Lecture: Pipelining Hazards

- Topics: structural and data hazards
- HW2 posted later today; due in a week
Problem 1

- An unpipelined processor takes 5 ns to work on one instruction. It then takes 0.2 ns to latch its results into latches. I was able to convert the circuits into 5 equal sequential pipeline stages. Answer the following, assuming that there are no stalls in the pipeline.

- What are the cycle times in the two processors? 5.2ns and 1.2ns
- What are the clock speeds? 192 MHz and 833 MHz
- What are the IPCs? 1 and 1
- How long does it take to finish one instr? 5.2ns and 6ns
- What is the speedup from pipelining? 833/192 = 4.34
Problem 2

- An unpipelined processor takes 5 ns to work on one instruction. It then takes 0.2 ns to latch its results into latches. I was able to convert the circuits into 5 sequential pipeline stages. The stages have the following lengths: 1ns; 0.6ns; 1.2ns; 1.4ns; 0.8ns. Answer the following, assuming that there are no stalls in the pipeline.

- What is the cycle time in the new processor? **1.6ns**
- What is the clock speed? **625 MHz**
- What is the IPC? **1**
- How long does it take to finish one instr? **8ns**
- What is the speedup from pipelining? **625/192 = 3.26**
- What is the max speedup from pipelining? **5.2/0.2 = 26**
A 5-Stage Pipeline

Time (in clock cycles)

Source: H&P textbook
A 5-Stage Pipeline

Use the PC to access the I-cache and increment PC by 4
A 5-Stage Pipeline

Read registers, compare registers, compute branch target; for now, assume branches take 2 cyc (there is enough work that branches can easily take more)
A 5-Stage Pipeline

ALU computation, effective address computation for load/store
A 5-Stage Pipeline

Memory access to/from data cache, stores finish in 4 cycles
A 5-Stage Pipeline

Write result of ALU computation or load into register file
RISC/CISC Loads/Stores

Registers and memory
Complex and reduced instrs
Format of a load/store
Problem 3

• Convert this C code into equivalent RISC assembly instructions

\[ a[i] = b[i] + c[i]; \]
**Problem 3**

- 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]
```
## 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>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 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 instructions 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 I3</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 I3</td>
</tr>
<tr>
<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 I3</td>
<td>ALU I3</td>
</tr>
<tr>
<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>
<td>DM I3</td>
</tr>
<tr>
<td>RW I1</td>
<td>RW I2</td>
<td>RW I2</td>
<td>RW I2</td>
<td>RW I1</td>
<td>RW I1</td>
<td>RW I1</td>
<td>RW I1</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>
Show the instruction occupying each stage in each cycle (with 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 $R_3 + R_8 \rightarrow R_9$.

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>$I_1$</td>
<td>$I_2$</td>
<td>$I_3$</td>
<td>$I_4$</td>
<td>$I_5$</td>
<td>$I_1$</td>
<td>$I_2$</td>
<td>$I_3$</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>$I_1$</td>
<td>$I_2$</td>
<td>$I_3$</td>
<td>$I_4$</td>
<td>$I_5$</td>
<td>$I_1$</td>
<td>$I_2$</td>
<td>$I_3$</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>L3</td>
<td>L3</td>
<td>L4</td>
<td>L5</td>
<td>L3</td>
<td>L3</td>
<td>L3</td>
<td>L3</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>