CMPT 295
Unit – Processor Design & Instruction Execution
Lecture 29 – Pipelined Execution – Complete Lecture
Sequential execution of machine instructions

**Processor design #1: Sequential execution**
- Executing 1 machine instruction per clock cycle
- **Issue:** Long clock cycle -> slow computer 🙁 (small throughput)

**Analysis of processor instruction execution:**
- **Latency** (propagation delay): Time required to execute a single instruction
- **Throughput:** Number of instructions executed per second - GIPS
How to improve throughput

- Divide the execution of instructions into stages: **fetch, decode, execute, memory, write back**
- Introduce clocked registers (pipeline registers) after each stage
- **Processor design #2: Staged execution**

Result: Shorter (faster) clock cycle -> faster computer 😊

Issues:
- Added clocked registers increase latency
- Stages may not have equal propagation delay
Today’s Menu

- Instruction Set Architecture (ISA)
  - Definition of ISA
  - ISA design
  - ISA evaluation
    - Improving our ISA -> Decreasing effect of von Neumann bottleneck
    - 3 Strategies

- Execution of machine instructions
  - Intro to logic design, combinational and sequential logic circuits
  - Sequential execution of machine instructions
  - Pipelined execution of machine instructions
Pipelining

- Example: fast food counter versus cafeteria

- Start executing a new instruction at every clock cycle
- Effect: Different stages of different instructions overlap
A closer look:

Instruction 1:

ADD r0, r5:

- F: IR ← M[PC]
  PC ← PC+1

- D: valA ← r[0]
  valB ← r[5]

- E: valE ← valA + valB


- W: val ->
Clock cycle:

Pipelined Execution

Time
Analysis

- **Sequential Execution – 1 stage:**
  - Latency - cost per instruction:
    - $400 \text{ ps} + 20 \text{ ps} = 420 \text{ ps}$
  - CPU throughput: \( \frac{1}{420 \text{ ps}} = 2.38 \text{ GIPS} \)
  - Clock cycle: $400 \text{ ps} + 20 \text{ ps} = 420 \text{ ps}$

- **Staged Execution – 5 stages:**
  - Latency - cost per instruction:
    - \(\frac{400}{5} \text{ ps} + 20 \text{ ps})^5 = 500 \text{ ps}\)
  - CPU throughput: \( \frac{1}{500 \text{ ps}} = 2 \text{ GIPS} \)
  - Clock cycle: $80 \text{ ps} + 20 \text{ ps} = 100 \text{ ps}$

- **Pipelined Execution – 5 stages:**
  - Latency - cost per instruction: \(\frac{400}{5} \text{ ps} + 20 \text{ ps})^5 = 500 \text{ ps}\) **has not changed**
  - CPU throughput: \( \frac{5}{500 \text{ ps}} = 10 \text{ GIPS} \) (or \( \frac{1}{100 \text{ ps}} = 10 \text{ GIPS} \))
  - Clock cycle: $80 \text{ ps} + 20 \text{ ps} = 100 \text{ ps}$ **has not changed**

**Conclusion:**
- Instruction latency: ▲
- CPU throughput: ▲
- Clock cycle: ▼
Analysis

Sequential Execution – 1 stage:
- Latency - cost per instruction: 400 ps + 20 ps = 420 ps
- CPU throughput: 1 / 420 ps = 2.38 GIPS
- Clock cycle: 400 ps + 20 ps = 420 ps

Staged Execution – 5 stages:
- Latency - cost per instruction: \((400/5 \text{ ps} + 20 \text{ ps})5 = 500 \text{ ps}\)
- CPU throughput: 1 / 500 ps = 2 GIPS
- Clock cycle: 80 ps + 20 ps = 100 ps

Pipelined Execution – 5 stages:
- Latency - cost per instruction: \((400/5 \text{ ps} + 20 \text{ ps})5 = 500 \text{ ps}\) (has not changed)
- CPU throughput: 5 / 500 ps = 10 GIPS
- Clock cycle (cc): 80 ps + 20 ps = 100 ps (has not changed)

Conclusion:
- Instruction latency: \(\uparrow\)
- CPU throughput: \(\uparrow\)
- Clock cycle: \(\downarrow\)
## A closer look at Analysis of Pipelined Execution

**Pipelined Execution – 5 stages:**
- 5 instructions per 5 clock cycles -> 1 instruction per clock cycle
- One way of seeing the situation would be that we have the equivalent of 5 instructions being executed at every 5 clock cycles:

In this snapshot of the processor executing instructions:
- Instruction 5 (F, D, E, M, W) -> 5 stages executed
- Instruction 4 (D, E, M, W) + Instruction 9 (F) -> 5 stages executed
- Instruction 3 (E, M, W) + Instruction 8 (F, D) -> 5 stages executed
- Instruction 2 (M, W) + Instruction 7 (F, D, E) -> 5 stages executed
- Instruction 1 (W) + Instruction 6 (F, D, E, M) -> 5 stages executed

<table>
<thead>
<tr>
<th>cc 1</th>
<th>cc 2</th>
<th>cc 3</th>
<th>cc 4</th>
<th>cc 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>F5</td>
<td>D4</td>
<td>E3</td>
<td>M2</td>
<td>W1</td>
</tr>
<tr>
<td>F6</td>
<td>D5</td>
<td>E4</td>
<td>M3</td>
<td>W2</td>
</tr>
<tr>
<td>F7</td>
<td>D6</td>
<td>E5</td>
<td>M4</td>
<td>W3</td>
</tr>
<tr>
<td>F8</td>
<td>D7</td>
<td>E6</td>
<td>M5</td>
<td>W4</td>
</tr>
<tr>
<td>F9</td>
<td>D8</td>
<td>E7</td>
<td>M6</td>
<td>W5</td>
</tr>
</tbody>
</table>
Limitations of pipelining - 1

1. Would doubling the number of stages (5 -> 10) double the throughput (10 GIPS -> 20 GIPS)?

- Cost of each stage
  -> 400/10 ps + 20 ps = 40 ps + 20 ps = 60 ps
- CPU throughput -> 1 / 60 ps = 16.67 GIPS
- So?
  - Nop! (not quite 20 GIPS)
Limitations of pipelining - 2

2. What if stages have different delays?
   - Sum of delays through all the stages remains the same
   - But, clock cycle limited by delay of slowest stage
Summary

- Pipelining
- Analysis of pipelined execution using best case pipeline scenario (all stages have the same propagation delay)
  - Impact on CPU throughput
- Limitations on pipelined execution
  - Limitation 1
    - Doubling the number of stages does not double the throughput
  - Limitation 2
    - Non-uniform stage delays – not every stage has same delay
    - Longest delay wins all
Instruction Set Architecture (ISA)
- Definition of ISA
- ISA design
- ISA evaluation
  - Improving our ISA -> Decreasing effect of von Neumann bottleneck
  - 3 Strategies

Execution of machine instructions
- Intro to logic design, combinational and sequential logic circuits
- Sequential execution of machine instructions
- Pipelined execution of machine instructions
  - Hazards