico que ayudará a explicar algunos conceptos necesarios para el entendimiento del problema, luego de esto se dará paso al desarrollo donde se explicará como fue resuelto el problema, seguido de este se analizarán pruebas y resultados que se ejecutaron para el algoritmo desarrollado y finalmente se tienen las conclusiones que se tienen respecto al problema, la solución generada, los objetivos cumplidos y no cumplidos y lo que dejó la experiencia completa en el programador.

2. Marco teórico

Para la realización de este laboratorio, hay que tener ciertos conocimientos básicos sobre algunos temas en particular, mas en especifico sobre el funcionamiento de MIPS. MIPS es un lenguaje de bajo nivel, lenguaje ensamblador, quiere decir que por cada instrucción hay una línea, en la cual intervienen registros, direcciones de memoria, entre otros. Lo más importante a conocer sobre MIPS para el entendimiento del informe es comprender las instrucciones a utilizar y como funciona cada una de ellas, para efectos de este laboratorio se trabaja con las siguientes: addi, subi, y sw. Las primeras dos funcionan de la misma manera, la primera suma y la segunda resta, estas reciben 3 valores o parámetros, con este formato addi R1, R2, NUM, donde a el valor que existe en el registro R2 se le realiza la operación correspondiente con NUM que puede ser cualquier número, y el resultado se almacena en el registro R1, la siguiente instrucción recibe lo siguiente sw R1, NUM(R2), su funcionamiento consiste en tomar el valor del registro R1 y le otorga ese valor a un arreglo en una posición de memoria determinada por NUM(R2).

Para este laboratorio también es importante conocer cómo funciona este juego llamado Gato. El juego comienza con dos jugadores y un tablero vacío con 9 espacios en blanco para realizar jugadas.

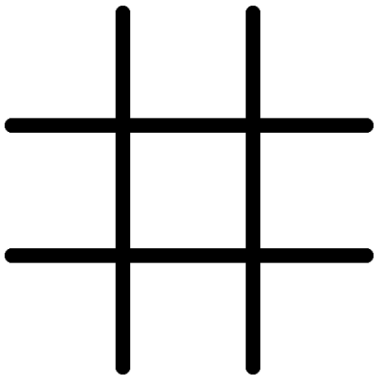


Figura 1: Tablero inicial del juego El Gato.

La figura 1 muestra cómo luce un tablero al momento de iniciar el juego, la idea es que los jugadores lo rellenen con jugadas, las cuales pueden ser ser O o X, donde cada jugador puede escoger solo una para el resto del juego. El objetivo del juego es rellenar el

tablero y lograr una linea de 3 jugadas iguales, es decir 3 veces O o 3 veces X. esta linea puede ser vertical, horizontal o en diagonal. El jugador que primero logre hacer la linea gana y acaba el juego, en la siguiente figura se muestra un tablero con una partida terminada y con un ganador.

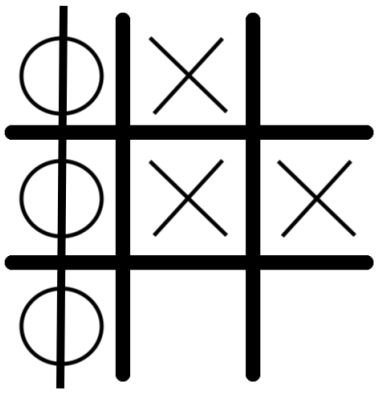


Figura 2: Ejemplo de partida ganada

Conociendo las instrucciones y el funcionamiento del juego solo falta relacionarlas, la asignación de jugadores va a ser a través de un addi, que le suma 1 al registro que representará al jugador 1 y 2 al registro que representa al jugador 2. Luego se hace una asignación de memoria a través de un addi que asignará el espacio de memoria necesaria para la creación del tablero. Finalmente las jugadas pueden ser a través de addi, con el formato addi registroJugador, $zero, lugarJugada, donde lugarJugada indica la posición en el tablero donde se va a realizar la jugada, pudiendo ser esta del 1 al 9, por los 9 espacios disponibles en este. La otra forma de jugada es con sw, con el formato sw registroJugador, lugarJugada($sp), lugarJugada aquí funciona un tanto distinto, ya que se comienza desde el cero, que representa el espacio 1 del tablero y luego se va de 4 en 4 para las siguientes jugadas, esto se debe a que en MIPS se debe correr 4 bytes para la siguiente posición, así, si se quiere escribir la jugada en la posición 3, lugarJugada será 8. Finalmente se tiene subi, que es para eliminar jugadas con la estructura subi registroJugador, $zero, lugarJugada, donde lugarJugada funciona igual que en addi, se debe tener la consideración que un jugador solo puede eliminar una jugada que él mismo realizó, si no es así simplemente el tablero se mantiene tal como está.

3. Desarollo

El desarrollo del laboratorio se llevó a cabo de la siguiente forma: Primero se comenzó con la creación de una función que fuera capaz de realizar lectura y recopilación de datos del archivo, el que contiene las instrucciones, que tendría que lucir como el siguiente:

```
#Definicion de jugadores
addi $s5, $zero, 1 #Jugador con X
addi $s6, $zero, 2 #Jugador con O
#Definicion de espacio para almacenar Gato
addi $sp, $sp, -36 #Arreglo con 9 posiciones posibles
#Jugadas de X
sw $s5, 0($sp) #Guardo X en la posicion 1
sw $s5, 8($sp)
sw $s5, 16($sp)
sw $s5, 24($sp)
sw $s5, 32($sp)
#Jugadas de O
sw $s6, 12($sp) #Guardo O en la posicion 4
sw $s6, 4($sp)
sw $s6, 20($sp)
sw $s6, 28($sp)
```

Figura 3: Ejemplo de archivo de entrada.

Luego de esto se procede con la creación de una función que trabaje con las instrucciones que se guardaron del archivo de entrada esta función primero va a tomar las primeras dos instrucciones, que son las que asignan los jugadores y va a cambiar el valor del registro para que se asocie al jugador correspondiente. Terminado esto se debe aumentar a 2 los valores de IF, ID y EX y WB ya que se ejecutaron dos instrucciones addi en la asignación. Para cada jugador se creó una estructura que contiene el registro que se le asignó, el jugador que es (1 o 2) y las jugadas realizadas. Luego de la lo anterior se va a tomar la tercera instrucción y se va a asignar el espacio de memoria correspondiente para el tablero y se va a aumentar en 1 más IF, ID y EX y WB ya que esta también se hace a través de un addi. Al terminar esto se procede con las jugadas, la función se encargará de ir recorriendo línea a línea las instrucciones correspondientes a las jugadas y se irán almacenando en el arreglo jugadas de la estructura asociada a cada jugador, así cada estructura solo almacenará las

jugadas que él ha realizado, así al terminar de recorrer las instrucciones en estos arreglos quedarán solo las jugadas finales de los jugadores, es decir si el borró alguna jugada son subi esa no se encontrará en estos arreglos, quedarán por lo tanto las jugadas que darán el estado final al tablero de juego.

Hecho esto se implementó la función que traspasa las jugadas al tablero, la cual lo llena con 1 y 2 según el valor que se le asoció al registro de jugador en primera instancia, después de lo cual revisa el tablero y busca que existe algún ganador, esto lo hace comprobando todas las lineas posibles hasta que exista una con tres elementos iguales, encontrando esta deja de revisar y guarda quien es el ganador. Si no existe ganador también lo detecta.

Finalmente se creó la función que genera los archivos de salida, siendo el primero el que entrega el tablero y el ganador del juego y el segundo el que escribe cuantas veces se pasó por cada etapa el camino de datos, este último tiene una extensión '.csv', que corresponde a un documento de Excel, esto para que se visualicen los valores de manera ordenada en una tabla. Con esto entonces, se da por finalizado el algoritmo.

4. Experimentos a realizar

En el enunciado del problema se especifica que existen tres tipos de formatos para los archivos de entrada y se muestra un ejemplo de cada uno, por lo tanto, las pruebas que se realizaron, se hicieron sobre estos 3 archivos que están en el enunciado.

El primer experimento que se realizó fue para el formato 1 de los archivos de entrada, en este todas las jugadas son realizadas mediante sw, el archivo es el siguiente:

```
p1.txt
1  #Definicion de jugadores
2  addi $s5, $zero, 1 #Jugador con X
3  addi $s6, $zero, 2 #Jugador con O
4  #Definicion de espacio para almacenar Gato
5  addi $sp, $sp, -36 #Arreglo con 9 posiciones posibles
6  #Jugadas de X
7  sw $s5, 0($sp) #Guardo X en la posicion 1
8  sw $s5, 8($sp)
9  sw $s5, 16($sp)
10 sw $s5, 24($sp)
11 sw $s5, 32($sp)
12 #Jugadas de O
13 sw $s6, 12($sp) #Guardo O en la posicion 4
14 sw $s6, 4($sp)
15 sw $s6, 20($sp)
16 sw $s6, 28($sp)
```

Figura 4: Archivo prueba formato 1

El segundo fue para el formato de entrada 2, en el cual todas las jugadas se hacen mediante addi, aquí también existen subi, es decir también se borran jugadas, por lo tanto en este archivo se prueban ambas funcionalidades.

El tercer experimento se realizó con el formato de entrada 3, en el cual todas las jugadas se hacen tanto con addi como con sw, dicho archivo tambíén incluye subi, así que aquí el programa hace uso de todas las instrucciones posibles para el juego.

```
p2.txt
#Definicion de jugadores
addi $s5, $zero, 1 #Jugador con X
addi $s6, $zero, 2 #Jugador con O

#Ejecucion de jugadas
addi $s5, $zero, 9 #Guardo X en la posicion 9
addi $s6, $zero, 2 # O en 2
addi $s5, $zero, 8 # X en 8
addi $s6, $zero, 1 # O en 1

#Con subi borro la jugada guardada en la posicion 9
#Si fuese $s6 esta instruccion no debiese tener efecto,
#ya que la realizo $s5
subi $s5, $zero, 9 # X borra 9
addi $s6, $zero, 6 # O en 6
addi $s6, $zero, 3 # O en 3
addi $s6, $zero, 7 # O en 7
```

Figura 5: Archivo prueba formato 2

```
p3.txt
#Definicion de jugadores
addi $s5, $zero, 1 #Jugador con X
addi $s6, $zero, 2 #Jugador con O

#Definicion de espacio para almacenar Gato
addi $sp, $sp, -36 #Arreglo con 9 posiciones posibles

#Lecturas y jugadas
sw $s6, 12($sp)   # O en 4
sw $s5, 8($sp)    # X en 3
sw $s6, 20($sp)   # O en 6
sw $s6, 28($sp)   # O en 8
addi $s5, $zero, 1 # X en 1
addi $s6, $zero, 2 # O en 2
subi $s5, $zero, 3 # X borra en 3
addi $s6, $zero, 3 # O en 3
subi $s6, $zero, 4 # O borra en 4
addi $s5, $zero, 4 # X en 4
addi $s5, $zero, 5 # X en 5
sw $s5, 32($sp)    # X en 9
```

Figura 6: Archivo prueba formato 3

4.1. Resultados obtenidos

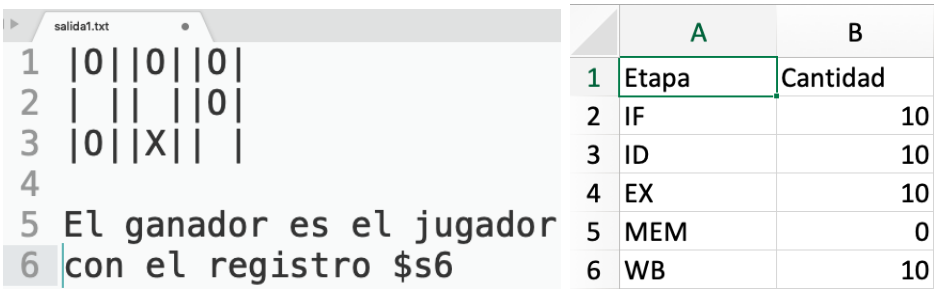
Resultado experimento 1

```
salida1.txt
1 |X||O||X|
2 |O||X||O|
3 |X||O||X|
4
5 El ganador es el jugador
6 con el registro $s5
```

	A	B
1	Etapa	Cantidad
2	IF	12
3	ID	12
4	EX	12
5	MEM	9
6	WB	3

Figura 7: Experimento 1, archivos de salida.

Resultado experimento 2

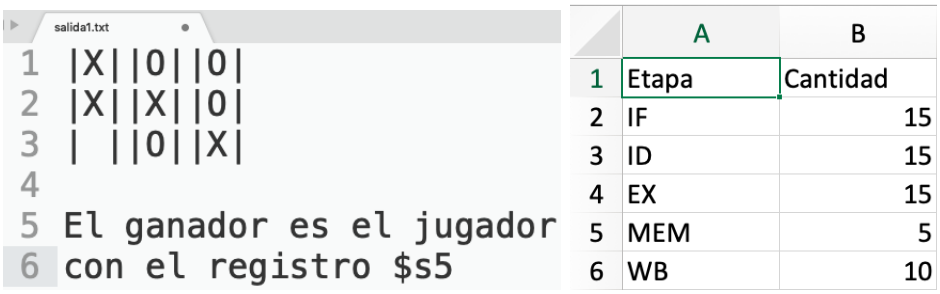


```
salida1.txt
1 |O||O||O|
2 | || ||O|
3 |O||X|| |
4
5 El ganador es el jugador
6 con el registro $s6
```

	A	B
1	Etapa	Cantidad
2	IF	10
3	ID	10
4	EX	10
5	MEM	0
6	WB	10

Figura 8: Experimento 2, archivos de salida.

Resultado experimento 3



```
salida1.txt
1 |X||O||O|
2 |X||X||O|
3 | ||O||X|
4
5 El ganador es el jugador
6 con el registro $s5
```

	A	B
1	Etapa	Cantidad
2	IF	15
3	ID	15
4	EX	15
5	MEM	5
6	WB	10

Figura 9: Experimento 3, archivos de salida.

4.2. Análisis de resultados

Análisis del archivo de entrada 1: Para los 3 addi existentes el valor de IF, ID, EX y WB será de 3, luego se tienen 9 sw, lo que implica que sus IF, ID, EX y MEM serán 9, por lo tanto en total se tiene IF = 12, ID = 12, EX = 12, MEM = 9 y WB = 3. Correpondiente con el archivo de salida 2 arrojado por el programa, y analizando las jugadas en sí, se tiene que el jugador 1 con registro $s5 llenó los espacios de tablero 1 - 3 - 5 - 7 - 9 y el jugador 2 con registro $s6 4 - 2 - 6 - 8. Lo que corresponde con el tablero del archivo de salida 1, el cual también coincide con la jugada ganadora en 1 - 5 y 9, asociada al jugador 1. Por lo tanto los resultados son los correctos.

Análisis del archivo de entrada 2: Aquí en ningún momento se utiliza sw, como está compuesto solo de addi y subi sumando un total de 10 instrucciones, las cuales pasan por las mismas etapas del camino de datos, se tiene que IF, ID, EX y WB valen 10, y como no se accede a memoria MEM tiene un valor de 0 lo cual corresponde con el archivo de salida 2, y analizando las jugadas en sí, hasta antes del subi el jugador 1 asociado a $s5 llevaba las jugadas 9 - 8 y el jugador 2 con el registro $s6 las jugadas 2 - 1, con el subi el jugador $s5 elimina su jugada en 9, por lo que queda solo con 8 y en lo que sigue del juego $s6 realiza 3 jugadas quedando finalmente con 2 - 1 - 6 - 3 - 7 y $s5 termina solo con 8. Esto corresponde al tablero del archivo de salida 1 en la figura 9 y finalmente el ganador es generando la linea ganadora con las posiciones 1 - 2 - 3. Por lo tanto los resultados son los correctos.

Análisis del archivo de entrada 2: Aquí se tienen entre addi y subi 10 instrucciones por lo que IF, ID, EX y WB toman valor de 10 y se tienen 5 para sw por lo que IF, ID, EX y MEM toman valor 5, sumando, los valores finales son IF = 15, ID = 15, EX = 15, MEM = 5 y WB = 10. Correspondientes con el archivo de salida 2, y analizando las jugadas en sí, hasta antes del primer subi el jugador 1 asociado a $s5 llevaba las jugadas 3 - 1 y el jugador 2 con el registro $s6 las jugadas 4 - 6 - 8 - 2, con el subi el jugador $s5 elimina su jugada en 3, por lo que queda solo con 1, luego $s6 realiza 1 jugadas más quedando con 4 - 6 - 8 - 2 - 3 pero también hace uso de un subi eliminando su jugada en 4, quedando entonces con 6 - 8 - 2 - 3. Para finalizar el juego $s5 realiza 3 jugadas más quedando con 1 - 4 - 5 - 9. Esto corresponde al tablero del archivo de salida 1 en la figura 9 y finalmente el ganador es $s5 con linea en la diagonal con 1 - 5 - 9. Por lo tanto los resultados son los correctos.

5. Conclusiones

Primero analizando el objetivo general que se propuso en un comienzo, este fue cumplido, ya que para los archivos que se han probado se han entregado las salidas correctas. Pero como siempre pueden existir pruebas que el programador no haya considerado y hagan que el programa entregue resultados erróneos. Donde existirá un funcionamiento inadecuado del programa es cuando se utilizan formatos de entrada incorrectos, por ejemplo cambiar el orden de la lista de datos, ya que el programa fue hecho para que en las primeras dos instrucciones esté la asignación de jugadores y en la tercera la del espacio de memoria para el tablero. Si la estructura no se sigue, el programa no funcionará o tendrá salidas incorrectas. Además otra ineficiencia que tiene el código es que espera almacenar todas las jugadas y luego las pone en el tablero, una forma más eficiente de funcionamiento sería que se vaya añadiendo cada jugada al momento de leerla e ir comprobando de inmediato si hay ganador o no, el programa actual está generado para leer todas las jugadas y entregarlas todas juntas al final y recién ahí buscar si hay ganador.

Analizando el algoritmo, no es muy eficiente porque existen demasiados bucles en el programa, casi todas las funciones utilizan a lo menos uno para su implementación, además hay demasiados arreglos estáticos por lo que al final de programa no existe mucha memoria que liberar. Se destaca la correcta lectura de los archivos, ya que independiente de lo que este contenga va a almacenar solo las instrucciones, pero si se va a añadir algo que no es instrucción se debe anteponer siempre un # para alertar al programa.

Finalmente a modo de análisis general no se sabe si el algoritmo fue desarrollado de la forma esperada, ya que existen varias formas de llegar al resultado final y no siempre se escoge la correcta o esperada. Pero la experiencia siempre aporta positivamente a los conocimientos del programador, ya que estos desafíos hacen crecer el conocimiento y la práctica de este para poder resolver todo tipo de problemas, además este proyecto en específico ayuda con el entendimiento de la teoría vista en clases, específicamente camino de datos y como funciona y ejecuta MIPS sus instrucciones. Por lo que lo positivo y rescatable de la experiencia supera lo negativo. Y por último se espera que lo no logrado en esta experiencia sea logrado en la siguiente, en específico el trabajo con bucles y la dependencia a arreglos estáticos.