Two's complement addition

Range of a bit vector of length w

$$-2^{w-1} \leq x, y \leq 2^{w-1} - 1$$

(negative overflow / positive overflow)

If $w = 4$

$$-8 \leq x, y \leq 7$$

$$-2^{w} \leq x + y \leq 2^{w-1} - 2$$

If we add 2 w-bit vectors, we need w+1 bits to store the output

Definition

$$x +_w^t y \begin{cases} x + y - 2^w ; & 2^{w-1} \leq x + y \text{ (positive overflow)} \\ x + y ; & -2^w \leq x + y \leq 2^{w-1} \\ x + y + 2^w ; & x + y < -2^{w-1} \end{cases}$$

2	5	7	7
0010	0101	~~0~~0111	0111
5	5	10	-6
0101	0101	~~0~~1010	1010
-8	-5	-13	
1000	1011	*0011	

$-1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = -6$

-13 → 0011

+13 → 3

Two's complement negation

→ additive inverse

$$x + (-x) = 0$$

For $w = 4$

$$-8 \leq x \leq 7$$

1 → -1 0 → 0

6 → -6 -8 → (-2^{w-1}) ?

-7 → 7

$$(-2^{w-1}) + (-2^{w-1}) = 0$$

$$= -2 \times 2^{w-1}$$

$$= -2^{w} + 2^{w}$$ (since we have a negative overflow)

$$= 0$$

Unsigned Multiplication

$$0 \leq x, y \leq 2^{w} - 1$$

$$0 \leq x, y \leq (2^{w} - 1)^{2}$$

$$\leq 2^{2w} - 2^{w+1} + 1$$

$w = 2$ This will need $2w$ bits to store.

$3 \times 3 = 9$

$[11]_2 \; [11]_2 = [1001]_2$ $\quad 01_2 = 1_w$

$$\begin{array}{r} 111 \\ 11- \\ \hline 1001 \\ \hline 8+0+0+1 = 9 \end{array}$$

$$x *_w^u y = (x * y) \bmod 2^w$$

$9 \bmod 4 = 1$

Floating point

$0.1 + 0.5 = 0.6$

$0.1 + 0.5$

$= 0.6000000000000001$

Because it is stored as a binary number.

Humans

	100	10	1	.	1/10	1/100	1/1000
Base 10			0	.	1		

Computers

	4	2	1	.	1/2	1/4	1/8	1/16	1/32	1/64
Base 2					0	0	0	1	1	0
								0.0625	0.03125	

$0.0\,0011\,0011\,0011\,0011$ → 0.09375

0.9999999991234 //

Analogy in base 10

$$\frac{1}{3} + \frac{1}{3} + \frac{1}{3} = \frac{3}{3} = 1$$

$$= 0.\overline{3} + 0.\overline{3} + 0.\overline{3}$$

$$= 1$$

0.33333333 ——

0.33 > ——

0.3 ——

0.999 ——

3 0000 | 0000 → speed of light 3×10^8

0.0000004

$0.0_{10} \longrightarrow 0.0_2$

0.25_{10} (1/4) $\rightarrow 0.01_2$

0.125_{10} 1/8 $\rightarrow 0.001_2$

0.25_{10} 2/8 $\rightarrow 0.010_2$

significant digits $\Rightarrow 3$

$$3 \times 10^8 \times 4 \times 10^{-7}$$

$$12 \times 10^1$$

120.

① Find if a number is a power of 2 using bitwise operators.

+ - * | ^ << >>

2^0 - 0001

2^1 - 0010

2^2 - 0100

2^3 - 1000

2^4 - 10000

6 → 0110

5 → 0101

100

int x = 4;

0100 (x)

* 0011 (x-1)

0000

0 → 0000

-1 → ~1111

0000

1000 - 8

* 0111 - 7

0000

(x & (x-1))

↓

0 → It is a ~~multiple~~ power of 2

Not 0 → It is not a power of 2.

(2) Find if a number is odd or even.

$0_{10} \rightarrow 0_2$

$1_{10} \rightarrow 1_2$

$2_{10} \rightarrow 0010_2$

$3_{10} \rightarrow 0011_2$

$6_{10} \rightarrow 0110_2$

$15_{10} \rightarrow 1111_2$

$16_{10} \rightarrow 10000_2$

is Even //

return $((x \& 1) == 0)$

MASK

$2 \rightarrow 0000\ 0010_2$

$1 \rightarrow 0000\ 0001_2$ (AND)

$0000\ 0000_2 \Rightarrow 0$

$15 \rightarrow 0000\ 1111$

$1 \rightarrow 0000\ 0001$

AND $0000\ 0001 \Rightarrow 1$

$6 \rightarrow 0000\ 0110$

$\& \quad 0000\ 0001$

$0000\ 0000$

you could go with this

(or)

this !

$(x \& (x-1) == 0)$

$(x \& (x-1))$

return $(x > 0) \&\& !(x \& (x-1))$

(3) Check if 2 numbers have opposite signs.

(int x, int y) (8, -7) => true

(5, 5) => false

(-2, -1) => false

+6 → 0110₂

-6 → 1010₂

7 → 0111₂

-7 → 1001₂

0 ^ 1 → 1 (true)

1 ^ 0 → 1 (true)

0 ^ 0 → 0 (false)

1 ^ 1 → 0 false

7 → 0111

-6 → 1010

XOR 1101 → some negative number

6 → 0110₂

7 → 0111₂

XOR 0001 → some positive number

-6 1010

-7 1001

0011

↓

some positive number.

Check if $x \wedge y < 0$

Yes → They have opposite signs

No → They have the same sign.

---

④ If the i^{th} bit is 1 or 0

0 0 0 1 0 1 1

1 ↑ ↑ ↑

7 2 1 0

1 → 0 0 0 0 0 0 0 1 → 0^{th} bit

This work perfectly.

1 << 4 ←

0 0 0 1 0 0 0 0

17 - 0 0 0 1 0 0 0 1

0 0 0 1 0 0 0 0

AND 0 0 0 1 0 0 0 0

return (1 << i) & number