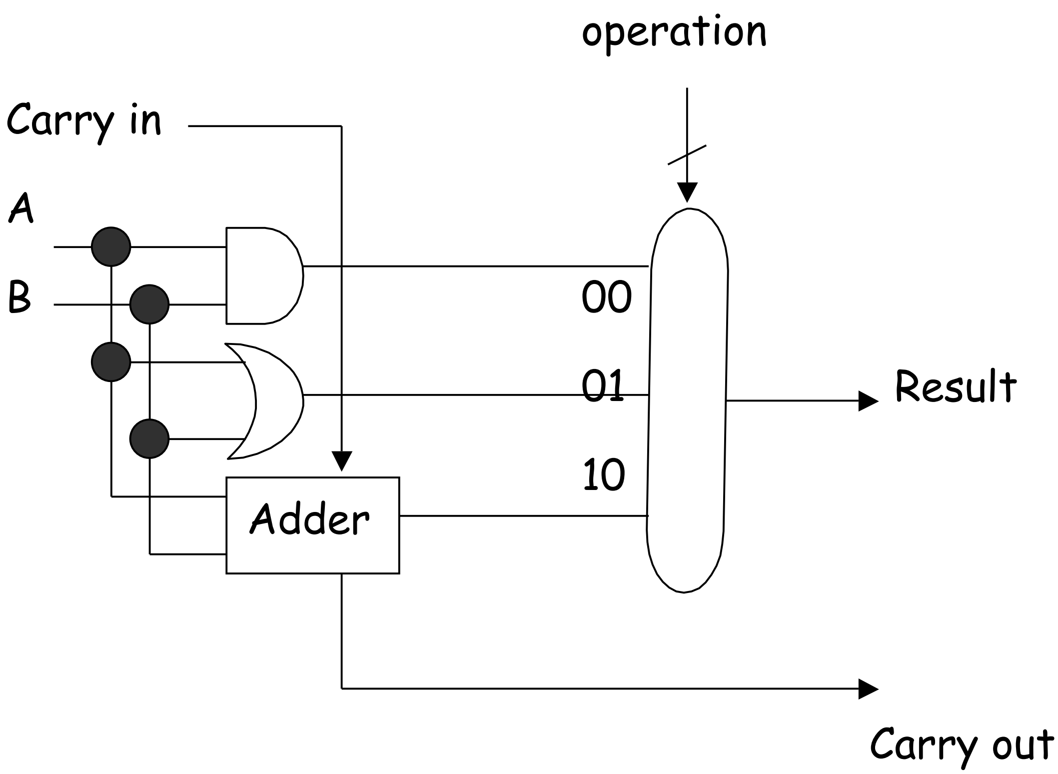
A 1-bit ALU



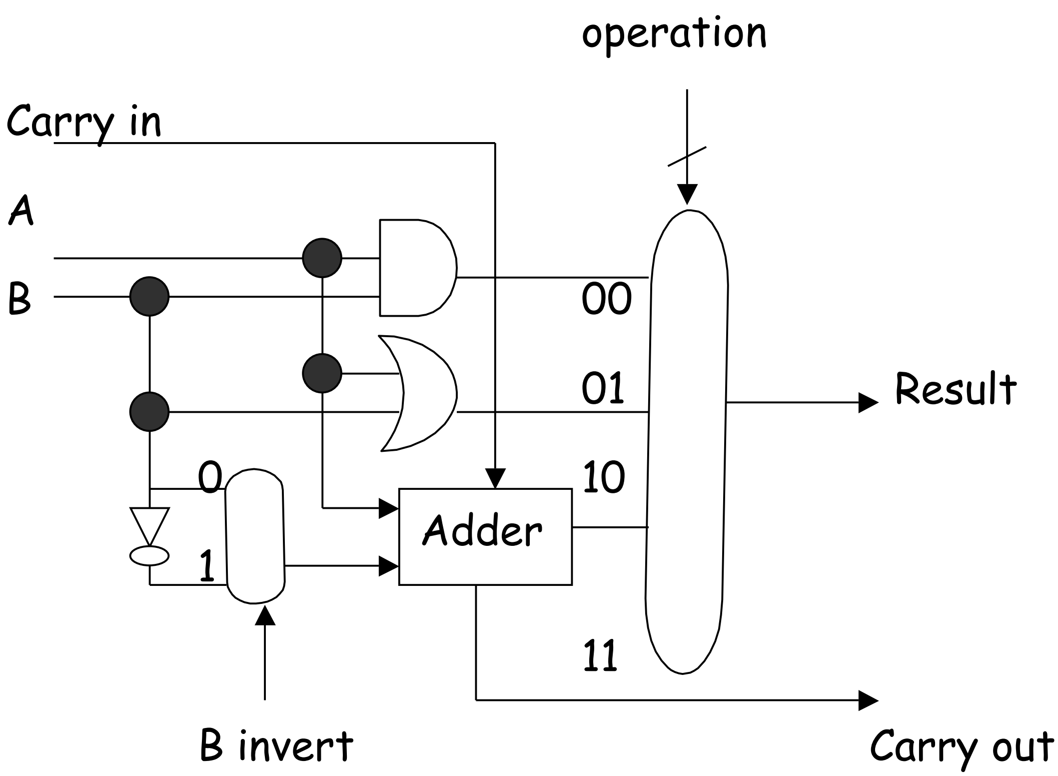
Understand how this circuit works.

Then look at the diagram of the 32-bit ALU in p.235.

Need to add one more input to the mux to implement **slt**

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	

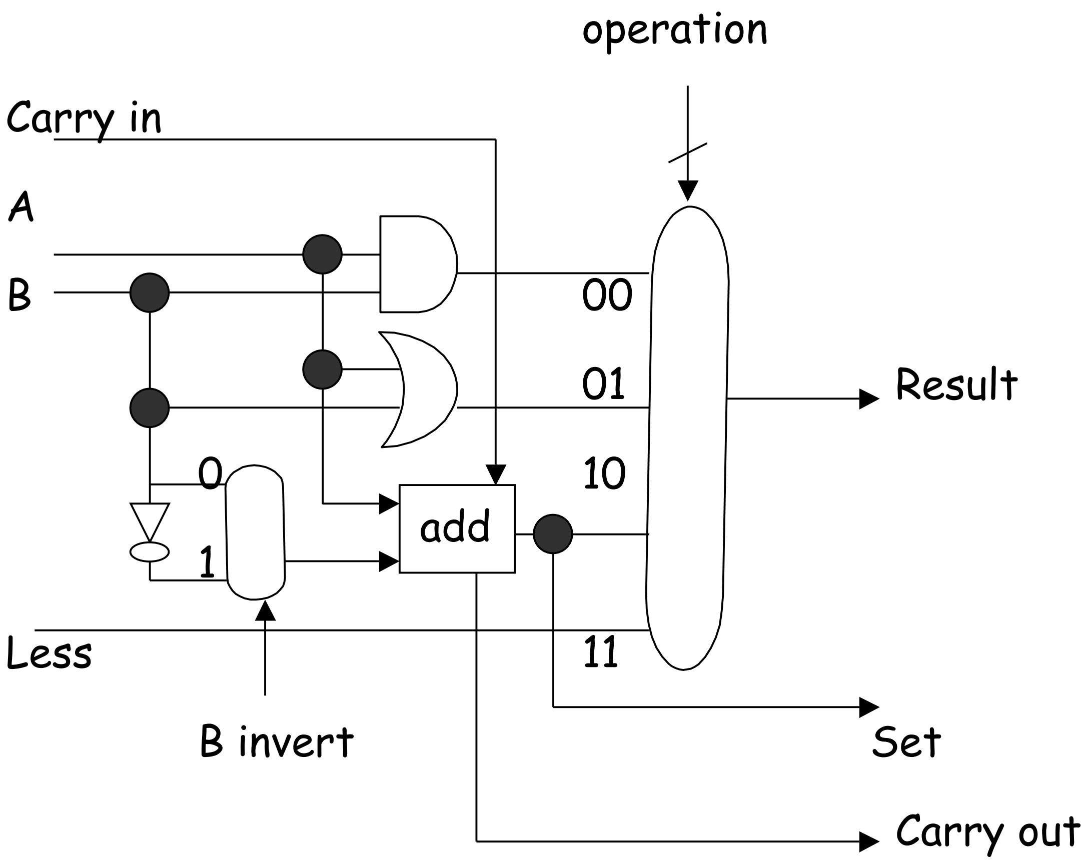


1-bit adder/subtractor

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

1-bit ALU for MIPS

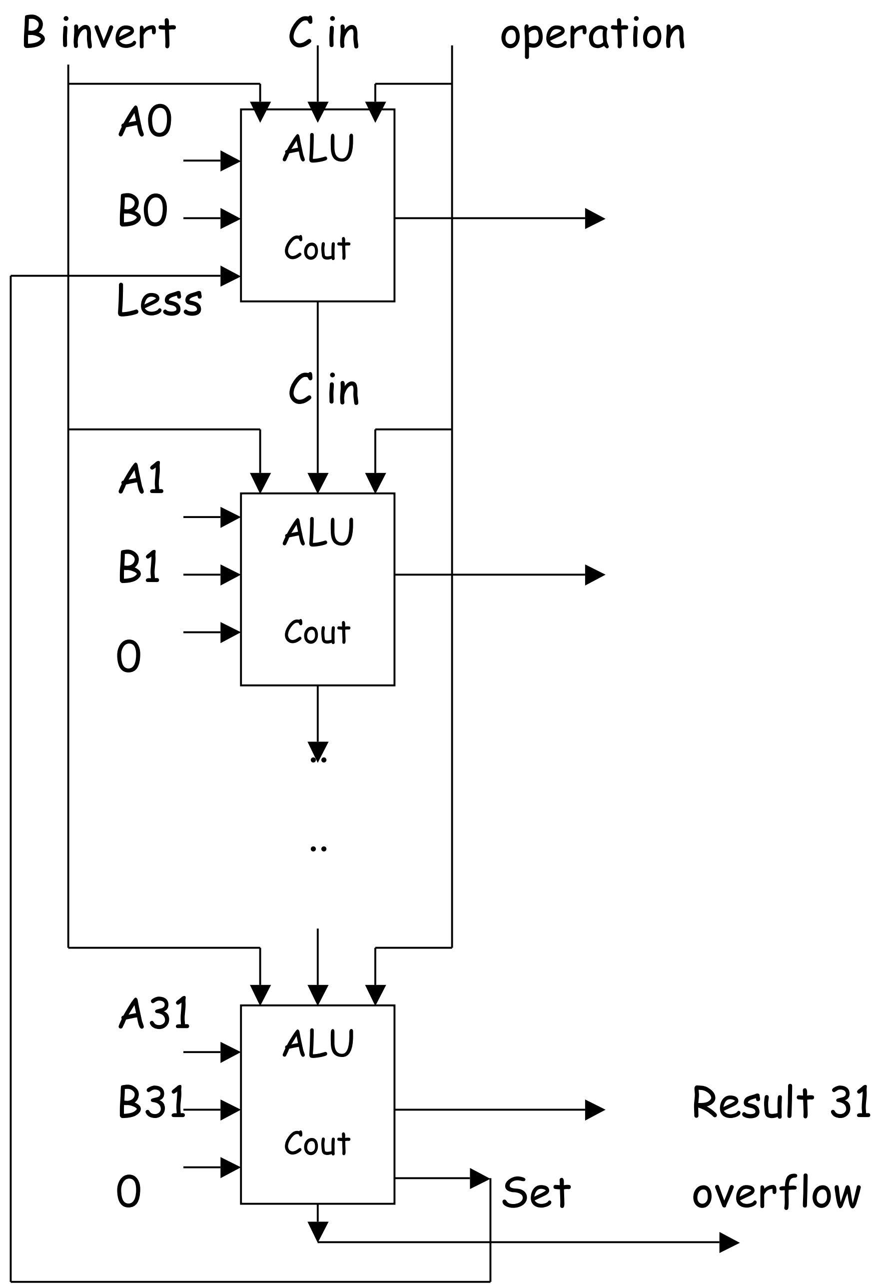
Assume that it has the instructions add, sub, and, or, slt.



Less will be used to detect if the 32-bit number A is less than the 32-bit number B. See the next page.

If A < B then Set = 1 else Set = 0

A 32-bit ALU for MIPS



Fast Carry Propagation

During addition, the carry can trigger a "ripple" from the LSB to the MSB. This slows down the speed of addition.

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	+
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	

How to overcome this? Consider the following:

$$c1 = a0.b0 + a0.c0 + b0.c0$$
$$= a0.b0 + (a0 + b0).c0$$
$$= g0 + p0.c0 \text{ (where } g0 = a0.b0,\ p0 = a0+b0)$$

$$c2 = a1.b1 + (a1 + b1).c1$$
$$= g1 + p1.(g0 + p0.c0)$$
$$= g1 + p1.g0 + p1.p0.c0$$

$$c4 = g3 + p3.g2 + p3.p2.g1 + p3.p2.p1.g0 + p3.p2.p1.p0.c0$$

$c32 = ?$

It will be complex. But you can use a two-level circuit to generate c4. This will expedite addition. But it is impractical due to the complexity.

Practical circuits use a two-phase approach. See the example of the 16-bit adder, designed from four 4-bit adders in p.246. Let

$$G0 = g3 + p3.g2 + p3.p2.g1 + p3.p2.p1.g0$$

$$G1 = g7 + p7.g6 + p7.p6.g5 + p7.p6.p5.g4$$

$$G2 = g11 + p11.g10 + p11.p10.g9 + p11.p10.p9.g8$$

$$G3 = g15 + p15.g14 + p15.p14.g13 + p15.p14.p13.g12$$

$$P0 = p3.p2.p1.p0$$

$$P1 = p7.p6.p5.p4$$

$$P2 = p11.p10.p9.p8$$

$$P3 = p15.p14.p13.p12$$

Then

$$C1 = G0 + P0.c0$$

$$C2 = G1 + P1.G0 + P1.P0.c0$$

$$C3 = G2 + P2.G1 + P2.P1.G0 + P2.P1.P0.c0$$

$$C4 = G3 + P3.G2 + P3.P2.G1 + P3.P2.P1.G0 + P3.P2.P1.P0.c0$$

This is implemented in the **carry look-ahead adder**.

See the figure in p. 246 of your textbook.

How much faster is the carry look-ahead adder?

Multiplication

Be familiar with shift operations first

0 1 1 1 0 1 0 0	1 0 1 0 0 0 1 1
Shift right →	← shift left
0 0 1 1 1 0 1 0	0 1 0 0 0 1 1 1

```
            1  0  0  1      (multiplicand)
            0  1  0  1      (multiplier)
------------------------------------------
            1  0  0  1
         0  0  0  0
      1  0  0  1
   0  0  0  0
------------------------------------------
   0  1  0  1  1  0  1
```

See the diagrams in page 254 and 255 of your textbook, and learn how a multiplier works.

Floating point representation

A scheme for representing a number **very small** to **very large**. It is widely used in the scientific world. Consider, the floating point number

	Exponent E	significand F
+/-	x x x x	y y y y y y y y y y y y

In **decimal** it means (+/-) 1. yyyyyyyyyyyy $\times 10^{xxxx}$

In **binary**, it means (+/-) 1. yyyyyyyyyyyy $\times 2^{xxxx}$

(The 1 is implied)

IEEE 754 representation

s	xxxxxxxx	yyyyyyyyyyyyyyyyyyyyyyy
1	8	23 bits

Single precision

Largest = 1.111... $\times\ 2^{+127}$ $\approx 2 \times 10^{+38}$

Smallest = 1.000 ... $\times\ 2^{-128}$ $\approx 1\ \times 10^{-38}$

These can be positive and negative, depending on s.

IEEE 754 double precision (64 bits)

S	exponent	significand
1	11 bits	52 bits

Largest = $1.111\ldots \times 2^{+1023}$

Smallest = $1.000\ldots \text{X } 2^{-1024}$

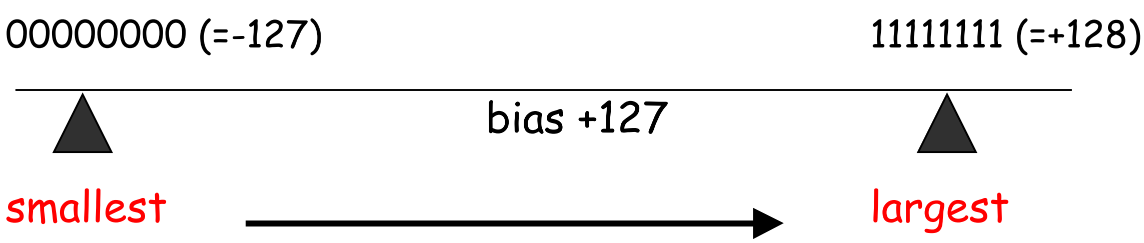
What do you mean by **overflow** and **underflow** in FP?

An overflow occurs when the number if too large to fit in the frame. An underflow occurs when the number is too small to fit in the given frame.

Biased Representation

$$\left.\begin{array}{ll} \text{Exponent} = 11111111 & 2^{-1} \\ \text{Exponent} = 00000000 & 2^{0} \end{array}\right\} \quad \text{awkward for sorting}$$

However, to facilitate sorting, IEEE 754 treats 00…0 as the most negative, and 1,11..1 as the most positive exponent. This amounts to using a **bias** of 127.



So, value = $(1)^{s} \times (1+\text{significand}) \times 2^{(\text{exponent} - \text{bias})}$

Practice simple conversions from fraction to FP.

Floating Point Addition

Example using decimal

$A = 9.999 \times 10^{1}$, $B = 1.610 \times 10^{-1}$, $A+B = ?$

Step 1. Align the smaller exponent with the larger one.

$B = 0.0161 \times 10^{1} = 0.016 \times 10^{1}$ (round off)

Step 2. Add significands

$9.999 + 0.016 = 10.015$, so $A+B = 10.015 \times 10^{1}$

Step 3. Normalize

$A+B = 1.0015 \times 10^{2}$

Step 4. Round off

$A+B = 1.002 \times 10^{2}$

Now, try to add 0.5 and –0.4375 in binary.

Floating Point Multiplication

Example using decimal

A = 1.110 x 1010, B = 9.200 x 10-5 A x B =?

Step 1. Exponent of A x B = 10 + (-5) = -5

Step 2. Multiply significands

$1.110 \times 9.200 = 10.212000$

Step 3. Normalize the product

$10.212 \times 10^{-5} = 1.0212 \times 10^{-5}$

Step 4. Round off

$A \times B = 1.021 \times 10^{-5}$

Step 5. Decide the sign of A x B (+ x + = +)

So, $A \times B = + 1.021 \times 10^{-5}$

Now try to multiply 0.5 with –0.4375 in binary. Use IEEE 754 format.