Pipeline Computations

Pipelined Addition &
System of Linear Equations

CS370 Parallel Processing
Spring 2014
Pipelining

• Introduction to pipelining
• Instruction pipeline vs. software pipeline
  – RISC pipeline (MISP Instruction Set)
  – Learned in CS365 Computer Architecture
  – Software pipeline is at a higher level and more general
Laundry analogy

Task order

A
B
C
D
Instruction Pipeline

Program execution order
(in instructions)

Iw $10, 20($1)
sub $11, $2, $3
add $12, $3, $4
Iw $13, 24($1)
add $14, $5, $6

Time (in clock cycles)
CC 1  CC 2  CC 3  CC 4  CC 5  CC 6  CC 7  CC 8  CC 9
Pipelined Computations

• Definition
  – Original problem is divided into series of tasks that can be completed one after the other
  – Each task executed by a separate process or processor
When to use Pipelining

• Back to laundry, will pipelining be useful if:
  – Only have 1 load?
  – Washing takes 30 mins
  – Drying takes 1 hour
  – Folding takes 20 mins
  – Storing takes 10 mins
When to use Pipelining

• If the problem can be divided into a series of sequential tasks
• pipelining can increase execution speed under the following three types of computations
Different “types” of Pipelines

1. If more than 1 instance of the complete problem is to be executed
2. If a series of data items must be processed, each requiring multiple operations
3. If information to start the next process can be passed forward before the process has completed all its internal operation
“Type 1” Pipeline

- Execution time = \((p - 1) + m\) cycles,
  for a \(p\)-stage pipeline with \(m\) instances.
Laundry analogy

Also “type 1” pipeline, different diagram representation
“Type 1” Pipeline

• Alternative space-time diagram
If a series of data items must be processed, each requiring multiple operations.
“Type 3” Pipeline

If information to start the next process can be passed forward before the process has completed all its internal operation.

Information transfer sufficient to start next process

Information passed to next stage

(a) Processes with the same execution time

(b) Processes not with the same execution time
Computing Platform for Pipelining

Multiprocessor system with a line configuration

Pipeline may not be the best structure for a cluster, why?
Computing Platform for Pipelining

• Do we need one processor per stage?
  – No, if # stages > # processors, a group of stages can be assigned to each processor
Pipelined Addition

• Adding numbers

• Type 1 pipeline computation, if more than 1 instance of the complete problem is to be executed
Sample Code for Pipelined Addition

- **P₀**, first process
  send(&number, P₁);

- **Pᵢ**
  recv(&accumulation, Pᵢ₋₁)
  accumulation += number;
  send(&accumulation, Pᵢ₊₁);

- **Pₙ₋₁**, last process
  recv(&number, Pₙ₋₂);
  accumulation += number;
Why pipelining for addition?

- Pointless for adding 1 series of numbers
- Similar to having 1 load of laundry
- How about multiple series of numbers?
Pipelined Arithmetic

• Instead of addition, other arithmetic operations can also be pipelined
• Similar steps, just need to replace addition by another operation
• The final result is in the last process
Example of Pipelined Addition

A[] = {1, 2, 3}
B[] = {4, 5, 6}

2 arrays, A[] & B[], 3 stages pipeline, P₀ – P₂

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>3+3</th>
<th>9+6</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₂</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P₁</td>
<td>1+2</td>
<td>4+5</td>
<td></td>
</tr>
<tr>
<td>P₀</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
“Type 1” Pipeline

- Execution time = \((p - 1) + m\) cycles, for a \(p\)-stage pipeline with \(m\) instances

<table>
<thead>
<tr>
<th>P0</th>
<th>Instance 1</th>
<th>Instance 2</th>
<th>Instance 3</th>
<th>Instance 4</th>
<th>Instance 5</th>
<th>Instance 6</th>
<th>Instance 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Instance 1</td>
<td>Instance 2</td>
<td>Instance 3</td>
<td>Instance 4</td>
<td>Instance 5</td>
<td>Instance 6</td>
<td>Instance 7</td>
</tr>
<tr>
<td>P2</td>
<td>Instance 1</td>
<td>Instance 2</td>
<td>Instance 3</td>
<td>Instance 4</td>
<td>Instance 5</td>
<td>Instance 6</td>
<td>Instance 7</td>
</tr>
<tr>
<td>P3</td>
<td>Instance 1</td>
<td>Instance 2</td>
<td>Instance 3</td>
<td>Instance 4</td>
<td>Instance 5</td>
<td>Instance 6</td>
<td>Instance 7</td>
</tr>
<tr>
<td>P4</td>
<td>Instance 1</td>
<td>Instance 2</td>
<td>Instance 3</td>
<td>Instance 4</td>
<td>Instance 5</td>
<td>Instance 6</td>
<td>Instance 7</td>
</tr>
<tr>
<td>P5</td>
<td>Instance 1</td>
<td>Instance 2</td>
<td>Instance 3</td>
<td>Instance 4</td>
<td>Instance 5</td>
<td>Instance 6</td>
<td>Instance 7</td>
</tr>
</tbody>
</table>

If more than 1 instance of the complete problem is to be executed.
“Type 1” Pipeline

- Alternative space-time diagram

| Instance 0 | \( P_0 \) | \( P_1 \) | \( P_2 \) | \( P_3 \) | \( P_4 \) | \( P_5 \) |
| Instance 1 | \( P_0 \) | \( P_1 \) | \( P_2 \) | \( P_3 \) | \( P_4 \) | \( P_5 \) |
| Instance 2 | \( P_0 \) | \( P_1 \) | \( P_2 \) | \( P_3 \) | \( P_4 \) | \( P_5 \) |
| Instance 3 | \( P_0 \) | \( P_1 \) | \( P_2 \) | \( P_3 \) | \( P_4 \) | \( P_5 \) |
| Instance 4 | \( P_0 \) | \( P_1 \) | \( P_2 \) | \( P_3 \) | \( P_4 \) | \( P_5 \) |
Pipelining System of Linear Equation

• Pipelining to solve system of linear equation
• Type 3 pipeline computation, if information to start the next process can be passed forward before the process has completed all its internal operation
System of Linear Equations

• Upper-triangular form,
• a’s b’s are constants, x’s are unknowns

\[a_{(n-1,0)}x_0 + a_{(n-1,1)}x_1 + a_{(n-1,2)}x_2 \ldots + a_{(n-1,n-1)}x_{n-1} = b_{n-1}\]

\[
\ldots
\]

\[a_{(2,0)}x_0 + a_{(2,1)}x_1 + a_{(2,2)}x_2 = b_2\]

\[a_{(1,0)}x_0 + a_{(1,1)}x_1 = b_1\]

\[a_{(0,0)}x_0 = b_0\]
Solution Through Back Substitution

• First, the unknown $x_0$ is found from the last equation, $a_{(0,0)} x_0 = b_0$

$$x_0 = \frac{b_0}{a_{0,0}}$$

• Then $x_1$ is found, substitute $x_0$ into the next equation, $a_{(1,0)} x_0 + a_{(1,1)} x_1 = b_1$

$$x_1 = \frac{b_1 - a_{1,0} x_0}{a_{1,1}}$$

• And so on until all unknowns are found
Pipelining Back Substitution

- First stage computes $x_0$, and pass it on
- 2\textsuperscript{nd} stage computes receives $x_0$, computes $x_1$, and passes both $x_0$ and $x_1$ to the next stage
- So forth and so on,
Pipelining Back Substitution

Processes

$P_0$, $P_1$, $P_2$, $P_3$, $P_4$, $P_5$

Time

First value passed onward

Final computed value
Example for

Pipelined System of Linear Equations

• 5 equations, 5 unknowns

\[5x_0 + 2x_1 + 10x_2 + x_3 + 2x_4 = 65\]
\[x_0 + 7x_1 + 4x_3 = 45\]
\[2x_0 + 3x_1 + 3x_2 = 40\]
\[5x_0 + x_1 = 25\]
\[2x_0 = 10\]

\[x_0 = 5\]
\[x_1 = 0\]
\[x_2 = 10\]
\[x_3 = 10\]
\[x_4 = 15\]
“Type 3” Pipeline

If information to start the next process can be passed forward before the process has completed all its internal operation.

Information transfer sufficient to start next process

Time

Information passed to next stage

(a) Processes with the same execution time

(b) Processes not with the same execution time
Post-test

• Similar to the previous exercises that you’ve done last time
• This time you’re better prepared
• Try your best!