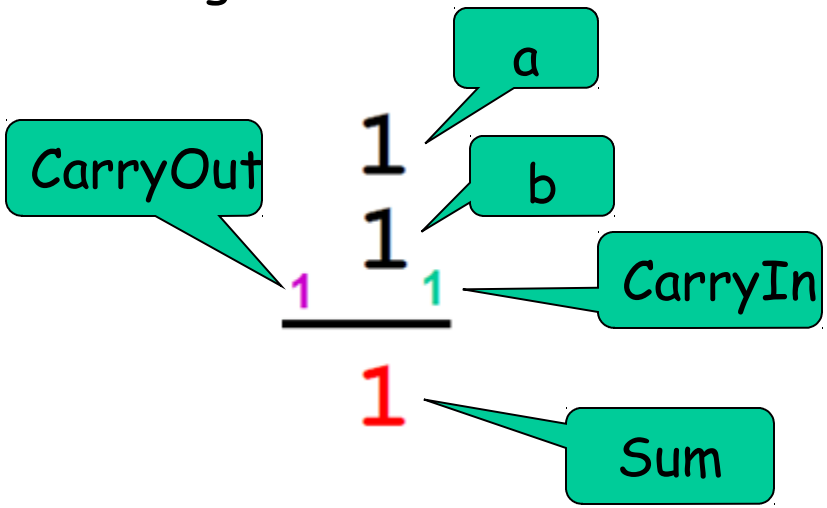
NWEN 242

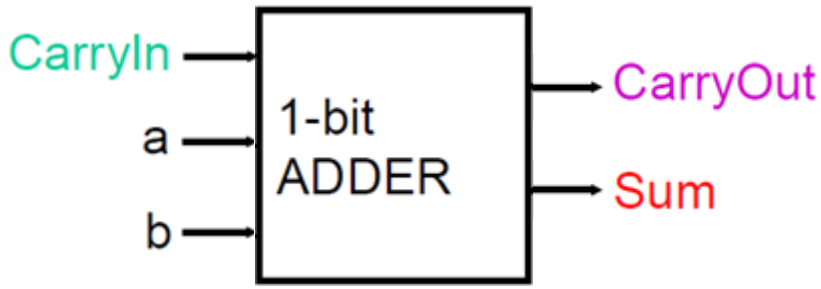
5. Arithmetic and logic unit (ALU)

Designing a circuit to do 1-bit addition

- A single-bit addition



A	b	CarryIn	CarryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



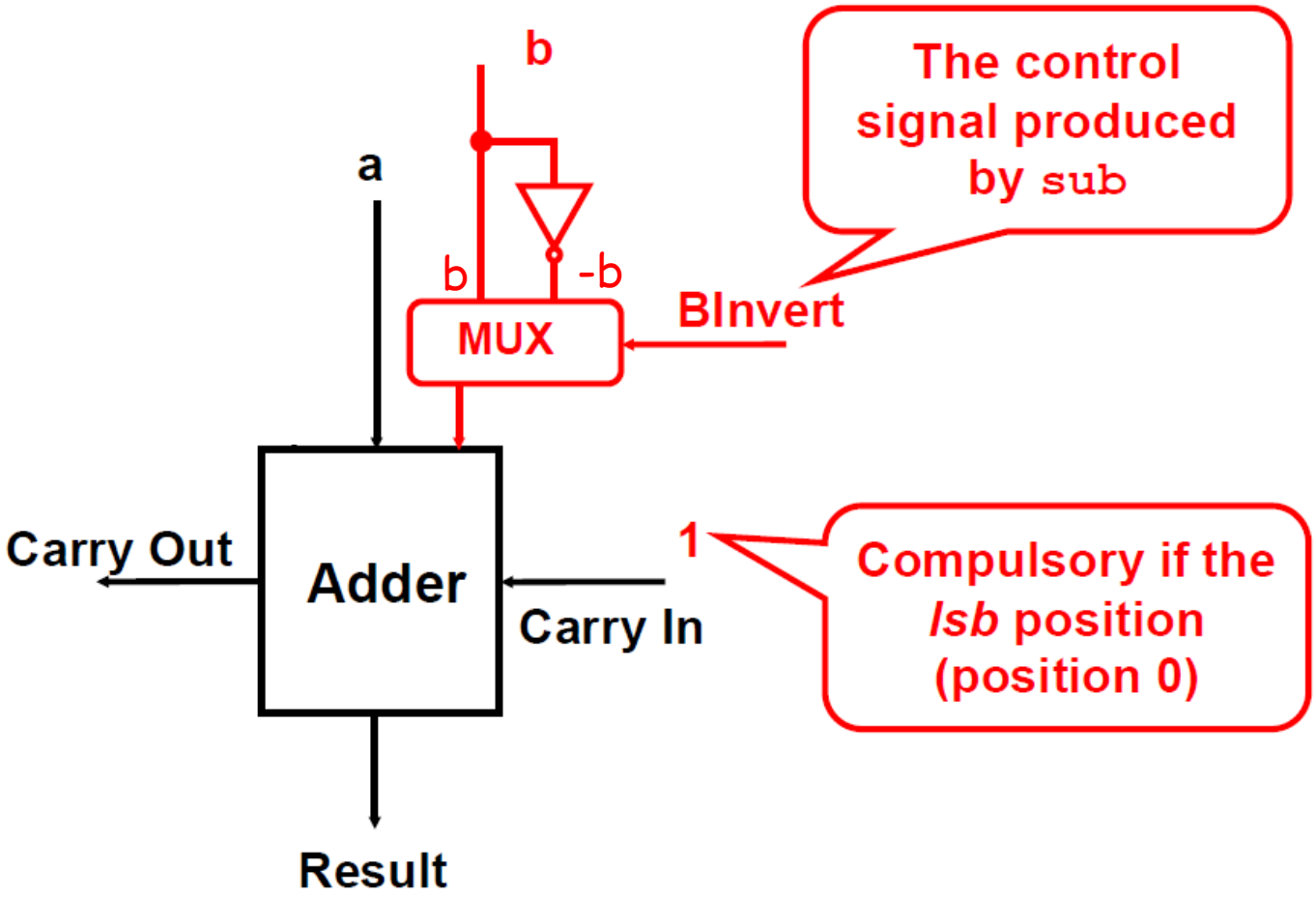
So how do we do subtraction?

- Remember we do not actually subtract

$$A - B$$

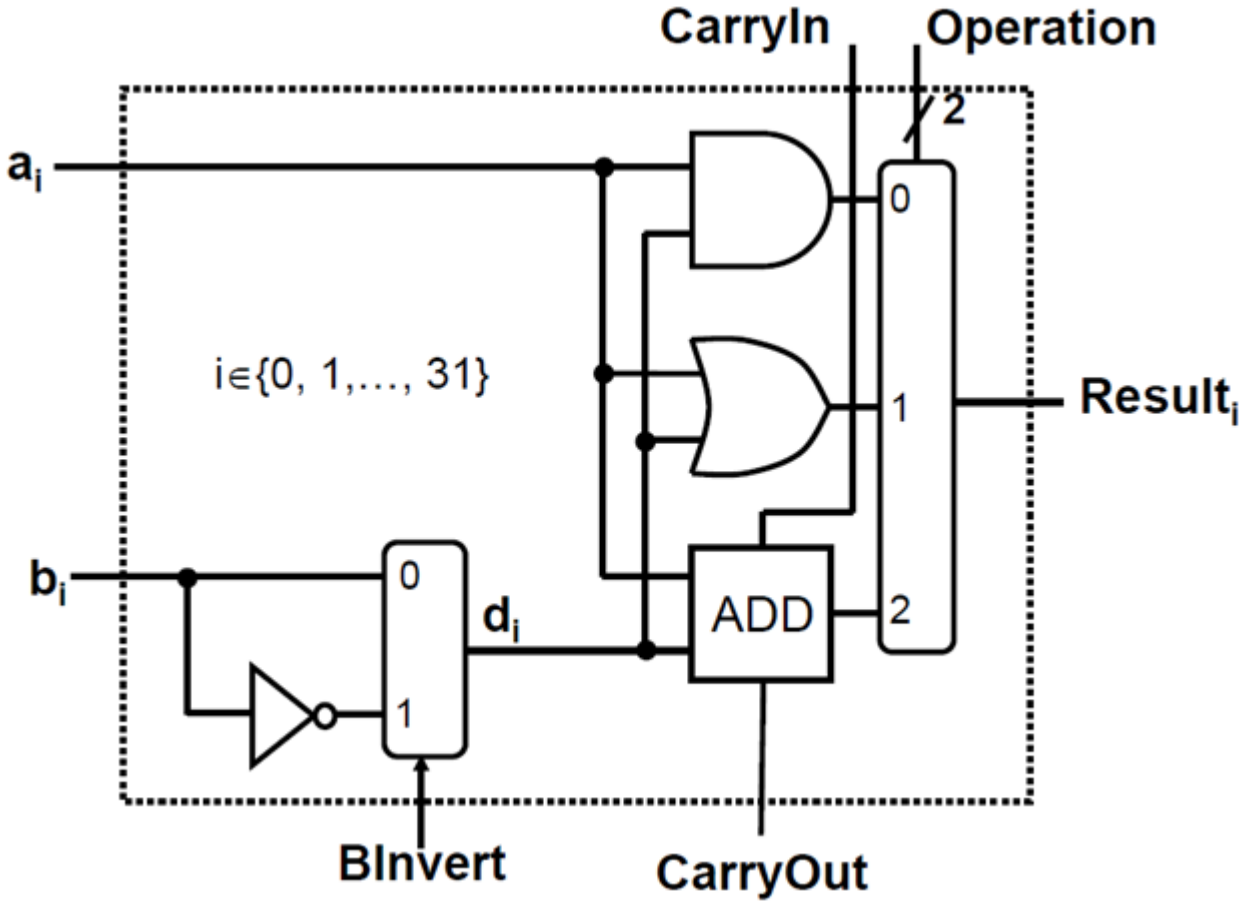
- Instead, we perform the following two steps
  - Calculate the two's complement of the negative number (-B)
  - Add A and –B through normal addition

1-bit ALU with subtraction



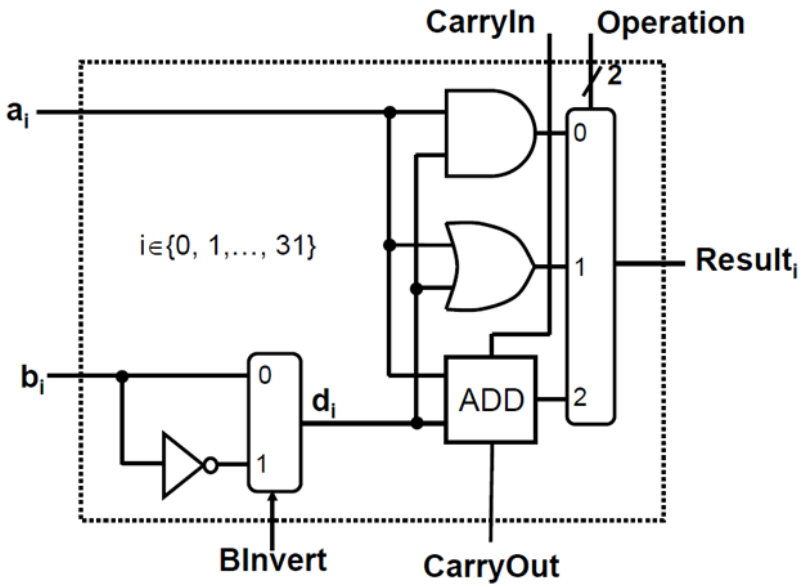
Circuit for a 1-bit ALU

- Notice that the operations are always performed but we don't care about the output



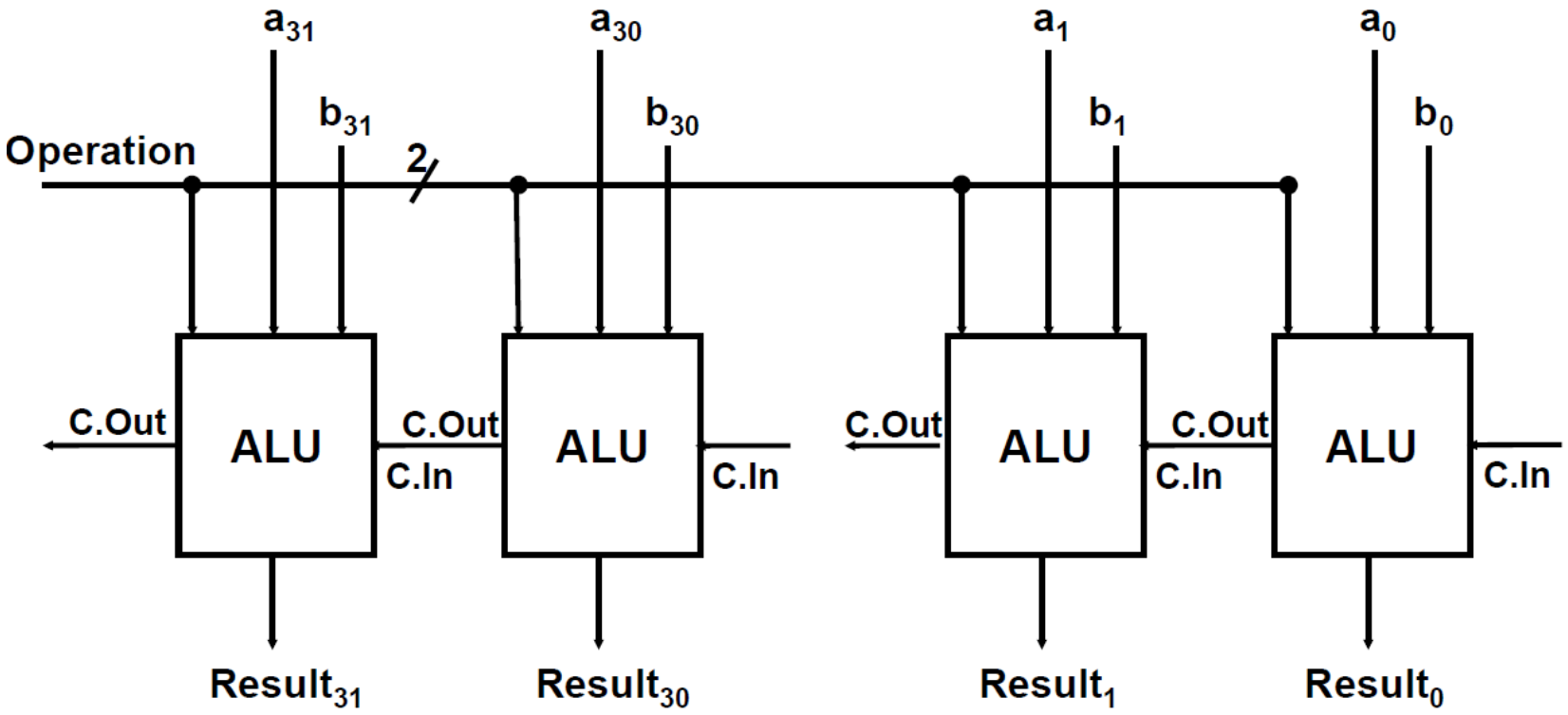
Quick exercise

- What would be the suitable control input in order to perform an AND operation?
  - A. BInvert = 1, Operation = 01
  - B. BInvert = 0, Operation = 01
  - C. BInvert = 1, Operation = 10
  - D. BInvert = 0, Operation = 00



32-bit (Ripple Carry) ALU

- Does ADD, SUB, OR and AND on 32-bits operands a and b
- Produces answer as Result and CarryOut



Extending ALU to do slt

- Consider the following slt instruction:

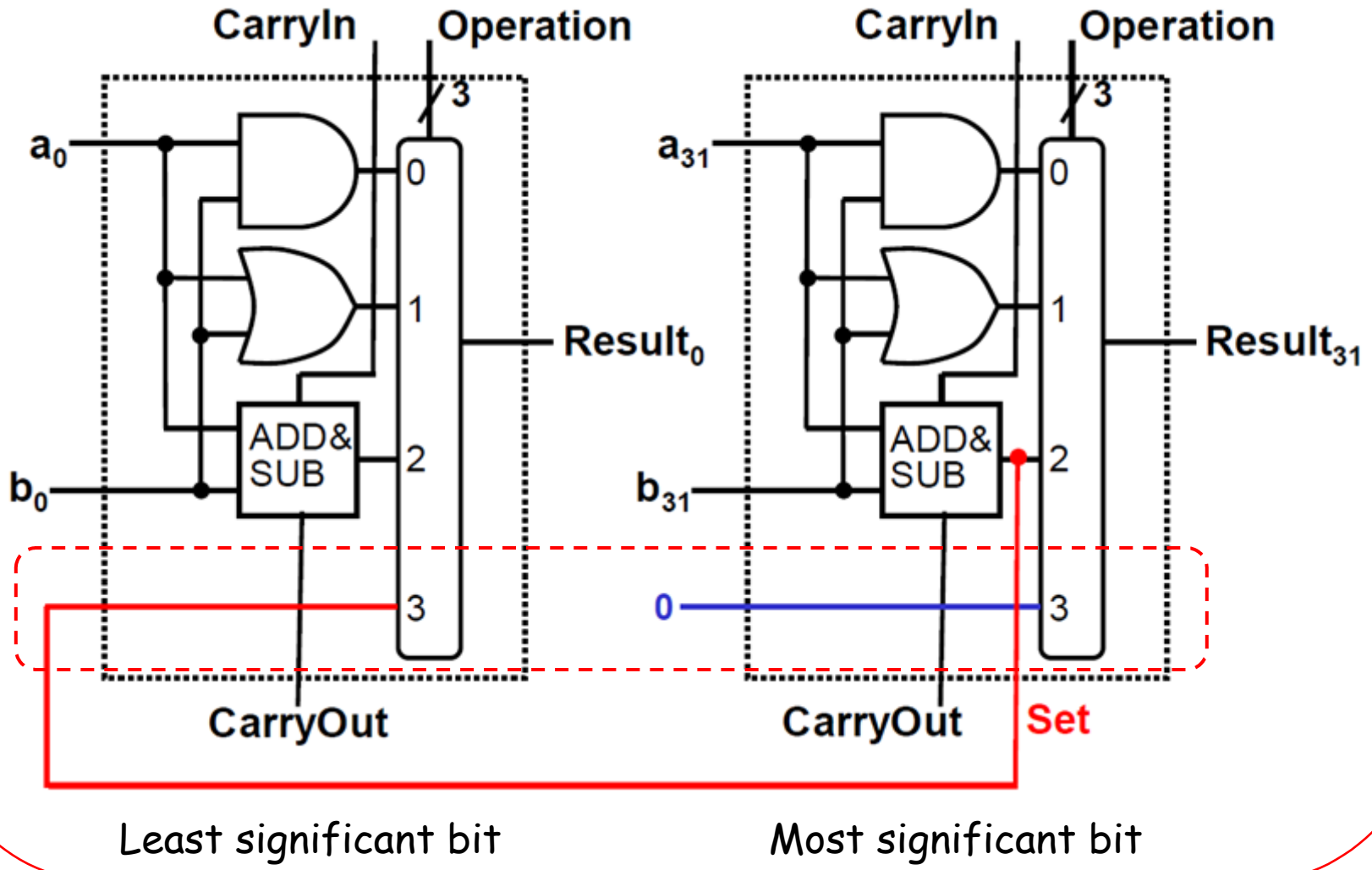
```
slt $d, $s, $t
```

- Set $d to 1 if $s<$t, otherwise set $d to 0

- Note
  - If $s<$t, then $s-$t is negative and bit 31 of the result is 1
  - If $s≥$t, the bit 31 of the result is 0

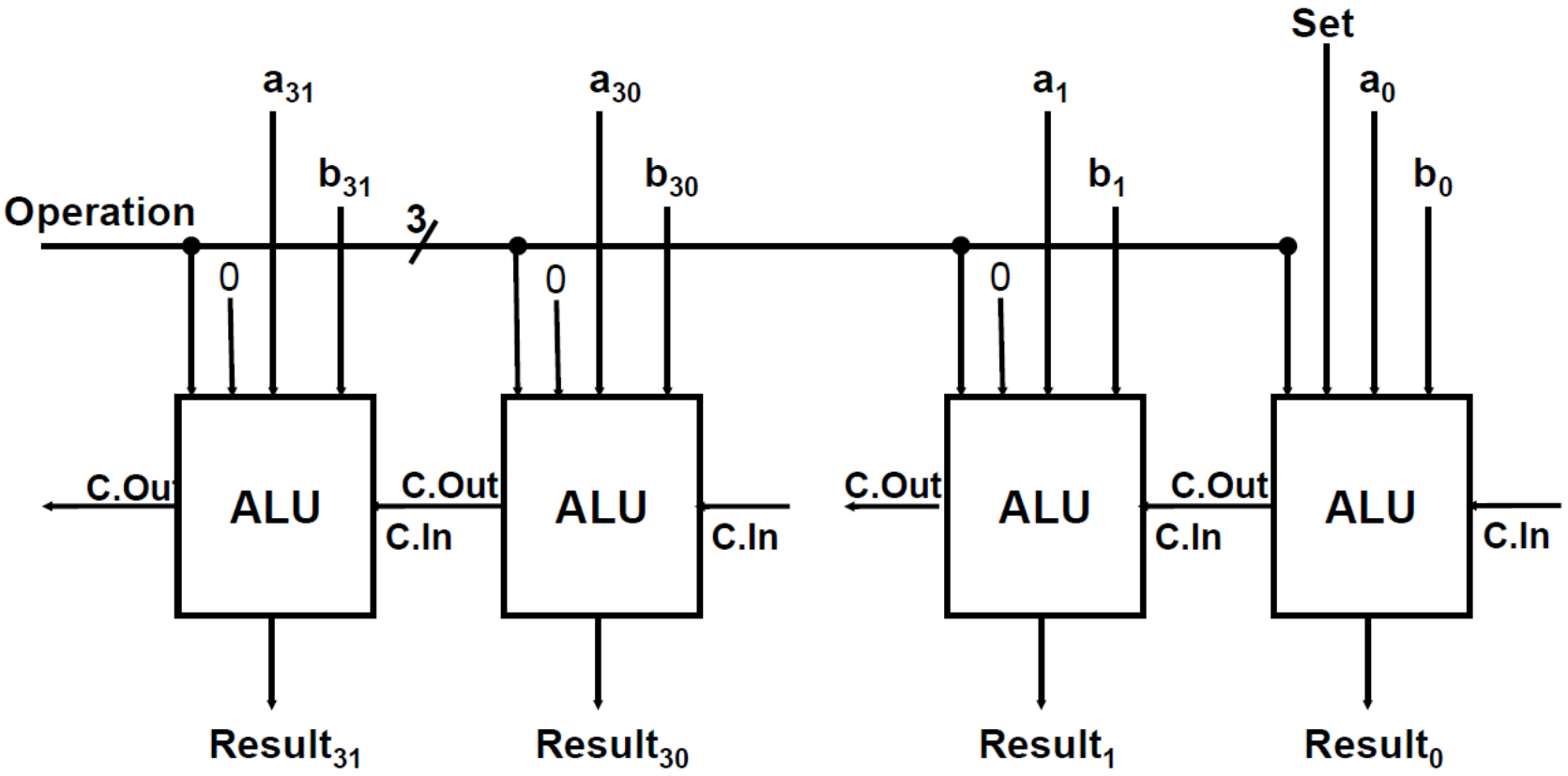
- Set comparison result
  - Copy the sign bit the of the result $s-$t to bit 0 of $d
  - Other bits of $d are simply set to 0

Circuit for producing comparison result



32-bit ALU with slt

- Supports ADD, SUB, OR, AND, and SLT on 32-bit operands a and b

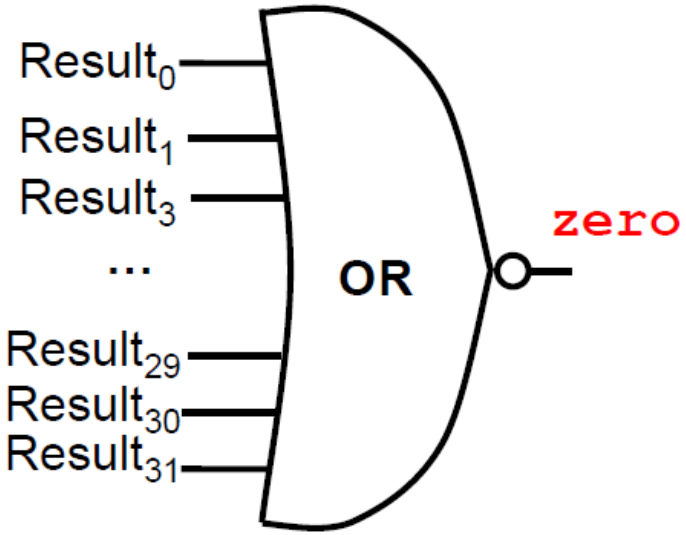


Implementing beq

- Consider the branch on equal instruction

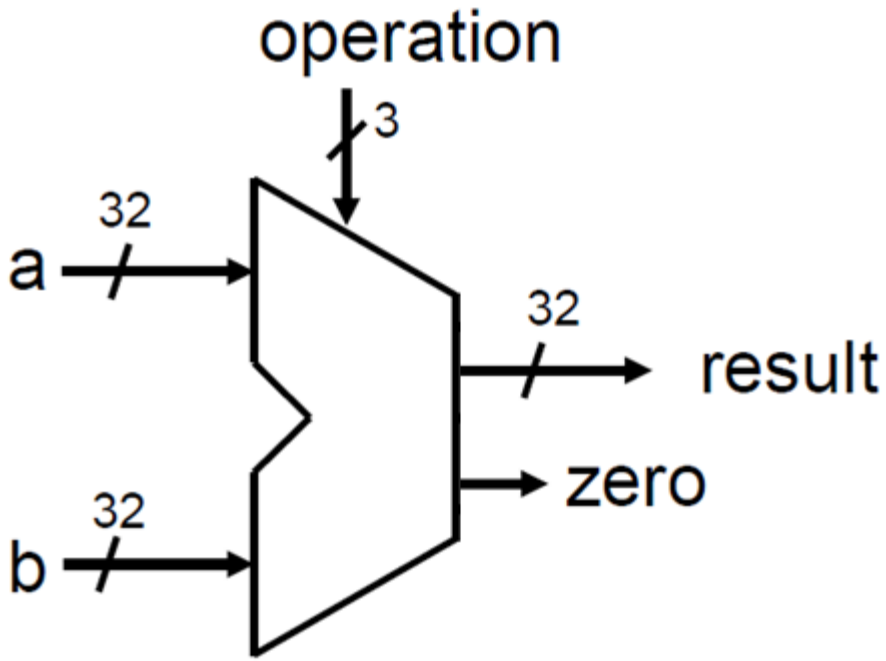
```
beq $s, $t, label
```

- We need a control signal, called zero
- Procedure
  - Use ALU to subtract $t from $s
  - OR the result bits
  - Invert



Block representation of ALU

- From now on, we shall refer to ALU using the following graphical symbol:



Overflow detection

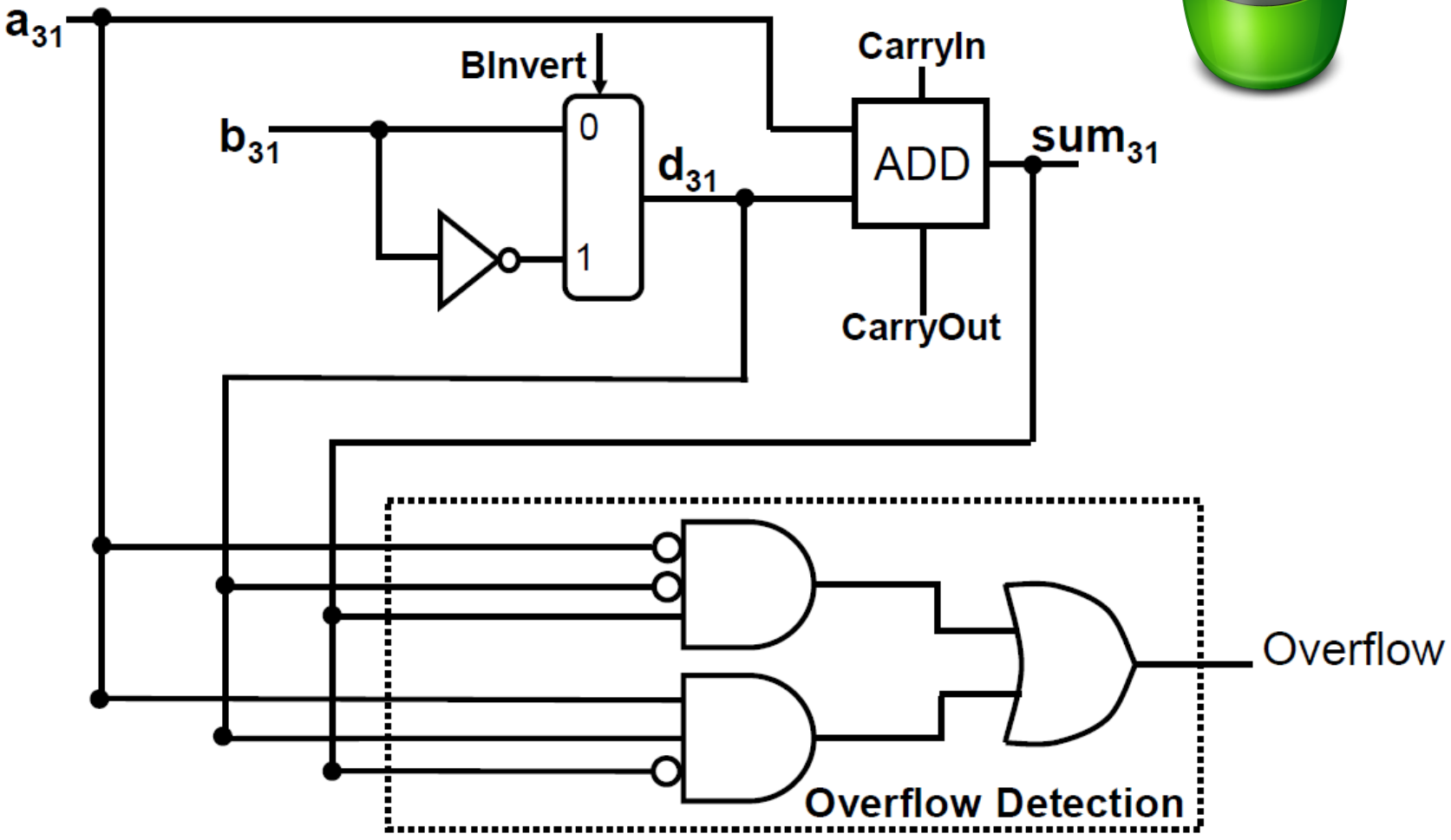
Operation	A Operand	B Operand	Result	Overflow
add	A ≥ 0	B ≥ 0	Result < 0	Yes
add	A < 0	B < 0	Result ≥ 0	Yes
sub	A ≥ 0	B < 0	Result < 0	Yes
sub	A < 0	B ≥ 0	Result ≥ 0	Yes

- Bit-level relationship (the most significant bit)
  - d is the two's complement of b

BInvert	a_{31}	d_{31}	sum_{31}	Overflow
0	0	0	1	1
0	1	1	0	1
1	0	0	1	1
1	1	1	0	1

$$\overline{a_{31}} \cdot \overline{d_{31}} \cdot sum_{31} + a_{31} \cdot d_{31} \cdot \overline{sum_{31}}$$

Overflow detection hardware



$$\overline{a_{31}} \cdot \overline{d_{31}} \cdot sum_{31} + a_{31} \cdot d_{31} \cdot \overline{sum_{31}}$$

Quick exercise

- Will overflow affect the ALU comparison result?
  - A. Yes
  - B. No
- Will overflow affect the zero output?
  - A. Yes
  - B. No

Summary

- A 1-bit ALU always performs
  - AND, OR
  - Either a 1-bit add or a 1-bit subtract
  - SLT
- A multiplexor decides what will appear at the output according to the operation on selector inputs
- A 32-bit ALU performs arithmetic and logic operations on whole words
- Instructions slt and beq use ALU to subtract target (rt) register from source (rs) register