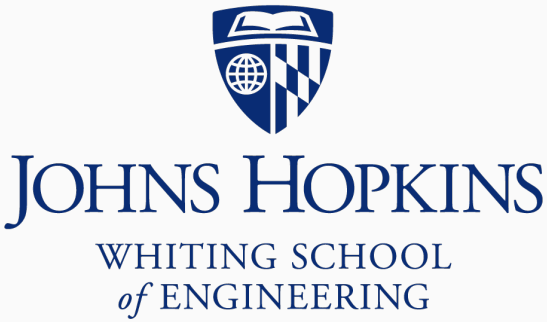
Numeric types

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



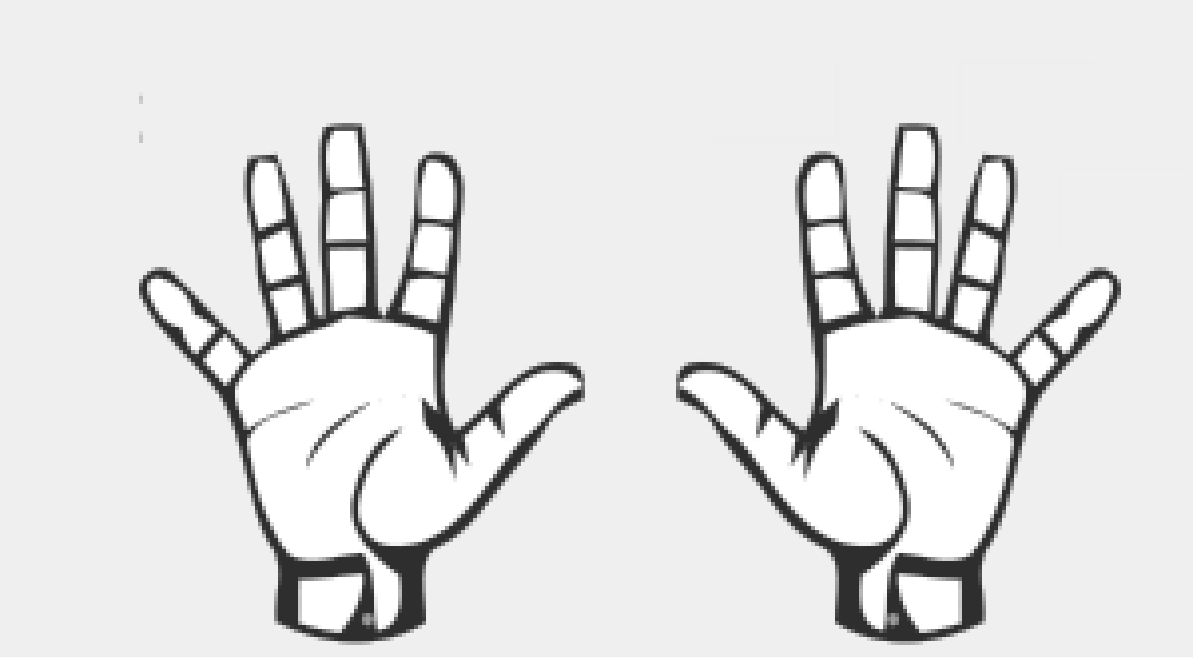
Source markdown available at github.com/BenLangmead/c-cpp-notes

Numeric types

In computers, all data are stored in binary

Binary is the number system where each digit is a power of 2

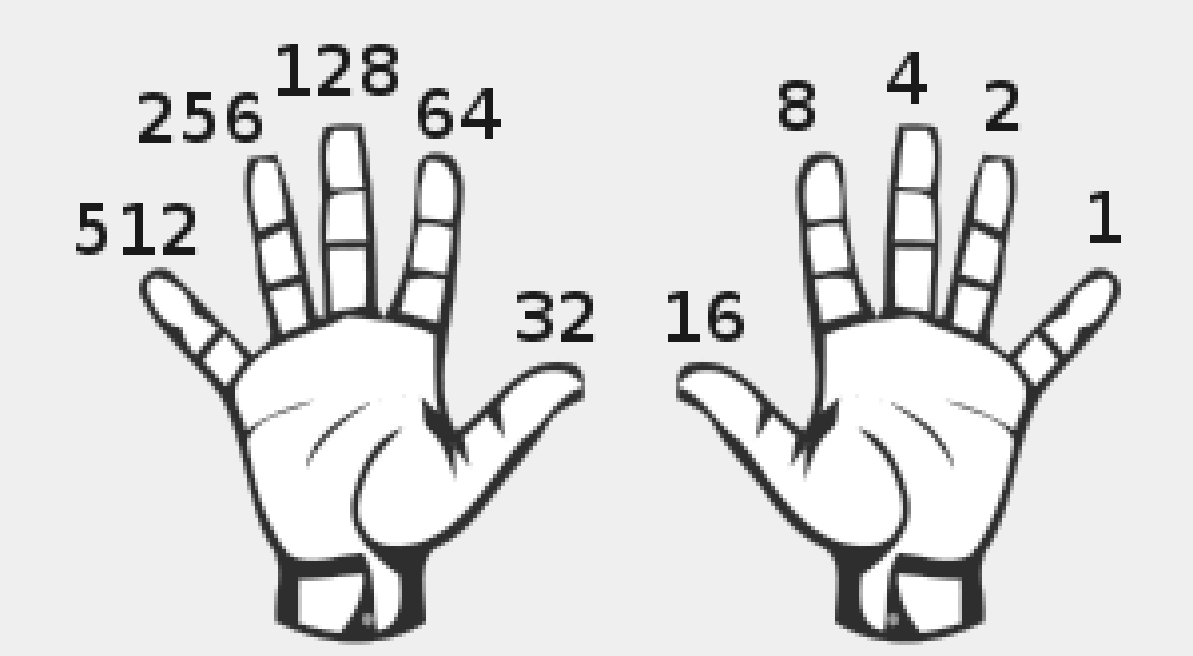
We are used to powers of 10 (decimal)



https://biscitmx.com/category/unplugged/

Bits and binary

If we used our fingers to count in binary, we could count to $2^{10} - 1 = 1023$



https://biscitmx.com/category/unplugged/

Bits and binary

Integer is like an array of bits, but we can't use `[]` for individual bits

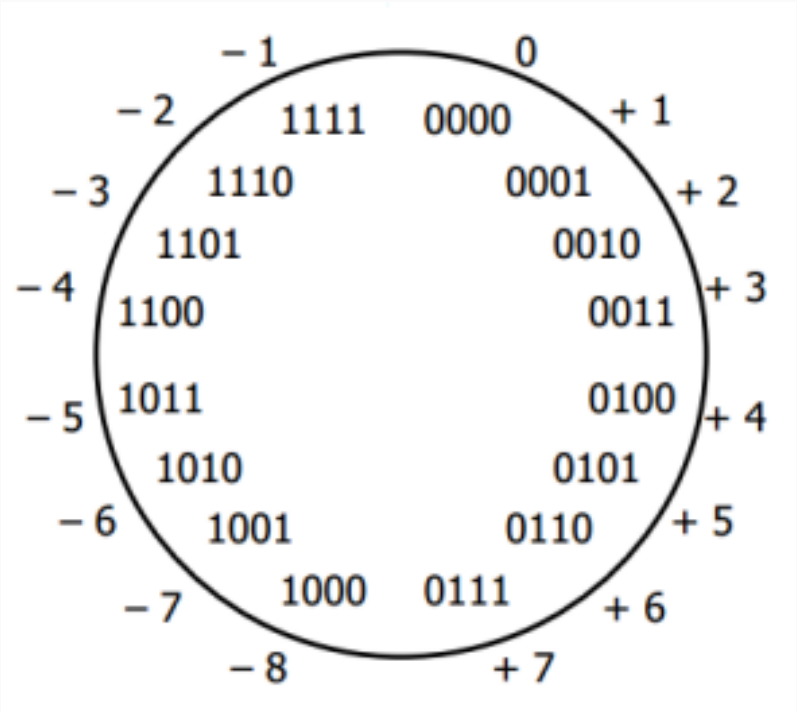
Binary:	0	0	1	1	0	1	0	1
Place value:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$$2^5 + 2^4 + 2^2 + 2^0 = 32 + 16 + 4 + 1 = 53$$

- Instead, we need *bitwise operators*, discussed later

Bits and binary

C integers use "two's complement" representation for signed integers. Illustration with 4 bits:



http://www.bogotobogo.com/cplusplus/quiz_bit_manipulation.php

When a two's complement number overflows, it wraps around to a negative number

Bits and binary

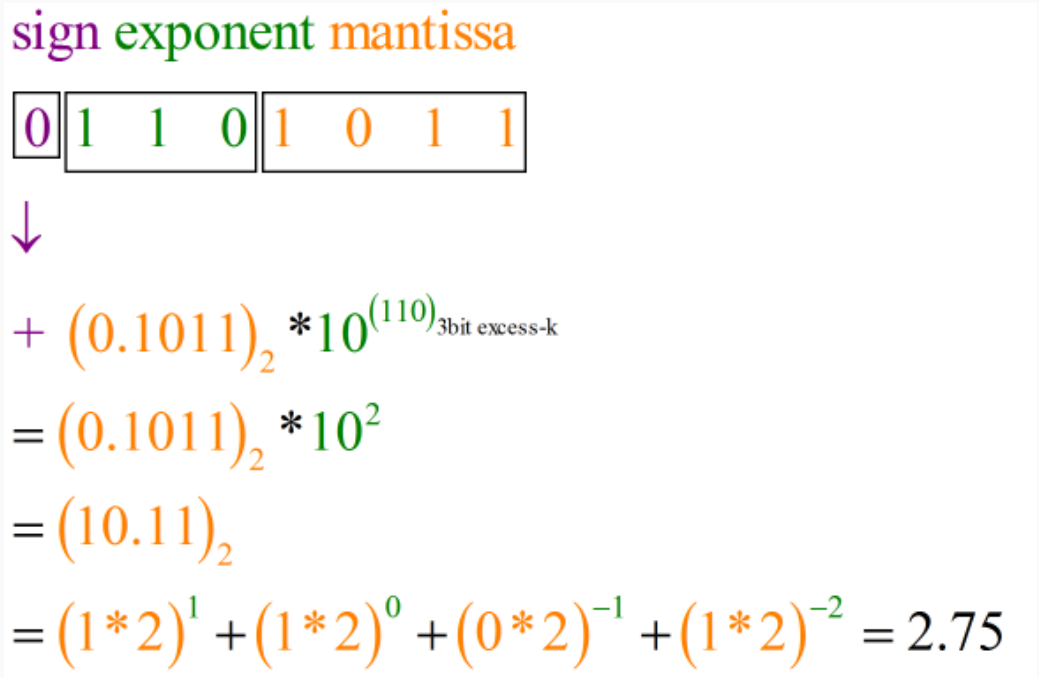
```c
#include <stdio.h>

int main() {
    int i = 2147483647;
    int i_plus_1 = i + 1;
    printf("i = %d, i+1 = %d\n", i, i_plus_1);
    return 0;
}
```

```
$ gcc -c overflow.c -std=c99 -pedantic -Wall -Wextra
$ gcc -o overflow overflow.o
$ ./overflow
i = 2147483647, i+1 = -2147483648
```

Bits and binary

Floating point numbers use their bits to store a few different things:

- Sign: 1 bit, positive or negative
- Exponent
- Mantissa



https://aha.betterexplained.com/t/8bit-floating-point-representation/1121

Bits and binary

Integer and floating-point representations differ:

- Integers have limited range, but integers in the range can be represented precisely. Floating point have limited range and can only approximate most numbers in the range.
- Integers use all available bits for two's-complement representation. Floating point have separate sets of bits for sign, exponent and mantissa.

Bits and binary

`float a = 1` or `int i = 3.0`, it's not as simple as copying bits

When going from integer types to to `float` (or `double`), we are getting an approximation, not the exact integer