Universidad Zaragoza
1542

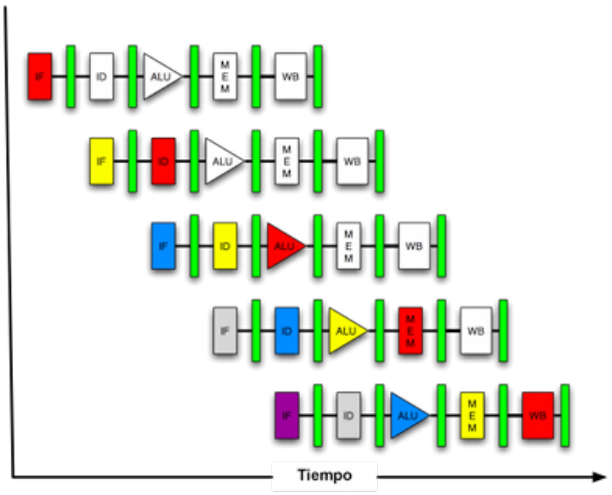
ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORES 2

2º DE GRADO EN INGENIERÍA INFORMÁTICA, CURSO 2020/2021

UNIVERSIDAD DE ZARAGOZA

PRÁCTICA 5

GESTIÓN DE RIESGOS DE DATOS



José Luis Briz

Javier Resano

Francisco J. Martínez

Logisim
una herramienta de diseño y simulación de circuitos lógicos

Universidad Zaragoza
1542

CONTENIDO

1 INTRODUCCIÓN

El objetivo de esta práctica es que repaséis los riesgos de datos y que seáis capaces de gestionarlo para garantizar que la ejecución en una máquina segmentada sea correcta y así reducir al máximo su impacto en el rendimiento. Las técnicas desarrolladas, son similares a las que aparecen en el Tema 5 de la asignatura (para el procesador segmentado MIPS), se incluirán en el procesador segmentado que habéis implementado en la Práctica 4.

Para poder realizar la práctica, debéis disponer de vuestro **procesador segmentado** diseñado en la segunda parte de la práctica anterior, **finalizado y funcionando correctamente**.

Importante: los riesgos de control se seguirán gestionando con saltos retardados.

Duración: 2 sesiones de laboratorio (4 horas)

2 TÉCNICAS A INCLUIR

PASO 1: Banco de registros de escritura/lectura en 1 ciclo

En este paso, modificaremos nuestro diseño en logisim para permitir que en un mismo ciclo se pueda escribir en un registro y leer el dato actualizado (es decir, tendremos un Banco de Registros de Escritura-Lectura en 1 ciclo). Para ello, el banco de registros debe trabajar con el flanco de reloj opuesto al resto del diseño, en este caso con flanco de bajada. Normalmente, esto se hace utilizando biestables sensibles al flanco de bajada en lugar de al de subida.

Si comparamos la ejecución del código que habíais preparado para la máquina segmentada de la práctica anterior y la ejecución en esta máquina (capaz de escribir y leer un registro en el mismo ciclo), ¿podríais eliminar alguna instrucción nop? ¿cuál sería la mejora obtenida? Justificad vuestras respuestas.

Y si aplicamos esta técnica al código que habíamos reordenado, ¿podríamos eliminar alguna instrucción nop? ¿cuál sería la mejora obtenida en ese caso? Justificad vuestras respuestas.

PASO 2: Anticipación de operandos

En este segundo paso, tenéis que implementar la técnica conocida como cortocircuito o anticipación de operandos, de forma parecida a como se hace en la máquina MIPS estudiada en el Tema 5.

Recordad que **el destino de los cortocircuitos será la etapa EX**, **y el origen serán los registros de salida de las etapas EX y MEM**. Para aplicar los cortocircuitos teniendo en cuenta esto, lo haremos de la siguiente forma:

- En la etapa ID se leerán los operandos teóricos del banco de registros
- En la etapa EX usaremos dos multiplexores, así como el hardware necesario para leer los operandos de dónde realmente sea necesario:
    - Los registros RegA o RegB
    - El registro a la salida del sumador (RegSumOut)
    - El registro MEM/WB de la máquina segmentada, que se encuentra a la salida de la etapa de memoria

Si comparamos la ejecución del código que habíamos preparado para la máquina segmentada de la práctica anterior y la ejecución en esta nueva máquina (que ahora capaz de escribir y leer un registro en el mismo ciclo, así como anticipar operandos de instrucciones anteriores), ¿cuántas instrucciones nop podríamos eliminar? ¿cuál sería la mejora obtenida con respecto a la máquina primera? Justificad vuestras respuestas.

PASO 3: Unidad de Detección de Riesgos y Detención

Esta técnica hardware se realizará de forma similar a lo estudiado en la máquina MIPS segmentada. Si una instrucción no tiene sus operandos fuente disponibles, se detiene su ejecución en la etapa ID (y la ejecución de la instrucción que le sigue) hasta que sepamos que dicha instrucción podrá leer sus operandos de forma correcta.

Para parar la ejecución se realizan dos acciones:

- Hay que detener las instrucciones que están en las etapas Fetch (IF) y Decode (ID): es decir, para ello, será necesario que, tanto el contador de programa como el registro IR no cambien.

- La instrucción que estaba en la etapa ID avanzará a la etapa EX, pero fijando el valor de las señales de control para que no pueda modificar nada. Es decir, hay que poner las señales de control adecuadas para que no modifique el estado visible a nivel de la arquitectura de lenguaje máquina (al programador). Dicho de otro modo: no se debe permitir que la instrucción escriba ni en memoria ni en el banco de registros.
  Nota: En el procesador MIPS se ponían todas las señales de control a 0, pero quizá en vuestro procesador (dependiendo de vuestra implementación en concreto) algunas señales se puedan activar a 0. En ese caso, obviamente, habría que ponerlas a 1.

¿Cuál sería la ventaja principal de añadir dicha técnica a vuestra máquina segmentada? Justificad vuestra respuesta. Para ello, podéis utilizar algunos ejemplos.

3 RESULTADOS

En primer lugar, debéis comprobar que el diseño funciona correctamente. Para ello, comprobad el funcionamiento del mismo programa de la práctica anterior, pero debéis modificarlo teniendo en cuenta las técnicas que habéis implementado en esta práctica. Además, debéis presentar una memoria comentando vuestro diseño y explicando qué hardware habéis añadido. Debéis explicar qué cortocircuitos soporta, y en qué casos se activarían. También debéis contestar a las preguntas que han ido apareciendo en los apartados anteriores.

4 MATERIAL A ENTREGAR

Deberéis entregar todo lo realizado dentro del plazo establecido (ficheros .circ y .txt, así como la propia memoria de la práctica en la que expliquéis en detalle lo que habéis hecho). La fecha de entrega se comunicará en moodle.

La forma de entrega será la misma que en las prácticas anteriores.