CHAPTER 1: COMPUTER ORGANISATION AND ARCHITECTURE

Introduction

The term computer is derived from the word "compute" which means "to calculate." This simply means that the computer is a calculating machine. Every thousandth of a second or so, a computer performs thousands, millions, or even billions of simple arithmetic operations in order to carry out its tasks.

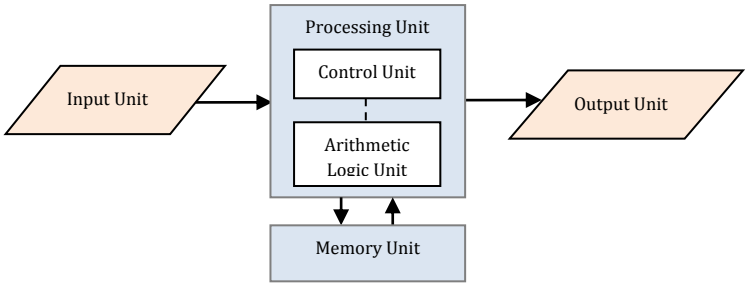
Formally, a computer is defined as an electronic device that can accept, store and process data to produce information. It can also be defined as a machine that can accept instructions and perform computations based on those instructions.

The term computer organization refers to the physical organization of the components in a computer system, and to the characteristics of those components. It includes the high-level aspects of a computer's design, such as the memory system, the memory interconnect, and the design of the internal processor or CPU.

1. Hardware

Hardware refers to the physical and tangible components of the computer like the keyboard, mouse, monitor and internal circuits. Computer hardware components can be grouped into five functionally independent main units: input, control, arithmetic and logic, memory and output units.

The input unit accepts coded information from human operators using devices such as keyboards, or from other computers over digital communication lines. The information received is stored in the computer's memory, either for later use or to be processed immediately by the arithmetic and logic unit. The processing steps are specified by a program that is also stored in the memory. Finally, the results are sent back to the outside world through the output unit. All of these actions are coordinated by the control unit.



Functional Units of a Computer

1.1. Input Unit

The input unit consists of hardware components that allow the computer user to enter data and commands into the computer. These components are called input devices. Examples of input devices are keyboard, mouse, scanner, joystick, light pen, touchpad, tracker ball and microphone.

1.1.1. Keyboard

The keyboard is the standard input device attached to computers. It is a device with keys that are pressed to enter data and commands into the computer. These keys represent letters, numbers, symbols and control characters. When a key is pressed, a number (code) representing each character, is sent to the computer to tell it which key has been pressed. The keyboard has a total of 101-104 keys divided into different groups or keypads.

- ✓ Function keys, labeled F1-F12, perform specific functions based on the software used. F1 for example is used in most software to display help.
- ✓ Alphabetical keys
- ✓ Numerical keys
- ✓ Navigation keys also called direction keys are used to navigate through documents and websites. Examples of navigation keys are the up, down, left and right arrows.
- ✓ Action keys

The way keys are arranged on a keyboard is known as the layout of the keyboard. There are different keyboard layouts but the best known are QWERTY mainly used by English speakers and AZERTY used by French speakers. Another keyboard layout modified greatly from a standard layout is DVORAK, devised to increase typing speed by placing frequently used keys more naturally.

A keyboard connects to the computer through cable PS/2 cable, USB cable or wireless (cordless).

1.1.2. Mouse

A mouse is a handheld device which is moved across a flat surface to control the movement of a cursor or pointer on the computer screen. It is a pointing device. A standard mouse has two buttons, the right and the left buttons, which are pressed to enter commands into the computer.

Different actions that can be performed with a mouse are:

- ✓ **Click**: Pressing the left mouse button. This action can be used to select objects (files, folders, commands) or options in a menu.
- ✓ **Double-click**: Pressing the left button two times in quick succession. It can be used to open a file, folder or program.
- ✓ **Right-click**: Pressing the right mouse button. A right-click opens a context menu from which options like cut, copy and paste can be selected.
- ✓ **Drag** and **drop**: Pressing the left mouse button and moving the mouse while holding, is a drag while releasing the left mouse button after a drag is a drop. Drag and drop can be used to move files or documents from one place to another, and to select text.

A mouse connects to a computer through PS/2 cable (PS/2 mouse), USB cable (USB mouse), or wireless (optical mouse).

1.1.3. Scanner

A scanner is a device that is used to input printed images such as photographs or pages of text directly into the computer. It converts hardcopy information into electronic form (softcopy) for storage in the computer. A scanner works by shining a light at the image being scanned and measuring how much light is reflected back using an optical sensor. The amount of light that is reflected back tells the computer how light or dark the image is at each point. Common scanner devices are the flatbed scanner, optical mark reader, optical character reader, barcode reader and magnetic ink character reader.

a. Flatbed Scanner

A flatbed scanner is a scanner in which the object to be scanned is placed flat against a piece of glass. When the object is placed, the scanner moves the light and sensor itself and scans the whole image automatically. Most flatbed scanners are A4 size

b. Optical Mark Reader (OMR)

c. Optical Character Reader (OCR)

d. Barcode Reader

e. Magnetic Ink Character Recognition (MICR)

1.2. Output Unit

The output unit consists of devices that allow the user to retrieve information from the computer. Output devices provide the results of computations to the person using the computer, in a way they can understand. Examples are monitor, printer, speaker and projector.

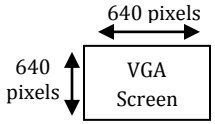
1.2.1. Monitor

Also referred to as VDU (Visual Display Unit), the monitor is the most popular output device. It displays information generated by a computer on a screen. Such output is known as softcopy output. Monitors are characterized by the technology they use, their resolution and their size.

- ✓ By the technology used, monitors are of two main types: Cathode Ray Tube (CRT) monitors and Flat Panel Display.
  - o CRT monitors are similar to a television set.
  - o Flat panel display monitors are of different types: Liquid Crystal Display (LCD), Electroluminescent Display (ELD), Gas Plasma (GP) and Thin Film Transistor (TFT) monitors.

***Question**: What are the advantages of flat panel monitors over CRT?*

- ✓ Monitor resolution refers to the number of dots (pixels – picture elements) on the screen. It is expressed as a pair of numbers that give the number of dots on a row and the number of rows. A variety of different resolutions are available. For example, VGA is 640 x 480. This means that there are 640 pixels in each row across the screen and 480 pixels in each column up and down the screen. SVGA is 800x600.



The size of a pixel is known as dot pitch. The smaller the dot pitch, the higher the number of dots the screen has. Displays with lots of pixels are called high resolution while those with fewer pixels are called low resolution. The higher the resolution, the clearer and sharper the image will appear on the monitor. Most monitors come with a .28 dot pitch.

- ✓ The size of a monitor refers to how big the monitor is. It is measured in inches along the diagonal from the bottom left hand corner to the top right hand corner of the screen. Typical sizes are 10" or 12" for LCDs and 14", 15" or 21" for desktop monitors.

1.2.2. Printer

A printer is a device that produces computer-generated information on paper. Such output is referred to as printout or hardcopy. Based on the technology used, printers can be classified into impact and non-impact printers.

- ✓ Impact printers have mechanical contact between paper and printing head (e.g. daisy wheel, dot matrix and line printers).
- ✓ Non-impact printers have no mechanical contact between paper and printing head (e.g. ink-jet, desk- jet, laser printers).

***Question**: What are the advantages and disadvantages of impact and non-impact printers?*

1.3. The Processing Unit

The processing unit, also called central processing unit (CPU), is an integrated circuit or "chip" which processes instructions and data. It is the part of the computer that interprets and executes program instructions. It also controls the other components of the system. The CPU is made up of three main components: the arithmetic-logic unit (ALU), the control unit (CU) and registers. Examples of CPUs are Intel Pentium II, III, IV, Pentium Celeron, and AMD Athlon.

1.3.1. Control Unit

The control unit acts like supervisor seeing that things are done as they ought to. It locates and retrieves program instructions from memory, interprets them and ensures that they are executed in proper sequence. It also ensures that data is selected from memory as necessary and information is stored correctly as well.

1.3.2. Arithmetic-Logic Unit

The arithmetic-logic unit (ALU) performs arithmetic and logic operations. It contains arithmetic circuits that perform arithmetic operations like addition, subtraction, multiplication and division, and logic circuits that perform comparisons like equal to, less than, greater than, greater than or equal to and less than or equal to.

1.3.3. Registers

Registers are special storage locations within the CPU that offer an advantage of speed. They work under the direction of the control unit to accept and hold data that is being processed. Since the CPU uses registers for the processing of data, the number of registers in a CPU and the size of each register affect the power and speed of the CPU.
Registers are grouped into two: special purpose registers and general purpose registers.

- ✓ Special purpose registers are dedicated to specific tasks like:
  - o the **accumulator** which collects the result of computations,
  - o the **memory address register** (MAR) which keeps track of where a given instruction or piece of data is stored in memory uses also the base and limit register
  - o the **memory data register** (MDR)which holds data values.
  - o the **program counter** (PC) which holds the address of the next instruction to be executed.
  - o The **current instruction register** (CIR) which holds the instruction being executed.

- ✓ General purpose registers on the other hand have no specific function; they are used according to the need of the program being executed.

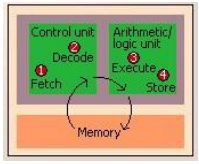
***Question:** How would you define processing?*

1.3.4. The Instruction Cycle

The instruction cycle describes how program instructions are repeatedly fetched, decoded and executed, one instruction at a time, until an instruction to HALT is encountered. Before an instruction can be fetched, it must be placed into memory as well as related data, from an input or secondary storage device. Once the necessary data and instructions are in memory, the central processing unit performs the following four steps for each instruction:

1. The control unit fetches (gets) data and instructions from memory.
2. The control unit decodes the instructions i.e. determines what they mean, and directs that the necessary data be moved to the arithmetic-logic unit.

3. The arithmetic-logic unit then executes the instruction on the data. That is, the ALU is given control and performs the actual operation on the data.
4. The arithmetic-logic unit stores the result of this operation in memory or in a register.



The Instruction Cycle

Steps 1 and 2 together are called instruction time or I-time and steps 3 and 4 together are called execution time or E-time. The combination of I-time and E-time is called the machine or instruction cycle or the fetch-decode-execute cycle. The length of time taken to fetch, decode and execute an instruction is measured in clock cycles.

1.3.5. System Clock

The CPU has a small quartz crystal circuit called the system clock that controls the timing of all computer operations. The system clock generates regular electronic pulses, or ticks, that set the operating pace of components of the system unit. Each tick is known as clock cycle and the pace of the system clock is called clock speed. Clock speed is measured in megahertz (Mhz) or gigahertz (Ghz) and refers to the number of clock cycles per second that the CPU runs at. Mega and giga stand for million and billion respectively while hertz means cycles per second. Thus, 1Ghz means one billion cycles per second. A computer that operates at 3 Ghz has 3 billion (giga) clock cycles in one second (hertz).
The faster the clock speed, the more instructions the processor can execute per second. The speed of the system clock has no effect on devices such as a printer or disk drive. The speed of the system clock is just one factor that influences a computer's performance. Other factors, such as the type of processor chip, amount of cache, memory access time, bus width, and bus clock speed.

1.4. Memory Unit

The memory unit, also called storage unit, consists of computer components that hold programs and data for use in the computer. Memory or storage devices hold programs and data that can be made available for initial or additional processing when required. A storage device is made of two parts: the storage medium and the device.

- ✓ The medium is the surface or substrate that holds actual data
- ✓ The device reads information from or stores information onto the medium

Computer storage can be classified basically into two: primary storage and secondary storage.

1.4.1. Primary Memory

Primary memory/storage is directly accessible to the CPU. It holds programs and data that the CPU is currently working with. Primary memory is also called internal memory, immediate access memory or primary storage. Primary memory consists of random access memory, read only memory and cache memory.

a. Random Access Memory

Random access memory (RAM) also called "main memory" is the temporary storage space into which a computer loads programs and user data when it is running. It is the computer's working space. It is read/write meaning that data can be read from and written onto it. RAM is also volatile meaning that everything held in it is lost when power is switched off. Two types of RAM exist: Static RAM and dynamic RAM.

- ✓ Dynamic RAM (DRAM) consists of capacitors that slowly leak their charge over time. Thus, they must be refreshed every few milliseconds to prevent data loss. DRAM is cheap memory owing to its simple design.

- ✓ Static RAM (SRAM) consists of circuits that retain their charge over time. SRAM is faster and more expensive than dynamic RAM, and does not need to be refreshed as DRAM does. Due to its cost it is not used as main memory but rather to build cache memory. also used in registers

b. Read Only Memory

Read only memory (ROM) is a kind of memory whose contents can only be read by the computer. Data found in ROM is written by the manufacturer and cannot be modified by the user. ROM is useful for holding data that never changes like the "boot" or start-up program which is run when the computer is switched on.ROM is non-volatile meaning that its content is preserved even without power. There are four types of ROM:

- ✓ Masked ROM is ROM programmed with its data when the chip is fabricated.
- ✓ Programmable ROM (PROM) is ROM that can be programmed once but not reprogrammed.
- ✓ Erasable Programmable ROM (EPROM) is ROM that can be erased by strong ultraviolet light and new data burnt into it. To do this the chip has to be removed from the machine and put back after the changes have been made.
- ✓ Electrically Erasable Programmable ROM (EEPROM) is ROM whose content can be erased electrically. In this case, the chip need not be removed from the machine. The programming is done using special software.

c. Cache Memory

Cache memory is a smaller and faster memory between the CPU and main memory, which stores copies of data from the most frequently accessed memory locations. The purpose of cache memory is to speed up accesses by storing recently used data closer to the CPU, instead of storing it in main memory. Cache is static RAM and is usually organized in levels:

- ✓ Level 1 (L1) cache, Level 2 and Level 3 cache.
- ✓ Level 1 cache is closest to the CPU or within it.
- ✓ L2 and L3 caches are outside it.

When the CPU needs to access memory, cache memory is examined first. If the data is found in cache, it is read from it. Otherwise, main memory is accessed. When the CPU refers to memory and finds the data in cache, it is said to be a HIT. Otherwise, it is a MISS.

1.4.2. Secondary Memory

Secondary memory is not directly accessible to the CPU. It is used to store programs and data for backup purposes (future use). It could be placed within the computer or connected externally. Programs and data from secondary memory must be transferred to main memory for processing. Secondary memory is also called secondary storage, mass storage, backing storage or external storage. It can be divided into magnetic storage, optical storage and solid state storage.

a. Magnetic Storage

Magnetic storage devices store data as electromagnetic charges on the magnetic surfaces of the storage units. Examples are floppy disks, hard disks and magnetic tape.

✓ **Floppy Disk**

A floppy disk consists of a round flexible plastic disk coated with a magnetic substance and protected by a plastic cover lined with a soft material that wipes the disk clean as it spins. The disk is made of two recordable surfaces which are divided into a number of circular paths called tracks. The number of tracks per surface varies with the particular type of disk. Each track is in turn divided into a number of smaller units called sectors. A sector is the basic unit of storage on the disk and has a capacity of 512 bytes.

✓ **Hard Disk**

A hard disk consists of several metallic platters which store data. Each platter has two sides and is divided into a number of rings called tracks. Tracks on a platter are numbered 0 from the outside and usually go up to 1023. Each track is divided into sectors. A sector is the basic unit of storage on the disk and has a capacity of 512 bytes. Sectors are grouped together to form clusters. A cluster is the smallest logical amount of disk space that can be allocated to hold a file. A cylinder is a sum set of all the tracks on all the platters that have the same track value.

Factors that determine the performance of hard disks are, seek time and drive rotational speed.

- Seek time is the time taken to move the read/write head over the right track and sector.
- Drive rotational speed is the total number of revolutions the disk platters make per minute. Higher rotational speed leads to higher transfer rate.

✓ **Magnetic Tape**

A magnetic tape consists of a magnetically coated stripe on which data is stored. Data is stored on the magnetic tape in chronological order or sequentially. This means that any piece of data is always stored in the next available space on the tape. To access data, the tape drive has to move through all the preceding data before it can access the desired data. This mode of access is known as sequential access. Less susceptible to environment, they are suitable for long-term storage and backup.

b. Optical Storage

Optical storage devices store data as microscopic light and dark spots on the disk surface. Examples are Compact discs, digital versatile discs and Blu-ray discs. They are less susceptible to environmental damage.

✓ **Compact Disc (CD)**

A CD is a round disk coated with a metallic surface on which data can be stored and accessed via laser technology. To store or access data on a CD, the CD drive focuses a laser beam on the disc surface. Different variations of CD exist: CD-ROM, CD-R and CD-RW. A CD can store 650MB to 700MB of data.

- CD-ROM stands for compact disc read only memory. CD-ROMs can only be read but not recorded on by the user's computer. Their content is set during manufacture.
- CD-R stands for compact disc recordable. It is a type of CD that can be recorded by the user. Once the user records on the CD, the content is set and cannot be changed. CD-R can be read by CD-ROM drives but to write on them, you need a CD-R drive.
- CD-RW stands for compact disc rewritable. It is a type of CD that can be recorded, erased and reused by the user. CD-RW cannot be read by a CD-ROM and CD-R drives. CD-RW drives are required to read and write on them.

✓ **Digital Versatile Disc (DVD)**

A DVD is similar to a CD in size and thickness but has a higher storage capacity than the CD. DVDs use a laser beam of wavelength shorter than used by CDs. This allows for smaller indentations and increased storage capacity. Just like the CD, different ions of the DVD exist: DVD-ROM, DVD-R and DVD-RW. A DVD can store up to 17GB of data. Common DVD storage capacities are:

Types	Characteristics	Capacity
DVD-5	Single-sided, Single-layer	4.7GB
DVD-9	Single-side, Dual-layer	8.5GB
DVD-10	Double-side, Single-layer	9.4GB
DVD-18	Double-side, Dual-layer	17.1GB

✓ **Blu-Ray Disc**

A Blu-ray disc is an optical disc similar to a DVD and of same size, but read and written with a blue or violet laser, whose shorter wavelength makes a higher data density possible. Blu-ray discs can hold 25 GB for single layer or 50 GB for double layer.

c. Solid State Storage

The term solid-state essentially means no moving parts. Solid-state storage devices are based on electronic circuits with no moving parts (no reels, no spinning disks). They store data using a special type of memory called flash memory. Flash memory is a type of EEPROM that can only be erased in blocks; it cannot be erased one byte at a time. In this regard it resembles a disk that is divided into sectors. Flash memory is usually used for storing larger amounts of data while EEPROM is used for small amounts of data, such as machine configuration. Examples of solid state devices are USB memory sticks, memory cards and secure digital cards.

✓ **USB Flash Drive**

A flash drive is a small, keychain-sized flash memory device with a USB interface, treated by the computer as if it were a disk drive. A flash drive is also called thumb drive, jump drive or memory stick.USB flash drives have practically replaced diskettes as a handy way to transport data. They can be carried in one's pocket and plugged into any computer for immediate access.

✓ **Secure Digital Cards**

A secure digital (SD) card is a type of flash-memory card that incorporates a cryptographic security system to prevent copyright violations, often used in digital music players and digital cameras.

1.4.3. Characteristics of Storage Devices

Five important characteristics of storage devices are capacity, access time, access method, volatility and mutability.

a. Capacity

The capacity of a storage device is the maximum amount of data that can be stored on the device's medium. It is expressed in terms of the number of data bytes the device can hold. This simply means the unit of measurement of storage capacity is byte. Units of storage are summarized as follows:

1 bit = 0 or 1	
4 bits = 1 nibble	
8 bits = 1 byte	1 character
1024 bytes = 1 kilobyte (KB)	approx. 1/2 page
1024 KB = 1 megabyte (MB)	approx. 500,000 pages
1024 MB = 1 gigabyte (GB)	approx. 5 million pages
1024 GB = 1 terabyte (TB)	approx. 5 billion pages

b. Access time

Access time is the average time taken for a storage device to search and read required data on its medium. In other words, it is how fast data can be read from or written to a memory device's medium. It is measured in seconds. Units of time are summarized as follows:

Millisecond (ms) =
Microsecond (μs) =
Nanosecond (ns) =
Picoseconds (ps) =
Femtoseconds (fs) =

c. Access Method

An access method is the technique used to retrieve information from or store information to a medium. Storage media can be accessed in two ways: sequentially or randomly.

✓ **Sequential Access**

The medium is accessed by proceeding from the beginning of the medium until the designated area is reached. Any new data is stored in the next available space on the medium. To read any data stored on the medium, the device has to start from the beginning going through each data until the required data is found. An example of sequential access medium is magnetic tape.

✓ **Random Access**

Random access means that any location in storage can be accessed at any moment in approximately the same amount of time. Data is accessed in any order, regardless of its location on the medium. To read any data stored on the medium, the device does not need to go through all preceding data. It is also called direct access. Examples of direct access devices are RAM, ROM, CDs, DVDs and magnetic disks.

d. Volatility

Volatility refers to the behavior of the device without power. A device can be volatile or non-volatile. Volatile means that the device loses its content when power is switched off. Examples of volatile devices are Cache and RAM.

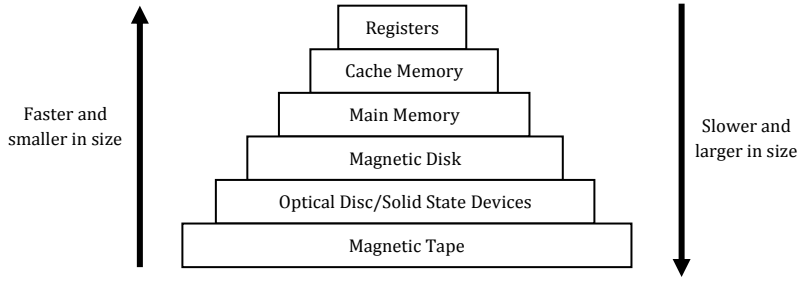
Non-volatile means that the device preserves its content even without power. Examples of non-volatile devices are ROM, magnetic disks, optical discs and solid state devices.

e. Mutability

Mutability simple refers to the ability for the device to be accessed for reading and/or writing. Storage devices to which data can be written and read from are said to be mutable (read/write). Examples are RAM, Flash memory, CD-RW, DVD-RW. Those that can be accessed only for reading are immutable (read only). Examples are ROM, CD and DVD.

1.4.4. Memory Hierarchy

Modern computers manage memory by organizing memory into a hierarchy in which large and slower memories feed data into smaller but faster memories for faster processing of data. This organization of computer memory is known as memory hierarchy. At the top of the hierarchy are the CPU registers followed by cache memory. The next level in the hierarchy is made up of main memory, which is followed by magnetic disk.



Memory hierarchy

1.5. The Motherboard

The motherboard is the main printed circuit board in the computer which holds the chipset and other electronic components that give function to the computer. The motherboard is indispensable to the computer and provides the main computing capability.

1.5.1. Motherboard Form Factor

Motherboard types are better described by what we call the motherboard form factor**.** The form factor of a motherboard determines the specifications for its general shape and size. It also specifies what type of case and power supply will be supported, the placement of mounting holes, and the physical layout and organization of the board. The most common form factors found in modern PCs are:

- ✓ Advanced Technology (AT)
- ✓ Advanced Technology Extended (ATX)
- ✓ Low Profile Extension (LPX)
- ✓ New Low profile Extended (NLPX)

1.5.2. The System Bus

A computer bus is a set of parallel lines that interconnects various components inside the computer, allowing the exchange of data between them. It is the pathway between these components, enabling data to be transferred from one component to another. The width or size of a bus is determined by the number of lines it has. The system bus is made up of three different busses: the data bus, the address bus and the control bus.

a. The Data Bus

The data bus carries data between the CPU and main memory or peripherals. During a write operation, data is carried from the CPU and during a read operation, data is carried into the CPU.

This means that the data bus is bidirectional. The size of the data us determines how much data can be transferred in a single operation.

b. The Address Bus

The address bus carries address information from the CPU to main memory or peripherals. It is unidirectional. The CPU uses the address bus to send the address of the memory location to be written to or read from. Also, when the CPU reads data from or writes to a port, it sends the port address out on the address bus. The size of the address bus determines the maximum amount of main memory (RAM) that can be addressed. A computer with a 32 bit bus size can address a maximum RAM of 2^{32} bits = 4GB.

c. The Control Bus

The control bus is used by the CPU to send out signals to enable the outputs of addressed memory devices or port devices. Typical control bus signals are memory read, memory write, I/O read and I/O write.

When the CPU wants to read data from a memory location, it sends out the memory address of the desired data on the address bus and then sends out a Memory Read signal on the control bus. The memory read signal enables the addressed memory device to output the data onto the data bus. The data from the memory travels along the data bus to the CPU.

1.5.3. Input/Output Interfaces

The CPU communicates with I/O devices through bus interfaces connected to the system bus. These bus interfaces also called expansion buses are then connected to ports which allow the exchange of data and information between the computer and external (peripheral) devices.

a. I/O Ports

A port is a pathway for data and information to go into and out of the computer from external devices such as keyboards, monitors and printers.
There are many standard ports as well as custom electronic ports designed for special purposes. Examples of ports are:

- PS/2 ports for connecting the keyboard and mouse
- VGA (Video Graphics Adapter) port for connecting the monitor
- RJ45 port for connection to an Ethernet network
- RJ14 for connection to the Internet via phone line
- USB port for connecting USB devices like Flash drives, printers, keyboards and mice
- Serial ports for connecting serial devices like PDAs
- Parallel port connecting parallel devices like printers
- Firewire is high performance serial bus, for connecting devices to your personal computer. FireWire provides a single plug-and-socket connection on which up to 63 devices can be attached with data transfer speeds up to 400 mbps.

b. I/O Bus Standards

Ports follow standards that define their use. Examples of such standards are:

- SCSI: Small Computer System Interface
- USB: Universal Serial Bus
- IDE: Integrated Device Equipment
- PCI: Peripheral Component Interconnect
- ISA: Industry Standard Architecture
- EISA: Extended ISA
- VESA: Video Electronics Standard Architecture
- SIMM: Single Inline Memory Module
- DIMM: Dual Inline Memory Module
- PCMCIA: Personal Computer Memory Card International Association

2. Computer Architecture

Computer architecture refers to those attributes of a system that have a direct impact on the logical execution of a program like the instruction set, the number of bits used to represent various data types, I/O mechanisms and techniques for addressing memory.

2.1. The Von Neumann Architecture

The term Von Neumann computer in its strictest definition refers to a specific type of computer architecture in which instructions and data are stored together in a common memory. This type of architecture is distinguished from the Harvard architecture in which separate memories are used to store instructions and data. It is a stored-program computer model that was designed by the Hungarian born Mathematician, John Von Neumann. It is based on three concepts:

- Both data and instructions (programs) are stored in a single storage structure called memory
- The contents of this memory are addressable by location, without regard to the type of data contained there.
- It has a single processing unit which is. As such, execution occurs in a sequential fashion from one instruction to the next.

Programs being stored in memory ensures that by altering the stored program, the computer can perform a different task – reason why a computer is called a general purpose machine.
Neumann divided the computer into four functional units: input, processing, storage and output.

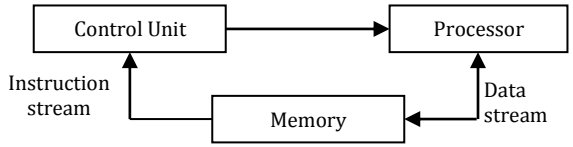
2.2. Flynn's Taxonomy of Computer Architecture

Michael Flynn classified computer architecture into four distinct categories based on the notion of stream or flow of information into the CPU. He identified two types of information streams:

- ✓ The instruction stream which is the flow of instructions into the processing unit
- ✓ The data stream which is the flow of the data on which the instructions are performed.

2.2.1. Single Instruction, Single Data Stream

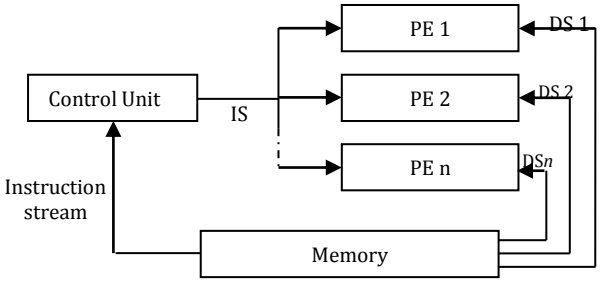
A SISD machine is a uniprocessor machine which receives a single stream of instructions that operate on a single stream of data. At any given moment, a single instruction is being executed on a given data set. In SISD machines, instructions are executed sequentially. That is, instructions are executed one after the other, one at a time. Hence these machines are also called sequential or serial processor machines. They are typical examples of Von Neumann's computer model.



SISD Architecture

2.2.2. Single Instruction, Multiple Data Stream

SIMD machines are multiprocessor machines which are capable of executing the same instruction on all the processors at the same time. They have *n* identical processors which operate under the control of a single instruction stream issued by a central control unit on different data sets. At any given moment, the control unit broadcasts the same instruction to all processors which operate on different data sets from distinct data streams. SIMD machines are also called array processor machines. An example is the CRAY's vector processing machine.

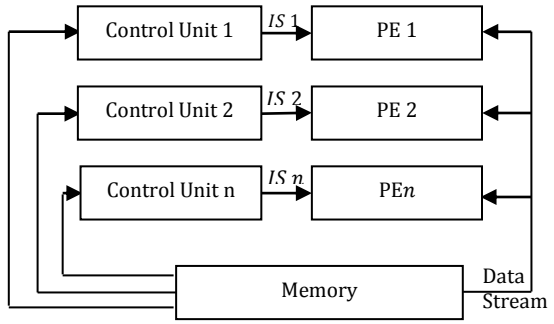


DS = Data Stream
IS =Instruction Stream
PE = Processor Element

SIMD Architecture

2.2.3. Multiple Data, Single Instruction Stream

MISD machines are multiprocessor machines capable of executing different instructions on different processors, but all of them operating on the same data set. They have *n* processors, each with its own control unit, that share a common memory. Each processor receives a distinct instruction stream but all operate on the same data stream. Such machines no longer exist.

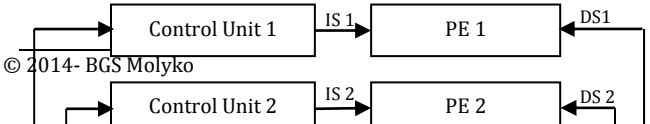


IS = Instruction Stream
PE = Processor Element

MISD Architecture

2.2.4. Multiple Data, Multiple Instruction Stream

MIMD machines are multiprocessor machines capable of executing multiple instructions on multiple data sets. They have *n* processors, *n* streams of instructions and *n* streams of data. Each processor element in this model has a separate instruction stream and data stream hence such machines are well suited for any kind of application. Examples of MIMD machines include multi-core computers like the Blue Gene, Fujitsu K computer and CRAY Jaguar.



DS = Data Stream

***Question**: SIMD, MISD and MIMD machines are known as parallel processing machines. What do you understand by parallel processing?*

2.3. Parallelism

The performance of computers can be increased by making them perform a number of operations in parallel. This is known as parallelism. Parallelism can be achieved through parallel processing and pipelining.

2.3.1. Parallel Processing

Parallel processing is the use of multiple (independent) processors simultaneously to execute a single program or task. The problem is divided into portions so that multiple processors work on their assigned portion of the problem at the same time. Parallel processing requires special software that recognizes how to divide the problem and then bring the results back together again. Some personal computers implement parallel processing with multiprocessors while others have multicore processors.

a. Multicore Processors

Multiple processing units can be fabricated on a single chip. In this case, each of these processors is termed core. The term processor is then used for the complete chip. Hence, we have the terminology *dual-core*, *quad-core*, and *octo-core* processors for chips that have two, four, and eight cores, respectively.

b. Multiprocessors

Computer systems may contain many processors, each possibly containing multiple cores. Such systems are called *multiprocessors*. These systems either execute a number of different application tasks in parallel, or they execute subtasks of a single large task in parallel. All processors usually have access to all of the memory in such systems, and the term *shared-memory multiprocessor* is often used to make this clear. The high performance of these systems comes with much higher complexity and cost, arising from the use of multiple processors and memory units, along with more complex interconnection networks.

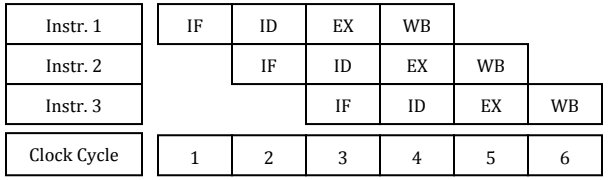
2.3.2. Pipelining

The simplest way to execute a sequence of instructions in a processor is to complete all steps of the current instruction before starting the steps of the next instruction. If we overlap the execution of the steps of successive instructions, total execution time will be reduced.

For example, the next instruction could be fetched from memory at the same time that an arithmetic (execute) operation is being performed on the register operands of the current instruction. This form of parallelism is called *pipelining*.

***Definition**: Pipelining is a technique that allows for the simultaneous execution of the parts of an instruction.*

Below is an example of a four-stage pipeline with three instructions. The stages are:
IF: instruction fetch ID: instruction decode EX: execute WB: write back



Cycle 1: Instruction 1 is being fetched from memory.
Instructions 2 and 3 have not entered the pipeline yet

Cycle 2: Instruction 1 is being decoded
Instruction 2 is being fetched
Instruction 3 has not entered the pipeline yet

Cycle 3: Instruction 1 is being executed
Instruction 2 is being decoded
Instruction 3 is being fetched

Cycle 4: Instruction 1's results are being written back to register or memory
Instruction 2 is being executed
Instruction 3 is being decoded

Cycle 4: Instruction 1 is completed
Instruction 2's results are being written back to register or memory
Instruction 3 is being executed

Cycle 5: Instruction 1 is completed
Instruction 2 is completed
Instruction 3's results are being written back to register or memory

Cycle 6: Instruction 1 is completed
Instruction 2 is completed
Instruction 3 is completed

2.4. Instruction Set Architecture

Every CPU has a predefined instruction set which defines the primitive operations that it can perform.

***Definition**: An instruction set is the collection of bit patterns or binary codes for the machine operations that a processor has been designed to perform.*

An instruction set architecture (ISA) consists of the instruction set together with the resources needed for their execution. It provides the interface between the computer's software and hardware and can be viewed as the programmer's view of the machine. Besides instructions, the ISA defines items in the computer that are available to a program - e.g. data types, registers, addressing modes, and memory.
There are two types of fundamental CPU architectures based on the notion of instruction set: complex instruction set computers and reduced instruction set computers.

2.4.1. Complex Instruction Set Computer (CISC)

CISC (pronounced *sisk*) is a CPU design with a large amount of different and complex instructions. In CISC processors, the control unit contains a number of micro-electronic circuitry to generate a set of control signals and each micro-circuitry is activated by a microcode. Complex instructions are first decoded and the corresponding microcode routine dispatched to the execution unit.
The standard features of CISC processors are:

- ✓ a large number of different and complex instructions.
- ✓ the use of complex addressing modes.
- ✓ execution of different machine programs on a CISC machine.
- ✓ the use micro-program control unit.
- ✓ limited number of registers.
- ✓ variable length instruction encoding
- ✓ direct memory access

Examples of CISC processors are: Intel 386, 486, Pentium, Pentium Pro, Pentium II, Pentium III, Motorola's 68000, 68020, 68040, etc.

2.4.2. Reduced Instruction Set Computers (RSIC)

RISC (pronounced *risk*) is a CPU design with a small number of basic and simple machine language instructions, from which more complex instructions can be composed. RISC instructions are hardwired. That is, they are built into the chip with hardware rather than programming. Hardware implementation of instructions is much faster and uses less silicon than a microcode implementation.
The standard features of RISC processors are:

- ✓ a small and limited number of instructions.
- ✓ the use of hardwired control unit.
- ✓ consumption of less power and high performance
- ✓ instructions are very simple and consistent
- ✓ the use of simple addressing modes
- ✓ fixed length instructions (easier to decode)
- ✓ pipelining possible because of fixed length instructions

Examples of RISC processors are: IBM RS6000, MC88100
DEC's Alpha 21064, 21164 and 21264 processors, Motorola/IBM PowerPC

2.4.3. Addressing Modes

A machine instruction specifies to the CPU what to do, where the data is located, and where the output data (if any) will be put. Every instruction is made up of two parts: Op-code and operand.

- ✓ The Op-code (operation code) denotes the basic machine operation e.g. ADD, STORE, SHIFT, XOR.
- ✓ The operand (one or more) provides the data which the instruction manipulates.

For example,

ADD A, $0E	ADD is the op-code; A and $0E are operands
MOV AX, #0	MOV is the op-code; AX and #0 are operands

Operands can be specified in different ways in an instruction. The way an operand is specified in an instruction is known as an addressing mode.

Definition*: An addressing mode refers to the way in which the processor locates the data (operand) associated with an instruction.*

a. Register Addressing

In register mode, operand is the contents of a processor register; the name of the register is given in the instruction. For example:

ADD R1, R2, R3

The above instruction uses three register to hold all operands. Registers R2 and R3 hold the two source operands while register R1 holds the result of the computation.

b. Immediate Addressing

In immediate mode, the operand is given explicitly in the instruction. In other words, the value to be manipulated by the instruction is immediately part of it. There is no need for any additional information. For example:

ADD R2, R3, #10

This instruction adds the number 10 to the content of register R3 and stores the result in register R2. The operand 10 has been addressed immediately.

c. Direct Addressing

In this addressing mode, the operand is in a memory location; the address of this location is given explicitly in the instruction. It is also called absolute addressing. For example:

LOAD R2, $0FF

The above instruction loads the content of the memory location $0FF into register R2. The operand $00FF has been addressed directly.

d. Indirect Addressing

Here, the effective address of the operand is the contents of a register that is specified in the instruction. For example:

ADD R2, R2, #R3

This instruction adds the contents of the memory location held in register R3 to the content of register R2 and the result of the operation is held in register R2.

e. Indexed Addressing

The effective address of the operand is generated by adding a constant value to the contents of a register.

3. Number Systems and Data Representation

Computers use binary patterns (fixed number of bits) to represent data, which could be numbers, letters, videos, images or other symbols. It is important to decide on how these patterns will be interpreted. The interpretation of binary patterns is called data representation or encoding. Different representation schemes exist.

3.1. Number Systems

A number system is a set of symbols and rules used to represent numbers. The number of different symbols used in a given number system is known as the base or radix of the system. The largest value of a symbol (digit) in a given number system is always one less than the base or radix of that system. If the base of a system is represented by"b", then the largest value a digit in that system can assume is "$b-1$".

Examples of number systems include the binary, octal, decimal (denary) and hexadecimal systems.

3.1.1. The Decimal System

The decimal system has a base value of 10. Its maximum or largest value of digit is $(b-1)=10-1=9$, meaning thatthe decimal system uses the digits $0, 1, 2, 3, 4, 5, 6, 7, 8,$ and 9 to represent any quantity.

Consider the number 234_{10}

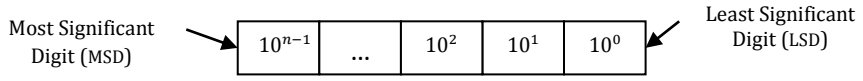
It can be written as$234 = 200 + 30 + 4$

$$234 = (2\times 100) + (3\times 10) + 4$$

$$234 = (2\times 10^2) + (3\times 10^1) + (4\times 10^0)$$

This means that there are two one hundreds, three tens and four ones in the number 234.

For any n-digit number, the value each digit represents depends on its position in the number. Decimal positions (place values) are powers of ten as shown below:



Decimal positions for an n-bit number

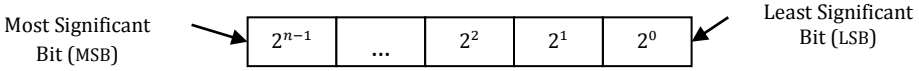
Where, 10^0 is the ones place
10^1 is the ten's place
10^2 is the hundreds place

The value of any digit in a given n-digit number is obtained by multiplying the digit by its place value. The value of the number is the sum of the products of the digits and their place values. That is, we multiply each digit by its place value, then we add the different products obtained.

Remark Each digit position in decimal has a weight that is ten times the one to its immediate right. That is, $10^2$is ten times greater than 10^1 which is ten times greater than 10^0.

3.1.2. The Binary System

The binary system has a base value of 2. Only two digits 0 and 1, are used to represent any quantity in binary. These digits are called binary digits or more commonly bits. To express any quantity in binary we use powers much like in the decimal system but this time, the weights are powers of 2 as shown below.



Binary positions for an n-bit number

e.g. the binary number 1101_2 is expressed as$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
This means that there are 1 ones, 0 twos, 1 fours, and 1 eights in the number 1101.

a. Decimal to Binary Conversion

To convert from decimal to binary we use the repeated division method. The general technique of this method can be used to convert any decimal number to any other number system.

Step 1: divide the decimal number you want to convert by 2 in regular long division until you obtain a final remainder
Step 2: use the remainder as the least significant bit of the binary number
Step 3: divide the quotient you got from the first division operation by 2 until you obtain a final remainder
Step 4: use the remainder as the next digit of the binary number
Step 5: repeat steps 3 and 4 as many times as necessary until you get a quotient that cannot be divided by 2.
Step 6: use the last remainder (that cannot be divided by 2) as the most significant bit of the binary number.

Example 1: Convert $213_{10}$to base 2

$213/2 = 106$	remainder 1	LSB
$106/2 = 53$	remainder 0	
$53/2 = 26$	remainder 1	
$26/2 = 13$	remainder 0	
$13/2 = 6$	remainder 1	
$6/2 = 3$	remainder 0	
$3/2 = 1$	remainder 1	
$1/2 = 0$	remainder 1	MSB

Therefore, $213_{10} = 11010101_2$

Example 2: Convert 25.75_{10} to binary

$25_{10} = 11001_2$,

$0.75 \times 2 = 1.5 \quad 1$
$0.5 \times 2 = 1.0 \quad 1$
$0.75_{10} = 11_2$

$\Rightarrow 25.75_{10} = 11001.11_2$

Example 3: Convert: 33.33_2 to decimal.

b. Binary to Decimal Conversion

To convert a binary number to decimal, we proceed as follows:

Step 1: starting with the 1s place, write the binary place value over each digit in the binary number to be converted.
Step 2: add up all the place values that have a "1" in them

Example 1: Convert 11010_2 to base 10

16	8	4	2	1
1	1	0	1	0

$16 + 8 + 2 = 26_{10}$

Convert: 1111_2 (15), 10011_2 (19), 10101_2(21) to base 10.

Example 2: Convert 1101.011_2 to base 10

$1101.011_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$
$= 8 + 4 + 0 + 1 + 0 + {}^1/_4 + {}^1/_8$
$= 13 + 0.25 + 0.125$
$= 13 + 0.25 + 0.125$
$= 13.375$

***Exercise**: Convert the following binary numbers to decimal*
1010.101₂ (10.625) *(ii) 10011.001₂ (19.125)*

***c.* Binary Arithmetic**

✓ **Addition**

Rules for addition:
0+0=0; 0+1=1; 1+0=1; 1+1=10; 1+1+1 = (1+1) + 1 = 10+1 = 11

i. $100 + 01 = 101$ .....$(4 + 1 = 5)$
ii. $1101 + 101 = 10010$ ..... $(13 + 5 = 18)$
iii. $1111 + 110 = 10101$ ..... $(15 + 6 = 21)$

✓ **Subtraction**

Rules for subtraction:
0-0 = 0; 1-0 = 1; 0-1 = 0 (we borrow a digit); 1-1 = 0

i. $10101 - 100 = 10001$ $(21 - 4 = 17)$
ii. $1010 - 101 = 101$ ..... we borrow a digit worth 2 to continue with the calculation $(10 - 5 = 5)$
iii. $1001011 - 110101 = 10110$ $(75 - 53 = 22)$

✓ **Multiplication**

Rules for multiplication:
$0 \times 0 = 0$; $0 \times 1 = 0$; $1 \times 0 = 0$; $1 + 1 = 1$

i. $101 \times 10 = 1010$ ..... $(5 \times 2 = 10)$
ii. $111 \times 101 = 100011$ ..... $(7 \times 5 = 35)$
iii. $11011 \times 1101 = 101011111$ ..... $(27 \times 13 = 351)$

✓ **Division**

Rules for division:

$0/1 = 0$; $1/1 = 1$; $1/0$ = undefined.

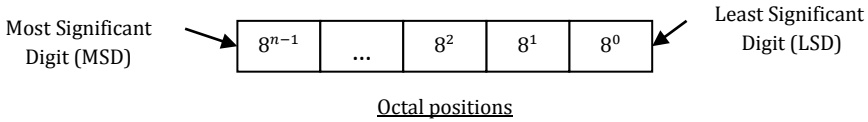
i. $1000 \div 10 = 100$ ..... $(8 \div 2 = 4)$

ii. $100001 \div 11 = 1011$.....$(33 \div 3 = 11)$

$11011 \div 101 = 101\ r\ 10$ .....$(27 \div 5 = 5\ r\ 2)$

3.1.3. The Octal System

The octal system has a base value of 8. It uses the digits 0-7 to represent any quantity. Weights in octal are powers of eight as shown below.



Octal positions

Where 8^0 is the one's place

$8^1$is the eights place

$8^2$is the sixty fours place

Example: $367_8 = 3 \times 8^2 + 6 \times 8^1 + 7 \times 8^0$

(This means that there are 3 sixty fours, 6 eight's and 7 one's in 367)

a. Binary to Octal Conversion

Theorem

If base R_1 is the integer power of another base, R_2 (i.e. $R_1 = R_2^d$), then every group of d digits in R_2 is equivalent to 1 digit in the R_1 base.

For example:

Assume that: $R_1 = 8$ (octal) and $R_2 = 2$ (binary)

From the theorem, $8 = 2^3$

Hence, 3 digits in base-2 is equivalent to 1 digit in base-8.

From the stated theorem, the following is a binary-octal conversion table.

Binary	Octal
000	0
001	1
010	2
011	3

Binary	Octal
100	4
101	5
110	6
111	7

To convert from binary to decimal, proceed as follows

Step1: make groups of three bits starting from the least significant bit and move towards the most significant bit.

Step 2: replace each group of bits by its octal representation.

Example: 1. Convert $100110_2$to base 8

$100110_2 = 100\ \ 110$

$100_2 = 4_8, 110_2 = 6_8$

$\therefore 100110_2 = 46_8$

Example 2: Convert 1011101_2 to base 8

$1011101_2 = 001\ 011\ 101$

$001_2 = 1_8, \quad 011_2 = 3_8, \quad 101_2 = 5_8$

$\therefore\ 1011101_2 = 135_8$

b. Octal to Binary Conversion

To convert from octal to binary, we replace every octal digit by its 3-bits binary equivalent.

Example1: Convert 73_8 to binary

$7_8 = 111_2, \ \ 3_8 = 101_2$

$\therefore\ 73_8 = 111101_2$

Example 2: Convert 450_8 to binary

$4_8 = 100_2, \ \ 5_8 = 101_2, \ \ 0_8 = 000_2$

$\therefore 450_8 = 100101000_2$

3.1.4. The Hexadecimal System

The prefix "hex" means 6 and "deci" means 10. The hexadecimal number system is thus a base-16 number system. Each digit position represents a power of 16. The digits used in this system are 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F, where$A = 10_{10}$, $B = 11_{10}$, $C = 12_{10}$, $D = 13_{10}$, $E = 14_{10}$, $F = 15_{10}$.

Assume that:

$R_1 = 16$ (hexadecimal)

$R_2 = 2$ (binary)

From the theorem, $16 = 2^4$

Hence, 4 digits in a binary number is equivalent to 1digit in the hexadecimal number system.

The following is the binary-hexadecimal conversion table

Binary	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binary	Hexadecimal
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

- ✓ To convert from binary to hexadecimal

Step1: make groups of four bits starting from the least significant bit and move towards the most significant bit.
Step 2: replace each group of bits by its hexadecimal value representation.

Example 1: Convert 100110_2 to base 16
$100110_2 = 0010\ \ 0110$
$0010_2 = 2_{16}, 0110_2 = 6_{16}$

$\therefore\ 100110_2 = 26_{16}$

Example 2: Convert 1011101_2 to base 16
$1011101_2 = 0000\ \ 0101\ \ 1101$
$0000_2 = 0_{16},\quad 0101_2 = 5_{16},\quad 1101_2 = C_{16}$

$\therefore 1011101_2 = 5C_{16}$

- ✓ To convert from hexadecimal to binary, we carry out the inverse operation. That is, we replace every hexadecimal digit by its 4-bits binary equivalent.

Example 1: Convert 450_8 to binary
$4_{16} = 0100_2,\ 5_{16} = 0101_2,\ 0_{16} = 0000_2$
$\therefore 450_{16} = 10001010000_2$

Example 2: Convert $6E$ to base 2
$6 = 0110,\quad E = 1110$
$\therefore\ 6E = 1101110_2$

***Assignment**:*

1) Convert the following octal numbers to hexadecimal.
i) 65_8 (ii) 53_8

2) Convert the following hexadecimal numbers to octal.
i) $12B_{16}$ (ii) $F2E_{16}$

3.2. Representation of Numbers

A number can be a whole number (integer) or a fractional number (real number). It may also be a positive or a negative number. There are various ways by which numbers are represented using 0s and 1s.

3.2.1. Representation of Unsigned Integers

Unsigned integers can represent zero and all positive integers. The value of an unsigned integer is interpreted as "the magnitude of its underlying binary pattern". That is, for an unsigned integer, all the bits in the binary pattern are used to represent the magnitude of the integer.
In binary, an n-bit pattern can represent 2^n distinct integers. Therefore, an n-bit pattern will represent integers from 0 to $2^n - 1$.

For example:
- A 4-bit pattern will represent the integers 0 to $(2^4) - 1 = 3$
- An 8-bit pattern will represent the integers 0 to $(2^8) - 1 = 255$

3.2.2. Representation of Signed Integers

Signed integers can represent zero, positive integers, as well as negative integers. Four representation schemes are available for signed integers:

1. Sign-Magnitude representation
2. 1's Complement representation
3. 2's Complement representation
4. Biased representation (Excess-n)

a. Sign-Magnitude Representation

In sign-magnitude representation, the most significant bit is the sign bit with 0 for positive and 1 for negative. The remaining $(n-1)$ bits represent the absolute value of the integer. The absolute value of the integer is interpreted as "the magnitude of the $(n-1)$-bit pattern".

Example 1: Suppose $n = 8$ and the binary representation 01000001.
Sign bit is 0 ⇒number is positive
Absolute value is $1000001 = 65$
Hence, the integer is 65.

Example 2: Suppose $n = 8$ and the binary representation 10001001.
Sign bit is 1 number is negative
Absolute value is $0001001 = 9$
Hence, the integer is -9

The drawbacks of sign-magnitude representation are:
- There are two representations (0000 0000 and 1000 0000) for the number zero, which could lead to inefficiency and confusion.
- Arithmetic is cumbersome making the design of electronic circuits for this scheme difficult.

b. 1's Complement Representation

In 1's complement representation the most significant bit is still the sign bit. The remaining $(n-1)$ bits represent the magnitude of the integer as follows:
- for positive integers, the absolute value of the integer is equal to "the magnitude of the $(n-1)$-bit pattern".
- for negative integers, the absolute value of the integer is equal to "the magnitude of the complement (inverse) of the $(n-1)$-bit pattern".

Example 1: Using 1's complement, represent the following base 10 numbers on 8 bits.
i) $26_{10} = 11010$
Number is positive ⇒ sign bit is 0
$\Rightarrow 26_{10} = 0\ 0011010$

ii) -25
Number is negative ⇒ sign bit is 1
$25 = 11001$
Complement of 0011001 is 1100110
$\Rightarrow -25 = 1\ 1100110$

Example 2: Give the decimal equivalent of the following 1's complement binary representations.

i) 0010 0001.

Sign bit is 0 ⇒ positive
Absolute value is 010 0001 = 33
Hence, the integer is +33

ii) 1000 0001

Sign bit is 1 ⇒ negative
Absolute value is the complement of 000 0001, i.e., 111 1110 = 126
Hence, the integer is −126

The problem with this representation is that there are still two representations for zero (0000 0000 and 1111 1111).

- ✓ **Addition in 1's Complement**
  - Add binary representations of the two numbers
  - If there is a carry (referred to as end-round carry), add it to the result.

- ✓ **Subtraction in 1's complement**

Subtraction is implemented using addition as follows:

- Determine the 1's complement of the negative number
- Add the binary representations of the two numbers
- If there is any carry, add it to the result

Example: subtract 37 from 51 in binary
37 = 0100101 ⇒ 37 = 1011010 ⇒ −37 = 1101101in 1's complement

```
 (51)   00110011
(-37) + 11011010
       ----------
      1 00001101
      +        1
       ----------
  (14) 00001110
```

c. 2's Complement Representation

In 2's complement representation, the remaining $(n-1)$ bits represent the magnitude of the integer as follows:

- for positive integers, the absolute value of the integer is equal to "the magnitude of the $(n-1)$-bit pattern".
- for negative integers, the absolute value of the integer is equal to "the magnitude of the complement of the $(n-1)$-bit pattern plus one". That is, we just add one to 1's complement to get 2's complement.

An alternative and simple way of getting 2's complement is

- to write the representation of the positive number
- starting from the least significant bit, flip all the bits to the left of the first 1.

Example 1: Using 2's complement, store the following base 10 numbers.

i) −24

Number is negative ⇒ sign bit is 1
24 = 11000 = 0011000
1's complement of 24 is 1100111
2's complement of 24 is 1100111 + 1 = 1101000
⇒ −24 is represented by 1 1101000

ii) −13

Number is negative ⇒ sign bit is 1
13 = 1101 = 0001101
1's complement of 13 is 1110010
2's complement of 13 is 1110010 + 1 = 1110011
⇒ −13is represented by 1 1110011

Example 2: Give the decimal equivalent of the following 2's complement binary representations.

i) 01100000

Sign bit is 0 ⇒ positive number
Absolute value is 110 0000 = 96
Hence, the integer is +96

ii) 1 001 0001

Sign bit is 1 ⇒ negative number
Absolute value is the complement of 001 0001 plus 1, i.e.,110 1110 + 1 = 111
Hence, the integer is −111 $(-2^7 + 2^4 + 2^0 = -128 + 16 + 1 = -111)$

Modern computers use 2's complement in representing signed integers. This is because:

1. There is only one representation for the number zero unlike in sign-magnitude and 1's complement representations.
2. Positive and negative integers can be treated together in addition and subtraction. Subtraction can be carried out using the "addition logic".

- ✓ **Addition in 2's Complement**
  - Add binary representations of the two numbers
  - If there is a carry, ignore it

- ✓ **Subtraction in 2's Complement**

Subtraction is implemented as follows:

- Determine the 2's complement for the negative number
- Add the binary representations of the two numbers
- If there is any carry, ignore it

Example: subtract 37 from 51
37 = 0100101 ⇒ 37 = 1011010 (1's C) ⇒ − 37 = 1 1011011 (2's C)

```
(51)   00110011
(-37) + 11011011
       - ------------
(14) 1 00001110   discard end-round carry
```

d. Excess-n Representation

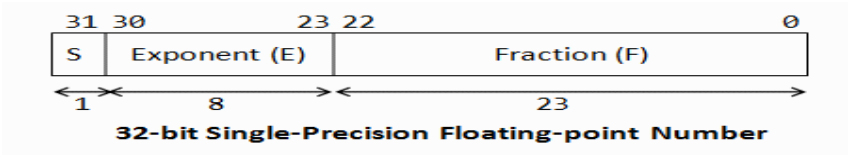
In excess-n notation, where n is the number of bits used, the value represented is the unsigned value with a fixed value 2^{n-1} subtracted from it.

Example 1: Represent the

3.2.3. Representation of Real Numbers

Computers represent real numbers in a form similar to that of scientific notation of *mantissa* (M) and an *exponent* (E) of a certain *radix* (R), in the form of $M \times R^E$. Both M and E can be positive as well as negative. For example, 1.25×10^{-1} is expressed in scientific notation where 1.25 represents the mantissa, -1 the exponent and 10, the radix.
The IEEE 754 standard for representing floating-point numbers in the computer has two representation schemes: 32-bit single-precision and 64-bit double-precision.

- ✓ In 32-bit single-precision floating-point representation:
  - the most significant bit is the *sign bit* (S), with 0 for negative numbers and 1 for positive numbers.
  - the following 8 bits represent the *exponent* (E).
  - the remaining 23 bits represent the *mantissa* (M) or *fraction* (F).



32-bit Single-Precision Floating-point Number

- ✓ In 64-bit single-precision floating-point representation:
  - the most significant bit is the *sign bit* (S), with 0 for negative numbers and 1 for positive numbers.
  - the following 16 bits represent the *exponent* (E).
  - the remaining 47 bits represent the *mantissa* (M) or *fraction* (F).

In floating-point representation, the mantissa and exponent play two important roles. The mantissa affects the precision of the number while the exponent affects the range. Precision has to do with the exactness of a number while range has to do with the set of numbers that can be represented.

For simplicity, we will use a 14-bit pattern for floating point with a 5-bit exponent and an 8-bit mantissa. In this case, we will use excess-16 representation for the exponent.
Examples:

1. Represent the following numbers as 32-bit single precision floating-point numbers.
   a. 32.2
   b. -15.37
   c. 0.125

Solution:

a. We convert the number to binary

$32 = 100000$
$0.2 = 0.2 \times 2 = 0.4 \quad 0 \quad MSB$
$= 0.4 \times 2 = 0.8 \quad 0$
$= 0.8 \times 2 = 1.6 \quad 1$
$= 0.6 \times 2 = 1.2 \quad 1$
$= 0.2 \times 2 = 0.4 \quad 0$
$= 0.4 \times 2 = 0.8 \quad 0$
$= 0.8 \times 2 = 1.6 \quad 1$
$= 0.6 \times 2 = 1.2 \quad 1$

So a binary representation of 0.2is given by 0.001100110011
64.2 is 100000.001100110011

Step 2: we normalize the binary representation
1000000.001100110011 becomes $1.00000001100110011 \times 2^5$

Step 3: 5 is the true exponent.
In excess-16, we have $5 + 16 = 21$
21in 5-bit unsigned representation is 10101

Step 4: The mantissa stored is what is on the right side of the radix point of the normal form.
So M is 00000001100110011001100110

Putting all these together we have 64.2 represented as:

0 10101 00000001[10011001100110]

NB: On 8 bits we see that part of the number will be missing. That is why we say the mantissa determines the precision (exactness) of the number.

b. -15.375

Negative number ⇒ sign bit is 1
$15 = 1111$
$0.375 \times 2 = 0.75 \quad 0 \quad MSB$
$0.75 \times 2 = 1.5 \quad 1$
$0.5 \times 2 = 0.96 \quad 0$
$0.5 \times 2 = 1.0 \quad 1$

$0.375 = 0101$

$15.375 = 1111.0101$
Normalized we have, 1.1110101×2^3
$E = 3 \Rightarrow$ 19in excess-16
$E = 10010$
$-15.375$is represented as1 10010 11101010

c. 0.125

Positive number ⇒Sign bit is 0
$0 = 0000$
$0.125 \times 2 = 0.25 \quad 0 \quad MSB$
$0.25 \times 2 = 0.5 \quad 0$
$0.5 \times 2 = 1.0 \quad 1$

$0.125 = 0.001$
Normalized we have, 1.0×2^{-3}
$E = -3 + 16 = 13 = 01101$

0.125 is represented as 0 01100 00000000

2. Suppose the following 14-bit floating-point representation patterns, what decimal numbers do they represent?

a. 01000011000000
b. 10111010000000
c. 11111000000001

Solutions:

a. 010000110 00000

Sign bit is $S = 0$ ⇒positive number
$E = 10000 = 16$ (in normalized form)
Fraction is 1.11 (with an implicit leading 1) $= 1 + 1 \times 2^{-1} + 1 \times 2^{-2} = 1.75$
The number is $+1.75 \times 2^{(16-16)} = +1.75 \times 2^{0} = +1.75$

b. 10111010000000

Sign bit $S = 1$ ⇒ negative number
$E = 01110 = 13$ (in normalized form)
Fraction is 1.1 (with an implicit leading 1) $= 1 + 1 \times 2^{-1} = 1.5$
The number is $-1.5 \times 2^{(13-16)} = -1.5 \times 2^{-3} = -1.625$

c. 1 11110 00000001

Sign bit $S = 1$ ⇒ negative number
$E = 11110 = 30$ (in normalized form)
Fraction is 1.00000001 (with an implicit leading 1) $= 1 + 2^{-8}$
The number is $-(1 + 2^{-8}) \times 2^{(30-16)}$

3.3. Representation of Characters

Characters are represented using a chosen character encoding scheme (character set, charset, character map, or code page). The most commonly-used character encoding schemes are: ASCII, Latin-x for western European characters, and Unicode for internationalization.

3.3.1. ASCII Character Code

ASCII stands for American Standard Code for Information Interchange. The ASCII character code is a 7-bit code used to represent numeric, alphabetic, and special printable characters. It also includes codes for *control characters,* which are not printed or displayed but specify some control function. A 7-bit means a total of $2^7 = 128$ characters can be represented.

- Code numbers 0 to 31 and 127 are special control characters, which are non-printable (non-displayable)
- Code numbers 65 to 90 represent 'A' to 'Z', respectively.
- Code numbers 97to 122 represent 'a' to 'z', respectively.
  Code numbers 48 to 57 represent '0' to '9', respectively.

3.3.2. Latin-1

Latin alphabet No. 1 or Latin-1 in short, is the most commonly-used encoding scheme for western European languages. It has 191 printable characters from the Latin script, which covers languages like English, German, Italian, Portuguese and Spanish. Latin-1 is backward compatible with the 7-bit ASCII code. That is, the first 128 characters in Latin-1 (code numbers 0 to 127 (7FH)), is the same as ASCII. Code numbers 128 (80H) to 159 (9FH) are not assigned. Code numbers 160 (A0H) to 255 (FFH) represent other special characters like Å, Æ, Ç, ~, £, ¤, ¥, § and ©.

3.3.3. Unicode Character Code

Unicode originally uses 16 bits (called UCS-2 or Unicode Character Set - 2 byte), which can represent up to 65,536 characters. Before Unicode, no single character encoding scheme could represent characters in all languages. Unicode is backward compatible with the 7-bit ASCII and 8-bit Latin-1. That is, the first 128 characters are the same as ASCII and the first 256 characters are the same as Latin-1.
The original 16-bit range of U+0000H to U+FFFFH (65536 characters) is known as *Basic Multilingual Plane* (BMP), covering all the major languages in use currently. The characters outside BMP are called *Supplementary Characters*, which are not frequently-used.
UCS-4 (Universal Character Set - 4 Byte), uses 4 bytes (32 bits), covering BMP and the supplementary characters.

4. Digital Electronics

4.1. Boolean Algebra

Boolean algebra also known as the algebra of logic was developed by an English mathematician called George Boole. It deals with binary variables and logic operators operating on these variables. A binary variable has only two possible values 0 or 1. Logic operators operating on these variables are AND (·), OR (+) and NOT (‘). Operations are defined for the values 0 and 1 as follows:

AND	OR	NOT
$0 \cdot 0 = 0$	$0 + 0 = 0$	$\bar{0} = 1$
$0 \cdot 1 = 0$	$0 + 1 = 1$	$\bar{1} = 1$
$1 \cdot 0 = 0$	$1 + 0 = 1$	
$1 \cdot 1 = 1$	$1 + 1 = 1$	

4.1.1. Laws and Theorems of Boolean Algebra

- **Commutative law**
  $A \cdot B = B \cdot A | A + B = B + A$
- **Associative law**
  $(A \cdot B) \cdot C = A \cdot (B \cdot C) | (A + B) + C = A + (B + C)$
- **Distributive law**
  $A(B + C) = (AB) + (AC) \;\; | \;\; A + (BC) = (A + B)(A + C)$
- **Identity law**
  $A \cdot 1 = A \;\; | \;\; A + 0 = A$
- **Redundancy law**
  $A \cdot 0 = 0 \;\; | \;\; A + 1 = 1$
- **Complement law**
  $A \cdot \bar{A} = 0 \;\; | \;\; A + \bar{A} = 1$

- **Idempotence law**
  $A \cdot A = A \mid A + A = A$

- **Absorption law**
  $A(A + B) = A \mid A + (AB) = A$

A	B	$\bar{A}$	$\bar{A}B$	$A + \bar{A}B$
0	0	1	0	0
0	1	1	1	1
1	0	0	0	1
1	1	0	0	1

- **Involution law**
  $\bar{\bar{A}} = A$

- **No name**

$A + \bar{A}B = A + B \mid A(\bar{A} + B) = AB$

$AB + A\bar{B} = A \mid (A + B)(A + \bar{B}) = A$

A	B	C	AC	$B + AC$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

- **De Morgan's Theorems**

$\overline{AB} = \bar{A} + \bar{B} \mid \overline{A + B} = \bar{A} \cdot \bar{B}$

4.1.2. Simplifying Boolean Expressions

Simplifying a Boolean expression consists of using the laws of Boolean algebra to write the expression in its simplest form.

Example 1: Simplify the expression $AB + A(B + C) + B(B + C)$
Solution:
$AB + A(B + C) + B(B + C) = AB + AB + AC + BB + BC$

$= AB + AB + AC + B + BC$ $(BB = B)$
$= AB + AC + B + BC$ $(AB + AB = AB)$
$= AB + AC + B$ $(B + BC = B)$
$= B + AC$ $(AB + B = B)$

Example 2: Simplify the expression
$A\bar{B} + A(\overline{B + C}) + B(\overline{B + C}) = A\bar{B} + A(\bar{B}\bar{C}) + B(\bar{B}\bar{C})$ De Morgan's
$= A\bar{B} + A\bar{B}\bar{C} + B\bar{B}\bar{C}$
$= A\bar{B} + A\bar{B}\bar{C} + 0\bar{C}$ $(B\bar{B} = 0)$
$= A\bar{B} + A\bar{B}\bar{C} + 0$ $(0C' = 0)$
$= A\bar{B}(1 + C)$ $(1 + C' = 1)$
$= A\bar{B} \cdot 1$ $(A\bar{B} \cdot 1 = A\bar{B})$
$= A\bar{B}$ $(A\bar{B} \cdot 1)$

Example 3: Use De Morgan's theorems to simplify the following
i) $(\overline{A + \bar{B}C + C\bar{B}})$
ii) $(\overline{A\bar{B} + A\bar{C}})$

Example 4: Show that
i) $A + AB = A$
ii) $(\overline{\overline{A + B}}) = \bar{B}(\bar{A} + B)$
iii) $(A + B)(A + \bar{B}) = A$

4.1.3. Constructing Truth Tables

A truth table shows the output for all possible values of the input variables for a given Boolean expression. To construct a truth table, we evaluate the Boolean expression for all possible combinations of values for the input variables. The number of possible combinations is always equal to 2^n where n is the number of input variables.

Example 1: Construct a truth table for the expression $A + \bar{A}B$
We have 2 input variables A and $B \Rightarrow$ we will have $2^2 = 4$ different combinations.

Example 2: Construct a truth table for the expression $B + AC$
3 input variables $\Rightarrow 2^3 = 8$ different combinations

Example 3: Construct the truth tables for the following
i) $\bar{A}B + A\bar{B}$
ii) $\bar{A}BC + AB\bar{C}$ [$B(A$ xor C)]
iii) $A(A + \bar{C})(A + B)$

4.2. Logic Gates and Circuits

Logic gates and circuits are simply ways of representing Boolean functions and expressions electronically.

2.1. Logic Gates

A logic gate is an electronic device that implements a simple Boolean function. Logic gates carry out the operations that the microprocessor performs. When a microprocessor is designed, a logic gate cell library (*collection of all low level logic functions used to implement the logic*) is also deeply planned and developed.
There are six basic logic gates: AND, OR, NOT, NAND, NOR and XOR.

a. AND Gate

It is a logic gate whose output is “true” only when both inputs are “true”. If neither or only one of the inputs is “false”, the output is “false”. The following illustration and table show the circuit symbol and logic combinations for an AND gate.

Logic AND

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

b. OR Gate

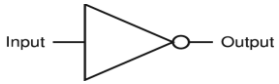
It is a logic gate whose output is “true” when either or both of the inputs are “true”. If both inputs are "false," then the output is "false."

Logic OR

A	B	A Or B
0	0	0
0	1	1
1	0	1
1	1	1

c. NOT Gate

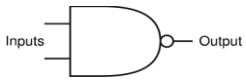
It is a logic gate whose output is “false” if its input is “true” and vice versa. It is called logical inverter, because it reverses the state of its input. A NOT gate accepts one input and produces a single output.



Logic NOT (inverter)

d. NAND Gate

The NAND gate operates as an AND gate followed by a NOT gate. The output is "false" if both inputs are "true." Otherwise, the output is "true.

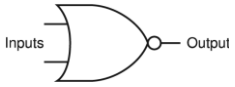


Logic NAND

A	B	¬(A B)
0	0	0
0	1	1
1	0	1
1	1	1

e. NOR Gate

A NOR gate is equivalent to an OR gate followed by a NOT gate. Its output is "true" if both inputs are "false." Otherwise, the output is "false."



Logic NOR

A	B	¬(A+B
0	0	0
0	1	1
1	0	1
1	1	1

f. XOR Gate

Exclusive-OR is a logic gate whose output is “true” when one but not both of its inputs is “true”. The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that the output is “true” if the inputs are different, but “false” if the inputs are the same.

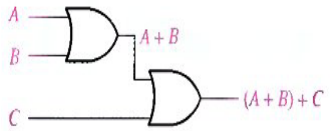
Logic XOR

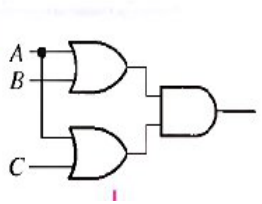
2.2. Logic Circuits

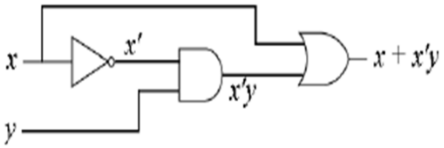
Circuits are made by combining two or more logic gates. Gates are combined into circuits by using the output of one gate as the input for another.

Examples

A	Output
0	1
1	0



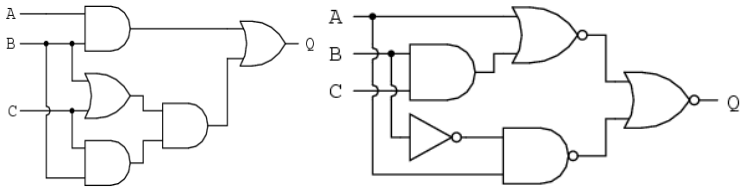




Exercise 1: Construct logic circuits for the following expressions

1) $\bar{A}B + A\bar{B}$
2) $(A + B)(\overline{A + \bar{B}})$
3) $\bar{A}(\bar{B} + A)$
4) $(\bar{A} + B)(\overline{ABC})$
5) $(A + \bar{B})AC$

Exercise 2: What are the outputs for the following logic circuits



CHAPTER 2: SYSTEM SOFTWARE

Introduction

Computer hardware require instructions to perform tasks. These instructions are provided by the programs or software installed in them. Software refers to the collection of computer programs and data that run on a computer, and which make the hardware useful. They are the intangible components of the computer system.

Computer software has two major categories namely system software and application software.

System software control and coordinate computer resources (hardware and operations) so that the computer user and applications can smoothly interact. They help the computer carry out its basic operating tasks. System software are designed to perform computer related tasks. They include operating systems, firmware, utility programs, device drivers, library programs and language translators.

1. Operating System

The operating system is the essential software that is required for a computer to become operational. It is the software layer that is on top of the hardware to provide functionality to computer components, manage the hardware and serve as interface between the computer user and the computer.

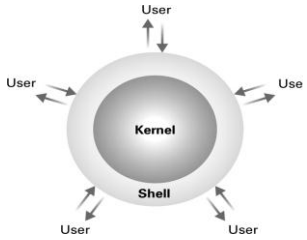
***Definition**: An operating system is software that manages the computer hardware resources and provides basic interface for execution of various application software.*

The operating system is called a virtual machine as it hides the complexity of the hardware and presents the user with an interface that is easier to understand and work with.

The operating system is stored on disk, but it needs to be loaded into memory (RAM) once the computer is switched on and before any other program can be run. The term bootstrapping refers to the process of loading the operating system into a computer's memory. This process is done by a program called the bootstrap loader that is stored permanently in the computer's electronic ROM chip. Examples of operating systems are Windows (95, 98, 2000, XP, Vista, 7 and 8), Macintosh Operating System (Mac OS), Linux and UNIX.

1.1. Operating System Structure

The operating system can be divided into two main components: kernel and command interpreter (shell).



OS structure

a. OS Kernel

The kernel is the central part of an operating system that is running at all times on the computer. It loads first and remains in memory as long as the computer is on. It consists of utilities (file manager, memory manager, device drivers, process manager etc.) that perform basic required functions. In many operating systems, only the kernel can access hardware directly.

b. The Shell

The shell, command shell or command interpreter, is the part of the operating system that provides an interface between users and operating system of a computer to access the services of the kernel. It understands and executes commands that are entered interactively by a human being or from a program.

There are two types of shell:

1. Command-line shell eg. Bash (sh), Command Prompt (cmd), C shell, Bourne shell, Korn shell (ksh) etc..
2. GUI-shell eg. Windows Explorer or Windows Shell

A third type of shell is recently developed - a GCLI (Graphical Command Line Interface) shell. A GCLI shell combines the features of both CLI and GUI shell and provides an interface which is both user-friendly and powerful.

When a user logs in, a shell is started up.

c. System Calls

The interface between the OS and the user program is defined by a set of instructions called system calls. User programs use system calls to talk with the operating system.

***Definition**: System call is the mechanism by which a program requests a service from the operating system's kernel.*

When a running program needs a service from the operating system, it calls a system call.

1.2. Operating System Functions

1.2.1. Memory Management

For a program to be executed, it must be found in main memory (RAM). In a multiprogramming environment in which several programs can reside in memory at the same time, every program and its data must be protected from the actions of other programs. Memory management keeps track of what programs are in memory and where in memory they reside.

Memory allocation determines where a program as well as its related data will be kept in memory for execution. Memory needs to be allocated efficiently to pack as many jobs in memory as possible. Memory can be subdivided into segments or frames (pages).

a. Segmentation

Segmentation is when memory is divided into variable sized units called segments. When segments are used, memory allocation can be done in three different ways:

- First fit allocates the first free segment that is large enough for the new process.
- Best fit allocates the smallest block among those that are large enough for the new process.
- Worst fit allocates the largest block among those that are large enough for the new process.

b. Paging

Paging is when memory is divided into fixed-size units called frames. Jobs are broken up into blocks of same size as frames called pages which are allocated a number of frames. The OS then uses a *page table* to map program pages to memory frames. The pages for each job could be in logical order or they may be scattered about wherever there is a free frame.

c. Virtual Memory

A program may require more memory than it is available. To solve this problem, virtual memory is used. Virtual memory is part of the hard disk that is used as an extension of RAM. It is slower, but it is considerably bigger. As execution goes on, data is being swapped between RAM and virtual memory.

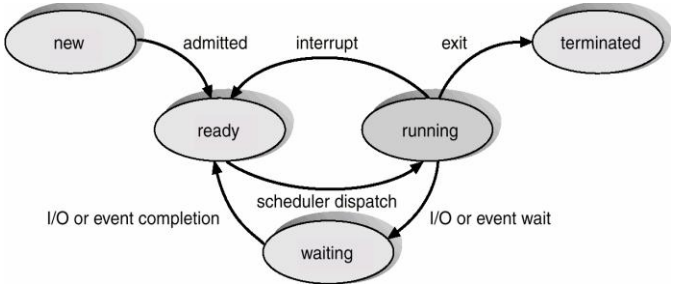
When a program is running, only the pages that contain the necessary data are kept in RAM while those that are not needed are kept on disk. For example, a program that has been minimized for a long time may be transferred to virtual memory so as not to fill up the main memory.

Disk thrashing occurs when the OS has to spend a considerable proportion of its time swapping data between virtual and real memory.

1.2.2. Process Management

A process is a program in execution. It consists of the program's instructions and the resources allocated to it for execution. A program is static while a process is active. The operating system performs process management to carefully track the progress of a process and all of its intermediate states.

A process changes state as it executes. The different states a process can have are shown in the diagram below.



Process states

- New - The process is being created.
- Ready - Process has all needed resources - waiting for CPU only.
- Running - Instructions are being executed.
- Waiting - Process is waiting for some event to occur (human, hardware or another process)
- Terminated - Process has finished execution

***Definition**: The turnaround time for a process is the amount of time between the time the process arrives the ready state and the time it exits the running state for the last time.*
In other words, it is the length of time it takes to run a process from initialization to termination, including all the waiting time.

***Question**: What is a scheduler?*

a. Process Control Block

The operating system manages a large amount of data about a process like the program counter (PC), CPU registers, memory management information, I/O status, scheduling data and process state. This data is stored in a data structure called a process control block (PCB) also called process descriptor or state vector. Every process has its PCB and each time a process is moved to the running state, its register values are loaded into the CPU while register values for the currently running process are stored into its PCB. This exchange of information is called a context switch.

b. Process Synchronization

In a multi-tasking system, processes compete for resources. A resource is anything that is required by a process to accomplish its task (processor, memory, I/O device, bus, file etc). Some resources can only be used in a non-sharable or exclusive mode. That is, they cannot be used by more than one process at a time. Such resources are known as critical resources. A critical section is a part of a program where it has access to a non-sharable (critical) resource. To prevent two or more processes from entering their critical sections over the same resource, processes must be synchronized.

***Definition**: Process synchronization is about getting processes to coordinate together in order to avoid two or more processes entering into critical section over the same resource.*

If processes are not synchronized, it leads to deadlock and starvation.

c. Deadlock

Deadlock is a permanent blocking of a set of processes competing for resources. It is a situation in which each of two processes is waiting for the other to do something; thus, neither one can proceed. For a deadlock to occur, the following four conditions must hold.

- ✓ Mutual Exclusion: At least one resource must be held in a non-sharable way.
- ✓ Hold and Wait: A process must be holding a resource and waiting for another.
- ✓ No Preemption: No resource can be forcibly removed from a process holding it.
- ✓ Circular Wait: A waits for B, B waits for C, C waits for A.

Deadlock can be prevented by ensuring that one of the above conditions does not hold. For example using pre-emptive scheduling.

d. Starvation

Starvation is a situation where a task can never finish because it can never get a necessary resource such as a large block of memory. The operating system should detect such tasks and do its best to allocate the resources that they need.

1.2.3. CPU Scheduling

CPU scheduling is the act of determining which process in the ready state should be moved to the running state. It decides which process in memory is to be executed by the CPU at any given moment. Scheduling decisions may take place when a process:

1. Switches from running to waiting state
2. Switches from running to ready state
3. Switches from waiting to ready
4. Switches from running to terminated

Scheduling can be preemptive or non-preemptive.

- ✓ Preemptive scheduling is scheduling in which the currently executing process is forced to give up the CPU. Scheduling under (2) and (3) is preemptive.

- ✓ Non preemptive scheduling is scheduling in which the currently executing process gives up the CPU voluntarily. Scheduling under (1) and (4) is non-preemptive.

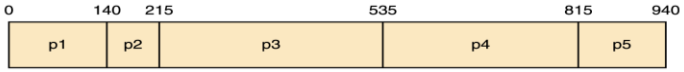
There exist different algorithms used for scheduling. Examples are first come, first served, shortest job first and round robin algorithms.

a. First Come, First Served

First come first served algorithm moves processes to the CPU in the order in which they arrive in the ready queue. It is non-preemptive. As such, when a process has the CPU it runs to completion before giving up the CPU.
Consider that the following processes arrive in the order they are given below.

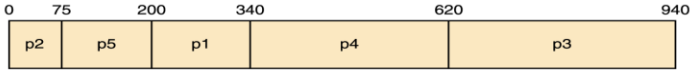
Process	Service time
P1	140
P2	75
P3	320
P4	280
P5	125

Using FCFS algorithm to schedule the processes, we get



b. Shortest Job First

Shortest job first (SJF) algorithm looks at all the processes in the ready state and dispatches the one with the smallest execution time. It is also generally implemented as a non-preemptive algorithm. Using the SJF algorithm, we have

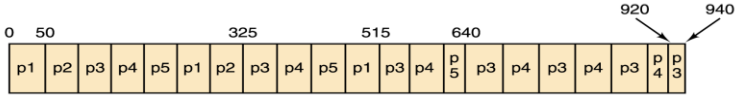


Disadvantage

- Some jobs may never be executed as they have a high service time (starvation).

c. Round Robin

Round-robin algorithm distributes time equitably among all ready processes by establishing a particular time slice (or time quantum), during which each process executes. At the end of the quantum, the process is preempted. It returns to the ready state to allow another process its turn. Using round-robin algorithm with time slice of 50, the above processes will be scheduled as follows:



1.2.4. Management of I/O Devices

a. Interrupts

An interrupt is a signal generated by hardware or software that causes the CPU to suspend what it is doing to handle another task of higher priority. Any event that will cause an interrupt is called an interrupt request (IR). For example, a key pressed on the keyboard. An interrupt handler or interrupt service routine (ISR) is a program that services an interrupt request. It contains the actions that will be executed for a given interrupt request.

b. Polling

c. Buffering

A buffer is an area of memory used to temporarily store data while it is being moved from one place to another.

1.2.5. File Management

The file system is the portion of the operating system that manages how files are stored. Examples of file systems are FAT used in MS DOS, NTFS used in Windows, ext2 used in Linux and HPFS used in OS/2. The operating system is responsible for the following activities in connection with file management:

- File creation and deletion.
- Directory creation and deletion.
- Support of primitives for manipulating files and directories
- Mapping files onto secondary storage.
- File backup on stable (nonvolatile) storage media.

1.2.6. Providing User Interface

Working on a computer, a user has to interact with the computer. A user interface is the means of communication or interaction between the user and the computer. The operating system provides this means that enables an individual to see and work when using a computer.
Different operating systems provide different types of user interfaces.

a. Command Line Interface (CLI)

A command line interface allows the user to interact with the computer by typing the commands in a specified format. It provides a prompt through which the user types the commands. Here the user mostly makes use of the keyboard.
In this type of interface, the user has to remember the name and format of the commands. Spelling mistakes and deviations in format lead to errors and the task is not performed. Examples of operating systems that provide a command line interface are MS DOS, early versions of UNIX and Linux.

b. GraphicalUserInterface (GUI)

A graphical user interface allows the user to interact with the computer through graphical items such as icons, menus, dialog boxes, etc. Here, the user mostly makes use of the mouse to point and click on these graphical items. GUI is also known as WIMP system where WIMP stands for windows, icons, menus and pointers.
This type of user interface requires a lot of memory space to store the graphics and can cause machines with low processing power to be slow. Examples of operating systems with a graphical user interface are Windows, Macintosh operating systems, some versions of Linux and UNIX.

c. VoiceRecognition Interface

A speech recognition interface allows the user to give verbal commands to the computer. The user communicates with the computer through natural language. They are also called natural language interface

1.2.7. Providing Security

Security is the most desirable characteristic of any operating system. An operating system should provide a means for safeguarding system resources from unauthorized users and protection of one user's resources from other users of the system.

Popular operating systems offer security features through incorporation of the following:

- Login name.
- Login password.
- Read/write access file permissions.
- Encryption of data
- Virus protection software layers.

1.3. Types of Operating Systems

1.3.1. Single User OS

A single user operating system is an operating system which allows only one user to work on the system at a time. No two or multiple users can work on the system simultaneously.

Examples are Control Program for Microcomputers (CP/M) and Microsoft Disk Operating System (MS DOS).

1.3.2. Multi User OS

A multi-user operating system is an operating system which allows multiple users to work on the system simultaneously. This type of operating system is larger and more complex than a single user operating system.

Features of multi-user operating systems, which are not provided in single user operating systems, are:

- Time sharing (CPU devotes time to all the users in round robin fashion).
- Tight security features.
- Resource sharing among users.
- System administrator privileges

Examples of multi-user operating systems are Linux, Unix and Virtual Machine System (VMS).

1.3.3. Single task OS

A single task operating system allows a user to execute one and only program at a time. The user cannot run two or more programs at the same time. It is not possible to be preparing a worksheet on the computer while printing a report or listening to music. Once a user invokes a program, the computer gets dedicated to that task only.

CP/M and MS-DOS are examples of single user single task operating systems.

1.3.4. Multiprogramming OS

A multi-programming operating system is an operating system that allows multiple programs to be held in main memory at the same time. The concept of multi-programming is that the operating system keeps several jobs in memory simultaneously and decides which can be executed at a given moment.

1.3.5. Multitasking OS

A multitasking operating system allows a user to execute more than one program at a time. It allows a user to be preparing a worksheet on the computer while printing a report or listening to music. Multitasking is an extension of multiprogramming as two programs cannot be executed simultaneously if they are not found in memory at the same time.

Windows Me, Windows-XP, Macintosh operating system, OS/2 are examples of single user, multitasking operating systems.

1.3.6. Embedded OS

An embedded operating system is an operating system that is used in an embedded system. An embedded system is a small computing device that is built into a larger equipment often as a single chip and dedicated to a given task. Embedded systems control many devices in use today such as digital watches, mobile phones, microwave ovens, washing machines, vehicles, photocopiers, cameras and process controllers.

Embedded OS are ROM based. That is, they cannot be modified as ROM is read only.

1.3.7. Network OS

A network operating system is an operating system which includes networking features. It contains special functions, protocols and device drivers that enable the computer to be connected to a network.

Examples of network operating systems are Windows-NT, Windows-2000 server, Windows server 3000, Novell Netware and Artisoft LANstatic.

Some multi-purpose operating systems like Windows XP, Windows 7 and Mac OS 10, come with capabilities that enable them to be described as network operating systems.

2. System BIOS

Basic input output system (BIOS) is software that contains hundreds of programs that allow for communication between the CPU and devices. It is stored on ROM, which is a permanent chip on the motherboard.

There are three kinds of BIOS for hardware devices:

- ✓ Permanent never changing BIOS for never changing hardware like the keyboard
- ✓ BIOS for hardware that changes occasionally. It requires extra volatile information so it is stored on a separate chip called CMOS (**C**omplementary **M**etal **O**xide **S**emiconductor).

3. Utility Programs

Utility software is used to enhance the operating system, or in some other way improve the usefulness of the system. They help analyze, configure, optimize and maintain the computer. Rather than providing user-oriented or output-oriented functionality, utility software focuses on how the computer infrastructure operates. Most major operating systems come with several pre-installed utilities. Examples of utility software include: disk defragmenters, backup utilities, disk compression utilities, disk cleaners, file managers, disk formatters and virus checkers.

- ✓ Disk defragmenters detect computer files whose contents are broken across several locations on a disk, and move the fragments to one location to increase efficiency.
- ✓ Disk cleaners find and delete files that are unnecessary to computer operation, or take up considerable amounts of space. They help users decide what to delete when their hard disk is full.
- ✓ Backup utilities make copies of all information stored on a disk, and restore either the entire disk (e.g. in an event of disk failure) or selected files (e.g. in an event of accidental deletion).
- ✓ Disk compression utilities reduce the space that a file takes up on disk, increasing the capacity of the disk.
- ✓ File managers provide a convenient method of performing routine data management tasks, such as deleting, renaming, moving, copying, merging, generating and modifying files.
- ✓ Disk partition utilities divide an individual drive into multiple logical drives, each with its own file system which can be mounted by the operating system and treated as an individual drive.
- ✓ Disk formatters
- ✓ Virus checkers prevent, detect, and remove malware.

4. Device Drivers

A device driver is software that allows interaction between the operating system and a hardware device. It is an interface for communicating with the device through the specific computer bus that the hardware is connected to. Without an appropriate device driver the system cannot communicate with a device, rendering the device useless. Installation of device drivers usually happens automatically when hardware is connected (plug n play), or from a CD provided with the device. Sometimes a device driver needs to be updated to stay functional.

5. Library Programs

6. Language Translators

A language translator is a computer program that translates program instructions from one programming language to another. There are three types of language translators: compilers, interpreters and assemblers.

- ✓ Compilers translate instructions written in a high-level language into machine language instructions.
- ✓ Interpreters translate instructions written in a high-level language into machine language instructions and execute them, one line at a time.
- ✓ Assemblers translate instructions written in assembly language to machine language instructions.

CHAPTER 3: COMMUNICATION SYSTEMS

1. Computer Networks

A computer network is a collection of computers and other devices that are connected together so they can communicate and share resources. The smallest network can be as simple as two computers linked together. The resources shared include files, folders, printers, disk drives and anything else that exists on a computer. Any computer or device on a network is called a node.
Networking is the term that describes the processes involved in designing, implementing, upgrading, managing and otherwise working with networks and network technologies

1.1. Types of Computer Networks

Different criteria exist for classifying computer networks. According to geographical area covered, networks can be classified as local area networks, wide area networks, metropolitan area networks, personal area networks etc.

1.1.1. Local Area Network

A local area network (LAN) is a network that is used for communication among computer devices, usually within an office building or home. It enables the sharing of resources such as files or hardware devices that may be needed by multiple users in an organization. A LAN is limited in size, spanning a few hundred meters, and not more than a mile. It is fast, with speeds from 10 Mbps to 10 Gbps. An example of LAN is the network in the Multimedia Resource Centre.

1.1.2. Metropolitan Area Network

A metropolitan area network (MAN) is a large computer network that usually spans a city or a large campus. It is optimized for a larger geographical area than a LAN, ranging from several blocks of buildings to entire cities. A MAN typically covers an area of between 5 and 50 km diameter. An example of a MAN is a cable TV network.

1.1.3. Wide Area Network

A wide area network (WAN) covers a large geographic area such as a country, a continent or even the whole world. It is a distributed collection of LANs. That is, it connects two or more LANs together. This is done using devices such as bridges, routers or gateways, which enable them to share data. The largest and most well-known example of a WAN is the Internet.

1.1.4. Personal Area Network

A personal area network (PAN) is a network that is used for communication among computers and computer devices in close proximity of around a few meters within a room. A PAN usually includes laptops, mobile phones, personal digital assistants, digital cameras and headsets. It can be used for communication between the devices themselves, or for connection to a larger network such as the Internet. The most popular is the Wireless PAN (WPAN), a Bluetooth connection between two laptop computers or phones.

1.2. Computer Network Components

All types of computer networks require special networking software and hardware to allow different computers to communicate with each other. The most important software component required for a network is the network operating system (NOS) while there are many types of hardware devices which are either installed or connected to the computer terminals in order to construct a network.

1.2.1. Network Operating System

A network operating system is an operating system which includes networking features. It contains special functions, protocols and device drivers that enable the computer to be connected to a network. NOS provide the ability to share resources and the ability to manage a network name directory, security, and other housekeeping aspects of a network.
Examples of network operating systems are Windows-NT, Windows-2000 server, Windows server 3000, Novell Netware and Artisoft LANstatic.
Some multi-purpose operating systems like Windows XP, Windows 7 and Mac OS 10, come with capabilities that enable them to be described as network operating systems.

1.2.2. Network Interface/Adapter Card

A network interface card (NIC) provides the physical interface (link) between the computer and the communication medium. A NIC manages the communication and network protocol for the PC.It prepares data, sends data and controls the flow of data. It plugs into the system board and provides ports for connection to the network. A NIC is also called a LAN card or network adapter card. There are two kinds of NIC: wired NIC for wired networks and wireless NIC (WNIC) for wireless networks. A NIC may be designed as an Ethernet card, a Token Ring card, or an FDDI card (but not all three).

1.2.3. Hub

A hub is a device that works as central connecting point for multiple computers in a network. It has ports to which the computers in the network are connected. Data sent to the hub is broadcasted to all the ports but, only the destination computer receives it. There are three kinds of hubs:

- ✓ Passive hubs which only split the transmission signal so it can be sent to all the ports
- ✓ Active hubs (also called Multiport Repeaters) which regenerate data bits to maintain a strong signal over extended cable lengths
- ✓ Intelligent hubs (also called Concentrators) usually have their own microprocessor chips and network operating systems. They can be managed remotely on the network.

1.2.4. Switch

A switch is used at the same place as a hub but the difference between the two is that a switch has a switching table within it. A switching table stores the Media Access Control (MAC) address of every computer connected to the switch and sends the data only to the requested address, unlike the hub which broadcasts the data to all the ports. Switches can therefore be considered as an advanced form of hubs.
A MAC address is a built-in number (i.e. set by the manufacturer) consisting of 12 hexadecimal digits that uniquely and permanently identifies the network adapter of a computer. Examples of a MAC addresses are 00-14-22-DA-67-15 and 00-13-02-31-E8-BA.MAC address is also called the physical address. Under Windows, the MAC address of a computer can be displayed by typing *ipconfig/all* at a Command prompt.

1.2.5. Repeater

A repeater is a device used to expand the boundaries of a wired or wireless network. With physical media, data transmissions can only span a limited distance before the quality of the signal degrades. Repeaters are used to preserve signal integrity and extend the distance over which data can safely travel by regenerating the signals they receive. Active hubs are considered as repeaters (multiport repeaters).

1.2.6. Bridge

A bridge, also called a layer 2 switch, is a device used to create a connection between two separate computer networks or to divide one network into segments. Creating multiple segments in a local network reduces the network traffic making the network to be faster and more efficient. A bridge

performs its segmenting function by examining the data packet and forwarding it to other physical segments only if necessary.

1.2.7. Router

A router is a device that joins several networks together and is responsible for routing data from one network to another. It keeps track of the IP addresses of the computers on the networks connected to its network interface cards and directs data packets appropriately. It is more powerful than a bridge because instead of just choosing network segments based on previous traffic, a router can look up the best route for a packet to take. Routers can be computers with operating systems and special network software, or they can be other dedicated devices built by network manufacturers. The Internet relies heavily on routers.

1.2.8. Modem

A modem (modulator/demodulator) is a device that encodes data for transmission over a particular medium, such as telephone line, coaxial cable, fiber optics, or microwaves. It converts digital signals from a computer to analog signals or waveform for transmission over a medium (modulation) and converts analog signals from the medium to digital signals understandable by the computer(demodulation).

Common types of modems are:

- ✓ Dial-up Modem
- ✓ Cable Modem
- ✓ DSL Modem
- ✓ Sat modem

1.2.9. Multiplexer

A multiplexer abbreviated MUX, is a device that takes input signals from different sources and transmits them over a single transmission line. This process is known as multiplexing. There are different types of multiplexing:

- ✓ Frequency-division multiplexing (FDM), in which the carrier bandwidth is divided into sub channels of different frequency widths, each carrying a signal at the same time in parallel.
- ✓ Time-division multiplexing (TDM), in which the multiple signals are carried over the same channel in alternating time slots.
- ✓ Code-division multiplexing (CDM), in which the multiple signals are carried over the same channel but every signal is coded differently.

1.2.10. Cables

Cables are used to link computers in a LAN. There are three types of cables commonly used:

- ✓ Coaxial cable
- ✓ Twisted pair cable
- ✓ Fiber optic cable

1.2.11. Gateway

A gateway is a device that connects two dissimilar computer networks using direct and systematic translation between protocols. A gateway translates outgoing network traffic to the protocol needed by the destination network. The term gateway is also sometimes loosely used to describe any device that acts as the entry or exit point for a network.

1.3. Network Topologies

Network topology is the layout or arrangement of the components of a network. It refers to the way in which computers and cables are connected together to build a network. Different types of topologies exist.

1.3.1. Bus Topology

In bus topology, all computers are connected to a single cable (trunk or backbone) known as bus, by a transceiver either directly or by using a short drop cable. Bus transmits in both directions such that any transmission can be received by all stations. All ends of the cable must be terminated, that is plugged into a device such as a computer or terminator, to avoid signals from bouncing back.

Bus topology

a. Advantages

- ✓ Easy and inexpensive to set up as little cabling is required
- ✓ Easy to include additional stations without disrupting the network
- ✓ Failure of one node does not affect network

b. Disadvantages

- ✓ High rate of data collision
- ✓ Fails if there is any damage to the bus
- ✓ Any break in the bus is difficult to identify

1.3.2. Star Topology

In a star topology, all the computers are connected to a central device which could be a computer, a hub or a switch. Any communications between computers in this topology must pass through the central node. As such, the central node controls all the activities of the network.

Star topology

a. Advantages

- ✓ Breakdown of a node does not affect the network
- ✓ No disruption of the network when connecting or removing devices
- ✓ It is easy to detect faults

b. Disadvantage

- ✓ Failure of the central node affects the entire network
- ✓ It is costly due to the amount of cables required to connect the devices

1.3.3. Ring Topology

In ring topology, all the nodes are connected in the form of a closed loop such that each node is connected to two others. It uses an empty data packet called a token and a special protocol called token ring. Packets travel around the ring in a clockwise direction. To transmit, a node requires an empty token.

Ring topology

a. Advantage

- ✓ No collision as a station needs the token to transmit
- ✓ Each computer acts like a repeater so signals are not attenuated

b. Disadvantage

- ✓ If a node in the network fails, the entire network fails
- ✓ Network is disrupted when additional stations are added

1.4. Network Technologies

1.4.1. Ethernet Network

Ethernet (IEEE 802.3 standard) is the most common and widely used technology to establish a local area network. An Ethernet network is formed by physically connecting the individual computer units to each other in a bus topology or a star topology. Ethernet's media access policy is CSMA/CD (Carrier Sense Multiple Access with Collision Detection).

- ✓ CS: means that a station listens to (senses) the medium and transmits only if medium is idle
- ✓ MA: means that any station can use (access) the medium
- ✓ CD: means that each station stops transmitting immediately it senses a collision

When a collision is detected, the two stations involved will retransmit after a random time wait created by a backoff algorithm.

1.4.2. Token Ring Network

Token ring (IEEE 802.5 standard) is a network technology developed by IBM in which computers are connected together in a ring. Token ring's media-access method is called token passing. A special message, called token, circulates along the ring from one computer to another and each computer can transmit only while it is holding the token. Information flows in one direction along the ring from source to destination and back to source. When a station wishes to transmit, it waits for the empty token to pass by. It seizes it and inserts data into it and then releases it to the medium. The token circulates until it gets to the destination computer that picks it and retrieves the data. After retrieving the data, it regenerates the token and sends it back to the medium.

1.4.3. Fiber Distributed Data Interface

FDDI is a network technology that uses fiber-optic cables in a ring topology with dual rings on which information can travel in opposite directions. The media access method for FDDI is token passing. The primary ring is used for data transmission, and the secondary ring remains idle. Because of this double ring topology, if a station fails or a cable becomes damaged, the dual ring is automatically wrapped around itself, forming a single ring. This prevents downtime as a result of a failed machine or faulty wiring.

1.4.4. Wireless Network Standards

Wireless networks are established without physical wiring techniques involved. They use radio and infrared signals and are based around one of these technologies: Bluetooth, Wi-Fi, WiMax, terrestrial microwaves and satellite.

a. Bluetooth

Bluetooth is a low power, short-range wireless technology largely used to interconnect computing devices into a personal area network. It is based on IEEE standard 802.15 which gives specifications for Wireless Personal Area Network (WPAN).

b. Wi-Fi

Wi-Fi stands for Wireless Fidelity. It is based on a set of wireless networking technologies known as 802.11. These include 802.11b, 802.11a, 802.11g and 802.11n. The range of Wi-Fi network transmission is about 30-40m indoors and up to about 100m outdoors.
The table below shows the different 802.11 standards for wireless networking.

Specification	Popular name	Frequency	Speed	Compatible with
802.11a	Wireless-A	5 GHz	54 Mbps	
802.11b	Wireless-B	2.4 GHz	11 Mbps	Wireless-A
802.11g	Wireless-G	2.4 GHz	54 Mbps	Wireless-B, -G
802.11n	Wireless-N	2.4 GHz	100 Mbps	Wireless-B, -G, -N

c. WiMax

WiMax stands for Worldwide Interoperability for Microwave Access. It is based on IEEE standard 802.16 and facilitates high speed wireless network links to both fixed and mobile devices. The range of a WiMax wireless connection is around 3-10km.WiMax service providers are now just entering the market, offering customers an alternative to a DSL Internet connection.

1.5. Network Architectures

1.5.1. Client/Server Architecture

Client/server is a network architecture in which a more powerful computer called server is dedicated to serving less powerful computers called clients. Servers hold shared resources like files, programs and the network operating system. They provide access to network resources to all the users of the network. There are many different kinds of servers, and one server can provide several functions. For example, there are file servers, print servers, mail servers, database servers and Web servers. Users run applications on client workstations which rely on servers for resources such as files, devices and even processing power.
Internet services are organized according to a client/server architecture. Client programs, such as Web browsers and file transfer programs create connections to servers, such as Web and FTP servers. The clients make requests and the server responds to the requests by providing the services requested by the client.

1.5.2. Peer-to-Peer Architecture

Peer-to-peer (P2P) is a network configuration in which all the workstations (computers) have equal capabilities and responsibilities. Each workstation acts both as a server and a client. This means that any computer on the network can provide services to any other computer. Peer-2-peer is usually implemented where strict security is not necessary. P2Pnetworks are generally simpler and less expensive, but they usually do not offer the same performance under heavy loads.

Remark A hybrid network combines client/server and peer-to-peer architectures. It is the most commonly used network architecture.

1.6. Benefits and Limitations Of Computer Networks

1.6.1. Benefits

- ✓ Sharing devices such as printers saves money.
- ✓ Site (software) licenses are likely to be cheaper than buying several standalone licenses.
- ✓ Files can easily be shared between users.
- ✓ Network users can communicate by email and instant messaging.
- ✓ Data is easy to backup as all the data is stored on the file server.
- ✓ Organizations can organize videoconferences (videoconferencing)
- ✓ Employees can work from home (telecommuting)

1.6.2. Limitations

- ✓ Purchasing the network cabling and file servers can be expensive.
- ✓ Managing a large network is complicated. It requires training and a network manager usually needs to be employed.
- ✓ If the file server breaks down the files on the server become inaccessible. Email might still work if it is on a separate server. The computers can still be used but are isolated.
- ✓ Viruses can easily spread to other computers throughout the network, if one computer is infected..
- ✓ There is a danger of hacking, particularly with wide area networks. Security procedures are needed to prevent such abuse, eg a firewall.

2. Data Communication

Data communication refers to the exchange of data between two devices via some form of communication channel. In data communication the following basic terms are frequently used:

- ✓ Data: a collection of facts in raw form that becomes information after processing.
- ✓ Signal: an electric or electromagnetic encoding of data.
- ✓ Signaling: propagation of signals across a communication channel.
- ✓ Transmission: sending of data from one place to another by means of signals.

There are five basic components in a communication system.

- ✓ Data Source: creates data for transmission
- ✓ Transmitter: encodes data for transmission
- ✓ Communication channel: connecting medium between communicating devices
- ✓ Receiver: decodes transmitted signals back to data
- ✓ Destination: the final destination of the transmission

Example: John calls Peter on phone.
The data source is John, the transmitter is John's phone, the communication channel is the telephone cable or microwave, the receiver is Peter's phone and the destination is Peter.

2.1. Analog and Digital Signals

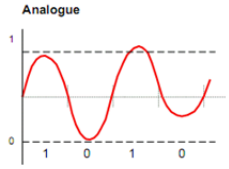
Data is transmitted from one point to another by means of electrical signals that may be in analogue or digital form.

2.1.1. Analogue Signals

An analog signal is one in which information is represented as a continuous variation of some physical property or quantity. Analog signals are continuous waves that carry information by varying the frequency or amplitude of the wave.

- ✓ When the amplitude of the signal is varied the technique is called amplitude modulation (AM)
- ✓ When the frequency of the signals is varied, the technique is called frequency modulation (FM).

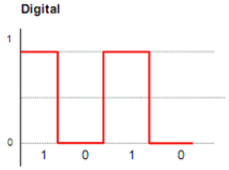
Human speech is an example of an analog signal. Telephone lines use analog signals because they were originally designed for speech.



Analogue signal

2.1.2. Digital Signals

A digital signal is one in which information is represented as a sequence of binary values 0 and 1.These two values represent two conditions, on or off, corresponding to two known levels of voltage or current.
Digital signals do not continuously vary as analogue signals. Signals are transmitted within the computer as digital signals. Systems that use digital technology are known as baseband systems.



Digital signal

2.2. Broadband and Baseband Transmissions

2.2.1. Baseband System

A baseband system is a single-channel system that supports a single transmission at any given time. In a baseband system, data is sent as a digital signal through the media as a single channel that uses the entire bandwidth of the media. Baseband communication is bi-directional, which means that the same channel can be used to send and receive signals. In Baseband, frequency-division multiplexing is not possible.

2.2.2. Broadband System

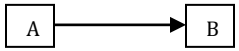
A broadband system is a system that supports multiple transmissions via multiple frequency channels. In a broadband system, data is sent in the form of an analog signal where each transmission is assigned a portion of the bandwidth. Broadband communication is unidirectional, so in order to send and receive, two pathways are needed. This can be accomplished either by assigning a frequency for sending and assigning a frequency for receiving along the same cable or by using two cables, one for sending and one for receiving.

2.3. Transmission Modes and Techniques

Transmission modes simply refer to the direction of flow of information between two communicating devices. It could be simplex, half duplex or full duplex.

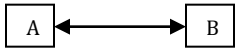
2.3.1. Simplex

In simplex mode, signals are transmitted in only one direction all the time. The flow of information is unidirectional from transmitter to receiver always. Examples are television broadcasting, computer to the printer connection and CPU to monitor communication.



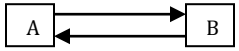
2.3.2. Half Duplex

In half duplex mode, signals can be transmitted in both directions but only one way at a time. The flow of information is bidirectional but information can only be sent if it is not being received. It is suitable for data transmission between a computer and dumb terminals. An example is the police radio (walkie-talkie).



2.3.3. Full Duplex

In full duplex mode, signals can be transmitted in both directions simultaneously. The communicating devices can transmit at the same time. The flow of information is bidirectional. It is suitable for interactive systems. An example is the telephone.



2.3.4. Parallel Transmission

Parallel transmission is the method of transferring several bits at the same time over separate channels. For example, eight separate channels will be required if a block of eight bits is to be transmitted in parallel. Parallel transmission is fast but it is suited only for short distances as cabling for long distances will be expensive. It is mainly used for connections within the computer and for connecting the computer to the printer.

2.3.5. Serial Transmissions

Serial transmission is the method of transferring data one bit at a time through the same channel. If a block of 8 bits is to be transmitted in series, the bits will be transmitted one after the other on the same channel. Serial transmission can be asynchronous or synchronous.

a. Asynchronous Serial Transmission

Asynchronous transmission describes the process where transmitted data is encoded with start and stop bits, specifying respectively the beginning and end of each character. Data is sent character by character with each character preceded by a start bit and a stop bit is added to the end. Other control bits like the parity bit are added to the group before the stop bit and small gabs are inserted to distinguish each group.

b. Synchronous Serial Transmission

Synchronous transmission describes a continuous and consistent timed transfer of data blocks. Data is sent as one long bit stream or block of data without start or stop bits and with no gabs. Upon reception, the receiver counts the bits and reconstructs bytes. It is essential that the same timing is maintained by both sender and receiver as there are no start and stop bits and no gaps. Another channel is therefore used to transfer timing signals to keep the both parties synchronized. Accuracy is dependent on the receiver keeping an accurate count of the bits as they arrive.

Serial transmission is slower than parallel transmission but it is suited for long distances. It is cheaper as only one transmission line is required. Synchronous transmission is faster than asynchronous transmission because fewer bits have to be transmitted; only data bits and no extra control bits. For this reason it is the choice for network communications links.

2.4. Communication Switching Techniques

Long distance transmission is done over a network of switched nodes. Data is routed by being switched from one node to another. Three switching techniques exist: packet switching, circuit switching and message switching.

2.4.1. Packet Switching

Packet switching is a switching method in which the message to be transmitted is broken into small data packets and sent over the network. Each packet contains a portion of data and some control information. The packets may take different routes to arrive their destination and they may arrive in any other. On arrival, they are put back into order and the message is reconstituted. Each packet is sent with a header address which tells what its destination is. The header address also describes the sequence for reassembly at the destination. One packet contains information on how many packets should be arriving. If a packet fails to arrive, the destination computer sends a message to the sender's computer asking it to send the missing packet again. This method is suitable for transmission of data.

2.4.2. Circuit Switching

Circuit switching is a switching method in which a dedicated communication path in physical form between two stations within a network is established, maintained and terminated for each communication session. This channel remains open throughout the communication process and cannot be used by anyone else. It has basically three phases: circuit establishment, data transfer and circuit disconnect. The message is sent without being broken up, so it is received in the order it was sent. This method was designed for voice transmissions. Telephone networks use circuit switching for transmission of phone calls.

2.5. Transmission Media

A transmission medium is the physical pathway that connects computers and other devices on a network. Each transmission medium requires specialized network hardware that is compatible with that medium, and most networks need to use a combination of transmission media types selected based on the network's needs. There are two categories of transmission media: guided and unguided media.

2.5.1. Guided Media

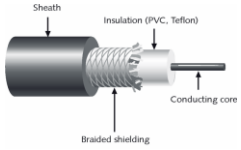
Guided media are the physical links through which signals are confined to narrow path. They are made up of an internal conductor bounded by jacket material. They are also called bounded or conducted media. Three common types of guided media are coaxial cable, twisted pair cable and fiber optical cable.

a. Coaxial Cable

Coaxial cable consists of an inner core and an outer flexible braided shield, both of conductive material separated by an insulator. The braided shield prevents the cable from picking up or emitting electrical noise. There are two types of coaxial cable:

- ✓ thinnet and

✓ thicknet.



Coaxial cable

To connect coaxial cable to devices, we need coaxial connectors. The most common type of connectors used today is the Bayone-Neill-Concelman, or BNC connector.

b. Twisted Pair Cable

Twisted-pair cable is the most common type of cabling used in LAN networks today. It consists of a pair or pairs of insulated wires twisted together. Cable twisting helps reduce noise pickup from outside sources and crosstalk on multi-pair cables. There are two types of twisted pair cables: shielded twisted pairs (STP) and unshielded twisted pairs (UTP).

Twisted pair cable uses RJ-14 and RJ-45 connectors

Twisted pair cable

✓ **Unshielded Twisted Pair**

UTP cables consist of 2 or 4 pairs of twisted cable. Cable with 2 pair use RJ-11 connector and 4 pair cable use RJ-45 connector. RJ stands for registered jack. There are five levels of UTP:

Category 1: These are used in telephone lines and low speed data cable.
Category 2: These cables can support up to 4 mps implementation.
Category 3: These cable supports up to 16 mps and are mostly used in 10 mps.
Category 4: These are used for large distance and high speed. It can support 20mps.
Category 5: This is the highest rating for UTP cable and can support up to 100mps.

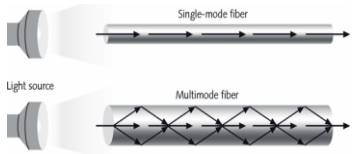
UTP can be connected as straight through or crossover. A straight-thru cable has identical ends. A crossover cable has different ends.

c. Fiber Optic Cable

Fiber optic cables use optical fibers that carry digital data signals in the form of modulated light pulses. An optic fiber consists of an extremely thin cylinder of glass, called the core, surrounded by a concentric layer of glass, known as the cladding. Each cable has two fibers - one to transmit and one to receive.

There are two types of optic fibers:

- ✓ A single mode fiber (SMF) uses a single ray of light to carry transmissions over long distances.
- ✓ A multi-mode fiber (MMF) uses multiple rays of light simultaneously with each ray of light running at a different reflection angle to carry transmissions over short distances.



Optic fiber

The light source can be LED (light emitting diode) or LD (laser diode)

2.5.2. Unguided Media

Unguided media do not use physical means to define the path to be taken. They provide a means for transmitting electromagnetic waves but do not guide them. They are also called unbounded media. Examples of unguided media are infrared waves, radio waves and microwaves.

a. Infrared

Infrared uses transmitters/receivers (transceivers) that modulate non-coherent infrared light. Infrared signals do not penetrate walls as such transceivers must be within line-of-sight either directly or via reflection. Line of sight is a type of propagation that can transmit and receive data only where transmit and receive stations are in view of each other without any sort of an obstacle between them.

b. Radio waves

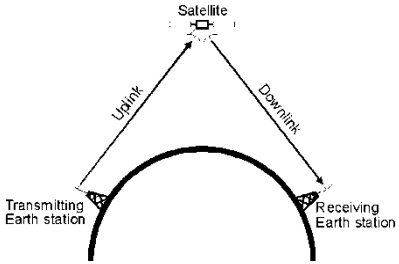
Radio wave systems transmit signals by modulation of electromagnetic waves with frequencies below that of visible light. Radio waves carry information by systematically changing some property of the radiated waves such as amplitude (AM radio), frequency (FM radio) and phase. Radio waves are omnidirectional. This means that signals spread out in all directions and can be received by many antennas.

c. Microwaves

Microwaves are electromagnetic radiations beyond the frequency range of radio and television. There are two types of microwave systems: terrestrial microwave systems and satellite systems.

✓ Terrestrial microwave systems are land-based. Microwaves being line-of-sight and traveling in a straight line, the earth's curvature poses a problem to long distance microwave transmissions. As such, long distance transmissions require directional antennas (repeaters)to be used at intervals of 25 to 30 kilometers between the transmitting and receiving end.

✓ Satellite systems use communication satellites to solve the problem posed by the earth's curvature to terrestrial microwave systems. A communication satellite is a microwave relay station placed in outer space. A microwave signal is transmitted from earth to the satellite which amplifies the signal and sends it back to earth. The earth station transmits the signal to the satellite on an up-link, on one frequency and the satellite repeats those signals on a down link which is on another frequency.



- ✓ **Advantages of microwave systems**
  - No cables needed
  - Multiple channels available
  - Wide bandwidth

- ✓ **Disadvantages**
  - Line-of-sight will be disrupted if any obstacle, such as new buildings, are in the way
  - Signal absorption by the atmosphere. Microwaves suffer from attenuation due to atmospheric conditions.
  - Towers are expensive to build

2.6. Transmission Checks

Network data transmissions often produce errors, such as toggled, missing or duplicated bits. As a result, the data received might not be identical to the data transmitted, which is obviously a bad thing. Because of these transmission errors, network protocols very often use error-detection codes. An error detection code is a binary code that detects digital errors during transmissions. The detected errors can be corrected, but can prompt the data to be retransmitted. Examples of error-detection codes include parity checking, checksums and cyclic redundancy checks.

2.6.1. Parity Checking

Parity checking refers to the process of using a parity bit to check that data has been transmitted accurately. A parity bit is an extra bit transmitted with a data unit that will be used to check its integrity. There are two types of parity: odd parity and even parity.

- ✓ In odd parity, the parity bit is added such that the total number of bits at 1, in the data unit, is an odd number.
- ✓ In even parity, the parity bit is added so that the total number of 1s is an even number.

Example: What are the parity bits for the following data units in odd parity?

i) 11101100 ii) 00100101 iii) 11010101

Example 2: What are the parity bits for the following data units in even parity?

i) 01101010 ii) 11000101 iii) 10011100

2.6.2. Checksum

A checksum or hash sum is a count of the number of bits in a transmission unit that is included with the unit for the purpose of detecting errors that may have been introduced during transmission. The checksum or hash sum may be computed according to the number of set or unset bits in the message. On reception, the receiver applies the same checksum function/algorithm to the message. If the checksum obtained matches the one sent, the transmission is considered to be successful and error-free.

2.6.3. Cyclic Redundancy Check

A CRC is an error-detection code in which each segment of the original message is combined with additional bits to make a binary number that is divisible by some previously chosen divisor.

- ✓ ***k*** is the length of the message we want to send, i.e., the number of information bits.
- ✓ ***n*** is the total length of the message we will end up sending the information bits followed by the check bits. Peterson and Brown call this a code polynomial.
- ✓ ***n-k***is the number of check bits. It is also the degree of the generating polynomial. The basic (mathematical) idea is that we're going to pick the n-k check digits in such a way that the code polynomial is divisible by the generating polynomial. Then we send the data, and at the other end we look to see whether it's still divisible by the generating polynomial; if it's not then we know we have an error, if it is, we hope there was no error.

2.7. Peripheral Device Control

2.7.1. Buffering

A buffer is an area of memory used to temporarily store data while it is being moved from one place to another. Buffers are used to compensate for differences in rate of flow of data or time of occurrence of events, when transferring data from one device to another. Routers use buffers to route data packets on the Internet. When a packet is sent from one router to another via one or more intermediate routers, the packet is received at each intermediate router in its entirety, stored there until the required output line is free, then the packet is forwarded.

2.7.2. Interrupt

An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor of a high-priority condition requiring the interruption of the current task the processor is executing. Interrupts are used to handle such events as data receipt from a modem or network, or a key press or mouse movement.

2.7.3. Polling

Polling is the process by which the central computer or communications controller in a network, "polls" or asks each device in the network if it has a message to send and then allows each in turn to transmit data. Access and control of star network typically is maintained by a polling system.

2.7.4. Handshaking

Handshaking is the process by which two devices initiate communications. It begins when one device sends a message to another device indicating that it wants to establish a communications channel. The two devices then send several messages back and forth that enable them to agree on a communications protocol.

2.8. Communication Protocols

For proper communication in a network, different entities must speak the same language. There must be mutually acceptable conventions and rules about the content, timing and underlying mechanisms. These conventions and associated rules are referred as protocols.

Definition*: A protocol is a set of rules that govern how devices on a network communicate.*

Protocols allow different computers from different vendors and with different operating characteristics to 'speak the same language'. They set the message formats and procedures that allow machines and application programs to exchange information. The same protocols must be

followed by each machine involved in the communication in order for the receiving host to be able to understand the message. A protocol may be physical or logical.

2.8.1. Physical Protocols

Physical protocols are concerned with how a device connects to a medium. They ensure that a device connected to a medium can transmit through the medium. They make sure that the layout of pins on the connectors is the same and that devices are correctly connected and configured. Few examples of physical protocols are 802.11 for Wi-Fi connections and DSL for broadband.

2.8.2. Logical Protocols

Logical protocols are concerned with data handling. They ensure that data are in the right format for the application, the bit rates match at both ends, and the same error correction is used. Examples of logical protocols are TCP/IP, HTTP, POP3, FTP, SMTP and WAP.

2.8.3. The OSI Reference Model

The Open Systems Interconnection (OSI) reference model or more commonly the OSI model is an ISO standard that defines how network communications take place by providing a framework for standardization. The OSI model divides network communications into seven layers. Each layer is responsible for carrying out specific functions when transmitting data on the network. The table below shows the layered architecture of the OSI reference model.

7	Application Layer
6	Presentation Layer
5	Session Layer
4	Transport Layer
3	Network Layer
2	Data Link Layer
1	Physical Layer

OSI Reference Model

These layers can be recalled using the following mnemonics: All People Seem To Need Data Processing. (Layers 7 - 1)

Layer 7: Application

It provides network services directly to the user's applications such as a web browser or e-mail client. This layer is said to be "closest to the user". Examples of protocols that operate at this layer are: TELNET, HTTP, FTP, SMTP and POP.

Layer 6: Presentation

The Presentation layer represents the data in a particular format to the Application layer. It defines encryption, compression, conversion and other coding functions. Examples of specifications defined at this layer are: GIF, JPEG, MPEG, MIME and ASCII.

Layer 5: Session

It establishes, maintains and terminates end-to-end connections (session) between two applications on two network nodes. It controls the dialogue between the source and destination nodes, which node can send when and for how long. Examples of protocols that operate on this layer are: RPC, NETBIOS and X.225

Layer 4: Transport

It is responsible for end-to-end delivery of entire messages. It allows data to be transferred reliably and uses sequencing to guarantee that it will be delivered in the same order it was sent. It also provides services such as error checking and flow control. Examples of protocols at this layer are: TCP, UDP, NETBEUI and SPX.

Layer 3: Network

It is responsible for path determination, routing, and the delivery of packets across internetworks. It is also responsible for addressing (also known as logical addressing) for example IP addressing. Examples of protocols at this layer are: IP, IPX and ICMP.
Examples of devices that operate at this level are Layer-3 switches and routers. WAPs (wireless access points) with built-in routing capabilities also act at this layer.

Layer 2: Data Link

It is responsible for reassembling bits taken off the wire by the physical layer to frames and makes sure they are in the correct order and requests retransmission of frames in case an error occurs. It provides error checking by adding CRC to the frame. Examples of protocols at this layer are: Ethernet, Token Ring, PPP and ISDN.
Examples of devices that operate at this layer are: switches, bridges, NICs and WAPs (Wireless Access Points).

Layer 1: Physical

This layer communicates directly with the communication medium. It is responsible for activating, maintaining and deactivating the physical link. It defines electrical and optical signaling, voltage levels, data transmission rates, as well as mechanical specifications such as cable lengths, and connectors, the amount of pins and their functions. Examples of devices that operate at this layer are: hubs, repeaters, and NICs.

These layers can be recalled using the following mnemonics: All People Seem To Need Data Processing. (Layers 7 - 1)

3. The Internet

3.1. Brief History

Many years ago, the military of the United States of America desired to interconnect or link their computers in order to better understand and manage information and communication with respect to enemy attacks in times of crisis. In the year 1969 the Department of Defense (DoD) then developed an experimental network called the Advanced Research Project Agency Network (ARPANet)
In the year 1980, the National Science Foundation of the United States of America then developed the technology of ARPANet to produce the National Science Foundation Network (NSFNet) which now enabled universities and other school establishments in the USA to be interconnected. After a great deal of work, a network which enabled the transfer of large amounts of information at very high speed which is today called the Internet was developed.
The Internet can be defined as a worldwide/global system of interconnected computer networks. It is the network of networks in which users can view information on the World Wide Web, exchange electronic mail, participate in electronic discussion forums (newsgroups), send files from any computer to any other and even use each other's computers directly if they have appropriate passwords. Another name for the Internet is information superhighway.

3.2. ISP and Internet Access

An Internet service provider (ISP), also sometimes referred to as an Internet access provider (IAP), is a company that offers Internet access to individuals and organizations. The ISP connects to its customers using a data transmission technology appropriate for delivering Internet Protocol Paradigm, such as dial-up, digital subscriber line (DSL), cable modem, wireless or dedicated high-speed interconnects.

ISPs may provide Internet e-mail accounts to users which allow them to communicate with one another by sending and receiving electronic messages through their ISP's servers. They may also provide services such as remotely storing data files on behalf of their customers, as well as other services unique to each particular ISP.

Different methods exist for connection to the Internet.

3.2.1. Dial-Up Connection

A dial-up connection is a connection that is established by dialing a telephone number through a modem. A dial-up connection uses a dial-up modem to transmit digital information over the Plain Old Telephone System (POTS). POTS refers to the standard telephone network designed for analog transmission of voice over copper wire. This type of connection offers relatively slow transfer rates and is established on demand. This method has long been the most widely used method to connect to the Internet but it has been replaced by high-speed broadband and wireless connections.

3.2.2. Digital Subscriber Line

DSL uses the standard copper telephone wires, often already installed in homes and offices to provide a high-speed Internet connection. xDSL means that there are different types of DSL: asynchronous DSL (ADSL), synchronous DSL (SDSL), High bit-rate DSL (HDSL), Rate Adaptive DSL (RADSL) and ISDN DSL (IDSL).

- ✓ ADSL allows the telephone wires to be used for analog POTS system and digital data transfer simultaneously. The download speed (downstream) for ADSL is faster than the upload speed (upstream).
- ✓ SDSL cannot share the physical medium with standard telephone communications and has a download speed equal to the upload speed.

A DSL connection requires a transceiver (DSL modem) which allows an Ethernet UTP or a USB connection directly to a PC, or to a hub, router, or switch to provide Internet access to an entire network. The transceiver can be integrated into a router or switch.

3.2.3. Broadband Cable

TV channels only take up 6MHz of cable bandwidth each, which usually leaves several hundred MHz available. This additional space on cable is used for high-speed Internet connection. Information from the Internet travels through the cable as a single TV channel. Just as with DSL, cable Internet requires a special transceiver (cable modem) which allows information to be sent and received on frequencies not used by TV channels. The cable modem provides one or more LAN interfaces, usually Ethernet or USB which connect directly to a client or a device such as a hub, switch, or wireless router to allow additional clients or entire networks to use the same connection. The cable modem is also equipped with connections for TV and radio.

3.2.4. Wireless Internet Access

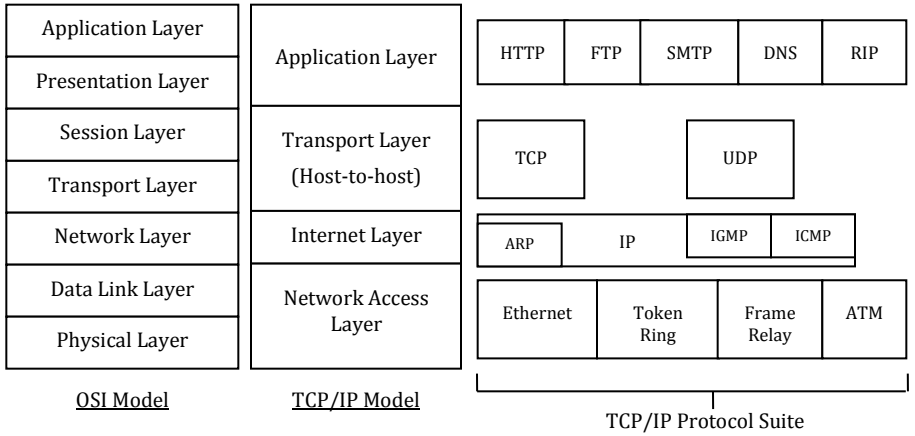
Wireless Internet access or wireless broadband is particularly useful for mobile users. With handheld devices becoming more advanced and increasingly popular, wireless access is becoming one of the major ways of connecting to the Internet. This method provides an "always-on connection" which can be accessed from anywhere as long as you are geographically within network coverage. Wireless Internet access includes deploying Wi-Fi hotspots for accessing the Internet. Technologies such as GPRS and UMTS (Universal Mobile Telecommunication System) allow Smart phones and other handhelds with Internet capabilities to access the Internet using existing cell phone networks.

3.2.5. Internet over Satellite

Internet over satellite (IoS) allows a user to access the Internet via a satellite that orbits the earth. A satellite placed at a static point above the earth's surface, communicates with the ISP's dish giving the user access to the internet.

3.3. The TCP/IP Model

The Internet uses a collection of protocols known as the TCP/IP protocol suite. It is called TCP/IP after two of its most prominent protocols, but there are other protocols as well. The TCP/IP model is based on a four-layer model for networking. The layers from top to bottom are application layer, transport layer, Internet layer and network access layer. Below is a comparison between the TCP/IP model and the OSI model.



During a transmission, a message travels down all network layers at the source machine. Before sending a message to the next layer, each layer places it in an envelope of overhead information related to that layer. This process is called encapsulation. Encapsulation at each layer produces a data unit called protocol data unit (PDU).

At the receiving end, the message travels up through the network layers, each layer removing the envelopes added when the message was sent. So, upon its receipt, the message is in its original state.

3.3.1. Network Access Layer

This layer is responsible for sending and receiving TCP/IP packets on the network medium (Physical/Data Link). The network Access layer PDU called frame, is obtained by adding a header and a trailer to the PDU from the Internet layer.

Applicable LAN technologies: Ethernet, Token Ring, FDDI etc.

Applicable WAN technologies: X.25 (old), Frame Relay, ATM etc.

3.3.2. Internet Layer

This layer is responsible for packaging, addressing and routing of packets. The PDU at this layer is called datagram or packet.

Core Internet layer protocols are: IP, ARP, ICMP and IGMP.

a. Internet Protocol

Internet Protocol (IP) specifies the format of packets and the addressing scheme. All computer devices (desktops, laptops, PDAs, phones, tablets) connected to the Internet, have IP addresses by which they are identified.

***Definition**: An IP address is a unique identifying number given to every single computer on a TCP/IP network.*

Two versions of IP addresses are available: IPv4 that uses 32 bits and IPv6 that uses 128 bits.

- ✓ An IPv4 is made up of four sets of numbers separated by dots such as 123.23.168.22. This notation is known as dotted decimal notation. Each of the four numbers separated by dots can be any number from 0 to 255, making for a total of 4.3 billion potential IPv4 addresses (i.e. 255×255×255×255).

- ✓ An IPv6 has eight sets of numbers separated by colons such as 3ffe:1900:4545:3:200:f8ff:fe21:67cf.

IP addresses are assigned manually by an administrator or automatically by DHCP (Dynamic Host Control Protocol) or APIPA (Automatic Private IP Addressing). An IP address is also known as a logical address.

b. Address Resolution Protocol

ARP resolves IP addresses to MAC addresses. It is used to obtain the MAC address of a node on a network when its IP address is known.

***Assignment:** Differentiate between an IP address and a MAC address.*

c. Internet Control Message Protocol

ICMP is responsible for diagnostics and error reporting. It is used to send control and error messages to computers and servers.

d. Internet Group Management Protocol

IGMP is responsible for management of group multicast.

3.3.3. Transport Layer

The transport layer is responsible for sequencing and transmission of packets, acknowledgment of receipts, recovery of packets and flow control. In essence, it engages in host-to-host transportation of data packets and the delivery of them to the application layer. A transport layer PDU is called segment.
Core protocols at this layer are TCP and UDP.

a. Transmission Control Protocol

TCP is a connection-oriented reliable protocol used in the accurate transmission of large amounts of data. Data packets are verified using checksums and retransmitted if they are missing or corrupted. The application plays no part in validating the transfer.

b. User Datagram Protocol

UDP is a connectionless unreliable protocol used for the transmission of small amounts of data. Data packets are sent without testing to verify whether they actually arrive at the destination, nor whether they were corrupted in transit. It is up to the application to determine these factors and request retransmissions. UDP is faster compared to TCP.

3.3.4. Application Layer

The application layer provides user applications with the ability to access the services of the other layers. Some protocols of this layer are HTTP, FTP, POP, SMTP, Telnet, IMAP and WAP.

a. Hypertext Transfer Protocol

HTTP is a standard method of publishing information as hypertext in HTML format on the Internet. It provides the ability to supply web pages between a browser and the server. HTTPS is a secure version of HTTP used for accessing secure web servers, whereby all data transferred are encrypted.

b. File Transfer Protocol

FTP is a standard for transferring files between a server and a client on a TCP/IP network. It provides the ability to upload and download files between hosts on the network.

c. Simple Mail Transfer Protocol

SMTP is used for sending e-mails between servers on the Internet and other TCP/IP networks. It governs the transmission of mail messages and attachments. SMTP is used in the case of outgoing messages.

d. Post Office Protocol

POP is a standard protocol for delivering e-mails to personal computers. There are different versions of the post office protocol indicated by POP*n* where *n = 1, 2, 3* or*4*.

e. Telnet

Telnet is a protocol that allows a computer on the network to be accessed remotely. It provides the ability to login into a remote host and administer the machine. Using Telnet a computer can be used as a terminal on another.

f. Wireless Application Protocol

WAP is a protocol which runs on mobile phones and provides a universal open standard for bringing Internet content to mobile phones and other wireless devices.

***Assignment**: Give the full meaning of the following protocols and state their functions.*

i) IMAP (ii) RIP (iii) DNS

3.4. TCP/IP Ports

A computer has a single physical connection to the network. All data destined for a particular computer arrives through that connection. However, the data may be intended for different applications running on the computer. To identify the application for which the data is intended, TCP requires port numbers on the host and destination for communication.

***Definition**: A communication port is a number (16-bits) that identifies an application on the Internet or TCP/IP network.*

Popular Internet application protocols are associated with well-known ports assigned by the Internet Assigned Number Authority (IANA). Sample TCP port numbers are:

Port number	Protocol
20	FTP data channel
21	FTP control channel
23	Telnet
25	SMTP
80	HTTP
110	POP

Ports are usually combined with IP addresses to form a socket. For example 127.102.10.0:80.

3.5. Internet Services

3.5.1. The World Wide Web

The World Wide Web (WWW) is a system of interconnected hypertext documents that can be accessed via the Internet. Documents are connected to other documents by hypertext links, enabling the user to search for information by moving from one document to another. It consists of a large number of web servers that host websites. A website consists of a number of web pages connected by hypertext links. A web page is a text file that contains information stored using a structured language called HTML (Hypertext Markup Language).
A website can be accessed by typing its address or URL (Uniform/Universal Resource Locator) into the address bar of a web browser. An example of a URL is http://www.crtv.cmwhere http is the protocol used and www.crtv.cm, the domain name (address) of the site.

Example 1: http://www.bgsmolyko.edu/Ls3,4/ict796/intenet.pdf

- ✓ http is the protocol used (hypertext transfer protocol)
- ✓ www.bgsmolyko.edu is the domain name (the machine at BGS Molyko that hosts the website)
- ✓ Ls3,4/ict796/internet.pdf is the path of the document (resource) on the host computer. Ls3,4 is the folder, ict796 is the subfolder and internet.pdf is the file(resource).

Example 2: www.minsup.gov.cm

- ✓ gov is the top level domain which specifies that the URL is for a government institution.
- ✓ cm specifies the country in which the URL is hosted or the country in which the institution is found.

Assignment: *What is a home page?*

a. Domain Name System

A domain name system (DNS) is a service which performs the function of turning human-understandable domain names into IP addresses.

b. Web Browser

A web browser (or simply browser) is a computer program that enables a user to read hypertext in files or on the World Wide Web. Popular browsers include Mozilla Firefox, Microsoft Internet Explorer, Opera Mini and Netscape.

c. Search Engine

A search engine is a computer program that searches for specific words on the World Wide Web and returns a list of documents in which they were found. Examples of search engines include Google and yahoo.

3.5.2. Electronic Mail

Electronic mail or e-mail (email) is a means of sending messages, text, and computer files between computers via the Internet. To send and receive e-mails, you need an Internet connection and an e-mail account which can be created within a webmail service such as Yahoo, Hotmail or Gmail. When you create an e-mail account, you are given a unique email address that gives you access to your mail box. An email address is made up of two parts separated by the symbol @ pronounced "at". For example bgsmolyko@yahoo.com.
In the above address,

- ✓ bgsmolyko is the user ID, user name or login
- ✓ yahoo.com is the domain name. The domain specifies the mail server (computer) on which the mail box is located.

The part of the domain name after the dot is called top-level domain, and specifies the type of organization or the country the host server is located. Some common top-level domains are:

- o .com - for commercial enterprises
- o .edu - for educational institutions and universities
- o .gov - for United States government agencies
- o .net - for organizations such as Internet Service Providers
- o .org - for non-commercial organizations

3.5.3. Instant Messaging

Instant messaging is a live (or real time) communication which occurs when brief text messages are exchanged instantly over the Internet. Instant Messaging requires that both users be on-line at the same time. Common IM applications are AOL Instant Messenger, Yahoo Messenger and Microsoft MSN messaging.

3.5.4. Internet Telephony

Internet telephony or voice over IP (VoIP) is the transmission of voice telephone conversations through the Internet or IP networks. It allows users to have voice-talk with others through the Internet. The telephone calls are digitized and transmitted through the Internet. Internet telephone services can be mainly categorized into net-to-net and net-to-phone telephony.
In net-to-net telephony, both caller and receiver must be online. When both are online, one dials the other person's phone number. If they accept the call, then voice communication is established.
In net-to-phone, only one person has to be online. This person dials the other person's phone number and the latter receives a ring on their phone. Yahoo messenger and Skype provide services for both types.

3.5.5. Interpersonal Computing

Interpersonal computing refers to person-to-person interactions facilitated by websites that enable collaborative content creation, sharing and manipulation. Interpersonal computing involves: blogs, social networks, wikis and viral video sites.

a. Blogs

A blog (web log) is a chronological, journal-style website which its author (or "blogger") maintains like an online diary, with regular entries of commentary, descriptions of events, or other material such as graphics or video. Many blogs provide commentary or news on a particular subject; others

function as more personal online diaries. They also provide the readers with the ability to leave comments in an interactive format.

b. Social Networking Sites

Social networking sites are websites that allow user to build personalized communities to socialize with. Common features include a customizable profile, the ability to add other users as friends, the ease of sharing pictures, music, text, and links, and built-in chat and mail features. Examples of social networking sites are Facebook, Twitter and Instagram.

c. Wikis

Wikis are websites that allow visitors to easily add, remove and edit content, hence enabling the collaborative authorship of comprehensive documents. The best example of a wiki is the multi-lingual, web-based encyclopedia Wikipedia, and which currently includes over two million articles.

d. Viral Video Sites

A viral video is a video that is distributed by sharing. Viral video sites are websites that allow anybody to post videos online. Whilst it is now not difficult to put a video on any website, the significance of viral video sites is that they provide somewhere to put videos where it is likely that at least some other people will actually find them. Examples are YouTube and Kaltura.

3.5.6. Electronic Commerce

E-commerce refers to the buying and selling on the Internet. Different models of e-commerce exists: business-to-business, business-to-consumer, business-to-government and m-commerce

a. Business-to-Consumer

B2C model sells goods or services to the consumer, generally using online catalog and shopping cart transaction systems. For example, an online pharmacy giving free medical consultation and selling medicines to patients is following a B2C model. Amazon is an example of one of the first and still one of the most successful B2C e-commerce companies.

b. Business-to-Business

B2B describes commerce transactions between businesses, such as between a manufacturer and a wholesaler, or between a wholesaler and a retailer. In this form, the buyers and sellers are both business entities and do not involve an individual consumer.

c. Business-to-Government

B2G is a derivative of B2B marketing. B2G sites provide a platform for businesses to bid on government opportunities which are presented as solicitations requests for proposal (RFPs) to tender.

d. M-Commerce

M-commerce refers to the use of mobile devices for conducting transactions. The mobile device holders can contact each other and can conduct the business. Even the web design and development companies optimize the websites to be viewed correctly on mobile devices.

Some e-commerce websites are: www.bruneiair.com for airline ticket bookings, www.amazon.com for sales of books and magazines, www.brumedia.com/shop for sales of computers, shirts, and cameras

- ✓ Some advantages of setting up an e-commerce website are:
  - Products can be sold to local customers and those from abroad.
  - It is accessible 24 hours each day.
  - It needs a small number of staff to run.
  - It does not need huge office space.
  - Products can be sold at cheap prices

- ✓ Some disadvantages of e-commerce are:
  - Credit card fraud - hackers are able to steal credit card numbers on computers.
  - Certain websites spy or track the buying habits of their customers.
  - Some goods do not arrive after they are paid for.
  - It lacks human interaction as one only sees pictures and some text descriptions.

3.5.7. Online Banking

Online banking (Internet banking) is simply the use of the Internet to perform banking operations like opening an account, accessing account information, transferring funds, getting a bank statement etc. In an Internet banking system, the bank has a centralized database that is web-enabled. All the services that the bank has permitted on the Internet are displayed in a menu. Any service can be selected and further interaction is dictated by the nature of service.

3.6. Intranet and Extranet

An intranet is a private network that is set up using the same technology and protocols as the Internet but is restricted to users inside an organization. It provides similar services within an organization to those provided by the Internet without necessarily being connected to the Internet. An intranet can be seen as a private version of the Internet. To access an intranet, some form of user authentication is usually required. External access to an intranet is not always provided.

An extranet is an interconnection of two or more intranets. It allows an organization to share information with other organizations using Internet standards but with security features preventing access to others.

CHAPTER 4: INFORMATION SYSTEMS

Introduction

A system is an integrated set of regularly interacting or interdependent components created to accomplish a defined objective, with defined and maintained relationships among its components. Basically, there are three major units in every system namely input, processing and output. The objective of a system demands that some output be produced as a result of processing the suitable inputs.

An information system can therefore be seen as a set of interrelated components that collect data, process the data to produce information.

1. Data and Information

Data are raw facts and figures that have no context or purposeful meaning. In computing, data is simply any number, letter or symbol that can be entered into a computer system. When data has been processed it gives information.

Information is data that has been processed. The processing gives it meaning and a context. Information can also be defined as the useful knowledge derived from facts placed in the right context with the purpose of reducing uncertainty.

For example, the number 12.5 is data because we do not know why or in what context it is being used. However, if the number appears on a student's report card to show that they have an average of 12.5, then this data has changed into information, because it has acquired a context (it's an average) and meaning.

Also, the binary patterns that describe an icon on your desktop are data. They become information after the operating system software has processed them, because then they become meaningful to you as the icons representative of your hard disk or Internet explorer.

In summary,

$$Information = Data + Context$$

1.1. Sources of Data

a. *Questionnaire*

A questionnaire is a set of questions used for collecting data from people. A questionnaire may be in paper format or online.

b. *Interview*

An interview is a meeting during which somebody is asked questions. Interviews allow you to collect a greater depth of data and understanding from people than is possible by just using a questionnaire.

c. *Observation*

In observation, the data gatherer observes what is happening during a process or event and produces some kind of data file as a result

d. *Data logging*

Data logging is an automated method of gathering data by using sensors.

e. *Document review*

Document review is getting relevant data from a document, an article or a book.

f. *Data mining*

Data mining is the exploration of databases to collect data.

1.2. Characteristics of Information

Good information is that which is used and which creates value. Experience and research show that good information has numerous qualities.

a. *Timeliness*

Delay destroys the value of information. For effective decision making, information must reach the decision-maker at the right time. Timeliness means that information must reach its recipients within the prescribed timeframes.

b. *Accuracy*

Wrong information given to decision-makers would result in wrong decisions. Accuracy means that information should be free from mistakes and errors.

c. *Current*

For the characteristic of timeliness to be effective, information should be current or up-to-date. Information must be current as a fact of yesterday may not be a fact of today.

d. *Completeness*

Information should have every necessary part or everything that is wanted. If information is not complete, it may lead to wrong decisions being made as only half of an entirety of the information is known.

e. *Explicitness*

Good information should not require further analysis for decision making. It should be clear and obvious, leaving no doubts as to its intended meaning.

1.3. Data Collection

All computer systems need to have data input into them otherwise they have nothing to process. Getting the data for the computer to process is known as data collection. Data collection can be manual or automatic.

1.3.1. Manual Data Collection

Manual data collection uses forms and questionnaires. Data collected through this method has to be entered into the computer by typing and clicking.

Many different errors can occur when entering data into a system. To try and reduce the amount of input errors, a system designer can build in validation and verification checks into the software that the data is entered into.

a. Data Verification

Verification means checking the input data to make sure it has been entered correctly. Verification tries to ensure there have been no transcription errors. It is a check on accuracy. Two methods of data verification are double entry and proofreading.

- ✓ **Double Entry**

Double entry consists of entering the data twice. The two entries are then compared against each other and a warning given if they do not match. For example, a new password is always entered twice.

- ✓ **Proof Reading**

Proofreading consists of reading the data entered either on screen or printout, to be sure that it matches the data source. It is also known as visual check.

b. Data Validation

Validation is a check on input data to ensure that the data is sensible or reasonable. It compares the input with a set of rules that the computer has been told the data must follow. If the data does not match up with the rules then there must be an error. Validation only checks that the data is valid. The data may be valid but not correct. Five types of validation checks are:

✓ **Type/Format check**

A type check is used to ensure that data entered in a field fits the required data type. For example, a person's name will consist of letters of the alphabet and sometimes hyphens and apostrophe. Any name that contains numbers will be rejected as invalid.

✓ **Length check**

A length check ensures that an entered value is no longer than a certain number of characters. For example, a phone number has 8 digits. Entering fewer or more digits makes a number invalid.

✓ **Range check**

A range check is used to ensure that the data entered falls between a specified minimum and maximum values. For example, a mark in an exam is between 0 and 20. Any mark below 0 or above 20 is rejected as invalid.

✓ **Presence check**

A presence check ensures that an entry has been made in a particular field. If it has not, the system will not allow the record to be saved or any entries to be made in later fields. Such fields called mandatory fields are indicated on some systems by the used of an asterisk.

✓ **Check digits**

A check digit is a digit attached to the end of a string of digits that can be used to check that the string is correct. It is calculated from the other digits in the string. One example where a check digit is used is in the 10 digit ISBN number which uniquely identifies books. The last number of the ISBN is actually the check digit for the other numbers. For example, in the ISBN 185813415**3**, the 3 at the end of the number is the check digit**.**

The check digit for ISBNs is obtained using a calculation method known as the Modulus-11 weighted check digit calculation.

- Start with original number i.e. 185813415
- Weight each digit by its position in the string and add up the results.

Position	10	9	8	7	6	5	4	3	2	1
Digit	1	8	5	8	1	3	4	1	5	
Weightings	10	72	40	56	6	15	16	3	10	

Total = 10+72+40+56+6+15+16+3+10= 228

- Divide the total by 11 and then subtract the remainder from 11. The check digit is the result of this operation.
  228 / 11 = 20 remainder 8 => Check digit is 11-8 = 3.
- Add the check digit to the end of the original number to get the complete product number. i.e. 185813415**3.**

To check whether the ISBN is correct,

- Input the number including the check digit.
- Weight each digit by its position in the string and add up the results.

Position	10	9	8	7	6	5	4	3	2	1
Digit	1	8	5	8	1	3	4	1	5	3
Weightings	10	72	40	56	6	15	16	3	10	3

Total = 10+72+40+56+6+15+16+3+10+3 = 231

- Divide the total by 11. If the remainder is 0, then the number has passed the validation check and so it is likely that it has been inputted correctly.
  231 / 11 = 21 remainder 0

Exercise:

i) Dates are read into a computer in the following format: DDMMMYY e.g. 15DEC92. The following dates were rejected by a validation program: 3JAN71, AUG2166, 31SEP72. State the validation check used to discover each error.

*ii) Calculate the check digits to complete the ISBNs: 019276150-**1** and 995640216-**8**.*

iii) Set up a spreadsheet which will calculate and check the validity of Modulo-11 weighted check digits for any given ISBN.

1.3.2. Automatic Data Collection

Automatic data collection is a form of data input in which there is no data entry. It uses sensors and specialized input devices to collect data that is directly entered into the computer without any human involvement. It is also called data capture. Different automatic data collection methods are:

a. Optical Mark Recognition (OMR)

OMR uses a device called an optical mark reader to read marks made with prescribed pens, pencils or special writing material on OMR forms, and convert them into information in the computer. This system is good for multiple choice examination questions.

b. Optical Character Recognition (OCR)

This method uses a device called an optical character reader to read characters from printed or handwritten text and transmit them to the computer as if they were typed from the keyboard. This method is suitable for capturing data from airline tickets; reading postal codes; capturing data from telephone and electric bills.

c. Magnetic Ink Character Recognition (MICR)

The device used is a magnetic ink character reader that reads characters written in magnetic ink, using magnetic stripe readers or card swipe machines that capture the information on the magnetic card. These are seen on the back of credit cards and bank cards.

d. Barcode Reading

An optical device called barcode reader is used to read the barcode on products and convert them into a form that can be processed by the computer. A bar code is a sequence of vertical lines and numbers that identify a product. They are used in libraries, supermarkets and retail shops.

e. Voice Recognition

This method converts speech into text or a sequence of computer commands. It is most common for data entry and word processing environments.

2. Information System Components

There are five basic components in an information system: hardware, software, procedures, data and people.

✓ **Hardware**

Hardware refers to the physical devices that make up the system. They are the whole set of equipment used for input, processing, storage and communication of data.

✓ **Software**

Software is the collection of computer programs used in the system. They provide the instructions that tell the computer what to do.

✓ **Data**

Data are raw, unorganized, potentially useful facts and figures that are processed to produce information.

✓ **People**

People are the main actors of the system. They are the users of the information system. They input data into the computer, give some direction to the computer to perform tasks and review information on the computer for output.

✓ **Procedures**

Procedures are the series of documented actions taken to achieve a particular goal. A procedure is more than a single simple task. It can be complex and involved, such as reinstalling software, performing a backup etc.

3. Organizational Information Systems

There are three levels at which information can be used in an organisation: *strategic, tactical* and *operational* levels. This can be represented using the pyramid below.

✓ At the strategic level, information is needed by senior managers (executives) to help them with their business plans. Information at this level is used for making long term decisions.

✓ At the tactical level, information is needed by middle managers to help them monitor and control business activities. Tactical planning and decision-making takes place within the guidelines set by the strategic plan.

✓ At the operation level, employees with operational roles need information to help them carry out their duties. Results of operational work are passed upwards to let the tactical planners evaluate their plans.

In order to meet with the information needs of the organization, different types of information systems exist which can be grouped into two: operations support systems and management support systems.

3.1. Operations Support Systems

Operation support systems process data generated by business operations. They act at the operational level of the organization. Major categories of OSS are transaction processing systems, office automation systems and process control systems.

3.1.1. Transaction Processing Systems

A transaction is any event of interest to an organization. It may be a business activity such as a payment, a deposit, a customer's order, a reservation or a student's registration. Transaction processing systems capture and process data generated during an organization's day-to-day transactions and maintain records about the transactions. They are vital for any organization or business as they gather all the input necessary for other types of systems. TPS are also called Data Processing Systems.

There are two types of TPS: batch processing and online processing systems.

a. Batch Processing

With batch processing, transaction data is collected over a period of time and all processing is done as a group. Batch processing is ideal in situations where large amounts of data requiring similar processing are to be processed. Examples are:

- ✓ Payroll systems for calculating employee salaries
- ✓ Billing systems for calculating consumer bills.

b. Online Transaction Processing

With online transaction processing (OLTP) the computer processes transactions as they are entered. Such systems are ideal for situations where the master file needs to be updated each time a transaction is made. Examples are:

- ✓ Stock control systems which reduce automatically the number of items in stock once an item has been bought
- ✓ Reservation systems which reduce automatically the number of seats available on a flight or bus once a seat has been booked.

3.1.2. Office Automation Systems

Office automation systems automate office procedures and enhance office communication and productivity. They support a wide range of office activities such as creating and distributing documents, sending messages and scheduling. The software an OAS uses to support these activities include word processing, spreadsheets, databases, presentation, graphics, e-mail, Web browsers, personal information management, and groupware. They use communication technologies such as voice mail, facsimile (fax), videoconferencing, and electronic data interchange (EDI) for the electronic exchange of text, graphics, audio, and video. OAS are also called Office Information Systems (OIS).

3.2. Management Support Systems

Management support systems provide information and support needed for effective decision making by managers. They act at the tactical and strategic levels of the organization. Major categories of MSS are management information systems, decision support systems and executive information systems.

3.2.1. Management Information systems

Management information systems generate accurate, timely and organized information needed by middle managers to take decisions, solve problems, supervise activities, and track progress. They provide routine information for routine tasks. The source of data for an MIS usually comes from numerous databases. These databases are usually the data storage for Transaction Processing Systems. MIS take information from TPS and summarize them into a series of management reports. As such, MIS are sometimes called Management Reporting Systems (MRS).

MIS generate three basic types of information or reports: detailed, summary and exception.

- ✓ Detailed reports confirm transaction processing activities. A detailed order report is an example of a detail report.
- ✓ Summary reports consolidate data into a format that an individual can review quickly and easily. To help synopsize information, a summary report typically contains totals, tables, or graphs. An inventory summary report is an example of a summary report.
- ✓ Exception reports report information that is outside of a normal condition. These conditions called the exception criteria, define the range of what is considered normal activity or status. An example of an exception report is an inventory exception report that notifies the purchasing department of items it needs to reorder. Exception reports help managers save time because they do not have to search through a detailed report for exceptions. Instead, an exception report brings exceptions to the manager's attention in an easily identifiable form. Exception reports thus help them focus on situations that require immediate decisions or actions.

Examples of MIS are:

- ✓ Sales management systems
- ✓ Inventory control systems
- ✓ Budgeting systems
- ✓ Management reporting systems

3.2.2. Decision Support Systems

Decision support systems are designed to help tactical and strategic decision-making in situations where there is uncertainty about the possible outcomes of those decisions. They provide interactive support for non-routine decisions or problems.

TPS and MIS provide information on a regular basis. However, managers need information not provided in their reports to help them make decisions. Decision support systems therefore use data from internal (TPS and MIS) and external sources.

- o Internal sources of data might include sales, manufacturing, inventory, or financial data from an organization's database.
- o Data from external sources could include interest rates, population trends, and costs of new housing construction or raw material pricing.

Examples of DSS are:

- ✓ Logistics Systems
- ✓ Financial Planning Systems
- ✓ Spreadsheet Models

3.2.3. Executive Information Systems

Executive information systems (EIS) are designed to support the information needs of executive management. Their purpose is to analyse, compare and identify trends to help the strategic direction of the organisation. Information in an EIS is presented in charts and tables that show trends, ratios, and other managerial statistics. Because executives usually focus on strategic issues, EISs rely on external data sources that can provide current information on interest rates, commodity prices, and other leading economic indicators.

To store all the necessary decision-making data, DSSs or EISs often use extremely large databases, called data warehouses.

4. Other Information Systems

4.1. Expert Systems

An expert system is a computer program that tries to emulate the decision making of a human expert. It does this by combining the knowledge of human experts and then, following a set of rules, it draws inferences. An expert system is made up of three parts: a knowledge base, an inference engine and a user interface.

- ✓ The knowledge base stores all of the facts, rules and information needed to represent the knowledge of the expert.
- ✓ The inference engine is the part of the system that interprets the rules and facts using backward and forward chaining to find solutions to user queries.
- ✓ The user interface allows the user to enter new knowledge and query the system.

Expert systems are one part of an exciting branch of computer science called artificial intelligence (AI). AI is the science and engineering of making intelligent machines which are able to simulate human behavior. AI technology can sense your actions and, based on logical assumptions and prior experience, will take the appropriate action to complete the task. AI has a variety of capabilities, including speech recognition, logical reasoning, and creative responses.

Example 1:

A medical diagnosis expert system could be used in a doctor's waiting room. Patients would use a touch screen to answer questions on symptoms etc. created by the system. Based on the patient responses, the system could use its database of diseases and symptoms, along with its programmed rules, to prepare a list of possible diagnosis for the doctor to investigate further.

Advantages

- o The doctor saves time because they do not have to ask the patient to describe their symptoms in person.
- o The doctor is given a suggested list of possible diagnosis to investigate further.
- o The computer can store far more information than the doctor and can search it far faster and more efficiently.
- o The database can easily be updated or extended.

Disadvantages

- o It can be difficult to describe symptoms to a computer system.
- o It relies on a basic level of skills from the user.
- o It lacks the 'human touch' of a doctor actually talking to a patient.

Example 2:

A expert system used by a car mechanic could help to diagnose faults in a car by asking the mechanic to carry out tests or answer questions. This could also be automated as the computer could have

inputs from sensors or have a direct interface with a computer system built into the car. The system would then give the mechanic a list of probable faults with the car even make automatic adjustments using the computer system built into the car. It would also be helpful to a newly qualified mechanic as they would have access to a wider range of knowledge, could save time compared to having to contact an expert and the system could be used as a training aid.

Advantages

- The user saves time because the mechanic is taken through a logical series of things to try out that are likely to solve common problems.
- The system can directly access computer systems built into many modern cars.
- The system can store details of a huge range of common faults with different makes of cars.
- The database can easily be updated or extended as new problems are identified.

Disadvantages

- It can be difficult to answer questions on something the user may know nothing about.
- It relies on a basic level of skills from the user.

4.2. Geographic Information System

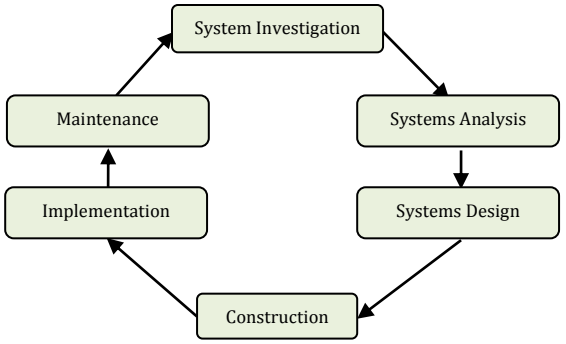
A geographic information system (GIS) is a computer system for capturing, storing, checking, and displaying data related to positions on the earth's surface. GIS stores information about the world as a collection of layers that can be linked together by a common locational component such as latitude and longitude, a postal zip code, census tract name, or road name.

Data in many different forms can be entered into GIS. Data that are already in map form can be included in GIS. This includes such information as the location of rivers and roads, hills and valleys. Digital or computerized data can also be entered into GIS. An example of this kind of information is data collected by satellites that show land use - the location of farms, towns, or forests. GIS can also include data in table form, such as population information. GIS technology allows all these different types of information, no matter their source or original format, to be overlaid on top of one another on a single map.

5. Design of Information Systems

Most computer-based information systems are conceived, designed, and implemented using some form of systematic development process called software development process. In this process, end users and system analysts design systems based on an analysis of the information requirements of the information system to be built. The software development process is also called system development life cycle (SDLC).

SDLC is a structured step-by-step approach for creating and maintaining information systems. It consists of a number of stages that describe the activities involved in an information system development process. SDLC involves the following stages: system study, system analysis, system design, development and testing, implementation, and maintenance.



System Development Life Cycle

5.1. System Investigation

System investigation is a brief study of the system under consideration that gives a clear picture of what actually it is. During this phase, the system is evaluated and deficiencies are identified. This can be done by interviewing users of the system and consulting with support personnel. Main activities at this stage are:

- ✓ Determining whether a business problem or opportunity exists. i.e. identifying problems and opportunities.
  - A problem is a basic condition that is causing undesirable results
  - An opportunity is a basic condition that presents the potential for desirable results.
- ✓ Conducting a preliminary feasibility study to determine whether a new or improved information system is a feasible solution.
- ✓ Developing a project management plan and obtaining management approval.
- ✓ Building the project team

5.2. System Analysis

Systems analysis is an in-depth study of end user information needs which produces functional requirements that are used as the basis for the design of a new information system. Here, the systems analysts analyses end-user requirements and refines projects goals into defined functions and operations of the intended system. System analysis describes what a system should do to meet the information needs of users. It involves:

- ✓ ***Analysis of the present (old) system***

Analysis of present system involves:

- Collecting factual data about the present system (questionnaires, interviews, observations, etc.)
- Identifying how input, processing, storage and output are being accomplished.
- Analyzing how the present system uses resources (hardware, software and people) to convert input data into useful information
- Understanding information flow within the system
- Identifying problems with the system

- ✓ ***Gathering Business Requirement***

Business requirements are the detailed set of knowledge users request the system must meet to be successful. They explain what has to be done by identifying the necessary tasks, actions or activities that must be accomplished.

It involves:

- Determining specific information needs
- Determining the information processing capabilities required for each system activity (input, processing, output, storage, and control) to meet the needs. The goal here is to identify "what" should be done not "how" to do it.
- Determining functional requirements (information requirements that are not tied to the hardware, software, and people resources that end users presently use or might use in the new system).

✓ ***Feasibility Analysis***

Feasibility analysis is a study which investigates the information needs of prospective users and determines the resource requirements, cost, benefits, and workability of a proposed project. Its goal is to evaluate alternative systems and propose the most feasible and desirable system for development.
Feasibility of a system can be evaluated in terms of four major categories:

- Organizational feasibility focuses on how well a proposed information system supports the objectives of the organization and its strategic plan for information systems.
- Technical feasibilityfocuses on the reliability/capabilities of the hardware and software to meet the needs of the proposed system, and whether they can be acquired or developed in the required time.
- Economic feasibilityfocuses on whether the tangible costs and benefits of the proposed system will exceed the costs of developing and operating it.
- Operational feasibility focuses on the ability of the end users to operate, use, and support the proposed system.

The outcome of a feasibility analysis is a feasibility report which is presented to the user management for approval. It may be accepted or accepted with modifications or rejected.

✓ **Documenting System Analysis**

The outcome of systems analysis is a system proposal or requirements specification document which describes what the new system should do without specifying how to do it. At the end of systems analysis phase, the system analyst produces a system proposal that will be used as basis for the design phase.

5.3. System Design

Systems design consists of design activities, which produce systems specifications satisfying the functional requirements developed in the systems analysis stage. While system analysis specifies what is to be done by the new system, system design describes how the system will accomplish what is to be done.
System design focuses on three main activities: user interface design, data design and process design.

✓ ***User Interface Design***

A user interface is a means of interaction between the user and the computer-based application. This activity focuses on designing how data will be introduced into the system and how the information generated will be retrieved. It produces detailed specifications for information products such as:

- Display screens
- Interactive user/computer dialogues
- Forms (on-screen forms for data input and output)
- Reports (on-screen and printed)

✓ ***Data Design***

Data design focuses on the design of the structure of data and files to be used by the proposed (new) system. It provides detailed descriptions of:

- Attributes (characteristics) of the entities about which the proposed system needs to maintain information.
- Relationships between these entities (E-R diagrams, data flow diagrams)
- Specific data elements (databases, files, records, etc.) that need to be maintained for each entity.
- Data dictionary
- Integrity rules (data validation and verification) that govern how each data element is specified and used in the system.

✓ ***Process Design***

Process design focuses on the design of software resources, that is, computer programs and of procedures needed by the proposed system. It concentrates on developing detailed specifications for the program modules that will have to be purchased as software packages or developed by custom programming. Process design produces:

- Detailed specification of algorithms (pseudo-codes, flow charts, etc.)
- Detailed specifications of the procedures needed to meet the user interface and data design specifications.
- Detailed specification of the database schema (E-R diagram, object diagrams)

The design stage is very important because it is the place where quality is fostered in software engineering. Design provides us with representations of software that can be assessed for quality. Design is the only way that we can accurately translate a customer's requirements into a finished software product or system.

5.4. Construction

Once the design of the system is complete, it has to be converted into a computer understandable form. Development is the stage where the design is converted into a computer program.

✓ ***Coding (programming)***

Coding is an important activity by which a programmer converts the systems specifications from the design stage into computer instructions referred to as programs. It is generally felt that the programs must be modular in nature. This helps in fast development, maintenance and future change if required.

✓ ***Prototyping***

Prototyping is the rapid development and testing of a working model of a product in an interactive and iterative process involving both systems analysts and end users. This working model or prototype, is a partially developed product that enables customers and developers to examine some aspects of the proposed product and decide if it is suitable for a finished product.
Various types of prototyping exist.

- ***Throw-away prototyping***

In throw-away prototyping, the prototype is discarded once the actual requirements have been understood and the final system is developed with a much clear understanding of user requirements.

- ***Evolutionary Prototyping***

In evolutionary prototyping, a functional prototype with minimal functionality is built in the beginning and is refined over time, as requirements are better understood.

- ***Incremental Prototyping***

In incremental prototyping, functional prototypes of the various subsystems are built and then integrated to form a complete system. In other words, the product is built as separate prototypes which are later merged into a final product.

✓ ***Testing***

Testing is the process of executing a program with the intent of finding an error. During testing, trial runs are done to check for errors and whether or not the new system meets the users' needs. Once source code has been generated, the software must be tested to uncover and correct as many errors as possible before delivery.

There are three sets of data that can be used to test the system: normal data and abnormal data.

- Normal data is data which the system will accept.
- Abnormal (erroneous) data is invalid data which the system will reject.
- Extreme data are data values that are chosen at the absolute limits of the normal range. This is to ensure that all normal values will be accepted and processed correctly.

Using these test data, the following test runs can be carried out:

- ***Unit testing*** tests the individual units or modules separately with prepared test data so that any errors can be corrected.

- ***Integration testing*** tests the complete system after the individual units have been tested and put together. This tests that separately developed modules/units work together as planned without error.

- ***System testing*** tests the integrated system to evaluate the system's compliance with its specified requirements.

Two testing techniques that can be used are black-box testing and white-box testing.

- ***Black-box Testing***

Black-box testing is a test that relies on the input/output behavior of the system, without any assumptions to what is happening within the system. It examines some fundamental aspects of a system with little regard for the internal logical structure of the system. Black-box tests are used to demonstrate that system functions are operational, that input is properly accepted and output is correctly produced, while at the same time searching for errors in each function.

- ***White-box Testing***

White-box testing, also called glass-box testing, is a test that relies on information about how the system has been designed and constructed. It requires knowledge of the internal structure or implementation of the system. White-box tests are conducted to ensure that internal operations are performed according to specifications and all internal components have been adequately exercised.

✓ ***Documentation***

The job of the programmer does not end with the code or software instructions. The organization or users need to know how to get the best out of the system. This is done through documentation. System documentation ensures continuity of the system.

There are two types of documentation; user documentation and technical documentation.

- ***User Documentation***

User documentation is a complete description of the system from the user's point of view detailing how to use or operate the system. It could be a paper-based user manual or help incorporated into the software that can be accessed when the software is installed. User documentation always covers the following:

- A guide that describes what the system is supposed to do in non-technical terms
- Instructions for installing and running the program
- Definition for hardware and Operating System requirements
- The format of the output data
- Explanation of common error messages and how to recover from them
- Description of how to make backups against accidental data loss

- ***Technical Documentation***

Technical documentation is a description from the designer's point of view. Technical documentation often contains:

- Detail functioning of the software showing algorithms, formulae, source codes etc.
- Description of data structures
- Test plans and testing procedures
- User interface and reports
- Location and version of the software

5.5. Implementation/Conversion

Implementation is the conversion from the use of the present (old) system to the operation of the new system. It involves:

✓ Installation of new system
✓ Loading of data into new system
✓ Education and training of users of the system

There are different types of conversions:

- ***Direct Conversion/Cutover:*** in which the old system is completely replaced by the new one. Its disadvantage is that, if the new system fails, there is no back-up system, so data can be lost.

- ***Pilot Conversion:*** in which the new system is installed in one part of the business or organization. This allows the new system to be fully developed and tested. Once the pilot system is running successfully, the new system is introduced to all of the business/organization.
  - Its advantages are that, if something goes wrong with the new system, only a small part of the organization is affected, and the staff that were part of the pilot scheme can help train other staff.
  - As a disadvantage, there is no back-up system for the office/department doing the pilot, if things go wrong.

- ***Parallel Conversion***: in which old system and new system operate alongside each other (in parallel) until new system is proven capable.
  - It is advantageous in that, if the new system fails, the old system will act as a back-up. Also, the outputs from the old and new systems can be compared to check that the new system is running correctly.
  - Its disadvantage is that, entering data into two systems, and running two systems together, takes a lot of extra time and effort.

- ***Phased Conversion:*** in which the new system is installed in phases (stages or steps) gradually replacing parts of the old system until eventually, the new system takes over.
  - Its advantages are that, it allows users to gradually get used to the new system, and training of staff can be done in stages.
  - Its disadvantage is that, if a part of the new system fails, there is no back-up system, so data can be lost

5.6. Maintenance

Maintenance is the general process of changing a system after delivery to correct faults, improve performance or adapt the system to a changing environment or business requirements. Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environment.
Maintenance can be adaptive, preventive, corrective or perfective.

- ✓ ***Adaptive Maintenance***

Adaptive maintenance focuses on adjusting a software product to properly interface with a changing environment. Changes are made to increase system functionality to meet new business requirements.

- ✓ ***Preventive Maintenance***

Preventive maintenance aims in retaining the system's capabilities before the occurrence of any problem (e.g. system failure). It locates weaknesses in the system and provides repairs in order to avoid any eventual breakdown of the system. *Making changes to prevent future system failures.*

- ✓ ***Corrective Maintenance***

Corrective maintenance aims in restoring a defective system to a required state. This implies that repairs are made after a breakdown of the system. (*Making changes to repair system defects*)

- ✓ ***Perfective Maintenance***

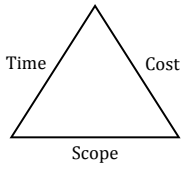
Perfective maintenance refers to enhancements to the product in order to either add new capabilities or modify existing functions. *Making changes to enhance the system and improve such things as processing performance and usability*.

6. Project Management

A project is an endeavor to accomplish a specific objective through a unique set of interrelated tasks and the effective utilization of resources. A project has a definite start and finish time, well-defined outcomes or performance goals, and consumes scarce resources such as money, personnel, material, and equipment.
A project is considered constrained by three functions:

- ✓ Scope: what it is intended to accomplish. In other words, the customer's requirements for the project.
- ✓ Time allocation: the time schedule for the project.
- ✓ Cost: the money, budget, and resources for a project.

These three functions are called the Triple Constraint. The relationship between them is represented using the Project Management Triangle.



The Project Management Triangle visualizes the fact that time, cost and scope of a project are interdependent; changing one of them causes changes in the other two. For example, if you want to shorten a schedule, you can hire more resources which would increase cost, or reduce customer requirements which would affect quality. This simply means "you can have any two of quick, good or cheap, but not all three."

Definition*: Project management is defined as the application of knowledge, skills, tools and techniques to activities of a project for the achievement of the project's objectives/requirements.*

Project management ensures that an acceptable system is developed within time and budget.

6.1. Project Life Cycle

The activities related to a project can be structured and grouped into stages according to the aim of the activities. A typical project goes through the following stages called project life cycle: initiation, planning, execution, monitoring and control, and closing.

a. Initiation

Project initiation determines the main objective of the project and forms a clear understanding about the necessity and suitability of the project. This stage answers the questions "*what?*" and "w*hy?*" Common activities at this stage are:

- ✓ Identification and initial analysis of the business needs.
- ✓ Determination of the main objective(s).
- ✓ Resource analysis (people, equipment, financial; needs and availability).
- ✓ Composition of the project charter.
  (*Project charter - document issued by the project initiator or sponsor that formally authorizes the existence of a project, and provides the project manager with the authority to apply organizational resources to project activities*.)

b. Planning

Project planning involves the project plan development and approval. It determines an optimal scheme/algorithm for project execution. This stage answers the question "*how?*" The main activities at this stage are:

- ✓ Needs analysis
- ✓ Description of the project (including determination of activities and necessary resources)
- ✓ Composition of project plan
- ✓ Planning and performing necessary PR-activities.
  (*PR is the practice of managing the spread of information between an individual or an organization and the public. Public relations activities include: launchings, media conferences, sales promotions, open day, product testing, websites, press release, newsletters*.)

c. Execution

Project execution is the phase within which the deliverables are physically constructed and presented to the customer for acceptance. It integrates people and other resources to carry out the project management plan for the project. The activities undertaken to construct each deliverable will vary depending on the type of project being undertaken. Main activities are:

- ✓ Starting up the execution.
- ✓ Building the deliverables
- ✓ Day-to-day management and reporting

d. Monitoring and Control

Project control measures and monitors progress to identify variances from the project management plan so that corrective action can be taken when necessary to meet project objectives. Control occurs throughout the duration of the project and has a range relatively similar to that of execution. While the

project is being executed, a series of management processes are undertaken to monitor and control the deliverables being output by the project. This includes:

- ✓ Requesting, evaluating and approving changes to the project scope, deliverables, timescales or resources (change management)
- ✓ Controlling the amount of time spent undertaking each activity within the project (time management)
- ✓ Identifying, approving and paying cost/expenses incurred on project (cost management)
- ✓ Reviewing deliverable quality (quality management)
- ✓ Identifying, quantifying and managing risks to the project (risk management)
- ✓ Identifying and handling issues currently affecting the ability of the project to produce the required deliverables (issue management)
- ✓ Measuring each deliverable produced against acceptance criteria (acceptance management)
- ✓ Handling sourcing of products from an external supplier (procurement management)
- ✓ Identifying, creating, and reviewing communication messages within the project (communication management)
- ✓ Performing a phase review at the end of execution to ensure the project has achieved its objectives as planned.

e. Closing

Once all the deliverables have been produced and the customer has accepted the final solution, the project is ready for closure.

Project closure involves:

- ✓ releasing the final deliverables to the customer
- ✓ handing over project documentation to the business
- ✓ terminating supplier contracts,
- ✓ releasing project resources
- ✓ communicating the closure of the project to all stakeholders

6.2. Basic Project management Terminology

- ***Task/Activity***: anything that needs to be done that requires time and consumes resources.
- ***Dependent task***: a task that can only begin after a previous one is finished. For example, roofing a house depends on the construction of the walls.
- ***Slack time or float time***: the amount of delay that can be tolerated between the starting time and completion time of a task without causing a delay in the completion date of the entire project. If we have tasks A and B that start at the same time and task C that is dependent on both tasks A and B. If task A takes 3 days and task B takes 5 days, then task A has 2 days slack time. That is, it can run for 2 days before it affects the planned starting time for task C.
- ***Lag time***: the delay or amount of time that passes between the end of one activity and the beginning of another if the two are dependent. For example, if task A is laying of cement blocks and dependent task B is building the walls of the house, there would be some lag time between the end of task A and the start of task B to let the blocks get dry.
- ***Lead time***: occurs when a task should theoretically wait for its predecessor to finish, but can actually start a little early. The time that the tasks overlap is lead time.
- ***Milestone***: an event that signifies the accomplishment or completion of a major deliverable during a project.
- ***Deliverable***: some concrete thing which is to be delivered, to the client or internally to the development team.
- ***Leveling***: the process of adjusting tasks to match resources available. There are two techniques for leveling: task delay and task split.
- ***Critical path***: a sequence of dependent tasks that have the largest sum of most likely durations. The critical path determines the earliest possible completion date of the project.
- ***Critical task***: a task found on the critical path. A critical task cannot be delayed without delaying the entire project schedule.
- ***Work Breakdown Structure***: a hierarchical decomposition of the project into phases, activities, and tasks.
- ***Project management plan*** (PMP): a document that describes how the project is to be executed, monitored and controlled, which includes creating a project work breakdown structure, identifying and planning to mitigate risk, identifying manners in which to effectively communicate with stakeholders and other project team members, and developing a plan to manage changes.

6.3. Project Analysis and Scheduling

Project scheduling is the process of converting a general or outline plan for a project into a time-based schedule based on the available resources and time constraints. Different techniques exist for analyzing and scheduling project activities.

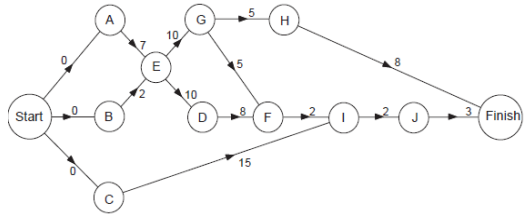
a. Critical Path Method (CPM)

CPM is an analysis technique used to predict project duration by analyzing which sequence of activities (which path) has the least amount of scheduling flexibility (the least amount of total float). Early dates are calculated by means of a forward pass using a specified start date while late dates are calculated by means of a backward pass starting from a specified completion date, usually the forward pass's calculated project early finish date.

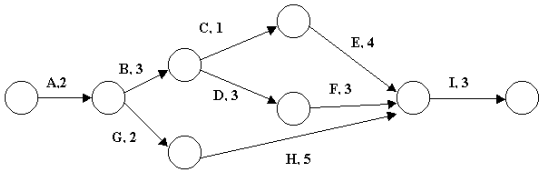
- ✓ ***Forward pass****: The calculation of the early start and early finish dates for the uncompleted portions of all network activities, determined by working forward through the schedule network logic from the project's start date.*
- ✓ ***Backward pass****: The calculation of late finish and late start dates for the uncompleted portions of all schedule activities, determined by working backward through the schedule network logic from the project's end date.*

CPM models the events and activities of a project as a network. Activities are depicted as nodes on the network and events that signify the beginning or ending of activities are depicted as arcs or lines between the nodes.

a)



b)



Network diagrams

In the above network diagram,

- Tasks are lettered from A to J. There are two ways of representing the activities on a network diagram: (a) activity on node and (b) activity on arc.
- Joining task A to E shows that task A must be completed before task E can be started.
- Joining tasks A and B to E shows that both task A and B must be completed before task E can be started.
- The number marked on each arc (arrow) shows the duration of the task from which the arc starts.

The critical path can be identified by determining the following four parameters for each activity:

- ES – earliest start time: the earliest time at which an activity can begin given that its predecessor activities must be completed first.
- EF – earliest finish time, equal to the earliest start time for the activity plus the time required to complete the activity.
- LF – latest finish time: the latest time at which an activity can be completed without delaying the project.
- LS – latest start time, equal to the latest finish time minus the time required to complete the activity.

Activities with the same earliest and latest start times (ES=LS) or with same earliest and latest finish times (EF=LF) define the critical path. This means that these activities have a float time of 0.

For the above network diagram, we have:

Activity	Duration	Start times		Float time
		Earliest	Latest	
A	7	0	0	0
B	2	0	5	5
C	15	0	12	12
E	10	7	7	0
D	8	17	17	0
F	2	25	25	0
G	5	17	19	2
H	8	22	24	2
I	2	27	27	0
J	3	29	29	0
Finish		32	32	

The critical path is A-E-D-F-I-J

The total estimated duration of the project = sum of duration of critical tasks = 7 + 10 + 8 + 2 + 2 + 3 = 32 *days*

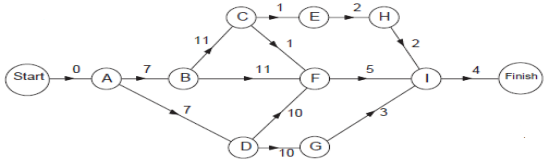
Exercise 1: Given the task description table below:

Activity	Duration	Precedence
A	3	-
B	3	A
C	4	-
D	1	C
E	3	B, D
F	2	A, B, D
G	2	C, F
H	4	G
I	1	C
J	3	E, G
K	5	F, H, I

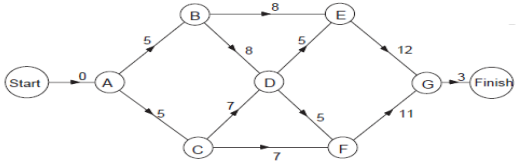
1. Draw the corresponding PERT diagram for this project
2. Determine the critical path
3. Calculate the total estimated duration of the project
4. State the float time for all non-critical activities

Exercise 2: Find the critical path for each of the activity networks below.

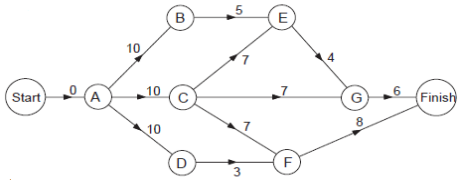
a.



b.



c.



Remark! CPM was developed for complex but fairly routine projects with minimal uncertainty in project completion times. For less routine projects there is more uncertainty in the completion times, and this uncertainty limits the usefulness of the deterministic CPM model. An alternative to CPM is the PERT (Program Evaluation and Review Technique) project planning model.

b. Program Evaluation and Review Technique (PERT)

PERT is an event-oriented network analysis technique used to estimate project duration when there is a high degree of uncertainty with the individual activity duration estimates. Each activity is assigned three time estimates

m= most likely time estimate, (mode)
a = optimistic time estimate (best case)
b = pessimistic time estimate (worst case)

These three estimates are then used to calculate a weighted duration for each task by using the formula$T_E = (a + 4m + b)/6$
The weighted durations are then used as a more realistic estimate of task durations for constructing a PERT chart (network diagram).
Using PERT, the probability of completing the project by a certain date t, can now be found by finding

$$P(t \geq T) = \frac{t - T}{\sqrt{\sigma_T{}^2}} = \frac{t - T}{\sigma}$$

Where
T is the expected completion time of the project
And

$$\sigma_T{}^2 = \left(\frac{b - a}{6}\right)^2$$

is the variance of T = Σ(variances of activities on the critical path).
Example:
If a project's expected completion time is $T = 246$ *days* with its variance $\sigma_T{}^2 = 25$, then what is the probability that the project

a) is actually completed within 246 days?
b) is actually completed within 240 days?
c) is actually completed within 256 days

Solution

a) $t = 246, T = 246$ and $\sigma_T{}^2 = 25$

$$P(t \geq T) = \frac{246 - 246}{25} = 0$$

b) $t = 240$, $T = 246$ and $\sigma_T{}^2 = 25$

$$P(t \leq T) = \frac{240 - 246}{25} = \frac{-6}{25} = -0.24$$

$$\therefore P(t \leq T) = 1 - (P(t < 0.24)) = 1 - 0.24 = 0.76$$

c) $t = 256$, $T = 246$ and $\sigma_T{}^2 = 25$

$$P(t \geq T) = \frac{256 - 246}{25} = \frac{6}{25} = 0.24$$

Definition: *A PERT chart/diagram is a graphic illustration of a project as a network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the project linked by labeled vectors (directional lines) representing tasks in the project. The direction of the arrows on the lines indicates the sequence of tasks.*

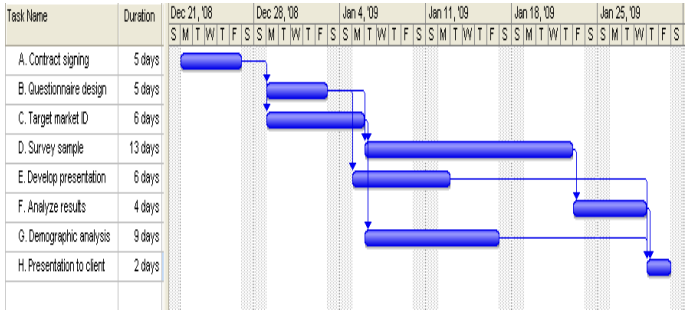
c. Comparison between CPM and PERT

CPM task durations are known with certainty. CPM is therefore said to be deterministic while PERT is probabilistic.

	CPM	PERT
1	Uses network, calculate float or slack, identify critical path and activities, guides to monitor and controlling project	Same as CPM
2	Uses one value of activity time	Requires 3 estimates of activity time Calculates standard deviation and variance of time
3	Used where times can be estimated with confidence, familiar activities	Used where times cannot be estimated with confidence. Unfamiliar or new activities
4	Minimizing cost is more important	Meeting time target or estimating percent completion is more important

d. Gantt Chart

A Gantt chart is a horizontal bar graph that helps plan and monitor project development or resource allocation on a horizontal time scale. It depicts project tasks against a calendar.
A Gantt chart is constructed with a horizontal axis representing the total time span of the project, broken down into increments (days, weeks, or months) and a vertical axis representing the tasks that make up the project. Horizontal bars of varying lengths represent the sequences, timing, and time span for each task. The bar spans may overlap, as, for example, you may conduct research and choose software during the same time span. As the project progresses, secondary bars, arrowheads, or darkened bars may be added to indicate completed tasks, or the portions of tasks that have been completed. A vertical line is used to represent the report date.



A Gantt chart

The critical path is A-C-D-F-H = 5+6+4+2 = 17 days

Exercises!

Exercise 1: A project has been defined to contain the following list of activities along with their required times for completion.

Activity No.	Activity	Time (weeks)	Immediate Predecessors
1	Collect requirements	3	
2	Analyze processes	2	1
3	Analyze data	2	2
4	Design processes	6	2
5	Design data	3	3
6	Design screens	2	3,4
7	Design reports	4	4,5
8	Program	5	6,7
9	Test and Document	7	7
10	Install	2	8,9

b. Draw a network diagram for the activities.
c. Calculate the earliest completion time of the project
d. Show the critical path.
e. What would happen if activity 6 were revised to take 6 weeks instead of 2 weeks?
f. Construct a Gantt chart for this project

Exercise 2: At 4:30 pm one day CRTV news team hear of a Government Minister resigning. They wish to prepare an item on the event for that evening's 6 o'clock news. The table below list the jobs needed to prepare this news item, the time each job takes and the constraints on when the job can commence.

Job	Time needed	Constraints
A - Interview the resigning minister	15 mins.	Starts at 4:30 pm
B - Film the ministry	20 mins.	None
C - Get reactions from regions	25 mins.	Cannot start until A and B are completed
D - Review possible replacements	40 mins	Cannot start until B is completed
E- Review the minister's career	25 mins	Cannot start until A is completed
F- Prepare film for archives	20 mins.	Cannot start until E and C are completed
G- Edit	20 mins.	Cannot start until A, B, C, D, E and F are completed

b. Construct an activity network for this problem and by finding the critical path, in your network, show that the news item can be ready before 6 pm that day.

c. Construct the corresponding Gantt chart for this project

CHAPTER 5: DATABASE DESIGN SYSTEMS AND MODELING

Introduction

A database is an organized collection of related data stored in a way that it can be easily retrieved and manipulated. A telephone directory, a library catalogue and a class register are examples of manual or paper-based database systems. A paper-based database requires much paper as the database becomes larger, making it difficult to manipulate the database. The problems caused by paper-based systems are solved by the development of computer-based systems.

The capacity for computers to store large amounts of data and their ability to quickly and efficiently retrieve the data makes them ideal for creating and using electronic or computerized databases. A computerized database refers to a collection of related files that are digitized. Computerized databases are created using database software called database management systems (DBMS).

1. Database Models

1.1. Flat-file Database

A flat file database is a single table database, with separate copies of data in each part of the business. An example is a phone directory.

The problems encountered with flat file databases are

- ✓ Data duplication: data is repeated and hence stored many times. This wastes disk space and slows down query time.
- ✓ Maintenance is difficult as every occurrence of a piece of data needs to be updated if its value changes
- ✓ More manual data entry is required and therefore a greater likelihood of errors when data is being entered.

The solution to these problems is to divide the data into logical groups and store the data in multiple tables, then connect (relate) the tables to each other. This results to a Relational database.

1.2. Relational Database Model

In a relational database, data is organized in two dimensional tables called relations, which are linked. A relation has the following features:

- ✓ Name: the name of the table
- ✓ Tuples: a tuple is a single row of a table, which contains a single record for that relation.
- ✓ Attributes: an attribute is a single column in the table, which contains a fact about each record in the relation.

1.3. Object-Oriented Database Model

An object-oriented database tries to keep the advantages of the relational model and at the same time allows applications to access structured data.

1.4. Hierarchical Database Model

2. Database Normalization

Database normalization is the process of organizing the fields and tables of a relational database to minimize redundancy and dependency. Normalization usually involves dividing large tables into smaller and less redundant tables and defining relationships between them. Normalization works through a series of stages known as normal forms. In order to achieve one level of normal form, each previous level must be met.

2.1. First Normal Form

A relation (Table) is in first normal form (1NF) if and only if

- It contains a primary key. A primary key is an attribute that identifies each entity in a unique way.
- It contains no multivalued field or repeating groups. A multivalued field is one that may take several values for a single record. A repeating group is a set of one or more multivalued attributes that are related.

For example:

StudID	Stud_Name	School	Subject	Grade
7023	Mary Watson	BGS Molyko	Chem, Phy, Maths	A, C, B
7140	Peter Rowling	BGS Molyko	Bio, Chem, Maths	B, C, D
7254	Quincy Greg	BGS Molyko	CSC, Maths, FMaths	A, C, D

Problems:

- ✓ The table has no primary key
- ✓ It contains multivalued fields “Subject” and “Grade”

To solve the problem

- ✓ We take StudID as the primary key
- ✓ Create new rows so each cell contains only one value

StudID	StudName	School	Subject	Grade
7023	Mary W.	BGS Molyko	Chem	A
7023	Mary W.	BGS Molyko	Phy	C
7023	Mary W.	BGS Molyko	Maths	B
7140	Peter R.	BGS Molyko	Bio	B
7140	Peter E.	BGS Molyko	Chem	C
7140	Peter E	BGS Molyko	Maths	D
7254	Quincy G.	BGS Molyko	CSC	A
7254	Quincy G.	BGS Molyko	Maths	C
7254	Quincy G.	BGS Molyko	FMaths	D

But still, there is a problem because StudID is no longer valid as primary key. It no longer identifies each row (record) uniquely.

To solve this problem, we declare StudID and Subject together to uniquely identify each row.

Now, the new primary key is StudID and Subject. It is a composite key.

2.2. Second Normal Form

A relation is in second normal form (2NF)if and only if

- It is in 1NF
- Every non-key attribute is fully dependent on the primary key. In other words, there should be no partial dependencies.

To put the data model to 2NF, we have to ensure that every non-key attribute is functionally dependent on the entire primary key. An attribute B is said to be functionally dependent on another attribute A if A determines B, and is written as AB.

{StudID} →{StudName}
{StudID} → {School}
{StudID} → {Grade}
{Subject} → {StudName}
{Subject} → {School}
{Subject} → {Grade}

StudName is dependent on StudID but it is not dependent on subject, the other part of the key.
School is neither dependent on StudID nor Subject.
Grade is dependent on Subject but to have the grade in a subject you need a StudID. Therefore, grade is dependent on both Subject and StudID.

To solve these problems, we create separate tables for the fields that are not functionally dependent on the entire key. The primary key for these tables is the part of the primary key on which they are dependent.

We now have,

STUDENT: (StudID, StudName)
SCHOOL: (SchoolID, SchoolName)
SUBJECT: (SubjCode, SubjTitle)
STUDENT_SUBJECT: (StudID,SubjCode, Grade)

2.3. Third Normal Form

A relation is in third normal form (3NF) if and only if

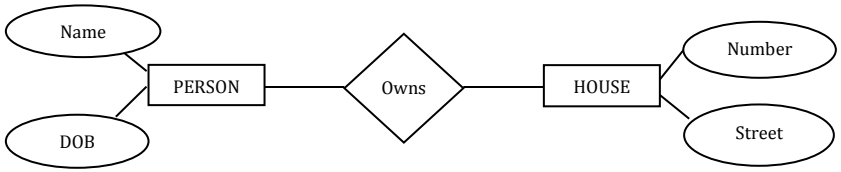
- It is in 2NF
- There are no transitive dependencies. Transitive dependency is a situation where a non-key attribute depends on another non-key attribute.

3. Entity-Relationship Modeling

When a relational database is to be designed, an entity-relationship model is drawn at an early stage and developed as the requirements of the database and its processing become better understood.

Definition*: An E-R model is a diagram which uses basic graphic symbols to show the organization of and relationships between data in a database.*

An E-R diagram serves as a schema diagram for the required database. A schema diagram is any diagram that attempts to show the structure of the data in a database.



An E-R diagram

The basic elements of an E-R diagram are entity sets, attributes and relationship types.

3.1. Entity Set

An entity is a person, place, concept or thing for which we intend to collect data. For example, a customer, an employee, a book, an appointment.
A group of entities that share the same properties is an entity set. An entity is therefore a member or an instance of an entity set. In an E-R diagram, an entity set is represented by a rectangle. In the above E-R diagram, PERSON and HOUSE are entity sets

3.2. Attributes

An attribute is a fact about an entity or a property that describes an entity. For example, a person's name, date of birth or gender, a vehicle's model, color or brand. Attributes store the actual data we want to keep about each entity within an entity set. An attribute is represented by an ellipse. In the above E-R diagram, name and date of birth are attributes of the entity set PERSON while Number and Street are attributes of the entity set HOUSE.
An attribute can be simple, composed, derived, single valued or multi-valued.

a. **Simple attribute**: simple attribute is an atomic value, which cannot be divided further. For example, a person's phone number is an atomic value of 9 digits.

b. **Composite attribute**: an attribute made of more than one simple attribute. For example, a person's complete name may have FName, MName and LName.

c. **Derived attribute**: an attribute which does not exist physically in the database, but its value is derived from other attributes present in the database. For example, a person's age can be derived from his date_of_birth.

d. **Single-valued attribute**: an attribute that contains one single value. For example ID car number.

e. **Multi-valued attribute**: an attribute than can contain more than one value for the same entity. For example, a person can have more than one phone numbers or e-mail addresses.

3.3. Relationship Type

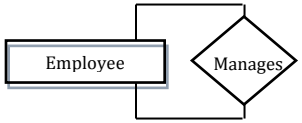
A relationship type is a named association between entities. A person (entity) owns (relationship) a house (entity), a teacher (entity) teaches (relationship) a subject (entity). Normally, individual entities have individual relationships of the type between them but in an E-R diagram, this is generalized to entity sets and relationship types. For example, the entity set PERSON is related to the entity set HOUSE by the relationship type OWNS.A relationship type is represented by a diamond.

A relationship can be characterized by a degree and cardinality ratio.

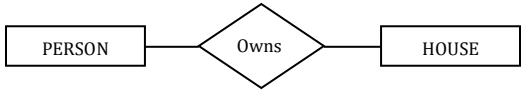
3.3.1. Degree of a Relationship

The degree of a relationship refers to the number of participating entities in the relationship. It can be unary, binary, ternary or n-ary.

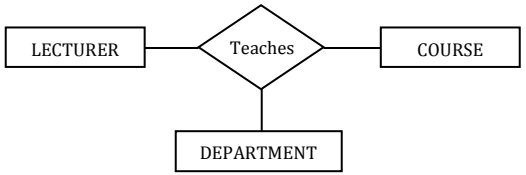
- ✓ A unary relationship type is one that involves entities from a single entity set. E.g. the relationship MANAGES between entities within the entity set EMPLOYEE.



- ✓ A binary relationship type is a relationship between entities from two different entity sets. An example is the relationship OWNS in the E-R diagram above.



- ✓ A ternary relationship type is one that involves entities from three different entity sets. An example is a LECTURER who teaches a certain COURSE in a DEPARTMENT.

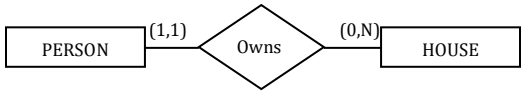


- ✓ An *n*-ary relationship type involves entities from *n* different entity sets.

3.3.2. Cardinality Ratio of a Relationship Type

Cardinality is the maximum number of entities within each entity set that can take part in a relationship. For example, what is the maximum number of houses that a person can own? On the other hand, the minimum number of entities within each entity that can take part in the relationship is the Optionality. For example, what is the minimum number of houses that can be owned by a person?
The cardinality ratio is a ratio of the cardinalities of the entity sets involved in a relationship. It can be one-to-one (1:1), one-to-many (1:N) or many-to-many (M:N).

- ✓ In the relationship OWNS between PERSON and HOUSE, a person can own zero or many houses. Therefore, the cardinality of PERSON in the relationship OWNS is many while the optionality is zero.
  On the other hand, a house is owned by one person. The cardinality of HOUSE in the inverse relationship IS_OWNED_BY, is one while the optionality is one. The relationship OWNS is therefore described as a one-to-many (1:N) relationship.



(1:N) relationship

- ✓ In the relationship RECEIVES between STUDENT and SLIP, each student receives one and only one result slip. The cardinality of student is one and the optionality is one. Each result slip is issued to one and only one student. The cardinality of SLIP is one and the optionality is one. This relationship is described as one-to-one (1:1).

(1:1) relationship

- ✓ In the relationship TEACHES between TEACHER and SUBJECT, a teacher teaches one or many subjects. The cardinality of TEACHER is many and the optionality is one.
  A subject is taught by one or many teachers. The cardinality of SUBJECT in the inverse relationship IS_TAUGHT_BY, is many and the optionality is one. This relationship is described as many-to-many (M:N).

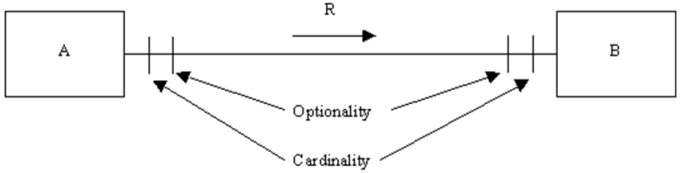
(M:N) relationship

Remark In a normalized database, many-to-many relationships are eliminated by creating a link entity between the entities involved in the relationship.
The above TEACHES relationship when normalized becomes

Normalized (M,N) relationship

The primary key of the link entity is a composite key that consists of the primary keys of the entities TEACHER and SUBJECT.

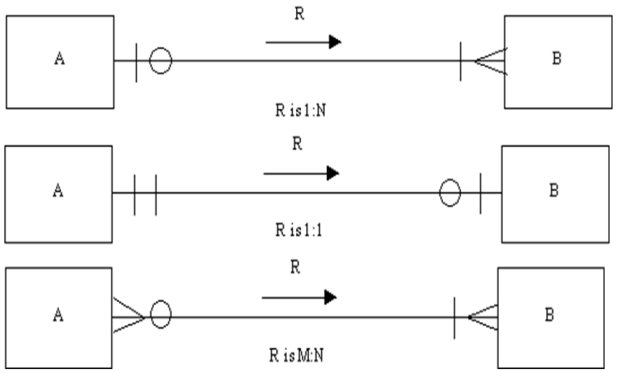
The above representation of an E-R diagram is the Chen Convention. Another way of representing E-R diagrams is the crow's foot notation that uses three symbols to show cardinality ratios. Here, a circle means *zero*, a line means *one* and a crow's foot means *many*. The cardinality is shown next to the entity type and the *optionality* (if shown at all) is shown behind it.



Where A and B are entity sets and R is the relationship type.

Exercise! Exercise! Exercise!

1) Identify the cardinality ratios of the following relationships.



2) A database will be made to store information about patients in a hospital. On arrival, each patient's personal details (name, address, and telephone number) are recorded where possible, and they are given an admission number. They are then assigned to a particular ward (Accident and Emergency, Cardiology, Oncology, etc.). In each ward there are a number of doctors and nurses. A patient will be treated by one doctor and several nurses over the course of their stay, and each doctor and nurse may be involved with several patients at any given time.

   From the description, draw a corresponding E-R diagram showing all entity sets, attributes, relationship types and cardinality ratios.

4. Database Management Systems

A database management system (DBMS) is a computer program that is used to create, manipulate and access a database. Examples are MS Access, MySQL, Oracle, Sybase, dBase, Paradox, Objecstore and O_2.

A DBMS provides users with:

- ✓ Data Definition Language (DDL): language which programs the structure (schema) of the database
- ✓ Data Manipulation Language (DML): language which changes the contents (instance) of the database
- ✓ Data Control Language (DCL)

4.1. Microsoft Access 2007

MS Access 2007 is made up of several components including:

- ✓ Tables
- ✓ Forms
- ✓ Queries
- ✓ Reports

These components are called database objects. One or more of these components are formed when a database is created. These components are stored in a single database file.

4.1.1. Tables

A table is an area of the database where information is stored. Tables organize data into columns (called fields) and rows (called records). Example: a database storing information about books and authors for a library could have a "BOOKS" table to store information about books (title, publisher, Year, etc) and an "AUTHORS" table to store information about authors (name, birth date, birth place, etc).

- ✓ A field is a specific piece of data stored in a table. The above mentioned 'BOOKS' table has fields 'Title', 'Publisher', 'Year' etc.

- ✓ A record is an individual entry in a database table. It comprises of one or more fields, depending on the number of fields defined to the table. For example: if five books are entered in our books and authors database, the "BOOKS" table would have five records (one for each book).

- ✓ Each field in a database table is assigned a data type when the field is created. The data type determines the kind and format of the information stored in the field. Example: the "Title" field in the "BOOKS" table will store text that will vary with each book in the database, so the data type in this case will be text or character type.

a. Creating a New Table

To create a new table,

- Click the Create Tab
- Select Table in the Tables group to create a new table in Datasheet view or Table Design to create a new table in Design view

b. Adding Fields in a Table

To add a field to a table in Datasheet View,

- Click the Add New Field column label.
- Click Rename in the Fields & Columns group.
- Type the field name and press Enter. Access creates the field. Continue until you have created all of the fields in your table.
- Press Enter without entering a field name to end your entries.

To add a field to a table in Design View,

- Select a cell in the Field Name column
- Type a name for the first field in the table and press Enter
- Select a data type and press Enter
- Type a description and press Enter

c. Naming and Saving a Table

To name and save a table,

- Click the Save button on the Quick Access toolbar. The Save As dialog box appears.
- Type the name you want to give your table.
- Click OK. Access names your table.

Tip: You can use the Rename option at any time to rename any column. For example, you can rename the ID column Author_ID.

d. Setting a Primary Key

A primary key is a field in a table that allows each entry in the table to be identified in a unique way. No two entries in the database can have the same primary key. For example your ID card number is unique. By default, Access sets the first field in the table as the Primary Key field.

To set a Primary Key:

- Switch to Design View

- Position your cursor in the field you wish to set as the Primary Key
- Click the Primary Key button on the Ribbon

e. The Foreign Key

The primary key is used to link the table with other tables in the database. When a primary key in one table (known as the source table) is linked to another table (known as the target table), the connecting field in the target table is called the foreign key. A foreign key must have the same structure, data type, and field size as the associated primary key, but it must not have the same name. Also, the foreign key in a relationship between two tables need not be a primary key in its own table.

4.1.2. Relationships

Relating tables reduces the unnecessary duplication of data (redundancy).The general procedure for creating relationships in Access is to

- Determine the type of the relationship and identify the source and target tables
- Create the foreign key field(s) in the target table if they are not already present
- Select the Database Tools tab
- Click on the Relationships command
- Add all the tables involved in the relationship to the window
- Create the relationships by dragging the primary keys from the source table(s) and dropping them on the associated foreign key(s) in the target table(s).

For example: Let's create a relation between the AUTHORS table and the BOOKS table.

4.1.3. Queries

A query is a command that tells the database what data the user wants to see. A query allows information to be found and retrieved from one or more tables based on a set of search conditions you define. For example: a query can be written specifying that the database should retrieve all the books from the "BOOKS" table written by William Shakespeare.

To create a query,

- Click the Create tab on the Ribbon
- Click the Query Design command
- Double-click Create Query in Design View
- Select the table that you would like to base your Query on
- Click Add
- Close the Show Table window

In order to control which records are displayed, you must define criteria in a Query.
To define criteria for your query:

- Position your cursor in the criteria row in the field for which you wish to define the criteria for
- Enter the criteria
- Click the Run Query button

For example: a query to find the books written by Barack Obama

The result of a query is called a recordset.

To Save the Query**:**

- Click the Save Icon
- Enter a name for the query
- Click OK

4.1.4. Forms

A form is an interface that allows users to enter and view data. Forms give the ability to choose the format and arrangement of fields. They can be used to enter, edit and display data. Forms can retrieve data from one or more tables, and display the output on the screen.

To create a Form using the Form Wizard

- Navigate to the table you want to base the form on
- Click Create on the Ribbon
- Click Forms

You are able to navigate using the navigation arrows at the bottom of the form.

To enter a record on the Form:

- Click the View button on the Ribbon to switch from Layout View to Form View
- Enter the data for each field in the record, pressing the Enter key to move to the next field
- Press Enter after you have entered data for the last field

NB. The form feeds the table. If you edit a record on the form, or create a new record, that data will be passed to the table it is associated with.

4.1.5. Reports

A report is an effective way to analyze and present data in printable format using a specific layout. Reports are the primary method of retrieving and viewing information. Using queries, reports pull information the user has requested and either print that information or display it on screen. The information that appears and the formatting and appearance of a report are set when the report is created.

To create a Report using the Report Wizard:

- Click the Create tab on the Ribbon
- Click the Report Wizard command
- Select the table or query upon which the report will be based
- Select the fields that you want to include on the report by double clicking on them
- Click Next
- Select a style for the report
- Click Next
- Type a title for the report

Click Finish

4.2. Structured Query Language

Structured query language (SQL)

CHAPTER 6: ALGORITHM DESIGN AND PROGRAMMING

1. Data Types and Data Structures

1.1. Data Types

Computers manipulate data of different types like numbers, letters, sound and images. No matter the type, each data is stored as a pattern of binary digits. Numbers can be added, subtracted, divided and multiplied whereas letters cannot. This implies that data of a given type will be treated differently from data of another type. As such, the type of any data stored in the computer must be specified so that the correct operations will be performed on it.
A data type is defined as a set of values and a collection of operations on those values. Examples of data types are integer, real, string and Boolean.

a. Integer

An integer is a whole number that can be positive, negative or zero. Examples: -3, 5, 67, -134 and 4231.

b. Real

A real is a number that contains a decimal point. Computers manipulate numbers with fractions as floating point numbers. Examples: 3.14, 12.25 and 0.001.

c. Boolean

A Boolean data type stores one of only two values – true or false, yes or no, or on or off. Usually these values are stored as 1 for true and 0 for false.

d. Character

A character is any letter, digit, space, punctuation mark, or symbol that can be typed on a computer. The word "computer" for example, consists of eight characters. The phrase "data type" takes up nine characters. A character data type stores only a single character or digit. Each character requires one byte of space, so "computer" takes up 8 bytes ($8 \times 8 = 64\ bits$). Examples: A, 5, -, and !.

e. String/Text

A string is a set of characters grouped together. A string is sometimes just referred to as 'text'. It can contain a mixture of numbers, letters and, spaces. Numeric data stored as text is only for representation not for calculation. Any type of alphabetic or numeric data can be stored as a string. Examples: "Limbe City", "10base-2" and "36.85".

f. Date/ Timestamp

Date and timestamp are data types for storing date and time respectively. The format used for date is either dd-mm-yy or mm-dd-yy and hh:mm:ss or ss:mm:hh for time.

g. Container

A container is a data type used for storing images, video, sound or another type of 'complex' file. In some languages, binary large object (blob) is used rather than container.

1.2. Data Structures

A data structure is a way of organizing and storing data in memory so that it can be used efficiently. Any data structure is designed to organize data so that it can be accessed and worked with in appropriate ways. A well-designed data structure allows a variety of critical operations to be performed, using as few resources, both execution time and memory space, as possible. Data structures include arrays, records, stacks, queues, trees, graphs and hash tables.

2.1.1. Arrays

An array is a data structure, which allows a set of items of identical data type to be stored together using the same identifier name. When an array is created, its dimension (size) and the type of its elements are specified.

For example: **String** Names[5]
Names : **Array** [1..5] of **string**

The above statements create an array called "Names" of five elements in C and Pascal respectively. The size of the array is 5 and its elements are of type string.
To reference an element in an array, an index is used. An index is an integer that gives the element at the indexed position in the array.
For example in C, Names[0] references the first element in the array Names, Names[1] references the second element, Names[2] references the third and so on…
In Pascal, the referencing starts from 1 rather than 0.

a. One-dimensional Arrays

A one-dimensional array is an array which is declared using a single index and can be visually represented as a list. The array "Names" above is one-dimensional.
If the array "Names" contains five elements (names), Raissa, Peter, James, Leticia and Isaac, it can be visually represented as follows:

Element	Raissa	Peter	James	Leticia	Isaac
Index	[0]	[1]	[2]	[3]	[4]

Names[0] == Raissa, Names[4] == Isaac

b. Two-dimensional Arrays

A two-dimensional array is an array which is declared using two indices and can be visually represented as a table.
For example:

int Tab[3][2] or Tab : **Array** [1..3, 1..2] of **integer**
3 is the number of rows and 2 is the number of columns in the table Tab.

The elements of Tab will be referenced as:
Tab[0][0] Tab[0][1]
Tab[1][0] Tab[1][1]
Tab[2][0] Tab[2][1]

Example: Given the 2-dimensional array Tab[4][2] below, what are the elements given by the references Tab[0][0], Tab[1][1], Tab[3][0] and Tab[2][1]?

Index	[0]	[1]
[0]	5	3
[1]	1	4
[2]	3	1
[3]	2	8

Each individual element can be referenced by its row and column indices. For example:
Tab[0][0] = 5
Tab[1][1] = 4
Tab[3][0] = 2

2.1.2. Record

A record is a data structure which allows items of different types that define a particular structure or object, to be stored together. The elements of a record are usually called *fields* or *members*.
Defining a record includes specifying the data type of each field and a name (label) by which the field can be accessed.
For example,

```
Date= RECORD
   Day: Integer
   Month: String
   Year: Integer
End RECORD
```

```
Student = RECORD
   Name: String
   Gender: Character
   DOB: Date
End RECORD
```

```
struct Student
  {
    string  Name
    char    Gender
    Date    DOB
  }
```

The above Date record has a numeric field Day, a text field Month and a numeric field Year. The Student record has fields Name, Gender and DOB which is of type Date. The table below gives examples of student records.

Name	Gender	DOB
Raissa	F	01 Apr 1996
Peter	M	17 Oct 1995
James	M	05 May 1995
Leticia	F	09 Dec 1997
Isaac	M	29 Feb 1995

} Records

Each row in the table corresponds to a different record. This table has 5 records.
A field in a record is referenced as follows:

RecordName.fieldName = “”
Raissa.DOB = 01 Apr 1996

Record data structures are always used in association with arrays. That is, we define an array whose elements are of type record. When this is done, we have an *array of records*.

For example: let’s define an arrayClassList, which stores information about the students in the table above.

Student ClassList[5] or ClassList : **Array** [1..5] of **Student**

An element in this type of array is referenced as follows:
ArrayName[index].fieldName

ClassList[0].Gender == F; ClassList[4].DOB == 29 Feb 1995; ClassList[3].DOB.Month == Dec

2.1.3. Stack

A stack is a linear list in which items are added and removed from the same end (head). Items being added and removed from the same end means that the last item to be added will be the first to be removed. As such stacks are also called last in, first out (LIFO) lists as they use the LIFO principle. (FILO)
Stack operations are:

- ✓ Stack(S): creates an empty stack named S
- ✓ Push(S,x): inserts the element x to the top of the stack S. Inserting an element into a stack that is full leads to a situation known as overflow. (START == NULL)
- ✓ Pop(S): removes an element from the top of the stack S. Removing an element from an empty stack leads to a situation known as underflow. (START == HEAD)
- ✓ IsEmpty(S): checks whether the stack S is empty. Returns true if it is empty and false otherwise.

Application of stacks:

a. Converting a Decimal Number to Binary

Initially we push the binary digit formed into the stack, instead of printing it directly. After the entire digit has been converted into the binary form, we pop one digit at a time from the stack and print it. Therefore we get the decimal number converted into its proper binary form.

b. Evaluating Arithmetic Expressions

Arithmetic expressions are usually written in the form$(a + b)$, where the operator is in between the operands. This is known as the infix notation. This notation poses problems for more complicated expressions. For example:

$$a \times b + c = \begin{cases} (a \times b) + c? \\ a \times (b + c)? \end{cases}$$

$$2 \times 6 + 5 = \begin{cases} (2 \times 6) + 5 = 17? \\ \quad or \\ 2 \times (6 + 5) = 22? \end{cases}$$

Infix notation requires the use of order of precedence of operators and parentheses making it complicated and difficult for computers to evaluate expressions in this form.
To ease evaluation of arithmetic expressions, other notations are used: the prefix and postfix notations.
In prefix notation also called polish notation, operations are written before the operands. For example the expression $(a \times b) + c$ will be written in prefix notation as

$$(a \times b) + c = (\times ab) + b$$
$$= + \times abc$$

In postfix notation also called reverse polish notation, operands come before operators. For example
$(a \times b) + c = (ab \times) + c = ab \times c +$ $(2 \times 3) + 1 = 23 \times 1 +$
Also,
$a \times (b + c) = a \times (bc +) = abc + \times$ $2 \times (3 + 1) = 231 + \times$

Prefix and postfix notations do not require parentheses or precedence rules. Calculators use postfix notation to evaluate arithmetic expressions. A simple way to understand is by using the following expression: **Pre** A **In** B **Pos.**

To convert from infix to prefix or postfix, priorities are assigned to operators as follows: $priority(\times) = priority(/) > priority(+) = priority(-)$ and the following rules are used.

- When an operand lies between two operators, it associates with the operator that has higher priority.
- When an operand lies between operators of same priority, it associates with the operator on the left.

Example 1: Let's convert the expression P: $A + B \times C/D$ to postfix.
Considering the priority rules given above, we can rewrite P as follows:
P:$A + ((B \times C)/D)$

Stack[empty]
Output: [ ]

Read A	push it into stack ➔stack[A]	output: $[A]$
Read +	➔stack[+]	output: $[A]$
Read (	➔stack[+(]	output: [ ]
Read (	➔stack[+((]	output: [ ]
Read B	➔stack[+((]	output: $[AB]$
Read *	➔stack[+((*]	output: $[AB]$
Read C	➔stack [+((*]	output: $[ABC]$
Read)	➔stack [+(]	output: $[ABC \times]$ pop until you meet an opening)
Read /	➔stack [+(/]	output: $[ABC \times]$
Read D	➔stack [+(/]	output: $[ABC \times D]$
Read)	➔ stack [+]	output: $[ABC \times D/]$
	➔stack[ ] output:	$[ABC \times D/+]$

$A + ((B \times C)/D)$ in postfix notation is $ABC \times D/+$

Example 2: Let's evaluate the expression $234 \times 5 + -$ written in postfix notation.

Read 2	➔ stack [2]
Read 3	➔ stack [2, 3]
Read 4	➔ stack [2, 3, 4]
Read ×	➔ stack $[2, 3 \times 4 = 12]$= [2, 12]
Read 5	➔ stack [2, 12, 5]
Read +	➔stack $[2, 12 + 5 = 17]$ = [2, 17]
Read −	➔ stack $[2 - 17 = -15]$ = $[-15]$

The result of the expression $234 \times 5 + -$ is -15

Exercise: Convert the following to postfix notation and evaluate them.

i)	$2 - 3 \times 4 + 5$	$(34 \times 2 - 5+)$
ii)	$(2 - 3) \times (4 + 5)$	$(23 - 45 +\times)$
iii)	$2 \times 3/(2 - 1) + 5 \times 3$	(23x

Exercise 2: Evaluate the following expressions

i) $623 + -382/+ \times 2\text{^}3 +$
ii) $12, 7, 3, -, /, 2, 1, 5, +, \times, +$ (June 2013, Paper 2, Q5)

2.1.4. Queues

A queue is a linear list in which items are inserted at one end (tail/rear) and removed from the other end (head/front) such the first item added to the queue is the first to be removed. Queues are also called FIFO lists as they use the FIFO principle. (LILO)
Queue operations include:

- Create(Q): creates a queue Q.
- Enqueue(Q,x): inserts the element x to the tail of the queue Q.
- Dequeue(Q): removes the first element from the head of the queue Q
- Top(): checks the next element to be removed (do not remove)
- IsEmpty(): checks whether the queue is empty. Returns true if it is empty and false otherwise.

Application of queues:

a. Spooling

It is a method used to place input and output on a fast access storage device, such as a disk, so that slow peripheral devices do not hold up the processor. For example, print spooler stores output to be printed in a queue while waiting for the user's program to finish creating the output. The spooler then sends the stored output to the printer at the proper speed. If new output arrives for printing, it is added to the queue.

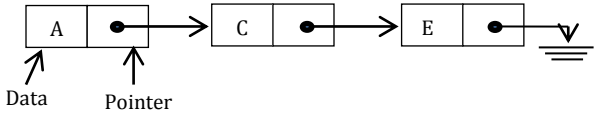
b. Scheduling

It is the process of determining which of the jobs in memory will be executed by the CPU. When a job is created, it is added to a queue, the ready queue. Any other job that arrives is added to the tail of the queue. A scheduling algorithm like round-robin is then used to allocate the jobs one by one, to the CPU for execution.

Remark A queue in which elements can be added and removed from any end is known as a double ended queue (Deque).

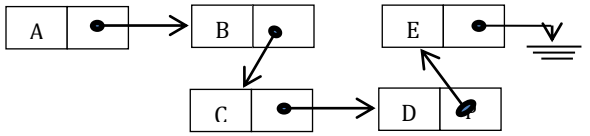
2.1.5. Linked Lists

A linked list is a linear list in which each element contains a pointer to the next element in the list. Each element in a linked list is known as a node and consists of two parts: the data item and the pointer to the next element.



A linked list that stores characters

Linked lists make it possible to insert items in the middle of the list without moving other items to make room.



Linked list with elements C and D inserted.

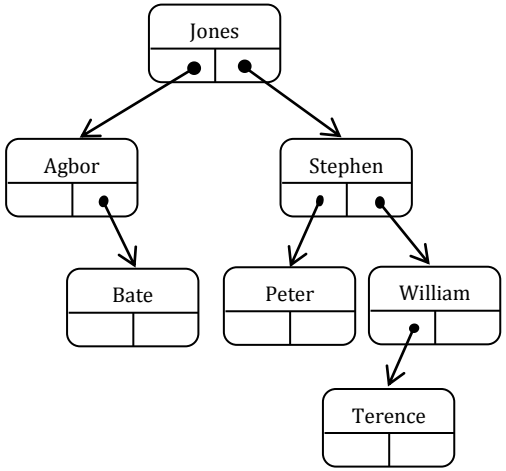
NB: In a double linked list, every node has two pointers: one for its predecessor and another for its successor in the list.

2.1.6. Binary Tree

A binary tree is a finite set of nodes that is either empty or consists of a root and two disjoint binary trees called the left sub-tree and the right sub-tree. In other words, it is a non-linear linked list where each node may point to two other nodes. Binary trees are a very efficient way of storing items that must be searched for and retrieved quickly. Suppose, for example, that you want to store the following names in a computer using a binary tree:

Jones, Stephen, Agbor, William, Bate, Peter, Terence

The names can be arranged into a binary tree by using the following two-step procedure:
1. Use the first name on the list as the root of the tree.
2. To find where to put each subsequent name, start at the root of the tree. If the name you are dealing with precedes the root name alphabetically, follow the left pointer; otherwise follow the right pointer. Proceed in this way until you come to an empty pointer; attach the name to it.



A binary tree that stores names

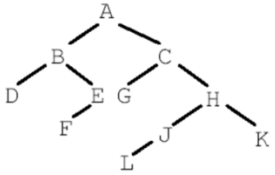
The topmost node in the tree is known as the *root node*. Each node in a binary tree may have at most two *children* or *child nodes*. A node that has a child is called the *child's parent node* (or ancestor node, or superior). A node has at most one parent except the root node that has no parent. Nodes that have the same parent are called *siblings*. Every node in a tree can be seen as the root node of the sub-tree rooted at that node.
Nodes at the bottom most level of the tree are called leaf nodes**.** Since there are at the bottom most level, they will not have any children.

Tree traversal is the process of systematically visiting all the nodes in a tree and performing some computation. When describing a traversal strategy, we need not concern ourselves with what that computation is. There are two methods for traversing a tree: breadth first traversal and depth first traversal.

a. Breadth First Traversal

In a breadth first traversal all of the nodes on a given level are visited and then all of the nodes on the next level are visited, usually in a left to right fashion. It is also called level order traversal.

Consider the following binary tree.



A level order traversal of this tree gives: ABCDEGHFJKL

b. Depth First Traversal

In depth first traversal, the left subtree is traversed, the right subtree is traversed and the root is visited. There are three different depth first traversal techniques; preorder, in order and post order traversals. What distinguishes them is the order in which the subtrees and the root are visited.

- ✓ Preorder Traversal:
  - Visit the root
  - Traverse left subtree
  - Traverse right subtree

A preorder of the above tree gives: ABDEFCGHJLK

- ✓ Inorder Traversal:
  - Traverse left subtree
  - Visit the root
  - Traverse right subtree

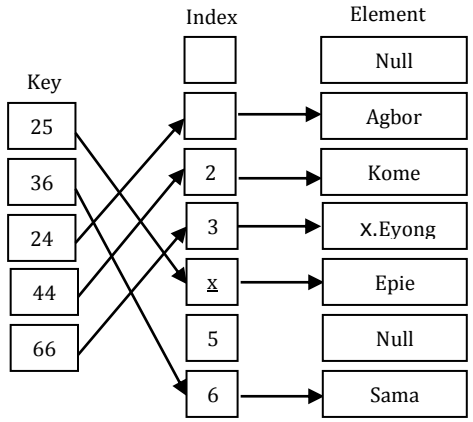
An in order traversal of the tree above gives: In order traversal: DBFEAGCLJHK

- ✓ Postorder Traversal:
  - Traverse left subtree
  - Traverse right subtree
  - Visit the root

A post order traversal of the tree above give: DFEBGLJKHCA

2.1.7. Hash Table

A hash table is an array in which data is stored at specific locations designated by a hash function. A hash function is a function that transforms the value of a record key into an index that corresponds to a location for storing the record. A hash function maps the set of input data to a set of integers. Each element to be stored in the array has a unique key that is mapped by the hash function to a numeric value that represents an index in the array.
For example,



A hash table

The above hash table stores the names Agbor, Epie, Sama, Kome and Eyong using the hash function *h(k) = k mod tablesize*, where k, the key of each string is the sum of its letters' positions in the alphabet. This means that if an element *p* has key *k*, then *p* will be stored at position *h(k)* in the table.
Keys:

Agbor = 1+7+2+15+18 = 43
Epie = 5+16+9+5 = 25
Sama = 19+1+13+1 = 34
Kome =11+15+13+5 = 44
Eyong = 5+25+15+14+7 = 66

Applying the function *h* to each key we get:

Agbor = 43 mod 7 = 1
Epie = 25 mod 7 = 4
Sama = 34 mod 7 = 6
Kome = 44 mod 7 = 2
Eyong = 66 mod 7 = 3

Hash table operations are:

- Search: compute $h(k)$ and see if an element exists
- Insert: compute $h(k)$ and place element in the resulting position
- Delete: compute $h(k)$ and remove element in that position

Remarks! The size of the array should be preferably a prime number.
With hash tables, there always exists the possibility that two data elements will hash to the same integer value. This situation is known as collision. Two methods to solve collision are separate chaining and probing (closed hashing).

3. Algorithms

An algorithm is a well-defined set of step-by-step instructions for solving a problem in a finite amount of time. A set of instructions is not an algorithm if there is no definite stopping place, or if the instructions are too vague to be followed clearly.
A good algorithm:
- should be explicit (i.e. clear and obvious)
- should be precise (i.e. exact and accurate)
- should be unambiguous (i.e. no doubts about what to do/ only one way of interpreting the instructions)
- should be effective (i.e. produce good results)
- should be finite (i.e. have a definite stopping place)

3.1. Representation of Algorithms

There are different ways for representing algorithms such as pseudo code, flow chart and structured English.

3.1.1. Pseudo code

A pseudo code is an outline of a computer program, written in a mixture of a programming language and English. Writing pseudo code is one of the best ways to represent an algorithm as it allows the programmer to concentrate on how the program works while ignoring the details of the language.
In a pseudo code, some terms are commonly used to represent the various actions. For example, for inputting data the terms may be (INPUT, GET, READ), for outputting data (OUTPUT, PRINT, DISPLAY), for calculations (COMPUTE, CALCULATE), for incrementing (INCREMENT), in addition words like ADD, SUBTRACT, INITIALIZE are used for addition, subtraction, and initialization, respectively.
For example, here is a pseudo code for an algorithm that reads two numbers, computes and displays their sum:

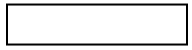
```
Begin
 Get A
 Get B
 Result = A + B
 Print Result
End
```

3.1.2. Flowchart

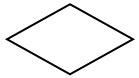
A flow chart is a diagram that uses graphic symbols to describe the nature and flow of steps in an algorithm. Each step in a flowchart is followed by an arrow that indicates which step to go next. The following symbols are used in flow charting:

a. Elongated circle: indicates the beginning (start) or end (stop) of the algorithm

b. Rectangle: indicates instructions or actions

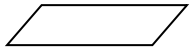


c. Diamond: indicates a decision that has to be made



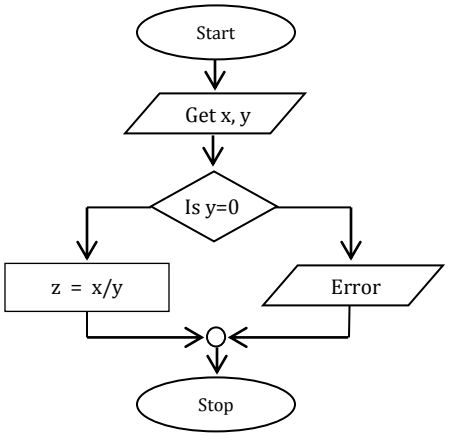
d. Arrow: indicates the direction of flow

e. Parallelogram: indicates data input and output



f. Circle: serves as a connector

Example: flow chart for an algorithm that divides two numbers x and y.



3.2. Variables, Constants and Literals

A variable is an object in a program whose value can be modified during the execution of the program. In the above flow chart, x, y and z are variables.

A constant is an object whose value cannot be modified in the course of the algorithm or program. A constant is given a value that remains the same all through the program.

Variables and constants are characterized by:

- an identifier: which is the name of the object
- a value: which is the content of the object
- a type: which defines the domain in which the object gets its value

A literal is anything (numbers or text) that is usually written within double quotes. For example, "Enter a number", "The result is".

3.3. Basic Instructions

Three basic instructions used in an algorithm are input, output and assignment instructions:

- An input instruction allows information to be typed from the keyboard. Example: read (a, b), get (number)
- An output instruction allows:
  - ✓ display of information on the screen
  - ✓ printing of information on paper

  For example: write "a is greater than b" or print "b is greater than a"
- The assignment statement allows a value to be assigned to a variable. A variable can be assigned the content of another variable, a constant, a literal, an arithmetic or Boolean expression. The symbol used is ←.

Examples: $z \leftarrow x/y, sum \leftarrow a + b, total \leftarrow total + 1, pi \leftarrow 3.14$
In an assignment statement, the value to the right is assigned to the variable in the left.
In the case of $sum \leftarrow a + b$, "$a + b$" is calculated and the result is assigned (kept) in the variable sum.
For $total \leftarrow total + 1$, "$total + 1$" is calculatedand the result is assigned to the variable total,meaning that the value of total has been increased by 1.
E.g. Let $total = 3$
$total \leftarrow total + 1 \Rightarrow total = 3 + 1 = 4$.

Activity: Complete the table below.

$Command$	x	y	z
$x \leftarrow 2$		–	–
$x \leftarrow x + 1$		–	–
$y \leftarrow 1$			–
$z \leftarrow x + y$			
$y \leftarrow z + x$			

3.4. Control Structures

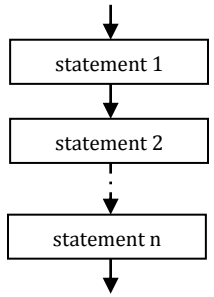
The logic of a program may not always be a linear sequence of statements to be executed in that order. The logic of a program may require execution of a statement based on a decision. It may repetitively execute a set of statements unless or until some condition is met. Control structures specify the statements to be executed and their order of execution.

3.4.1. Sequential Control

A sequence control structure executes a set of instructions one after the other from the first to the last in the order they are given.

Syntax: ***begin***
statement 1
statement 2
...
statement n
end

```
Begin
  get a
  get b
  c← a + b
  print "Sum =", c
End
```

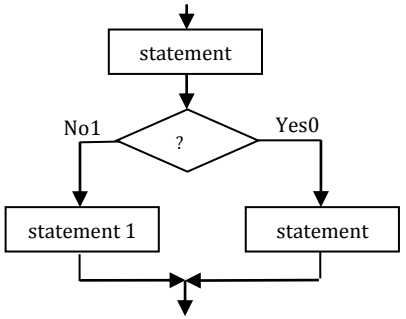


3.4.2. Selection Control

A selection control structure (condition control structure) chooses the instruction or instructions to be executed based on the validity of a certain condition. Examples are IF and CASE statements.

b. The IF Statement

Syntax: ***if*** *condition* ***then***
statement 1
else
statement 2



Explanation *Condition* is a Boolean expression meaning that it can take only one of two values true or false. The *condition* is evaluated, if it is true, *Action3* is executed. If it is false, *Action 2* is executed.
Note that actions 2 and 3 could be a block statements.

E
```
Get a, b
if (a = 0) then
  Print "Error"
else
  Print b/a
```

It is possible to nest many selection structures.

Syntax: i***f*** *condition1* ***then***
if *condition2* ***then***
statement 1
else
statement 2
else
statement 3

E
```
get a, b
if a <> 0 then
if b <> 0 then
   print b/a
else
  print "Answer is 0"
else
 print "Error: division by 0"
end
```

Explanation If *condition1* is true, we move to *condition2*. If *condition2* is true, then *statement 1* is executed otherwise, *statement 2* is executed. If condition 1 is false, instruction 3 is executed. Instruction1 or instruction 2 will be executed if and only if condition 1 is true.

c. The CASE Statement

Syntax: ***case*** *variable* ***of***
case 1: statement 1
case 2: statement 2
…
case n: statement n
end case

Explanation The value of variable is evaluated, if it matches with case 1, instruction 1 is executed. If it matches with case 2, instruction 2 is executed and so on. CASE is a multiple selection structure. It is used when an important number of choices are to be considered depending on the value of a variable.

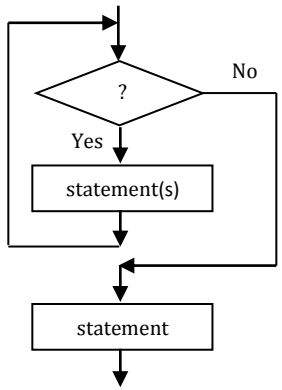
E
```
Get a, b
Get op
caseop of
   1: print a+b
   2: print a*b
   3: print a/b
   4: print a-b
end case
```

3.4.3. Repetition Control

The repetition (iteration) control structure executes a statement or group of statements many times until a certain condition is reached. Repetition structures define the order of operations and the number of repetitions. They are also called loops. Examples are, the WHILE, REPEAT and FOR loops.

a. The WHILE Loop

Syntax: ***while*** *condition* ***do***
statement(s);
end while

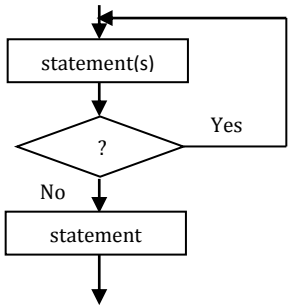


Explanation The *condition* is evaluated, if it is true statement(s) is/are executed. Instruction(s) is/are executed as long as condition remains true. When the condition becomes false, the loop stops. The condition for the loop to stop comprises of a variable called control or iteration variable whose value must change at the end of each execution of the loop. In the example above, the control variable is "i".

F
```
Get n
i← 1
while (i <= n) do
print "this is a while loop"
i←i + 1
endwhile
```

b. The REPEAT Loop

Syntax: ***repeat*** *statement(s);* ***until*** *(condition)*



Explanation The statement(s) is(are) executed and the condition is evaluated. If it evaluates to false, the statement or set statements is/are executed again. If condition evaluates to true, the program exits the loop.

```
Get n
i← 1
repeat
print "This is a repeat loop"
i := i + 1
until(i <= n)
```

Remark! The repeat loop must be executed at least once as the condition is evaluated only at the end of the loop.

c. The FOR Loop

Syntax: ***for*** *var* ← *low_limit* ***to*** *hi_limit* ***do***
statement(s);
end for
Or
for *var* ← *hi_limit* ***downto*** *low_limit* ***do***
statement(s);
end for

Explanation var (variable) is given a value *low_limit* or *hi_limit* depending on the loop, which is automatically incremented or decremented (by 1) after each iteration of the loop. The loop stops when *low_limit* becomes greater than *hi_limit*. In both cases, if *hi_limit* is less than *low_limit*, the loop body is not executed at all.

E

```
Get n
For i ← 1 to n do
   print "this is a for loop"
end For
                Or
Get n
For  i← n downto 1 do
   print "this is a for loop"
end For
```

Exercise! Exercise! Exercise!

1) Write an algorithm that reads a number n and returns the first n numbers. n should be a whole number greater than 0.

2) Write an algorithm that reads a number n and returns the sum of the first n numbers.

3) Write an algorithm to calculate the area of a circle.

4) Write an algorithm to solve a linear equation

5) Write an algorithm that reads a person's name and sex, and returns the message "good morning Mr. name" if the person is a man and "Good morning Mrs. Name" for a woman.

3.5. Recursion

Some problems are recursive in nature. This means that the solution to such problems involves the repeated application of the solution to its own values until a certain condition is reached. Algorithms for such problems are known as recursive algorithms.
A recursive algorithm is an algorithm that calls (invokes) itself during its execution. Examples are the factorial function and the sum function.
Recursion can be defined as the calling of a procedure by itself, creating a new copy of the procedure.

a. Factorial Function

Factorial is defined as:

$1! = 1$
$2! = 2 \times 1 = 2$
$3! = 3 \times 2 \times 1 = 6$
$4! = 4 \times 3 \times 2 \times 1 = 24$
$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$
…
$n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$

By studying the above equations closely, we see that the factorial of any number n can be calculated by multiplying the number by the factorial of the preceding number.
We therefore have:

$1! = 1$
$2! = 2 \times 1!$
$3! = 3 \times 2!$
$4! = 4 \times 3!$
$5! = 5 \times 4!$
…
$n! = n \times (n-1)!$

Factorial is defined recursively as follows:

$$fact\,(n)\begin{cases}1 & if\ n=1 \quad \text{base case}\\ n\times fact(n-1) & if\ n>1\end{cases}$$

```
Get n
If n = 0 or n = 1 then
        fact ← 1
else
    fact ← n × fact(n − 1)
return fact
```

Remark 1! = 1 is known as the base case. Every recursive problem must always have some base case which can be solved without recursion. For cases that are to be solved recursively, the recursive call must always make progress towards the base case.

b. The Sum Function

The sum function is a function that calculates the sum of the first *n* integers. For example we want to calculate the sum of the first 5 integers 1, 2, 3, 4 and 5. Their sum is calculated as follows:

Sum(5) = 1+2+3+4+5

We can see that for any number *n*, the sum *sum(n)*, is the number *n* plus the sum of the previous numbers.
The sum function can therefore be defined recursively as:

$sum(1) = 1$
$sum(2) = 2 + sum(1)$
$sum(3) = 3 + sum(2)$
$sum(4) = 4 + sum(3)$
…
$sum(n) = n + sum(n-1)$

The base case is $n = 1$ which gives $sum(1) = 1$

```
get n
sum ← 0
if n = 1 then
        sum ← 1
else
    sum ← n + sum(n − 1)

return sum
```

c. The Fibonacci Series

The Fibonacci series is a series in which each number is the sum of the two previous numbers in the series. E.g. 0, 1, 1, 2, 3, 5, 8, 13, 21…

The Fibonacci series is defined as follows:

$fib(0) = 0$
$fib(1) = 1$
$fib(n) = fib(n-1) + fib(n-2)$

$fib(0)$ and $fib(1)$ are the base cases.

```
Get n
fib ← 0
if n = 0 then
        fib ← 0
else
  if n = 1 then
        fib = 1
  else
     fib ← fib(n − 1) + fib(n − 2)
return fib
```

Exercise! Exercise! Exercise!
Write a recursive algorithm to find the greatest common divisor (gcd) of two numbers.

3.6. Sort Algorithms

Sorting is a programming technique which is used to arrange a list of pre-stored data in an ascending or descending order according to a preset criterion. There are lots of useful sorting methods. For example: insertion sort, bubble sort, selection sort, quick sort, merge sort and heap sort algorithms.

a. Bubble Sort

Bubble sort algorithm arranges items in order as follows:

- First, examine the first two items in the list. If they are in order, leave them alone; if not, interchange them.
- Do the same with the second and third items, then with the third and fourth items, until you have reached the last two. At this point you are guaranteed that the item that should come last in the list has indeed "bubbled" up to that position.
- Now repeat the whole process for the first (*n-1)* items in the list, then for *(n-2)* and so on until the list is sorted.

```
begin
   for i = size downto 1 do
      for j ← 2 to i do
         if (List[j-1] >List[j]) then
            temp← List[j-1];
            List[j-1] ← List[j];
            List[j] ← temp;
         end_if;
      end_for;
   end_for;
end.
```

Example: Sort the list [4, 3, 2, 1]

List = [4, 3, 2, 1] size = 4

i = 4 (outer loop, 1st pass)
j = 2 → [4, 3, 2, 1] becomes [3, 4, 2, 1]
j = 3 → [3, 4, 2, 1] becomes [3, 2, 4, 1]

j = 4 → [3, 2, 4, 1] becomes [3, 2, 1, 4]

i = 3 (outer loop. 2nd pass)
j = 2 → [3, 2, 1, 4] becomes [2, 3, 1, 4]
j = 3 → [2, 3, 1, 4] becomes [2, 1, 3, 4]
i = 2 (outer loop, 3rd pass)
j = 2 → [2, 1, 4, 3] becomes [1, 2, 3, 4]

i = 1
Internal loop does not execute because j is greater than i.

b. Selection Sort

Selection sort arranges the elements of a list by first selecting the lowest-valued item, then the next lowest, and so on. In practice, the lowest-valued item is interchanged with the first item in the part of the list being searched, and the search is confined to the remainder of the list from then on.

```
begin
   for i ←1to size-1 do
      iMin←i
      for j ←i+1 to sizedo
         if List(j) < List(iMin) then
            iMin← j;
      end_for
      if m <> i then
         temp← List(iMin)
         List(iMin)← List(i)
         List(i) ←temp
      end_if
   end_for
end.
```

Example: Sort the list [4, 3, 2, 1]

List = [4, 3, 2, 1] size = 4

i = 1 iMin = 1
j = 2 → iMin = 2
j = 3 → iMin = 3
j = 4 → iMin = 4

iMin<> i, so swap item at position iMin (4) and item at position i (1).
[4, 3, 2, 1] becomes [1, 3, 2, 4]

i = 2 iMin = 2
j = 3 → iMin = 3
j = 4 → iMin = 3

iMin<> i, so swap item at position iMin (3) and item at position i (2).
[1, 3, 2, 4] becomes [1, 2, 3, 4]

i = 3 iMin = 3
j = 4 →iMin = 3

iMin = i, so no swap is done

c. Insertion Sort

```
begin
   for i ← 2 to size do
      index← List[i]
      j← i
      while ((j > 1) and (List[j-1] > index)) do
         List[j] ←List[j-1]
         j←j - 1
      end_while
      List[j] ← index
   end_for
end.
```

For example:
[4, 3, 2, 1] size = 4
i = 2 index = 3
j = 2 → [4, 3, 2, 1] becomes [_, 4, 2, 1]
j = 1 → no execution of internal loop
insert index in the place vacated → [3, 4, 2, 1]

i = 3 index = 2
j = 3 → [3, 4, 2, 1] becomes [3, _, 4, 1]
j = 2 → [3, 4, 4, 1] becomes [_, 3, 4, 1]
j = 1 → no execution
insert index in the place vacated → [2, 3, 4, 1]

i = 4 index = 1
j = 4 → [2, 3, 4, 1] becomes [2, 3, _, 4]
j = 3 → [2, 3, 4, 4] becomes [2, _, 3, 4]
j = 2 → [2, 3, 3, 4] becomes [_, 2, 3, 4]
j = 1 no execution of internal loop
insert index in the place vacated → [1, 2, 3, 4]

3.7. Search Algorithms

A search algorithm is a method of locating a specific item in a larger collection of data. They can be used to search for items within an array or list. Common search algorithms are sequential search and binary search.

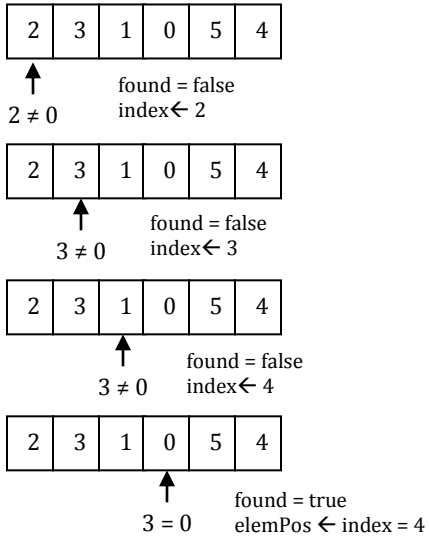
a. Sequential Search

Sequential search is a simple technique for searching an item in a list by comparing each element with the element searched for, beginning with the first element until the element is found.

```
   begin
      index ← 1
      found ← false
```

```
    while ((not found)  and (index <sizeof(List))) do
      if List[index] = elemthen
         found← true
         elemPos← index
      end-if
      index←index + 1
    end_while
    returnelemPos
end.
```

For example: search for the 0 in the list [2, 3, 1, 0, 5, 4]



b. Binary Search

Binary search is a more specialized algorithm than sequential search as it takes advantage of data that has been sorted. The underlying idea of binary search is to divide the sorted data into two halves and to examine the data at the point of the split. Since the data is sorted, we can easily ignore one half or the other depending on where the data we're looking for lies in comparison to the data at the split. This makes for a much more efficient search than linear search.

4. Programming

Programming is the activity of writing computer programs. A computer program is a set of instructions that will be followed by a computer to perform a computation. These instructions are made up of statements written in some languages specially designed for this purpose. These languages are called programming languages.

In other words, a program is an algorithm expressed in a programming language.

4.1. Programming Languages

A programming language is a set of predefined words, symbols and rules that are used to write computer programs. Programming languages can be grouped into two: low-level languages and high-level languages.

4.1.1. Low-Level Languages

A low-level language is a language whose instruction set reflects the processor architecture. An instruction set is the set of bit patterns or binary codes for the machine operations that a processor has been designed to perform. Low-level languages include machine language and assembly language.

a. Machine Language

Machine language is the computer's language. It is the language the computer understands. Machine language instructions are written in binary (a series of 0s and 1s), and are directly executable by the computer. Each machine language statement corresponds to one machine action. Machine language is the first generation of programming languages. For example a short (3 instruction) program might look like this:

```
0111 0001:    0000 1111
1001 1011:    0001 1010
1110 0001:    0011 1110
```

b. Assembly Language

Assembly language is a low-level language consisting of mnemonic codes and symbolic addresses corresponding to machine language instructions. Assembly language is the second generation of programming languages. For example:

```
LOAD  R0  Number1          Load number1 in register 0
LOAD  R1  Number2          Load Number2 in register 1
ADD   R2  R0   R1    Add register 0 and register 1 and keep result in register 2
```

✓ **Advantages of assembly language**

- It is easier to write and understand when compared to machine language.
- It can produce small program sizes
- It can produce very fast code as it allows low-level access to hardware features

✓ **Disadvantages of assembly language**

- Programs are not as easy to write and understand when compared to high level languages.
- Programs are tied to specific computer hardware and can't be reused on another kind of computer.
- Writing programs is very time consuming, tedious, and error-prone.

4.1.2. High-Level Languages

High-level languages are closer to human language. They allow programmers to write programs without having to understand the inner workings of the computer. One high-level language statement will generally be translated into several low-level language statements. They are the third generation of programming languages. Examples are C, BASIC (**B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode), Pascal, Java, FORTRAN (**For**mula **Tran**slator) and COBOL (**Co**mmon **B**usiness-**O**riented **L**anguage). Below is a small code for adding two numbers in Pascal and C.

Pascal

```
program addition;
uses wincrt;
var number1, number2, sum: integer;
begin
   read(Number1);
   read(Number2);
   sum := Number1 + Number2;
   write(sum);
end.
```

C

```
#include <stdio.h>
int main()
{   int number1, number2, sum;
    scanf("%d", &number1);
    scanf("%d", &number2);
    sum = number1 + number2;
    printf("%d", sum);
    return 0;
}
```

- ✓ **Advantages of high-level languages**
  - o easy to understand and write programs as they are user oriented
  - o they have built in libraries to perform routine tasks
  - o programs can be ported to multiple hardware setups from same code

- ✓ **Disadvantages of high-level languages**
  - o programs may be slower than second generation language programs
  - o may produce larger program files for same functionality as second generation languages.
  - o may not allow for low level hardware access

4.2. Language Translators

To run a program on a computer, the program needs to be translated into the machine language of the computer on which it will run. A language translator is a computer program that translates program instructions from one programming language to another without loss of original meaning. There are three types of language translators: compiler, interpreter and assembler.

4.2.1. Assembler

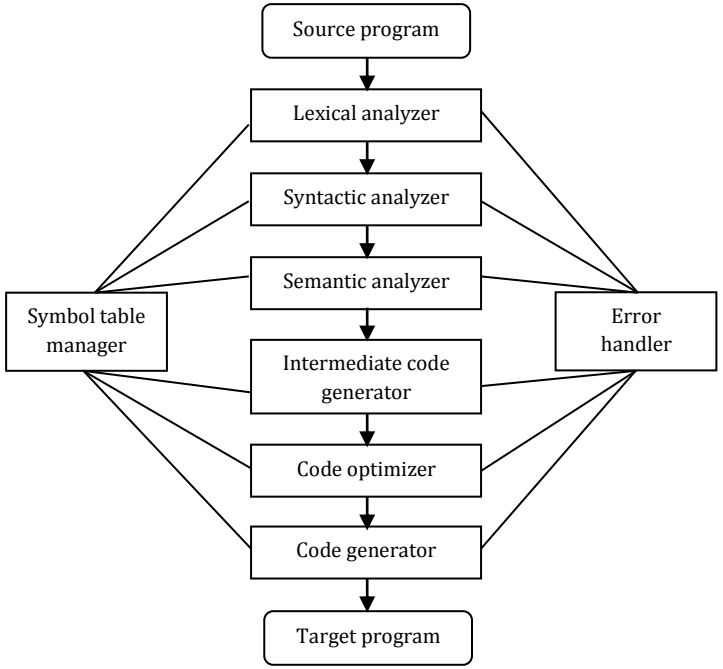
An assembler translates assembly language into machine language. The process is called assembling.

4.2.2. Compiler

A compiler translates the entire high-level program into a machine language program. The high-level language program is called source program or source code and the generated machine language program is called object program or object code. This process is called compilation.

***Definition**: Compilation is the process of translating a program in one language (the source language) into an equivalent program in another language (the object or target language).*

The process of compilation is split up into six phases: lexical analysis, syntactic analysis, semantic analysis, intermediate code generation, code optimizing and code generation.



a. Lexical Analysis

During lexical analysis, a lexical analyzer (scanner) reads input characters from source code and produces a sequence of tokens as output. A token is a symbolic name for an entity that makes up the text of the program.

When the lexical analyzer reads the input characters, it groups them into lexemes which are then replaced by tokens. A lexeme is a group of characters that matches the pattern for a token. In other words, it is an instance of a token. A pattern is a rule that specifies when a sequence of characters constitutes a token. It is simply a rule that describes how a token can be formed.

For example: sum := x + 2;

Lexeme	Token
sum	ident(sum)
:=	assign_op
x	ident(x)
+	bin_op
2	integer
;	semicolon

Lexical analysis performs the following:

- Divides the character stream into meaningful sequences called lexemes.
- Labels each lexeme with a token that is passed to the parser (syntax analysis)
- Updates the symbol tables with all identifiers (and numbers)
- Removes non-significant blanks and comments

b. Syntactic analysis

Syntactic analysis determines how the tokens retuned by the lexical analyser should be grouped and structured according to the syntax rules of the language.

converts a token stream, from lexical analysis, to an abstract syntax tree. The syntactic analyzer (parser)

c. Semantic analysis

d. Intermediate code generation

e. Code optimizing

f. Code Generation

✓ **Advantages of a Compiler**

- Fast in execution
- The object code produced by a compiler can be distributed or executed without having to have the compiler present.
- The object program can be used whenever required without the need of recompilation.

✓ **Disadvantages of a Compiler**

- Debugging a program is much harder. Therefore not so good at finding errors
- When an error is found, the whole program has to be re-compiled

4.2.3. Interpreter

An interpreter is a computer program that translates and executes instructions written in a high-level language into machine language instructions one line at a time. The interpreter translates an instruction and allows it to be executed before translating the next line. If a program performs a section code 1000 times, then the section is translated into machine code 1000 times since each line is interpreted and then executed.

✓ **Advantages of an Interpreter**

- It is good at locating errors in programs
- Debugging is easier since the interpreter stops when it encounters an error.
- If an error is corrected, there is no need to retranslate the whole program

✓ **Disadvantages of an Interpreter**

- It is slow as interpretation and execution is done line by line.
- Translation has to be done every time the program is to be executed since no object code is produced.
- For the program to run, the interpreter must be present

4.3. Program Errors and Correction

4.3.1. Syntax Errors

Syntax is the set of rules that specify how the symbols of a language can be put together to form meaningful statements. In other words, syntax defines the structure of legal statements in a language. A syntax error is an error in a program that occurs due to the non-respect of the syntax rules of the language used. A syntax error will cause a compiler/interpreter to stop trying to generate machine code and will not create an executable. However, a compiler will usually not stop at the first error it encounters but will attempt to continue checking the syntax of a program right to the last line. For example, a misspelled key word, a missing punctuation mark or the incorrect use of an operator is a syntax error.

4.3.2. Semantic Errors

Semantics specify the meaning of a well-formed program. A semantic error occurs when you write a program that works, but does not do what you intend it to do. Compilation and interpretation do not detect semantic errors. Semantic or logic errors are detected from wrong results. Something may be syntactically correct but semantically incorrect.

4.3.3. Run-time Errors

A run-time error is an error that occurs during program execution. For example, a run-time error may occur if division by 0 is attempted. A run-time error may cause the program to stop execution, or it may be handled by an error-trapping routine.

4.3.4. Debugging

An error in a computer program is known as a bug. Debugging is the process of detecting and removing bugs. Syntax errors and semantic errors are bugs. A debugger is the software tool used for this purpose.

4.4. Programming Paradigms

A programming paradigm (or technique) is a fundamental style of computer programming. It describes a programming language's approach to solving a problem. Paradigms differ in the concepts and abstractions used to represent the elements of a program and the steps that compose a computation. High level languages can be classified under four different paradigms: procedural, functional, object-oriented and declarative paradigms.

4.4.1. Procedural Paradigm

In the procedural/imperative paradigm, a program is a collection of statements and procedures that affect data. Here, a program can be seen as an active agent that manipulates passive objects (variables). These objects are passive because they cannot initiate an action by themselves, but can only receive actions from active agents. The focus in procedural programming is to write good functions and procedures.
Examples of imperative languages are Pascal, C, Ada, FORTRAN, and COBOL.

4.4.2. Object Oriented Paradigm

The object-oriented paradigm presents a program as a collection of classes for interacting objects. Unlike in imperative programming, object-oriented programming deals with active objects instead of passive objects. These objects are active because the actions to be performed on the objects are included in them (the objects). The objects need only to receive the appropriate stimulus from outside to perform one of the actions.
Examples of object-oriented languages are C++, Java, Visual Basic and Smalltalk.
Some important concepts related to object oriented programming (OOP) are: class, object, abstraction, encapsulation, inheritance and polymorphism.

a. Class

A class is a description of an object or a real life concept. Classes are templates for creating objects, providing initial values for instance variables (attributes) and the bodies for methods. All objects generated from the same class share the same methods, but contain separate copies of the instance variables. New objects can be created from a class by applying the *new* operator to the name of the class.

```
Class Person {
    private:
        char name[20];
        char sex;
        date birthdate;
public:
        updateInfo();
        returnAge();
  };
```

The name of the class is person. It has four instance variables (attributes) *name, sex* and *age,* and three methods: *updateInfo(),*and *returnAge()*. These methods act on the instance variables when invoked.

b. Object

An object is an instance of a class. An object consists of a collection of *attributes*, representing the state of the object, and a collection of *methods*, representing the behavior that the object is capable of performing. Attributes are sometimes referred to as the fields of an object. The methods are routines that are capable of accessing and manipulating the values of the attributes of the object. Objects interact with each other by sending messages. When a message is sent to an object, the corresponding method of the object is executed.

c. Abstraction

Abstraction is the process of representing important details and hiding away the implementation details. Through abstraction unnecessary details are removed so that one can focus on only the relevant parts of a problem in order to solve it. Abstraction is obtained by separating the interface and the implementation. For example, functions use the concept of abstraction.

The following are the advantages of abstraction.

- The programmer does not have to write the low-level code.
- The programmer does not have to specify all the register/binary-level steps or care about the hardware or instruction set details.
- Code duplication is avoided and thus programmer does not have to repeat fairly common tasks every time a similar operation is to be performed.
- It allows internal implementation details to be changed without affecting the users of the abstraction.

d. Encapsulation

Encapsulation is the process of combining together the attributes and methods of a class into a single abstract data type with a public interface and a private implementation. The goal of encapsulation is to protect the implementation from the users of the object. It ensures that all access to the internal representation of the object pass through the class methods, which act as an "interface" to the object. This is done by making properties and methods "private."

e. Inheritance

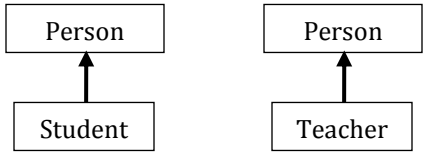
Inheritance is the derivation of one class from another so that the attributes and methods of the derived class are part of the definition of the initial class. The initial class is often called the base class, parent class or super class while the derived class is often referred to as the child class or sub-class. The subclass usually contains all the attributes and methods of the superclass plus some of its own.

For example: the class Student inheriting from the class person will have all the attributes and methods of Person plus the following:

```
Class Student extends Person {
   private
      int   admNumber;
      int   level;
   public
      changeLevel(int newLevel) {
            level=newLevel;
       }
}
```

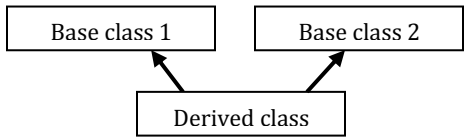
Inheritance is usually represented using an inheritance diagram.

For example: if the classes Student and Teacher are derived from the superclass Person, this is represented as:

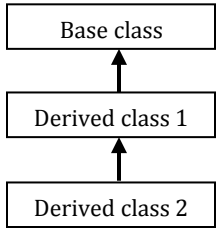


There are two types of Inheritance: multiple inheritance and multilevel inheritance.

- ✓ Multiple inheritance is when a derived class inherits features from more than one superclass. It is illustrated as follows:



- ✓ Multi-level inheritance is when a class inherits from a class which is itself inherited from another class. It is illustrated as follows:



f. Polymorphism

Polymorphism is the use of different methods, each with the same name, which are associated with different object types. In other words, it is the ability for different objects to respond to the same message in different, class-specific ways. Polymorphic methods are used which have one name but different implementations for different classes.
Assume you have a "shape" super class. This class has a method called "area" which returns the area of the shape. Polymorphism allows you to make subclasses like "circle," "square," and "triangle" which inherit the "area" method, but each subclass would return the correct value even though they have different formulas to calculate their areas.

4.4.3. Functional Paradigm

In functional (applicative) programming, a program is a collection of function definitions. Lambda calculus forms the basis of almost all functional programming languages. Lambda calculus is the use of lambda expressions to define functions. A lambda expression is a formula that defines a function.
For example: $f(x) = x + 2$
Examples of functional languages are Haskell, LISP, ML and Scheme.

Sum function in Haskell

$$add :: (Int, Int) \rightarrow Int$$
$$add\ (x, y) = x + y$$

Quick sort in Haskell

$f :: [a] \rightarrow [a]$
$f[\ ] = [\ ]$
$f(x:xs) = fys\ ++ [x]\ ++ fzs$
Where
$ys = [a \mid a \leftarrow xs, a \leq x]$
$zs = [b \mid b \leftarrow xs, b > x]$

4.4.4. Declarative Paradigm

In declarative (logic) programming, a program is a collection of facts and rules involving relational expressions. Declarative paradigm uses the principle of logical reasoning to answer queries. It is based on formal logic defined by Greek mathematicians and later developed into first-order predicate calculus. The point of logic programming is to bring the style of mathematical logic to computer programming. An example of a declarative programming language is PROLOG (programming logic).

These facts are given:
human (John)
mortal (human)

The user asks the following question:
?-mortal (John)

The program answers yes.

CHAPTER 7: COMPUTATIONAL COMPLEXITY THEORY

Introduction

There may be several different methods for solving a particular problem. How can one decide which method is the best in a certain situation? How would one define "best" – is it the fastest method or the one that takes up the least amount of memory space?

Understanding the relative efficiencies of algorithms designed to do the same task is very important in every area of computing.

1. Computational Complexity

Complexity theory is the mathematical study of the time and amount of memory needed to perform a computation. The goal of computational complexity is to classify algorithms according to their performances.

The efficiency of an algorithm can be measured in terms of:

- ✓ The execution time (time complexity)
- ✓ The amount of memory required (space complexity).

1.1. Time Complexity

Time complexity is a measure of the amount of time required to execute an algorithm. It is expressed as the number of times the algorithm's basic operations are executed on inputs of size n. Time in seconds is not used because there are too many factors that can vary e.g. computer speed.

As an illustration, consider a pseudo code algorithm that calculates the mean (average) of a set of *n* numbers:

```
1. n = read input from user
2. sum = 0
3. i = 0
4. while(i<n)
5.   number = read input from user
6.   sum = sum + number
7.   i = i + 1
8. mean = sum / n
```

Statements 1, 2, and 3 are each executed once. Statements 5, 6, and 7 are each executed n times. Statement 4 (which controls the loop) is executed n + 1 times (one additional check is required), and statement 8 is executed once. The running time T(n) of this algorithm is given as follows:

Statement	Number of times executed
1	1
2	1
3	1
4	n+1
5	n
6	n
7	n
8	1

$$T(n) = 1 + 1 + 1 + n + 1 + n + n + n + 1 = 4n + 5$$

1.1.1. Big-O Notation

Big-O notation also called Landau's symbol, is a symbolism used to describe the asymptotic behavior of functions. Basically, it assesses by what factor execution time increases when the size of the input is increased. The letter O is used because the rate of growth of a function is also called its order of growth.

For example, let's suppose that the execution time for an algorithm of input size n is given by:

$$T(n) = 4n^2 - 2n + 2$$

Ignoring constants as they depend on the particular hardware the program is running on, and slower growing terms, we could say "T(n) grows at the order of n^2" and write:

$$T(n) = O(n^2)$$

This simply means that if the input size (n) doubles, the algorithm takes four times longer to execute and the order of growth is said to be quadratic.

Some programs perform the same number of operations every time they are executed while others may perform different numbers of operations, depending on the value of a parameter.

Suppose a computer is going to process n items of input. The time complexity of the calculation may be any of the following:

a. *Constant Time or O(1)*

An algorithm is said to run in constant time if the computation takes the same number of steps regardless of how many items are to be processed. For example storing and retrieving array elements.

b. *Linear Time or O(n)*

An algorithm is said to run in linear time if its time of execution (number of steps) is directly proportional to the input size n. That is, time grows linearly as input size increases. An example would be finding the sum of n numbers, or any program that contains only one loop.

c. Quadratic Time or O(n^2)

An algorithm is said to run in quadratic time if its execution time is proportional to n squared. For example, if the computation involves comparing each item of input with all of the other items, it will take n^2 steps. Many sorting algorithms are $O(n^2)$.

d. Polynomial Time or O(n^k)

An algorithm is said to run in polynomial time if its execution time is proportional to n raised to some constant power. A program that contains k nested loops, each with a number of steps proportional to n, will take time proportional to n^k.

e. *Exponential Time* or O(k^n)

An algorithm is said to run in exponential time if its execution time is proportional to some constant k raised to the nth power. Exponential-time computations are generally not practical.

f. Logarithmic Time or O(log n)

An algorithm is said to run in logarithmic time if its execution time is proportional to the logarithm of the input size. An example is binary search.

Remark! Big-O expressions do not have constants or low-order terms. This is because, when n gets large enough, constants and low-order terms don't matter.
A constant-time algorithm will be faster than a linear-time algorithm, which will be faster than a quadratic-time algorithm.

Formally, the time complexity of an algorithm is of the order f(n)

1.1.2. Best Case, Worst Case and Average Case

Some algorithms perform differently on various inputs of similar size. It is sometimes helpful to consider the Worst-Case, Best-Case, and Average-Case efficiencies of algorithms. For example, say we want to search an array A of size n for a given value K.

- ✓ **Best Case:** K is the first item that we check, so only one comparison.
- ✓ **Worst Case:** K is not in the array, then we must search every item (n comparisons)
- ✓ **Average Case:**

Exercise! Exercise! Exercise!

1. The big-O notation is often used to describe the efficiency of an algorithm.

(a) Place the following algorithms in order of efficiency, the most efficient first.

Algorithm A that is O(n)
Algorithm B that is $O(a^n)$
Algorithm C that is $O(n^2)$ (1 mark)

(b) Describe a linear search and explain why it is O(n).
(4 marks)

(c) Describe a bubble sort and explain why it is $O(n^2)$.
(4 marks)

1.2. Space Complexity

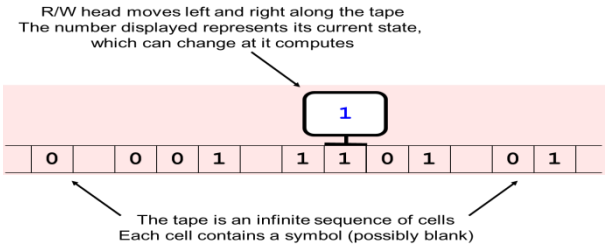
Space complexity is essentially the amount of memory (number of cells) which a program needs to execute. A good algorithm keeps this number as small as possible.

2. Computability

Computability deals with whether a problem can be solved at all, regardless of the resources required.

2.1. Turing Machines

A Turing machine (TM) is an imaginary machine conceived by Alan Turing in the 1930s to help identify the kinds of problems that are potentially solvable by machines. It consists of a read/write head and an infinite string of paper tape made up of cells where each cell can hold a symbol from the tape alphabet.
The head scans at a cell on the tape and can read, erase, and write a symbol on the cell. In each move, the head can move to the right cell or to the left cell or stay in the same cell.



Formally, a TM is defined as:
TM = <S, T, s_0, d, H> where,
S is a set of TM states
T is a set of tape symbols
s_0 is the start state
$H \subset S$ is a set of halting states
$d : S \times T \longrightarrow S \times T \times \{L,R\}$ is the transition function

A TM begins computation at state s_0. At each computational step, it reads the present tape symbol, changes its internal state, optionally writes another symbol onto tape, and moves one cell to the left or right in the tape.
The TM halts computation (and accepts the input string) when it reaches one of the halt states in H.

2.2. The Church-Turing Thesis

3. Computable Functions

A function is computable if it can be computed by a Turing machine.

4. Decidability

5. The Halting Problem