Review: A Single Cycle Datapath
- Rs, Rt, Rd, Imm16 connected to datapath
- We have everything except control signals

Recap: Meaning of the Control Signals
- nPC_MUX_sel: 0 ⇒ PC ← PC + 4
- 1 ⇒ PC ← PC + 4 + {SignExt(Im16), 00}
- Later in lecture: higher-level connection between mux and branch cond

An Abstract View of the Critical Path
- Critical Path (Load Operation) =
  Delay clock through PC (FFs) +
  Instruction Memory’s Access Time +
  Register File’s Access Time, +
  ALU to Perform a 32-bit Add +
  Data Memory Access Time +
  Stable Time for Register File Write

Recap: Meaning of the Control Signals
- ExtOp: “zero”, “sign”
  MemWr: 1 ⇒ write memory
- ALUSrc: 0 ⇒ regB; 1 ⇒ imm
  RegDst: 0 ⇒ “rt”; 1 ⇒ “rd”
- ALUctr: “add”, “sub”, “or”
  RegWr: 1 ⇒ write register
### RTL: The Add Instruction

**add rd, rs, rt**

- `MEM[PC]` Fetch the instruction from memory
- `PC = PC + 4` Calculate the next instruction's address

### Instruction Fetch Unit at the Beginning of Add

- Fetch the instruction from Instruction memory: `Instruction = MEM[PC]`
- same for all instructions

### The Single Cycle Datapath during Add

- `R[rd] = R[rs] + R[rt]`

### Instruction Fetch Unit at the End of Add

- `PC = PC + 4`
  - This is the same for all instructions except: Branch and Jump

### Single Cycle Datapath during Or Immediate?

- `R[rt] = R[rs] OR ZeroExt[Imm16]`
The Single Cycle Datapath during Load?

- \[ R[rt] = \text{Data Memory} \left( R[rs] + \text{SignExt}[imm16] \right) \]

\[ \text{RegDst} = \begin{cases} \text{Zero} = 1 & \text{if } (R[rs] - R[rt] == 0) \\ \text{Zero} = 0 & \text{otherwise} \end{cases} \]

The Single Cycle Datapath during Store?

- \[ \text{Data Memory} \left( R[rs] + \text{SignExt}[imm16] \right) = R[rt] \]

The Single Cycle Datapath during Branch?

- if \( (R[rs] - R[rt] == 0) \) then Zero = 1; else Zero = 0

The Single Cycle Datapath during Branch?

- if \( (R[rs] - R[rt] == 0) \) then Zero = 1; else Zero = 0
Instruction Fetch Unit at the End of Branch

- If \( (\text{Zero} == 1) \) then \( \text{PC} = \text{PC} + 4 + \text{SignExt[imm16]} \times 4 \); else \( \text{PC} = \text{PC} + 4 \)

- What is encoding of \( nPC\_sel \)?
- Direct MUX select?
- Branch / not branch
- Let’s pick 2nd option

A Summary of the Control Signals (1/2)

<table>
<thead>
<tr>
<th>Inst</th>
<th>Register Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>( R[d] \leftarrow R[r] + R[t] ); ( \text{PC} \leftarrow \text{PC} + 4 )</td>
</tr>
<tr>
<td>SUB</td>
<td>( R[d] \leftarrow R[r] - R[t] ); ( \text{PC} \leftarrow \text{PC} + 4 )</td>
</tr>
<tr>
<td>ORI</td>
<td>( R[r] \leftarrow \text{sign-ext[imm16]} ); ( \text{PC} \leftarrow \text{PC} + 4 )</td>
</tr>
<tr>
<td>LOAD</td>
<td>( R[r] \leftarrow \text{MEM}[R[r] + \text{sign-ext[imm16]}]; ( \text{PC} \leftarrow \text{PC} + 4 )</td>
</tr>
<tr>
<td>STORE</td>
<td>( \text{MEM}[R[r] + \text{sign-ext[imm16]}] \leftarrow R[r]; ( \text{PC} \leftarrow \text{PC} + 4 )</td>
</tr>
<tr>
<td>BEQ</td>
<td>( \text{if} (R[r] == R[t]) \text{then} \text{PC} \leftarrow \text{PC} + \text{sign-ext[imm16]} ); ( 00 ) else ( \text{PC} \leftarrow \text{PC} + 4 )</td>
</tr>
</tbody>
</table>

A Summary of the Control Signals (2/2)

<table>
<thead>
<tr>
<th>func</th>
<th>op</th>
<th>add</th>
<th>sub</th>
<th>reg</th>
<th>imm</th>
<th>J-type jump</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>op</td>
<td>00 00 00 00 00 00 1101 10 0011 10 1011 00 0100 00 0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Step 4: Given Datapath: RTL -> Control

The Single Cycle Datapath during Jump

- New PC = \( \{ \text{PC}[31..28], \text{target address}, 00 \} \)

Administrivia

- Project 2 Due Sunday
- HW 6 out tomorrow
- Slight shakeup of the schedule starting tomorrow (we’re only doing one Control lecture)
  - This allows us to have the second midterm before the drop deadline, if that would be preferable
The Single Cycle Datapath during Jump

- New PC = \{ PC\[31..28\], target address, 00 \}

Instruction Fetch Unit at the End of Jump

- New PC = \{ PC\[31..28\], target address, 00 \}

Instruction Fetch Unit at the End of Jump

Query:
- Can Zero still get asserted?
- Does nPC_sel need to be 0? If not, what?

Build CL to implement Jump on paper now

And in Conclusion... Single cycle control

5 steps to design a processor
1. Analyze instruction set => datapath requirements
2. Select set of datapath components & establish clock methodology
3. Assemble datapath meeting the requirements
4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
5. Assemble the control logic

Control is the hard part

MIPS makes that easier
- Instructions same size
- Source registers always in same place
- Immediates same size, location
- Operations always on registers/immediates

Peer Instruction

A. MemToReg='x' & ALUctr='sub'. SUB or BEQ?
B. ALUctr='add'. Which 1 signal is different for all 3 of: ADD, LW, & SW? RegDst or ExtOp?
C. “Don’t Care” signals are useful because we can simplify our Boolean equations?