Introduction to Digital Systems

CSE 4205: Digital Logic Design

Aashnan Rahman
Junior Lecturer
Department of Computer Science and Engineering (CSE)
Islamic University of Technology

01

Digital Systems

What are digital systems?

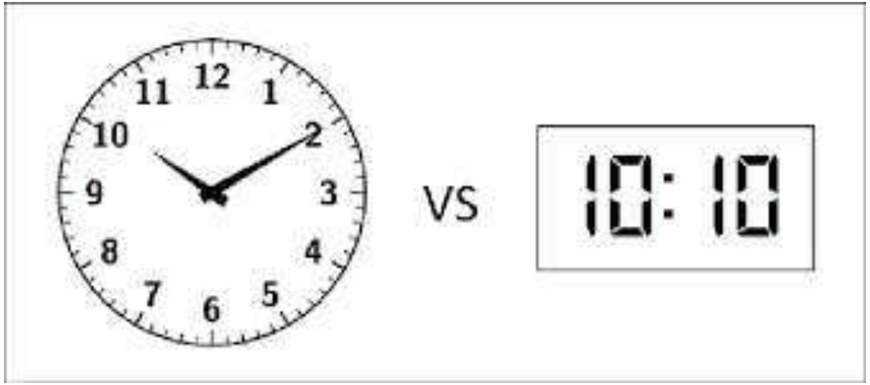
Digital Systems

A digital system is an electronic system that **processes**, **stores**, and **communicates** information using **digital data.**

Digital systems form the backbone of modern technology, playing a critical role in communication, automation, healthcare, education, transportation, and entertainment.

But what does '*Digital*' mean?

The word "digital" refers to **data** or **signals** that are represented by **discrete values.**



Digital data

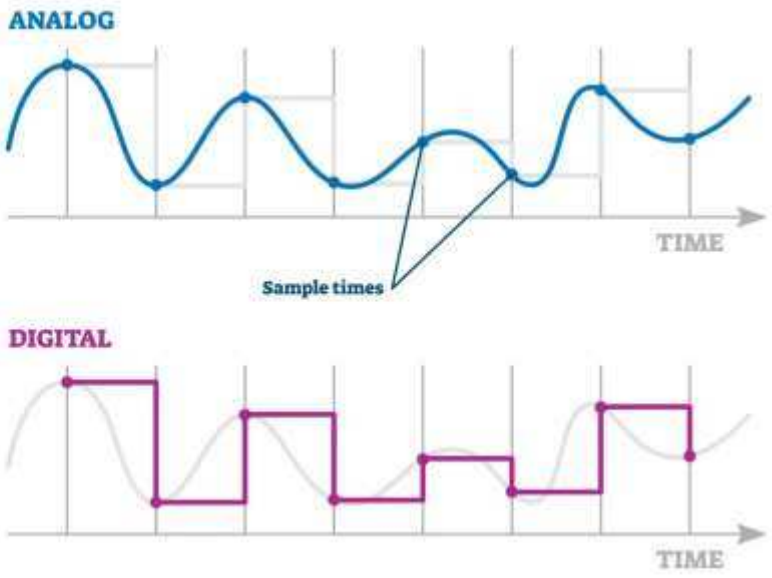
Analog data more closely mirrors **real-world phenomena**, it is **difficult to process** accurately using electronic systems because it is **highly susceptible** to **noise**, **distortion**, and signal **degradation**.

Digital data, on the other hand, is **easier to store**, **process**, and **transmit** with **high precision**. It allows for **error detection** and **correction**, making it more **reliable** over long distances and repeated use. Additionally, digital systems are more flexible and programmable, enabling a wide range of applications.

Digital vs Analog

Differences	Digital	Analog
Representation	Discrete	Continuous with varying magnitude
Signal form	Square waves	Sine waves
Response to noise	Less likely to get affected	Likely to get affected
Processing	Simple and programmable	Complex and less flexible
Storage	Easy and Accurate	Hard to store without loss

Digital vs Analog



Discrete Information

- Voltages – **HIGH** or **LOW**
- Switch States – **ON** or **OFF**
- Time – **24 HOURS, 60 Minutes, 60 Seconds**
- Alphabets – **26**
- **Number Systems**

02

Number Systems

How do we represent values?

Number Systems

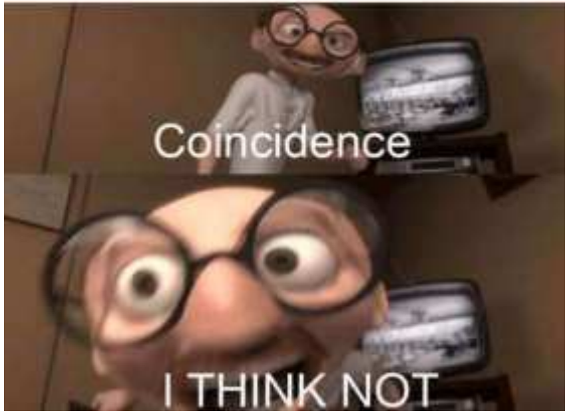
A number system is a way to **represent** and **express** numbers using a **set of symbols** (**digits**) and rules.

The most common number systems are –

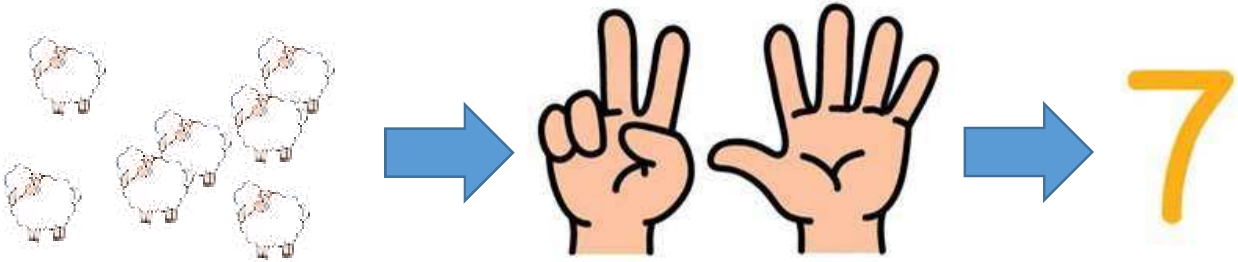
- Decimal
- Binary
- Octal
- Hexadecimal

A Brief History of Numbers

Humans have 10 fingers and the natural number system contains 10 digits as well.



Long before writing, people counted using their hands, that likely shaped our earliest way of counting. The decimal system (digits 0–9) reflects our anatomy. It's not just practical — it's biologically inspired.



A Brief History of Numbers

Different civilizations had different number systems:

1. **Arabs**
   As discussed before, number system with 10 digits.
   Used by Romans, Indians, Chinese and Egyptians as well

2. **Mayans**
   Used 20 digit number system.
   Counted with fingers and toes.

3. **Babylonians**

The Babylonian number system used base 60 (sexagesimal) because of its high divisibility (by 1,2,3,4,5,6,10,12,15,20,30 and 60).

Babylonians tracked the movements of the sun, moon, and planets. They needed a system that allowed precise fractions, especially for time and angular measurements. (1 minute = 60 seconds, 1 hour = 60 minutes)

Common Number Systems

Number 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

Base - number of unique digits (including 0) that the system uses to represent numbers.

Relationship between Base and Number of Digits

As the base (radix) of a number system increases, you need fewer digits to represent the same value.

But it comes with a trade-off, higher the based the more symbols you need, the harder it becomes to remember, write, and process numbers.

03

Number System Conversion

Convert numbers from one base to another

Number Systems Conversion

Decimal

Decimal → Binary

Binary → Decimal

Decimal → Octal

Octal → Decimal

Decimal → Hexadecimal

Hexadecimal → Decimal

Number Systems Conversion

Binary

Binary → Octal

Octal → Binary

Binary → Hexadecimal

Hexadecimal → Binary

Number Systems Conversion

Octal

Octal → Hexadecimal

Hexadecimal → Octal

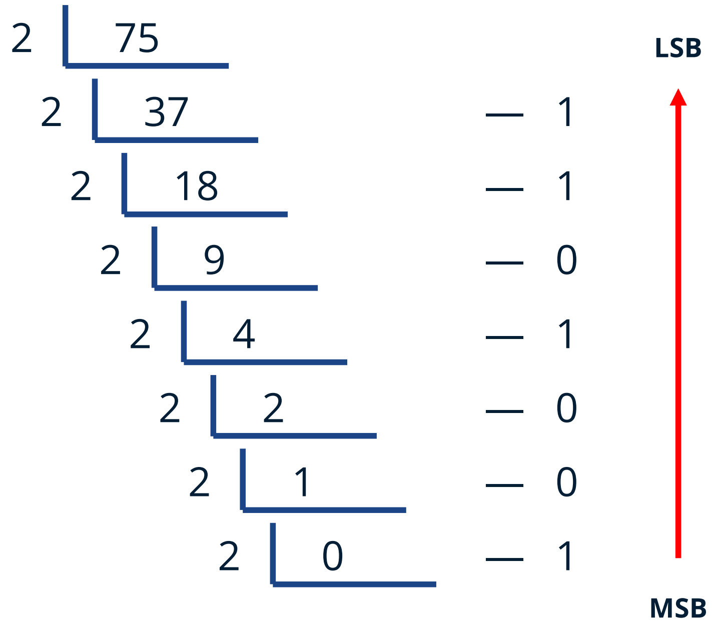
Number Systems Conversion (in short)

Decimal → Any	**Integer:** Keep dividing the number by the target base. Read remainders in reverse (bottom to top). **Fraction:** Multiply the fractional part by the base. Take the whole number part as the next digit. Repeat with the new fractional part.
Any → Decimal	Sum of product each digit by the base raised to its position
Binary → Octal/Hex	Group 3 (octal) or 4 (hex) bits
Octal/Hex → Binary	Convert each digit to 3 (octal) or 4 (hex) bits

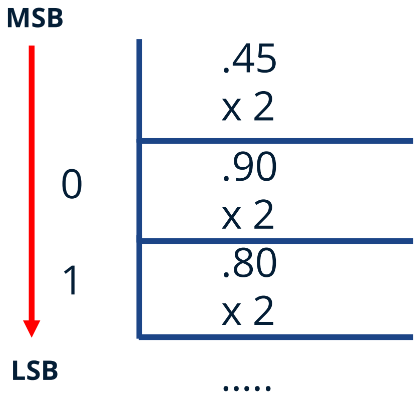
Decimal → Binary

Convert $(75.45)_{10}$ to binary.

Integer



Fraction

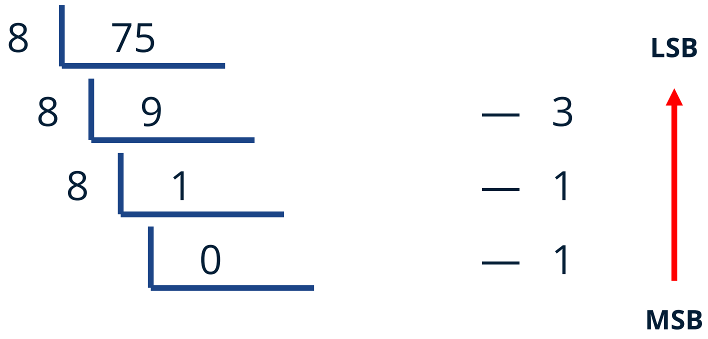


$$(75.45)_{10} = (1001011.01)_2$$

Decimal → Octal

Convert $(75.45)_{10}$ to octal.

Integer


8 | 75

8 | 9 — 3

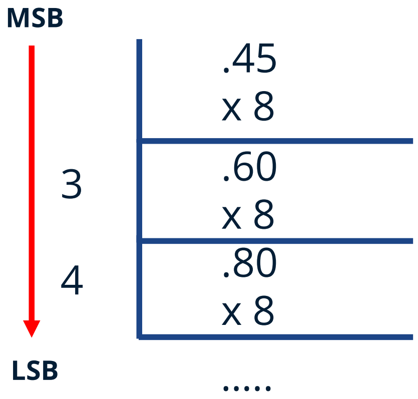
8 | 1 — 1

| 0 — 1

LSB

MSB

Fraction


MSB

.45
x 8

3 .60
x 8

4 .80
x 8

.....

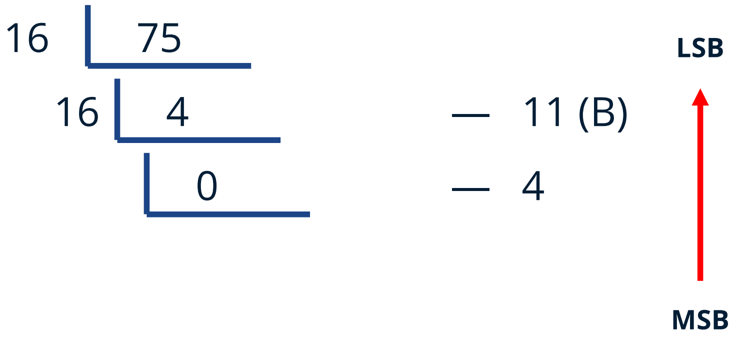
LSB

$(75.45)_{10} = (113.34)_8$

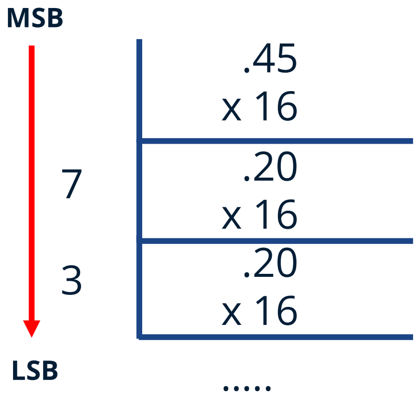
Decimal → Hexadecimal

Convert $(75.45)_{10}$ to hexadecimal.

Integer



Fraction



$$(75.45)_{10} = (4B.733..)_{16}$$

Binary → Decimal

Convert $(1001011.01)_2$ to decimal.

$$(\ \underline{1}\ \ \underline{0}\ \ \underline{0}\ \ \underline{1}\ \ \underline{0}\ \ \underline{1}\ \ \underline{1}\ \ \underline{.}\ \ \underline{0}\ \ \underline{1}\ )$$

Positional Value **6 5 4 3 2 1 0 . -1 -2)**

$(1001011.01)_2 = 1x2^{6} + 0x2^{5} + 0x2^{4} + 1x2^{3} + 0x2^{2} + 1x2^{1} + 1x2^{0} + 0x2^{-1} +1x2^{-2}$

$= 64 + 0 + 0 + 8 + 0 + 2 + 1 + 0.25$

$= (75.25)_{10}$

$$\boxed{(1001011.01)_2 = (75.25)_{10}}$$

Octal → Decimal

Convert $(113.34)_8$ to decimal.

$$(113.34)_8 = 1x8^{2} + 1x8^{1} + 3x8^{0} + 3x8^{-1} + 4x8^{-2}$$

$$= 64 + 8 + 3 + 8 + 0.375 + 0.0625$$

$$= (75.4375)_{10}$$

$$\boxed{(113.34)_8 = (75.4375)_{10}}$$

Hexadecimal → Decimal

Convert $(4B.73)_{16}$ to decimal.

$$(4B.73)_{16} = 4x16^{1} + Bx16^{0} + 7x16^{-1} + 3x16^{-2}$$

$$= 64 + 11 + 0.4375 + 0.1172$$

$$= (75.449)_{10}$$

$$\boxed{(4B.73)_{16} = (75.449)_{10}}$$

Binary → Octal

Convert $(10110101.11110)_2$ to octal.

(010 110 101 . 0 111

~~10~~)

2 6 5 7 4

$(10110101.11110)_2 = (265.74)_8$

Binary → Hexadecimal

Convert $(10110101.11110)_2$ to hexadecimal.

(1011 0101 . 1111 0000

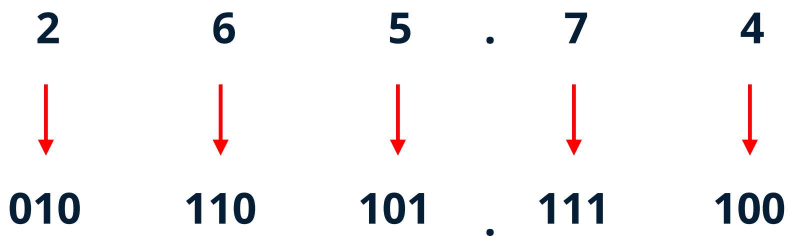
)

11(B) 5 15(F) 0

$(10110101.11110)_2 = (B5.F0)_{16}$

Binary → Octal

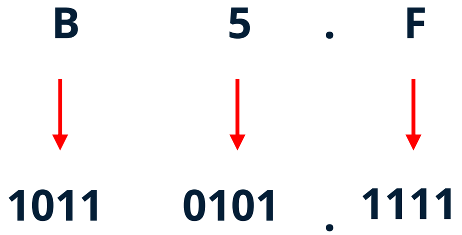
Convert $(265.74)_8$ to binary.



$(265.74)_8=(10110101.111100)_2$

Binary → Octal

Convert $(265.74)_8$ to binary.



$(B5.F0)_{16}=(10110101.1111)_2$

Octal ⟷ Hexadecimal

Hexadecimal ⟷ Octal

Octal ⟷ Binary ⟷ Hexadecimal

Hexadecimal ⟷ Binary⟷ Octal

Question

Find r such that $(121)_r=(144)_8$, where r and 8 are the bases.

04

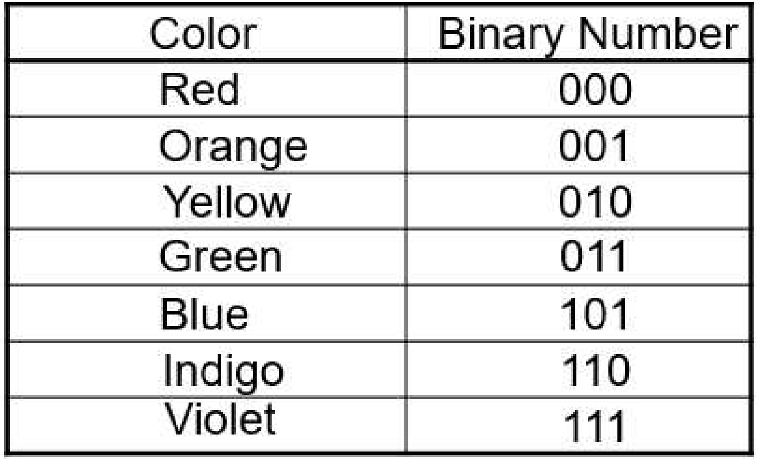
Binary Code

The Core of All Computational Logic

Binary Code

Given n binary digits (called bits), a binary code is a **mapping** from a **set** of **represented elements** to a **subset** of the 2^n binary numbers.

Example:



Color	Binary Number
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111

$$n = \log_2 M$$

Binary Code

Number:
Represents a quantity or value. It’s used for counting, measuring, and performing mathematical operations.
Example: 5, 23, 100.
Process: Conversion

Code:
A representation of information or data, typically in a specific format. It’s used to encode data for storage or communication in computers.
Example: Binary code (101 for 5), ASCII code (01000001 for 'A').
Process: Encoding

Binary Code

There are several Binary coding schemes

Such as :

- Binary Coded Decimal (BCD)
- Excess -3
- Gray

Binary Coded Decimal (BCD) (weighted)

In BCD, each decimal digit (0–9) is represented by its own 4-bit binary equivalent.

Decimal	Binary	BCD
5	101	0101
12	1100	0001 0010
93	1011101	1001 0011

Binary Coded Decimal (BCD)(weighted)

The weights of BCD scheme might vary,

Decimal	8421 BCD	84-2-1 BCD	2421 BCD
7	0111	1001	1101
2	0010	0110	0010
9	1001	1111	1111
6	0110	1010	1100
0	0000	0000	0000
3	0011	0101	0011

5043210(Bi-quinary)(weighted)

Weighted coding scheme consisting of exactly 2 ones.

Example :

Decimal	05-01234
0	10-10000
1	10-01000
2	10-00100
3	10-00010
4	10-00001

Excess-3(unweighted)

Obtained from the corresponding **BCD+3**

Decimal Digit	BCD (8421)	Excess-3 Code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000

Excess-3$_{\text{(unweighted)}}$

Convert $(15)_{10}$ to Excess-3

Step 1: Convert $(15)_{10}$ to BCD.

$(0001 \quad 0101)_{BCD}$

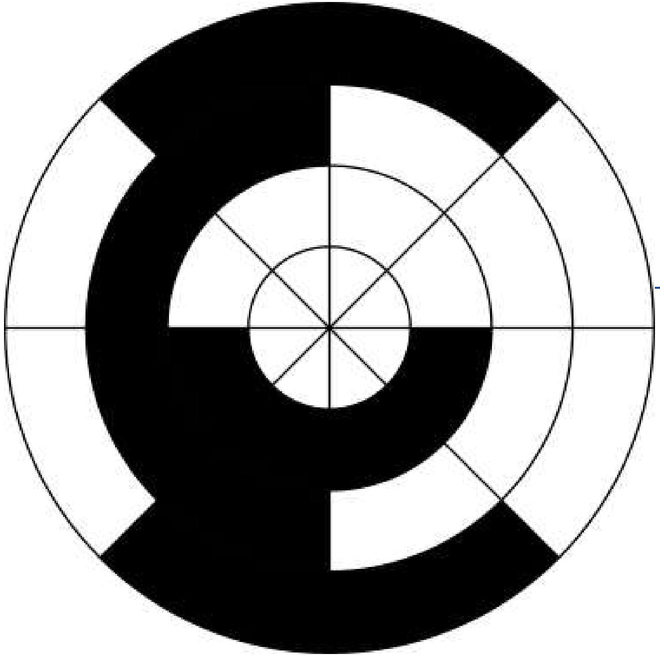
Step 2: Add $(3)_{10}$ or $(0011)_{BCD}$ to each BCD chunk.

$(0100 \quad 1000)_{BCD}$

Excess–3 is self complementing code!!

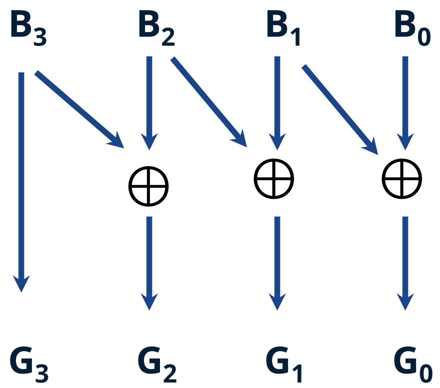
Gray Code(unweighted)

Gray Code, also called **Reflected Binary Code**, is a binary numeral system where two successive values differ in only one bit. This property is known as single-bit change, and it's what makes Gray code special compared to standard binary.

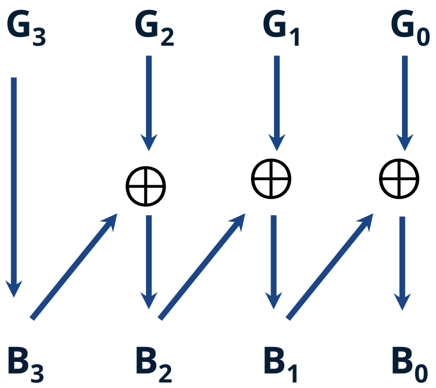


Gray Code(unweighted)

Binary to Gray



Gray to Binary



Gray Code(unweighted)

Decimal	Binary	Gray Code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

05

Alphanumeric Codes

Codes containing numbers and characters

Alphanumeric Codes

An alphanumeric code is a sequence of characters that includes both letters (alphabetic) and numbers (numeric).

- **ASCII (American Standard Code for Information Interchange)**
  Represents 128 characters (7-bit): A–Z, a–z, 0–9, symbols, and control characters
- **EBCDIC (Extended Binary Coded Decimal Interchange Code)**
  IBM-developed 8-bit code used in mainframes
- **Unicode (UTF-8, UTF-16, etc.)**
  A universal character set for all languages, emojis, symbols. Encodes over 140,000 characters

Excess-3
BCD
Gray Code
Binary
Unicode
What the hell is this?
ASCII

Thank You !!

Feel free to ask any questions