INFORME PRÁCTICA 2

Alejandro Santorum Varela - alejandro.santorum@estudiante.uam.es
David Cabornero Pascual - david.cabornero@estudiante.uam.es
Prácticas Arquitecturas de Ordenadores - Pareja PM19
Universidad Autónoma de Madrid

30-10-2018

Contents

1 Introducción

Nos encontramos en la práctica 2 de Arquitecturas de Ordenadores, que tiene como objetivo implementar algunas mejoras al microprocesador MIPS, que tienen que ver con los riesgos que se generan en el micrprocesador al ser segmentado. En la primera parte nos centraremos en los riesgos de datos, y en la segunda en los riesgos de control en saltos condicionales.

2 Ejercicio 1: Riesgos de Datos

2.1 Adelantamiento de datos hacia la ALU

Se ha añadido una nueva unidad en el microprocesador llamada *Forwarding Unit* que tiene como misión adelantar los datos obtenidos en las etapas MEM y WB a la etapa EX cuando una instrucción posterior a otra que modifica un registro utiliza ese registro modificado en particular.
Además, se han implementado dos nuevos multiplexores a las entradas de los operados de la ALU para poder escoger correctamente el registro/dato que se quiere utilizar.

2.2 Adelantamiento interno en el banco de registros

En este apartado la dependencia es respecto a la instrucción que entró en el pipeline 3 ciclos antes. En este caso es necesario crear un adelantamiento interno en el banco de registros.
Se ha modificado el banco de registros para que escriba en flanco de bajada, en lugar de en flanco de subida. Esta solución funciona porque la instrucción que entró tres ciclos antes es capaz de actualizar el valor del registro modificado antes de que se lea en la instrucción que acaba de entrar en el pipeline.

2.3 Detección del caso en que una instr. LW carga un registro que es utilizado por la instrucción que le sigue

En este caso, como no es posible adelantar datos, generamos un ciclo de "stall", repitiendo las etapas IF e ID actuales e insertando una burbuja a modo de instrucción nop.
Para realizar esto se ha añadido una nueva unidad llamada *Hazard Detection Unit*, que implementa toda la funcionalidad descrita anteriormente.

3 Ejercicio 2: Riesgos de Control en saltos condicionales

En este ejercicio nos centramos en controlar las instrucciones que preceden a una instrucción de tipo branch.
No implementamos una unidad de predicción para los saltos condicionales, pero el método que seguimos es similar a una predicción estática no efectiva, lo que quiere decir que después de una instrucción branch van a entrar en el pipeline la/las instrucción/es que lo precederían. En el caso que el branch sea finalmente efectivo, esta/s instrucción/es serán eliminada/s del pipeline.

A la hora de decidir en qué etapa se determina si se salta o no había disparidad de opiniones. Por un lado había la opción de determinarlo en la etapa ID (como el precesador MIPS real) lo que conllevaría mover toda esta funcionalidad dos etapas antes, pero solo introduciría una nop en caso de salto. Por otro lado, existía la opción de determinar la efectividad del salto en la etapa EX, lo que solo nos obligaría a mover la puerta AND (señal branch and señal ZFlag) una etapa antes, pero provocando dos nops en caso de salto efectivo.

Nosotros hemos optado por implementar la segunda opción, ya que se modificaba en menos medida el microprocesador del ejercicio 1. En el caso de salto efectivo no se guardan los datos de las instrucciones de las etapas IF y ID en los registros de las etapas siguientes.
Para corroborar la correcta funcionalidad del microprocesador se ha realizado un pequeño programa en esamblador llamado **testBranch.asm**, que tendrán a disposición en la carpeta de la entrega.