Lecture 8: Pipeline Complications and Case Study—MIPS R4000

Professor Alvin R. Lebeck
Computer Science 220
Fall 1999

Administrivia

• Get Ready for Floyd!
• Homework #1 Due Friday
• Read Chapter 4
Review: Hazards

Data Hazards
- RAW
  - only one that can occur in current DLX pipeline
- WAR, WAW
- Data Forwarding (Register Bypassing)
  - send data from one stage to another bypassing the register file
- Still have load use delay

Structural Hazards
- Replicate Hardware, scheduling

Control Hazards
- Compute condition and target early (delayed branch)

Review: Speed Up Equation for Pipelining

\[
\text{CPI}_{\text{pipelined}} = \text{Ideal CPI} + \text{Pipeline stall clock cycles per instr}
\]

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

\[
\text{Speedup} = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Clock Cycle}_{\text{unpipelined}}}{\text{Clock Cycle}_{\text{pipelined}}}
\]
Review: Evaluating Branch Alternatives

- **Two part solution:**
  - Determine branch taken or not sooner, AND
  - Compute taken branch address earlier

\[
Pipline \text{ speedup} = \frac{\text{Pipeline depth}}{1 + \text{Branch frequency} \times \text{Branch penalty}}
\]

<table>
<thead>
<tr>
<th>Scheduling scheme</th>
<th>Branch penalty</th>
<th>CPI speedup v. unpipelined</th>
<th>speedup v. stall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stall pipeline</td>
<td>3</td>
<td>1.42</td>
<td>3.5</td>
</tr>
<tr>
<td>Predict taken</td>
<td>1</td>
<td>1.14</td>
<td>4.4</td>
</tr>
<tr>
<td>Predict not taken</td>
<td>1</td>
<td>1.09</td>
<td>4.5</td>
</tr>
<tr>
<td>Delayed branch</td>
<td>0.5</td>
<td>1.07</td>
<td>4.6</td>
</tr>
</tbody>
</table>

Review: Evaluating Branch Prediction Strategies

- **Two strategies**
  - Backward branch predict taken, forward branch not taken
  - Profile-based prediction: record branch behavior, predict branch based on prior run

- “Instructions between mispredicted branches” a better metric than misprediction
Pipelining Complications

• **Interrupts (Exceptions)**
  – 5 instructions executing in 5 stage pipeline
  – How to stop the pipeline?
  – How to restart the pipeline?
  – Who caused the interrupt?

<table>
<thead>
<tr>
<th>Stage</th>
<th>Problem interrupts occurring</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>Page fault on instruction fetch; misaligned memory access; memory-protection violation</td>
</tr>
<tr>
<td>ID</td>
<td>Undefined or illegal opcode</td>
</tr>
<tr>
<td>EX</td>
<td>Arithmetic interrupt</td>
</tr>
<tr>
<td>MEM</td>
<td>Page fault on data fetch; misaligned memory access; memory-protection violation</td>
</tr>
</tbody>
</table>

• **Simultaneous exceptions in > 1 pipeline stage**
  – Load with data page fault in MEM stage
  – Add with instruction page fault in IF stage

• **Solution #1**
  – Interrupt status vector per instruction
  – Defer check til last stage, kill state update if exception

• **Solution #2**
  – Interrupt ASAP
  – Restart everything that is incomplete

• **Exception in branch delay slot,**
  – SW needs two PCs

• **Another advantage for state update late in pipeline!**
Pipeline Complications

• Complex Addressing Modes and Instructions
  • Address modes: Autoincrement causes register change during instruction execution
    – Interrupts? Need to restore register state
    – Adds WAR and WAW hazards since writes no longer last stage

• Memory-Memory Move Instructions
  – Must be able to handle multiple page faults
  – Long-lived instructions: partial state save on interrupt

• Condition Codes

Pipeline Complications: Floating Point

© Alvin R. Lebeck 1999
Pipelining Complications

- **Floating Point:** long execution time
- Also, may pipeline FP execution unit so they can initiate new instructions without waiting full latency

<table>
<thead>
<tr>
<th>FP Instruction</th>
<th>Latency</th>
<th>Initiation Rate</th>
<th>(MIPS R4000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add, Subtract</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Multiply</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Divide</td>
<td>36</td>
<td>35</td>
<td>(interrupts,</td>
</tr>
<tr>
<td>Square root</td>
<td>112</td>
<td>111</td>
<td>WAW, WAR)</td>
</tr>
<tr>
<td>Negate</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Absolute value</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>FP compare</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Cycles before use result Cycles before issue instr of same type

Summary of Pipelining Basics

- Hazards limit performance
  - Structural: need more HW resources
  - Data: need forwarding, compiler scheduling
  - Control: early evaluation & PC, delayed branch, prediction
- Increasing length of pipe increases impact of hazards; pipelining helps instruction bandwidth, not latency
- Compilers reduce cost of data and control hazards
  - Load delay slots
  - Branch delay slots
  - Branch prediction
- Interrupts, Instruction Set, FP makes pipelining harder
- Q: How would you handle context switches?
Case Study: MIPS R4000
(100 MHz to 200 MHz)

• 8 Stage Pipeline:
  – IF—first half of fetching of instruction; PC selection happens here as well as initiation of instruction cache access.
  – IS—second half of access to instruction cache.
  – RF—instruction decode and register fetch, hazard checking and also instruction cache hit detection.
  – EX—execution, which includes effective address calculation, ALU operation, and branch target computation and condition evaluation.
  – DF—data fetch, first half of access to data cache.
  – DS—second half of access to data cache.
  – TC—tag check, determine whether the data cache access hit.
  – WB—write back for loads and register-register operations.

• 8 Stages: What is impact on Load delay? Branch delay? Why?

TWO Cycle
Load Latency

<table>
<thead>
<tr>
<th>TWO Cycle</th>
<th>IF</th>
<th>IS</th>
<th>RF</th>
<th>EX</th>
<th>DF</th>
<th>DS</th>
<th>TC</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IF</td>
<td>IS</td>
<td>RF</td>
<td>EX</td>
<td>DF</td>
<td>DS</td>
<td>TC</td>
<td>WB</td>
</tr>
<tr>
<td></td>
<td>IF</td>
<td>IS</td>
<td>RF</td>
<td>EX</td>
<td>DF</td>
<td>DS</td>
<td>TC</td>
<td>WB</td>
</tr>
<tr>
<td></td>
<td>IF</td>
<td>IS</td>
<td>RF</td>
<td>EX</td>
<td>DF</td>
<td>DS</td>
<td>TC</td>
<td>WB</td>
</tr>
</tbody>
</table>

THREE Cycle
Branch Latency

<table>
<thead>
<tr>
<th>THREE Cycle</th>
<th>IF</th>
<th>IS</th>
<th>RF</th>
<th>EX</th>
<th>DF</th>
<th>DS</th>
<th>TC</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IF</td>
<td>IS</td>
<td>RF</td>
<td>EX</td>
<td>DF</td>
<td>DS</td>
<td>TC</td>
<td>WB</td>
</tr>
<tr>
<td></td>
<td>IF</td>
<td>IS</td>
<td>RF</td>
<td>EX</td>
<td>DF</td>
<td>DS</td>
<td>TC</td>
<td>WB</td>
</tr>
<tr>
<td></td>
<td>IF</td>
<td>IS</td>
<td>RF</td>
<td>EX</td>
<td>DF</td>
<td>DS</td>
<td>TC</td>
<td>WB</td>
</tr>
</tbody>
</table>

(conditions evaluated during EX phase)

Delay slot plus two stalls
Branch likely cancels delay slot if not taken
MIPS R4000 Floating Point

- FP Adder, FP Multiplier, FP Divider
- Last step of FP Multiplier/Divider uses FP Adder HW
- 8 kinds of stages in FP units:
  - Stage | Functional unit | Description
  - A     | FP adder       | Mantissa ADD stage
  - D     | FP divider     | Divide pipeline stage
  - E     | FP multiplier  | Exception test stage
  - M     | FP multiplier  | First stage of multiplier
  - N     | FP multiplier  | Second stage of multiplier
  - R     | FP adder       | Rounding stage
  - S     | FP adder       | Operand shift stage
  - U     | Unpack FP numbers

MIPS FP Pipe Stages

| FP Instr       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ...
|----------------|---|---|---|---|---|---|---|---|---
| Add, Subtract  | U | S+A| A+R| R+S|    |    |    |    |   
| Multiply       | U | E+M| M  | M  | M  | M  | N | N+R| R  
| Divide         | U | A  | R  | D^2^8 | ... | D+A | D+R, D+R, D+A, D+R, A, R 
| Square root    | U | E  | (A+R)^10^8 | ... | A | R  
| Negate         | U | S  |    |    |    |    |    |    |   
| Absolute value | U | S  |    |    |    |    |    |    |   
| FP compare     | U | A  | R  |    |    |    |    |    |   

Stages:

- M  First stage of multiplier
- N  Second stage of multiplier
- R  Rounding stage
- S  Operand shift stage
- U  Unpack FP numbers

© Alvin R. Lebeck 1999
R4000 Performance

- Not ideal CPI of 1:
  - Load stalls (1 or 2 clock cycles)
  - Branch stalls (2 cycles + unfilled slots)
  - FP result stalls: RAW data hazard (latency)
  - FP structural stalls: Not enough FP hardware (parallelism)

Next Time

- Homework #1 is Due
- Instruction Level Parallelism (ILP)
- Read Chapter 4