Lecture 19: Pipelining

• Today’s topics:
  - Data hazards and instruction scheduling
  - Control hazards
Example 2 – Bypassing

- Show the instruction occupying each stage in each cycle (with bypassing) if I1 is R1+R2 → R3 and I2 is R3+R4 → R5 and I3 is R3+R8 → R9.
- Identify the input latch for each input operand.

<table>
<thead>
<tr>
<th>CYC-1</th>
<th>CYC-2</th>
<th>CYC-3</th>
<th>CYC-4</th>
<th>CYC-5</th>
<th>CYC-6</th>
<th>CYC-7</th>
<th>CYC-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
</tr>
<tr>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td>I4</td>
<td>I5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D/R</td>
<td>D/R</td>
<td>D/R</td>
<td>D/R</td>
<td>D/R</td>
<td>D/R</td>
<td>D/R</td>
<td>D/R</td>
</tr>
<tr>
<td></td>
<td>I2</td>
<td>I2</td>
<td>I2</td>
<td>I4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>L3</td>
<td>L3</td>
<td>L3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALU</td>
<td>ALU</td>
<td>ALU</td>
<td>ALU</td>
<td>ALU</td>
<td>ALU</td>
<td>ALU</td>
<td>ALU</td>
</tr>
<tr>
<td></td>
<td>I1</td>
<td></td>
<td>I1</td>
<td>I1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2</td>
<td>I2</td>
<td>I3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM</td>
<td>DM</td>
<td>DM</td>
<td>DM</td>
<td>DM</td>
<td>DM</td>
<td>DM</td>
<td>DM</td>
</tr>
<tr>
<td></td>
<td>I1</td>
<td></td>
<td>I2</td>
<td>I3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I1</td>
<td>I2</td>
<td>I3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 0

\[
\begin{align*}
\text{add } & \$1, \$2, \$3 \\
\text{add } & \$5, \$1, \$4
\end{align*}
\]

Without bypassing:
\[
\begin{align*}
\text{add } & \$1, \$2, \$3: \quad \text{IF} \quad \text{DR} \quad \text{AL} \quad \text{DM} \quad \text{RW} \\
\text{add } & \$5, \$1, \$4: \quad \text{IF} \quad \text{DR} \quad \text{DR} \quad \text{DR} \quad \text{AL} \quad \text{DM} \quad \text{RW}
\end{align*}
\]

With bypassing:
\[
\begin{align*}
\text{add } & \$1, \$2, \$3: \quad \text{IF} \quad \text{DR} \quad \text{AL} \quad \text{DM} \quad \text{RW} \\
\text{add } & \$5, \$1, \$4: \quad \text{IF} \quad \text{DR} \quad \text{AL} \quad \text{DM} \quad \text{RW}
\end{align*}
\]

- Red: Point of Production
- Blue: Point of Consumption
Problem 1

add $1, $2, $3
lw $4, 8($1)
Problem 1

add $1, $2, $3
lw $4, 8($1)
Problem 2

lw $1, 8($2)
lw $4, 8($1)
Problem 2

lw $1, 8($2)
lw $4, 8($1)
Problem 3

lw $1, 8($2)
sw $1, 8($3)
Problem 3

lw $1, 8($2)
sw $1, 8($3)
Problem 4

A 7 or 9 stage pipeline, RR and RW take an entire stage

lw $1, 8($2)

add $4, $1, $3
Problem 4

A 7 or 9 stage pipeline, RR and RW take an entire stage

lw $1, 8($2)

add $4, $1, $3
Problem 4

Without bypassing: 4 stalls

With bypassing: 2 stalls

lw $1, 8($2)
add $4, $1, $3
Control Hazards

• Simple techniques to handle control hazard stalls:
  ➢ for every branch, introduce a stall cycle (note: every 6th instruction is a branch!)
  ➢ assume the branch is not taken and start fetching the next instruction – if the branch is taken, need hardware to cancel the effect of the wrong-path instruction
  ➢ fetch the next instruction (branch delay slot) and execute it anyway – if the instruction turns out to be on the correct path, useful work was done – if the instruction turns out to be on the wrong path, hopefully program state is not lost
  ➢ make a smarter guess and fetch instructions from the expected target
Control Hazards
Branch Delay Slots

a. From before

```
add $s1, $s2, $s3
if $s2 = 0 then
  Delay slot
```

Becomes

```
if $s2 = 0 then
  add $s1, $s2, $s3
```

b. From target

```
sub $t4, $t5, $t6
... 
add $s1, $s2, $s3
if $s1 = 0 then
  Delay slot
```

Becomes

```
add $s1, $s2, $s3
if $s1 = 0 then
  sub $t4, $t5, $t6
```