Control Signals: Write (Revisit)

What happen before write value to $1$ is ready?
- => need some time to be stabilized
- => this will be the longest path (critical path)
- => determines the clock cycle for the CPU
Control Signals: Multiplexor Selector (Revisit)

Control Signals: One More Multiplexor Selector (Revisit)
Control Signals:
ALU Control (Revisit)

- What should the ALU do with this instruction?
  - Information comes from the 32 bits of the instruction
  - ALU's operation based on instruction type and function code
- Multi-level control (for simplifying control logic)
  - Instruction’s opcode (bit31-bit26)
    => ALUOp1 & ALUOp0
      (with instruction’s funct (bit4-bit0))
    => ALU inputs: binvert (= carryin), operation

Simple Implementation:
Control Signals (Revisit)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>RegDst</th>
<th>ALUSrc</th>
<th>MemtoReg</th>
<th>MemRead</th>
<th>MemWrite</th>
<th>PCSrc</th>
<th>Branch</th>
<th>ALUOp1</th>
<th>ALUOp0</th>
</tr>
</thead>
<tbody>
<tr>
<td>lw</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sw</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>beq</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Simple Implementation: Generation of Control Signals (Revisit)

- 32-bit instruction
  - 6-bit opcode
  - 6-bit func

Control Logic

ALU

ALU control

Outputs

e.g. bit0 = A1(F3+F0)

Simple Implementation: Realization of Control Signals (Revisit)

- Simple combinational logic (truth tables)
Simple Implementation: Datapath and Control (Revisit)

Control Signals of Pipelined MIPS (Same as Simple Implementation)

- **At IF & ID stages**
- **At EX stage**
  - RegDst: 0 for lw, 1 for R-type
  - ALUSrc: 0 for R-type/beq (Reg.), 1 for lw/sw (inst.)
  - ALUOp1/0: 10 for R-type, 00 for lw/sw (add), 01 for beq (sub)
- **At MEM stage**
  - Branch: Activated if the instruction is “branch”
  - MemWrite / MemRead: 1 for sw and lw, respectively
  - PCSrc: PCSrc = Branch && Zero flag (derived, not generated)
- **At WB stage**
  - RegWrite: Activated if the instruction is R-type or lw
  - MemtoReg: 0 for R-type (ALU), 1 for lw (memory)
Pipeline Control

Pipeline Control: Problem!
Pipeline Control

- What’s the difference in pipelined MIPS?
  - add  IF  ID  EX  MEM  WB
  - lw  IF  ID  EX  MEM  WB

- Control signals should also be conflict-free
- Should we use a finite state machine?
- Our approach is
  - Generate control signals all at once at ID stage (why not IF?)
  - And passed them through stages

Pipeline Control

- Pass control signals along just like the data

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Execution/Address Calculation stage control lines</th>
<th>Memory access stage control lines</th>
<th>stage control lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reg</td>
<td>ALU</td>
<td>ALU</td>
</tr>
<tr>
<td></td>
<td>Dst</td>
<td>Op1</td>
<td>Op0</td>
</tr>
<tr>
<td>op1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>op2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>op3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Cleveland State University 13 c.yu91@csuohio.edu

Cleveland State University 14 c.yu91@csuohio.edu
Example

- Five instructions go through the MIPS pipeline:

  - lw  $10, 20($1)
  - sub $11, $2, $3
  - and $12, $4, $5
  - or  $13, $6, $7
  - add $14, $8, $9