18-447 Lecture 10: Pipelined Implementations

James C. Hoe
Dept of ECE, CMU
February 23, 2009

Announcements: Project 1 is due this week
Midterm graded, results posted

Handouts: H09 Homework 3 (on Blackboard)
Graded Midterms
Midterm solutions with statistics

Doing laundry more quickly: in theory

- "place one dirty load of clothes in the washer"
- "when the washer is finished, place the wet load in the dryer"
- "when the dryer is finished, place the dry load on a table and fold"
- "when folding is finished, ask your roommate (??) to put the clothes away"

- steps to do a load are sequentially dependent
- no dependence between different loads

Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]
Doing laundry more quickly: in theory

- 4-loads of laundry in parallel
- no additional resources
- throughput increased by 4
- latency per load is the same

Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]

Doing laundry more quickly: in practice

the slowest step decides throughput

Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]
Doing laundry more quickly: in practice

Throughput restored (2 loads per hour) using 2 dryers

Pipeline Idealism

Motivation: Increase throughput with little increase in hardware

- Repetition of identical operations
  The same operation is repeated on a large number of different inputs
- Repetition of independent operations
  No ordering dependencies between repeated operations
- Uniformly partitionable suboperations
  Can be evenly divided into uniform-latency suboperations (that do not share resources)

Good examples: automobile assembly line, doing laundry, but instruction pipeline???
Ideal Pipelining

- Combinational logic: $BW = \frac{1}{T}$
  - $BW = \frac{1}{T}$

- T/2 ps: $BW = \frac{2}{T}$

- T/3 ps: $BW = \frac{3}{T}$

Performance Model

- Nonpipelined version with delay $T$
  $BW = \frac{1}{T+S}$ where $S$ = latch delay

- k-stage pipelined version
  $BW_{k\text{-stage}} = \frac{1}{T/k + S}$
  $BW_{max} = \frac{1}{1 \text{ gate delay} + S}$
Cost Model

- Nonpipelined version with combinational cost $G$
  \[ \text{Cost} = G + L \text{ where } L = \text{latch cost} \]

- $k$-stage pipelined version
  \[ \text{Cost}_{k\text{-stage}} = G + Lk \]

Cost/Performance Trade-off

[Peter M. Kogge, 1981]

Cost/Performance:
\[ \frac{\text{Cost}}{\text{Performance}} = \frac{Lk + G}{1/(T/k + S)} = (Lk + G)(T/k + S) \]
\[ = LT + GS + LSk + GT/k \]

Optimal Cost/Performance: find min. $C/P$ w.r.t. choice of $k$

\[ \frac{d}{dk} \left( \frac{Lk + G}{1/(T/k + S)} \right) = 0 + 0 + LS - \frac{GT}{k^2} \]
\[ LS - \frac{GT}{k^2} = 0 \]
\[ k_{opt} = \sqrt{\frac{GT}{LS}} \]
The Reality of Pipelining Instruction Execution . . .

Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]

RISC Instruction Processing

- 5 generic steps
  - instruction fetch
  - instruction decode and operand fetch
  - ALU/execute
  - memory access (not required by non-mem instructions)
  - write-back

Based on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]
Dividing into Stages

Is this the correct partitioning? Why not 4 or 6 stages? Why not different boundaries

Based on Figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]

Pipelining

5-stage speedup is 4, not 5 as predicated by the ideal model

Based on Figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]
Pipeline Registers

No resource is used by more than 1 stage!

Pipelined Operation

All instruction classes must follow the same path and timing through the pipeline stages. Any performance impact?
Illustrating Pipeline Operation: Operation View
Illustrating Pipeline Operation: Resource View

<table>
<thead>
<tr>
<th></th>
<th>t₀</th>
<th>t₁</th>
<th>t₂</th>
<th>t₃</th>
<th>t₄</th>
<th>t₅</th>
<th>t₆</th>
<th>t₇</th>
<th>t₈</th>
<th>t₉</th>
<th>t₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>I₀</td>
<td>I₁</td>
<td>I₂</td>
<td>I₃</td>
<td>I₄</td>
<td>I₅</td>
<td>I₆</td>
<td>I₇</td>
<td>I₈</td>
<td>I₉</td>
<td>I₁₀</td>
</tr>
<tr>
<td>ID</td>
<td>I₀</td>
<td>I₁</td>
<td>I₂</td>
<td>I₃</td>
<td>I₄</td>
<td>I₅</td>
<td>I₆</td>
<td>I₇</td>
<td>I₈</td>
<td>I₉</td>
<td></td>
</tr>
<tr>
<td>EX</td>
<td>I₀</td>
<td>I₁</td>
<td>I₂</td>
<td>I₃</td>
<td>I₄</td>
<td>I₅</td>
<td>I₆</td>
<td>I₇</td>
<td>I₈</td>
<td>I₉</td>
<td></td>
</tr>
<tr>
<td>MEM</td>
<td>I₀</td>
<td>I₁</td>
<td>I₂</td>
<td>I₃</td>
<td>I₄</td>
<td>I₅</td>
<td>I₆</td>
<td>I₇</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WB</td>
<td>I₀</td>
<td>I₁</td>
<td>I₂</td>
<td>I₃</td>
<td>I₄</td>
<td>I₅</td>
<td>I₆</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Control Points

Identical set of control points as the single-cycle datapath!!
Sequential Control: Special Case

- For a given instruction
  - same control settings as single-cycle, but
  - control signals required at different cycles, depending on stage

⇒ decode once using the same logic as single-cycle and buffer control signals until consumed

⇒ or carry relevant "instruction word/field" down the pipeline and decode locally within each stage (still same logic)

Pipelined Control
Instruction Pipeline Reality

- Identical operations ... NOT!
  - unifying instruction types
    - coalescing instruction types into one "multi-function" pipe
    - external fragmentation (some idle stages)
- Uniform Suboperations ... NOT!
  - balance pipeline stages
    - stage quantization to yield balanced stages
    - internal fragmentation (some too-fast stages)
- Independent operations ... NOT!
  - resolve data and resource hazards
    - duplicate contended resources
    - inter-instruction dependency detection and resolution

MIPS ISA features are engineered for improved pipelineability

How to do better on Midterm 2

- Read the text before lecture
- Pay attention in lecture
- Do homework
- Do lab
- Ask questions in lecture, lab, office hour
- Eat breakfast every morning
- Don’t run with scissors

...... Do you really need me to tell you any of these?