Datapath: addu

Diagram showing the datapath for an addu instruction.
Datapath: addiu
Datapath: \( \text{l} \)w
Datapath: \( sw \)
Datapath: beq

Diagram showing the datapath for the beq instruction.
Datapath: j
How fast can we run the clock?
Multi-Cycle Datapath

Observation: we can run the clock faster by partitioning the execution into multiple *stages.*
Stages: instruction fetch, register fetch/decode, execute, memory, writeback.

Idea: break down single instruction execution into smaller parts, some instructions skip parts of execution.
Multi-Cycle Datapath

- PC needs a write-enable signal
- Design FSM for control

(textbook has details: sections 5.4, 5.5)
Performance

Suppose we had:

- **IF:** 5\text{ns}, **RD:** 3\text{ns}, **ALU:** 6\text{ns}, **MEM:** 5\text{ns}, **WB:** 4\text{ns}

**Single-cycle datapath cycle time:** 23\text{ns}

**Multi-cycle datapath:**

- **lw:** 30\text{ns}
- **add:** 24\text{ns}
- **beq:** 18\text{ns}

Is this better? Depends on the instruction mix!
(should have balanced pipeline stages)
Pipelining

Why not start executing the next instruction as soon as the first one has been fetched?

Hardware resource conflict...
Pipelining

Simple fix: make all instructions take the same number of cycles.

Each instruction takes 30ns, but we complete an instruction every 6ns (caveat: flop/latch overhead).
Introduce positive-edge triggered flip-flops between pipeline stages to hold intermediate values.
Pipelined Datapath: Control

Basic idea: copy the instruction, and generate local control in each pipeline stage.
## Pipelined Datapath: Timing

<table>
<thead>
<tr>
<th>Cycle</th>
<th>IF</th>
<th>RD</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>ins1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i+1</td>
<td>ins2</td>
<td>ins1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i+2</td>
<td>ins3</td>
<td>ins2</td>
<td>ins1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i+3</td>
<td>ins4</td>
<td>ins3</td>
<td>ins2</td>
<td>ins1</td>
<td></td>
</tr>
<tr>
<td>i+4</td>
<td>ins5</td>
<td>ins4</td>
<td>ins3</td>
<td>ins2</td>
<td>ins1</td>
</tr>
</tbody>
</table>

Instructions flow through the datapath, advancing every cycle.
Pipelined Datapath: Timing

- Flops between stages: positive-edge triggered
- PC, register file: negative-edge triggered
Single Instruction Execution
Single Instruction Execution
Single Instruction Execution

IF WBMEMEXRD
1
1
0
0
IR3IR2 IR4IR1
register
file
CLK
alu
X
CLK PC
CLK
Single Instruction Execution

Register file output can change!