Single Cycle Control

• Very simple
  – Control signals are functions of opcode and possibly function fields
  – Combinational logic suffices

• Ex: RegWrite
  – Asserted on $R_{-type}$, lw
  – Deasserted on beq, sw, j
Multi-Cycle Control

- Much harder
  - Control signals depend on instruction and cycle
- Consider RegWrite

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Fetch</th>
<th>Decode</th>
<th>Execute</th>
<th>Memory Access</th>
<th>Write Back</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-Type</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>sw</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>lw</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- CPU must “remember” what cycle it is in
  - Control unit must maintain state
  - Several ways to do this
Review
Multi-Cycle Control Implementation

- Two main control implementations
  - State machine
    - Translate finite state machine diagrams to hardware
    - Control signals function of current state
  - Microprogram
    - A small control program runs in parallel to CPU datapath
    - Program outputs are control signals

- Logically similar in many respects
  - Control “remembers” state and changes signals
  - Implementation very different
  - Combinations also possible
State Machine Control

- From digital design:
  - Create state machine
  - Assign state values
  - Derive control signal functions
  - Derive next state functions
Microprogram Control

- Control Unit is now an indexed ROM
  - Memory bits set control signals and next state
  - Microprogram state and instruction select the memory value

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Microprogram State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110011010</td>
<td>10010110011011</td>
</tr>
</tbody>
</table>

Instruction | RegWrite

- Microprogram State

Diagram:
- Control Unit
- Inputs
- Outputs
- Control Signals
- Next State
- Microprogram State
- Instruction
- RegWrite
Single Microprogram ROM

• Inputs
  – 6 bits from instruction opcode
  – 4 bits from current state

• Outputs
  – 16 bits for control signals
  – 4 bits for next state

• ROM Size
  – $2^{10}$ 20-bit words
  – Total size: 20 kbits