INSTRUCTION LEVEL PARALLELISM

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

- Announcement
  - HW1 solutions will be posted in Canvas
    - Recall that late submission = no submission
    - One of your lowest assignment scores will be dropped

- This lecture
  - Impacts of data dependence
  - Pipeline performance
  - Instruction level parallelism
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.

Ints. 1: producer

Inst. 2: consumer
Problem

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?

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

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

![Graph showing performance versus pipeline depth with stall cycles and no stalls](image-url)
Performance vs. Pipeline Depth

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

\[
\frac{1}{\text{latch latency}}
\]

![Graph showing performance vs. pipeline depth with lines for no stalls and stalls.](image-url)
Performance vs. Pipeline Depth

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

\[ \frac{1}{\text{latch latency}} \]

Graph showing performance vs. pipeline depth (number of stages) for cases with no stalls and fully stalled.

Legend:
- No Stalls
- Fully Stalled
Performance vs. Pipeline Depth

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

\[ \frac{1}{latch \ latency} \]

Graph showing performance vs. pipeline depth with lines for no stalls, fully stalled, and average cases.
Performance vs. Pipeline Depth

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

\[
\frac{1}{\text{latch latency}}
\]

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 ← A + B + C + D
```
Instruction Level Parallelism

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

\[ X \leftarrow A + B + C + D \]

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

\[ X \leftarrow A + B + C + D \]

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

Average ILP = \(3/3 = 1\)
Five registers

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

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 = \(3/3 = 1\)
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

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