Data Types and Number Representation

How do computers store numbers?

- All data in a computer is stored as binary using series of 1's and 0's
- Each binary digit is called a **bit**
- In the C programming language, each variable has a **fixed number of bits** that is can use to represent different values

Binary Representation

- How many numbers can we make with:
  - one bit?
    - 0, 1
    - 2 possible values
  - two bits?
    - 00, 01, 10, 11
    - 4 possible values
  - each time we add a digit we increase our value range by a power of 2
- n digits = 2^n possible values

Binary Representation

Decimal Number	Binary Representation		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Binary Representation

Decimal Number	Binary Representation		
-4	1	0	0
-3	1	0	1
-2	1	1	0
-1	1	1	1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1

Converting to two's complement:

- Invert all the digits
- Add 1
- Example
  - 3 in binary is 011
  - Inverting the digits, we have 100
  - Adding 1 to 100 we get 101 (-3)

Binary Representation

- How many numbers can we make with:
  - three bits?
    - 2^3=8 possible values
    - Range of values if we represent negative number
      - -4 to 3 (-2^2 to 2^2-1)
    - Range of values if we represent only non-negative numbers
      - 0 to 7 (0 to 2^3-1)
- In the C programming language, each variable has a **fixed number of bits** that is can use to represent different values

Storing Binary Data

- We group bits together in units of 8 called **bytes**.
- A byte is the smallest unit of data we can access from memory (RAM)
- **Data types** in C are used to representvalues
  - Data types have a certain number of bits available for storage
  - The amount of storage is always in groups of 8 bits (1 byte)

Int Data Type

- Stores positive and negative integer values
- According to the C standard must be **at least** 16 bits (2 bytes)
  - Most modern computers use 32 bits (4 bytes)
- Assuming we have 32 bits for an int, that is 2^{32} different numbers
  - Can be positive, negative, or zero
  - $2^{31} - 1$ positive numbers, 2^{31} negative numbers, and 0
    - Range: -2^{31} to $2^{31}-1$ (-2,147,483,648 to 2,147,483,647)
- A 32-bit unsigned int only stores positive numbers 0 to $2^{32} - 1$

Integer Overflow

Decimal Number	Binary Representation		
-4	1	0	0
-3	1	0	1
-2	1	1	0
-1	1	1	1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1

- Largest possible number = 3
  - Adding 1 to 3 (011) will result in
    - 100 (-4)

- Adding 1 to -1 (111) will result in
  - 1000
  - We have only 3 bit available to us
  - 4^{th} bit of the result will be truncated
  - Final result will be 000 (0)

Underflow

Decimal Number	Binary Representation		
-4	1	0	0
-3	1	0	1
-2	1	1	0
-1	1	1	1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1

- Smallest possible number = -4
  - Subtracting 1 from -4 (100) will result in
    - 011 (3)

Unsigned Integer

Decimal Number	Binary Representation		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

- Sometimes we know that the value to be stored in an integer variable will always be positive
- For example, when it's being used to only count things
- In that case, we can declare the integer variable to be **unsigned**

Float Data Type

- Represents real numbers
  - Values with decimal precision
- Uses 32 bits of storage (4 bytes)
- Range is wider than an int
  - Max value is ~$3.4*10^{38}$
- While there are infinite values in the range only 2^{32} values can be represented
  - This leads to approximation (rounding)

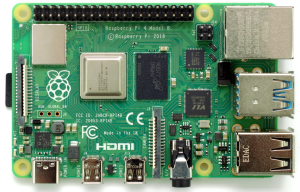
Double Data Type

- Like float and stores real numbers (decimal values)
- Unlike float, the storage is doubled to 64 bits (8 bytes)
- Max value is~$1.8*10^{308}$
- Much greater precision
- Usually better to use double than float
  - Unless you are storing a very large number of decimal values or have limited resources

Data Types on Different Platforms

Mac Laptop
(64-bit OS)

- _Bool: 8
- char: 8
- short int: 16
- int: 32
- **long int: 64**
- long long int: 64
- float: 32
- double: 64



Raspberry Pi
(32-bit OS)

- _Bool: 8
- char: 8
- short int: 16
- int: 32
- long int: 32
- long long int: 64
- float: 32
- double: 64

Arduino (8-bit
microcontroller)

- _Bool: 8
- char: 8
- short int: 16
- **int: 16**
- long int: 32
- long long int: 64
- float: 32
- **double: NA**

Data Types

Type	Declaration	Format Specifier
Integer	int x	%i
Unsigned Integer	unsigned int x	%u
Long	long int x	%li
Unsigned Long	unsigned long x	%ul
Float	float x	%f
Double	double x	%lf