

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

Arquitectura de Computadores I

Luis Alberto Chavarría-Zamora

ITCR

lachavarria@tec.ac.cr

29 de agosto de 2023

Dependencias

Datos
Nombre
Control

Riesgos de
datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de
software para
mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad
Breakout
Rooms

Referencias

Contenido

1 Dependencias

Datos
Nombre
Control

2 Riesgos de datos

RAW
WAR
WAW

3 Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de código
Renombramiento de Registros
Loop Unrolling

4 Actividad Breakout Rooms

5 Referencias

Ejecución fuera de orden (OoOE)

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

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

Ejecución fuera de orden (OoOE)

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

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

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

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)

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

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 .

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

Dependencia de datos

Ejemplo

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

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

¿Cuál es el problema?

Dependencia de datos (reales)

```
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
```

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de código

Renombramiento de Registros

Loop Unrolling


Actividad Breakout Rooms

Referencias

Dependencia de datos (reales)

```
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
```

```
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
```



Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de código

Renombramiento de Registros

Loop Unrolling


Actividad Breakout Rooms

Referencias


Dependencia de datos (reales)

```
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
```

```
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
```



```
DADDIU  R1,R1,#-8   ;decrement pointer
        ;8 bytes (per DW)
BNE     R1,R2,Loop  ;branch R1!=R2
```



Componentes de una dependencia de datos

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

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

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

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 datos (reales)

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de código

Renombramiento de Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

Considere el siguiente código, ejecutado en un pipeline de 5 etapas

SUB	X2,X1,X3	// Registro X2 escrito por SUB
AND	X12,X2,X5	// Primer operando depende SUB
OR	X13,X6,X2	// Segundo operando depende SUB
ADD	X14,X2,X2	// Operando 1 y 2 depende SUB
STUR	X15,[X2,#100]	// Base (X2) depende de SUB

¿Existen dependencias reales entre las instrucciones SUB y STUR?

Dependencias de datos (reales)

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

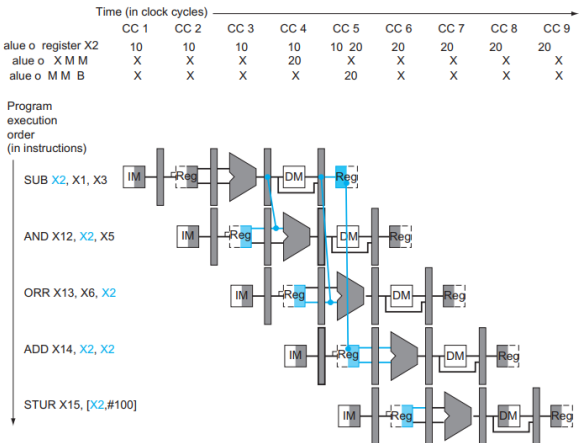
Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias



Dependencia de nombre

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de código

Renombramiento de Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

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

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

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

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

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

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

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

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

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

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

- 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.

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

Ejemplo I

_init :

ADD R1, R1, R2

BEQZ R1, T0, L1

SUB R1, R2, R3

L1 :

done

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de código
Renombramiento de Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

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)

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

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)

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

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)

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

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

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

Se consideran estáticas.

Técnicas de software para mejorar ILP

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

Se consideran estáticas.

Se realizan durante tiempo de compilación.

- 'Información' para branch prediction.
- Reordenamiento de código (memoria).
- Renombramiento de registros.
- *Loop unrolling*.

Dependencias

Datos
Nombre
Control

Riesgos de
datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de
software para
mejorar ILP

Branch Prediction

Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad
Breakout
Rooms

Referencias

Branch Prediction

```
1  #define likely(x)      __builtin_expect((x),1)
2  #define unlikely(x)    __builtin_expect((x),0)
3
4
5  if (unlikely(fd < 0))
6  {
7      /* Do something */
8  }
9  ...
10
11 if (likely(!err))
12 {
13     /* Do something */
14 }
```

Built-in Function: `long __builtin_expect(long exp, long c)`

You may use `__builtin_expect` to provide the compiler with branch prediction information. In general, you should prefer to use actual profile feedback for this (`-fprofile-arcs`), as programmers are notoriously bad at predicting how their programs actually perform. However, there are applications in which this data is hard to collect.

The return value is the value of `exp`, which should be an integral expression. The semantics of the built-in are that it is expected that `exp == c`. For example:

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

Reordenamiento de código

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

```
1 int A, B;  
2  
3 void foo()  
4 {  
5     A = B + 1;  
6     B = 0;  
7 }
```

```
> gcc -S -masm=intel foo.c  
> cat foo.s  
...  
mov     eax, DWORD PTR _B  
add     eax, 1  
mov     DWORD PTR _A, eax  
| mov     DWORD PTR _B, 0  
...
```

```
> gcc -O2 -S -masm=intel foo.c  
> cat foo.s  
...  
mov     eax, DWORD PTR B  
| mov     DWORD PTR B, 0  
add     eax, 1  
mov     DWORD PTR A, eax  
...
```

Reordenamiento de código

La reorganización de código puede ser evitado mediante instrucciones al compilador

```
1 int A, B;  
2  
3 void foo()  
4 {  
5     A = B + 1;  
6     asm volatile("" :: "memory");  
7     B = 0;  
8 }
```

```
> gcc -O2 -S -masm=intel foo.c  
> cat foo.s
```

```
...  
mov     eax, DWORD PTR _B  
add     eax, 1  
mov     DWORD PTR _A, eax  
mov     DWORD PTR _B, 0  
...
```

Qualifiers

volatile

The typical use of extended `asm` statements is to manipulate input values to produce output values. However, your `asm` statements may also produce side effects. If so, you may need to use the `volatile` qualifier to disable certain optimizations. See [Volatile](#).

Reordenamiento de código

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de código

Renombramiento de Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

Se pueden aplicar optimizaciones.

Normal

```
> gcc -S -masm=intel foo.c
> cat foo.s
...
mov     eax, DWORD PTR _B
add     eax, 1
mov     DWORD PTR _A, eax
mov     DWORD PTR _B, 0
...
```

Optimizado

```
> gcc -O2 -S -masm=intel foo.c
> cat foo.s
...
mov     eax, DWORD PTR B
mov     DWORD PTR B, 0
add     eax, 1
mov     DWORD PTR A, eax
...
```

Optimizado con *fence*

```
> gcc -O2 -S -masm=intel foo.c
> cat foo.s
...
mov     eax, DWORD PTR _B
add     eax, 1
mov     DWORD PTR _A, eax
mov     DWORD PTR _B, 0
...
```

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

Reordenamiento de código

Se pueden aplicar optimizaciones.

Normal

```
> gcc -S -masm=intel foo.c
> cat foo.s
...
mov     eax, DWORD PTR _B
add     eax, 1
mov     DWORD PTR _A, eax
mov     DWORD PTR _B, 0
...
```

Optimizado

```
> gcc -O2 -S -masm=intel foo.c
> cat foo.s
...
mov     eax, DWORD PTR B
mov     DWORD PTR B, 0
add     eax, 1
mov     DWORD PTR A, eax
...
```

Optimizado con *fence*

```
> gcc -O2 -S -masm=intel foo.c
> cat foo.s
...
mov     eax, DWORD PTR _B
add     eax, 1
mov     DWORD PTR _A, eax
mov     DWORD PTR _B, 0
...
```

Tarea moral investigue las optimizaciones del GCC y los FENCES en ARM-Cortex.

Renombramiento de Registros

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

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
**Renombramiento de
Registros**
Loop Unrolling

Actividad Breakout Rooms

Referencias

```
# Instruction
1  R1 = M[1024]
2  R1 = R1 + 2
3  M[1032] = R1
4  R1 = M[2048]
5  R1 = R1 + 4
6  M[2056] = R1
```

```
s/R1/R2/g in 4,5,6
4  R2 = M[2048]
5  R2 = R2 + 4
6  M[2056] = R2
```

Now 1-6 can be executed in parallel

```
1  R1 = M[1024]      R2 = M[2048]
2  R1 = R1 + 2       R2 = R2 + 4
3  M[1032] = R1      M[2056] = R2
```

Dependencias

Datos

Nombre

Control

Riesgos de datos

Riesgos de datos

RAW

WAR

WAW

Técnicas de software para mejorar ILP

Branch Prediction

Reordenamiento de
código

Renombramiento de
Registros

Loop Unrolling

Actividad Breakout Rooms

Referencias

Loop Unrolling

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.

Loop Unrolling

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.

Normal loop	After loop unrolling
<pre>int x; for (x = 0; x < 100; x++) { delete(x); }</pre>	<pre>int x; for (x = 0; x < 100; x += 5) { delete(x); delete(x + 1); delete(x + 2); delete(x + 3); delete(x + 4); }</pre>

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

Actividad Breakout Rooms

De las técnicas presentadas a cual se refiere la siguiente frase:
'eliminar el riesgo pero mantener la dependencia'

Dependencias

Datos
Nombre
Control

Riesgos de datos

Riesgos de datos
RAW
WAR
WAW

Técnicas de software para mejorar ILP

Branch Prediction
Reordenamiento de
código
Renombramiento de
Registros
Loop Unrolling

Actividad Breakout Rooms

Referencias

Referencias



J. Hennesy y D. Patterson (2012)

Computer Architecture: A Quantitative Approach. 5th Edition.
Elsevier – Morgan Kaufmann.



J. González y R. García (2019)

Notas de clase de los profesores: Jeferson González y Ronald García.
ARMv8-A Architecture Reference Manual
Intel® 64 and IA-32 architectures software developer's manual
combined volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4