

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:

- ➊ Dependencias de datos (reales).
- ➋ Dependencias de nombre.
- ➌ 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 .

```
L.D    F0,0(R1)    ;F0=array element
ADD.D  F4,F0,F2     ;add scalar in F2
S.D    F4,0(R1)     ;store result
DADDUI R1,R1,#-8    ;decrement pointer 8 bytes
BNE    R1,R2,LOOP   ;branch R1!=R2
```

Ejemplo

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

```
Loop:   L.D    F0,0(R1)    ;F0=array element
        ADD.D  F4,F0,F2     ;add scalar in F2
        S.D    F4,0(R1)     ;store result
```



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

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

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

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.

- ① Una instrucción dependiente de control en un salto NO puede moverse antes del salto.
- ② Una instrucción que NO es dependiente de control NO puede moverse justo después de un salto.

Implicaciones

Ejemplo I

_init:

```
ADD R1, R1, R2  
BEQZ R1, T0, L1  
SUB R1, R2, R3
```

L1:

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:

ReadAfter 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 sea 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

Técnicas de software para mejorar ILP

Se consideran estáticas.

Se realizan durante tiempo de compilación.

- 'Información' para branch prediction.
- Reordenamiento de código (memoria).

Durante compilación se puede reorganizar el código de forma que se optimice el uso de procesador.

- Renombramiento de registros.

Durante compilación se puede detectar falsas dependencias de datos y renombrar registros para aumentar el ILP.

- *Loopunrolling*.

La idea principal es reducir la cantidad de iteraciones y lógica de control (instrucciones condicionales) en un bloque de código para mejorar el ILP.

Puede darse como optimización del compilador o en el código fuente directamente.