Our ultimate goal: building the MIPS datapath
Arithmetic Logic Unit (ALU)

ALU operation:
- 000 = and
- 001 = or
- 010 = add
- 110 = subtract
- 111 = slt
What about subtraction \((a - b)\) ?

• Two's complement approach: just negate b and add.
• How do we negate?

• The solution:
Tailoring the ALU to the MIPS datapath

- Need to support the set-on-less-than instruction
  \[ \text{slt } rd, rs, rt \]
  - slt is an arithmetic instruction
  - produces a 1 if \( rs < rt \) and 0 otherwise
  - use subtraction: \( (a-b) < 0 \) implies \( a < b \)

- Need to support test for equality (beq $t5, $t6, label)
  - use subtraction: \( (a-b) = 0 \) implies \( a = b \) \( \Rightarrow \) Zero=1
Supporting slt

(a) CarryIn
Less
CarryOut
ALU0
Result0

(a1)
(b1) 0

Result1

(a2)
(b2) 0

Result2

(a31)
(b31) 0

Result31

Set
Overflow

Set detection
Overflow
Test for equality and complete ALU

ALU operation:

000 = and
001 = or
010 = add
110 = subtract
111 = slt

Control lines

<table>
<thead>
<tr>
<th>Bnegate</th>
<th>Operation</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>and</td>
</tr>
<tr>
<td>0</td>
<td>01</td>
<td>or</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>add</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>sub</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>slt</td>
</tr>
</tbody>
</table>

• Note: zero is a 1 when the result is zero!