CSC 411

Computer Organization (Spring 2025)
Lecture 2: Number Systems

Christian Esteves, University of Rhode Island

Original slides by Prof. Marco Alvarez, University of Rhode Island

Number systems

- A number system is a method for representing numbers
  - numbers are expressed in a certain **base**
- Importance in **CS**
  - to understand data representation in computers
  - to understand low-level programming and computer architecture
  - to learn how to optimize programs for performance and memory usage
- Common number systems in Computing
  - binary (base 2), decimal (base 10), hexadecimal (base 16)

Number systems

System	Base	Digits
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

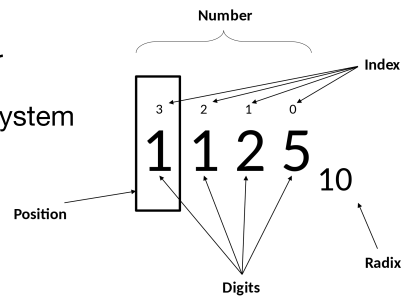
Number systems in Computing

- Binary system
  - directly represent "off" and "on" states in electronic circuits
  - facilitate efficient storage and manipulation of data
  - enables straightforward implementation of logical operations: AND, OR, NOT
- Hexadecimal system
  - compact representation of binary data
  - commonly used in modern computing for memory addresses and color codes

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

Positional notation

- Key concept for understanding all number systems
- The value of a digit depends on:
  - its value
  - its position in the number
  - the base of the number system



https://en.wikipedia.org/wiki/Positional_notation

Conversions to decimal

- Use positional notation
  - changing the base accordingly
- Key points:
  - the rightmost integer digit always has a weight of $b^0 = 1$
  - this method works for any base, making it a versatile tool
- Examples:

101010_2	42
$1A2B_{16}$	6699
137_8	95

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	1	0	0	1	0	1

$$0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$64 + 32 + 4 + 1 = 101$$

Conversions from decimal

- Method
  - divide the number by the base
    - keep track of quotients and remainders
  - repeat steps above until quotient becomes 0
  - read the remainder digits backwards

Number	Result	Remainder
4123	2061	1
2061	1030	1
1030	515	0
515	257	1
257	128	1
128	64	0
64	32	0
32	16	0
16	8	0
8	4	0
4	2	0
2	1	0
1	0	1

1000000011011_2

Practice

- Convert 257 to binary
- Convert 411 to octal
- Convert 1023 to hexadecimal

Binary to hexadecimal

- Method
  - group binary digits into sets of four, starting from the right
    - add leading zeros to the leftmost group if necessary
  - convert each group into its hexadecimal equivalent

Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

A **nibble** is a unit of digital information that consists of four bits. It represents **half of a byte.**

Practice

- Convert to hexadecimal:
  - 10101011_2
  - 110011010101010_2
  - $101010100010010100100011_2$

Practice

- Hexadecimal to binary
  - replace each hexadecimal digit with its 4-digit binary equivalent
- Solve:

$1A2B3C_{16}$

$FA1BFC_{16}$

Integer literals in C/C++

- Decimal literal
  - non-zero decimal digit, followed by zero or more decimal digits
- Octal literal
  - digit zero followed by zero or more octal digits
- Hex literal
  - character sequence 0x or the character sequence 0X followed by one or more hexadecimal digits
- Binary literal
  - character sequence 0b or the character sequence 0B followed by one or more binary digits

What is the output?

```
#include<stdio.h>

int main() {
    int d = 42;
    int o = 052;
    int x = 0x2a;
    int X = 0X2A;
    int b = 0b101010; // C++14

    printf("%d %d %d %d %d", d, o, x, X, b);

    return 0;
}
```