Complex pipelining
(L08, L09, L11)

Guowei Zhang
Dependence vs. hazard

• Dependence is a property of programs

• Whether a dependence results in a hazard is a property of pipeline organizations
Data hazard types

- **RAW**
- **WAR**
- **WAW**
  - Why?

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Source Registers</th>
<th>Target Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1: ADDI f0, f0, 0</td>
<td>R (I1)</td>
<td>W (I1)</td>
</tr>
<tr>
<td>I2: ADDI f3, f0, 3</td>
<td>R (I2)</td>
<td>R (I3)</td>
</tr>
<tr>
<td>I3: ADDI f4, f0, 4</td>
<td>W (I4)</td>
<td></td>
</tr>
<tr>
<td>I4: ADDI f0, f5, 1</td>
<td>W (I4)</td>
<td></td>
</tr>
<tr>
<td>I5: XOR f6, f6, f6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I6: ADDI f0, f7, 1</td>
<td>W (I6)</td>
<td></td>
</tr>
</tbody>
</table>
Scoreboard

• A data structure that detects hazards dynamically

• Applicable to both in-order and out-of-order issue

• Why do we need this?
  – Many execution units
  – Variable execution latency
  – Dynamic instruction scheduling
Scoreboard

• Can have many implementations!
• Example: In-order issue
  – WAR cannot happen

\[
\begin{align*}
I1: \text{ADDI} & \quad f1, \quad f0, \quad 1 \\
I2: \text{ADDI} & \quad f0, \quad \cdots \quad f2, \quad 1
\end{align*}
\]

Due to in-order issue

Register read happens before write for an instruction

– Can be simplified as Busy[FU#] and WP[reg#]
  (if WAW resolved conservatively)
Scoreboard

• What strategy does it use to resolve RAW?
  – Stall

• How about bypass?
  – Less beneficial since the register write can happen right after execution finishes
  – Can still be incorporated to allow register read and write to happen in the same cycle
Static vs. dynamic scheduling

• Reorder instructions to avoid hazards

• Static scheduling: programmer/compiler

• Dynamic scheduling: architectures
  – No need to re-compile!
  – Can handle unknown dependences and execution latencies
Out-of-order execution

• Register renaming: an approach to resolve WAR and WAW hazards (caused by name dependences)

• Design tradeoffs
  – Data-in-ROB vs. unified-register-file
  – Centralized vs. distributed
  – ROB vs. issue queue + commit queue
• Practice!

• Ask questions on Piazza

• Stay tuned for future updates on syllabus
Wish you all the best!