Announcements

● **Reading Assignment**
  – Chapter 5. The Processor: Datapath and Control
    Sections 5.6
  – Chapter 6. Enhancing Performance with Pipelining
    Sections 6.1 - 6.10

● **Homework 5: Due Mon., November 14th in class**
  5.49, 5.50
  6.1, 6.2, 6.3, 6.4, 6.6, 6.14, 6.15, 6.17

● **Quiz**
  **When:** Mon., November 14th
  **Topic:** Pipelining and Hazards
# Course Schedule

<table>
<thead>
<tr>
<th>Lecture #</th>
<th>Date</th>
<th>Day</th>
<th>Lecture Topic</th>
<th>Quiz Topic</th>
<th>Homework Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9/26</td>
<td>Monday</td>
<td>Introduction, Ch. 1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>9/28</td>
<td>Wednesday</td>
<td>ISA, Ch. 2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>10/3</td>
<td>Monday</td>
<td>Arithmetic Part 1, Ch. 4</td>
<td>ISA</td>
<td>#1</td>
</tr>
<tr>
<td>4</td>
<td>10/5</td>
<td>Wednesday</td>
<td>Arithmetic Part 2, Ch. 4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>10/10</td>
<td>Monday</td>
<td>Performance, Ch. 3</td>
<td>Arithmetic</td>
<td>#2</td>
</tr>
<tr>
<td>6</td>
<td>10/12</td>
<td>Wednesday</td>
<td>Single cycle CPU, Ch. 5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>10/17</td>
<td>Monday</td>
<td>Single cycle CPU, Ch. 5</td>
<td>Performance</td>
<td>#3</td>
</tr>
<tr>
<td>8</td>
<td>10/19</td>
<td>Wednesday</td>
<td>Multi-cycle CPU, Ch. 5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>10/24</td>
<td>Monday</td>
<td>Multi-cycle CPU, Ch. 5</td>
<td>Single Cycle CPU</td>
<td>#4</td>
</tr>
<tr>
<td>10</td>
<td>10/26</td>
<td>Wednesday</td>
<td>Review for the Midterm</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10/31</td>
<td>Monday</td>
<td>Mid-term Exam</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>11/2</td>
<td>Wednesday</td>
<td>Exceptions, Ch. 5 and Pipelining, Ch. 6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>11/7</td>
<td>Monday</td>
<td>Pipelining, Ch. 6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>11/9</td>
<td>Wednesday</td>
<td>Data and control hazards, Ch. 6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>11/14</td>
<td>Monday</td>
<td>Data and control hazards, Ch. 6</td>
<td>Pipeline Hazards</td>
<td>#5</td>
</tr>
<tr>
<td>15</td>
<td>11/16</td>
<td>Wednesday</td>
<td>Memory &amp; cache design, Ch. 7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>11/21</td>
<td>Monday</td>
<td>Memory &amp; cache design, Ch. 7</td>
<td>Cache</td>
<td>#6</td>
</tr>
<tr>
<td>17</td>
<td>11/23</td>
<td>Wednesday</td>
<td>Virtual Memory &amp; cache design, Ch. 7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>11/28</td>
<td>Monday</td>
<td>Course Review</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12/8</td>
<td>Thursday</td>
<td>Final Exam 7 - 10 PM</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Would our pipeline design work in any case?

- What happens when...
  
  add $3, $10, $11
  
lw $8, 1000($3)
  
  sub $11, $8, $7
Data Hazards

- When a result is needed in the pipeline before it is available, a “data hazard” occurs.

<table>
<thead>
<tr>
<th>Value of register $2:</th>
<th>CC 1</th>
<th>CC 2</th>
<th>CC 3</th>
<th>CC 4</th>
<th>CC 5</th>
<th>CC 6</th>
<th>CC 7</th>
<th>CC 8</th>
<th>CC 9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10/20</td>
<td>-20</td>
<td>-20</td>
<td>-20</td>
<td>-20</td>
</tr>
</tbody>
</table>

Program execution order (in instructions)

- sub $2, $1, $3
- and $12, $2, $5
- or $13, $6, $2
- add $14, $2, $2
- sw $15, 100($2)

- Result of SUB instruction not available until CC5 or later!
Software Solutions to Data Hazards

• Have compiler guarantee no hazards
  – Rearrange code to remove hazard
    ➢ Not possible every time

• Insert “nops”
  – Where do we insert the “nops”? 

    sub $2, $1, $3
    and $12, $2, $5
    or $13, $6, $2
    add $14, $2, $2
    sw $15, 100($2)

• Problem: Data hazards are very common!
  – “nops” really slows us down!
Hardware Solutions to Data Hazards

- Stall the pipeline (insert bubbles)
  - Data hazards are too common
    - Same as “nops”
  - Severe performance hit

- Forward the data as soon as it is available
  - Modify the pipeline to forward (bypass data)
Forwarding

- Use temporary results, don’t wait for them to be written
Forwarding

a. No forwarding

b. With forwarding

Forwarding

Pramod Argade  
CSE 141, Fall 2005
Reducing **EX** Data Hazards Through Forwarding
Reducing **EX** Data Hazards Through Forwarding

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs)) ForwardA = 10

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRt)) ForwardB = 10
Reducing **MEM** Data Hazards Through Forwarding
Reducing **MEM** Data Hazards Through Forwarding

```plaintext
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01
```
Simultaneous EX/MEM Forwarding

- Consider following code
  
  ```
  add $1, $1, $2
  add $1, $1, $3
  add $1, $1, $4
  ...
  ```
Simultaneous EX/MEM Forwarding

- Consider following code
  \[\begin{align*}
  \text{add} & \, \$1, \, \$1, \, \$2 \\
  \text{add} & \, \$1, \, \$1, \, \$3 \\
  \text{add} & \, \$1, \, \$1, \, \$4 \\
  \ldots
  \end{align*}\]

- Must forward from MEM stage

- Disable WB stage forwarding
  \[
  \begin{align*}
  \text{if (MEM/WB.RegWrite} \\
  \text{and (MEM/WB.RegisterRd} \neq \text{0}) \\
  \text{and (EX/MEM.RegisterRd} \neq \text{ID/EX.RegisterRs}) \\
  \text{and (MEM/WB.RegisterRd} = \text{ID/EX.RegisterRs}) \text{) ForwardA} = \text{01} \\
  \text{if (MEM/WB.RegWrite} \\
  \text{and (MEM/WB.RegisterRd} \neq \text{0}) \\
  \text{and (EX/MEM.RegisterRd} \neq \text{ID/EX.RegisterRt}) \\
  \text{and (MEM/WB.RegisterRd} = \text{ID/EX.RegisterRt}) \text{) ForwardB} = \text{01}
  \end{align*}
  \]
Forwarding in Action

```
sub $1, $12, $3
and $12, $3, $4
add $3, $8, $11
```

Memory Access Write Back
Forwarding in Action

Instruction Fetch

sub $1, $12, $3
and $12, $3, $4
add $3, $8, $11
Write Back
Forwarding in Action

Instruction Fetch

- Sub: $1, $12, $3
- AND: $12, $3, $4
- Add: $3, $8, $11

Write Back
Forwarding in Action

sub $1, $12, $3
and $12, $3, $4
add $3, $9, $11
Forwarding in Action

sub $1, $12, $3
and $12, $3, $4
add $3, $9, $11
Data Hazard: Load followed by Store

lw $2, 10($1)

st $2, 0x1000($5)

or $13, $6, $2

add $14, $2, $2

sw $15, 100($2)

This forwarding can be done but is there a forwarding path?
M↦M Forwarding for LW↦SW
(Exercise 6.20 in the Textbook)
Forwarding does not eliminate Data Hazard in all cases

- Consider this code:

  lw $2, 10($1)
  and $12, $2, $5
  or $13, $6, $2
  add $14, $2, $2
  sw $15, 100($2)
Data Hazard: Load followed by R-type

lw $2, 10($1)

and $12, $2, $5

or $13, $6, $2

add $14, $2, $2

sw $15, 100($2)
Eliminating Data Hazards via Forwarding and stalling

```
lw $2, 10($1) and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
```
Pipeline Interlocks

- Not all data hazards can be handled by forwarding

- Pipeline Interlock or Hazard Detection Unit
  - detects a hazard and stalls the pipeline until the hazard is clear

- A stall creates a pipeline bubble:
  - Preventing the IF and ID stages from proceeding
    - don’t write the PC (PCWrite = 0)
    - don’t rewrite IF/ID register (IF/IDWrite = 0)
  - Inserting “nops”
    - set all control signals propagating to EX/MEM/WB to zero (inserts a no-op instruction)
Hazard Detection Unit

- We can stall the pipeline by keeping an instruction in the same stage

if (ID/EX.MemRead and ((ID/EX.RegisterRt = IF/ID.RegisterRs) or (ID/EX.RegisterRt = IF/ID.RegisterRt)))
then stall the pipeline $\Rightarrow$ PCWrite = 0, IF/IDWrite = 0, EX/M/WB = 0
Hazard Detection Unit

and $4, $2, $5

lw $2, 20($1)
Hazard Detection Unit

and $4$, $2$, $5$

bubble

\textit{lw} $2$, 20($1$)

Pradam Argade

CSE 141, Fall 2005

13-29
Hazard Detection Unit

and $4, $2, $5  bubble  lw $2, 20($1)
Data Hazard Key Points

- Pipelining provides high throughput
- Data dependencies cause *data hazards*
- Data hazards can be solved by:
  - Software (nops)
  - Hardware data forwarding
  - Hardware pipeline stalling
- Our processor, and indeed all modern processors, use a combination of forwarding and stalling
Announcements

- **Reading Assignment**
  - Chapter 5. The Processor: Datapath and Control
    Sections 5.6
  - Chapter 6. Enhancing Performance with Pipelining
    Sections 6.1 - 6.10

- **Homework 5: Due Mon., November 14th in class**
  5.49, 5.50
  6.1, 6.2, 6.3, 6.4, 6.6, 6.14, 6.15, 6.17

- **Quiz**
  **When:** Mon., November 14th
  **Topic:** Pipelining and Hazards