Chapter Six- 1st Half
Pipelined Processor
Delayed Controls, Hazards

EE3055
Web: www.csc.gatech.edu/copeland/jac/3055-05

Pipelining

• Improve performance by increasing instruction throughput

<table>
<thead>
<tr>
<th>Program</th>
<th>Cycles</th>
<th>Cycles</th>
<th>Cycles</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
<td>4000</td>
<td>8000</td>
<td>12000</td>
</tr>
<tr>
<td></td>
<td>1800</td>
<td>3600</td>
<td>7200</td>
<td>10800</td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td>3200</td>
<td>6400</td>
<td>9600</td>
</tr>
<tr>
<td></td>
<td>1400</td>
<td>2800</td>
<td>5600</td>
<td>8400</td>
</tr>
</tbody>
</table>

Cycles can be different time durations.

Cycle always the longest time duration.

Ideal speedup is number of stages in the pipeline. Do we achieve this?

Pipelining

• What makes it easy
  – all instructions are the same length
  – just a few instruction formats
  – memory operands appear only in loads and stores

• What makes it hard?
  – structural hazards: suppose we had only one memory
  – control hazards: need to worry about branch instructions
  – data hazards: an instruction depends on a previous instruction

• We’ll build a simple pipeline and look at these issues

• We’ll talk about modern processors and what really makes it hard:
  – exception handling
  – trying to improve performance with out-of-order execution, etc.
Basic Idea

What do we need to add to actually split the datapath into stages?

Pipelined Datapath

Can you find a problem even if there are no dependencies?
What instructions can we execute to manifest the problem?

Corrected Datapath

[Diagram of Corrected Datapath]

Fig. 6.11
Graphically Representing Pipelines

- Can help with answering questions like:
  - how many cycles does it take to execute this code?
  - what is the ALU doing during cycle 4?
  - use this representation to help understand datapaths

Pipeline Control

- Output of one stage stored until next clock cycle, then delivered as input to the next stage.

Pipeline Control

- We have 5 stages. What needs to be controlled in each stage?
  - Instruction Fetch and PC increment
  - Instruction Decode / Register Fetch
  - Execution
  - Memory Stage
  - Write Back

- How would control be handled in an automobile plant?
  - a fancy control center telling everyone what to do?
  - should we use a finite state machine?
Pipeline Control

- Pass control signals along just like the data

<table>
<thead>
<tr>
<th>Instruction/Execution</th>
<th>Memory access stage</th>
<th>Write-back stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reg</td>
<td>Addr</td>
<td>Addr</td>
</tr>
<tr>
<td>In</td>
<td>Out</td>
<td>Out</td>
</tr>
</tbody>
</table>

Datapath with Control

Dependencies

- Problem with starting next instruction before first is finished
- Dependencies that "go backward in time" are data hazards

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Control Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>lw</td>
<td>000000001100</td>
</tr>
<tr>
<td>sw</td>
<td>000000101010</td>
</tr>
<tr>
<td>beq</td>
<td>00000100100</td>
</tr>
</tbody>
</table>
### Software Solution

- Have compiler guarantee no hazards
- Where do we insert the "nops"?

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

- Problem: this really slows us down!
- Hardware solutions - next set of slides.