Datapath

- Abstract / Simplified View:

Two types of functional units:
- Combinational logic
- State elements: D-latches and D flip-flops
- Clocking methodology: edge triggered
Building the Datapath
Control

- Selecting the operations to perform (ALU, read/write, etc.)
- Controlling the flow of data (multiplexor inputs)
- Information comes from the 32 bits of the instruction
- Example:

  add $8, $17, $18   Instruction Format:

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>10001</td>
<td>10010</td>
<td>01000</td>
<td>00000</td>
<td>10000</td>
</tr>
</tbody>
</table>

- ALU's operation based on instruction type and function code
Control

- Simple combinational logic (truth tables)
Our Simple Control Structure

- All of the logic is combinational
- We wait for everything to settle down, and the right thing to be done
  - ALU might not produce “right answer” right away
  - we use write signals along with clock to determine when to write
- Cycle time determined by length of the longest path

We are ignoring some details like setup and hold times
Pipelining

- Improve performance by increasing instruction throughput

Ideal speedup is number of stages in the pipeline. Do we achieve this?
Implementation of the pipelined datapath

IF: Instruction fetch
ID: Instruction decode/register file read
EX: Execute/address calculation
MEM: Memory access
WB: Write back

- Instruction memory
- Address
- Instruction file
- ALU result
- Zero
- Shift left 2
- Mux
- Add
- Write data
- Read registers
- Read data
- Mux
- Data memory
- Write register
- Write data
Datapath

Diagram showing the datapath and processes involved in a typical CPU pipeline with stages such as IR/ID, ID/EX, EX/MEM, and MEM/WB.
Datapath with Control

Pass control signals along just like the data
Dependencies and forwarding

<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>Value of register $2$:</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10/−20</td>
<td>−20</td>
<td>−20</td>
<td>−20</td>
<td>−20</td>
</tr>
<tr>
<td>Value of MEM/WB:</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>−20</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Program execution order (in instructions):
- sub $2, $1, $3
- and $12, $2, $5
- or $13, $6, $2
- add $14, $2, $2
- sw $15, 100($2)
Branch Hazards

- When we decide to branch, other instructions are in the pipeline!
Pipeline with hazard detection and forwarding