uc3m | Universidad Carlos III de Madrid

Paralelismo a nivel de instrucción

Arquitectura de Computadores

J. Daniel García Sánchez (coordinador)

Departamento de Informática
Universidad Carlos III de Madrid

Licencia Creative Commons

Predicción de bifuración

- Las **bifurcaciones** tienen un alto impacto sobre el rendimiento de los programas.

- Para **reducir** el impacto:
  - Desenrollamiento de bucles.
  - Predicción de saltos.
    - Tiempo de compilación.
    - Comportamiento de cada bifurcación de forma aislada.
  - Predicción avanzada de bifurcación:
    - Predictores con correlación.
    - Predictores por turnos.

Dependencia entre bifurcaciones

- Si la primera y la segunda bifurcación se toman, la tercera **NO** se toma.

Ejemplo

```
if (a==2) { a=0; }
if (b==5) { b=0; }
if (a!=b) { f(); }
```

Código generado

```
     addi x3, x1, -2
     bnez x3, L1
     add  x1, x0, x0
L1:  addi x3, x2, -5
     bnez x1, L2
     add  x2, x0, x0
L2:  sub  x3, x1, x2
     beqz x3, L3
```

- La predicción aislada para cada salto no aporta ventajas.
  - Es necesaria predecir dependiendo de lo ocurrido en otras bifurcaciones.

Predicción con correlación

- Se mantiene la **historia** de las últimas bifurcaciones para seleccionar entre varios predictores.
- Un predictor (m, n):
  - Usa el resultado de las m últimas bifurcaciones para seleccionar entre 2^m predictores.
  - Cada predictor de n bits.
- Predictor $(1, 2)$:
  - Resultado de última bifurcación para elegir entre 2 predictores.

Tamaño del predictor

- Un predictor (m,n) tiene varias entradas por cada dirección de bifurcación.
- Tamaño total:

$$T = 2^m \times n \times entradas_{\text{dirección}}$$

- Ejemplos:
  - $(0, 2)$ con 4K entradas $\rightarrow$ 8 Kb
  - $(2, 2)$ con 4K entradas $\rightarrow$ 32 Kb
  - $(2, 2)$ con 1K entradas $\rightarrow$ 8 Kb

Tasa de fallos

- **Actividad**
  1. *Lee* **Section C.3** – Reducing Branch Costs With Advanced Branch Prediction. **Solamente** subsección *Correlating Branch Predictors* (pág. 182–184 y fig. 3.3).
     - **Computer Architecture: A Quantitative Approach**. 6th Ed. Hennessy and Patterson. Morgan Kaufmann. 2017.
  2. **Aspectos clave**:
     - ¿Cómo se diferencian los predictores correlados de los predictores simples con el mismo tamaño?
     - ¿Cómo se diferencias los predictores correlados de los predictores simples con número ilimitado de entradas?
     - ¿Cuáles son las propiedades de un predictor **gshare**?

Predicción por turnos

- **Combina dos predictores**:
  - Predictor basado en **información global**.
  - Predictor basado en **información local**.
- Utiliza un selector para elegir entre predictores.
  - El cambio entre dos selecciones usa un contador con saturación (2 bits).
- **Ventaja**:
  - Permite comportamiento distinto para enteros y FP.
- **SPEC**:
  - **Benchmarks enteros** $\rightarrow$ predictor global 40%
  - **Benchmarks FP** $\rightarrow$ predictor global 15%.
- **Usos**: Alpha y AMD Opteron.

Intel Core i7

- **Actividad**
  1. *Lee* **Section 3.3** – Reducing Branch Costs With Advanced Branch Prediction. **Solamente** subsecciónn *The Evolution of Intel Core i7 Branch Predictor* (pg. 190–191).
     - **Computer Architecture: A Quantitative Approach**. 6th Ed. Hennessy and Patterson. Morgan Kaufmann. 2017.

Planificación dinámica

- El hardware **reordena** la ejecución de instrucciones para **reducir** detenciones manteniendo el comportamiento de flujo de datos y excepciones.
- **Ventajas**:
  - Código compilado optimizado para un pipeline ejecuta eficientemente en otro pipeline.
    - Simplificación del compilador.
  - Gestiona dependencias desconocidas en tiempo de compilación.
  - Permite tolerar retrasos no predecibles en tiempo de compilación (ej. Fallo de caché).
- **Desventaja**:
  - Mayor complejidad del hardware.
- Permite la **especulación hardware**.

Planificación dinámica

- **Efectos**:
  - Ejecución fuera de orden (OoO).
  - Finalización fuera de orden.
  - Puede introducir **riesgos** WAR y WAW.
- Separación de etapa **ID** en **dos etapas**:
  - **Emisión**: Decodifica la instrucción y comprueba riesgos estructurales.
  - **Lectura de operandos**: Espera hasta que no haya riesgos de datos y lee operandos.
- **Etapa de captación** (**IF**):
  - Capta en registro de instrucciones o cola de instrucciones.

Ideas de la planificación dinámica

1. **Lee** la sección **3.4** – Overcoming Data Hazards with Dynamic Scheduling. **Solamente** la subsección *Dynamic Scheduling: The idea* (pág. 193–195).
   - **Computer Architecture: A Quantitative Approach**. 6th Ed. Hennessy and Patterson. Morgan Kaufmann. 2017.
2. **Aspectos clave**:
   - ¿Cuál es la diferencia entre orden de emisión y orden de ejecución?
   - ¿Por qué la ejecución fuera de orden introduce nuevos riesgos de datos? ¿Cuáles?
   - ¿Cuáles son las interacciones entre finalización fuera de orden y las excepciones hardware?
   - ¿Qué es una excepción imprecisa?
   - ¿Cuáles son los requisitos derivados de tener múltiples instrucciones en ejecución?

Técnicas de planificación dinámica

- **Scoreboard**:
  - Detiene instrucciones emitidas hasta que hay recursos suficientes y no hay riesgos de datos.
  - Ejemplos: CDC 6600, ARM A8.

- **Algoritmo de Tomasulo**:
  - Elimina dependencias WAR y WAW con renombrado de registros.
  - Ejemplos: IBM 360, Intel Core i7.

Bifurcaciones y límites de paralelismo

- Al aumentar el paralelismo conseguido, las **dependencias de control** se convierten en problema.
  - La predicción de bifurcaciones no es suficiente.

- El siguiente paso es la **especulación** sobre el **resultado de las bifurcaciones** y la **ejecución** asumiendo que la **especulación fue correcta**.
  - Se capta, emite y ejecuta instrucciones.
  - Se necesita un mecanismo de tratamiento si la especulación no era correcta.

Componentes

- **Ideas**:
  - **Predicción dinámica de saltos**: Selecciona las instrucciones a ejecutar.
  - **Especulación**: Ejecución antes de que se resuelvan dependencias de control y capacidad para deshacer.
  - **Planificación dinámica**.
- Para conseguirlo se debe **separar**:
  - El **paso del resultado** de una instrucción a otra que lo usa.
  - La **finalización** de la instrucción.
- **IMPORTANTE**:No se actualiza el estado del procesador (registros/memoria) hasta que no se tiene confirmación.

Solución

- **Reorder Buffer** (**ROB**):
  - Cuando se finaliza una instrucción se escribe en el **ROB**.
  - Cuando se confirma su ejecución se escribe en destino real.
  - Las instrucciones leen datos modificados del **ROB**.

- Entradas del **ROB**:
  - **Tipo de instrucción**: branch, store, operación de registro.
  - **Destino**: Id de registro o dirección de memoria.
  - **Valor**: Valor del resultado de la instrucción.
  - **Ready**: Indica si la instrucción se ha completado.

CPI < 1

- CPI $\geq 1 \rightarrow$ Emisión de una instrucción por ciclo.
- Procesadores de múltiple emisión (**CPI** $< 1 \rightarrow$ **IPC** > 1):
  - Procesadores superescalares planificados **estáticamente**.
    - Ejecución en orden.
    - Número variable de instrucciones por ciclo.
  - Procesadores superescalares planificados **dinámicamente**.
    - Ejecución fuera de orden.
    - Número variable de instrucciones por ciclo.
  - Procesadores **VLIW** (*Very Long Instruction Word*).
    - Varias instrucciones empaquetadas en paquete.
    - Planificación estática.
    - Detección de riesgos por el compilador.

Enfoques de la emisión múltiple

- Varios enfoques posibles con emisión múltiple.
  - **Superescalar estático**. Planificación estática con ejecución en orden.
  - **Superescalar dinámico**. Planificación dinámica con ejecución fuera de orden.
  - **Superescalar especulativo**. Planificación dinámica con ejecución fuera de orden y especulación.
  - **LIW/VLIW/EPIC**. Planificación principalmente estática con apoyo del compilador.

Procesadores superescalares

1. **Lee** la sección **3.7** – Exploiting ILP using Multiple Issue and Static Scheduling (pág. 218–222).
   - **Computer Architecture: A Quantitative Approach**. 6th Ed. Hennessy and Patterson. Morgan Kaufmann. 2017.

2. **Aspectos clave**:
   - ¿Cuáles son las diferencias clave para cada categoría de procesador superescalar?
   - ¿Dónde se usan principalmente los procesadores VLIW?
   - ¿Qué diferencias hay entre VLIW y EPIC?

¿Por qué TLP?

- Algunas aplicaciones presentan más **paralelismo natural** que el que se puede conseguir con **ILP**.
  - Servidores, aplicaciones científicas, ...
- Emergen **dos modelos**:
  - **Paralelismo a nivel de hilos (TLP)**:
    - **Hilo**: Proceso con sus propias instrucciones y datos.
    - Puede ser parte un programa o programa independiente.
    - Cada hilo tiene asociado su **estado** (instrucciones, datos, PC, registros, ...).
  - **Paralelismo a nivel de datos (DLP)**:
    - Operaciones idénticas sobre distintos datos.

TLP

- **ILP** explota paralelismo implícito dentro un bloque básico o en bucles.
- **TLP** utiliza múltiples hilos de ejecución que son inherentemente paralelos.
- **Objetivo de TLP**:
  - Utilizar múltiples flujos de instrucciones para mejorar:
    - **Tasa de procesamiento** de computadores que ejecutan muchos programas.
    - **Tiempo de ejecución** de programas multi-hilo.

Ejecución multi-hilo

- Múltiples hilos comparten las unidades funcionales de un procesador solapando su uso.
  - Necesidad de replicar n-veces el estado.
    - Banco de registros, PC, tabla de páginas (si hilos no pertenecen al mismo programa).
    - Memoria compartida mediante mecanismos de memoria virtual.
    - Hardware para cambio de hilo rápido.
  - **Tipos**:
    - **Grano fino**: Cambio de hilo en cada instrucción.
    - **Grano grueso**: Cambio de hilo en detenciones (ej. Fallo de caché).
    - **Multi-hilo simultáneo**: Grano fino con emisión múltiple y planificación dinámica.

Multi-hilo de grano fino

- Se alterna entre hilos en cada instrucción.
  - Se entrelaza la ejecución de los hilos.
  - Normalmente se hace *round-robin*.
  - Se excluyen de *round-robin* hilos en detención (*stall*).
  - El procesador debe poder cambiar de hilo en cada ciclo de reloj.
- **Ventaja**:
  - Puede ocultar detenciones cortas y largas.
- **Desventaja**:
  - Retrasa la ejecución de hilos individuales por reparto.
- **Ejemplo**: Sun Niagara.

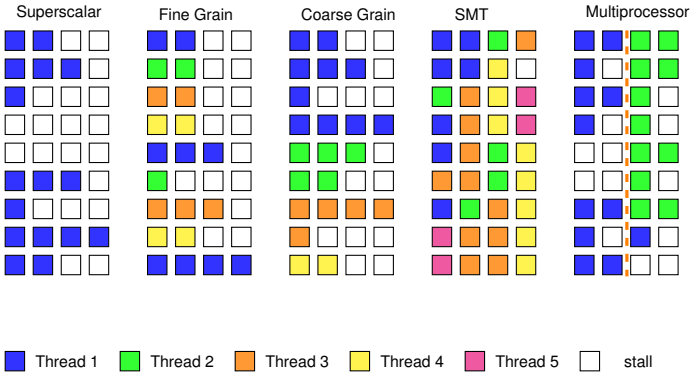
Multi-hilo de grano grueso

- Cambia de hilo solamente en detenciones largas.
  - **Ejemplo**: Fallo en caché L2.
- **Ventajas**:
  - No hace falta cambio de hilo excesivamente rápido.
  - No retrasa hilos individuales.
- **Desventajas**:
  - Se debe vaciar o congelar el pipeline.
  - Se debe llenar el pipeline con instrucciones del nuevo hilo (latencia).
- Apropiado cuando rellenado del pipeline tarda mucho menos que tiempo de detención.
  - **Ejemplo**: IBM AS/400.

SMT: Multi-hilo simultáneo

- **Idea**: Procesadores con planificación dinámica ya tienen muchos mecanismo de soporte para multi-hilo.
  - Grandes conjuntos de registros virtuales.
    - Registros para múltiples hilos.
  - Renombrado de registros.
    - Evita mezcla en acceso a registros de hilos.
  - Finalización fuera de orden.
- **Modificaciones**:
  - Tabla de renombrado por hilo.
  - Registros PC separados.
  - ROB separados.
- **Ejemplos**: Intel Core i7, IBM Power 7

TLP: Resumen


Superscalar
Fine Grain
Coarse Grain
SMT
Multiprocessor
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
stall

Resumen

- La planificación dinámica gestiona detenciones desconocidas en tiempo de compilación.
- Las técnicas especulativas se apoyan de la predicción de saltos y la planificación dinámica.
- La emisión múltiple en ILP queda limitada de forma práctica de 3 a 6.
- SMT como aproximación a TLP dentro un núcleo.

Referencias

- **Computer Architecture: A Quantitative Approach**. 6th Ed. Hennessy and Patterson. Morgan Kaufmann. 2017.
  - 3.3 – Reducing Branch Costs with advanced Branch Prediction.
  - 3.4 – Overcoming Data Hazards with Dynamic Scheduling.
  - 3.6 – Hardware-Based Speculation.
  - 3.7 – Exploiting ILP using Multiple Issue and Statich Scheduling.
  - 3.11 – Multithreading: Exploiting Thread-Level Parallelism to Improve Uniprocessor Throughput.

uc3m | Universidad Carlos III de Madrid

Paralelismo a nivel de instrucción

Arquitectura de Computadores

J. Daniel García Sánchez (coordinador)

Departamento de Informática
Universidad Carlos III de Madrid