Program execution order (in instructions):

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

Value of register $z2:

<table>
<thead>
<tr>
<th>Time (in clock cycles)</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>s</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>
Implementation of the pipelined datapath

- What do we need to add to actually split the datapath into stages?
Can you find a problem even if there are no dependencies?
What instructions can we execute to manifest the problem?
Lw: Instruction fetch and Instruction decode
Lw: Execution
Lw: Memory and Write back
Sw: Memory and Write back

©1998 Morgan Kaufmann Publishers
Corrected Datapath
Graphically Representing Pipelines

- Multiple-clock-cycle diagram vs. Single-clock-cycle diagram
Pipeline Control
Pipeline control

- We have 5 stages. What needs to be controlled in each stage?
  - Instruction Fetch and PC Increment: the control signals are always asserted (this stage is executed at every clock cycle)
  - Instruction Decode / Register Fetch: the control signals are always asserted (this stage is executed at every clock cycle)
  - Execution: set RegDst, ALUOp, ALUSrc (select the result register, the ALU operation, and either Read data 2 or a sign-extended immediate for the ALU).
  - Memory Stage: Branch, MemRead, MemWrite (set by beq, lw, sw).
  - Write Back: MemtoReg, RegWrite (send the ALU result to the memory or to the register file, write the value).
Pipeline Control

- Pass control signals along just like the data

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Execution/Address Calculation stage control lines</th>
<th>Memory access stage control lines</th>
<th>stage control lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reg Dst</td>
<td>ALU Op1</td>
<td>ALU Op0</td>
</tr>
<tr>
<td>R-format</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>lw</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sw</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>beq</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Example

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