

# CSC 211: Computer Programming

## Number Systems, Further look into DataTypes

Michael Conti

Department of Computer Science and Statistics  
University of Rhode Island

Fall 2022



Original design and development by Dr. Marco Alvarez

# Administrative Notes

## Administrative notes

---

- **A01 Due 10/03**

# Number Systems

# Number systems

- A way to represent numbers
  - ✓ numbers are expressed in a certain **base**
- Why study number systems in CS?
  - ✓ to understand data representation
- Examples of number systems
  - ✓ binary
  - ✓ decimal
  - ✓ octal
  - ✓ hexadecimal

5

# Positional number systems

assuming base **b**:

$$\dots d_2 b^2 + d_1 b^1 + d_0 b^0 + d_{-1} b^{-1} + d_{-2} b^{-2} \dots$$

$$43.23 = 4 \cdot 10^1 + 3 \cdot 10^0 + 2 \cdot 10^{-1} + 3 \cdot 10^{-2}$$

6

# Decimal number system

- Base 10
- Symbols

0 1 2 3 4 5 6 7 8 9

$$456 = 4 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0$$

7

# Binary number system

- Base 2
- Symbols

0 1

Most  
Significant Bit

Least  
Significant Bit

$$1010 = (1 \cdot 2^3) + (0 \cdot 2^2) + (1 \cdot 2^1) + (0 \cdot 2^0)$$



8

# Binary to Decimal?

1 0 0 1 0 1 0 0 0

$2^0$	$2^1$	$2^2$	$2^3$	$2^4$	$2^5$	$2^6$	$2^7$	$2^8$
1	2	4	8	16	32	64	128	256

C

Try these ..

1 0 0 1 1 1 0 1

1 1 0 1 0 0 1 1

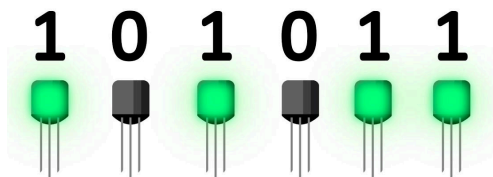
1 1 1 1 1 1 1 1

What is a **bit**? What is a **byte**?

10 |

# Bits and computers

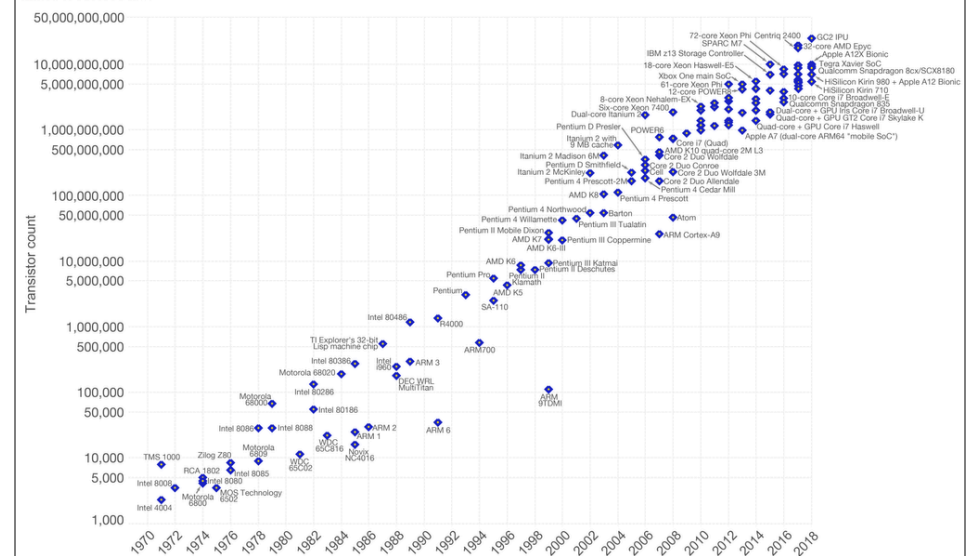
- A bit can only have two values (states)
  - easy to embed into physical devices
- **Transistor**
  - processors have billions of transistors
  - transistors can be switched **on** and **off**



11

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia ([https://en.wikipedia.org/wiki/Transistor\\_count](https://en.wikipedia.org/wiki/Transistor_count))

The data visualization is available at [OurWorldinData.org](https://ourworldindata.org). There you find more visualizations and research on this topic

Licensed under [CC-BY-SA](#) by the author Max Roser.

## Decimal to other bases

- Repeatedly divide by **base**
  - collect remainders
  - output in reverse order

$57_{10}$

✓  $57 / 2 = 28 \text{ R } 1$   
✓  $28 / 2 = 14 \text{ R } 0$   
✓  $14 / 2 = 7 \text{ R } 0$   
✓  $7 / 2 = 3 \text{ R } 1$   
✓  $3 / 2 = 1 \text{ R } 1$   
✓  $1 / 2 = 0 \text{ R } 1$

$111001_2$

13

## Hexadecimal number system

- Base 16
- Symbols

0 1 2 3 4 5 6 7 8 9 A B C D E F

$$4A1C = (4 \cdot 16^3) + (10 \cdot 16^2) + (1 \cdot 16^1) + (12 \cdot 16^0)$$

14

## Hexadecimal to decimal

1 D Bx16

A 0 1 0 F

15

## Binary to hexadecimal

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Bin	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Oct	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17

1 0 0 1 1 1 0 1

1 1 0 1 0 0 1 1

1 1 1 1 1 1 1 1

Humans think in **base 10**. Computers think in **base 2**.  
Humans use **base 16** to easily manipulate data in **base 2**.

16



# Integer literals in C++

```
int d = 42;
int o = 052;
int x = 0x2a;
int X = 0X2A;
int b = 0b101010; // C++14
```

- ✓ **decimal-literal** is a non-zero decimal digit (1, 2, 3, 4, 5, 6, 7, 8, 9), followed by zero or more decimal digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- ✓ **octal-literal** is the digit zero (0) followed by zero or more octal digits (0, 1, 2, 3, 4, 5, 6, 7)
- ✓ **hex-literal** is the character sequence 0x or the character sequence 0X followed by one or more hexadecimal digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, A, b, B, c, C, d, D, e, E, f, F)
- ✓ **binary-literal** is the character sequence 0b or the character sequence 0B followed by one or more binary digits (0, 1)

[https://en.cppreference.com/w/cpp/language/integer\\_literal](https://en.cppreference.com/w/cpp/language/integer_literal)

21

## DISPLAY 2.2 Some Number Types

Type Name	Memory Used	Size Range	Precision
<i>short</i> (also called <i>short int</i> )	2 bytes	-32,768 to 32,767	(not applicable)
<i>int</i>	4 bytes	-2,147,483,648 to 2,147,483,647	(not applicable)
<i>long</i> (also called <i>long int</i> )	4 bytes	-2,147,483,648 to 2,147,483,647	(not applicable)
<i>float</i>	4 bytes	approximately $10^{-38}$ to $10^{38}$	7 digits
<i>double</i>	8 bytes	approximately $10^{-308}$ to $10^{308}$	15 digits
<i>long double</i>	10 bytes	approximately $10^{-4932}$ to $10^{4932}$	19 digits

*These are only sample values to give you a general idea of how the types differ. The values for any of these entries may be different on your system. Precision refers to the number of meaningful digits, including digits in front of the decimal point. The ranges for the types float, double, and long double are the ranges for positive numbers. Negative numbers have a similar range, but with a negative sign in front of each number.*

from: Problem Solving with C++, 10th Edition, Walter Savitch

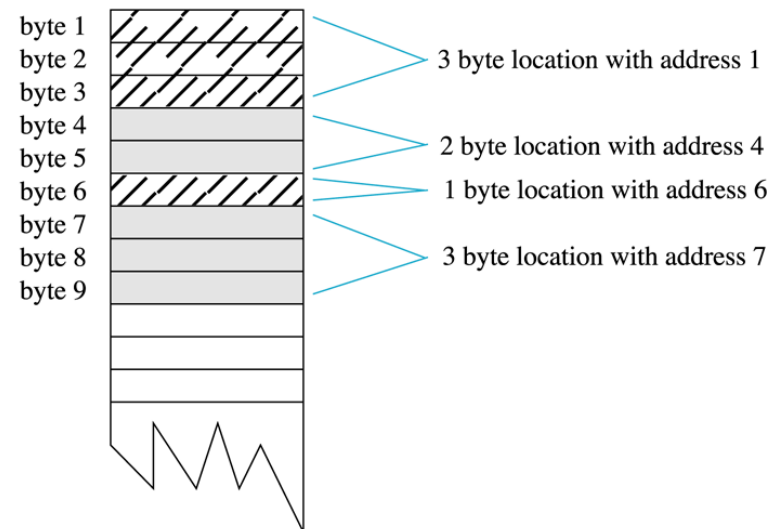
22

Type	Size in bits	Format	Value range	
			Approximate	Exact
character	8	signed		-128 to 127
		unsigned		0 to 255
	16	unsigned		0 to 65535
integer	32	signed	$\pm 3.27 \cdot 10^4$	-32768 to 32767
		unsigned	0 to $6.55 \cdot 10^4$	0 to 65535
	32	signed	$\pm 2.14 \cdot 10^9$	-2,147,483,648 to 2,147,483,647
		unsigned	0 to $4.29 \cdot 10^9$	0 to 4,294,967,295
	64	signed	$\pm 9.22 \cdot 10^{18}$	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
		unsigned	0 to $1.84 \cdot 10^{19}$	0 to 18,446,744,073,709,551,615
floating point	32	IEEE-754	<ul style="list-style-type: none"> <li>min subnormal: <math>\pm 1.401,298,4 \cdot 10^{-45}</math></li> <li>min normal: <math>\pm 1.175,494,3 \cdot 10^{-38}</math></li> <li>max: <math>\pm 3.402,823,4 \cdot 10^{38}</math></li> </ul>	<ul style="list-style-type: none"> <li>min subnormal: <math>\pm 0x1p-149</math></li> <li>min normal: <math>\pm 0x1p-126</math></li> <li>max: <math>\pm 0x1.ffffep+127</math></li> </ul>
	64	IEEE-754	<ul style="list-style-type: none"> <li>min subnormal: <math>\pm 4.940,656,458,412 \cdot 10^{-324}</math></li> <li>min normal: <math>\pm 2.225,073,858,507,201,4 \cdot 10^{-308}</math></li> <li>max: <math>\pm 1.797,693,134,862,315,7 \cdot 10^{308}</math></li> </ul>	<ul style="list-style-type: none"> <li>min subnormal: <math>\pm 0x1p-1074</math></li> <li>min normal: <math>\pm 0x1p-1022</math></li> <li>max: <math>\pm 0x1.ffffffffffffp+1023</math></li> </ul>

<https://en.cppreference.com/w/cpp/language/types>

23

## Memory Locations and Bytes



from: Problem Solving with C++, 10th Edition, Walter Savitch

24