**Organización de computadores**

**Laboratorio 1: PIPELINE**

|  |  |
| --- | --- |
| Nombre: Eduardo Pailemilla |  |
| Profesor: Leonel Medina |  |
| Ayudante: Ricardo Álvarez |  |
| Fecha de Entrega: 23/11/18 |  |

Santiago de Chile

2 - 2018

Tabla de contenido

[Capítulo 1. Introducción 1](#_Toc530771467)

[1.1 Enunciado del problema 1](#_Toc530771468)

[1.2 Motivación 1](#_Toc530771469)

[1.3 Objetivos 1](#_Toc530771470)

[1.3.1 Objetivo general 1](#_Toc530771471)

[1.3.2 Objetivos específicos 1](#_Toc530771472)

[1.4 Herramientas 2](#_Toc530771473)

[1.5 Estructura del informe 2](#_Toc530771474)

[Capítulo 2. Marco teórico 3](#_Toc530771475)

[MIPS 3](#_Toc530771476)

[Registros 3](#_Toc530771477)

[CPU 3](#_Toc530771478)

[Unidad de control 3](#_Toc530771479)

[Unidad aritmética lógica 3](#_Toc530771480)

[Datapath 4](#_Toc530771481)

[Capítulo 3. Desarrollo 5](#_Toc530771482)

[3.1 Leer instrucciones del archivo de texto 5](#_Toc530771483)

[3.2 Iniciar Registros 5](#_Toc530771484)

[3.3 Simuación del Datapath 5](#_Toc530771485)

[3.4 TICTACTOE 6](#_Toc530771486)

[Capítulo 4. Experimentos a realizar 7](#_Toc530771487)

[4.1 Resultados 7](#_Toc530771488)

[4.2 Análisis de resultados 8](#_Toc530771489)

[Capítulo 5. Conclusión 9](#_Toc530771490)

[Capítulo 6. Referencias 10](#_Toc530771491)

Índice de Figuras

[Figura 4.1 Conjunto de instrucciones asociadas a la jugada 3 7](#_Toc530769284)

[Figura 4.2 Prueba con jugada3.txt 7](#_Toc530769285)

# Introducción

Es necesario destacar que las pruebas realizadas para este programa han sido ejecutadas en el sistema operativo macOS.

## Enunciado del problema

Se pide hacer un programa en c capaz de simular la ejecución de un pipeline a través de un archivo de texto que contiene instrucciones MIPS.

Se debe realizar una deteccion de riesgos de datos en tiempo de ejecución para luego mostrar las soluciones pertinentes a cada uno, además de detectarlos es necesario hacer las debidas correcciones (ciclos NOP y forwardings) para que no existan problemas con los datos.

## Motivación

El desarrollo de este laboratorio se hace con el propósito de tener un acercamiento al cómo funciona un procesador del tipo pipeline, para este caso se utiliza como objeto de estudio un procesador para instrucciones MIPS. Estas instrucciones tienen muchas bondades para el ámbito académico.

Se busca comprender las fases por las cuales las instrucciones de este tipo transitan para poder ejecutarse, y de esta manera llevar lo aprendido teóricamente a la práctica.

## Objetivos

### Objetivo general

El objetivo general de este laboratorio es el de simular un procesador con pipeline, evaluando los posibles riesgos de datos y de control en tiempo de ejecución además de mostrar sus soluciones.

### Objetivos específicos

* El programa debe tener usabilidad. El usuario debe tener la capacidad de ingresar el nombre de los archivos de entrada.
  + Se deben entregar tres archivos de salida, uno que contenga los riesgos de datos, otro que contenga las soluciones a estos riesgos, y por ultimo uno que contenga el valor de los registros.
  + El programa debe reconocer multiples instrucciones: add, sub, addi, subi, addiu, bgt, beq, blt, bne, div, j, jal, jr, lw, mul, sw.

## Herramientas

* Lenguaje C para el programa.
* Estándar ANSI C para asegurar funcionamiento en LINUX y Windows.
* Se ha usado MACOS para realizar las pruebas de este programa.

## Estructura del informe

Los contenidos a lo largo de este documento son los siguientes:

* + Marco teórico: Contiene los conceptos necesarios para entender el funcionamiento de un procesador con pipeline e instrucciones MIPS.
  + Desarrollo: Contiene la explicación de cómo ha sido construido el programa para que cumpla con la simulación del pipeline y la identificación de los respectivos riesgos de datos con sus soluciones.
  + Experimentos a realizar: Se exponen los resultados obtenidos a través del software desarrollado mediante el uso de los archivos de textos dados como ejemplo, estos posteriormente se analizan para conocer sus características. En el análisis se busca conocer el significado de lo obtenido para cada archivo de salida.
  + Conclusiones: Finalmente se tienen las conclusiones de la información obtenida en el proyecto a través de la etapa de análisis de los resultados y su comparación con los objetivos propuestos. También se exponen los alcances y limitaciones que se han presentado a lo largo de este.

# Marco teórico

MIPS: Es un conjunto de instrucciones para un procesador, éstas están compuestas por 32 bits y pueden ser clasificadas en 3 tipos: Tipo I, Tipo R, Tipo-J, donde para este laboratorio solo se utilizan como tipo funcional las de Tipo I (sw, addi y subi).

Las instrucciones Tipo I tienen 6 bits para el opcode (define el tipo de instrucción), 5 bits para el registro rs, 5 bits para el registro rt y 16 bits para el valor inmediato que es usado en addi y subi como una constante y en sw como un offset.

Registros: Un registro en MIPS contiene 32 bits. Hay muchos registros en el procesador, pero sólo algunos de ellos son visibles en lenguaje ensamblador. Estos contienen información que puede ser operada de forma rápida.

CPU: Su nombre proviene de unidad central de procesamiento, y es la pieza de hardware encargada de interpretar las instrucciones (MIPS en este caso). Esta interpretación se realiza a través de sus componentes principales: CU o unidad de control en español, la cual es la encargada de controlar el flujo de datos a través del dispositivo, y por otro lado se tiene la ALU, la que se encarga de realizar procesos lógicos y aritméticos básicos cuando sea necesario.

Pipeline: Se le denomina pipeline a cuando el datapath monociclo se divide en 5 etapas las cuales se ejecutan de forma simultanea en cada ciclo de reloj, de esta manera se optimiza el rendimiento del procesador haciendo que mayoritariamente no existan etapas que estén vacias.

buffer: Un buffer es utilizado entre cada etapa del datapath para que no exista una perdida de datos. Estos buffers tambien otorgan la posibilidad de reconocer posibles reisgos de datos que se produscan entre instrucciones.

Riesgo de dato: Un riesgo de datos es aquel que se produce cuando una instrucción que está siendo ejecutada necesita el valor de un registro que aún no ha sido escrito en la memoria, para poder solucionar esta situación se utilizan comunmente las instrucciones “NOP” y el Forwarding como método de adelantamiento de datos.

Riesgo de CONTROL: Un riesgo de control es aquel que se produce cuando se realizan saltos a través de instrucciones tipo Jump o Branch’s. Cuando se hacen estos saltos las instrucciones que han ingresado al pipeline con posterioridad deben ser quitadas si existe una política de reemplazo “NOT TAKEN” (para el caso del branch), esto debido a que esas instrucciones no deben ser ejecutadas, ya que, la siguiente para hacerlo debiera ser la apuntada por la dirección del propio salto.

NOT TAKEN: Esta es una politica utilizada en las instrucciones branch, cuando es “not taken” quiere decir que el procesador siempre asume que el salto no se toma, por lo que hace ingresar las siguientes instrucciones de forma normal.

FLUSH: Operación la cual el procesador hace uso cuando es necesario reemplazar una instrucción que ya ha ingresado al pipeline por una del tipo “NOP”.

INSTRUCCIONES NOP: Este tipo de instrucciones es utilizada por el procesador cuando es necesario crear esperas. Estas esperas son producidas cuando existen riesgos de datos los cuales no pueden ser solucionados por forwarding o existen riesgos de control.

FORWARDING: Técnica usada por el procesador para adelantar datos que de otra manera no estarían disponibles en un determinado ciclo de reloj. Esta técnica es usada para darle solución a ciertos riesgos de datos.

Datapath: El datapath o camino de datos es una trayectoria por la que deben pasar las instrucciones. Este camino se divide en 5 etapas, y en el caso de un pipeline, estas etapas se ejecutan de forma paralela:

* **Instruction Fetch (IF)**: En esta primera etapa se va a buscar una instrucción a la memoria del programa, la dirección en donde se encuentra esta instrucción en memoria está determinada por el Program Counter (PC), el cual la almacena en 32 bits.
* **Instruction decode (ID)**: En esta etapa, el opcode de la instrucción que está siendo leída es interpretado por la unidad de control del procesador, de esta manera se envían señales de control que activan caminos por donde debe ir el flujo de información para que se haga una ejecución correcta de dicha instrucción.
* **Execute (EX)**: En esta etapa se utiliza la ALU del procesador, esto se hace con el propósito de realizar los cálculos que demanda la instrucción que está siendo ejecutada.
* **Memory access (MEM)**: Etapa que se utiliza para aquellas instrucciones que tienen acceso a la memoria, en MIPS esas instrucciones pueden ser “sw” o “lw”. En la memoria se pueden realizar tanto escritura de valores de registro como lectura de estos.
* **Write back (WB)**: Etapa que se utiliza cuando es necesario realizar una escritura de un valor en un registro, un ejemplo de esto es la instrucción addi, que realiza una escritura en su registro rt.

# Desarrollo

Para abordar la creación de este programa se ha decidido realizar una división en sub-problemas. La política que se ha usado para los Branch ha sido la de “NOT TAKEN”.

## Leer instrucciones del archivo de texto

Lo primero que se debe realizar para abordar este problema es el la lectura del archivo de texto que contiene la instrucciones MIPS. A través de la función llamada “cargarPrograma” se lee el texto, luego se retorna a través de la estructura “Programa”, luego, a través de la funcion “cargarProgramaMemoria” se llena una lista enlazada con estructuras del tipo instrucción, lase cuales poseen un nombre, los registros asociados, y valores necesarios para realizar operaciones de forwarding.

## Iniciar Registros

Se crea una estructura que contiene el nombre, valor y número de cada registro, y estos se inician en una lista enlazada y con un valor inicial de 0. Esto se hace con el propósito de realizar las operaciones pertinentes entre estos, las cuales están guiadas por los tipos de instrucciones.

## SIMULACION DEL PIPELINE

Una vez se tenga el texto cargado en memoria se procede a realizar la simulación del pipeline, donde primero, se crea un arreglo de instrucciones el cual hace de contenedor y permite que en cada “ciclo de reloj” se ejecuten más de una instrucción. Estos ciclos del reloj están representados por iteraciones dentro de un ciclo while.

Las etapas descritas a continuación son ejecutadas dentro de una función llamada “pipeline”, la cual de forma iterativa va desplazando a tráves del arreglo las instrucciones que va llamando a través de la funcion “instructionFetch”.

**Instruction Fetch (IF):** Se va en busca de una línea de instrucción que está almacenada en la estructura “Program”, esta línea de texto se le es pasada a una función llamada “makeLine”, la cual retorna un arreglo de strings con la información de la instrucción separada en el opcode y sus registros. Cada vez que una instrucción ingresa a la función “makeLine”, se suma el contador de IF en “ControlSignal” en una unidad.

**Instruction decode (ID):** En esta etapa se utiliza el retorno de la función “makeLine” para a través de la función “instructionDecode” retornar una estructura del tipo “Instruction”, la cual contiene los registros de la instrucción, el opcode y su valor inmediato. Esto se realiza para poder tener mejor acceso a cada instrucción y para llevar de esta manera un mejor seguimiento de cada una. Cada vez que una instrucción ingresa a la función “instructionDecode” se suma el contador de ID en “ControlSignal” en una unidad.

**Execute, MEM y WB**: Estos procesos se realizan en la función llamada “executeInstruction” debido a que estos dos últimos no son obligatorios como sí lo eran los anteriores. Se utiliza el retorno de la etapa de ID, y cada vez que una instrucción ingresa a esta función se le suma al EX en “ControlSignal” una unidad. También aquí se verifica el opcode de la instrucción para sumar o no al MEM o al WB en “ControlSignal” una unidad.

## TICTACTOE

El “TicTacToe” se hace de forma separada al Datapath pero utilizando las estructuras que ya estaban definidas para el manejo de las instrucciones y registros.

Para este sub-problema se utilizan los registros que han sido inicializados, esto se hace para poder asignar correctamente los valores presentados en las dos primeras líneas de cada archivo de texto a los jugadores, y además para utilizar estos mismos valores en las jugadas realizadas con las instrucciones “sw”, “addi” y “subi”.

Para los archivos de texto que contienen instrucciones del tipo “sw” y por lo tanto asignación de memoria, se utiliza el valor inmediato de esta para asignar la memoria del tablero.

Para los jugadores se crea una estructura “Player” que contiene el símbolo asignado y el registro que lo representa.

Para que las jugadas sean realizadas alternadamente por cada jugador, se crean dos listas enlazadas con las instrucciones de cada uno, las cuales se separan utilizando como distinción los registros que los representan. Una vez las jugadas estén separadas para cada jugador, se comienzan a realizar las jugadas en el tablero, esto se hace a través de las instrucciones almacenadas en cada lista y de forma iterativa hasta que estas se acaben. Para cada instrucción se toma el registro rt, se busca el nombre de ese registro en la lista de registros para consultar el valor que tiene y de esta forma saber si corresponde al jugador 1 o al jugador 2 (esto se hace a través del valor asignado al registro). Dependiendo del jugador que coincida con la búsqueda es que se marca el tablero con una X o O, o si es una instrucción “subi”, se elimina una jugada.

# Experimentos a realizar

## Resultados

Con propósitos de síntesis se muestran los resultados obtenidos para las instrucciones asociadas a la jugada3.txt de los ejemplos otorgados en el enunciado, esto debido a que en este archivo se contienen jugadas de todo tipo.

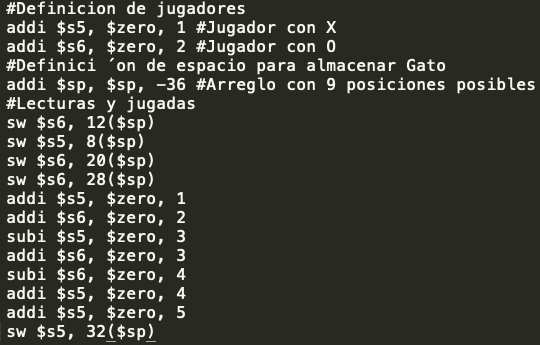


Figura . Conjunto de instrucciones asociadas a la jugada 3

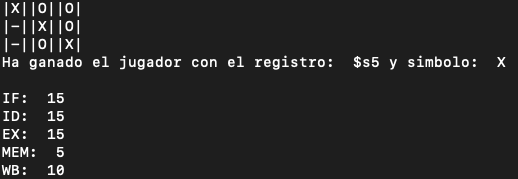


Figura . Prueba con jugada3.txt

## Análisis de resultados

Para el archivo de entrada utilizado en el los resultados (ver figura 4.1), se tienen 15 instrucciones de las cuales todas son del Tipo I, y de estas 15 existen 5 que son “sw”.

* Teóricamente todas las instrucciones deben pasar por las 3 primeras etapas del procesador, por lo que la sumatoria de estas 3 debiese ser 15 para todas, lo cual corresponde con lo mostrado en la imagen (figura 4.2).
* Como se tienen 5 instrucciones “sw”, las cuales pasan por la etapa de MEM, la sumatoria de esta etapa debe ser 5, lo cual también se corresponde.
* Diez de estas instrucciones escriben datos en un registro (instrucciones addi y subi), por lo que hacen uso de la etapa de “WB”, es por esto que la sumatoria de esta etapa debe ser 10, lo cual también se cumple.

Para el caso del “TicTacToe”, es necesario que las jugadas que se realicen respeten una serie de reglas, como el no poder borrar la jugada de otro jugador y tampoco es posible hacer una jugada sobre otra. Además hay que tener en consideración las condiciones de victoria.

Se procede a hacer el seguimiento a 3 instrucciones dentro del archivo de entrada:

“sw $s5, 8($sp)”, “subi $s5, $zero, 3”, “addi $s6, $zero, $3”. Si estas instrucciones se ejecutan en ese orden debería añadirse primeramente una X en la tercera posición del tablero, luego se debería quitar esa X y por ultimo añadir un O en esa posición, lo cual corresponde al símbolo que contiene el tablero de salida en esa posición.

# Conclusión

A través del analisis de resultados se puede concluir que el programa ejecuta de forma correcta ambas partes del software, tanto el datapath como tambien el “TicTacToe”, además los objetivos propuestos se han cumplido debido a que se ha logrado simular correctamente el camino de datos y el juego generando los dos archivos de salida propuestos.

A través de este laboratorio se ha aprendido de forma más exhaustiva el funcionamiento de un procesador monociclo MIPS, y también ha ayudado a tener un acercamiento a cómo estas instrucciones se ejecutan a lo largo de dicho procesador.

Los problemas más importantes presentados durante el desarrollo de este software ha sido la lectura del archivo para poder separarlo en instrucciones manejables, una vez que la instrucción pasa de una linea de texto a formato de estructura, es posible realizar las interacciones de forma más sencilla.

# Referencias

**Páginas web:**

Organization Computer Systems (recuperado el 23-11-18) desde:

<https://www.cise.ufl.edu/~mssz/CompOrg/CDA-proc.html>

Registros del camino de datos MIPS (recuperado el 23-11-18) desde:

<http://wikitronica.labc.usb.ve/index.php/Registros_del_camino_de_datos_del_Mips>

**Documentos en línea:**

Diseño de procesador monociclo (recuperado el 23-11-18) desde:

<http://www2.elo.utfsm.cl/~lsb/elo311/clases/c12.pdf>

Leonel Medina (27-10-2018) Organización de computadores, Laboratorio 1:

https://classroom.google.com/u/1/w/MjI0MjMzMTYzNjBa/t/all