Lecture: Pipelining Hazards

- Topics: structural and data hazards
- HW2 posted later today; due in a week
A 5-Stage Pipeline

Source: H&P textbook
RISC/CISC Loads/Stores

Registers and memory
Complex and reduced instrs
Format of a load/store
## Pipeline Summary

<table>
<thead>
<tr>
<th>Instruction</th>
<th>RR</th>
<th>ALU</th>
<th>DM</th>
<th>RW</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD R3 ← R1, R2</td>
<td>Rd R1,R2</td>
<td>R1+R2</td>
<td>--</td>
<td>Wr R3</td>
</tr>
<tr>
<td>BEZ R1, [R5]</td>
<td>Rd R1, R5</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compare, Set PC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD R6 ← 8[R3]</td>
<td>Rd R3</td>
<td>R3+8</td>
<td>Get data</td>
<td>Wr R6</td>
</tr>
<tr>
<td>ST R6 → 8[R3]</td>
<td>Rd R3,R6</td>
<td>R3+8</td>
<td>Wr data</td>
<td>--</td>
</tr>
</tbody>
</table>
Problem 3

- For the following code sequence, show how the instrs flow through the pipeline:
  ADD R3 ← R1, R2
  LD R7 ← 8[R6]
  ST R9 → 4[R8]
  BEZ R4, [R5]
Problem 3

• For the following code sequence, show how the instrs flow through the pipeline:
  ADD  R3 ← R1, R2
  LD   R7 ← 8[R6]
  ST   R9 → 4[R8]
  BEZ  R4, [R5]
Problem 4

- Convert this C code into equivalent RISC assembly instructions

\[ a[i] = b[i] + c[i]; \]
Problem 4

• Convert this C code into equivalent RISC assembly instructions

\[ a[i] = b[i] + c[i]; \]

LD R2, [R1]  # R1 has the address for variable i
MUL R3, R2, 8  # the offset from the start of the array
ADD R7, R3, R4  # R4 has the address of a[0]
ADD R8, R3, R5  # R5 has the address of b[0]
ADD R9, R3, R6  # R6 has the address of c[0]
LD R10, [R8]  # Bringing b[i]
LD R11, [R9]  # Bringing c[i]
ADD R12, R11, R10  # Sum is in R12
ST R12, [R7]  # Putting result in a[i]
Hazards

• Structural hazards: different instructions in different stages (or the same stage) conflicting for the same resource

• Data hazards: an instruction cannot continue because it needs a value that has not yet been generated by an earlier instruction

• Control hazard: fetch cannot continue because it does not know the outcome of an earlier branch – special case of a data hazard – separate category because they are treated in different ways
Structural Hazards

• Example: a unified instruction and data cache → stage 4 (MEM) and stage 1 (IF) can never coincide

• The later instruction and all its successors are delayed until a cycle is found when the resource is free → these are pipeline bubbles

• Structural hazards are easy to eliminate – increase the number of resources (for example, implement a separate instruction and data cache)
Problem 5

- Show the instruction occupying each stage in each cycle (no bypassing)

  if I1 is R1+R2 $\rightarrow$ R3 and I2 is R3+R4 $\rightarrow$ R5 and I3 is R7+R8 $\rightarrow$ R9

<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>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>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>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>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
</tbody>
</table>
Problem 5

- Show the instruction occupying each stage in each cycle (no bypassing)
  if \( I_1 \) is \( R_1 + R_2 \rightarrow R_3 \) and \( I_2 \) is \( R_3 + R_4 \rightarrow R_5 \) and \( I_3 \) is \( R_7 + R_8 \rightarrow R_9 \)
Bypassing: 5-Stage Pipeline

Source: H&P textbook
Problem 6

- 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.
Problem 6

- 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.