Tema 4:
Segmentación *-pipelining- 3ª Parte*

Arquitectura de Computadores

Grado en Ingeniería Informática

Autores: Mercedes Rodríguez García, Miguel Ángel López Gordo ©

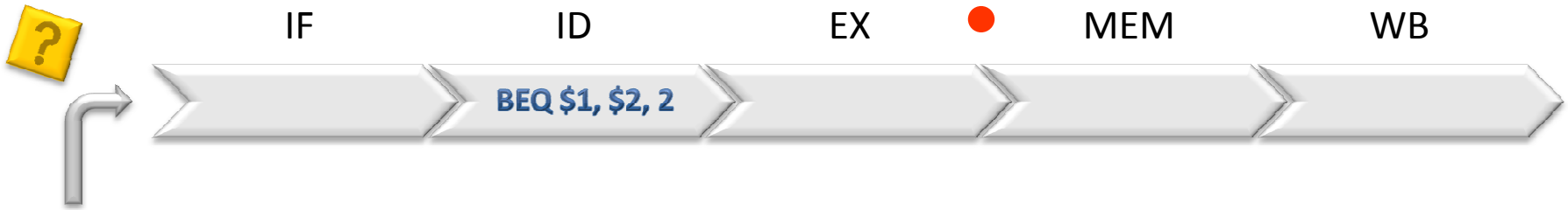
Índice

Bibliografía

Estructura y diseño de computadores: Capítulo 4.
Patterson y Hennessy. Editorial Reverte, 2011.

5.3. Riesgos de control

Se producen cuando entra un **salto condicional** en el pipeline. El procesador no sabe todavía si hay que saltar y ya tiene introducir la siguiente instrucción en el pipeline.



32: ADD $5, $5, 1
36: SUB $7, $5, $4
40: ADD $5, $5, 2

5.3.1. Métodos para resolver los riesgos de control

1 **Bloqueo (o burbuja)** GESTIONADO POR EL **PROCESADOR**

Se **detienen** las instrucciones en el pipeline hasta que se resuelva el riesgo, es decir, hasta que se evalúe el salto y se determine cuál es la siguiente instrucción que se debe ejecutar.

2 **Reordenación (o salto retardado)** GESTIONADO POR EL **COMPILADOR**

El compilador reordena el código para colocar inmediatamente después del salto condicional instrucciones independientes, es decir, instrucciones que siempre se deban ejecutar.

3 **Anticipación** GESTIONADO POR EL **PROCESADOR**

El cálculo del salto se adelanta a la segunda etapa del pipeline (etapa ID). Para implementarlo hay que situar en la etapa ID:

- El sumador que calcula la dirección del salto.
- Un comparador para evaluar la condición.

5.3.1. Métodos para resolver los riesgos de control

4 **Especulación (o predicción del salto)** GESTIONADO POR EL **PROCESADOR**

- **Predicción fija**.- Siempre se predice lo mismo: "el salto no será tomado", de forma que la próxima instrucción que entre en el pipeline será la siguiente en la secuencia de código. Si falla la predicción, se descarta la ejecución de todas las instrucciones que no debían haber entrado en el pipeline.

- **Predicción dinámica**.- Se añade un buffer al procesador para registrar **el comportamiento** de los saltos que se han ido ejecutando a lo largo del programa. Con esta información, el procesador predice el resultado del salto en ejecución. Este sistema tiene alrededor de un 90% de aciertos (es más preciso que la predicción fija).

6. ILP avanzado

1. Segmentación *-pipelining-*
2. Consideraciones en el diseño del repertorio de instrucciones
3. Camino de datos de un procesador segmentado
4. Representaciones gráficas de la segmentación
5. Riesgos del pipeline
   - 5.1. Riesgos estructurales
   - 5.2. Riesgos de datos
     - 5.2.1. Métodos para resolver los riesgos de datos
     - 5.2.2. Tipos de riesgos de datos
   - 5.3. Riesgos de control
     - 5.3.1. Métodos para resolver los riesgos de control
6. ILP avanzado
   - 6.1. Estrategias para incrementar ILP
   - 6.2. Ejecución múltiple con planificación estática
   - 6.3. Ejecución múltiple con planificación dinámica
7. Casos reales

6.1. Estrategias para incrementar el ILP

ILP = Instruction Level Parallelism

1 **Aumentar el número de etapas del pipeline**

Conseguimos solapar la ejecución de más instrucciones. Por ejemplo, un pipeline de 12 etapas permite solapar la ejecución de 12 instrucciones.

2 **Replicar componentes del procesador**

Conseguimos ejecutar varias instrucciones dentro de cada etapa de segmentación. A esta técnica se le llama ejecución múltiple -*multiple issue*-. Hay dos formas de implementar un procesador de ejecución múltiple:

- Planificación **estática**
- Planificación **dinámica**

6.2. Ejecución múltiple con planificación estática

“El compilador es el protagonista”

Se dice que la planificación es estática porque se realiza en tiempo de compilación. El compilador determina qué instrucciones se lanzarán a ejecutar simultáneamente en cada ciclo de reloj.

Todas las instrucciones que el compilador decide ejecutar simultáneamente se agrupan para formar una única **instrucción muy larga**.

Estos procesadores se conocen como procesadores **VLIW** *-Very Long Instruction Word-*.

6.3. Ejecución múltiple con planificación dinámica

“El hardware del procesador es el protagonista”

Se dice que la planificación es dinámica porque se realiza en tiempo de ejecución. El procesador, gracias a una unidad adicional (issue unit), determina qué instrucciones se lanzarán a ejecutar simultáneamente en cada ciclo de reloj.

Estos procesadores se conocen como procesadores **superescalares**.

Para alcanzar unas buenas prestaciones en estos procesadores, es recomendable que el **compilador** también intervenga reordenado las instrucciones.

7. Casos reales

7. Casos reales

1 AMD Opteron (2003)

Arquitectura:	x86-64 (CISC)
RISC:	Las instrucciones x86-64 internamente son transformadas en instrucciones RISC, que AMD llama **operaciones RISC**. Lo que se ejecuta realmente son las operaciones RISC
Pipeline:	**12** etapas
Superescalar:	Grado 3 con planificación dinámica

2 INTEL Pentium 4 Prescott (2004)

Arquitectura:	x86-64 (CISC)
RISC:	Las instrucciones x86-64 internamente son transformadas en instrucciones RISC, que Intel llama **microoperaciones**. Lo que se ejecuta realmente son las microoperaciones.
Pipeline:	**31** etapas
Superescalar:	Grado 3 con planificación dinámica

7. Casos reales

3 **INTEL Nehalem (2008) → Core i7**

Arquitectura:	x86-64 (CISC)
RISC:	Las instrucciones x86-64 internamente son transformadas en instrucciones RISC, que Intel llama **microoperaciones**. Lo que se ejecuta realmente son las microoperaciones.
Pipeline:	**14** etapas
Superescalar:	Grado 4 con planificación dinámica

Documentación adicional:

http://es.wikipedia.org/wiki/X86-64
http://www.chip-architect.com/news/2003_09_21_detailed_architecture_of_amds_64bit_core.html
http://classes.soe.ucsc.edu/cmpe202/Fall04/papers/opteron.pdf
http://es.wikipedia.org/wiki/Core_i7
http://www.realworldtech.com/nehalem/
http://sc.tamu.edu/systems/eos/nehalem.pdf