Paralelismo a nivel de instrucción

J. Daniel García Sánchez (coordinador)
David Expósito Singh
Javier García Blas
Óscar Pérez Alonso
J. Manuel Pérez Lobato

Arquitectura de Computadores
Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

1. Estructura del módulo

Este módulo está estructurado en dos lecciones:

1. **Introducción al paralelismo a nivel de instrucción**. Introduce el concepto de arquitectura segmentada, presentando como caso práctico el diseño de un procesador básico de cinco etapas. También introduce el concepto de riesgo y realiza una clasificación de los distintos tipos. Así mismo, presenta el concepto de operación multiciclo.

2. **Explotación del paralelismo a nivel de instrucción**. Presenta diversos aspectos de la explotación de las arquitecturas segmentadas como la planificación y desenrollamiento de bucles. También presta especial atención a las técnicas de predicción de saltos. Otros aspectos a los que se prestan atención son las técnicas de emisión múltiple y las limitaciones del paralelismo a nivel de instrucción. Finalmente se introduce el concepto de paralelismo a nivel de hilo.

2. Introducción al paralelismo a nivel de instrucción

Esta lección tiene la siguiente estructura general:

1. Introducción a la segmentación.

2. Riesgos.

3. Operaciones multiciclo.

2.1. Introducción a la segmentación

La segmentación (*pipelining*) es una técnica de implementación que se basa en dividir la ejecución de cada instrucción en varias etapas. Esta técnica no afecta (o afecta mínimamente) a la latencia de cada instrucción, pero permite incrementar el *throughput*, puesto que permite finalizar (idealmente) una instrucción por ciclo.

En el material puede observar los detalles de diseño de un procesador segmentado simplificado de 5 etapas:

- Captación (*Instruction Fetch* – IF). Lectura de instrucción y actualización del registro contador de programa.
- Decodificación (*Instruction Decode* – ID). Decodificación de la instrucción, lectura de registros, extensión de signo de desplazamientos y cálculo de posible dirección de salto.
- Ejecución (*Execution* – EX). Operación de ALU sobre registros y cálculo de dirección efectiva de salto.
- Memoria (*Memoria* – M). Lectura o escritura en memoria.
- Post-escritura (*Write-back* – WB). Escritura de resultado en el banco de registros.

2.2. Riesgos

Un riesgo es una situación que impide que la siguiente instrucción pueda comenzar en el ciclo de reloj previsto. Estas situaciones reducen el rendimiento de las arquitecturas segmentadas. Los riesgos pueden ser de tres tipos: estructurales, de datos o de control. La aproximación más simple (y menos eficiente) consiste en detener el flujo de instrucciones hasta que se elimina el riesgo.

Los riesgos estructurales se producen cuando el hardware no puede soportar todas las posibles secuencias de instrucciones. Esto se produce si dos etapas necesitan hacer uso del mismo recurso hardware. Las razones suelen ser la presencia de unidades funcionales que no están totalmente segmentadas o unidades funcionales no duplicadas. En general, los riesgos estructurales se pueden evitar en el diseño pero encarecen el hardware resultante.

Un riesgo de datos se produce cuando la segmentación modifica el orden de acceso de lectura/escritura a los operandos. Los riesgos de datos pueden ser de tres tipos RAW (*Read After Write*), WAR (*Write After Read*) y WAW (*Write After Write*). De estos tres, solamente los riesgos RAW pueden darse en una arquitectura de cinco etapas tipo MIPS. Los riesgos de tipo RAW pueden resolverse en algunos casos mediante el uso de la técnica del envío adelantado (*forwarding*).

Un riego de control se produce en una instrucción de bifurcación cuando no dispone todavía del valor sobre el que se debe tomar la decisión de salto. Los riesgos de control pueden resolverse en tiempo de compilación (soluciones estáticas) o en tiempo de ejecución (soluciones dinámicas).

Las soluciones estáticas ante los riesgos de control pueden ir desde la congelación del *pipeline* o la predicción prefijada (predecir siempre a no tomado o siempre a tomado), hasta el uso de bifurcaciones con ranuras de retraso.

Las soluciones dinámicas pueden usar una tabla histórica de saltos (BHT – *Branch History Table*) y una máquina de estados para realizar la predicción. De esta manera se tiene una máquina de estados asociada a cada entrada de la tabla.

2.3. Operaciones multiciclo

La asignación de un único ciclo a las operaciones de coma flotante requiere un ciclo de reloj extremadamente largo o el uso de una lógica de coma flotante muy compleja (con el consiguiente consumo de recursos). La alternativa a estas opciones es la segmentación de la unidad de coma flotante, por lo que estas instrucciones requerirán múltiples ciclos en la etapa de ejecución.

3. Explotación del paralelismo a nivel de instrucción

Esta lección tiene la siguiente estructura general:

1. Técnicas de compilación e ILP.
2. Técnicas avanzadas de predicción de salto.
3. Introducción a la planificación dinámica.
4. Especulación.
5. Técnicas de emisión múltiple.
6. Límites del ILP.
7. Paralelismo a nivel de hilo.

3.1. Técnicas de compilación e ILP

El paralelismo de instrucción es aplicable dentro de cada bloque básico (secuencia de instrucciones sin saltos). Sin embargo, la longitud media de un bloque básico es de 3 a 6 instrucciones lo que reduce bastante su posible aprovechamiento. Una técnica para mejorar el aprovechamiento del ILP dentro de un bucle se conoce como desenrollamiento de bucles.

El desenrollamiento de bucles consiste en entrelazar la ejecución de instrucciones de varias iteraciones de un bucle. Al tratarse de instrucciones no relacionadas no generan dependencias y permiten un mejor aprovechamiento de la arquitectura segmentada.

3.2. Técnicas avanzadas de predicción de salto

Tanto el desenrollamiento de bucles como la predicción de saltos permiten reducir el impacto de las bifurcaciones en el aprovechamiento de las arquitecturas segmentadas. Otras alternativas son las predicciones avanzadas de saltos: los predictores con correlación y los predictores por turnos.

La predicción con correlación mantiene una historia la historia de las últimas bifurcaciones para seleccionar entre varios predictores.

La predicción por turnos combina un predictor global y un predictor local y utiliza un selector para elegir entre los dos predictores.

3.3. Introducción a la planificación dinámica

La planificación dinámica se basa en la reordenación durante la ejecución de las instrucciones para reducir las detenciones. Aunque esto implica una mayor complejidad del hardware permite una mayor independencia de la arquitectura concreta y permite gestionar dependencias desconocidas en tiempo de compilación y retrasos no predecibles. No obstante, al permitir ejecución fuera de orden y finalización fuera de orden puede introducir riesgos WAR y WAW.

Existen dos técnicas de planificación dinámica: *scoreboard* y algoritmo de *Tomasulo*. La técnica de *scoreboard* detiene las instrucciones emitidas hasta que hay recursos suficientes y no hay riesgos de datos. El algoritmo de *Tomasulo* elimina las dependencias WAR y WAW realizando renombrado de registros.

3.4. Especulación

Para mejorar los resultados de la predicción de saltos, el siguiente paso es la especulación sobre el resultado de las bifurcaciones y la ejecución asumiendo que la especulación ha sido correcta. Esto requiere un mecanismo de tratamiento para los casos en que la especulación es incorrecta. Los componentes básicos de la especulación son la predicción dinámica de saltos, la especulación y la planificación dinámica. Para conseguirlo, se separa la finalización de una instrucción del el paso del resultado de una instrucción que produce un resultado a otra que lo usa.

3.5. Técnicas de emisión múltiple

Las técnicas de emisión múltiple se basan en la emisión de más de una instrucción por ciclo con el objetivo de alcanzar un CPI menor que uno (lo que equivale a más de una instrucción por ciclo). Este da lugar a tres soluciones: el uso de procesadores superescalares planificados estáticamente, procesadores superescalares planificados dinámicamente y procesadores VLIW.

3.6. Límites del ILP

Para evaluar los límites del paralelismo a nivel de instrucción se puede definir un procesador segmentado ideal. Si bien el ILP disponible en algunas aplicaciones puede ser alto, se puede considerar que de forma práctica el límite se encuentra entre 3 y 6 instrucciones por ciclo.

3.7. Paralelismo a nivel de hilo

En un procesador multi-hilo los distintos hilos (hardware) de ejecución comparten las unidades funcionales del procesador lo que hace necesario su replicación. Se distinguen tres aproximaciones: multi-hilo de grano fino, multi-hilo de grano grueso y multi-hilo simultáneo.

En el multi-hilo de grano fino se alterna entre hilos en cada instrucción. En el multi-hilo de grano grueso solamente se producen alternancias en las detenciones largas (como los fallos en caché L2). El multi-hilo simultáneo permite la ejecución de instrucciones de varios hilos de forma simultánea.