CSCI-564 Advanced Computer Architecture

Lecture 11: Out-of-order Execution

Bo Wu
Colorado School of Mines
• **Problem:** A true data dependency stalls dispatch of younger instructions into functional (execution) units

• **Dispatch:** Act of sending an instruction to a functional unit
Can We Do Better?

• What do the following two pieces of code have in common (with respect to execution in the previous design)?

<table>
<thead>
<tr>
<th>IMUL</th>
<th>R3 ← R1, R2</th>
<th>LD</th>
<th>R3 ← R1 (0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>R3 ← R3, R1</td>
<td>ADD</td>
<td>R3 ← R3, R1</td>
</tr>
<tr>
<td>ADD</td>
<td>R1 ← R6, R7</td>
<td>ADD</td>
<td>R1 ← R6, R7</td>
</tr>
<tr>
<td>IMUL</td>
<td>R5 ← R6, R8</td>
<td>IMUL</td>
<td>R5 ← R6, R8</td>
</tr>
<tr>
<td>ADD</td>
<td>R7 ← R9, R9</td>
<td>ADD</td>
<td>R7 ← R9, R9</td>
</tr>
</tbody>
</table>

• Answer: First ADD stalls the whole pipeline!
  – ADD cannot dispatch because its source registers unavailable
  – Later independent instructions cannot get executed

• How are the above code portions different?
  – Answer: Load latency is variable (unknown until runtime)
  – What does this affect? Think compiler vs. microarchitecture
Preventing Dispatch Stalls

• Multiple ways of doing it
• You have already seen TWO:
  – 1.
  – 2.
• What are the disadvantages of the above two?

• Thoughts to improve?
Out-of-order Execution
(Dynamic Scheduling)

• **Idea:** Move the dependent instructions out of the way of independent ones
  – Rest areas for dependent instructions: Reservation stations

• Monitor the source “values” of each instruction in the resting area

• When all source “values” of an instruction are available, “fire” (i.e. dispatch) the instruction

• **Benefit:**
  – **Latency tolerance:** Allows independent instructions to execute and complete in the presence of a long latency operation
In-order vs. Out-of-order Dispatch

- In order dispatch + precise exceptions:

```
F D E E E E R W
FD  STALL ER W
F  STALL DER W
FD E E E E E R W
FD  STALL ER W
```

- Out-of-order dispatch + precise exceptions:

```
F D E E E E E R W
FD  WAIT ER W
FD E E R W
FD  E E E E E E R W
FD  WAIT ER W
```

- 15 vs. 12 cycles
Data Dependence

Flow dependence

\[ r_3 \leftarrow r_1 \text{ op } r_2 \]  \hspace{1cm} \text{Read-after-Write (RAW)}

\[ r_5 \leftarrow r_3 \text{ op } r_4 \]

Anti dependence

\[ r_3 \leftarrow r_1 \text{ op } r_2 \]  \hspace{1cm} \text{Write-after-Read (WAR)}

\[ r_1 \leftarrow r_4 \text{ op } r_5 \]

Output-dependence

\[ r_3 \leftarrow r_1 \text{ op } r_2 \]  \hspace{1cm} \text{Write-after-Write (WAW)}

\[ r_5 \leftarrow r_3 \text{ op } r_4 \]

\[ r_3 \leftarrow r_6 \text{ op } r_7 \]
Register Renaming

ADD R1, R2, R3
ADD R1, R4, R5
ADD R6, R4, R5
ADD R1, R6, R3
ADD R7, R4, R5
ADD R8, R6, R5
ADD R8, R7, R5
Overview of Out-of-order Execution

TAG and VALUE Broadcast Bus

in order  out of order  in order