Lecture 8-part 2

- Today:
  - Single-cycle implementation
  - Maybe start discussing performance

It's Friday!

Branching hardware

The final datapath

What is a cycle?

Control

- The control unit is responsible for setting all control signals so that each instruction is executed properly.
  - The control unit’s input is the 32-bit instruction word.
  - The outputs are values for the blue control signals in the datapath.
  - Most of the signals can be generated from the instruction opcode alone, and not the entire 32-bit word.
  - To illustrate the relevant control signals, we will show the route that is taken through the datapath by R-type, I-type, and B-type instructions.
R-type instruction path
- The R-type instructions include add, sub, and or, and slt.
- The ALUOp is determined by the instruction’s `func` field.

Lw instruction path
- An example load instruction is lw $t0, 4($sp).
- The ALUOp must be 010 (add), to compute the effective address.

Sw instruction path
- An example store instruction is sw $s0, 4($sp).
- The ALUOp must be 010 (add), to compute the effective address.

Beq instruction path
- One example branch instruction is beq $t0, $t0, $0.
- The ALUOp is 110 (subtract), to test for equality.

Control signal table

<table>
<thead>
<tr>
<th>Operation</th>
<th>RegSel</th>
<th>RegLitte</th>
<th>ALUOp</th>
<th>HamWrite</th>
<th>HamRead</th>
<th>HamSel</th>
<th>RegSel</th>
<th>RegLitte</th>
<th>ALUOp</th>
<th>HamWrite</th>
<th>HamRead</th>
<th>HamSel</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sub</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>or</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>slt</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>j</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- sw and beq are the only instructions that do not write any registers.
- lw and sw are the only instructions that use the constant field. They also depend on the ALU to compute the effective memory address.
- ALUOp for R-type instructions depends on the instructions’ func field.
- The PCInc control signal (not listed) should be set if the instruction is beq and the ALU’s Zero output is true.

Generating control signals
- The control unit needs 13 bits of inputs.
  - Six bits make up the instruction’s opcode.
  - Six bits come from the instruction’s func field.
  - It also needs the Zero output of the ALU.
- The control unit generates 10 bits of output, corresponding to the signals mentioned on the previous page.
  - You can build the actual circuit by using k-maps, Boolean algebra, or big circuit design programs.
  - The textbook presents a slightly different control unit.
Summary

- A datapath contains all the functional units and connections necessary to implement an instruction set architecture.
  - For our single-cycle implementation, we use two memories, an ALU, some extra adders, and lots of multiplexers.
  - MIPS is a 32-bit machine, so most of the buses are 32-bits wide.
- The control unit tells the datapath what to do, based on the instruction that’s currently being executed.
  - Our processor has ten control signals that regulate the datapath.
  - The control signals can be generated by a combinational circuit with the instruction’s 32-bit binary encoding as input.
- Next, we’ll see the performance limitations of this single-cycle machine and try to improve upon it.