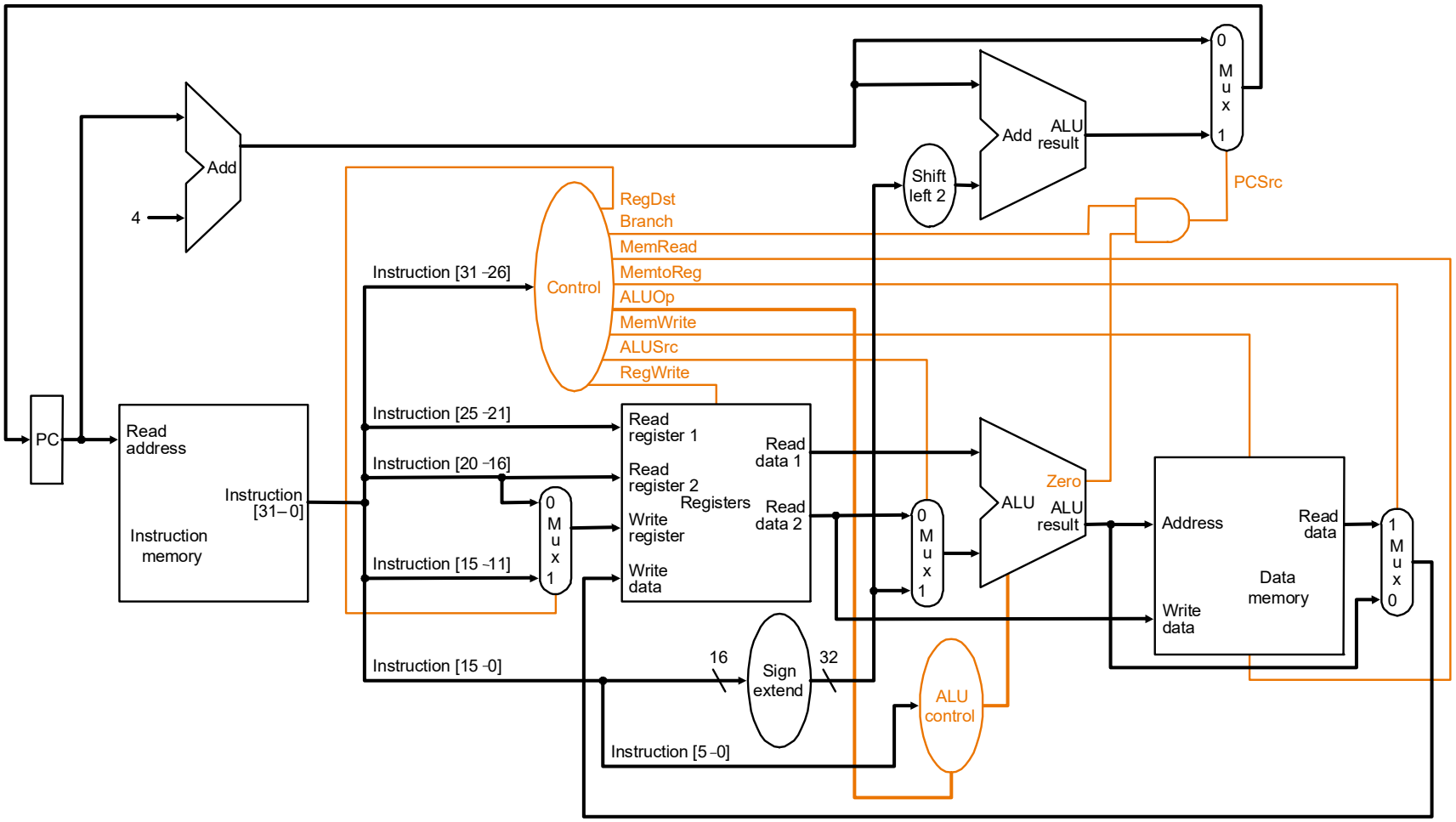
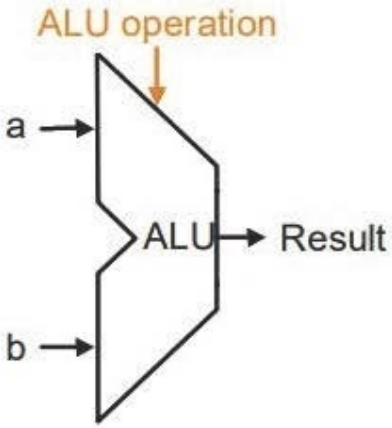
Our ultimate goal: building the MIPS datapath

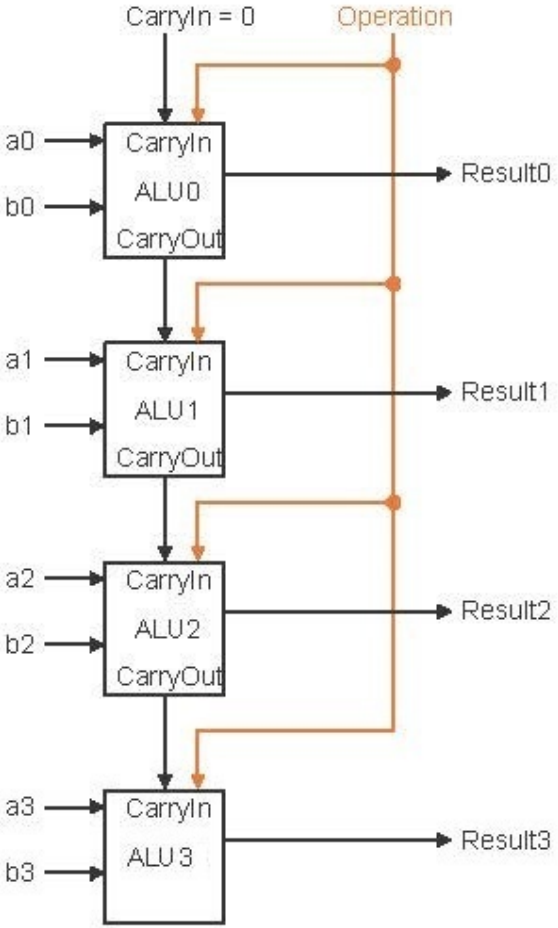


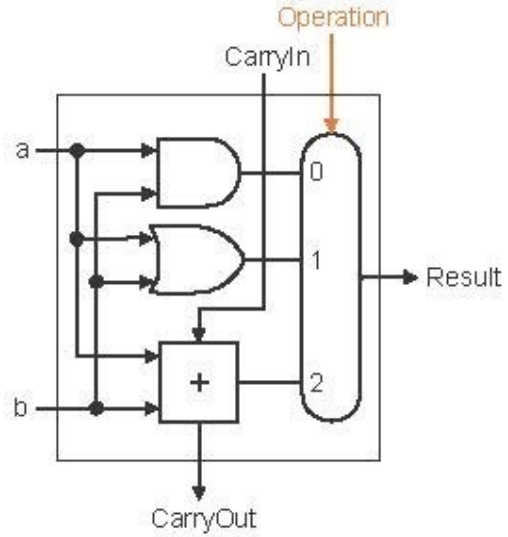
Arithmetic Logic Unit (ALU)



ALU operation (2-bit):

```
00 = and
01 = or
10 = add
```

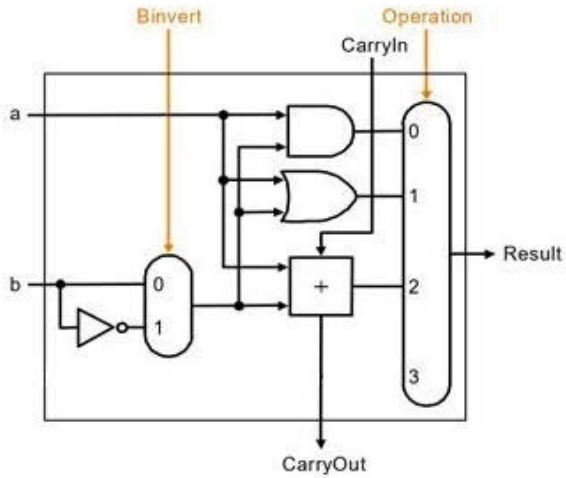




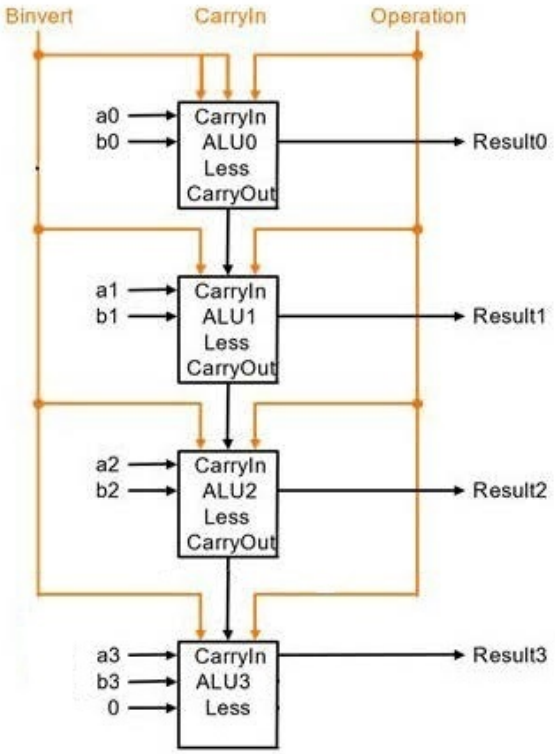
What about subtraction (a – b) ?

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

Invert all bits of b



Add 1



ALU operation (3-bit):

Binvert	Operation	
0	00	= and
0	01	= or
0	10	= add
1	10	= sub

Tailoring the ALU to the MIPS datapath

- Need to support the set-on-less-than instruction

  `slt rd, rs, rt`

  – slt is an arithmetic instruction

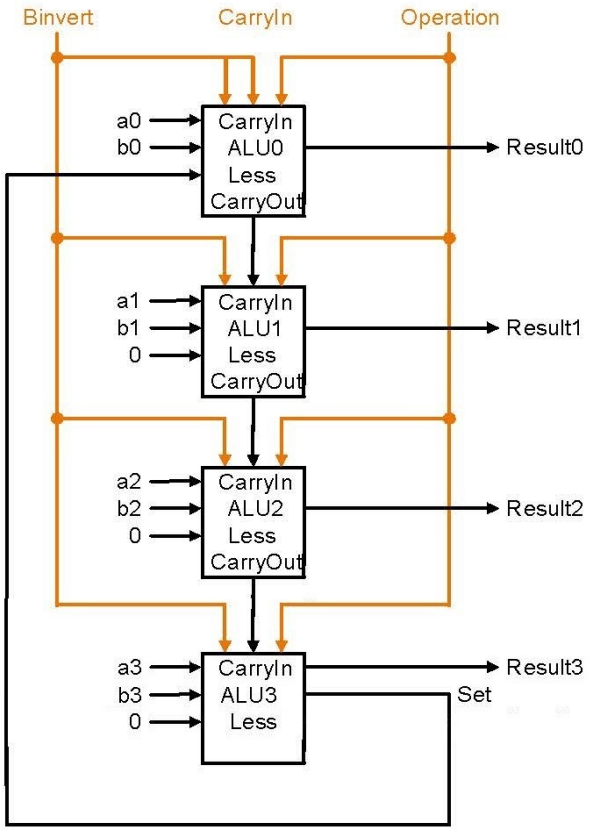
  – produces a 1 if rs < rt and 0 otherwise

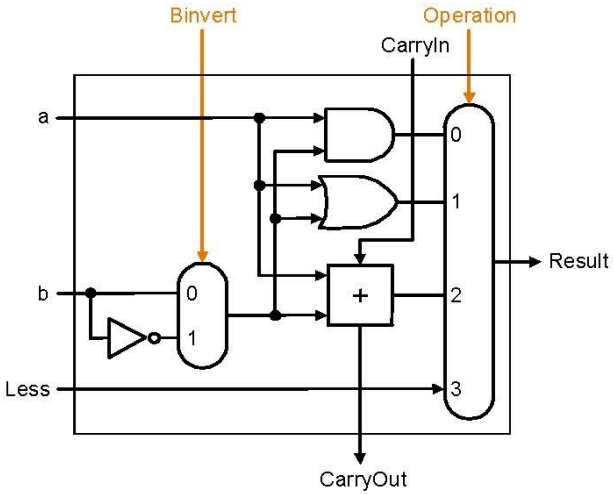
  – use subtraction: (a-b) < 0 implies a < b

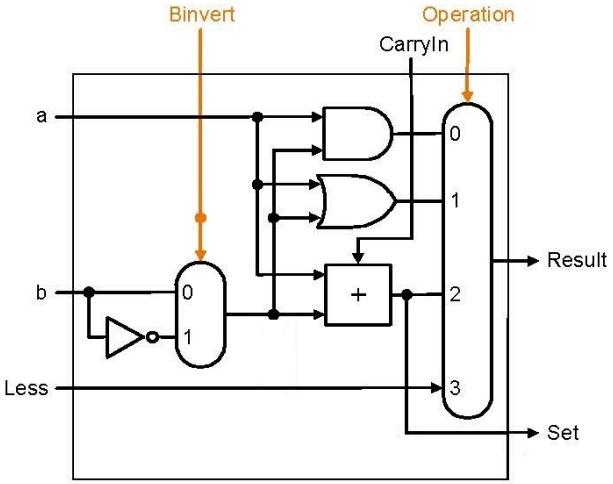
- Need to support test for equality (beq $t5, $t6, label)

  – use subtraction: (a-b) = 0 implies a = b => Zero=1

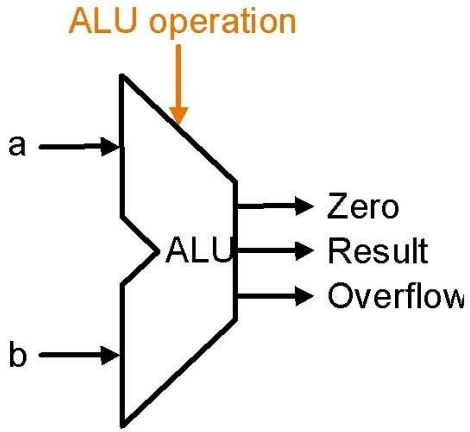
Supporting slt







Test for equality and complete ALU (3-bit control)

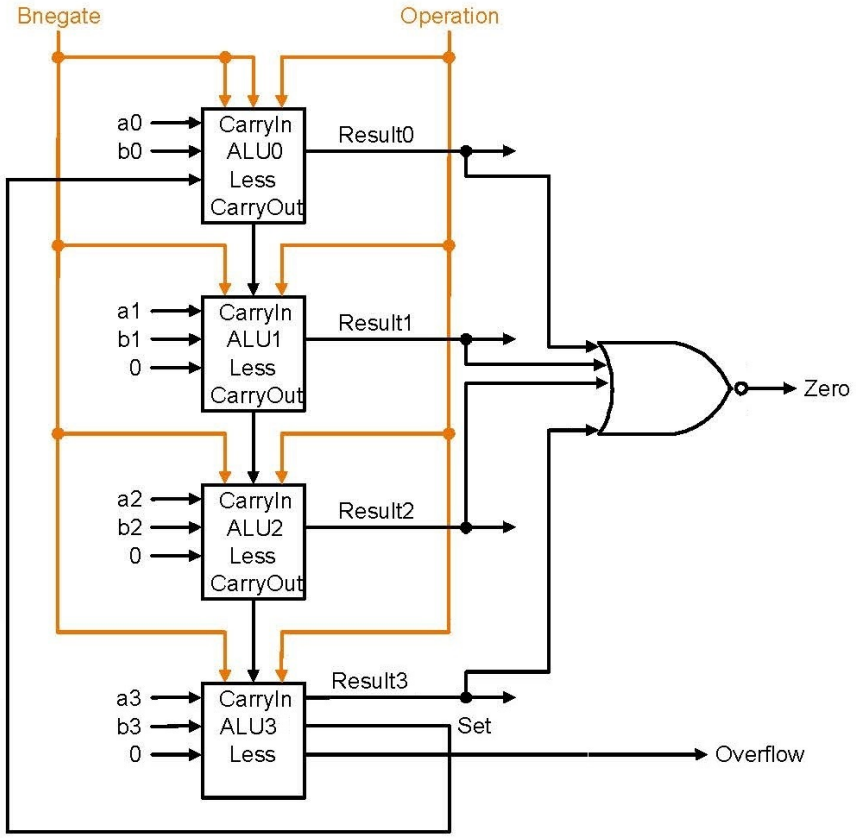


ALU operation:

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

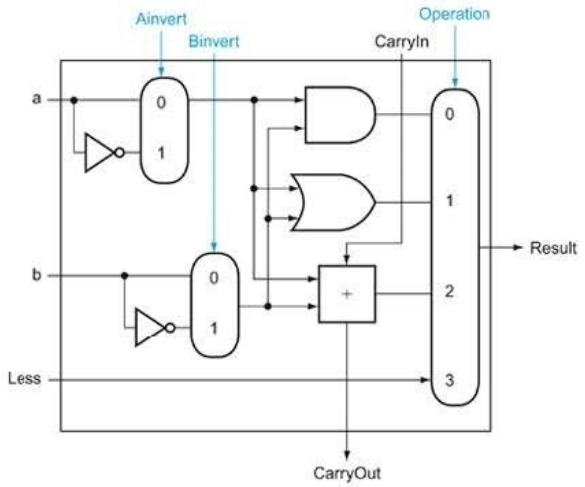
Control lines

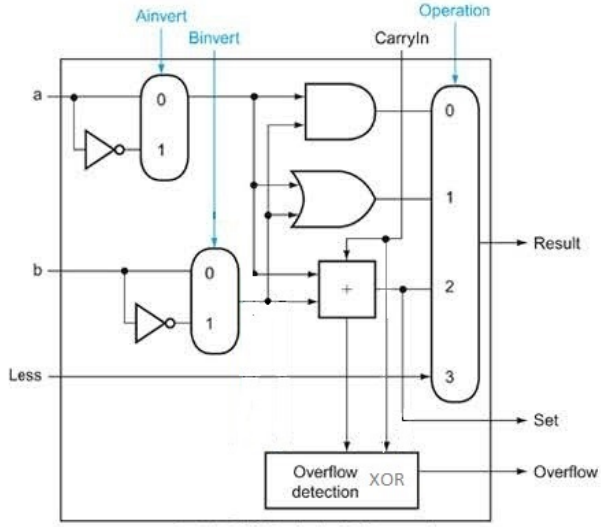
Bnegate	Operation	Instruction
0	00	and
0	01	or
0	10	add
1	10	sub
1	11	slt

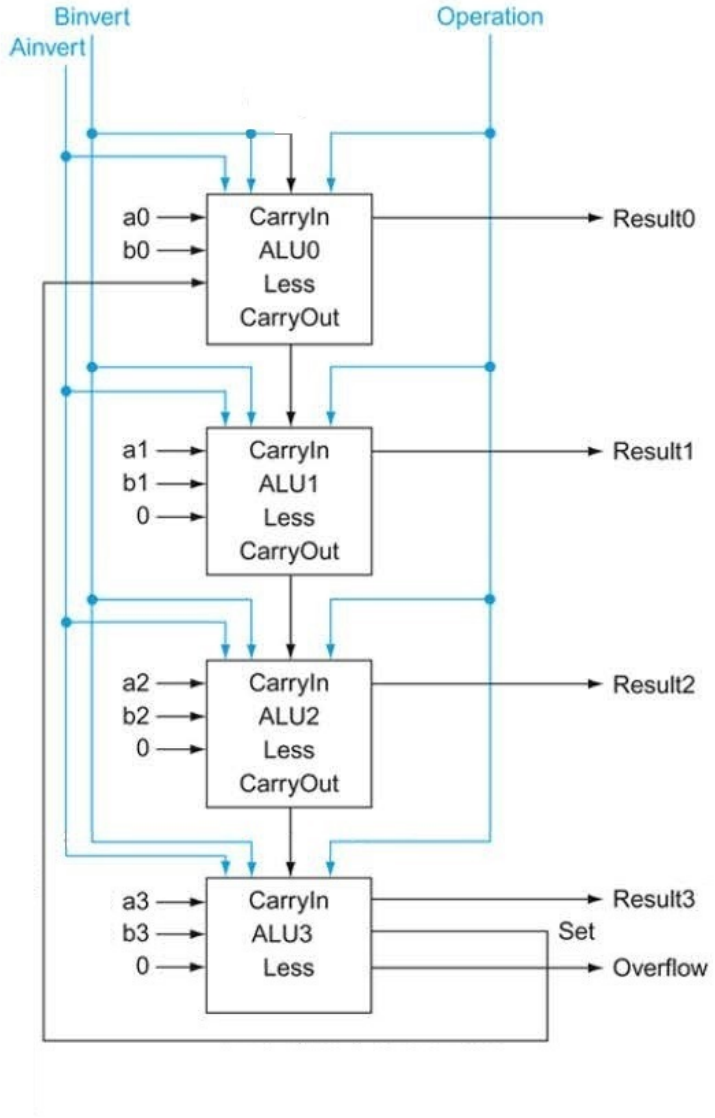


- ***Note: zero is a 1 when the result is zero!***

Adding NOR and NAND operations







Final ALU (4-bit control)

ALU Control lines

Ainvert	Bnegate	Operation	Instruction
0	0	00	and
0	0	01	or
0	0	10	add
0	1	10	sub
0	1	11	slt
1	1	00	nor
1	1	01	nand