Fetch Unit Operation

CSE 141L Lab 2
Front-End vs Back-End

- Front-end *fetches* instructions (Lab 2)
  - Reads from instruction memory
  - Stores fetched instructions in FIFO
  - Predicts which addresses to fetch
- Back-end *executes* instructions (Lab 3)
  - Pulls instructions from front-end’s FIFO
  - Reads/writes to data memory
  - Must check correctness of front-end prediction.

*Why separate fetch from execution?*
Pipelined Fetch Unit

• Fetch unit split into two separate stages
  • Address Calculation
    • Determines what address goes into SRAM
  • FIFO storage
    • Stores instruction address and SRAM output into FIFO
Fetch Pipeline In Action

Cycle # | 0 | 1 | 2 | 3 | 4 | 5
--- | --- | --- | --- | --- | --- | ---
0x0: add $1, $2, $3 | AC | FS |
0x1: sub $3, $3, $4 | AC | FS |
0x2: bez $3, 0x10 | AC | FS |
0x12: add $7, $5, $9 | AC | FS |
0x13: ... | AC | ... |

Prediction known here
Restarting Fetch

<table>
<thead>
<tr>
<th>Cycle #</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2: bez $3, 0x10</td>
<td>AC</td>
<td>FS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x12: add $7, $5, $9</td>
<td>AC</td>
<td>FS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x13: mul $7, $7, $4</td>
<td>AC</td>
<td>FS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3: add $3, $6, $7</td>
<td>AC</td>
<td>FS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4: ...</td>
<td>AC</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

back-end detects mispredict! restart @ 0x3

Clear FIFO. Why?
Stalling for loads/stores

Cycle # | 0 | 1 | 2 | 3 | 4 | 5
--- | --- | --- | --- | --- | --- | ---
0x5: add $1, $2, $3 | AC | FS | | | | |
0x6: sub $3, $3, $4 | AC | FS | | | | |
0x7: muli $3, 0x8 | | | Load | AC | FS | |
0x8: add $7, $5, $9 | | | | AC | FS | |
Stalled on Full FIFO

Cycle #

<table>
<thead>
<tr>
<th>Cycle</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x5:</td>
<td>AC</td>
<td>FS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>add</td>
<td>$1,</td>
<td>$2,</td>
<td>$3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x6:</td>
<td>AC</td>
<td>FS</td>
<td>FS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sub</td>
<td>$3,</td>
<td>$3,</td>
<td>$4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7:</td>
<td>AC</td>
<td>AC</td>
<td>FS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>muli</td>
<td>$3,</td>
<td>0x8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8:</td>
<td>AC</td>
<td>FS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add</td>
<td>$7,</td>
<td>$5,</td>
<td>$9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIFO Full asserted

FIFO Full deasserted
Lab 2B: Fetch Unit Control
Control logic implementation

- Fetch unit is pipelined
- Not modeled as a finite state machine
- Each pipeline stage depends on a subset of the control signals
- Need to identify which signals those are
Simplified Implementation Strategy

- For each pipeline stage, ask yourself the following questions:
  - What control signals are needed for the datapath during this stage?
  - What does each of these signals do?
  - What inputs do I need to correctly set those control signals?
- Encode relationship between control input and output using Verilog
A Simple Example

• Address Calculation stage:
  • What are the control signals associated with that stage?
    • `sel_mux[3:0]`, ...
  • What does `sel_mux[0]` do?
    • Selects between PC+1 and PC+offset
  • What is value of `sel_mux[0]` based on?
    • “P” bit: dictates whether we branch or not
  • Anything else?