Pipeline - Dependencias y Riesgos
Lección 9

Prof.Ing. Fabián Zamora Ramírez

CE-4301 Arquitectura de Computadores I

Área de Ingeniería en Computadores

Instituto Tecnológico de Costa Rica

Agenda

Ejecución fuera de orden (OoOE)

Tipo de ILP en el que las instrucciones de **ejecutan** en un orden distinto al que fueron programadas.

OoOE

1. SUB R1, R2, R3
2. ADD R4, R3, **R1**
3. ROR R2, R2, #4

En lugar de 1,2,3 (riesgo de datos), se puede cambiar el orden a 1,3,2 y ganar desempeño.

Dependencias en pipeline

En una arquitectura que implementa pipeline (y/o otras formas de ILP) se pueden presentar 3 tipos de dependencias entre instrucciones:

1. Dependencias de datos (reales)
2. Dependencias de nombre
3. Dependencias de control

Las dependencias, en general, son producto de los programas.

Si una dependencia lleva a un riesgo, su detección y corrección son propiedades de la **organización** del pipeline.

Dependencias de datos (reales)

Una dependencia de datos entre instrucciones puede surgir en los siguientes casos:

- La instrucción *i* produce un resultado que puede ser utilizado por la instrucción *j*
- La instrucción *j* depende de un dato de la instrucción *k*, y la instrucción *k* depende de un dato de la instrucción *i*

Dependencia de datos

Ejemplo

1. ADD **R3**,R2, R1
2. SUB R1, **R3**, 1

¿Cuál es el problema?

Componentes de una dependencia de datos

Al tratar con dependencia de datos se debe tomar en cuenta:

- La **posibilidad** de un riesgo.
- El orden en que las instrucciones deben ejecutarse (caso OoOE).
- Límite máximo de paralelismo que puede ser explotado.

Soluciones a una dependencia de datos

Una dependencia no implica necesariamente un riesgo, pero deben ser atendidas.

- Mantener la dependencia, pero evitar el riesgo
- Eliminar la dependencia por la transformación del código**

Pueden ser implementadas por software o por hardware.

Dependencia de nombre

Ocurre cuando dos instrucciones usan el mismo registro (o dirección de memoria), pero **NO** hay relación o flujo entre las instrucciones.

Dos tipos:

- Antidependencia
- Dependencia de salida

Antidependencia

Ocurre cuando una instrucción *j* escribe a un registro o posición de memoria que una instrucción *i* lee.

Ejemplo

1. ADD R3, **R2**, R1
2. SUB **R2**, R5, 1

¿Cuál es el problema?

Dependencia de salida

Ocurre cuando una instrucción *i* y una instrucción *j* escriben al mismo registro o dirección de memoria.

Ejemplo

1. ADD R3, R1, R2
2. SUB R4, R3, 1
3. ADD R4, R5, R5

¿Cuál es el problema?

Solución dependencias de nombre

Dado que no hay transmisión entre las instrucciones, **no son dependencias** reales.

- Pueden ser ejecutadas en paralelo

Solución: **Renombramiento de registros**

- Por hardware: Calendarización dinámica de instrucciones. RRU: register renaming unit.
- Por software:Calendarización estática. Compilación.

Dependencias de control

Una dependencia de control determina el orden de ejecución de una instrucción *i*, con respecto a una instrucción de salto previa.

```
if p1{
        S1;
}
if p2{
        S2;
}
```

No debe invertirse el orden de ejecución cuando existen dependencia de control.

Dependencias de control : Implicaciones

1. Una instrucción dependiente de control en un salto NO puede movida antes del salto.
2. Una instrucción que NO es dependiente de control NO pude ser movida justo después de un salto.

Ejemplo I

```
_init:
    ADD R1, R1, R2
    BEQ R1, T0, ext
    SUB R1, R2, R3

_ext:
    done
```

Riesgos de datos

Un riesgo de datos se puede tener cuando se presenta una dependencia de **nombre** o real de **datos** entre instrucciones lo suficientemente cercanas para que se pueda producir un cambio en el orden de acceso a los operandos.

Tres tipos de riesgos de datos:

- RAW
- WAR
- WAW

Lectura después de escritura (Read After Write - RAW)

Se presenta cuando una instrucción *j* trata de leer un operando antes de que la instrucción *i* lo escriba, obteniendo un valor antiguo.

Ejemplo

1. ADD **R3**,R2, R1
2. SUB R1, **R3**, 1

Escritura después de lectura (WAR)

Se presenta cuando la instrucción *j* trata de escribir un destino **antes** de que se leído por la instrucción *i*, lo que provoca que *i* lea el nuevo valor (incorrecto).

Ejemplo

i ADD R4,R2, **R1**
j SUB **R1**, R3, 1

Escritura después de escritura (WAW)

Se presenta cuando la instrucción *j* trata de escribir un operando **antes** de que se escrito por la instrucción *i*. Las escrituras se realizan en el orden incorrecto.

Ejemplo

i ADD **R1**,R2, R3
j SUB **R1**, R3, 1

Referencias

J Hennesy and David Patterson (2005)

Computer Organization and Design. The hardware/software interface. 4th Edition. [Cap 4]

J Hennesy and David Patterson (2012)

Computer Architecture: A Quantitative Approach. 5th Edition. Elsevier – Morgan Kaufmann. [Cap 3]

Jeferson González G. (2017)

Material de clase: Arquitectura de computadores I.