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

LABORATORIO 2: PIPELINE

Organización de computadores

Nombre: Eduardo Pailemilla
Profesor: Leonel Medina
Ayudante: Ricardo Álvarez
Fecha de Entrega: 21/12/18

Santiago de Chile

2 - 2018

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

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

1.1 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 detección 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.

1.2 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.

1.3 OBJETIVOS

1.3.1 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.

1.3.2 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 último uno que contenga el valor de los registros.
    - El programa debe reconocer múltiples instrucciones: add, sub, addi, subi, addiu, bgt, beq, blt, bne, div, j, jal, jr, lw, mul, sw.

1.4 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.

1.5 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.

CAPÍTULO 2. 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 siguientes: add, sub, addi, subi, addiu, bgt, beq, blt, bne, div, j, jal, jr, lw, mul, sw.

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 (M.S Schmalz, Organization Computer Systems).

PIPELINE: Se le denomina pipeline a cuando el datapath monociclo se divide en 5 etapas las cuales se ejecutan de forma simultánea en cada ciclo de reloj, de esta manera se optimiza el rendimiento del procesador haciendo que mayoritariamente no existan etapas que estén vacías (D. Patterson, J. Hennessy 2014, pag. 286).

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 instrucciones "NOP" o "Forwarding", que para el caso de este desarrollo se usan ambas.

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. Para el caso del desarrollo de este software, se ha implementado una búsqueda de riesgos de control para poder darles solución a través del ingreso de instrucciones "NOP". Para el caso del "branch", es necesario realizar también un FLUSH en las instrucciones que ya habían ingresado al pipeline con el propósito de prevenir su ejecución (D. Patterson, J. Hennessy 2014, pag. 303)..

NOT TAKEN: Esta es una política utilizada en las instrucciones branch, y es la escogida para este programa, 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 de 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. Para el caso de este desarrollo se utilizan cuando en el riesgo está implicado una instrucción "lw" o cuando 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. En este programa se utiliza pasandole el valor del resultado de una instrucción a otra (D. Patterson, J. Hennessy 2014, pag. 308)..

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. Este program counter se representa por un número entero en el programa y como el índice de una lista enlazada de instrucciones..
- **Instruction decode (ID)**: En esta etapa, el opcode de la instrucción que está siendo leída es interpretado por la unidad de control del procesador. Para efectos de este desarrollo se utiliza para identificar los tipos de instrucciones.
- **Execute (EX)**: En esta etapa se utiliza la ALU, 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. Para el caso de este desarrollo la escritura y lectura se realiza sobre el registro "$sp", el cual está representado como un arreglo.
- **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.

CAPÍTULO 3. 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”.

3.1 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”, a continuación, a través de la función “cargarProgramaMemoria”, se llena una lista enlazada con estructuras del tipo instrucción, las cuales poseen un nombre, los registros asociados, y valores necesarios para realizar operaciones de forwarding.

3.2 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.

3.3 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 través del arreglo las instrucciones que va llamando a través de la función “instructionFetch”.

Instruction Fetch (IF): Se va en busca de una nueva instrucción que está almacenada en la lista enlazada de instrucciones, esta función recibe como parámetro la dirección de la instrucción para que soporte saltos a través de jumps o branch’s.

Instruction decode (ID): Se utiliza el campo "op" de la estructura instrucción para definir el tipo al cual pertenece, y de esta manera poder guiar los comportamientos de esta instrucción en las siguientes etapas.

Execute instruction(EX): En esta etapa se hace uso de la "ALU" para realizar las operación aritméticas necesarias para cada instrucción. Es en esta etapa en donde se decide si utilizar el valor de los registros o el uso de un valor de forwarding.

Memory Access(MEM): En esta etapa hace uso de las instrucciones "load" y "store", las cuales ingresan a memoria (representada por un arreglo) con el propósito de cargar o escribir un dato respectivamente.

Write back (WB): En la estructura instrucción existe un campo el cual almacena el resultado de la etapa EX, y es aquí en el WB donde ese resultado es escrito en el registro de destino.

3.3.1 Riesgos de datos.

Para poder ejecutar correctamente el pipeline, se hace una búsqueda a los riesgos de datos presentes en la ejecución del programa, y para poder encontrarlos se hace uso de fórmulas las cuales comprueban ciclo por ciclo si existe o no un riesgo de datos. Cuando el programa detecta un riesgo de datos, se entrega el resultado de la instrución que está siendo ejecutada a la que necesite dicho valor, y a esta última se le asigna un estado el cual indica que en vez de utilizar el valor de los registros que posea, utilice el valor que se le ha entregado.

3.3.2 Riesgos de control.

Para los riesgos de control y específicamente para las operaciones "jump", se hace una búsqueda en la etapa "ID" del pipeline, en la cual es posible saber si se trata o no de un salto. Al tratarse de un salto, se ingresa una instrucción del tipo "NOP" a la etapa "ID", se calcula la dirección de la siguiente instrucción y se hace ingreso de esta al pipeline por medio de la función instructionFetch. Esto se realiza un ciclo después.

Si se trata de una instrucción del tipo "Branch", se hace una búsqueda en la etapa de MEM del pipeline, ya que es en donde se decide si se realiza el salto o no. Si la condición de salto se cumple se utiliza la función "flush", la cual reemplaza todas las instrucciones pertenecientes a las etapas de "ID", "EX" y "MEM" por instrucciones "NOP".

CAPÍTULO 4. EXPERIMENTOS A REALIZAR

4.1 RESULTADOS

Se han realizado pruebas con los archivos que han sido facilitados a través de google classroom, los cuales han podido ser ejecutados con normalidad. Con propósito de ejemplo se muestran los valores obtenidos luego de la ejecución.

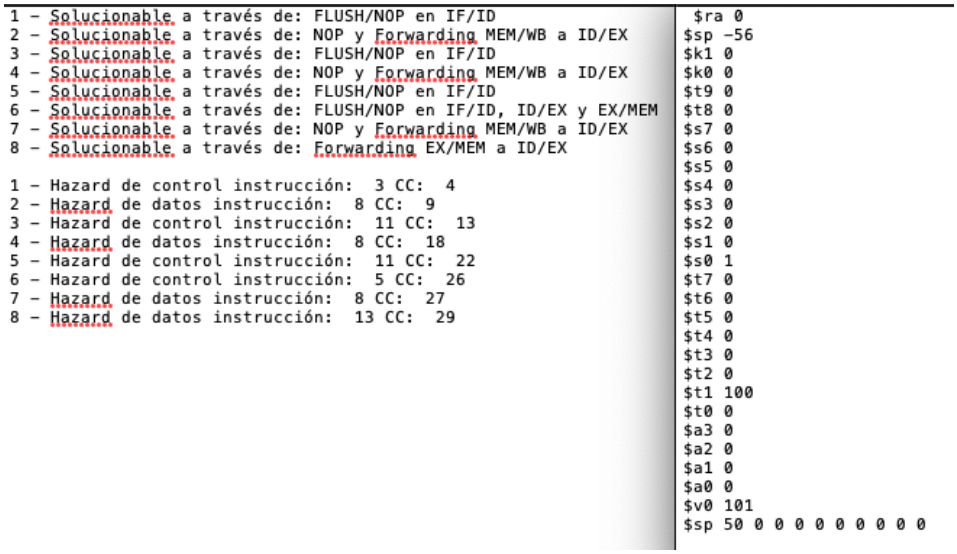

Figura 4.1 Resultados.

También se han hecho pruebas con múltiples instrucciones las cuales presentan riesgos de datos entre ellas, estas ejecuciones han sido comparadas con las respuestas obtenidas por un simulador de MIPS (Mars4_5). Con este propósito se han usado instrucciones tipo “load”, tipo “jump” (jal, jr y j), todos los branch propuestos (beq, bne, blt, bgt), además de aquellas más estándar como lo son las tipo R e I.

4.2 ANÁLISIS DE RESULTADOS

Para el archivo de entrada utilizado en el los resultados (ver figura 4.1), se tienen 8 riesgos de datos, y por lo tanto, también existen 8 soluciones para estos. Si estos resultados son comparados directamente con aquellos proporcionados como ejemplo, se puede hacer notar que existe una diferencia en el riesgo número 7. Haciendo un análisis de esta diferencia se deduce que este riesgo extra obtenido es producido por la última instrucción “lw” que entra en el pipeline, la cual es eliminada con un “FLUSH” al

momento que el último branch en ingresar es tomado, sin embargo, como para el caso de este programa se realiza la identificación de riesgos de datos para “lw” en la etapa del “ID” del pipeline, esta no alcanza a ser eliminada y es identificada como riesgo por el programa.

CAPÍTULO 5. CONCLUSIÓN

A través del análisis de resultados se puede concluir que el programa ejecuta de forma correcta las operaciones relacionadas con el cálculo aritmético de los valores en los registros, sin embargo, existe una pequeña diferencia en la identificación de riesgos debido a que las condiciones de reconocimiento varían un poco con respecto a las establecidas por las fórmulas de detección de riesgos, esto se debe a que en la implementación no se utilizaron buffers intermedios, si no que se hizo uso de las propias etapas del pipeline (IF,ID,EX,MEM,WB) para identificarlos.

A través de este laboratorio se ha aprendido de forma más exhaustiva el funcionamiento de un procesador con pipeline en MIPS, y también ha ayudado a tener un acercamiento a cómo estas instrucciones se ejecutan a lo largo de dicho procesador y de forma paralela. Se ha aprendido como los riesgos de datos y de control son solucionados en tiempo de ejecución a través del forwarding e instrucciones “NOP”.

Los problemas más importantes presentados durante el desarrollo de este software han sido la simulación de paralelización de las instrucciones dentro del pipeline, las cuales finalmente se han hecho en un arreglo dentro de un ciclo iterativo. Otro problema que se ha presentado es llevar a cabo la detección de riesgos y utilizar forwarding para solucionarlos, este pudo ser resuelto añadiendo a la estructura de instrucción un valor que permitiera almacenar un dato que proviene de otra ejecutada con anterioridad.

La lectura del archivo y la estructura de los registros no han presentado un gran problema, ya que se ha vuelto a utilizar los mismos procedimientos que para el laboratorio 1.

A raíz de lo dicho anteriormente, los objetivos y motivaciones presentados en la introducción, han sido completadas con éxito.

CAPÍTULO 6. REFERENCIAS

Libros:

Patterson, D. (2014). Computer organization and design. : Morgan Kaufman

Páginas web:

Organization Computer Systems (recuperado el 21-12-18) desde:

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

Registros del camino de datos MIPS (recuperado el 21-12-18) desde:

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

Documentos en línea:

Leonel Medina (22-12-2018) Organización de computadores, Laboratorio 2:

https://drive.google.com/file/d/1Aex4aMHyo3Fz1TkA-RDU2sxGNX0g2S19/view