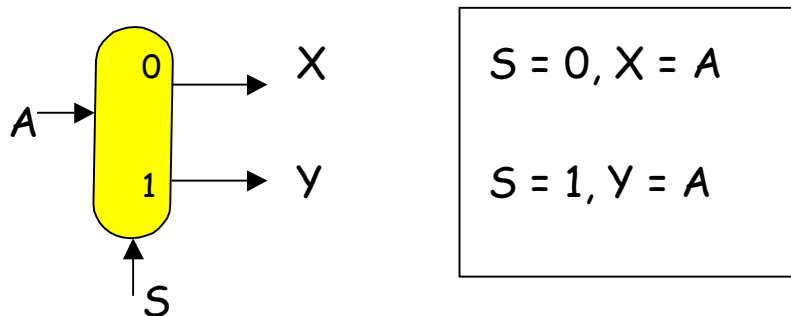


## Demultiplexors

A demux is a **one-to-many** switch.



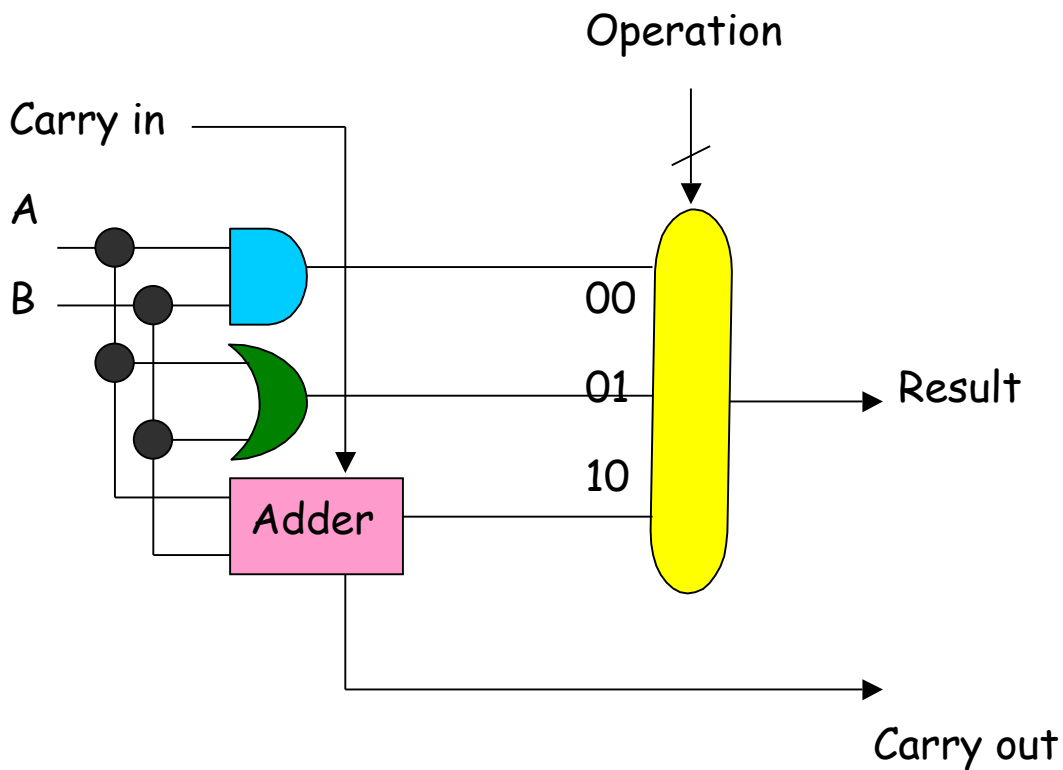
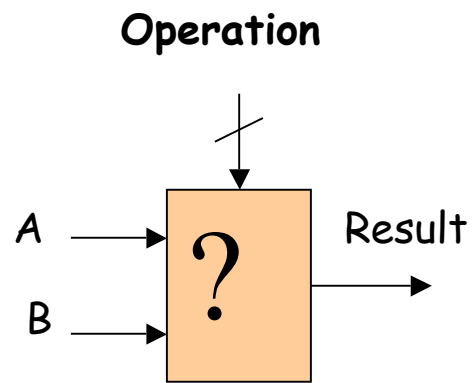
A 1-to-2 demux, and its specification.

So,  $X = \overline{S} \cdot A$ , and  $Y = S \cdot A$

**Exercise.** Design a 1-4 demux.

# A 1-bit ALU

Operation = 00 implies AND  
Operation = 01 implies OR  
Operation = 10 implies ADD



Understand how this circuit works.

Let us add one more input to the mux to implement

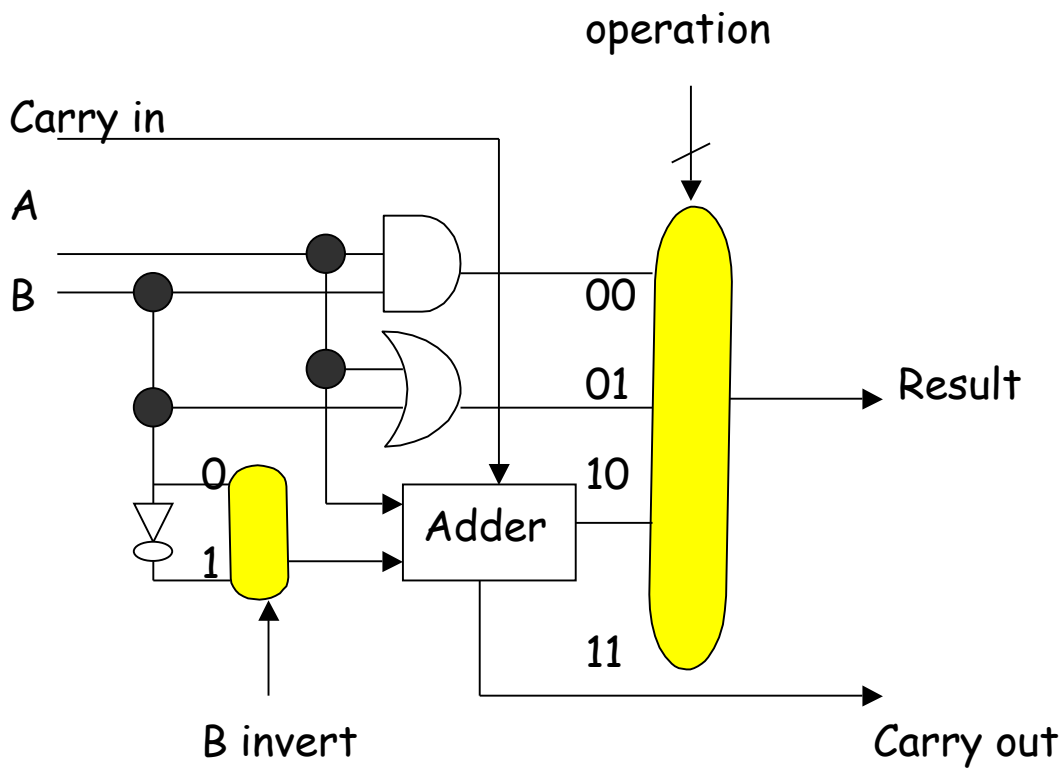
**slt** when the Operation = 11

## Converting an adder into a subtractor

$A - B$  (here - means arithmetic subtraction)

=  $A + 2$ 's complement of  $B$

=  $A + 1$ 's complement of  $B + 1$

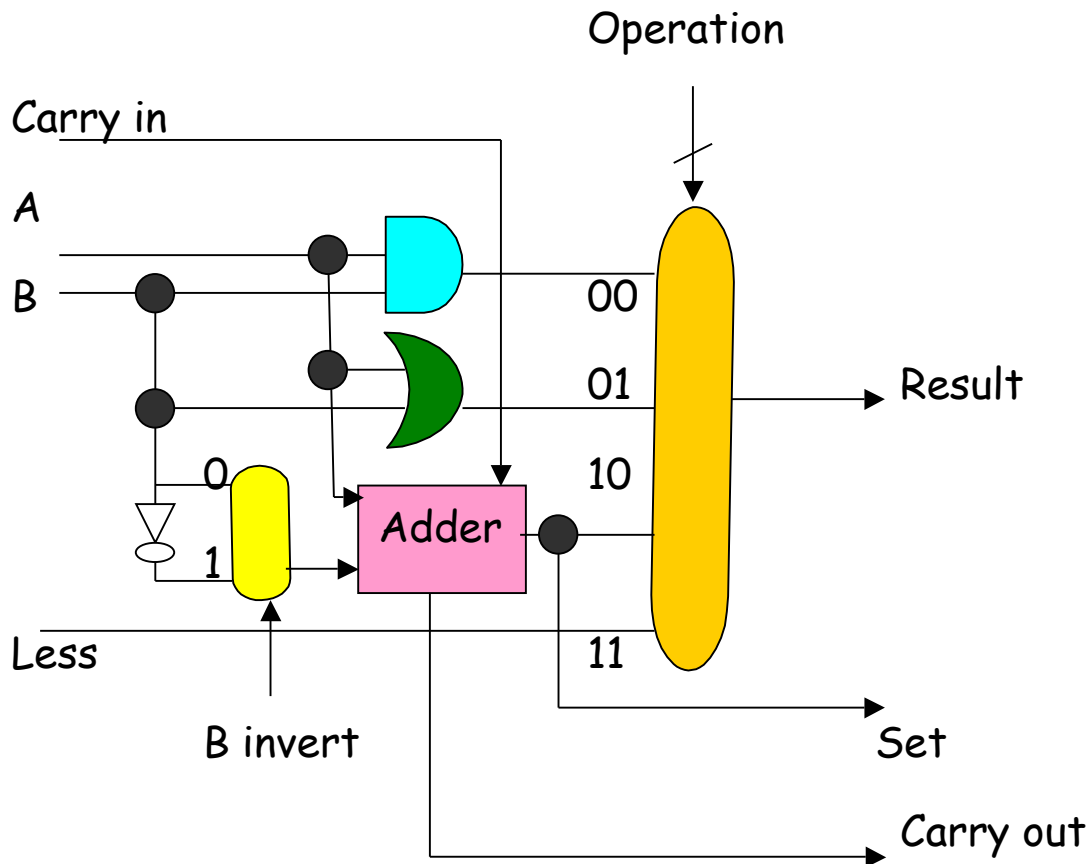


1-bit adder/subtractor

For subtraction, **B invert = 1** and **Carry in = 1**

## 1-bit ALU for MIPS

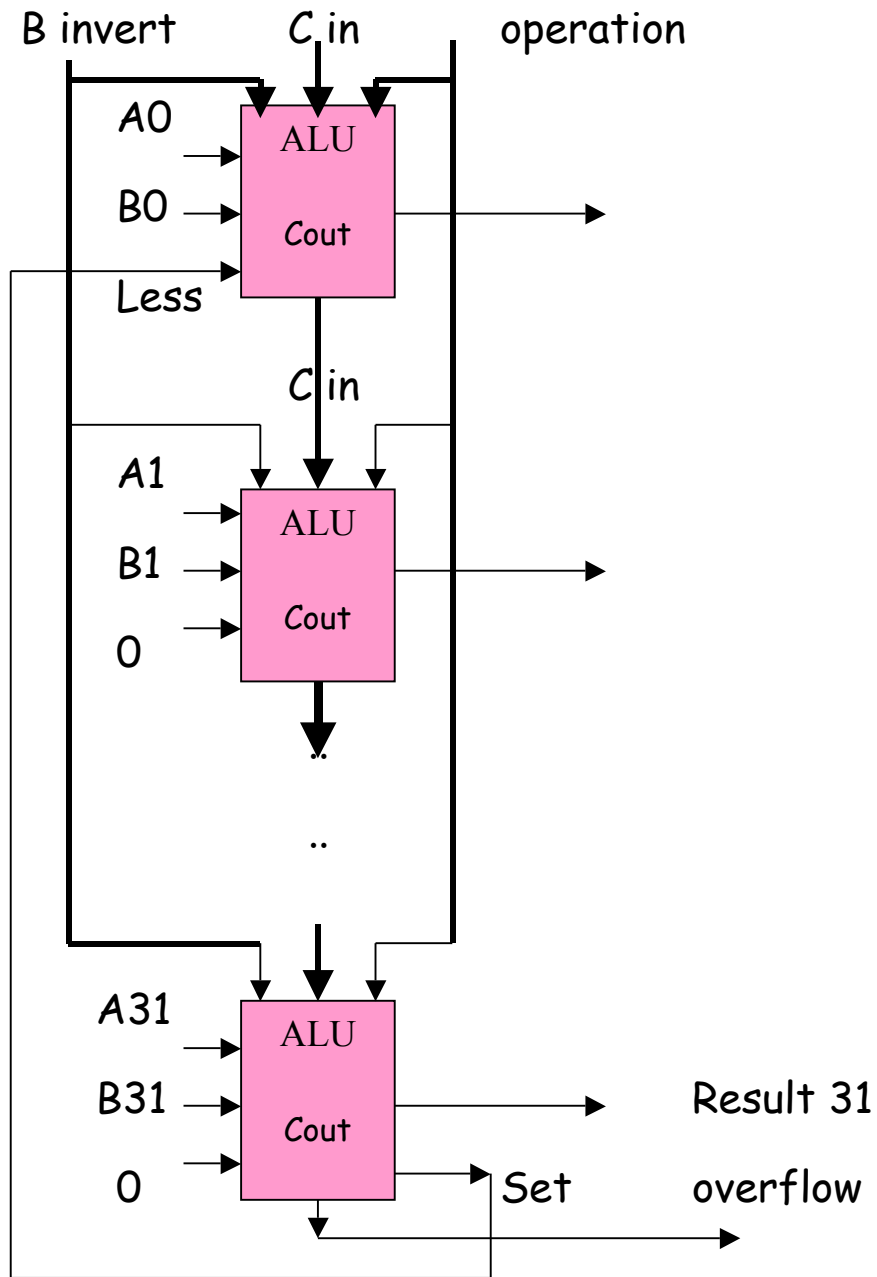
Assume that it has the instructions add, sub, and, or, slt.



**Less** = 1 if the **32-bit number A** is less than the **32-bit number B**. (Its use will be clear from the next page)

We now implement **slt** (If  $A < B$  then Set = 1 else Set = 0)

# A 32-bit ALU for MIPS



## Combinational vs. Sequential Circuits

### Combinational circuits

The output depends only on the current values of the inputs and not on the past values. Examples are adders, subtractors, and all the circuits that we have studied so far

### Sequential circuits

The output depends not only on the current values of the inputs, but also on their past values. These hold the secret of how to memorize information. **We will study sequential circuits later.**