Multi-disciplinary Course

Information and Communication Technologies (ICT)

Semester-I

MDC-1: Information and Communication Technologies-I

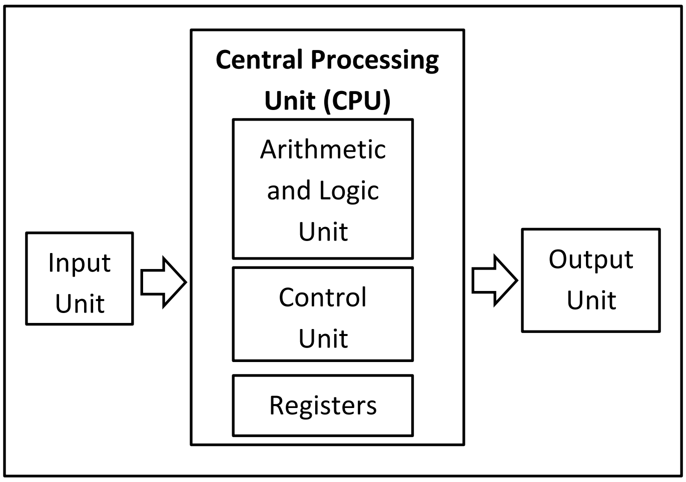
Notes compiled by Dr. Yajnya Sapkota, Department of Physics, Dudhnoi College

---

Chapter-2
Components of Computer System

There are three components of a computer system:

1. Input unit
2. Central Processing Unit (CPU)
3. Output unit



1. **Input Unit:** The input unit allows users to provide data and instructions to the computer through devices like keyboards, mice, scanners, or microphones. It converts user inputs into a machine-readable format for processing.
2. **Output Unit:** The output unit displays or delivers the processed results to the user via devices such as monitors, printers, or speakers. It converts digital data into a human-readable or perceivable form.
3. **Central Processing Unit (CPU):** The CPU, often called the "brain" of the computer, is responsible for executing instructions and processing data. It consists of three main components:
   a) **Arithmetic and Logic Unit (ALU):** Performs mathematical calculations and logical operations.
   b) **Control Unit (CU):** Directs the flow of data and manages the execution of instructions.
   c) **Registers:** Small, high-speed storage locations within the CPU for temporary data and instruction storage.

Functionalities of a computer

A computer performs the following primary functions:

1. **Input:** The computer accepts data and instructions from the user through input devices like a keyboard, mouse, scanner, or microphone.
2. **Processing:** The CPU processes the input data by executing instructions. It performs calculations, comparisons, and logical operations to generate meaningful results.
3. **Storage:** The computer stores data and instructions temporarily in primary memory (RAM) or permanently in secondary storage (HDD, SSD) for future use.
4. **Output:** The processed data is presented to the user through output devices like monitors, printers, or speakers in a readable or perceivable format.
5. **Control:** The computer directs the manner in which input, processing, storage, and output operations are carried out, ensuring smooth coordination among components.

These functionalities make the computer a versatile tool for data processing, communication, and automation.

Basic I/O devices

I/O devices mean Input/Output devices

Input Devices: These devices allow users to provide data and instructions to the computer:

- Keyboard: Used to input text, numbers, and commands.
- Mouse: A pointing device for navigating and selecting items on the screen.
- Scanner: Converts physical documents and images into digital formats.
- Microphone: Captures sound for recording or voice commands.
- Webcam: Captures video and images for communication or recording.

Output Devices: These devices display or deliver the results of the computer's processing:

- Monitor: Displays visual output, such as text, images, and videos.
- Printer: Produces hard copies of digital documents and images.
- Speakers: Output audio, such as music or sound effects.
- Headphones: Provide private audio output for the user.
- Input and output devices are essential for user interaction with the computer system.

Point and draw devices

Point-and-draw devices are input tools that allow users to interact with graphical interfaces by pointing, selecting, or drawing on a screen. Some examples are

- Mouse: A handheld device used to point, click, and drag items on a screen.

- Touchscreen: A display that detects touch gestures, enabling users to interact directly by tapping, swiping, or drawing with fingers or a stylus.
- Stylus: A pen-like device used for precise drawing or writing on touch-enabled devices like tablets and graphic design monitors.
- Trackball: A stationary device with a movable ball for pointing and selecting, often used in compact workspaces.
- Graphics Tablet (Digitizer): A flat input device used with a stylus for freehand drawing or creating detailed graphics.
- Light Pen: A device that detects light signals from a computer screen, allowing users to draw or select by touching the display.

These devices enhance user interaction with computers, especially in creative applications and design tasks.

Data Scanning Device

Data scanning devices are input tools used to convert physical documents, objects, or patterns into digital formats for processing and storage. Some examples are:

- **Flatbed Scanner:** It is used to scan documents, photos, or graphics into digital images. It converts printed text and images into computer-readable formats.
- **Barcode Scanner:** It reads barcode patterns on products for identification, typically used in retail or inventory management. It converts barcode data into alphanumeric information.
- **Optical Character Recognition (OCR) Scanner:** It converts printed or handwritten text into machine-readable text. It is often used for digitizing books or forms.
- **QR Code Scanner:** It reads Quick Response (QR) codes using cameras or specialized devices. It is used for quick data retrieval or web link access.
- **Biometric Scanners:** It captures unique physical characteristics like fingerprints, retina patterns, or facial features for identity verification. It is used in security systems and personal authentication.
- **Handheld Scanner:** It is a portable device used to scan documents or objects by moving it over the surface. It is useful for on-the-go scanning tasks.
- **Image Scanner (3D Scanner):** It captures the physical dimensions of an object to create a 3D digital model. It is commonly used in manufacturing and design industries.

Voice Recognition Devices

Voice recognition devices are input tools that capture and interpret spoken language, converting it into text or commands for computer processing. They are widely used for hands-free interaction and accessibility. Some examples are:

- **Microphone:** Captures audio input, including voice, for processing by software. Commonly used with voice recognition systems for dictation or voice commands.
- **Smart Speakers:** Devices like Amazon Echo or Google Nest, equipped with voice assistants (e.g., Alexa, Google Assistant). Respond to spoken commands for tasks like playing music, controlling smart home devices, or answering questions.
- **Voice-Activated Systems:** Found in smartphones, such as Apple's Siri or Android's Google Assistant. Recognize voice commands to perform tasks like making calls, setting reminders, or searching the web.
- **Dictation Devices:** Specialized devices or software designed to convert spoken words into text documents. Used in transcription, legal, and medical fields.
- **Biometric Voice Recognition Systems:** Use unique voice patterns for identity verification and security. Common in high-security areas or personal device authentication.

Digitizers

A digitizer is an input device used to convert analog information, such as drawings, sketches, or hand movements, into digital form that a computer can process. It is commonly used in design, engineering, and creative fields.

Types of Digitizers:

- **Graphics Tablet:** A flat surface where users draw or write with a stylus. Converts the stylus movements into digital data for creating or editing graphics. Widely used in digital art, animation, and CAD (Computer-Aided Design).
- **Touchscreen Digitizers:** Integrated into devices like smartphones, tablets, and touch-enabled monitors. Allows direct interaction with the display using fingers or stylus.
- **3D Digitizers:** Captures the three-dimensional shape and details of physical objects. Creates digital 3D models for use in design, manufacturing, or virtual reality applications.
- **Acoustic or Sonic Digitizers:** Uses sound waves to track the position of a stylus and convert movements into digital input.
- **Electromagnetic Digitizers:** Detects the position of a stylus using electromagnetic signals. Ensures high precision, making them popular in professional graphics applications.

Applications of Digitizers:

- **Digital Art and Design:** For creating illustrations, animations, and graphic content.
- **Engineering and Architecture:** For drafting blueprints and designing 3D models.
- **Healthcare:** Used in medical imaging for digitizing X-rays and other diagnostic data.
- **Education:** Facilitates interactive teaching with smartboards and touchscreens.

Introduction to number system

The most common number systems are:

1. Decimal Number System (Base 10)

- The decimal system is the standard system used in everyday life, consisting of 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- It is based on powers of 10, where each digit's place value is a power of 10 (e.g., $345 = 3\times10^2 + 4\times10^1 + 5\times10^0$).

2. Binary Number System (Base 2)

- Used primarily in digital computing and electronics.
- It consists of only two digits: 0 and 1.
- Binary numbers are based on powers of 2 (e.g., $1011_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$).

3. Octal Number System (Base 8)

- The octal system uses 8 digits: 0, 1, 2, 3, 4, 5, 6, 7.
- It is often used in computing as a shorthand for binary numbers (since 3 binary digits can be represented by one octal digit).
- Octal numbers are based on powers of 8.

4. Hexadecimal Number System (Base 16)

- Hexadecimal is widely used in computing for representing binary data in a more compact form.
- It consists of 16 symbols: 0-9 and A-F (where A = 10, B = 11, ..., F = 15).
- Hexadecimal numbers are based on powers of 16 (e.g., $2F_{16} = 2\times16^1 + 15\times16^0$).

5. Roman Numerals

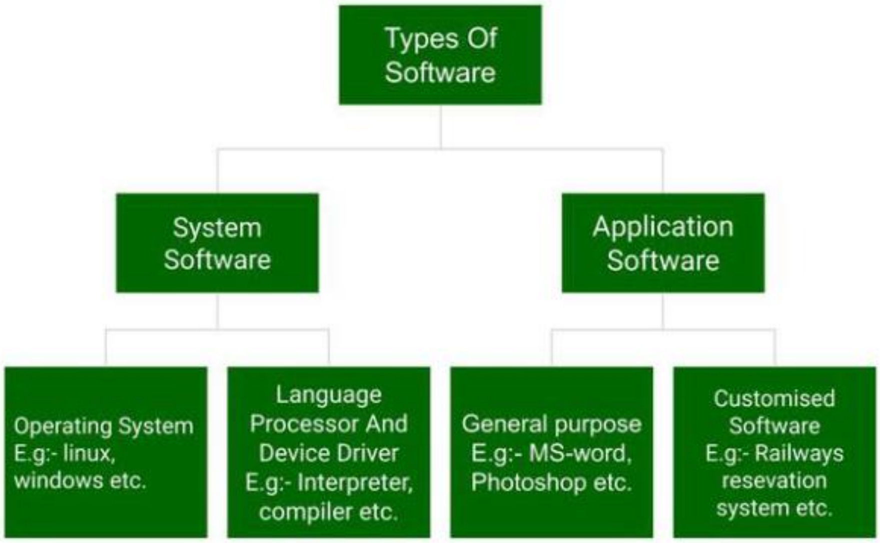
- A system used in ancient Rome, using combinations of letters from the Latin alphabet (I, V, X, L, C, D, M) to represent values.
- Roman numerals are still used today in various contexts (e.g., clock faces, movie titles).

Different computer software and types

System software is software that directly operates the computer hardware and provides the basic functionality to the users as well as to the other software to operate smoothly.

Application software is designed to help users perform specific tasks or work on particular projects, ranging from productivity to entertainment. Examples of application software:

- **Productivity Software:** These are general-purpose tools used to perform tasks like word processing, spreadsheet calculations, and presentations. Examples: Microsoft Office (Word, Excel, PowerPoint), Google Workspace (Docs, Sheets), LibreOffice.
- **Multimedia Software:** Software that enables users to create, edit, and view multimedia files such as audio, video, and images. Examples: Adobe Photoshop, VLC Media Player, Adobe Premiere Pro, Audacity.
- **Web Browsers:** Programs used to access the internet and view websites. Examples: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari.
- **Database Software:** Applications that allow users to store, manage, and retrieve data efficiently. Examples: Microsoft Access, Oracle Database, MySQL, PostgreSQL.
- **Development Software:** Tools used by developers to create software programs and applications. Examples: Integrated Development Environments (IDEs) like Visual Studio, Eclipse, PyCharm, and programming languages like Java, Python, C++.
- **Entertainment Software:** Software designed for leisure and entertainment, including video games and simulation software. Examples: Steam (gaming platform), video game titles like Fortnite, simulation software like The Sims.
- **Communication Software:** Programs that allow users to communicate with others via messaging, email, or video conferencing. Examples: Google Meet, Discord, Skype, Zoom, Microsoft Teams, Slack.



Difference between System software and Application software:

System Software	Application Software
It is designed to manage the resources of the computer system, like memory and process management, etc.	It is designed to fulfill the requirements of the user for performing specific tasks.

System Software	Application Software
Written in a low-level language.	Written in a high-level language.
Less interactive for the users.	More interactive for the users.
System software plays vital role for the effective functioning of a system.	Application software is not so important for the functioning of the system, as it is task specific.
It is independent of the application software to run.	It needs system software to run.

Classification of software based on task, source and licence

Classification Based on Task: This classification is based on the specific function or task the software is designed to perform.

- System Software: Software that provides a platform for running application software and manages hardware resources. Examples: Operating Systems (Windows, macOS), Device Drivers, Utilities (Antivirus, Disk Management Tools).
- Application Software: Software designed to perform specific tasks for the user, such as productivity, entertainment, or communication. Examples: Microsoft Office (Word, Excel), Adobe Photoshop, Web Browsers (Chrome, Firefox), Email Clients (Outlook, Thunderbird).
- Development Software: Software used by developers to create, test, and maintain other software applications. Examples: Integrated Development Environments (IDEs) like Visual Studio, Eclipse, programming languages like Python, Java.
- Embedded Software: Software designed to run on hardware devices, often with specific functions. Examples: Firmware in printers, routers, smart appliances, and automotive control systems.

Classification Based on Source: This classification focuses on the availability and modification rights of the software, based on whether the source code is available or closed.

- Proprietary Software: Software whose source code is owned and controlled by an individual or organization. Users are typically given a license to use the software, but they cannot modify or distribute the code. Examples: Microsoft Windows, Adobe Photoshop, Apple iOS.

- Open Source Software: Software whose source code is made available to the public, allowing anyone to view, modify, and distribute it under certain conditions. Examples: Linux, Mozilla Firefox, Apache HTTP Server, GIMP.
- Closed Source Software: Similar to proprietary software, this software's source code is not made available to the public, and users cannot modify or distribute it. Examples: Oracle Database, Microsoft Office.

Classification Based on License: This classification is based on the terms and conditions under which the software is distributed and used. It defines how users can use, share, and modify the software.

- Freeware: Software that is provided for free, often with limited functionality or features. The source code is typically closed, and users cannot modify or redistribute it. Examples: Adobe Acrobat Reader, Skype (free version), Google Chrome.
- Shareware: Software that is distributed for free for a limited time or with limited functionality. Users are encouraged to purchase a full version after a trial period. Examples: WinRAR, Norton Antivirus (trial version).
- Free and Open Source Software (FOSS): Software that is free to use, modify, and distribute, with the source code freely available under open licenses. Examples: Linux, Mozilla Firefox, GIMP.
- Commercial Software: Software that is developed and sold for profit. Users need to purchase a license to use the software, and the source code is typically not available. Examples: Microsoft Office, Adobe Photoshop, AutoCAD.
- Public Domain Software: Software that is not owned by anyone and is freely available for anyone to use, modify, and distribute without any restrictions. Examples: Classic games, software developed by government agencies for public use.
- Subscription-Based Software: Software that users pay for on a recurring basis (monthly or yearly) for continued access. It may come with regular updates and support. Examples: Microsoft Office 365, Adobe Creative Cloud.
- Copyleft (GNU GPL/General Public Licence): A type of open-source license that allows users to modify and distribute the software, but any modified version must also be released under the same license. Examples: GNU/Linux, WordPress.

Qualities of a good software

A good software is not only functional but also reliable, efficient, and user-friendly. Some qualities of a good software are:

1. Functionality: The software should perform the tasks it is intended to do, according to the requirements and specifications. It should meet user needs and solve the problems it was designed for.

2. Reliability: The software should operate consistently without failures, crashes, or unexpected behavior. It should be able to recover from errors gracefully.

3. Usability: The software should be easy to use, with an intuitive interface that is simple to navigate. The user experience (UX) should be positive, reducing the learning curve.

4. Performance: The software should be efficient in its resource usage (such as CPU, memory, and network) and respond quickly to user actions, even under heavy loads.

5. Security: The software should protect against unauthorized access and ensure data privacy and integrity. Security measures such as encryption, authentication, and authorization should be in place.

6. Maintainability: Good software should be easy to maintain, update, and enhance. It should have clear, well-documented code and be modular to allow for modifications and bug fixes without major disruptions.

7. Scalability: The software should be able to handle an increasing amount of work or users without a significant drop in performance. It should also be capable of being expanded or upgraded without major overhauls.

8. Portability: The software should be able to run on different platforms or environments with minimal modification. It should be compatible with various operating systems, browsers, and devices.

9. Compatibility: The software should be compatible with other systems, applications, and hardware. It should integrate well with other tools and work in different environments without conflicts.

10. Documentation: Clear, accurate, and comprehensive documentation is crucial for both end-users and developers. This includes user manuals, technical documentation, API references, and installation guides.

11. Flexibility: The software should be adaptable to changing requirements and be customizable to fit the evolving needs of the users.

12. Testability: A good software product should be easy to test and should support different testing techniques (unit testing, integration testing, etc.) to ensure its quality.

13. Efficiency: The software should perform its functions with minimal resource consumption (CPU, memory, bandwidth) while delivering optimal performance.

14. Affordability: The software should provide value for its price. It should be cost-effective to purchase, maintain, and operate, especially for small businesses and individuals.
