INSTRUCTION LEVEL PARALLELISM

Mahdi Nazm Bojnordi
Assistant Professor
School of Computing
University of Utah
Overview

- Announcement
  - Tonight: release HW2 (due 11:59PM, Sept. 18)
    - Note: late submission = no submission
    - One of your lowest assignment scores will be dropped 😊

- This lecture
  - Recap multicycle
  - Impacts of data dependence
  - Pipeline performance
  - Instruction level parallelism
Multicycle Instructions

- Data hazards
  - more read-after-write hazards

```plaintext
load f4, 0(r2)
mul f0, f4, f6
add f2, f0, f8
store f2, 0(r2)
```
Multicycle Instructions

- Data hazards
  - more read-after-write hazards

```
load f4, 0(r2)
mul f0, f4, f6
add f2, f0, f8
store f2, 0(r2)
```
Multicycle Instructions

- Data hazards
  - more read-after-write hazards

```
lod f4, 0(r2)
mul f0, f4, f6
add f2, f0, f8
store f2, 0(r2)
```

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MA</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>ID</td>
<td>M1</td>
<td>M2</td>
<td>M3</td>
</tr>
<tr>
<td>IF</td>
<td>ID</td>
<td>A1</td>
<td>A2</td>
<td>A3</td>
</tr>
<tr>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MA</td>
<td>WB</td>
</tr>
</tbody>
</table>

Multicycle Instructions

- Data hazards
  - potential write-after-write hazards

load f4, 0(r2)
mul f2, f4, f6
add f2, f0, f8
store f2, 0(r2)
Multicycle Instructions

- Data hazards
  - potential write-after-write hazards

```
load f4, 0(r2)
mul f2, f4, f6
add f2, f0, f8
store f2, 0(r2)
```
Multicycle Instructions

- Data hazards
  - potential write-after-write hazards

```
load f4, 0(r2)
mul f2, f4, f6
add f2, f0, f8
store f2, 0(r2)
```
Multicycle Instructions

- Data hazards
  - potential write-after-write hazards

```plaintext
load f4, 0(r2)
mul f2, f4, f6
add f2, f0, f8
store f2, 0(r2)
```

Out of Order Write-back!!
Multicycle Instructions

- Data hazards
  - potential write-after-write hazards

```plaintext
load f4, 0(r2)  
mul f2, f4, f6  
add f2, f0, f8  
store f2, 0(r2)
```
Multicycle Instructions

- Imprecise exception
  - Instructions do not necessarily complete in program order

```
load f4, 0(r2)
mul f2, f4, f6
add f3, f0, f8
store f2, 0(r2)
```
Multicycle Instructions

- Imprecise exception
  - Instructions do not necessarily complete in program order

```
load f4, 0(r2)
mul f2, f4, f6
add f3, f0, f8
store f2, 0(r2)
```

![Data flow diagram](Image)
Multicycle Instructions

- Imprecise exception
  - instructions do not necessarily complete in program order

```
load f4, 0(r2)
mul f2, f4, f6
add f3, f0, f8
store f2, 0(r2)
```

```
IF  ID  EX  MA  WB
IF  ID  M1  M2  M3  M4  M5  M6  M7  MA  WB
IF  ID  A1  A2  A3  A4  MA  WB
IF  ID  EX  MA  WB
```

Overflow!!
Multicycle Instructions

- Imprecise exception
  - state of the processor must be kept updated with respect to the program order

```
load f4, 0(r2)
mul f2, f4, f6
add f3, f0, f8
store f2, 0(r2)
```

In-order register file updates
Reorder Buffer

- Multicycle Instructions

mul f2, f4, f6
add f4, f0, f1
sub f6, f3, f7

Diagram showing the flow of instructions through different stages: IF, ID, M1, M2, M3, M4, M5, M6, M7, Integer unit, EX, FP/integrate multiply, FP adder, FP/integrate divider, MEM, WB.
Reorder Buffer

- Multicycle Instructions

mul f2, f4, f6
add f4, f0, f1
sub f6, f3, f7
Data Dependence

Point of production

- The pipeline stage where an instruction produces a value that can be used by its following instructions.
Data Dependence

- **Point of production**
  - The pipeline stage where an instruction produces a value that can be used by its following instructions

- **Point of consumption**
  - The pipeline stage where an instruction consumes a produced data

Inst. 1: producer
Inst. 2: consumer

PoC  PoP

\[ \text{time} \]
Consider a 10-stage pipeline processor, where point of production and point of consumption are separated by 4 cycles. Assume that half the instructions do not introduce a data hazard and half the instructions depend on their preceding instruction. What is the maximum attainable IPC?
Consider a 10-stage pipeline processor, where point of production and point of consumption are separated by 4 cycles. Assume that half the instructions do not introduce a data hazard and half the instructions depend on their preceding instruction. What is the maximum attainable IPC?
Consider a 10-stage pipeline processor, where point of production and point of consumption are separated by 4 cycles. Assume that half the instructions do not introduce a data hazard and half the instructions depend on their preceding instruction. What is the maximum attainable IPC?

<table>
<thead>
<tr>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stall Cycles</td>
</tr>
</tbody>
</table>

\[
\text{IPC} = \frac{2}{5} = 0.4
\]
Performance vs. Pipeline Depth

- Impact of stall cycles on performance
  - Independent instructions
  - Dependent instructions

![Graph showing performance vs. pipeline depth](image)
Impact of stall cycles on performance

- Independent instructions
- Dependent instructions
Performance vs. Pipeline Depth

- Impact of stall cycles on performance
  - Independent instructions
  - Dependent instructions

$$\frac{1}{\text{latch latency}}$$

Diagram showing performance vs. pipeline depth with lines for no stalls and fully stalled conditions.
Performance vs. Pipeline Depth

- Impact of stall cycles on performance
  - Independent instructions
  - Dependent instructions

![Graph showing performance vs. pipeline depth with curves for no stalls, fully stalled, and average cases.](image)
Performance vs. Pipeline Depth

- Impact of stall cycles on performance
  - Independent instructions
  - Dependent instructions

Increase overlap among instructions in the pipeline (Instruction Level Parallelism)
Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow

Code 1
- ADD R1, R2, R3
- SUB R4, R1, R5
- XOR R6, R4, R7
- AND R8, R6, R9

Code 2
- ADD R1, R2, R3
- SUB R4, R6, R5
- XOR R8, R2, R7
- AND R9, R6, R0
Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow

**Code 1**

- `ADD R1, R2, R3`
- `SUB R4, R1, R5`
- `XOR R6, R4, R7`
- `AND R8, R6, R9`

- ILP = 1
  - Fully serial

**Code 2**

- `ADD R1, R2, R3`
- `SUB R4, R6, R5`
- `XOR R8, R2, R7`
- `AND R9, R6, R0`

- ILP = 4
  - Fully parallel
Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler

$$X \leftarrow A + B + C + D$$
Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler

Code 1:

```
ADD  R5, R1, R2
ADD  R5, R5, R3
ADD  R5, R5, R4
```

\[ X \leftarrow A + B + C + D \]
Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler

X ← A + B + C + D

**Code 1:**
ADD R5, R1, R2  
ADD R5, R5, R3  
ADD R5, R5, R4

**Code 2:**
ADD R6, R1, R2  
ADD R7, R3, R4  
ADD R5, R6, R7
Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler

```
X ← A + B + C + D
```

**Code 1:**
```plaintext
ADD R5, R1, R2
ADD R5, R5, R3
ADD R5, R5, R4
```

**Code 2:**
```plaintext
ADD R6, R1, R2
ADD R7, R3, R4
ADD R5, R6, R7
```

- Average ILP = 3/3 = 1
  - Five registers
- Average ILP = 3/2 = 1.5
  - Seven registers
Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler

- An upper limit for attainable IPC for a given code
  - IPC represents exploited ILP

```
ADD R5, R1, R2
ADD R5, R5, R3
ADD R5, R5, R4
ADD R6, R1, R2
ADD R7, R3, R4
ADD R5, R6, R7
```

Average ILP = $\frac{3}{3} = 1$
Five registers

Average ILP = $\frac{3}{2} = 1.5$
Seven registers
Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler

- An upper limit for attainable IPC for a given code
  - IPC represents exploited ILP

- Can be exploited by HW-/SW-intensive techniques
  - Dynamic scheduling in hardware
  - Static scheduling in software (compiler)