Numbers

0, 1, 10…

- 3 + 2
- 5 - 2
- 4 * 4
- 16 / 8

- 16 % 2

%

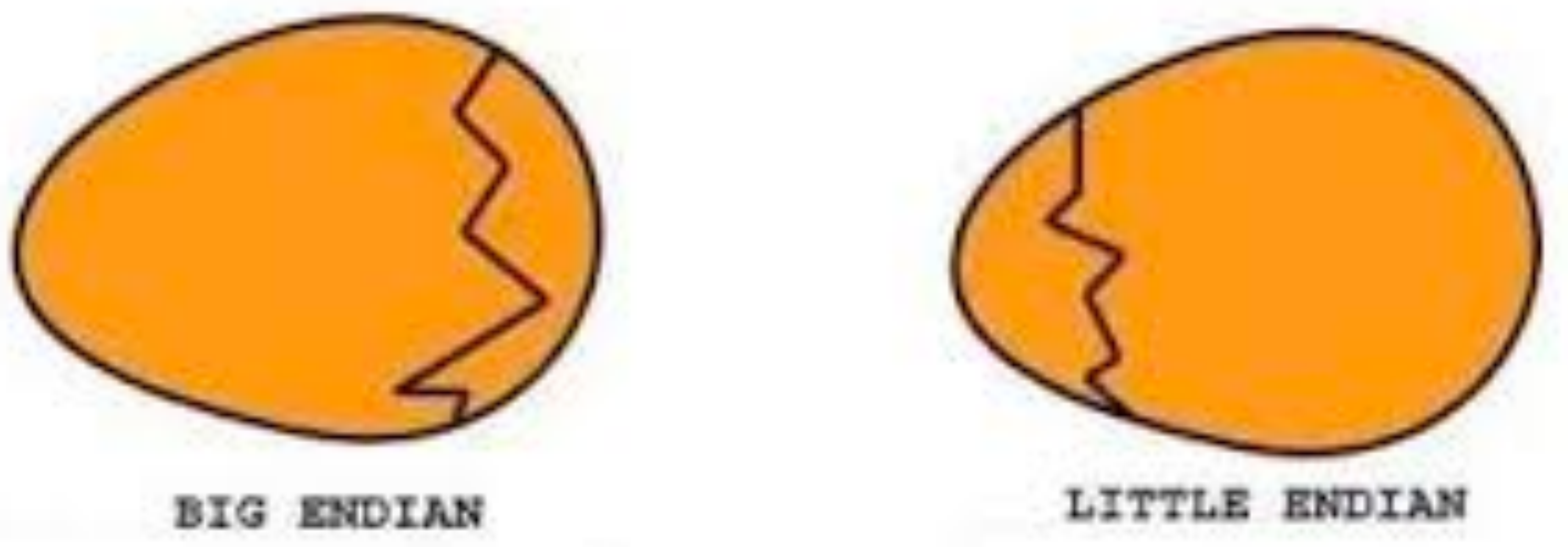
- 3 & 2
- 5 | 2
- 4 ^ 4
- ~16

11101000 00011110 11010100 10010110 01000010
10101111 11000100 10001010 11011101 10011111
11101111 11011111
01101101 There are only 10 types of 00000100
00100000 people in the world: those 10001111
00110010 who understand binary, and 11111111
01110010 those who do not. 01011100
00110111 00110001
11000110 10100011 01101010 10010110 11000111
00001000 10010001 00101011 11011101 11100101
01101110 00101011 00011100 10011111 01101001
01010111 00100111 11111010 11111000 10111011
01111000 11000010 01000110 00100000 01111101

Using {0, 1} to Represent [0, 255]

Physical State	OFF	ON	ON	OFF	OFF	ON	ON	OFF
Binary Notation	0	1	1	0	0	1	1	0
Order of Magnitude	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal Value	128	64	32	16	8	4	2	1
Applicable Value	0	64	32	0	0	4	2	0
Total Decimal Value	102 = 64 + 32 + 4 + 2							

Big Endian vs. Little Endian


BIG ENDIAN
LITTLE ENDIAN

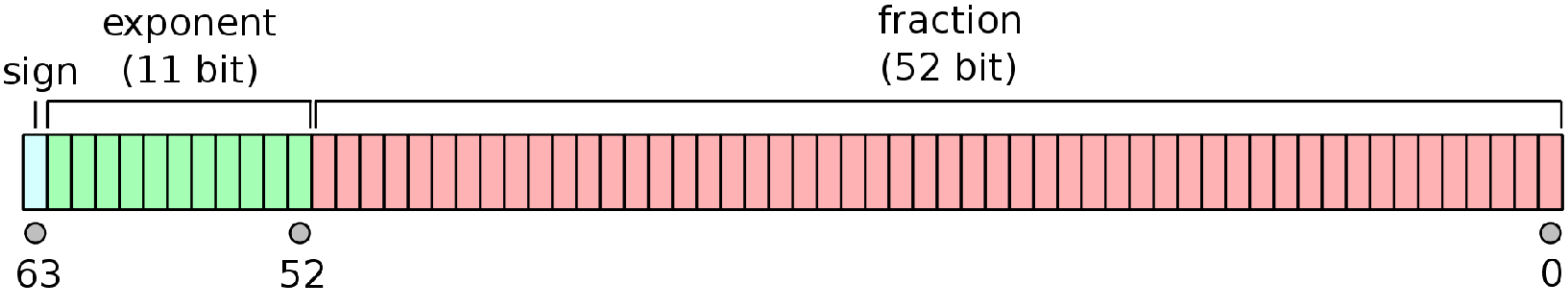
72 vs. 27

Endian-ness

- Refers to the order in which a sequence of bytes are stored in memory
- Big endian: the most significant (largest) byte is stored in the first (lowest) memory address
- Little endian: the least significant byte is stored in the first memory address


-1/2

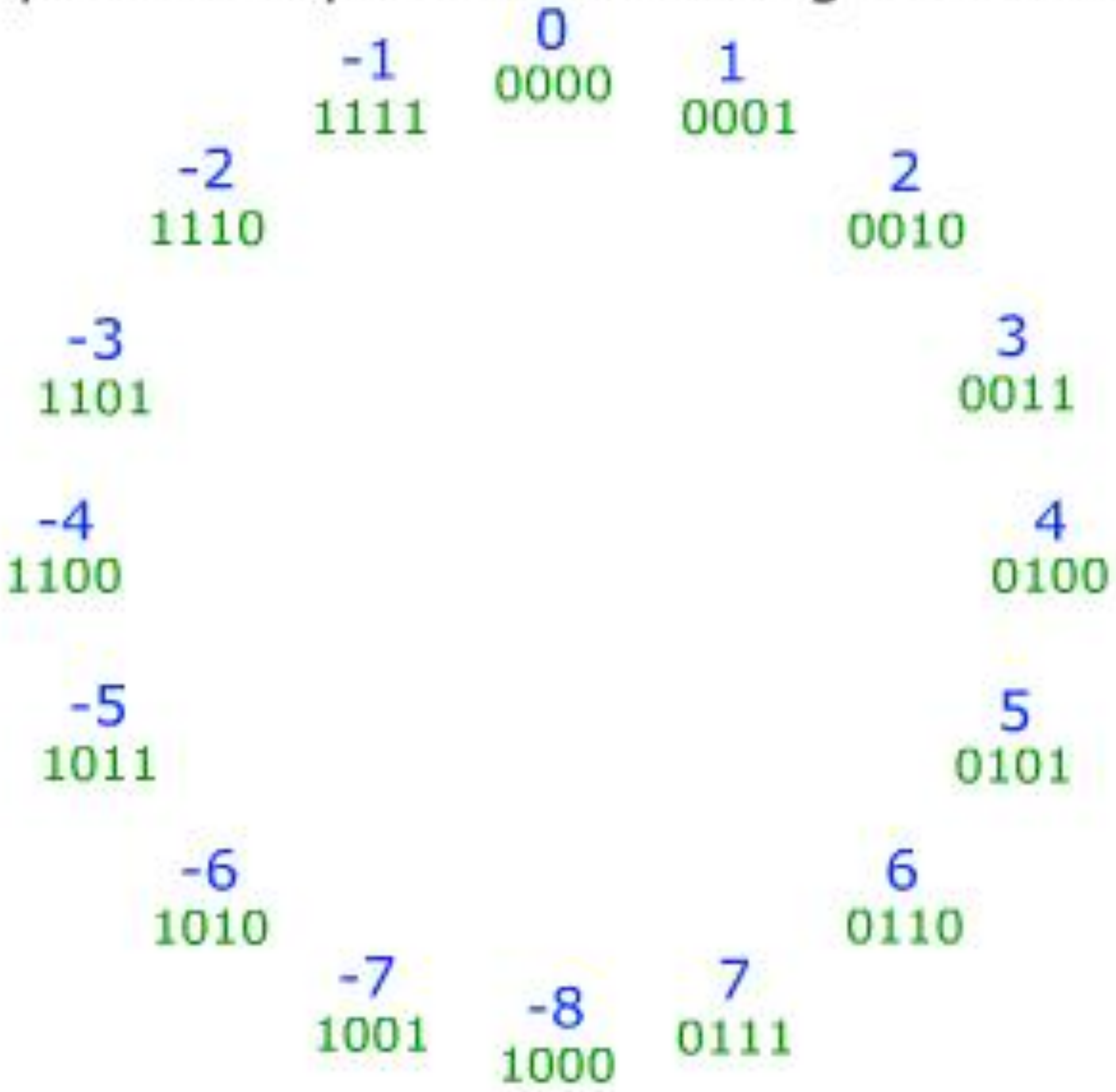
IEEE 754: Floating-Point Signed Double



- 64 bits total — "*double*" the previous 32-bit word size
- 52 bits for the *fraction* of the *significand* (with an implicit “1” bit)
  - Decimal example: in 1.5204 x 10^4, the *fraction* is 5204
- 11 bits for the *exponent* — where the *bicimal point* should *float* to (order of mag.)
  - Many special rules – encoding patterns for ± 0, ± infinity, NaN, etc.
- 1 bit for *sign* (0 positive, 1 negative)

Two's Complement

Two's Complement representation using 4 bit binary strings



Bitwise Operators

&

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

- `x & 0    ==  0`
- `x & -1   ==  x`

I

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

- `x | 0   ==  x`
- `x | -1  ==  -1.`

~

a	NOT a
0	1
1	0

- `~x    ==    -(x + 1)`

^

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

- `x ^ 0   ==  x`
- `x ^ -1  ==  ~x`

<<

```
     9 : 00000000000000000000000000001001
                  --------------------------------
9 << 2 : 00000000000000000000000000100100     = 36
```

- `x << y  ==  x * (2 ** y)`

>>

```
     9: 00000000000000000000000000001001
                  --------------------------------
9 >> 2: 00000000000000000000000000000010       = 2
```

```
     -9: 11111111111111111111111111110111
                   --------------------------------
-9 >> 2: 11111111111111111111111111111101      = -3
```

>>>

```
     9 : 00000000000000000000000000001001
                   ----------------
9 >>> 2:  00000000000000000000000000000010      = 2
```

```
-9          : 11111111111111111111111111110111
               ----------------
-9 >>> 2 : 00111111111111111111111111111101 = 1073741821
```

So what?