CIS 429/529

Pipeline and Hazards

Prof. Michel A. Kinsky
An Ideal Pipeline

- All objects go through the same stages
- No sharing of resources between any two stages
- Propagation delay through all pipeline stages is equal
An Ideal Pipeline

- The scheduling of an object entering the pipeline is not affected by the objects in other stages.

- These conditions generally hold for industrial assembly lines.

  ▶ But what about an instruction pipeline?
Clock period can be reduced by dividing the execution of an instruction into multiple cycles.
• However, CPI will increase unless instructions are pipelined
How to divide the datapath

• Suppose memory is significantly slower than other stages. In particular, suppose

\[
\begin{align*}
t_{IM} &= 10 \text{ units} \\
t_{DM} &= 10 \text{ units} \\
t_{ALU} &= 5 \text{ units} \\
t_{RF} &= 1 \text{ unit} \\
t_{RW} &= 1 \text{ unit}
\end{align*}
\]

• Since the slowest stage determines the clock, it may be possible to combine some stages without any loss of performance.
Alternative Pipelining

\[ t_C > \max \{ t_{IM}, t_{RF} + t_{ALU}, t_{DM} + t_{RW} \} = t_{DM} + t_{RW} \]

- Write-back stage takes much less time than other stages
5-Stage Pipelined

\[
\begin{align*}
\text{time} & \quad t0 & \quad t1 & \quad t2 & \quad t3 & \quad t4 & \quad t5 & \quad t6 & \quad t7 & \quad \ldots \\
\text{instruction1} & \quad \text{IF}_1 & \quad \text{ID}_1 & \quad \text{EX}_1 & \quad \text{MA}_1 & \quad \text{WB}_1 & \quad \text{IF}_2 & \quad \text{ID}_2 & \quad \text{EX}_2 & \quad \text{MA}_2 & \quad \text{WB}_2 \\
\text{instruction2} & \quad \text{IF}_2 & \quad \text{ID}_2 & \quad \text{EX}_2 & \quad \text{MA}_2 & \quad \text{WB}_2 & \quad \text{IF}_3 & \quad \text{ID}_3 & \quad \text{EX}_3 & \quad \text{MA}_3 & \quad \text{WB}_3 \\
\text{instruction3} & \quad \text{IF}_3 & \quad \text{ID}_3 & \quad \text{EX}_3 & \quad \text{MA}_3 & \quad \text{WB}_3 & \quad \text{IF}_4 & \quad \text{ID}_4 & \quad \text{EX}_4 & \quad \text{MA}_4 & \quad \text{WB}_4 \\
\text{instruction4} & \quad \text{IF}_4 & \quad \text{ID}_4 & \quad \text{EX}_4 & \quad \text{MA}_4 & \quad \text{WB}_4 & \quad \text{IF}_5 & \quad \text{ID}_5 & \quad \text{EX}_5 & \quad \text{MA}_5 & \quad \text{WB}_5 \\
\text{instruction5} & \quad \text{IF}_5 & \quad \text{ID}_5 & \quad \text{EX}_5 & \quad \text{MA}_5 & \quad \text{WB}_5 
\end{align*}
\]
5-Stage Pipelined Execution

I-Fetch (IF)  Decode, Reg. Fetch (ID)  Execute (EX)  Memory (MA)  Write-Back (WB)

Resources

<table>
<thead>
<tr>
<th>time</th>
<th>t0</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
</tr>
<tr>
<td>ID</td>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
</tr>
<tr>
<td>EX</td>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
</tr>
<tr>
<td>MA</td>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
</tr>
<tr>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pipelined Execution

- Not quite correct!
Pipelined MIPS Datapath

- What else is needed?
Instruction Interactions

- An instruction in the pipeline may need a resource being used by another instruction in the pipeline
  - structural hazard

- An instruction may depend on something produced by an earlier instruction
  - Dependence may be for a data calculation
    - data hazard
  - Dependence may be for calculating the next address
    - control hazard (branches, interrupts)
Data Hazards

\[ r_4 \leftarrow r_0 + 10 \]
\[ r_4 \leftarrow r_1 + 17 \]
Resolving Data Hazards

- **Strategy 1**: Wait for the result to be available by freezing earlier pipeline stages
  - interlocks

- **Strategy 2**: Route data as soon as possible after it is calculated to the earlier pipeline stage
  - bypass
Resolving Data Hazards

• **Strategy 3: Speculate on the dependence**
  - Two cases:
    - Guessed correctly
      - do nothing
    - Guessed incorrectly
      - kill and restart
Feedback to Resolve Hazards

- Later stages provide dependence information to earlier stages which can stall (or kill) instructions.
- Controlling a pipeline in this manner works provided the instruction at stage $i+1$ can complete without any interference from instructions in stages 1 to $i$.
  - Otherwise deadlocks may occur.
Interlocks to resolve Data Hazards

Stall Condition

... r1 ← r0 + 10
... r4 ← r1 + 17
Stalled Stages and Pipeline

\[
\begin{align*}
(I_1) & \quad r1 \leftarrow (r0) + 10 \\
(I_2) & \quad r4 \leftarrow (r1) + 17 \\
(I_3) & \\
(I_4) & \\
(I_5) & 
\end{align*}
\]

Resource Usage

<table>
<thead>
<tr>
<th>time</th>
<th>t0</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7 ..</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td>I3</td>
<td>I3</td>
<td>I3</td>
<td>I4</td>
<td>I5 ..</td>
</tr>
<tr>
<td>ID</td>
<td>I1</td>
<td>I1</td>
<td>I2</td>
<td>I2</td>
<td>I2</td>
<td>I2</td>
<td>I2</td>
<td>I3 ..</td>
</tr>
<tr>
<td>EX</td>
<td>I1</td>
<td>nop</td>
<td>nop</td>
<td>nop</td>
<td>nop</td>
<td>nop</td>
<td>nop</td>
<td>nop ..</td>
</tr>
<tr>
<td>MA</td>
<td>I1</td>
<td>I1</td>
<td>I1</td>
<td>I1</td>
<td>I1</td>
<td>I1</td>
<td>I2</td>
<td>I3 ..</td>
</tr>
<tr>
<td>WB</td>
<td>I1</td>
<td>I1</td>
<td>I1</td>
<td>I1</td>
<td>I1</td>
<td>I1</td>
<td>I1</td>
<td>I1 ..</td>
</tr>
</tbody>
</table>
Interlock Control Logic

Diagram showing the interlock control logic with modules and signals such as PC, IR, Add, Memory, and ALU.