CSC 411

Computer Organization (Fall 2024)
Lecture 2: Number Systems

Prof. Marco Alvarez, University of Rhode Island

Number systems

- A way to express numbers
  - numbers are expressed in a certain **base**
- Why study number systems in **CS**?
  - to understand how computers store and process data
  - to understand low-level programming and computer architecture
  - to learn how to optimize programs for performance
- Examples of number systems
  - binary (2), decimal (10), octal (8), hexadecimal (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

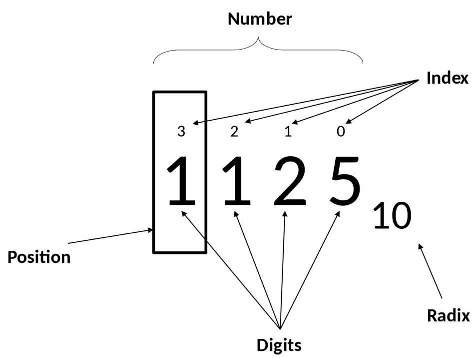
Number systems and computers

- Binary system
  - foundation of digital computers
  - digits represent "off" and "on" states in electronic circuits
  - allow efficient storage and manipulation
  - allow easy logical operations: AND, OR, NOT
- Hexadecimal system
  - shorthand for representing large binary numbers

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 number systems
- The value of a digit depends on both its position and base of the system



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

Practice

20010_{10}

$$20010 = 2 * 10^4 + 0 * 10^3 + 0 * 10^2 + 1 * 10^1 + 0 * 10^0$$

10411_{10}

$$10411 = 1 * 10^4 + 0 * 10^3 + 4 * 10^2 + 1 * 10^1 + 1 * 10^0$$

Conversions to decimal

- Use positional notation changing the base accordingly
- Solve:

101010_2

101010_{16}

101010_8

Conversions from decimal

- Divide the number by the base
  - write down the result and remainder
- Repeat steps above with the result until the result is 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

- Starting from the right, group the binary digits into sets of four binary digits
  - if there are fewer than four digits, add leading zeros
- Assign the corresponding hexadecimal digit to each group of four binary digits
- Combine the assigned hexadecimal digits to get the final hexadecimal representation

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
  - 11001101010101_2
  - $10101010001001010010011_2$

Hexadecimal to binary

- Starting from the left, 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;
}
```