MICROSOFT BETS ITS FUTURE ON A REPROGRAMMABLE COMPUTER CHIP
CS 2630
Computer Organization

Meeting 19: Building a MIPS processor
Brandon Myers
University of Iowa
implementing the addu rd,rs,rt instruction

**data to write**
these muxes select RD dataIn when that register is chosen, and take the existing value otherwise

**data to read**
these muxes select the register to take data from

**RD dataIn**

RS dataOut

RT dataOut

select register by number
the registers, where the data is stored

RS+RT

feed the sum back to the registers
Implementing the addu instruction

```
addu rd, rs, rt
```

Diagram:
- `rd`:
  - Read `reg1` → `Read data1`
  - Read `reg2` → `Read data2`
  - Write `reg` → `Write data`

- `rs`, `rt`:
  - Read `reg1` → `Read data1`
  - Read `reg2` → `Read data2`

- `cin`:
  - `cin + sum` → `cout`
Implementing the addu instruction

How do we program the “addu machine”?
Peer instruction

Give the sequence of addu machine inputs to perform
$t0 = t1 + t2 + t3$

a) 1st clock cycle: rd=8, rs=9, rt=10
   2nd clock cycle: rd=8, rs=8, rt=11

b) 1st clock cycle: rd=0, rs=9, rt=10
   2nd clock cycle: rd=8, rs=0, rt=0
   3rd clock cycle: rd=0, rs=8, rt=11
   4th clock cycle: rd=8, rs=0, rt=0

c) 1st clock cycle: rd=0, rs=9, rt=10
   2nd clock cycle: rd=8, rs=0, rt=0
   3rd clock cycle: rd=0, rs=8, rt=11
   4th clock cycle: rd=8, rs=0, rt=0

d) 1st clock cycle: rd=9, rs=10, rt=11
   2nd clock cycle: rd=8, rs=0, rt=0
Administrivia

• Reminder: HW4 due Oct 27 (Thursday)
• Project 2 out later this week; you will complete it in phases
  • brief break now for project groups
But, where do those inputs come from?

- the instruction memory

Recall the layout of bits in R-type instructions:

<table>
<thead>
<tr>
<th>R</th>
<th>opcode</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>31</td>
<td>26</td>
<td>25</td>
<td>21</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>31</td>
<td>15</td>
<td>11</td>
<td>10</td>
<td>6</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
How do we know which instruction we are on?
How do we know which instruction we are on?

- Store the current address in a 32-bit register called the *program counter* (PC)
- Add 4 each cycle to go to the next word (next instruction)
The complete addu machine
The complete addu machine

But, how do we get data into the addu machine? All registers start with the value 0.

Let’s modify the circuit to include addiu
Peer instruction

• Modify the processor so it also knows how to execute addiu

Assume you have a component that takes a MIPS opcode as input and provides a 1-bit signal \textit{isAddiu} as output.

\begin{align*}
\text{isAddiu} &= 0 \quad \text{if the opcode is 0x0 (the opcode for addu)} \\
\text{isAddiu} &= 1 \quad \text{if the opcode is 0x9 (the opcode for addiu)}
\end{align*}
Next steps

• Add more instructions to our processor:
  • other R and I types (or, ori, subu)
  • load and store (lw, sw)
  • branches (beq/bne)
  • jumps (j, jr, jal)

• How do we implement the control logic?