Lecture: Pipelining Extensions

- Topics: bypassing, deeper pipelines, control hazards
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>Operation</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, R1+R2</td>
<td>--</td>
<td>--</td>
<td>Wr R3</td>
<td></td>
</tr>
<tr>
<td>BEZ R1, [R5]</td>
<td>Rd R1, R5</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>Compare, Set PC</td>
</tr>
<tr>
<td>LD R6 ← 8[R3]</td>
<td>Rd R3, R3+8</td>
<td>Get data</td>
<td>Wr R6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST R6 → 8[R3]</td>
<td>Rd R3,R6, R3+8</td>
<td>Wr data</td>
<td>--</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 4

• 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

- 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]
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 → R3 and I2 is R3+R4 → R5 and I3 is R7+R8 → 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 I1 is R1+R2 → R3 and I2 is R3+R4 → R5 and I3 is R7+R8 → 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 I1</td>
<td>IF I2</td>
<td>IF I3</td>
<td>IF I3</td>
<td>IF I3</td>
<td>IF I4</td>
<td>IF I5</td>
<td>IF</td>
</tr>
<tr>
<td>D/R I1</td>
<td>D/R I2</td>
<td>D/R I2</td>
<td>D/R I2</td>
<td>D/R I3</td>
<td>D/R I4</td>
<td>D/R I4</td>
<td>D/R I2</td>
</tr>
<tr>
<td>ALU I1</td>
<td>ALU I1</td>
<td>ALU I1</td>
<td>ALU I2</td>
<td>ALU I2</td>
<td>ALU I3</td>
<td>ALU I3</td>
<td>ALU I2</td>
</tr>
<tr>
<td>DM I1</td>
<td>DM I1</td>
<td>DM I1</td>
<td>DM I2</td>
<td>DM I2</td>
<td>DM I3</td>
<td>DM I3</td>
<td>DM I3</td>
</tr>
<tr>
<td>RW I1</td>
<td>RW I2</td>
<td>RW I1</td>
<td>RW I1</td>
<td>RW I1</td>
<td>RW I2</td>
<td>RW I2</td>
<td>RW I2</td>
</tr>
</tbody>
</table>
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.

<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 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.

<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 I1</td>
<td>IF I2</td>
<td>IF I3</td>
<td>IF I4</td>
<td>IF I5</td>
<td>IF</td>
<td>IF</td>
<td>IF</td>
</tr>
<tr>
<td>D/R I1</td>
<td>D/R I2</td>
<td>D/R I3</td>
<td>D/R I4</td>
<td>D/R I5</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>

L3 L3 L4 L3 L5 L3
Pipeline Implementation

- Signals for the muxes have to be generated – some of this can happen during ID
- Need look-up tables in decode stage to identify situations that merit bypassing/stalling – the number of inputs to the muxes goes up
Problem 7

• For the 5-stage pipeline (RR and RW take half a cycle)

• For the following pairs of instructions, how many stalls will the 2nd instruction experience (with and without bypassing)?

  ▪ ADD R3 ← R1+R2
    ADD R5 ← R3+R4
  ▪ LD R2 ← [R1]
    ADD R4 ← R2+R3
  ▪ LD R2 ← [R1]
    SD R3 → [R2]
  ▪ LD R2 ← [R1]
    SD R2 → [R3]
Problem 7

• For the 5-stage pipeline (RR and RW take half a cycle)

IF  D/RR  AL  DM  RW

• For the following pairs of instructions, how many stalls will the 2\textsuperscript{nd} instruction experience (with and without bypassing)?

- ADD R3 $\leftarrow$ R1+R2
  ADD R5 $\leftarrow$ R3+R4
  without: 2  with: 0

- LD R2 $\leftarrow$ [R1]
  ADD R4 $\leftarrow$ R2+R3
  without: 2  with: 1

- LD R2 $\leftarrow$ [R1]
  SD R3 $\rightarrow$ [R2]
  without: 2  with: 1

- LD R2 $\leftarrow$ [R1]
  SD R2 $\rightarrow$ [R3]
  without: 2  with: 0
Summary

• For the 5-stage pipeline, bypassing can eliminate delays between the following example pairs of instructions:
  add/sub R1, R2, R3
  add/sub/lw/sw R4, R1, R5

        lw    R1, 8(R2)
    sw    R1, 4(R3)

• The following pairs of instructions will have intermediate stalls:
  lw  R1, 8(R2)
  add/sub/lw R3, R1, R4     or     sw R3, 8(R1)

        fmul    F1, F2, F3
    fadd    F5, F1, F4