Pipelining: the laundry analogy
Pipelining

- Improve performance by increasing instruction throughput

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.
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?
Corrected Datapath
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

• 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

### Pipeline Control Table

<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>

### Diagram

- Control
- IF/ID
- ID/EX
- EX/MEM
- MEM/WB
- WB

- Instruction flow through stages.