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

Organización de computadores

Laboratorio 2: Pipeline

Integrantes: Andrés Muñoz

Curso: Organización de computadores

Sección: B-2

Profesor: Daniel Wladdimiro Cottet

Santiago - Chile
2016

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

No se encuentran elementos de tabla de ilustraciones.

CAPÍTULO 1. INTRODUCCIÓN

1.1 CONTEXTO

El ser humano está en constante búsqueda para mejorar la tecnología, de esta forma trabajar con el menor esfuerzo posible y sacar el máximo provecho al valioso tiempo. Una de las herramientas tecnológicas que está en constante mejoría, es el computador, aumentando la velocidad y poder de computo.

Existen diferentes cosas que afectan al rendimiento del computador, tales como, ISA, compilador, lenguaje de programación, entre otros. Dentro de estas diferentes cosas también se encuentra el procesador. En el trabajo anterior se analizó el procesador monociclo, donde la instrucción más larga determinaba el tamaño del ciclo para el programa. En el presente informe se analizará un procesador de 5 etapas optimizado con forwarding.

Principalmente se llevará a la práctica el concepto de pipeline, analizando los diferentes tipos de hazard que pueden surgir en el transcurso de un programa, como los hazard de control y de datos. Para esto se realizará un programa y se analizarán estos conceptos.

1.2 PROBLEMA

El problema consiste en realizar un programa que interprete el lenguaje de programación ensamblador MIPS, con el fin de entregar dos archivos de salida, en el primero se entregará una traza de cómo se modifican los buffers en cada ciclo. En el segundo archivo de salida se mostrará una traza como varían los registros 32 registros del programa en cada ciclo.

1.3 MOTIVACIÓN

La motivación para realizar este trabajo, es entender de una mejor manera todos los conceptos involucrados cuando se realiza pipeline, ya que es una materia muy difícil de entender debido a que se realizan diferentes etapas en las distintas partes del procesador al mismo tiempo (concurrentemente). Además, una vez realizado el programa se podrán comprobar ejercicios realizados sobre pipeline gracias a la traza de los buffers y registros.

1.4 OBJETIVOS

Objetivo general: Realizar un programa que interprete el lenguaje de programación ensamblador MIPS, simulando tener un procesador optimizado de 5 etapas con optimización de forwarding.

Objetivos específicos:

- Leer las instrucciones entregadas en un archivo, para posteriormente guardarlas y poder trabajar con ellas dentro del programa.
- Implementar las estructuras requeridas para la creación del programa, tales como instrucción, etiqueta, programa, buffers.

- Implementar una simulación todas las instrucciones MIPS que se utilizarán en el archivo de entrada.
- Entregar los archivos de salida en formato HTML para una mejor lectura de la información entregada en éstos.

1.5 PROPUESTA DE SOLUCIÓN

Para llevar a cabo la implementación del programa primero será necesario leer el archivo entregado con el código MIPS e ir guardando estas instrucciones en un arreglo. Luego la idea principal para poder representar el pipeline es crear un arreglo de 5 instrucciones, donde cada posición del arreglo identifica una etapa como se define a continuación:

- Índice 0: Instrucción que se ejecutará en la etapa IF.
- Índice 1: Instrucción que se ejecutará en la etapa ID.
- Índice 2: Instrucción que se ejecutará en la etapa EX.
- Índice 3: Instrucción que se ejecutará en la etapa MEM.
- Índice 4: Instrucción que se ejecutará en la etapa WB.

Estas instrucciones se cargarán al inicio de cada ciclo, donde la primera se cargará con una nueva instrucción mientras que las otras se cambian de posición avanzando en un índice. Luego, se procede a ejecutar cada una de éstas en su respectiva etapa, cabe recalcar que el orden de ejecución de estas será de atrás hacia adelante por problemas de escritura en cada uno de los buffers, por lo tanto, la primera instrucción en ejecutarse será la que esté en la etapa de WB mientras que la última será que esté en la etapa IF.

1.6 HERRAMIENTAS

- Sistema operativo: Windows 10 Education de 64 bits.
- Editor de texto: Atom, versión 1.16.0.
- Lenguaje de programación “C “.
- Compilador GCC, versión 5.3.0.

1.7 ESTRUCTURA DEL INFORME

En la primera parte del informe se encuentra el marco teórico donde se explicarán conceptos claves para entender lo que se realiza. Luego en el capítulo de desarrollo se describe como es elaborado el programa pedido, así como también se presentan experimentos realizados y el posterior análisis de estos resultados obtenidos. Finalmente, en el último capítulo se encuentran las conclusiones a partir de todo el proceso realizado en este trabajo.

CAPÍTULO 2. MARCO TEÓRICO

Lenguaje ensamblador MIPS: El lenguaje de programación MIPS, uno de los lenguajes de programación de más bajo nivel. Esto quiere decir que sus capas de abstracción son menores que otros lenguajes como por ejemplo C, Java, Python, etc. Este lenguaje permite un máximo de 3 argumentos por instrucción, por lo que una de las características es que para realizar una operación suma de 4 variables se necesitan un mínimo de 3 instrucciones.

Pipeline: Consiste en ir transformando un flojo de datos en un proceso comprendido por varias fases o etapas secuenciales, siendo la entrada de cada una salida de la anterior.

Hazard: Es un tipo de error en pipeline, estos pueden ser de 3 tipos, de datos los cuales sucede cuando hay dependencia de datos y estos datos son requeridos antes de que se modifique su valor, de control el cual corresponde cuando debe saltar a una etiqueta y existen instrucciones planificadas en ese momento, y los estructurales ocurren cuando la arquitectura del computador no permite ejecutar una instrucción.

Forwarding: Es una optimización en pipeline cuando ocurre un hazar de datos, aquí se toma el valor que necesario de un buffer para poder operar con él.

Branch taken: Es una optimización para los hazar de control, planificando de igual forma las instrucciones, pero cuando salta se aplica un flush a los buffers a instrucciones planificadas para que no se ejecute.

CAPÍTULO 3. DESARROLLO

3.1 ELABORACION DEL PROGRAMA

Para el desarrollo del programa se utilizan cuatro estructuras; Instruction, Label y Program. En base a estas estructuras es implementada la solución.

- **Instruction**: Esta estructura representa a una instrucción MIPS. A continuación, se presentan las características:
    - Número el cual hace referencia a la cantidad de argumentos de la instrucción.
    - String que contiene el nombre de la instrucción MIPS.
    - String que hace referencia al primer argumento de la instrucción.
    - String que hace referencia al segundo argumento de la instrucción.
    - String que hace referencia al tercer argumento de la instrucción.
- **Label**: Representa una etiqueta en MIPS. Tiene las siguientes características.
    - Número que representa la instrucción a la cual apunta la etiqueta
    - String el cual contiene el nombre de la etiqueta
- **Program**: Esta instrucción hace referencia a un programa escrito en MIPS. Tiene las siguientes características.
    - Número el cual indica la instrucción que sigue.
    - Número que indica la cantidad de instrucciones contenidas en el programa.
    - Número que indica la cantidad de etiquetas contenidas en el programa.
    - Arreglo de 32 string los cuales contienen los nombres de los registros. Esto es utilizado para encontrar el índice del registro que se debe modificar su valor.
    - Arreglo de 32 números que representan los valores de los registros en mips.
    - Arreglo idéntico al anterior usado para comparar la modificación un registro.
    - Arreglo de 10 caracteres que representan las líneas de control.
    - Arreglo de estructura de Instruction. Se guardan las instrucciones del programa.
    - Arreglo de estructura Label. Se guardan las etiquetas del programa.
    - Arreglo de estructura Instruction de largo 5, aquí es donde se guardan las instrucciones que se van a ejecutar en un ciclo.

    - Arreglo de estructura Buffer de largo 4, cada buffer representa unos de los 4 buffers existentes en la arquitectura de un procesador optimizado con forwarding.
    - Arreglo de números, este arreglo funciona como memoria heap.
    - Arreglo de números, este arreglo funciona como memoria stack.

- **Buffer**: Esta estructura representa un buffer dentro de la arquitectura del computador, en él se guardan lo datos necesarios al ejecutar las instrucciones en su respectiva etapa. Es por esto que es necesario tener cuatro de estas estructuras dentro de la estructura programa, donde el primero representa el buffer IF/ID, el segundo el buffer ID/EX, el tercero el buffer EX/MEM, y el cuarto el buffer MEM/WB.

Las funciones que se presentan a continuación son las principales del programa, además se muestran y explican las diferentes ideas de cómo funciona el programa. Las funciones creadas fueron separadas en tres archivos para un orden mejor del código. Éstos archivos se describen a continuación con sus respectivas funciones principales.

Funciones principales(funciones.c): Aquí se encuentras las principales funciones del programa, cabe recalcar que dentro de estas funciones son utilizadas funciones de los otros dos archivos.

- Inicializar programa (programInit): Esta función es la encargada de crear un puntero de la estructura Program. Posteriormente se cargan todos los datos necesarios para la "ejecución" de este mismo. Siendo más específico es este el momento donde se recorre el archivo de texto que contiene las instrucciones MIPS. Estas instrucciones son guardadas al interior del arreglo de instrucciones, del mismo modo se guardan las etiquetas contenidas en el archivo. A medida que se van agregando las instrucciones y etiquetas se va incrementado las variables respectivas que hacen referencia a la cantidad de instrucciones y etiquetas. También se lee el archivo de registros para obtener los valores de estos. Luego, se marca en cero el "program counter". Finalmente, al tener cargado todos los datos en la estructura "Program" se retorna la dirección de memoria de la estructura.

- Obtener índice de un registro (getIndexRegister): Esta función obtiene el índice de un registro entregado, la forma de obtenerlo es recorrer el arreglo de nombres de registros contenido en la estructura "Program" hasta que se encuentre el nombre del registro que se desea, para posteriormente retornar este índice.

- Modificar las líneas de control (setLinesControl): En esta función se modifican las líneas de control de la estructura "Program", específicamente se modifica el arreglo que contiene lo valores de estas líneas de control. Esta función esta implementada a

través de bifurcaciones para cada instrucción que existe en el programa, por lo que se modifican las líneas de control de forma diferente para cada una de éstas.

- Modificar las líneas de control (getPClabel): Esta función obtiene el program counter (PC) de la instrucción que apunta una etiqueta del programa. Esta función es utilizada cuando existe una instrucción branch o jump. La forma de obtener este valor es recorriendo el arreglo de etiquetas que existe en la estructura "Program", preguntando si el nombre de la etiqueta es el mismo de la cual se desea obtener el program counter. En caso de que sea el mismo nombre se retorna el valor necesario.

- Ejecutar programa (exProgram): Esta función es la encargada de ejecutar el programa. Lo que hace esta función es recibir a través de paso por referencia a la estructura "Program" cargada con los datos necesarios previamente por la función "ProgramInit" anteriormente descrita. Lo primero que realiza esta función es abrir los archivos de salida en modo de escritura, además, se inserta en estos archivos las etiquetas necesarias base para poder construir tablas en HTML. Posteriormente se inicia un ciclo que termina cuando no quedan más instrucciones en los buffers. Dentro de este ciclo, lo primero que se hace es revisar si existe alguna dependencia de dato y si es necesario utilizar forwarding, luego se cargan todas las instrucciones a ejecutar en cada etapa, esto se realiza modificando el arreglo de cinco instrucciones que está contenido en la estructura programa. Posteriormente se ejecuta cada una de estas instrucciones en la etapa correspondiente desde atrás hacia adelante del arreglo anteriormente mencionado. Cabe recalcar que cuando se va a ejecutar la etapa IF, se verifica si es necesario agregar una espera, en otras palabras, si se necesita una espera no se ejecuta la etapa IF y se resetea el buffer IF/ID, en caso contrario si se realiza la etapa IF. Finalmente, se imprime en los archivos la información modificada de ese ciclo en modo de tablas HTM. Una vez fuera del ciclo, se cierran los archivos introduciendo previamente las etiquetas HTML necesarias.

Funciones de escritura(fprint.c): En este archivo se encuentran las funciones relacionadas con la escritura de los archivos de salida. Además, se encuentra la función que abre un archivo HTML e ingresa el código base para escribir tablas, así como también se encuentra la función que cierra el archivo introduciendo previamente las etiquetas HTML finales.

Funciones de MIPS(mips.c): En este archivo se encuentran las diferentes funciones que simulan las instrucciones de MIPS. Aquí se pueden encontrar las funciones; add, addi, mul, beq, lw, sw, j, jal, jr. También se encuentra agregada la función empty, la cual es una instrucción vacía, necesaria para la implementación. Todas estas funciones mencionadas reciben como parámetro, la instrucción, el programa, y la etapa en la cual se debe ejecutar. Las instrucciones realizan una tarea diferente en cada una de su etapa, las cuales son ejecutadas cuando se requieren en la función exProgram.

3.2 EXPERIMENTOS A REALIZAR

Durante la elaboración del programa se realizaron diferentes pruebas para ir verificando si el programa hacía las diferentes tareas que debía realizar, de esta forma se iba avanzando sin errores de por medio y sin tener que volver a tocar esa parte del código.

En este capítulo se verán los diferentes experimentos a realizar para comprobar la funcionalidad total del programa, dejando de lado las pruebas realizadas anteriormente mencionadas. A continuación, se presentan las diferentes pruebas a realizar:

- Se ejecuta el programa con los archivos de entrada entregados por los ayudantes, estos archivos están publicados en la plataforma usachvirtual, de donde fueron obtenidos.
- Analizando el archivo mencionado en la prueba anterior, se sabe que la combinación de esas instrucciones MIPS entregan el factorial de un número. Es por esto que se pretende modificar el valor del número al cual se le calculará el factorial.

3.3 RESULTADOS OBTENIDOS

C94	0	0	120	0	5	3	0	0	22	22	0	2147483647	0	0	4515151	2147483647	20488191
C95	0	0	120	0	5	3	0	0	22	22	0	2147483647	0	0	4515151	2147483647	20488191
C96	0	0	120	0	5	3	0	0	22	22	0	2147483647	0	0	4515151	2147483647	20488191
C97	0	0	120	0	5/6	3	0	0	22	22	0	2147483647	0	0	4515151	2147483647	20488191
C98	0	0	120	0	6	3	0	0	22	22	0	2147483647	0	0	4515151	2147483647	20488191
C99	0	0	120/720	0	6	3	0	0	22	22	0	2147483647	0	0	4515151	2147483647	20488191
C100	0	0	720	0	6	3	0	0	22	22	0	2147483647	0	0	4515151	2147483647	20488191
C101	0	0	720	0	6	3	0	0	22	22	0	2147483647	0	0	4515151	2147483647	20488191
C102	0	0	720	0	6	3	0	0	22	22	0	2147483647	0	0	4515151	2147483647	20488191/720

Ilustración 1:Factorial de 6

C184	0	0	39916800	0	11	3	0	0	73	73	0	2147483647	0	0	4515151	2147483647	20488191
C185	0	0	39916800	0	11	3	0	0	73	73	0	2147483647	0	0	4515151	2147483647	20488191
C186	0	0	39916800	0	11	3	0	0	73	73	0	2147483647	0	0	4515151	2147483647	20488191
C187	0	0	39916800	0	11/12	3	0	0	73	73	0	2147483647	0	0	4515151	2147483647	20488191
C188	0	0	39916800	0	12	3	0	0	73	73	0	2147483647	0	0	4515151	2147483647	20488191
C189	0	0	39916800/479001600	0	12	3	0	0	73	73	0	2147483647	0	0	4515151	2147483647	20488191
C190	0	0	479001600	0	12	3	0	0	73	73	0	2147483647	0	0	4515151	2147483647	20488191
C191	0	0	479001600	0	12	3	0	0	73	73	0	2147483647	0	0	4515151	2147483647	20488191
C192	0	0	479001600	0	12	3	0	0	73	73	0	2147483647	0	0	4515151	2147483647	20488191/479001600

Ilustración 2:Factorial de 12

C199	0	0	479001600	0	12	3	0	0	85	85	0	2147483647	0	0	4515151	2147483647	20488191
C200	0	0	479001600	0	12	3	0	0	85	85	0	2147483647	0	0	4515151	2147483647	20488191
C201	0	0	479001600	0	12	3	0	0	85	85	0	2147483647	0	0	4515151	2147483647	20488191
C202	0	0	479001600	0	12/13	3	0	0	85	85	0	2147483647	0	0	4515151	2147483647	20488191
C203	0	0	479001600	0	13	3	0	0	85	85	0	2147483647	0	0	4515151	2147483647	20488191
C204	0	0	479001600/1932053504	0	13	3	0	0	85	85	0	2147483647	0	0	4515151	2147483647	20488191
C205	0	0	1932053504	0	13	3	0	0	85	85	0	2147483647	0	0	4515151	2147483647	20488191
C206	0	0	1932053504	0	13	3	0	0	85	85	0	2147483647	0	0	4515151	2147483647	20488191
C207	0	0	1932053504	0	13	3	0	0	85	85	0	2147483647	0	0	4515151	2147483647	20488191/1932053504

Ilustración 3:Factorial de 13

3.4 ANÁLISIS DE LOS RESULTADOS

Como bien se puede apreciar en las imágenes, la columna de la casilla marcada representa el valor del registro $s0 en cada ciclo.

En la ilustración 1, se observa que el valor obtenido es el correcto ya que el factorial de 6 es 720, es por esto que el primer experimento realizado se exitoso con el resultado esperado.

Luego se procede a revisar el segundo experimento en donde se modificó el archivo de entrada para calcular el factorial de un numero diferente, en los resultados solo se muestran 2 de estas modificaciones, las cuales corresponden cuando se calcula el factorial de 12 y de 13, de los cuales se espera que los resultados sean 479,001,600 y 6,227,020,800 respectivamente.

En la ilustración 2, se observa el resultado del factorial de 12, el cual coincide con el esperado, que en la ilustración 3, se observa que el resultado del factorial de 13 no es el correcto. Este último resultado tiene una explicación muy simple, ya que el valor máximo que puede tomar el tipo de dato long en C es “2,147,483,647” por lo que el resultado no “cabe” en este tipo de dato. Entonces al sumar otro bit empieza desde el “-2,147,483,648” por lo que tiene mucha lógica que el resultado sea 1,932,053,504 ya que el valor se excede dos veces, es decir 6,227,020,800 - 2,147,483,648 - 2,147,483,648 = 1,932,053,504. Obteniendo el resultado entregado por el programa.

Con lo anterior se puede decir que el programa funciona perfectamente y que el valor máximo que puede calcular es el factorial de 12 ya un número mayor excede la cantidad máxima soportada por el programa. Esto se podría mejorar teniendo otro tipo de dato para almacenar el resultado como un long long teniendo como máximo valor 9,223,372,036,854,775,807.

CAPÍTULO 4. CONCLUSIONES

En el presente trabajo se logró completar exitosamente la elaboración del programa que consiste en interpretar un archivo en lenguaje de programación assembler (MIPS), logrando simular todas las instrucciones contenidas en este archivo. Además, se lograron los objetivos de generar dos archivos, uno que contiene la traza de los registros y otro que contiene los datos de los buffers en cada ciclo del programa. Estos archivos son entregados en formato HTML para una mejor comprensión en la entrega de los datos pedidos.

Este laboratorio ha sido muy complejo ya que se ha tenido que analizar cada instrucción en el sentido de cómo funcionan en pipeline, ya que es muy distinto al ejecutar la instrucción en un procesador monociclo, debido a que se divide en cinco etapas la ejecución de ésta.

De este laboratorio se concluye que es mucho más eficiente un procesador de cinco etapas optimizado con forwarding para los hazard de datos, además, de ocupar el brach taken, ya que se planifican las instrucciones siguientes y se ejecutan en los diferentes componentes del procesador para las instrucciones. Cabe recalcar que es muy eficiente el brach taken en este laboratorio, ya que se realizan muchos llamados recursivos y solo en uno de ellos es necesario hacer un flush de las instrucciones cuando se cumple la condición del branch, lo que es muy diferente a hacer un flush en cada llamado recursivo, ya que en esta última ocasión se tendrían artos ciclos extras.

CAPÍTULO 5. REFERENCIAS

Arquitectura MIPS: Formato de la instrucción máquina: http://www.fdi.ucm.es/profesor/jjruz/EC-IS/Temas/Tema%205%20-%20Repaso%20ruta%20de%20datos.pdf

Definición de pipeline: http://definicion.de/pipeline/

Enunciado de laboratorio 2, Profesores: Alfonso Guzman & Daniel Wladdimiro,

Ayudantes: Pablo Ulloa & Ariel Undurraga, Usachvirtual.