



## *Fundamentals of computer systems and data representation*

BURKINA INSTITUTE OF TECHNOLOGY

Computer Science and Entrepreneurship  
(C.S.E)

*Academic year : 2024-2025*

Semester 1

13 février 2025

# Course outline

- ➊ Introduction to computer architecture
- ➋ Information systems
- ➌ Numbering systems
- ➍ Information coding

# *1. Introduction to computer architecture*

# 1. Introduction to computer architecture

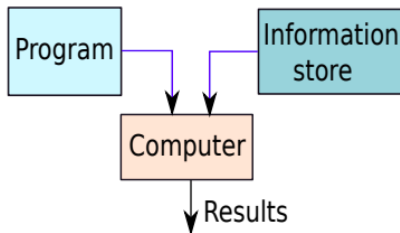
## 1.1. Introduction

- Computer architecture is the study of computers.
- The field of computer architecture is a very fast moving field, and every couple of years there are a plethora of new inventions
- We shall observe that there are two perspectives in computer architecture.
  - the point of view of software applications: **architecture**
  - the point of view of hardware designers: **organization**
- Computer architecture is a beautiful amalgam of software concepts and hardware concepts.

# 1. Introduction to computer architecture

## 1.2.Computer

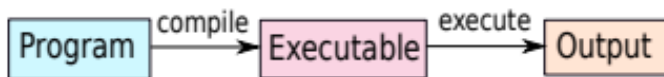
- A computer is a general purpose device that can be programmed to process information, and yield meaningful results.
- There are three important parts to the definition:  
information, program, response



# 1. Introduction to computer architecture

## 1.3.Computers are Dumb Machines

- Computers cannot take very *sophisticated decisions* on their own.
- They are extremely good at **executing programs**.



- 
- Computer vs the brain:

Feature	Computer	Our Brilliant Brain
Intelligence	Dumb	Intelligent
Speed of basic calculations	Ultra-fast	Slow
Can get tired	Never	After some time
Can get bored	Never	Almost always

# 1. Introduction to computer architecture

## 1.4. The instruction set architecture (ISA)

- The number of basic instructions/rudimentary commands that a processor can support have to be finite.
- Basic instructions are: *add, subtract, multiply, logical OR, and logical NOT, etc.*
- Intel and AMD CPUs use the **x86** instruction set
- *The semantics of all the instructions supported by a processor is known as the instruction set architecture (ISA). This includes the semantics of the instructions themselves, along with their operands, and interfaces with peripheral devices.*

# 1. Introduction to computer architecture

## 1.5. Types of ISA

- An ISA needs to be complete, concise, generic, and simple.
- A reduced instruction set computer (**RISC**) implements simple instructions that have a simple and regular structure. The number of instructions is typically a small number (64 to 128): *ARM, IBM PowerPC, HP PA-RISC*.
- A complex instruction set computer (**CISC**) implements complex instructions that are highly irregular, take multiple operands, and implement complex functionalities. Secondly, the number of instructions is large (typically 500+): *Intel x86, VAX*



# 1. Introduction to computer architecture

## 1.6. Turing machine

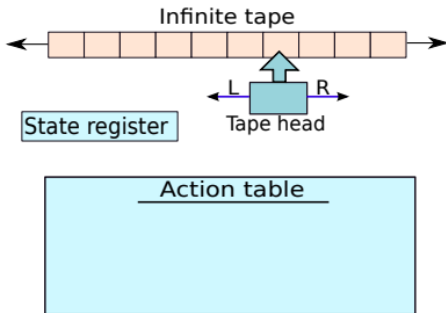
(1). A Turing machine contains an infinite tape that is an array of cells. Each cell can contain a symbol from a finite *alphabet*. There is a special symbol \$ that works as a special marker. A dedicated tape head points to a cell in the infinite tape. There is a small piece of storage to save the current state among a finite set of states. This storage element is called a state register.

(2). In each step, the tape head reads the symbol in the current cell, its current state from the state register, and looks up a table that contains the set of actions for each combination of symbol and state. Each entry in this table specifies three things - whether to move the tape head one step to the left or right, the next state, and the symbol that should be written in the current cell.

# 1. Introduction to computer architecture

## 1.6.Turing machine

(3). Thus, in each step, the tape head can overwrite the value of the cell, change its state in the state register and move to a new cell. The only constraint is that the new cell needs to be to the immediate left or right of the current cell



# 1. Introduction to computer architecture

## 1.7. Resume

- Computers are dumb yet ultra-fast machines.
- Instructions are basic rudimentary commands used to communicate with the processor.
- A computer can execute billions of instructions per second.
- The compiler transforms a user program to a program consisting of basic machine instructions.
- The instruction set architecture (ISA) refers to the semantics of all the instructions supported by a processor.
- The instruction set needs to be complete. It is desirable if it is also concise, generic, and simple.
- Turing Machine

## *2. Information systems*

## 2. Information systems

### 2.1. Introduction

- An information system (IS) is a coordinated set of components that work together to collect, process, store, and disseminate data.
- An IS integrates *hardware, software, data, people, and processes to collect, process, store, and distribute information*
- Two core parts of such systems are **hardware** and **software**.

## 2. Information systems

### 2.2. Hardware Components

Physical devices that enable computing, data storage, and communication.

- **Central Processing Unit (CPU)**

- **Role**: Executes instructions from software; the *brain* of the system.
- **Cores**: Modern CPUs have multiple cores for parallel processing (*e.g.*, *quad-core*, *octa-core*).
- **Clock Speed**: Measured in GHz; determines how fast the CPU processes instructions.
- **Examples**: Intel Core i9, AMD Ryzen, Apple M-series chips.

## 2. Information systems

### 2.2. Hardware Components

- Memory

- **RAM (Random Access Memory)**: called primary memory *volatile*, temporary storage for active data and programs. Faster than storage but loses data when powered off.
- **Cache**: High-speed memory closer to the CPU (L1, L2, L3) for frequently used data.
- **Hard Disk Drives (HDD)**: Magnetic storage for long-term data.
- **Solid-State Drives (SSD)**: Faster, durable flash-based storage.
- **Cloud Storage**: Remote storage accessed via networks (*e.g., AWS S3, Google Drive*).

## 2. Information systems

### 2.2. Hardware Components

- **Input/Output (I/O) Devices**

- **RAM (Input Devices)**: Capture or send data to the system (*e.g., keyboards, mice, scanners, sensors*).
- **Output Devices**: Present processed data (*e.g., monitors, printers, speakers*).
- **Communication Devices**: Enable connectivity (*e.g., network cards, routers, modems*).



## 2. Information systems

### 2.3. Software Components

Programs and instructions that control hardware and perform tasks.

- **System Software or Operating System (OS)**
  - **Role:** Manages hardware resources (*CPU, memory, I/O*), provides user interface (*GUI/CLI*), and runs applications (*Windows, macOS, Linux, Android, iOS*).
  - **Device Drivers:** Software that allows the OS to communicate with hardware (*printer drivers, GPU drivers*).
  - **Communication Devices:** Enable connectivity (*network cards, routers, modems*).

## 2. Information systems

### 2.3. Software Components

User-facing programs for specific tasks.

- **Application Software**

- **General-Purpose**: Broad applications (*e.g., Microsoft Office, web browsers*).
- **Specialized**: Industry-specific tools (*AutoCAD for engineering, IDE for developer, etc.*).
- **Custom Software**: Tailored for unique organizational needs (*enterprise CRM systems*).

## 2. Information systems

### 2.4. Interactions between components

- Data Flow

- ① Input devices (*keyboard*) send data to the CPU via the OS.
- ② CPU processes data using instructions from RAM.
- ③ Results are stored in secondary memory or sent to output devices (*monitor*).

### Role of the OS

- Allocates CPU time, memory, and I/O resources to applications.
- Manages file systems and user permissions.

## 2. Information systems

### 2.5. Resume

**Summary Table**

Component	Examples	Role
<b>CPU</b>	Intel Core i7, ARM Cortex-A78	Executes software instructions.
<b>RAM</b>	DDR4, DDR5	Temporarily holds active data.
<b>Storage</b>	SSD, HDD, cloud storage	Long-term data retention.
<b>OS</b>	Windows 11, Ubuntu, iOS	Manages hardware and software resources.
<b>Applications</b>	Excel, Photoshop, Slack	Performs user-specific tasks.

### *3. Numbering systems*

## 3. Numbering systems

### 3.1. Introduction

**Number System** is a method of representing numbers on the number line with the help of a set of Symbols and rules. These symbols range from 0-9 and are termed as digits.

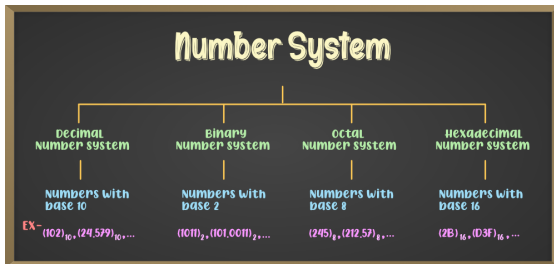
**Number system in Maths** Number system in Maths is a writing system for expressing numbers. It is a mathematical notation for representing numbers of a given set, consistently using digits or other symbols. It allows us to perform arithmetic operations like addition, subtraction, multiplication, and division.

## 3. Numbering systems

### 3.2. Types of Number System

Based on the base value and the number of allowed digits, number systems are of many types. There are four common types.

- Decimal Number System
- Binary Number System
- Octal Number System
- Hexadecimal Number System



## 3. Numbering systems

### 3.2.1 Decimal Number System

Number system with base value 10 is termed as Decimal number system. It uses 10 digits i.e. 0-9 for the creation of numbers.

Here, each digit in the number is at a specific place with place value a product of different powers of 10. The place value is termed from right to left as first place value called units, second to the left as Tens, so on Hundreds, Thousands, etc. Here, units has the place value as  $10^0$ , tens has the place value as  $10^1$ , hundreds as  $10^2$ , thousands as  $10^3$ , and so on.



# 3. Numbering systems

## 3.2.1 Decimal Number System

For example: 10285 has place values as

$$(1 \times 10^4) + (0 \times 10^3) + (2 \times 10^2) + (8 \times 10^1) + (5 \times 10^0)$$

$$1 \times 10000 + 0 \times 1000 + 2 \times 100 + 8 \times 10 + 5 \times 1$$

$$10000 + 0 + 200 + 80 + 5$$

$$10285$$

## 3. Numbering systems

### 3.2.2 Binary Number System

Number System with base value 2 is termed as Binary number system. It uses 2 digits i.e. 0 and 1 for the creation of numbers. The numbers formed using these two digits are termed as Binary Numbers.

Binary number system is very useful in electronic devices and computer systems because it can be easily performed using just two states *ON* and *OFF* i.e. 0 and 1.

Decimal Numbers 0-9 are represented in binary as: 0, 1, 10, 11, 100, 101, 110, 111, 1000, and 1001

# 3. Numbering systems

## 3.2.2 Binary Number System

### Binary Number Table

Decimal Number	Binary Number	Decimal Number	Binary Number
1	001	11	1011
2	010	12	1100
3	011	13	1101
4	100	14	1110
5	101	15	1111
6	110	16	10000
7	111	17	10001
8	1000	18	10010
9	1001	19	10011
10	1010	20	10100

## 3. Numbering systems

### 3.2.2 Binary Number System: Binary to Decimal Conversion

A binary number is converted into a decimal number by multiplying each digit of the binary number by the power of either 1 or 0 to the corresponding power of 2. Let us consider that a binary number has  $n$  digits,  $B = a_{n-1} \dots a_3 a_2 a_1 a_0$ . Now, the corresponding decimal number is given as

$$D = (a_{n-1} \times 2^{n-1}) + \dots + (a_3 \times 2^3) + (a_2 \times 2^2) + (a_1 \times 2^1) + (a_0 \times 2^0)$$

## 3. Numbering systems

### 3.2.2 Binary Number System: Binary to Decimal Conversion

Example 1: Convert  $(10011)_2$  to a decimal number.

*The given binary number is  $(10011)_2$ .*

$$(10011)_2 = (1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

$$= 16 + 0 + 0 + 2 + 1 = (19)_{10}$$

*Hence, the binary number  $(10011)_2$  is expressed as  $(19)_{10}$ .*

## 3. Numbering systems

### 3.2.2 Binary Number System: Binary to Decimal Conversion

Example 2: Convert  $(1010101)_2$  to a decimal number.

*Given Binary Number,  $(1010101)_2$*

$$= (1 \times 2^0) + (0 \times 2^1) + (1 \times 2^2) + (0 \times 2^3) + (1 \times 2^4) + (0 \times 2^5) + (1 \times 2^6)$$

$$= 1 + 0 + 4 + 0 + 16 + 0 + 64$$

$$= (85)_{10}$$

*Thus, Binary Number  $(1010101)_2$  is equal to  $(85)_{10}$  in decimal system.*


## 3. Numbering systems

### 3.2.2 Binary Number System: Decimal to Binary Conversion

A decimal number is converted into a binary number by dividing the given decimal number by 2 continuously until we get the quotient as 1, and we write the numbers from downwards to upwards.

Example: Convert  $(28)_{10}$  into a binary number.

2	28	
2	14	0
2	7	0
2	3	1
	1	1



$$(28)_{10} = (11100)_2$$

### 3. Numbering systems

#### 3.2.2 Binary Number System: Decimal to Binary Conversion

Example: Convert  $(98)_{10}$  into a binary number.

2	98	
2	49	0
2	24	1
2	12	0
2	6	0
2	3	0
	1	1

$(98)_{10} = (1100010)_2$

Thus, Binary Number for  $(98)_{10}$  is equal to  $(1100010)_2$



## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations

We can easily perform various operations on Binary Numbers. Various arithmetic operations on the Binary number include,

- Binary Addition
- Binary Subtraction
- Binary Multiplication
- Binary Division

## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations (+)

The result of the addition of two binary numbers is also a binary number. To obtain the result of the **addition** of two binary numbers, we have to add the digit of the binary numbers by digit. The table added below shows the rule of binary addition.

Binary Number (1)	Binary Number (2)	Addition	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations (+)

Example: Add  $(11011)_2$  and  $(10100)_2$

$$\begin{array}{r} 11011 \\ + 10100 \\ \hline 101111 \end{array}$$

Hence,  $(11011)_2 + (10100)_2 = (101111)_2$

## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations (-)

The result of the subtraction of two binary numbers is also a binary number. To obtain the result of the subtraction of two binary numbers, we have to subtract the digit of the binary numbers by digit. The table added below shows the rule of binary subtraction.

Binary Number (1)	Binary Number (2)	Subtraction	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations (-)

Example: Subtract  $(11010)_2$  and  $(10110)_2$

$$\begin{array}{r} 11010 \\ - 10110 \\ \hline 00100 \end{array}$$

Hence,  $(11010)_2 - (10110)_2 = (00100)_2$

## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations (\*)

The multiplication process of binary numbers is similar to the multiplication of decimal numbers. The rules for multiplying any two binary numbers are given in the table,

Binary Number (1)	Binary Number (2)	Multiplication
0	0	0
0	1	0
1	0	0
1	1	1

## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations (\*)

Example: Multiply  $(1110)_2$  and  $(1001)_2$

$$\begin{array}{r} 1110 \\ \times 1001 \\ \hline 1110 \\ 0000 \\ 0000 \\ 1110 \\ \hline 1111110 \end{array}$$

Thus,  $(1110)_2 \times (1001)_2 = (1111110)_2$

## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations (/)

The division method for binary numbers is similar to that of the decimal number **division** method.

- 1 Start by comparing the *divisor* with the leftmost part of the *dividend*.
- 2 If this portion is less than the divisor, *lower* the next bit until you obtain a number greater than or equal to the divisor.
- 3 Once the portion considered is at least equal to the divisor, you place a **1** in the quotient at the corresponding position and perform the binary *subtraction* of this portion by the divisor.
- 4 If the portion is smaller, you place a **0** in the quotient and set the next bit.



## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations (/)

Example 1: Divide  $(10110)_2$  by  $(11)_2$

$$\begin{array}{r} 111_2 \\ 11_2 \overline{) 10110_2} \\ \underline{101} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \\ 101 \phantom{0} \phantom{0} \phantom{0} \phantom{0} \\ \underline{101} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \\ 100 \phantom{0} \phantom{0} \phantom{0} \\ \underline{100} \phantom{0} \phantom{0} \\ 100 \phantom{0} \phantom{0} \end{array}$$

(first subtraction:  $101_2 - 11_2 = 10_2$ )  
(after lowering the bit)  
(second subtraction:  $101_2 - 11_2 = 10_2$ )  
(after lowering the last bit)  
(third subtraction:  $100_2 - 11_2 = 1_2$ )

## 3. Numbering systems

### 3.2.3 Binary Number System: Arithmetic Operations (/)

Example 2: Divide  $(11110)_2$  by  $(101)_2$

$$\begin{array}{r} 101 \overline{) 11110} \quad (110 \\ \underline{101} \phantom{00} \\ 101 \phantom{00} \\ \underline{101} \phantom{00} \\ 00 \end{array}$$

Ask 

## 3. Numbering systems

### 3.2.4 Binary Number System: Questions

- 1 What is a Binary Number System?
- 2 What is a Bit?
- 3 What is a Nibble?
- 4 What is Binary Value of 10?
- 5 What are Types of Number Systems?
- 6 How to Calculate Binary Numbers?
- 7 How to Add Binary Numbers?

## 3. Numbering systems

### 3.2.5 Octal Number System

Octal Number System is a number system with base 8 as it uses eight symbols (or digits) namely 0, 1, 2, 3, 4, 5, 6, and 7. For example,  $22_8$ ,  $13_8$ ,  $17_8$ , etc. are octal numbers.

This number system is mainly used in computer programming as it is a compact way of representing binary numbers with each octal number corresponding to three binary digits.

The octal number system is a base-8 system using digits 0-7, where each position represents a power of 8. It is commonly used in computing for easy conversion to binary.

## 3. Numbering systems

### 3.2.5 Octal Number System

#### Octal Number System Table

Octal Numbers	Binary Equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

## 3. Numbering systems

### 3.2.5 Octal Number System: Octal to Decimal Conversion

- 1 Step 1: Write the octal number.
- 2 Step 2: Multiply each digit of the given octal number with an increasing power of 8 starting from the rightmost digit.
- 3 Step 3: Sum all the products obtained in step 2.

Example 1: Represent  $123_8$  as a Decimal Number.

$$123_8 = 1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0$$

$$\Rightarrow 123_8 = 1 \times 64 + 2 \times 8 + 3 \times 1$$

$$\Rightarrow 123_8 = 64 + 16 + 3$$

$$\Rightarrow 123_8 = 83_{10}$$

Hence  $83_{10}$  is decimal representation of  $123_8$ .

## 3. Numbering systems

### 3.2.5 Octal Number System: Decimal to Octal Conversion

*Step 1: Divide the given decimal number by 8.*

*Step 2: Write down the quotient and remainder obtained.*

*Step 3: Divide the quotient obtained by 8.*

*Step 4: Repeat step 2 and step 3 until the quotient becomes 0.*

*Step 5: Write the obtained remainder in reverse order.*

Example 2: Represent  $164_{10}$  as Octal Number.

$164/8$ , Quotient = 20 and Remainder = 4

$20/8$ , Quotient = 2 and Remainder = 4

$2/8$ , Quotient = 0 and Remainder = 2

Now, By writing obtained remainders in reverse order we get, 244.

Hence  $244_8$  is octal representation of  $164_{10}$

## 3. Numbering systems

### 3.2.5 Octal Number System: Octal to Hexadecimal Conversion

- A hexadecimal number system has a base 16 and it is an alphanumeric number system consisting of digits from 0 to 9 and alphabets from A to F.
- To convert an octal number to a hexadecimal number: First convert the octal number to the decimal number;
- Then convert the obtained decimal number to the hexadecimal number.



## 3. Numbering systems

### 3.2.5 Octal Number System: Octal to Hexadecimal Conversion

Example 2: Represent  $174_8$  to a hexadecimal number.

**Step 1: Convert  $174_8$  to decimal**

$$174_8 = 1 \times 8^2 + 7 \times 8^1 + 4 \times 8^0$$

$$174_8 = 1 \times 64 + 7 \times 8 + 4 \times 1$$

$$174_8 = 64 + 56 + 4 = 124$$

$$\text{We get } 174_8 = 124_{10}$$

**Step 2: Covert  $124_{10}$  to hexadecimal**

$$124/16, \text{ Quotient} = 7, \text{ Remainder} = 12$$

$$7/16, \text{ Quotient} = 0, \text{ Remainder} = 7$$

Converting the obtained remainders to corresponding hexadecimal number and writing it in reverse order we get:

$$124_{10} = 7C_{16}$$

$$\text{Hence we get } 174_8 = 7C_{16}$$

## 3. Numbering systems

### 3.2.6 Hexadecimal Number System

Hexadecimal is a number system combining *hexa* for 6 and "deci" for 10. It uses 16 digits: 0 to 9 and A to F, where A stands for 10, B for 11, and so on. Similar to the regular decimal system, it counts up to F instead of stopping at 9. Each digit in hexadecimal has a weight 16 times greater than the previous one, following a positional number system.

When converting to another system, we multiply each digit by the power of 16 based on its position. For example, in the number 7B3, 7 is multiplied by 16 squared, B by 16 to the power of 1, and 3 by 16 to the power of 0.

## 3. Numbering systems

### 3.2.6 Hexadecimal Number System

#### Hexadecimal Number System Table

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Convert  $(A7B)_{16}$  to decimal.

#### Hexadecimal Number System Table

$$(A7B)_{16} = A \times 16^2 + 7 \times 16^1 + B \times 16^0$$

$$\Rightarrow (A7B)_{16} = 10 \times 256 + 7 \times 16 + 11 \times 1 \text{ (convert symbols A and B to their decimal equivalents; } A = 10, B = 11)$$

$$\Rightarrow (A7B)_{16} = 2560 + 112 + 11$$

$$\Rightarrow (A7B)_{16} = 2683$$

Therefore, the decimal equivalent of  $(A7B)_{16}$  is  $(2683)_{10}$ .

## 3. Numbering systems

### 3.2.6 Hexadecimal Number System

Convert  $(1F7)_{16}$  to octal

**Step 1: Convert  $(1F7)_{16}$  to decimal using the powers of 16:**

$$(1F7)_{16} = 1 \times 16^2 + 15 \times 16^1 + 7 \times 16^0$$

$$\Rightarrow (1F7)_{16} = 1 \times 256 + 15 \times 16 + 7 \times 1$$

$$\Rightarrow (1F7)_{16} = 256 + 240 + 7$$

$$\Rightarrow (1F7)_{16} = (503)_{10}$$

**Step 2: Convert the decimal number  $(503)_{10}$  to octal by dividing it by 8 until the quotient is 0**

$$503 \div 8 = 62 \text{ with a remainder of } 7$$

$$62 \div 8 = 7 \text{ with a remainder of } 6$$

$$7 \div 8 = 0 \text{ with a remainder of } 7$$

Arrange the remainder from bottom to top

Therefore,  $(1F7)_{16}$  is equivalent to  $(767)_8$  in octal

## 4. *Information coding*

## 4. Information coding

### 4.1. What is Encoding?

**Encoding** is the process of converting information into a format that can be easily transmitted, stored, and interpreted by computer systems. It involves the transformation of data into a standardized representation, allowing it to be understood by different devices and applications.

*Encoding* and *decoding* are fundamental processes in programming that involve converting data from one form to another. These processes are critical for data storage, transmission, and communication between different systems or components.

## 4. Information coding

### 4.1. What is Decoding?

**Decoding** is the inverse process of encoding, where the encoded data is transformed back into its original form. It involves reversing the steps performed during encoding to retrieve the original information.

Imagine this: you receive a mysterious package in the mail. You have no idea what's inside, but you're curious to find out. You carefully unwrap the layers of packaging, revealing the hidden contents. That's decoding in a nutshell - uncovering the true meaning behind the encoded data.

## 4. Information coding

### 4.2. Encoding Techniques

**Analog data to Analog signals** The modulation techniques such as Amplitude Modulation, Frequency Modulation and Phase Modulation of analog signals, fall under this category.

**Analog data to Digital signals** This process can be termed as digitization, which is done by Pulse Code Modulation (PCM). Hence, it is nothing but digital modulation. As we have already discussed, sampling and quantization are the important factors in this. Delta Modulation gives a better output than PCM.



## 4. Information coding

### 4.2. Encoding Techniques

**Digital data to Analog signals** The modulation techniques such as Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), Phase Shift Keying (PSK), etc., fall under this category. These will be discussed in subsequent chapters.

**Digital data to Digital signals** These are in this section. There are several ways to map digital data to digital signals.

## 4. Information coding

### 4.3. Types of Encoding

- ASCII Encoding
- Unicode Encoding
- Base64 Encoding
- URL Encoding
- Binary Encoding

## 4. Information coding

### 4.3.1 ASCII Encoding

ASCII (*American Standard Code for Information Interchange*) is a character encoding standard for electronic communication. It represents text in computers, telecommunications equipment, and other devices that use text.

Each character is represented by a numerical code. ASCII encoding is extensively used for creating and editing text files, including HTML, JSON, and source code for programming languages.

# 4. Information coding

## 4.3.1 ASCII Encoding

### ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	.
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[END OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41		81	51	1010001	121	Q					
34	22	100010	42		82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

## 4. Information coding

### 4.3.2 Unicode Encoding

*Unicode* is a computing industry standard designed to consistently encode, represent, and handle text expressed in most of the world's writing systems.

UTF-8, UTF-16, and UTF-32 are common Unicode encoding schemes, each using different byte sizes to represent a character.

UTF-8, is crucial for developing websites and applications that support multiple languages, including those with complex characters like *Chinese*, *Arabic*, and *emoji*. It ensures that text appears correctly regardless of the user's system language settings.

## 4. Information coding

### 4.3.3 Base64 Encoding

**Base64** is used to encode binary data into an ASCII string format by converting it into a *radix-64 representation*. It is commonly used in web applications to encode binary data like images or files for transmission over media that are designed to deal with textual data.

Example:

```

```

## 4. Information coding

### 4.3.4 URL Encoding

*URL encoding*, also known as percent encoding, is used to encode special characters in URLs by replacing them with one or more character triplets that consist of the percent character % followed by two hexadecimal digits.

Example:

Original URL: `https://example.com/query?name=John Doe & age=30`

Encoded URL: `https://example.com/query?name=John%20Doe%20&%20age=30`

## 4. Information coding

### 4.3.5 Binary Encoding

**Binary encoding**, involves converting data into a binary format, which is a base-2 numeral system representation using only two different symbols: typically 0 (zero) and 1 (one).

Binary encoding is foundational to data transmission in computer networks, where data, regardless of its original format (*text*, *images*, *video*), is converted into binary and transmitted over network protocols like TCP/IP.



==*END*==