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

Laboratorio N 1

Integrantes: Bastián Vera Palacios

Curso: Organización de Computadores

Sección 0-L-3

Profesor(a): Leonel Medina Daza

23 de Noviembre de 2018

Tabla de contenidos

1. Introducción

1.1. Contexto

La necesidad de buscar un medio de entreteción ha sido parte de la sociedad desde tiempos remotos. Asi mismo es donde bajo este contexto en persia, alrededor del siglo V, se dio la creación del juego TicTacToe, Tres en Linea, Ceros y Cruces, entre otros nombres. Este juego fue difundido por los musulmanes y posteriormente llevado a europa por comerciantes italianos. El juego tuvo un rapido reconocimiento, aunque tuvo un periodo oscuro donde por una asociación de este con rituales paganos macabros, este estuvo prohibido para todos los cristianos de fe catolica.
El juego consiste en una cuadricula de 3x3 dentro de los cuales cada jugador debe colocar su simbolo una vez por turno y no debe ser sobre una casilla ya jugada. El ganador del juego es el jugador el cual logre realizar una linea recta o diagonal entre 3 de sus simbolos

1.2. Problema

En este laboratorio se le solicitó a los estudiantes de ingeniería informática de la Universidad de Santiago de Chile realizar mediante una serie de algoritmos en lenguaje C, un programa que reciba un archivo de entrada con instrucciones en formato MIPS y que permita generar una partida del juego gato, es decir, poder entregar quien fue el ganador de la partida, especificar el caso en que exista un empate o que las instrucciones hayan generado una jugada invalida ,todo esto con el fin de poder medir las habilidades adquiridas durante el desarrollo de éste curso y a la vez entregar la cantidad de veces que se ha utilizado cada etapa del camino de datos.

1.3. Motivación

La motivación de este laboratorio es poder aprender de manera didáctica, como proceden algunas de las distintas instrucciones propias del lenguaje MIPS frente a las etapas pertencientes al camino de datos de un procesador monociclo.

1.4. Objetivos

1.4.1. Objetivo general

1. Simular el camino de datos de un procesador monociclo y determinar qué unidades funcionales del procesador son utilizadas en secuencias de instrucciones predefinida.

1.4.2. Objetivos específicos

1. Objetivo 3
2. Objetivo 2
3. Objetivo 1

1.5. Propuesta de solución

1.6. Herramientas

1. Herramienta 1
2. Herramietna 2

1.7. Estructura del informe

2. Marco teórico

2.1. Librerias

Entre las herramientas utilizadas para el desarrollo de este laboratorio, se encuentran las librerías de:

- stdio.h: Para la entrada, salida de datos y manejo de archivos.
- stdlib.h: Utilizada para el manejo de la memoria dinámica
- string.h: Utilizada para el manejo de concatenaciones de caracteres.

2.2. Punteros

Los punteros son elementos bastante utilizados durante el desarrollo del código debido a que permiten trabajar directamente a las direcciones de memoria de las estructuras de datos que tenemos. Se utilizan tanto como para lectura del archivo como para poder almacenar cada elemento de la instrucción en su estructura correspondiente.

2.3. Lectura y escritura de archivos

Para poder realizar la lectura de archivos anexos a los ingresados manualmente por terminal en nuestro código se utilizan funciones específicas llamadas por la librería stdio.h las cuales nos facilitan el acceso a éstos.

2.4. Memoria dinámica

Para un óptimo avance en este código se utilizó memoria dinámica durante la inicialización de las estructuras con el fin de que se pueda trabajar con los valores leídos en el archivo de manera más eficiente. Para esto se utilizó la función malloc() para poder asignar el espacio propio de cada instrucción y la función free() para ir liberando la memoria una vez que ésta ya ha sido utilizada.

2.5. Estructuras

Las estructuras son un tipo de dato utilizado para poder agrupar distintos tipos de información y manejar datos que sería muy complicado de realizar de manera más primitiva. En el desarrollo de este código se utilizan para la implementación de los nodos del árbol ya que así se puede acceder de manera inmediata a la información contenida en éstos.

3. Desarollo

Para el desarrollo de este laboratorio lo primero que se realiza es una estructura la cúal almacene de manera eficiente para un posterior trabajo la instrución a leer. Para esto se crea la estructura *Move* la cual posee 4 punteros de tipo char, los cuales almacenarán especificamente la instrucción, 2 registros, y un valor numérico (el cual solo se empleará para los registros *addi* y *subi*). Posterior a esto se realiza la lectura de cada linea del archivo de entrada, donde se revisa la primera palabra de la línea; si esta es una instrucción, dependiendo de su tipo, se realiza una lectura de los registro y datos asociados a este, para asi asociarlo a una estructura, la cual será almacenada posterior a este proceso en un arreglo que contendra todas las estructuras.

Una vez que todas las instrucciones se encuentren almacenadas de manera correcta en el arreglo de estructuras, se prosigue con la evaluación de cada una de estas y su respectiva función y a la declaración de un tablero, el cual consiste en un arreglo de caracteres con un largo de 9, el cual representara el tablero matricial. En primera instancia se revisan las primeras dos instrucciones y las define como los jugadores (jugadorX y jugadorO), ambas entragadas por una instrucción de tipo *addi*. En la siguiente linea se verifica que se pida el almacenamiento para las jugadas a insertar. Posterior a estas 3 revisiones principales, se recorre el arreglo leyendo el tipo de instrucción que posee cada una de las instrucciones y según ésta, define que acción realizará. Antes de realizar la acción, se le realiza la consulta al tablero si la jugada, ya sea insertar un simbolo o eliminar un simbolo, exista y que a la vez, esta sea propia del jugador que quiere realizar la acción.En el caso que se realize una jugada no valida, tal como la inserción en alguna casilla no existente o la sobreescritura respecto una jugada del otro jugador, el programa entrega un mensaje que la partida esta incompleta y procede al cierre del . Al final de la lectura de todas las instrucciones del arreglo, se realiza una comprobación respecto a si hay un ganador (indicando el nombre de este en caso de existir) o si se genera un empate. Para realizar el cálculo de cuántas veces pasa cada instrucción por cada etapa en el camino de datos, se realizan sumas de valor 1 por cada etapa que realiza de manera constante cada instrucción, es decir, se asume que si es una instrucción de tipo *addi* o *subi*, estas pasarán por las 5 etapas del camino de datos, en cambio, una instrucción

de tipo *sw* solamente utiliza 4 etapas del camino de datos (se excluye Mem). A modo de finalización, el programa entrega dos documentos, uno llamado *resultado.txt*, el cual posee la respuesta del ganador de la partida en caso de existir y el tablero con las jugadas realizadas, y un documento llamado *Etapas.txt*, el cual entrega la cantidad de veces que cada instrucción pasa por cada etapa del camino de datos del procesador monociclo.

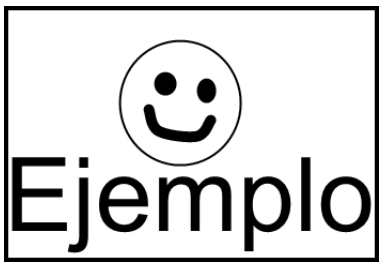

Figura 1: Ejecución de las instrucciones en las distintas etapas del datapath.

4. Experimentos a realizar

4.1. Resultados obtenidos

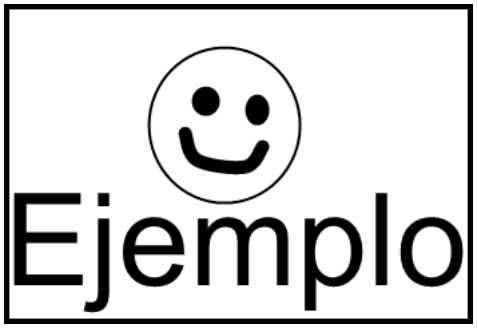

Figura 2: Resultados de los buffers para un branch no tomado.

4.2. Análisis de resultados

5. Conclusiones

Gracias a el desarrollo de este laboratorio se puede apreciar de mejor manera el manejo de los conocimientos ya adquiridos durante el desarrollo de las clases, ya que en cada uno de éstos, se profundizaron temas que ahora nos facilitan la compresión del camino de datos y como este es realizado por los distintos tipos de instrucciones que se pueden realizar. Referente a los algoritmos utilizados estos trabajan de manera correcta, gracias a una correcta implementación de la estructura *Move*, el cual tenía las mayores complicaciones en este código al momento de tener que trabajar las rotaciones con movimientos de punteros de una manera correcta, aunque existen situaciones las cuales podrían mejorar la efectividad del código tales como las mencionadas anteriormente referentes a una mejor búsqueda y una mejor implementación. Referente a la implementación del diccionario junto las tablas hash, se puede decir que logran el objetivo planteado generando una solución bastante cómoda de emplear, pero no lo satisfacen de la manera más eficiente que se pudo lograr. Respecto a objetivos planteados en este laboratorio, el programa funciona de manera correcta tanto en Windows como en Linux/OSX entregando los resultados deseados sin mayor complicaciones.

Bibliografía