EN164: Design of Computing Systems
Lecture 16: Processor / Pipeline Design 2

Professor Sherief Reda
http://scale.engin.brown.edu
Electrical Sciences and Computer Engineering
School of Engineering
Brown University
Spring 2011

[ material from Patterson & Hennessy, 4th ed and Harris 1st ed ]
Pipeline datapath summary

<table>
<thead>
<tr>
<th></th>
<th>IF/ID</th>
<th>ID/EX</th>
<th>EX/MEM</th>
<th>MEM/WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sub</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>and</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- How many cycles it takes to finish code?
Reminder of single-cycle control
Modifications to pipeline control

- Control signals derived from instruction
  - Same as in single-cycle implementation
  - Control delayed to proper pipeline stage
Pipelined datapath + control
Cycle 1

lw $10, 20($1)
sub $11, $2, $3
and $12, $4, $5
or $13, $6, $7
add $14, $8, $9
Cycle 2

lw $10, 20($1)
sub $11, $2, $3
and $12, $4, $5
or $13, $6, $7
add $14, $8, $9
Cycle 3

lw $10, 20($1)
sub $11, $2, $3
and $12, $4, $5
or $13, $6, $7
add $14, $8, $9
Cycle 4

IF: or $13, $6, $7
ID: and $12, $4, $5
EX: sub $11, ...
MEM: lw $10, ...
WB: before<1>

lw $10, 20($1)
sub $11, $2, $3
and $12, $4, $5
or $13, $6, $7
add $14, $8, $9

S. Reda EN164 Sp '11
Cycle 5

lw $10, 20($1)
sub $11, $2, $3
and $12, $4, $5
or $13, $6, $7
add $14, $8, $9
Cycle 6
Cycle 7

lw   $10, 20($1)
sub  $11, $2, $3
and  $12, $4, $5
or   $13, $6, $7
add  $14, $8, $9
Cycle 8

lw $10, 20($1)
sub $11, $2, $3
and $12, $4, $5
or $13, $6, $7
add $14, $8, $9
Cycle 9

```
lw    $10, 20($1)
sub   $11, $2, $3
and   $12, $4, $5
or    $13, $6, $7
add   $14, $8, $9
```
Pipelining hazards

- Situations that prevent starting the next instruction in the next cycle
  1. **Structural hazards**
     - A required resource is busy
  2. **Data hazard**
     - Need to wait for previous instruction to complete its data read/write
  3. **Control hazard**
     - Deciding on control action depends on previous instruction
1. Structural hazards

- Conflict for use of a resource
- In MIPS pipeline with a single memory
  - Load/store requires data access
  - Instruction fetch would have to *stall* for that cycle
    - Would cause a pipeline “bubble”
- Hence, pipelined datapaths require separate instruction/data memories
  - Or separate instruction/data caches
2. Data Hazards: compute-use

add $s0, $s2, $s3
and $t0, $s0, $s1
or $t1, $s4, $s0
sub $t2, $s0, $s5