CMSC 411
Computer Systems Architecture
Lecture 4
Basic Pipelining
Alan Sussman
als@cs.umd.edu

Administrivia

• Homework problems for Unit 1 due Thursday

5 Steps of MIPS Datapath
Figure A.17, Page A-29

Instruction Fetch
Instr. Decode
Execute Addr. Calc
Memory Access
Write Back

IR <= mem[PC];
PC <= PC + 4
Reg[IRe] <= Reg[IRr] + Reg[IRu]

5 Steps of MIPS Datapath
Figure A.18, Page A-31

Instruction Fetch
Instr. Decode
Execute Addr. Calc
Memory Access
Write Back

IR <= mem[PC];
PC <= PC + 4
A <= Reg[IRr];
B <= Reg[IRu];
rslt <= A opB
WB <= rslt
Reg[IRe] <= WB
**Inst. Set Processor Controller**

IR <= mem[PC];
PC <= PC + 4

A <= Reg[IR.rs];
B <= Reg[IR.rt]

if bop(A, b)
PC <= PC+IR.im

r <= A opIRop B
WB <= r
Reg[IR.rd] <= WB

**Visualizing Pipelining**

**Pipelining is not quite that easy!**

- Limits to pipelining: **Hazards** prevent next instruction from executing during its designated clock cycle
  - **Structural hazards**: HW cannot support this combination of instructions (single person to fold and put clothes away)
  - **Data hazards**: Instruction depends on result of prior instruction still in the pipeline (missing sock)
  - **Control hazards**: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).
One Memory Port/Structural Hazards
(Similar to Figure A.5, Page A-15)

<table>
<thead>
<tr>
<th>Time (clock cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle 1</td>
</tr>
<tr>
<td>Load</td>
</tr>
<tr>
<td>Instr 1</td>
</tr>
<tr>
<td>Instr 2</td>
</tr>
<tr>
<td>Stall</td>
</tr>
<tr>
<td>Instr 3</td>
</tr>
</tbody>
</table>

How do you “bubble” the pipe?

CMSC 411 - 4 (from Patterson)

Speed Up Equation for Pipelining

\[ CPI_{\text{pipelined}} = \text{Ideal CPI} + \text{Average Stall cycles per Inst} \]

\[ \text{Speedup} = \frac{\text{Ideal CPI} \times \text{Pipeline depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{\text{unpipelined}}}{\text{Cycle Time}_{\text{pipelined}}} \]

For simple RISC pipeline, ideal CPI = 1:

\[ \text{Speedup} = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{\text{unpipelined}}}{\text{Cycle Time}_{\text{pipelined}}} \]

Example: Dual-port vs. Single-port

- Machine A: Dual ported memory (“Harvard Architecture”)
- Machine B: Single ported memory, but its pipelined implementation has a 1.05 times faster clock rate
- Ideal CPI = 1 for both
- Loads are 40% of instructions executed
  \[ \text{SpeedUp}_A = \frac{\text{Pipeline Depth}}{1 + 0} \times \frac{\text{clock}_{\text{unpipe}}}{\text{clock}_{\text{pipe}}} = \text{Pipeline Depth} \]
  \[ \text{SpeedUp}_B = \frac{\text{Pipeline Depth}}{1 + 0.4 \times 1} \times \frac{\text{clock}_{\text{unpipe}}}{\text{clock}_{\text{unpipe}} / 1.05} = (\text{Pipeline Depth} / 1.4) \times 1.05 = 0.75 \times \text{Pipeline Depth} \]
  \[ \text{SpeedUp}_A / \text{SpeedUp}_B = \frac{\text{Pipeline Depth}}{0.75 \times \text{Pipeline Depth}} = 1.33 \]
- Machine A is 1.33 times faster

CMSC 411 - 4 (from Patterson)

Data Hazard on R1
(Figure A.6, Page A-16)

Time (clock cycles)

CMSC 411 - 4 (from Patterson)
Three Generic Data Hazards

- **Read After Write (RAW)**
  Instr\(_J\) tries to read operand before Instr\(_I\) writes it

  \[
  \begin{align*}
  I: & \quad \text{add } r1, r2, r3 \\
  J: & \quad \text{sub } r4, r1, r3
  \end{align*}
  \]

- Caused by a “Dependence” (in compiler nomenclature). This hazard results from an actual need for communication.

Three Generic Data Hazards

- **Write After Write (WAW)**
  Instr\(_J\) writes operand before Instr\(_I\) writes it.

  \[
  \begin{align*}
  I: & \quad \text{sub } r4, r1, r3 \\
  J: & \quad \text{add } r1, r2, r3 \\
  K: & \quad \text{mul } r6, r1, r7
  \end{align*}
  \]

  - Called an “anti-dependence” by compiler writers.
  - This results from reuse of the name “r1”.

  - Can’t happen in MIPS 5 stage pipeline because:
    - All instructions take 5 stages, and
    - Reads are always in stage 2, and
    - Writes are always in stage 5

  - Will see WAR and WAW in more complicated pipes

Forwarding to Avoid Data Hazard

Figure A.7, Page A-18

Time (clock cycles)

\[
\begin{align*}
\text{add } r1, r2, r3 \\
\text{sub } r4, r1, r3 \\
\text{and } r6, r1, r7 \\
\text{or } r8, r1, r9 \\
xor r10, r1, r11
\end{align*}
\]
HW Change for Forwarding
Figure A.23, Page A-37