Overview of processor operation

Fetch the next instruction from Mem[PC]

Read from registers

Perform ALU operation

Access memory (load or store)

Load result into register

Update the PC
A more complex architecture
Pipelining laundry

Speed up for 4 loads: \[
\frac{8}{3.5} = 2.3
\]
Basic operations

Fetch the next instruction from Mem[PC]

Read from registers

Perform ALU operation

Access memory (load or store)

Load result into register
Pipelined execution

Program execution order (in instructions)

\begin{align*}
\text{lw} & \; \$1, \; 100(\$0) \\
\text{lw} & \; \$2, \; 200(\$0) \\
\text{lw} & \; \$3, \; 300(\$0)
\end{align*}

Latency = 800ps

Throughput = \frac{1 \text{ instruction}}{800\text{ps}} \times \frac{1000\text{ps}}{1\text{ns}} = 1.25\text{GIPS}
Pipelined execution

Latency = 1000ps

\[
\text{Throughput} = \frac{1 \text{ instruction}}{200\text{ps}} \times \frac{1000\text{ps}}{1\text{ns}} = 5\text{GIPS}
\]

Increase = \frac{5}{1.25} = 4
A simplified pipeline
Pipelined instructions

Instruction0

Instruction1

Instruction2

Instruction3

Instruction4

time to fill pipeline
Structural hazard
What about the registers?

- `add $1, $5, $6`

- **Instruction1**

- **Instruction2**

- `add $2, $1, $3`

- **Instruction4**
Data hazards

add $1, $5, $6

Forwarding

add $1, $5, $6

add $2, $1, $6
Data Hazards

lw $1, 12($2)

add $2, $1, $6

lw $1, 12($2)

add $2, $1, $6
Control Hazards

beq

lw
Datapath revisited
Datapath revisited
Pipelined datapath
Superscalar

**Instruction Control**

- **Fetch Control**
- **Instruction Decode**
- **Instruction Cache**
- **Retirement Unit**
- **Register File**

**Execution**

- **Integer/Branch**
- **General Integer**
- **FP Add**
- **FP Mult/Div**
- **Load**
- **Store**

**Functional Units**

- **Data Cache**

**Operational Results**

- Address
- Data
- Addr.

**Register Updates**

- Prediction OK?
Nehalem CPU (i7)

- Multiple instructions can execute in parallel
  1 load, with address computation
  1 store, with address computation
  2 simple integer (one may be branch)
  1 complex integer (multiply/divide)
  1 FP Multiply
  1 FP Add

- Some instructions take > 1 cycle, but can be pipelined

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Latency</th>
<th>Cycles/Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load / Store</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Integer Multiply</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td><strong>Integer/Long Divide</strong></td>
<td>11--21</td>
<td>11--21</td>
</tr>
<tr>
<td>Single/Double FP Multiply</td>
<td>4/5</td>
<td>1</td>
</tr>
<tr>
<td>Single/Double FP Add</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td><strong>Single/Double FP Divide</strong></td>
<td>10--23</td>
<td>10--23</td>
</tr>
</tbody>
</table>