

**COURSE TITLE: INTRODUCTION TO COMPUTER SCIENCE**  
**COURSE CODE: CPT111**  
**CREDIT UNIT: 2**

## **Table of Content**

<b>MODULE 1: INTRODUCTION TO COMPUTING.....</b>	<b>3</b>
UNIT 1: Basic Concepts.....	4-11
UNIT 2: Computer Hardware.....	12-19
<b>MODULE 2: COMPUTER SOFTWARE.....</b>	<b>20</b>
UNIT 1 Introduction to Computer Software.....	21-27
UNIT 2 Operating systems.....	28-36
<b>MODULE 3: MACHINE LEVEL REPRESENTATION OF DATA.....</b>	<b>37</b>
UNIT 1 Units of Data and Fundamental Operations on Bits.....	38-48
UNIT 2 Number Bases and Types.....	49-60
UNIT 3 Representations of Non Numeric Data, Records and Registers.....	61-67
<b>MODULE 4: LOGIC.....</b>	<b>68</b>
UNIT 1 Digital Logic.....	69-76
UNIT 2 Memory.....	77-84
UNIT 3 Von Neumann Model of Computation.....	85-91
<b>MODULE 5: PROGRAMMING THE COMPUTER.....</b>	<b>92</b>
UNIT 1 Overview of Programming Languages.....	93-106
UNIT 2 Algorithms.....	107-114
UNIT 3 Flowcharts.....	115-124
<b>MODULE 6: THE INTERNET.....</b>	<b>125</b>
UNIT 1 Introduction to the Internet.....	126-134
UNIT 2 Web Technologies.....	135-152

# **Module1**

## **Introduction to Computing**

Unit 1: Basic Concepts

Unit 2: Computer Hardware

## Unit 1: Basic Concepts

### 1.0 Introduction

In the 21<sup>st</sup> century, computers have become indispensable tools in virtually all aspects of human lives. The development of handheld computers such as personal digital assistants (PDAs), tablet computers and smart phones has helped computers to become ubiquitous. This unit presents background information about computers by defining basic computer terms, providing a brief history of modern computing, and describing the parts of a computer system.

### 1.1 Definition of Terms

**Computer:** A computer is an electronic device that processes data, in order to convert it to information that is useful to people. A computer can perform the following tasks:

- i. Accept data through an input device (e.g. keyboard or mouse)
- ii. Process the data to convert it into information
- iii. Display the information on an output device (e.g. visual display unit or printer)
- iv. Store the information for future use in a storage device (e.g. hard disc or compact disc)

**Data:** Data refer to raw or unprocessed facts about a person, place or thing. Examples of data include name, age, height and profession. Data is the plural for datum.

**Information:** Information is processed data or data that has been converted into useful form e.g. the result of students in an examination or the net pay of an employee.

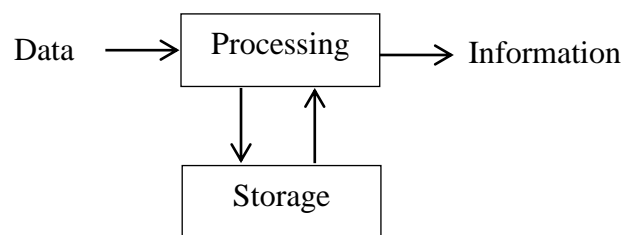


Figure 1.0 Schematic diagram of a compute

## 1.2 Computer Generations

Computer scientists and historians use the term *computer generations* to describe the stage-by-stage development of modern computing. Each generation is characterized by the technology used to fabricate computers at that time.

- **The First Generation (1950s)**

In 1951, Presper Eckert and John Mauchly delivered the Universal Automatic Computer (UNIVAC), the first successful general-purpose computer to the U.S. Census Bureau. The following year, the UNIVAC gained fame when it correctly predicted that Dwight Eisenhower will win the U.S. presidential election. The UNIVAC used punched cards and magnetic tape for input. It was made of vacuum tubes, which required lots of power and failed regularly. The UNIVAC was programmed using machine language, which is composed of strings of zeros and ones.

- **The Second Generation (1960s)**

First-generation computers were highly unreliable because the vacuum tubes burned out frequently. Second generation computers were made of transistors, which are small electronic devices that can control the flow of electricity in an electronic circuit. Owing to the use of transistors, second generation computers were faster, smaller, and more reliable than first-generation computers. Even though second generation computers still used punched cards for input, they had printers, tape storage, and disk storage.

Second generation computers were programmed using high level programming languages, rather than machine language. Unlike machine language which was cumbersome to work with, high-level programming languages are much easier for people to understand and work with because they use English commands and mathematical symbols. Furthermore, unlike machine language which was machine-dependent, a high-level language program can be used on computers produced by different manufacturers. An example of second generation computers is the IBM's line of computers called System/360.

- **The Third Generation (Mid-1960s to Mid-1970s)**

The key distinction of third generation computers was their use of Integrated Circuits (ICs). ICs incorporated many transistors and electronic circuits on a single silicon chip. They were much cheaper than transistors. Using a technology called small-scale integration (SSI), the

earliest ICs could contain 10 to 20 transistors on a chip. By the late 1960s, medium-scale integration (MSI) allowed between 20 and 200 transistors to be placed on a chip. In the early 1970s, large-scale integration (LSI) was achieved, allowing up to 5,000 transistors on a single chip. ICs made it possible to produce smaller, inexpensive computers that more organizations could afford to buy.

Another innovation of third generation computers was timesharing, a technique that allowed several people to simultaneously use a computer from their remote terminals. An example of third generation computers is DEC's PDP-8.

- **The Fourth Generation (1975 to the Present)**

The development of increasingly sophisticated ICs gave rise to very-large-scale integration (VLSI) technology, which allowed the entire processing circuits of a computer to be placed on a single chip. Dr. Ted Hoff built the Intel 4004, the world's first microprocessor, which had 2,300 transistors. A microprocessor is a single chip that holds the entire control unit and arithmetic-logic unit of a computer.

Shortly after the introduction of the 4004 and its successor the 8008, the first microcomputers began to appear. A microcomputer is a computer that uses a microprocessor as its central processing unit. In other words, the microprocessor is responsible for processing data in a microcomputer. Apple Computer, Inc. developed the Apple I microcomputer, followed by the highly successful Apple II. The Apple II was based on the Motorola 6502 microprocessor and it had a keyboard, monitor and floppy disk drive. The development of VisiCalc, the first electronic spreadsheet software in 1979 showed that the Apple II was not just a toy to be used in schools and homes; it , meant that microcomputers could be used for business applications. In 1981, the IBM Personal Computer (PC), was released. The PC used the Intel 8080 microprocessor as well as Microsoft's operating system called MS-DOS.

The first microcomputers were not easy to use because users had to type commands on the command line to perform such actions as formatting a disk or starting a program. However, from the mid-80s, graphical user interfaces (GUIs) were incorporated into microcomputers, allowing users to interact withon-screen, icons, windows and pull-down menus using a pointing device such as a mouse. GUIs were easier to use than command line interfaces because they eliminated the need to memorize commands.

- **1.3 Components of a Computer System**

Every computer is part of a system. The complete computer system consists of four parts, namely: hardware, software, data and user.

### **Hardware**

Hardware are the electrical and mechanical devices that make up a computer. They are the parts of the computer that can be touched and felt, such as keyboard, mouse, monitor, speaker and printer.

### **Software**

Software is the set of instructions that tells the computer what task to do, and how to do it. A piece of software is referred to as a program. Some programs such as operating systems exist to help the computer perform tasks and manage its resources. Other programs such as word processors and spreadsheets allow users to create and manage documents.

### **Data**

Data refers to individual facts that may not make sense on their own. The computer's job is to convert data into useful information

### **Users**

The people who operate computers systems are referred to as users. Even though a computer may function without anyone sitting in front of it and operating it, no computer is completely autonomous. Human beings are still needed to design, build, program and repair computers.

## Unit 2: Computer Hardware

### 1.0 Introduction

Computer hardware refers to the mechanical and electronic parts of a computer that can be touched and seen. Hardware is what the layman often refers to as the computer. This unit discusses the different parts of computer hardware.

### 1.1 Components of Computer Hardware

The following hardware components are required by all general-purpose computers:

- i. Central processing unit (CPU) for executing instructions
- ii. Memory for storing data and programs, at least temporarily (temporarily or permanently)
- iii. Input devices for sending data and instructions into the computer
- iv. Output devices for bringing retrieving information from the computer
- v. Storage devices for retaining large amounts of information permanently.

Figure 2.0: Shows how the different hardware components interact in a general-purpose computer.

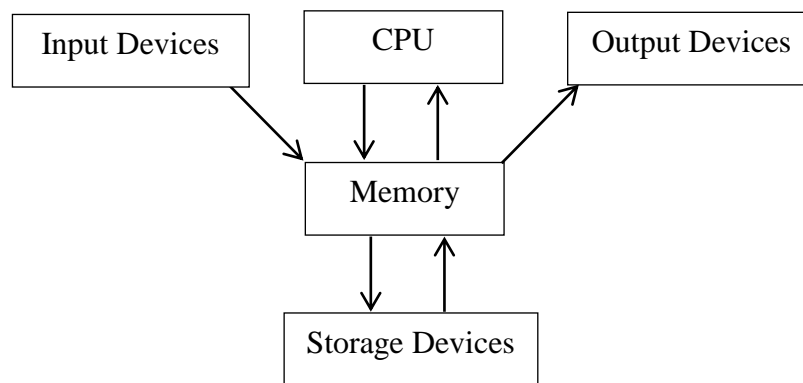


Figure 2.0 Parts of the Computer Hardware

### 1.2 Central Processing Unit

The central processing unit or simply the processor, is the part of the computer that executes program instructions and controls the operation of all other hardware components. It is sometimes described as the computer's brain. In a microcomputer, the CPU is contained in a single chip and referred to as a microprocessor.

The CPU is made up of three parts- the arithmetic-logic unit, control unit and registers.

**Control Unit:** it coordinates other hardware components in order to carry out program instructions. During program execution, it directs electronic signals between memory and arithmetic-logic unit as well as between CPU and input/output devices.

**Arithmetic-Logic Unit:** The arithmetic-logic unit (ALU) performs two types of operations – arithmetic operations and logical operations. Arithmetic operations are fundamental mathematical operations such as addition, subtraction, multiplication and division. Logical operations compare two values to determine whether one is less than, greater than or equal to the other.

**Registers:** registers are high speed storage locations within the CPU that temporarily hold data and instructions during processing. Because they are part of the CPU, register contents can be handled much faster than the contents of memory.

### 1.3 Memory

General-purpose computers have two kinds of memory:

- i. Random Access Memory (RAM) and
- ii. Read Only Memory (ROM).

**Random Access Memory:** RAM is the working memory of the temporary area for holding raw data for processing, instructions for processing the data, and information. It is the working memory of the computer. When a program is started, it is loaded from storage to RAM. The program remains in RAM **until it closed**. When data is entered into the computer through the input device, it is first stored in RAM. During processing, data and instructions are fetched from RAM and stored in CPU registers. At the end of processing, results are stored in RAM before they can be displayed to the user.

There are two reasons why RAM is a temporary storage:

- i. RAM is volatile, meaning that it requires electric power to hold data. When the computer is turned off, everything stored in RAM disappears.
- ii. Data stored in RAM can easily be changed

#### **Read Only Memory**



ROM is a type of memory that holds the built-in instructions that tell the computer what to do when it is turned on. Unlike Ram which is temporary and volatile, ROM is permanent and non-volatile. Instructions stored in ROM cannot be changed, and they are preserved even when the computer is turned off.

#### **1.4 Storage Devices**

Storage devices are non-volatile, long-term memories. Unlike Ram whose contents are lost when power supply goes off, storage devices preserve their contents even when there is no power supply. As a result, storage devices are commonly used to transfer data and programs from one computer to another. They can also be used to back up valuable data, so that the data can be restored after a computer crash results in loss of data. In comparison to ROM whose contents cannot be changed, information held in storage devices can be easily replaced. Examples of storage devices are hard disks, optical discs and flash drives.

##### **Hard Disks**

A hard disk is the main storage device in most computers. It is slower, less expensive and has higher capacity than RAM. A hard disk is made up of several platters, which are coated with magnetic material. Each platter requires read/write heads to retrieve or modify the information stored in the platter.

##### **Optical Disks**

Optical disks allow data to be read (i.e., retrieved) or written (i.e., changed) by lasers. The process of creating an optical disk is called burning. Even though optical disks can tolerate temperature fluctuations better than hard disks, they can easily get scratched hence they need to be stored in a case or jacket. The types of optical disks include:

- i. Compact Disk Read Only Memory: Compact Disk Read Only Memory (CD ROM) contains data stored on the disk while it was being manufactured. The data can be read many times but cannot be modified
- ii. Compact Disk Write Once Read Many times: Data can be written into a Compact Disk Write Once Read Many times (CD WORM) once, but after that it behaves like a CD ROM.
- iii. Compact Disk Rewriteable: A Compact Disk Rewriteable (CD RW) allows data to be written and read as many times as desired, just like a hard disk.

**Universal Serial Bus Flash drives:** Universal Serial Bus Flash drives or simply flash drives are amongst the most popular storage devices. There are several reasons for their popularity:

- i. They are portable i.e., easy to carry about
- ii. Virtually all computers have USB ports into which flash drives can be plugged
- iii. Flash drives can be used without bulky devices or burners

## 1.5 Input Devices

An input device is any machine used to send data and instructions into the computer. Examples of input devices are keyboard, mouse, joystick, digital cameras and scanners

**Keyboard:** The keyboard is the most important input device on a computer. Computer keyboards have the same layout as standard typewriters. The layout is called QWERTY, because the first six letters on the top row of letters are Q, W, E, R, T and Y. In addition to typing keys, computer keyboards have arrow keys for navigation, function keys for computer-specific tasks, and a calculator-like numeric keypad.

**Mouse:** A mouse is a device for controlling the movement of a pointer or cursor on the screen. It consists of a ball which is rolled on a surface. The mouse derives its name from its resemblance to a real mouse.

**Joystick:** A joystick resembles a car's gear stick. Moving the stick in any direction results in a corresponding movement of an on-screen object such as a pointer. Joysticks are mostly used for playing computer games.

**Digital cameras:** They are used to capture still or moving images, which can be transferred to a computer and manipulated using image or video processing software.

## 1.6 Output Devices

Output devices retrieve information from the computer and present it to the user. Examples of output devices are monitors, printers, plotters and speakers.

**Monitors:** The monitor is the computer's display screen. It is also known as visual display unit (VGA). The two commonly used technologies for monitors are liquid crystal display and light emitting diodes, which have replaced cathode ray tube monitors that resembled bulky old-fashioned televisions.

**Printers:** A printer is a device that expresses text or illustration on paper and other media. The two most commonly used types of printers today are inkjet and laser printers. Inkjet printers work by spraying ink on papers to produce text and characters. On the other hand, laser printers pass a beam on a cylindrical drum, which collects powdered ink (toner) and transfers the toner to paper. Laser printers work in the same way as photocopiers.

**Plotters:** A plotter is a device that draws pictures by moving one or more pens on paper. Plotters can be used to produce large printouts. They are typically used by engineers, architects and interior designers who require more precision than can be offered by printers.

### **References and Further Readings**

Norton, P. (2005), *Peter Norton's Introduction to Computers* 6th edition, McGraw-Hill/Irwin  
Pfaffenberger, B. (2002), *Computers in your Future* 4th edition, Prentice Hall  
Parsons, J. J., Oja, D. (2011), *Practical Computer Literacy* 3rd edition, Centage Learning.  
Norton, P. (2005), *Peter Norton's Introduction to Computers* 6th edition, McGraw-Hill/Irwin  
Leon, A., Leon, M., (1999), *Fundamentals of Information Technology*, leonVikas Hall

# **Module2**

## **Computer Software**

Unit 1: Introduction to Computer Software

Unit 2: Operating System

## Unit 1: Introduction to Computer Software

### 1.0 Introduction

Unlike hardware which refers to the physical parts of the computer that can be touched, software is the set of intangible instructions that tell the computer what to do. This unit discusses the two major categories of computer software, namely application and system software, as well as the types of software in each category.

### 1.1 Categories of Computer Software

Software is broadly divided into two categories:

**System Software:** These are the software that interact with the computer at a very basic level. They help the computer to carry out its basic operating functions such as managing files, interacting with input/output devices, and removing viruses from the computer. Examples of system software are operating systems, utilities and language translators.

**Application Software:** These are the software that help users to do real work such as creating documents, editing photos or tracking finances. Application software do not interact with the computer directly. Rather, they do so through system software such as operating systems and utilities.

Figure 1.0, provides an overview of application software and system software, as well as the types of software in each category.

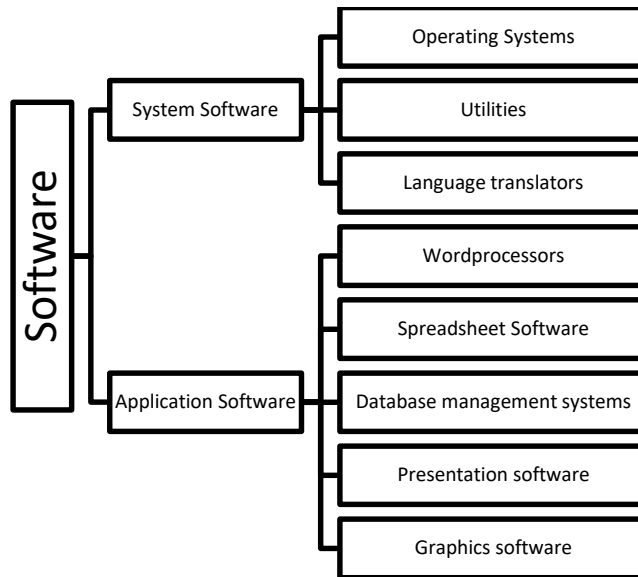


Figure 1.0: Overview of types of software

## 1.2 Operating Systems

Operating systems are the most important programs that run on a computer, because they control all activities that take place in a computer. Operating systems perform basic tasks such as keeping track of files, management of memory allocated to programs and data, and controlling input/output devices such as keyboard and printer.

Examples of operating systems for general-purpose computers are Microsoft Windows, Mac OS, UNIX and Linux. Handheld devices use operating systems such as Windows Mobile OS, iOS and Android OS.

## 1.3 Utilities

A utility is a program designed to perform tasks such as optimizing a computer's performance, protecting data and facilitating communication. Examples of utility programs are:

- i. **Antivirus software** for protecting the computer from computer virus.
- ii. **Backup software** that help store copies of files, which can be restored in case the files get lost or damaged.
- iii. **Compression utilities (or zip software)** that shrink files so that they require less storage space. Compression utilities can also return files to their original form when required.
- iv. **Email software** for managing the flow of data to/from an electronic post office box.

## 1.4 Language Translators

Computers understand only machine language, in which instructions are written as strings of zeros and ones. However, programming in machine language is tedious and error-prone for humans. As a result, programs are usually written in high level languages which contain English commands and mathematical symbols that humans are familiar with.

Language translators are system software that convert high level language programs to machine language programs which the computer understands. Examples of language translators are:

- i. **Compilers** which translate all instructions in a high level language program to machine language before executing any instruction.
- ii. **Interpreters** which translate and execute an instruction in a high level language program before moving to the next instruction.

## 1.5 Word Processors

A word processor is a program for producing documents such as letters, memos, reports and manuscripts. Word-processing software allow you to create, edit, format and spell-check documents on the screen before orienting on paper. Examples of word processors are Microsoft Word and OpenOffice Writer.

## 1.6 Spreadsheets

A spreadsheet is an arrangement of rows and columns containing values that can be manipulated. An electronic spreadsheet is a program for working with spreadsheets. Because electronic spreadsheets allow users to simple or complex formula, they can be used for performing tasks such as budgeting, tracking finances, calculating loan payments, computing student grades and estimating project costs. Electronic spreadsheets allow users to display varieties of colorful graphs. They often support *what-if-analysis*, for analyzing different scenarios such as “What if I score Bs in all my courses this semester? But what if I score only Cs?” examples of electronic spreadsheets are Microsoft Excel and OpenOffice Calc.

## 1.7 Database Management Systems

A database is a collection of data stored on one or more computers. A database can contain data such as details of books in a library, university student records, or bank customer details. A database management system (DBMS) is a program for storing, modifying, finding and

reporting data contained in a database. DBMS allow users to *query the database*, that is, to extract information that meets certain criteria. For example, a query could request the DBMS to list all students who scored at least a B grade in CPT 111, arranged according to their matriculation numbers. Examples of popular DBMS are Microsoft Access, Oracle and My SQL.

### 1.8 Presentation Software

Presentation software enable users to combine text, graphs, photos, sound clips and animation into series of electronic slides. For one-on-one presentations, the slides can viewed on the monitor. However, group presentations are commonly viewed using a computer projector. Presentation slides can also be posted on the Internet. Electronic slides can be used by instructors and students to deliver lectures and oral presentations. They are also useful for presenting ideas in meetings and conferences. Examples of common presentation software are Microsoft PowerPoint and OpenOffice Impress and Google Presentations.

### 1.9 Graphics Software

Graphics software allow users to create, edit and manipulate graphics. These graphics could be pictures, images, drawings, icons or photographs. There are different types of graphics software:

- i. **Paint software** help you paint images by providing pens, brushes and paints. They represent images using bitmap graphics formats such as BMP, PNG, TIF and JPEG. Examples of paint software are Microsoft Paint and Corel Painter.
- ii. **Drawing software** are designed to for combining lines, shapes and colors into diagrams. They represent diagrams using vector graphics files such as WMF and EPS. Examples of drawing software are Adobe Illustrator and Corel DESIGNER.
- iii. **Photo editing software** allow you to enhance poor-quality photos by adjusting brightness and contrast, cropping out unwanted portions, and so on. An popular example of photo editing software is Adobe Photoshop.



## Unit 2: Operating System

### 1.0 Introduction

Operating systems are the most important software that run on a computer. They manage the computer hardware, and act as an intermediary between the computer user and the hardware. This unit describes the history of operating systems, beginning with computers that didn't have any operating systems, and ending with operating systems used in of today's computers. Like most man-made systems, the many features of operating systems were introduced as engineers' experience accumulated over time.

Operating systems vary greatly. For example, personal computer operating systems are designed to support games, business applications and so on. On the other hand, operating systems for mainframe computers are designed to maximize the utilization of hardware.

### 1.1 Historical Evolution of Operating Systems

**Serial Processing:** From the late 1940s to the mid-1950s, the earliest computers had no operating systems, thus the programmer interacted directly with the hardware. Programs written in machine were loaded via a card reader (i.e., the input device). If a program successfully completed execution, the output appeared on a printer. However, if the program halted due to an error, the error condition was indicated by some display lights. The programmer and/or computer operator had to supply all the instructions to carry out even the most basic tasks. There were two main problems with the earliest computers.

- i. Because users had to reserve computer time, the computer remained idle if a user's program completed execution before the allocated time. On the other hand, if a user ran into problems, he/she might be forced to leave the computer once the allocated time was used up, even though the program has not completed execution.
- ii. Considerable time was needed to setup a program to run. A single program could involve loading a high-level program and compiler into memory, saving the compiled program, and so on. If an error occurred in the process, the user had to go back to the beginning of the sequence.

This mode of operation is referred to as serial processing, since users had to access the computer in series.

**Simple Batch Systems:** The first operating systems were batch systems which appeared in the mid-1950s. Batch systems used a simple operating system known as a monitor, which eliminated the need for users to directly access the processor. Instead, a user submits a program or job to the computer operator on tape or disk. The operator queues up the jobs, and later submits an entire batch of jobs to the monitor on an input device. Each program is designed to return to the monitor upon completion, so that the monitor can load the next job. The major bottleneck of batch processing systems were input/output operation: all computation had to stop to let an I/O operation take place.

**Multi-Programmed Batch Systems:** Even though job sequencing is automatically handled in simple batch systems, the processor is often idle. The reason is that I/O devices are slow compared to the processor. A processor may spend up to 90% of its time waiting to read from/write to a file. Multiprogramming or multitasking systems allow the monitor as well as several other programs to be loaded into the computer memory. Processor idle time is minimized because while one program is performing I/O, another program can make use of the processor. This approach is used in many modern operating systems.

**Time-Sharing Systems:** Multiprogramming allows batch systems to be efficient, but it doesn't allow the user to interact with the computer during processing. User interaction is essential in certain jobs such as transaction processing. Timesharing allows the processor time to be shared among multiple users. Each user accesses the computer through a terminal. The operating system allocates a small time slice to each job, and then moves to the next job. The process is repeated until all jobs are completed. Because the time slot is very small, the operating system returns to each job after a short time, causing each user to think that only his/her job is running on the computer.

## 1.2 Types of Operating Systems

There are a wide variety of operating systems in use today. Some of these types of operating systems are:

**Mainframe Operating Systems:** Mainframes are the room-sized computers still found in major corporate data centers. Operating systems for mainframes are geared towards processing multiple I/O-intensive jobs at the same time.

**Server Operating Systems:** Server operating systems run on servers, which could be large personal computers, or even mainframes. They serve multiple users at once over a network

and allow the users to share software and hardware resources. Servers can provide one or more of the following types of services: file service, print service, or Web service.

**Multiprocessor Operating Systems:** Multiprocessor operating systems are needed to help connect multiple CPUs into a single system. They differ from server operating systems, because they have special features for handling communication, connectivity, and consistency.

**Personal Computer Operating Systems:** Personal Computer operating systems provide good support to a single user. They support multiprogramming, and are widely used for word processing, spreadsheets, and Internet access.

**Handheld Computer Operating Systems:** Handheld computers include mobile phones as well as PDA (Personal Digital Assistants), which are used for performing a small number of functions, such as an electronic address book and memo pad. The operating systems that run on these handhelds have the ability to handle telephony, digital photography, emails, Internet access, and other functions.

**Embedded Operating Systems:** Embedded systems run on the computers that control devices that are not usually thought of as computers. Since they do not accept user-installed software. Examples are car engines, MP3 players, TV sets, microwave ovens, and DVD recorders. Embedded systems differ from handheld devices, because entrusted software will not run on embedded systems. For example, you cannot download new applications to your microwave oven, since all the software is already stored in the ROM.

**Sensor Node Operating Systems:** Sensors are small battery-powered computers with built-in radios. These nodes are tiny computers that communicate with each other and with a base station using wireless communication. They have limited power and must work for long periods of time unattended outdoors, frequently in environmentally harsh conditions. Sensor networks are used to detect fires in forests, measure temperature and precipitation for weather forecasting, protect the perimeters of buildings, guard national borders, and so on. The operating system for sensor networks need to be small and simple because the nodes have little RAM and battery lifetime is a major issue.

**Real-Time Operating Systems:** Real-time systems have time as a key parameter. These systems must provide absolute guarantees that a certain action will occur by a certain time

For example, if a car is moving down an assembly line, certain actions must take place at certain instants of time. If a welding robot welds too early or too late, the car will be ruined.

### 1.3 Functions of Operating Systems

The operating system acts as an interface between application software and the hardware. The key functions of an operating system include:

**User Interface:** Virtually all operating systems provide a user interface (UI) for users to interact with the computer. For example, a command line interface allows users to enter text commands, whereas a graphical user interface allows users to interact with on-screen objects and menus using a pointing device.

**Program Execution:** An operating system performs several tasks in order to execute a program: it loads the instructions and data into memory, it allocates the processor to the program, and so on.

**I/O Operations:** A running program may require I/O from a file or I/O device. In order to ensure efficiency and protection, users cannot control I/O directly. Instead, the operating system provides a means to carry out I/O.

**File System Manipulation:** Operating systems provide file systems that allow users and programs to create, list, search for, delete, read and write files and directories.

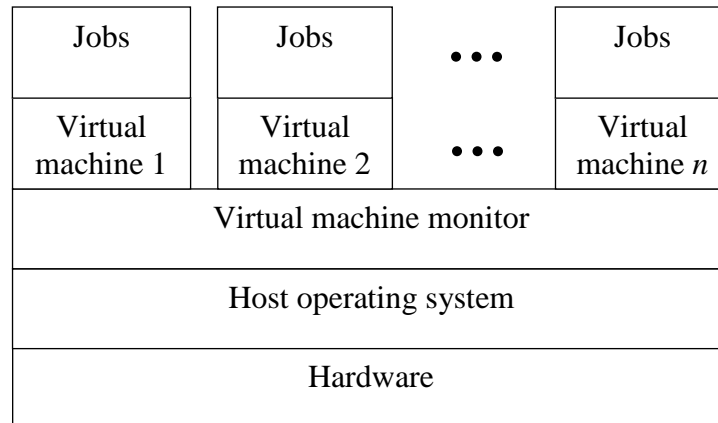
**Resource Allocation:** Operating systems handle the allocation of resources to multiple users and/or multiple jobs running at the same time. Such resources include CPU, main memory and file storage.

**Protection and Security:** Protection is concerned with controlling the access of legitimate users/jobs to the computer resources. On the other hand, security involves safeguarding the computer system from outsiders. An example of protection is when the operating system prevents several jobs running at the same time from interfering with one another or with the operating system. An example of security is when the operating system requires users to authenticate themselves by means of passwords, in order to gain access to system resources.

### 1.4 Virtual Machines

A virtual machine (VM) abstracts the hardware of a single computer (such as CPU, memory and disk drives) into a number of execution environments, making it seem like each

execution environment is running its own private computer. Virtualization allows a single computer to run multiple operating systems, or multiple sessions of an operating system. A host operating system can support several virtual machines. A virtual machine monitor (VMM) runs on top of the host operating system, or is incorporated into it. Each virtual machine runs a separate virtual operating system. In order to execute a job, the VMM hands over processor control to a virtual machine operating system.



Virtual machine concept

### References and Further Readings

- Stallings, W.(2012), Operating systems: Internals and design principles, 7<sup>th</sup>ed, Prentice Hall, New Jersey
- Tanenbaum, A. (2009), Modern Operating Systems, 3<sup>rd</sup>ed, Prentice Hall, New Jersey
- Silberschatz, A.Galvin, P. B., Gagne, G., (2010), Operating Systems Concepts, 8<sup>th</sup>ed, John Wiley & Sons, New Jersey

# **Module3**

## **Machine Level Representation of Data**

Unit 1: Units of Data and Fundamental Operations on Bits

Unit 2: Number Bases and Types

Unit 3: Representation of Non-Numeric Data, Records and  
Registers

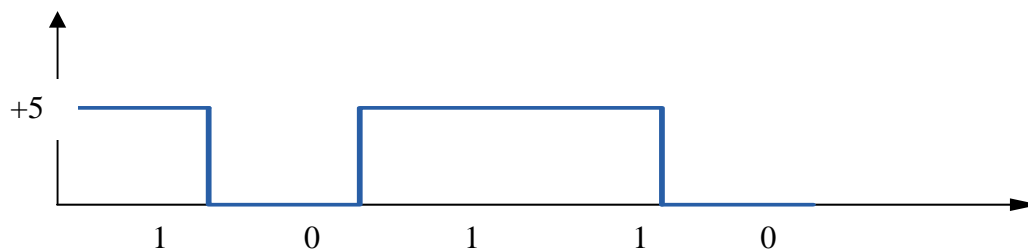
## Unit 1: Units of Data and Fundamental Operations on Bits

### 1.0 Introduction

The main use of a computer is to process data into information. Data refers to information in an unorganized and raw form. For the data to be processed, it has to be fed to the computer and then it is processed in batches. In this class, we shall be looking at the various units of data, starting from the smallest basic unit to the largest. We shall also introduce the fundamental operations that can be performed on the smallest unit of data.

### 1.1 Units of Information

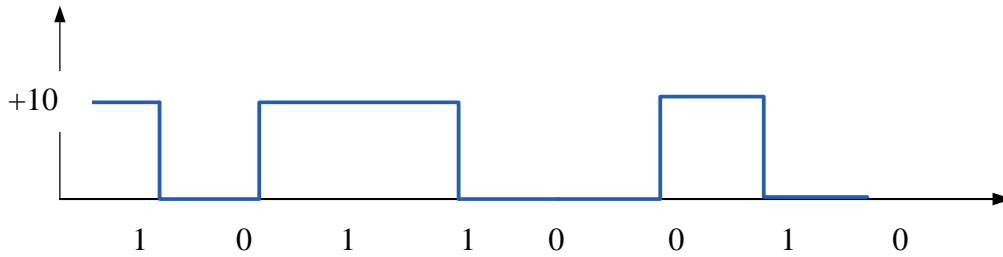
Bits are the shortened form of the term binary digit and it represents information in two states only. The two states are zeros and ones (0 and 1). The 0 represents OFF or LOW states while the 1 represents ON or HIGH states. The two states can be likened to light in a room with two options only, the lights can be ON or OFF with no state in between. Two important factors of a bit are its duration and the difference in voltage levels between the 0 and the 1.



Example 1: Figure showing a string of bits 10110

In example 1 is displayed a string of 5 bits. A +5v was used to represent the ON state (1) and 0v used for the OFF state (0). This is not fixed, and can be +5v for 1 and -5v for 0, +10v for 1 and +5v for 0. The duration of each bit in the example will depend on the duration of the pulse. If the pulse lasted for 10 seconds, the each bit will have a duration of  $10/5 = 2$  seconds.

A group of eight bits put together is referred to as a byte. The amount of information contained increases as the number of bits increases. This is because each of the bits can be either LOW or HIGH and the different sequences of ONs and OFFs can be used to convey information. 8 bits put together can be used to represent the various characters in most languages, and this was then adopted to be 1 byte. Bits is usually represented with a lower case letter 'b', while the upper case later 'B' is used to represent a byte.



Example 2: Figure showing one byte: 1011010

A computer has the capability to process large amount of data at a time. Feeding just 8 bits (1 byte) to a computer to process at a time is a waste of processing power. Just like tying a small bucket to a horse to pull. The number of bits a particular computer can process at a particular time is referred to as word. For example in a 32-bit computer, the word length is 32 bits and such a computer can address 32 memory locations for processing at a particular point of time. Other examples of word lengths are 64bits, 16bits etc.

## 1.2 Measures of Units of Information

Computers can process and store large amount of data and information. As such terms are used to refer to this large aggregate amount of bits or bytes. Table 1.1 shows the various units of information you need to comprehend.

Name	Representation	No of Bits	No of Bytes
1kilobits	1kb	$2^{10} = 1,024$	$1024/8 = 128$
1megabit	1mb	$2^{20} = 1,048,576$	$1048576/8 = 131,072$
1gigabit	1gb	$2^{30} = 1,073,741,824$	$1073741824/8 = 134,217,728$
1terabit	1tb	$2^{40} = 1,099,511,627,776$	$1099511627776/8 = 137,438,953,472$
1Kilobyte	1KB	$8 \times 2^{10} = 8,192$	$8192/8 = 1,024$
1Megabyte	1MB	$8 \times 2^{20} = 8,388,608$	$8388608/8 = 1,048,576$
1Gigabyte	1GB	$8 \times 2^{30} = 8,589,934,592$	$8589934592/8 = 1,073,741,824$
1Terabyte	1TB	$8 \times 2^{40} = 8,796,093,022,208$	$8796093022208/8 = 1,099,511,627,776$

Table 1: The different units of information

It is pertinent that you can comfortably transform one unit to the other.



**Example 3:** Find the number of bytes in 7mb.

To do this you will need to first find the number of bits in 7mb, which is  $7 \times 2^{20} = 7340032$  bits. You then divide by 8 to find the number of bytes as 1 byte = 8 bits. Therefore, number of bits in 7mb =  $7340032/8 = 917504$  bytes.

**Example 4:** A computer network has a download speed of 1MBps. What is the speed in mbps?

Remember that B is for Bytes while b is used to represent bits. The given speed is therefore 1 Megabytes per second, and you are required to find the speed in megabits per seconds. Since 1 bytes is equivalent to 8 bits, 1MB = 8mb. Therefore the computer download speed of 1MBps is equivalent to 8mbps.

### 1.3 Basic Operations on Bits

During processing, a computer manipulates the bits to give a desired result. This manipulation can be a complex process comprising of basic simple operations on the bits. In this section, you will learn how to carry out basic operations of addition, subtraction, division, multiplication of bits and logical operations on bits.

#### Addition of Bits

The addition of bits is similar to the addition of decimal numbers, just that a 2 is treated as a 10, and 1 added to the next bit on the left. The steps you will follow to add to bits together are outlined:

Step 1: Align the bits to be added to the right

Step 2: Start by adding the bits to the far right column

Step 3: Add the bits in each column together, using the rule ( $1+0 = 1$ ,  $0+0 = 0$ ,  $1+1=0$  add 1 to the next column)

Example 5: Add this two bits 10011010 and 11010111

Carry bit: 1 1 11

```

      10010010
    + 11010111
    101101001
  
```

Example 6: Find the sum of these bits 10111001, 110101, 10110  
                                 10111001                                  -1<sup>st</sup> Bits

$$\begin{array}{r}
 + \quad \begin{array}{r} \underline{110101} \\ 11101110 \end{array} \quad \begin{array}{l} -2^{\text{nd}} \text{ Bits} \\ -\text{Sum of } 1^{\text{st}} \text{ and } 2^{\text{nd}} \text{ Bits} \end{array} \\
 + \quad \begin{array}{r} \underline{10110} \\ \underline{\underline{100000100}} \end{array} \quad \begin{array}{l} -3^{\text{rd}} \text{ Bits} \\ -\text{Sum of } 1^{\text{st}}, 2^{\text{nd}} \text{ and } 3^{\text{rd}} \text{ Bits} \end{array}
 \end{array}$$

### Subtraction of Bits

The subtraction of bits is achieved by adding the 2's complement the number to be subtracted to the other bit. The procedure for obtaining the 2's complement is: Turn all the 0 to 1, and the 1 to 0, Add 1 to the last bit on the right. After getting the complement, add to the other binary, but ignore any overflow bit. Overflow bit is the extra bit to the left after the addition.

Example 7: Find  $1100 - 1001$

Step 1: Find the 2's complement of 1001. Flipping all bits gives 0110, adding 1 to the last bit gives 0111.

Step 2: Adding 1100 to the complement 0111.

$$\begin{array}{r}
 + \quad \begin{array}{r} 1100 \\ \underline{0111} \\ \underline{\underline{40011}} \end{array}
 \end{array}$$

The last bit to the left is the overflow bit, which is ignored after the addition. Therefore  $1100 - 1001 = 0011$ .

Example 8: Find  $11001 - 1110$

Step 1: Find the 2's complement of 01110. Note that it was padded with a 0 to make it equal in length to 11001. Flipping all bits gives 10001, adding 1 to the last bit gives 10010.

Step 2: Adding 11001 to the complement 0010.

$$\begin{array}{r}
 + \quad \begin{array}{r} 11001 \\ \underline{10010} \\ \underline{\underline{401011}} \end{array}
 \end{array}$$

The last bit to the left is the overflow bit, which is ignored after the addition.

Therefore  $11001 - 1110 = 1011$ .

It can be observed that the bit after the overflow bit in the two examples are 0. This indicates that the result is a positive number. If it is 1, then the result is a negative number. The following example demonstrates this result of getting a negative result.

Example 9: Find  $10010 - 11001$ .

Step 1: Find the 2's complement of 11001. Flipping all bits gives 00110, adding 1 to the last bit gives 00111.

Step 2: Adding 10010 to the complement 00111.

$$\begin{array}{r} 10010 \\ 00111 \\ \hline 11001 \end{array}$$

### Division of Bits

The division of bits is done by a series of subtractions of the divisor from the dividend, noting the remainder which is added to the other bits of the dividend. To achieve this, the following steps should be followed:

Step 1: Align the non-zero most significant bits of the divisor and the dividend.

Step 2: Subtract the divisor from the most significant bits of the dividend

Step 3: if a borrow was needed, then the quotient/result is 1, and is 0 if no borrow was needed.

Step 4: If a borrow was needed in step 4, add the divisor to the result of the subtraction to restore the dividend and drop the next significant bit. If no borrow was needed in the subtraction, just drop the next most significant bit to the result of the subtraction.

Step 5: Repeat steps 2 – 4 until all the bits of the dividend have been dropped down and the divisor subtracted and the quotient updated. The quotient and remainder after the last bit is dropped is returned as the result of the division.

Example: find  $101011 / 11$

Divisor	Dividend	Quotient
1 1	1 1 0 1 0 1 1	0 1 1 1
Borrow 1	- 1 1	
	1 1	
	+ 1 1	
	1 0 1	
	- 1 1	
	1 0 0	
	- 1 1	
	1 1	
	- 1 1	
	1	Remainder

Therefore  $101011 / 11 = 0111$  with 1 remainder

### Multiplication of Bits

The multiplication of binary number can be achieved by a series of shifting and addition. This can be achieved by following these steps

- Step 1: The numbers to multiplied are aligned to the left
- Step 2: The first bit to the left of the second number is used to multiply through all the bits of the first number. Here  $1 \times 0 = 0$ ,  $0 \times 1 = 0$ ,  $1 \times 1 = 1$ .
- Step 3: You then multiply the second bit of the second number through all the bits of the first, shift the result one bit to the right, and place below the result of step 1.
- Step 4: Keep multiplying through until all the bits of the second number have multiplied the bits of the first number.
- Step 5: Sum together the results of the multiplication of each bits

Example: Multiply  $101101 \times 110101$

		101101
	X	10101
		101101
		000000
		101101
		000000
+		101101
		<u>1110110001</u>

## Unit 2: Number Bases and Types

### 1.0 Introduction

In the previous unit, the bit was introduced to be the shortened form of the binary digit that can exist in two states, 1 and 0. This can be referred to as a base two number. The base of a number refers to the number of numerals or states that the number can contain. The common bases that are used to represent data to the computer system are the base 2 (binary), base 10 (decimal) and base 16 (hexadecimal). This unit will be focusing on the binary, decimal and hexadecimal number bases and the types of numbers.

### 1.1 Binary Numbers (Base two)

As previously explained, a base two number can only exist in two states, 0 and 1. Each of the numeral is referred to as a bit. Series of bits put together to represent data that have more than two states. Table 2.0 shows a string of four bits (known as a nibble, half of a byte) and the decimal equivalent of the various states.

State	Binary	Decimal Equivalent
1	0000	0
2	0001	1
3	0010	2
4	0011	3
5	0100	4
6	0101	5
7	0110	6
8	0111	7
9	1000	8
10	1001	9
11	1010	10
12	1011	11
13	1100	12
14	1101	13
15	1110	14
16	1111	15

A string of 4 bits can be used to represent 16 states of information to the computer system. This is from the fact that  $2^3 = 16$ . The base is raised to power 3 because counting the number of bits from 0 (for the least significant bit) to 3 for the most significant bit. As such each bit in base two have a value that depends on its position in the binary stream.

Power of Base	3	2	1	0	Decimal Equivalent
Binary Number	1	1	1	1	$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 15$

	1	0	1	0	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 10$
	0	1	1	0	$0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 6$

As table 2.1 shows, conversion of binary numbers to decimal numbers involves numbering the bits from 0 for the least significant to most significant. The value of the bit is then used to multiply the base raised to power of the base. The higher the number of bits in the system, the greater the amount of states and then data that can be fed into the computer, processed and stored.

Example: Convert 11001010 to a decimal. How many states of information can a byte contain?

In the solution, a conversion table is filled and worked out

Power of Base	7	6	5	4	3	2	1	0	Decimal Equivalent
Binary Number	1	1	0	0	1	0	1	0	$1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 202$


For the number of states, all the bits are set to 1, that is 11111111 and when converted to decimal gives 255, but 1 is added to the 255 to compensate for the 0 state to give 256 states.

## 1.2 Decimal Numbers (Base 10)

Decimal numbers are the most common base that are used to represent data input to computers and the output of processing are usually in decimals. This base has ten states/numerals ranging from 0 to 9. The position of the numeral also determines its value. The position 0 is referred to as unit, position 1 as tens, 2 as hundreds, 3 as thousands. For example the number 529 has 9 units, 2 tens and 5 hundreds all added together. Since this base is quite common, much explanations will not be necessary, but the conversion of base 10 to base 2 will be concentrated upon.

The conversion of a base 10 to base 2 involves series of division of the base 10 number by 2, recording the remainders from bottom to top yields the base 2 number. As an example, let's work out the conversion of  $105_{10}$  to base 2.

Divide		Remainder
2	105	
2	27	1
2	18	1
2	9	0
2	4	1
2	2	0
2	1	0
	0	1



Reading the remainders upwards as shown by the arrow give  $1001011_2$  as the solution.

### 1.3 Hexadecimal Numbers (base 16)

Hexadecimal numbers have 16 different states/numerals. It uses a combination of the first ten numerals of the decimal numbers and the first six alphabets. This shown in table 2.3

State	Hexadecimal	Decimal Equivalent	Binary Equivalent
1	0	0	0000
2	1	1	0001
3	2	2	0010
4	3	3	0011
5	4	4	0100
6	5	5	0101
7	6	6	0110
8	7	7	0111
9	8	8	1000
10	9	9	1001
11	A	10	1010
12	B	11	1011
13	C	12	1100
14	D	13	1101
15	E	14	1110
16	F	15	1111

A prefix of 0x before a number shows it is a hexadecimal number e.g. 0x3A7. The conversion of a binary number to hexadecimal involves dividing the binary number into a section of four bits and each nibble (4 bits stream) converted to its hexadecimal equivalent using table 2.4.

For example a conversion of 11010011100100101011 to hex will involve the following

Step 1: Segment the bit stream to 4 bits	1101	0011	1001	0010	1011
--	------	------	------	------	------

Step 2: Convert each bit to Hex	D	3	9	2	B
Step 3: Join the hex together with 0x prefix	0xD392B				


Conversion to decimal will involve a numbering all the numerals, starting from 0 for the least significant bit. As an example table 2.4 shows the conversion of 0xD392B to decimal.

Power of Base	4	3	2	1	0	Decimal Equivalent
Binary Number	D	3	9	2	B	$D(13) \times 16^4 + 3 \times 16^3 + 9 \times 16^2 + 2 \times 16^1 + B(11) \times 16^0 = 866603$

The conversion of decimal to hexadecimal, just like the conversion from decimal to binary, involves series of division (this time by 16) and remainders written from the last to the first gives the hexadecimal number.

For example conversion of 734593 to hex is:

Divide	Remainder
16   734593	
16   45912	1
16   2869	8
16   179	5
16   11	3
16   0	11 (B)



From the table,  $734593_{10} = 0xB3581$ .

#### 1.4 Integers and Floating Point Numbers

There are a lot of different types of numbers and each is different from another, yet they may share some common characteristics. To help keep track of and understand the similarities and differences between numbers, mathematicians have developed a grouping system that categorizes and describes numbers based on their characteristics. Some of the most common groups in the system are: Natural numbers, Whole numbers, Integers, Rationals, Real numbers



## Integers

An integer is a whole number (is not a fractional number) that can be positive, negative, or zero. Examples of integers are: -6, 2, 8, -95, and 2015. Examples of numbers that are not integers are: -2.37,  $\frac{5}{7}$ , 3.142, .07, and 45.1. The set of integers, denoted  $Z$ , is formally defined as follows:  $Z = \{\dots, -3, -2, -1, 0, 1, 2, 3 \dots\}$ . Integers are represented in computers using two notations: two's complement notation and the excess notation, both of which are based on binary numbers.

In the two's notation, a fixed number of bits are used to represent the integers. The number of bits used determines the range of integers that the notation can represent. Today's computers use 32 bits to represent integers, but we shall be using a 4 bit system to explain how it works. The most significant bit (that is the rightmost bit is used to represent the sign of the number, a 0 is used to represent positive, while a 1 depicts a negative. The procedure for obtaining the 2's complement of a positive number is: First to get its binary equivalent, secondly flip all the bits (convert all the 0 to 1, and the 1 to 0) and finally add 1 to the last bit on the right.

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

A four bit system can only represent the integers from +7 to -8, this is why today's computers use more bits to represent integers. For example, find how a computer would represent a -14 using its two's complement.

Step 1: Obtain the binary equivalent of 14, this is 1101

Step 2: Add the sign bit, 01101 (it is zero for the positive 14)

Step 3: Flip all the bits, this results in 10010

Step 4: Add 1 to the last bit, this results in 10011.

Therefore the two's complement representation for -14 is 10011. To represent -14, a minimum of five bits is required.

Integers can be represented also using another method known as the excess notation. To get this notation system, the bit pattern length to be used is identified and the different binary bit pattern listed from 0000.... Upwards. We then identify the point where the most significant bit changes sign, and assign 0 to that point. All bit patterns upwards of that point is positive, while those below represents negative integers. Table 2.7 shows a four bit length used to represent integers using the excess notation, the shaded portion represents the part with 1 as the most significant bit and therefore representing the positive integers.

		Excess Notation	Integer Represented
Positive		1111	7
		1110	6
		1101	5
		1100	4
		1011	3
		1010	2
		1001	1
		1000	0
Change of sign →		0111	-1
Negative		0110	-2
		0101	-3
		0100	-4
		0011	-5
		0010	-6
		0001	-7
		0000	-8

## Floating Point Numbers

Floating point numbers refers to numbers that are not whole, but contain a part that is less than one (a fractional part). There is usually a decimal point between the part of the number greater than one and the part that is less. These numbers are represented to the computer in a form called floating point notation. The highest order bit is designated as the sign bit, a zero in this bit signifies a positive number, while a 1 is used to represent a negative number. The remaining bit is divided into two parts, the exponent (used to hold the part greater than one) and the mantissa (which holds the part less than one). For example, in an eight bit system, the first bit is used as a sign bit, the next 3 are the exponent and the last four used to hold the mantissa.

For example, if a floating point notation contains 01011101, what is the binary number represented? What is the decimal equivalent of the notation?

Break up the given bit stream to the three segments

Sign Bit	Exponent	Mantissa
0	101	1101

The given stream represents a positive bit stream of 101.1101.

To obtain the decimal equivalent, a conversion to decimal using the table below is necessary

	Sign bit	Exponent			Mantissa			
Binary Number	0	1	0	1	1	1	0	1
Power of Base		2	1	0	-1	-2	-3	-4
Decimal Equivalent	+	$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 5.8125$						

Therefore the bit stream 101.1101 represents 5.8125.

The representation in the binary notation can introduce error into the system due to the limited number of bits used to hold both the exponent and the mantissa. This error is known as truncation error or round-off error. This error is reduced by using more bits to represent the mantissa field and in today's computers use at least 32 bits for storing values in the floating point notation instead of the 8 bits used in the examples.

## **Unit 3: Representation of Non-Numeric Data, Records and Registers**

### **1.0 Introduction**

In the previous unit, number bases were introduced to be the number of numerals or states that the number can contain. The common bases that are used to represent data to the computer system are the base 2 (binary), base 10 (decimal) and base 16 (hexadecimal). This unit will be focusing on the representation of non-numeric data such as alphabets and special characters, and also records and registers will be introduced.

### **1.1 Representation of Non-numeric Data**

Non-numeric data refers data that are not numbers and cannot be arranged sequentially based on their values. These cannot be added nor subtracted like numeric data and examples include alphabets, special characters, images and sound. The number bases explained earlier can be used to represent numeric data to computers, but the representation of non-numeric data makes use of a coded format. The coded format used for each type is different and can be varied. This unit will focus on the representation of alphabets and special characters.

The American National Standards Institute (ANSI) came up with the American Code for Information Interchange (ASCII) which have been adopted widely and used to code the representation of letters. In this code seven bits are used to represent the upper and lower case alphabets, punctuation marks and other control functions on the keyboard such as the carriage return, tabs and backspace. Shown in Table 3.1 is the ASCII representation for alphabets and special characters. From the sequence shown, the codes for the English letters are in the same sequence as their lexical order. As such the codes for alphabets can be determined if the codes for the lower and upper case of 'A' are known.

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	\$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-`	63	3F	?	95	5F	_	127	7F	DEL

Table 3.1: Table of ASCII Codes

Example: What will be the ASCII code (in binary) to represent CODEl!

From the table, you can obtain the decimal code for each of the characters of the word CODEl! Then convert each of the decimal code to binary. Note that the case of the letter matters as upper case codes differs from lower case codes.

Letter	C	O	D	E	l	!
Decimal	67	79	68	101	108	33
Binary	1000011	1001111	1000100	1100101	1101100	0100001

Therefore the ASCII code for CODEl! is 100001110011111000100110010111011000100001.

The ASCII code is limited in the number of bits used in the coding. As such cannot be used to represent the alphabets of languages that have a large number of alphabets such as Chinese and Japanese. A more recent coding system known as the Unicode have been developed by leading computer manufacturers which uses 16 bits to represent each symbol, resulting in 65,536 unique different pattern that can handle languages with large number of alphabets.

The International Organization for Standardization have also developed a code format that uses 32 bits for its encoding. This will be able to represent billions of different symbols.

## 1.2 Computer Records and Registers

Generally, a record is a combination of various data objects such as some integers, floating-point numbers, and character strings that are arranged for processing by a program. A file contains multiple records and can also be referred to as a data set. The programming language and/or the application used determines the organization of data in the record. Typically, records can be of fixed-length or be of variable length with the length information contained within the record.

A register is a memory that is built into the central processing unit to speed up its operations by providing access to commonly used values. They are usually of small value and can be accessed very quickly. They are made of semiconductor devices whose contents can be read and written to at extremely high speeds but the contents are held only temporarily, usually while in use or only as long as the power supply is on.

Registers are the fastest way for the system to manipulate data and are at the top of the memory hierarchy. Registers are normally measured by the number of bits they can contain. Examples include an 8-bit register (can store 8 bits of data), a 32-bit register (can store 32 bit of data). There are basically two kinds of registers, a set of 32 registers that is accessible to a programmer (known as the General Purpose Registers) and another set of 6 registers not accessible to programmers that is used in interpreting and executing instructions. Data and instructions must be put into the system. So we need registers for this. The basic computer registers with their names, size (for an 8085 microprocessor) and functions are listed below, the size indicate can vary depending on the processor in question.

Register Symbol	Register Name	Number of Bits	Description
AC	Accumulator	16	One of the Processor Register that holds operands and the result of arithmetic operations
DR	Data Register	16	Hold memory data
TR	Temporary Register	16	Holds temporary Data
IR	Instruction Register	16	Holds the current instruction code being execute
AR	Address Register	12	Holds memory address
PC	Program Counter	12	Holds address of next instruction, after executing the current instruction

INPR	Input Register	8	Holds Input data
OUTR	Output Register	8	Holds Output data

### References and Further Readings

Brookshear J. G. (2005), *Computer Science: An Overview*, 9th Edition, Pearson Addison Wesley.

Norton, P. (2005), *Peter Norton's Introduction to Computers*, 6th edition, McGraw-Hill/Irwin

Pfaffenberger, B. (2002), *Computers in your Future* 4th edition, Prentice Hall

Rajaraman V and Radhakrishanan T, (2009), *An Introduction to Digital Computer Design*, 5th Edition, PHI Learning Private Limited.



# Module4

## Logic

Unit 1: Digital Logic

Unit 2: Memory

Unit 3: Von Neumann Model of Computation

## **Unit 1: Digital Logic**

### **1.0 Introduction**

A large proportion of electronic equipment, including computers uses digital electronics where the quantities (usually voltage) are described by two states (on and off). These two states can also be represented by true and false, 0 and 1 but in most physical systems they are represented by the voltages 5V and 0V or something close to that. The operation of almost all modern digital computers is therefore, based on two-valued or binary system.

This unit introduces you to the fundamental concept behind all modern computer system and the basic motivation and rationality underlying the design and implementation of various digital/computer components and systems. The concept of Combinational logic and Sequential logic are presented. The unit also presented the introduction to the primitive logic devices called Logic Gate which form the building block of all digital logic circuit design.

### **1.1 Concept of Digital Logic**

Digital logic is simply a system of rules that allow us to make extremely complicated decision based on relatively two states questions, and therefore, formed the basic motivation and rationality underlying the design and implementation of various computer hardware components and systems. Boolean algebra (due to George Boole) is the mathematics of digital logic and is useful in dealing with binary system of numbers. Boolean algebra is used in the analysis and synthesis of logical expressions and for manipulation of object that can take only two values typically true or false or any pair of values. The principles of Boolean algebra are applied to switches, providing mathematical tools for analysis and synthesis of any switching system. The application of Boolean algebra in design of modern computing system and devices is very significant and obvious today. This is because the relationship between Boolean logic and physical component of any computer system is very strong. Each hardware component of a computer/digital system is built of several logic circuits. Digital logic circuit is classified into two categories namely: Combinational logic circuit and Sequential logic circuits. Details of each category is given in the next sub-section.

### **1.2 Combinational and Sequential Logic Circuit.**

A logic circuit is an interconnection of several primitive logic devices called *Logic Gates* to perform a desired function. When logic gates are connected together to give a specified output for certain specified combination of input variables, with no storage involved, the

resulting network is known as Combinational logic circuit. In other words, a combinational logic circuit contains only Logic gates but does not contain storage elements. The output of a combinational logic circuit at all the time is a pure function of the present input level only. Figure 1.0 shows you the block diagram of combinational logic circuit.

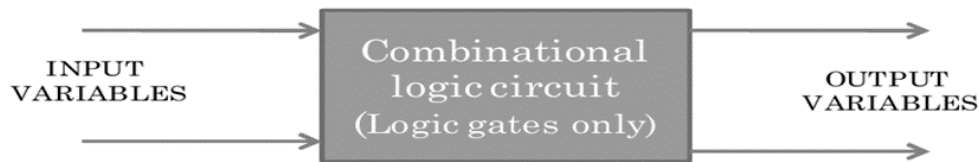


Figure 1.0: Block diagram of combinational logic circuit

However, a sequential logic circuit is a combinational logic circuit with memory element(s). It consists of a combinational circuit to which memory elements are connected to form a feedback path. In contrast to the combinational logic circuit, the output of a sequential logic circuit depends not only on the present input but also on the history of the input. In other words, sequential logic has memory while combinational logic does not. Figure 1.2 shows the block diagram of a sequential logic circuit.

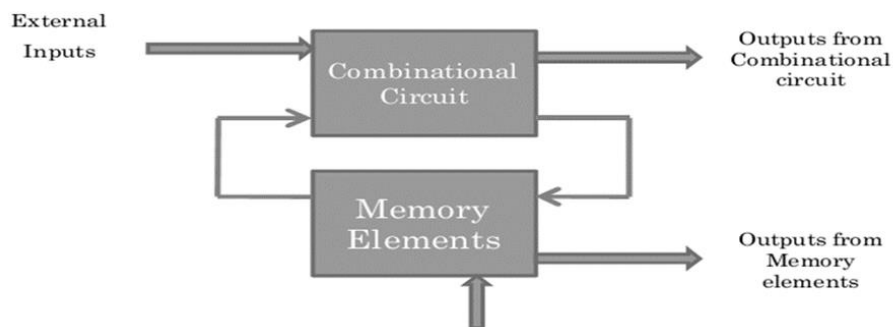


Figure 1.2: Block diagram of a sequential logic circuit

It can be seen from the diagram that the output(s) of sequential logic circuit is (are) dependent not only on external input(s) but also on the present state of the memory element(s). The next state of the memory element(s) is also dependent on external input and the present state.

Although, every digital system is likely to have combinational circuits, most systems encountered in practice also include storage elements, which require that the system be described in terms of sequential logic. Practical computer circuits normally contain a mixture of combinational and sequential logic. For example, the part of an arithmetic logic unit, or ALU, that does mathematical calculations is constructed using combinational logic. Other

circuits used in computers, such as half adders, full adders, half subtractors, full subtractors, multiplexers, de multiplexers, encoders and decoders are also made by using combinational logic. To implement a sequential circuit, a storage device is required to recognise what has happened in the past. The basic unit of storage is called flip-flop or Latch and the simplest kind of sequential circuit is a memory cell that has only two states. Such two state sequential circuits are called flip-flops, bi-stable-multi-vibrator, latch or toggle.

### 1.3 Introduction to Boolean Algebra and Logic Gate

Since computers/digital devices are built as a collections of switches that have only two states (either “on” or “off”). Boolean algebra is the natural way to represent digital information. As introduced earlier in 3.0, Boolean algebra, the mathematics of digital logic is used in the analysis and synthesis of logical expressions.

Logic circuits are usually represented by Logical expressions which is a combination of logical variables (x, y) and logical operators (AND, OR, NOT, etc.).

A Boolean function typically has one or more input values and yields a result, based on these input values in binary form {0, 1}. The three basic Boolean operations/operators are: OR, AND, and NOT operation.

Other two Boolean operators (NOR and NAND) are combination of two of this three basic operators as shows below:

NOR = NOT and OR

NAND = NOT and AND

Thus, the five most common Boolean operators/operations are:

**(OR, AND, NOT, NOR, and NAND)** operators.

A Boolean operator can be completely described using a truth table that lists the inputs, all possible values for these inputs, and the resulting values of the operation for all possible combinations of these inputs. A truth table shows the relationship, in tabular form, between the input values and the result of a specific Boolean operator or function on the input variables.

A **logic gate** is a small electronic device that implements a simple Boolean function. They makes the logical decisions based on the logic operations and became the basic building blocks for digital circuit design. Each of the five Boolean operations described above has its equivalent Logic Gate which is the actual physical electronic device that makes the logical decisions based on the corresponding logic operation. Thus, the five equivalent Logic Gates are:

- i. OR Gate which perform OR operation
- ii. AND Gate which perform AND operation
- iii. NOT Gate which perform NOT operation.
- iv. NOR Gate which perform NOR operation
- v. NAND Gate which perform NAND operation.

The graphical representation of the three basic logic gates and their equivalent true tables are respectively shown in figure 1.3 and 1.4 below:

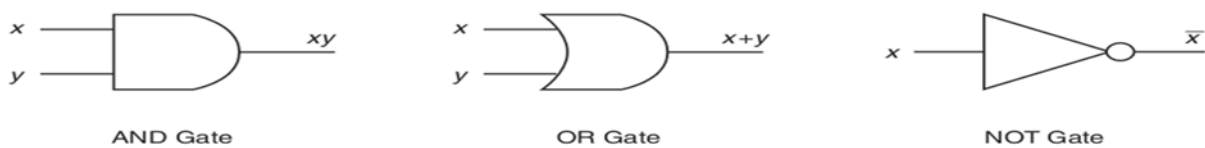


Figure 1.3: Logic gates graphical representation

Inputs		Outputs
x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

Inputs		Outputs
x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

Inputs	Outputs
x	$\bar{x}$
0	1
1	0

Figure 1.4: Truth Tables for AND, OR and NOT logic operations

**AND Operation:** AND operation is represented by  $xy = x \cdot y$ . Its associated Truth Table is shown in fig 4. A truth table gives the value of output variable (here xy) for all combinations of input variable values. Thus in an AND operation, the output will be 1 (True) only if all of the inputs are 1 (True).

**OR Operation:** OR operation is represented by  $x \text{ OR } y = x + y$  where  $x$  and  $y$  are logical (Boolean) variables and the  $+$  sign represents the logical addition called an 'OR' operation. Its associated Truth Table are shown in fig 4. Thus in an OR operation, the output will be 1 (True) if either of the inputs is 1 (True). If both inputs are 0 (False), only then the output will be 0 (False).

Note that though, the symbol  $+$  is used, the logical addition described above does not follow the rules of normal arithmetic addition.

**NOT Operation:** The NOT gate has only one input which is then inverted by the gate. The symbol and truth table for the operation are shown in fig 1.3 and 1.4.:

**NAND Gate:** As introduced to you earlier, we could combine AND and NOT gates together to form a NAND gate. Thus the logical expression for a NAND gate is  $C = \overline{A \bullet B}$

The symbol and truth table are shown in figure 1.5. The NAND gate symbol is given by an AND gate symbol with a circle at the output to indicate the inverting operation.

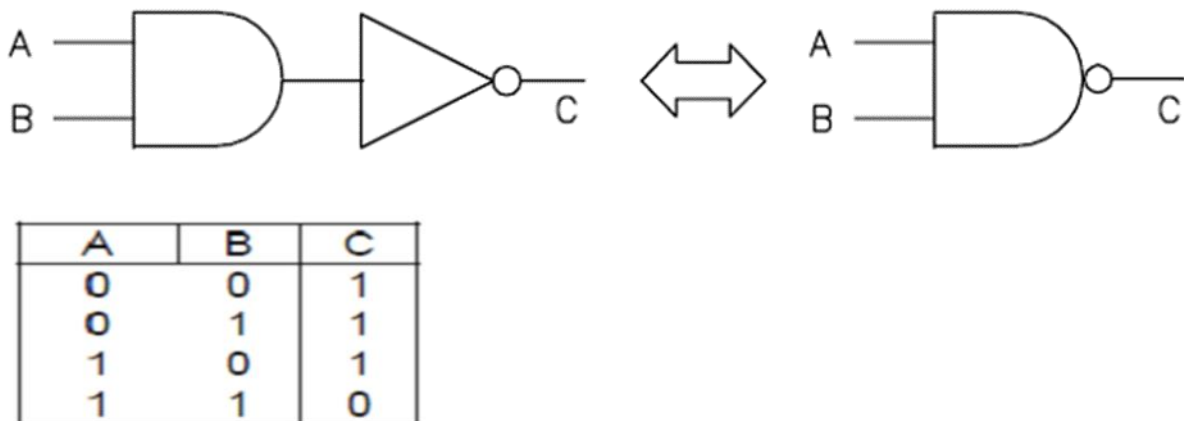
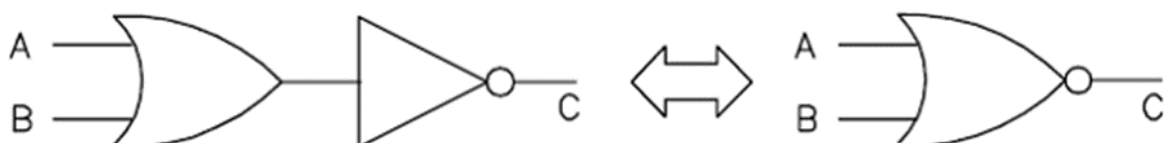


Figure 1.5: logic symbol and Truth Table for NAND Gate

**NOR Gate:** Similarly, OR and NOT gates could be combined to form a NOR gate. The Gate symbol and Table are depicted in figure 1.6.



A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

Figure 1.6: logic symbol and truth table for NOR Gate

## Unit 2: Memory

### 1.0 Introduction

We have introduced to you the term “Flip-flops” in this module as the basic unit of storing binary information in sequential logic circuit. A register is also a device used either to store data temporarily or to influence data stored in it using the logic circuitry around it. A set of such registers where programs and data are stored is called Memory. This unit introduces you to the concept of Memory subsystem of a digital computer. The two basic classification of memory devices is presented. They are Read only memory (ROM) and Random access memory (RAM). The unit also presents cache memory and the concept of Memory hierarchy then followed.

### 1.1 Memory Concept

The term Memory subsystem of a digital computer is simply a set of registers where programs and data are stored. It is logically structured as a linear array of locations, with addresses from 0 to the maximum memory size the system processor can address.

It is required, for optimum performance of the machine that, the stored data and programs be accessible by control and processing units as fast as possible. The main memory described as primary memory permits such a fast access. This fast-access requirement augment a significant amount of hardware to the main memory and thus makes it expensive. Data and programs not immediately needed by the machine are usually stored in secondary memory to reduce memory cost. They are brought into the main memory when need by processing unit. We examine various types of memory in the next sub section.

### 1.2 Types of Memory

Although, you may see different classification of memory devices depending on different mechanism. The two basic types of memory devices are:

*Random-access Memory (RAM) and Read-only Memory (ROM)*

**Random Access Memory (RAM):** Random Access simply means the content of data and programs stored in the memory can be accessed in a random manner. In this type of memory, any addressable location can be accessed randomly. In other words, the method of



‘writing into’ and ‘reading from’ a location in a RAM is identical and consumes the same time regardless of its position in the memory.

The terms *Read/Write memory (RWM)* and *RAM* are used interchangeably in most literature to describe the same thing. RAM is the most common type of Primary/main memory in utmost digital system/devices. The specification of 500 megabytes RAM of your computer simply refers to 500MB of RWM. Although, RAM stores data and programs the computer needs when executing instructions, it is volatile, and loses this information once the power is turned off. Each memory register or location in a RAM is associated with an ‘address’. Data are ‘written into’ and ‘read from’ RAM memory location by using address to locate the position of the data in the memory. There are generally, two types of semiconductor RAMs based on the chips used to build the bulk of RAM memory in today’s computer systems: they are Static and Dynamic RAM (SRAM and DRAM). They are differentiating as follow:

- i. Static RAM is constructed with a flip-flop while Dynamic RAM is constructed out of capacitors that leak electricity.
- ii. Consequently, the contents of SRAM (either 1 or 0) is static as long as the power is on whereas, the state of 1 or 0 is determined by the charge level of the capacitor in dynamic RAM.
- iii. Since the charge decays with time, the memory cell of DRAM requires often refresh (i.e. recharge) to hold its memory content.
- iv. Because of the refresh time required by DRAM, they are slower and less expensive than SRAM.
- v. However, DRAM uses less power, generates less heat and more DRAM cells can be fabricated on the same area of silicon than SRAM cell.
- vi. Thus, designers often use both technologies in combination: DRAM for main memory and SRAM for cache memory.

Although, the basic operation of all DRAM memories is the same, different types of DRAM exists as listed below:

- i. Multibank DRAM (MDRAM),

- ii. Fast-Page Mode (FPM) DRAM,
- iii. Extended Data Out (EDO) DRAM,
- iv. Burst EDO DRAM (BEDO DRAM),
- v. Synchronous Dynamic Random Access Memory (SDRAM),
- vi. Synchronous-Link (SL) DRAM,
- vii. Double Data Rate (DDR) SDRAM, and
- viii. Direct Rambus (DR) DRAM.

SRAM also exist in different types as listed below:

- i. Asynchronous SRAM,
- ii. Synchronous SRAM, and
- iii. Pipeline burst SRAM.

Note: The basic operation of all SRAM memories is also the same and not discuss in this course material.

**Read-Only Memory (ROM):** Remember in RWM, Data are ‘written into’ and ‘read from’ the memory location by using address to locate the position of the data in the memory. The characteristics of ROM are as follow:

- a. Although, ROM is also a random access memory, Data can only be ‘read from’ the memory location of ROM.
- b. Unlike RAM, data are usually ‘written into’ the ROM either by memory manufacturer or by the user using special devices that can write (burn) the data pattern into the ROM.
- c. In most computers, ROM is a smaller memory in addition to RAM that stores data and programs necessary to boot computer system.
- d. ROM is not a volatile memory and is used in any system that does not require alteration in the system programming.
- e. Many appliances, calculators, peripheral devices and most automobiles use ROM chips to store and preserve information when the power is shut off.

There are five basic different types of ROM: (1) ROM (2) PROM (3) EPROM, (4) EEPROM, and (5) flash memory.

- 1) PROM (programmable read-only memory) is a variation on ROM that can be programmed by the user with an appropriate equipment. While ROMs are hardwired,

PROMs have fuses that can be blown to program the chip. Data and instructions in PROM cannot be altered once programmed.

- 2) EPROM (erasable PROM) is a PROM with the added advantage of being reprogrammable with the aid of a special tool that emits ultraviolet light.
- 3) EEPROM (electrically erasable PROM) eliminates shortcomings of EPROM by applying an electrical field that can erase only portions of the chip, one byte at a time.
- 4) Flash memory is basically an EEPROM with the additional advantage that data can be written or erased in blocks, eliminating the one-byte-at-a-time limitation. This makes flash memory faster than EEPROM

### **1.3 Memory Hierarchy**

Computer processor usually and frequently reading information from computer memory and the processor speed is higher than the memory access time. The primary memory of a computer system is constantly built out of RAM devices, thereby allowing the system processing unit to access data and instructions in the memory as quickly as possible. Since primary memory devices are more expensive than secondary memory devices, a cost effective memory system results when the primary memory capacity is minimized. To reduce the cost and the speed gap, a small but faster memory called cache memory is usually introduced between the main memory and the processing unit.

Cache memory is simply a small, temporary, and very high-speed memory that the processor uses for information it is likely to need again in the very near future. It is usually faster than the primary memory.

In a cost effective practice, it is not necessary to store the whole data or program in the primary memory as long as the portion of the program or data needed by the processing unit is in the primary memory. A secondary memory built out of direct- or serial -access devices is then used to store programs and data not immediately needed by the processing unit. Modern computers usually have cache memory where data from frequently used memory locations may be temporarily stored.

Memory hierarchy is one of the most significant considerations in understanding the performance capabilities of a modern processor. However, as we have introduced to you so far, memories are differently created with different characteristics and capability. Some types are far less efficient and therefore cheaper than others. To overcome this discrepancy, a

combination of memory types is used in today's computer systems to achieve the best performance of the system at the lowest cost. This methodology is called Memory Hierarchy.

Thus, in a general-purpose computer system, there is a Memory Hierarchy in which the highest speed and most expensive memory device is closest to the processing unit while the slowest and least expensive memory devices are farthest from the processing unit.

The memory system hierarchy shared in most today's computer systems comprises of the following levels:

1. CPU registers (basic unit of storage)
2. Cache memory (a fast, small RAM block)
3. Primary (main) memory (the RAM from which the processor accesses all programs and data via the cache memory) and
4. Secondary (mass) memory.

**Note:** The registers in the processing unit are temporary storage devices and they are the fastest components of the computer system memory.

The speed and cost per bit of the memory is highest at level 1 and decreases across the levels to the lowest at level 4.

The main objective of the hierarchical memory design is to facilitate a speed-cost compromise that will provide a memory system of desired capacity with the highest speed possible at the lowest cost.

## **Unit 3: Von Neumann Model of Computation**

### **1.0 Introduction**

The primary function of a computer system is to process the input data to yield a results that can be useful in a specific application environment. In 1880, Herman Hollerith developed numerous unit record machines to process data on punched cards for census applications in the United States. In 1944, the first automated computer (Mark I), that used paper tape for program storage and punched cards for input and output data was announced. The need for faster computations than those provided by Mark I, give rise to the development of the first electronic computer (ENIAC), which employed stored-program concept' by a team led by Eckert and Mauchly. In ENIAC, a sequence of instructions called 'program' is stored in the memory for use in processing data by the machine. The programs were wired on a control board and a rewiring of the board is required for each computation sequence. The first stored-program computer (EDVAC), was developed by John von Neumann, a member of the Eckert-Mauchly team in 1945. This unit introduces you to the concept of the first stored-program computer called von Neumann Model of Computation. The von Neumann architecture and Fetch-Decode-Execute cycle of the architecture are also presented in this unit. We finally examined the basic machine or computer organisation.

### **1.1 Von Neumann Model**

The first stored-program computer (EDVAC), was developed by John von Neumann, a member of the Eckert-Mauchly team in 1945. All stored-program computers using the von Neumann architecture are known as von Neumann machine/system. Although, several different architecture have evolved, all modern computers are Von Neumann machine. As shown in Figure 3.0 below, Von Neumann model is a uniprocessor computer system that consist of the following four (4) basic units:

- i. Memory Unit; which carries Memory Buffer Register (MBR) and Memory Address Register (MAR) or Memory Data Register (MDR).
- ii. Control Unit ; which contains Program Counter (PC) and Instruction Registers (IR))
- iii. Arithmetic and Logic Unit (ALU); which carries Accumulator (ACC)
- iv. Input/output Unit.

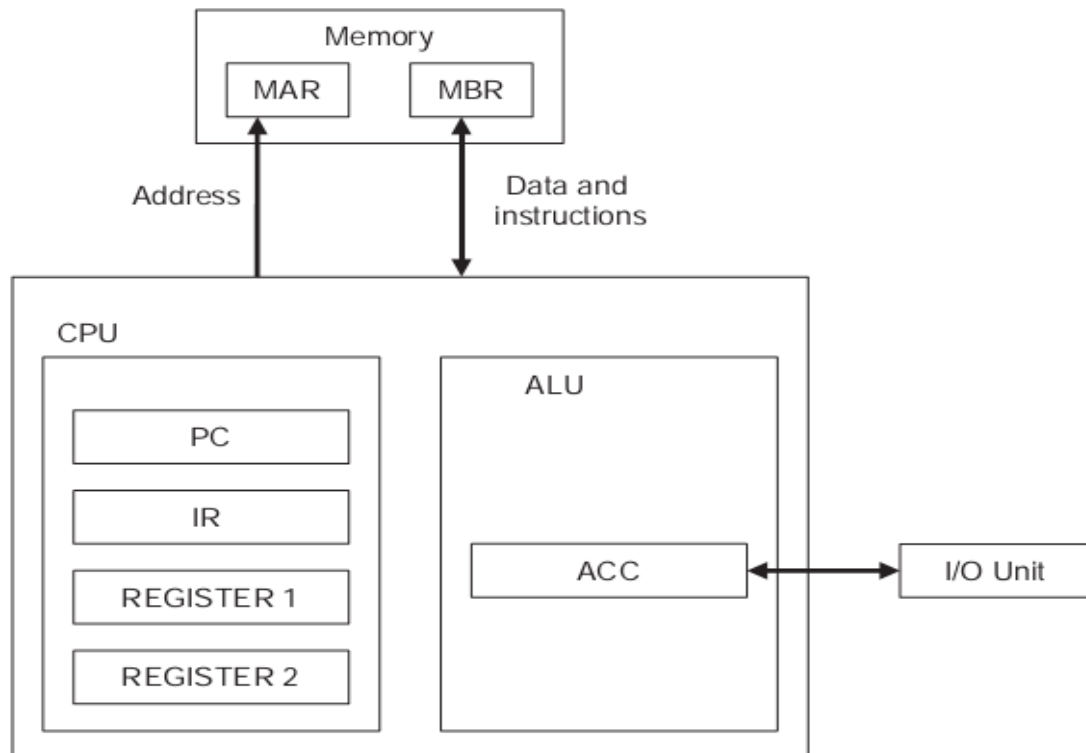


Figure 3.0 von Neumann architecture

**Remember:** Registers are small unit of storing data whereas, ACC is one form of register that store intermediate result of arithmetic and logic operation.

**Memory Unit:** This is a single port storing device in which the arrangement of memory cells is in form of several memory words. Each memory word is the unit of data that can be 'read' from or 'written' to the memory device through the single port of the memory.

**Arithmetic and Logic Unit:** This unit executes the arithmetic and logic operations on the data in ACC and/or MBR and the result of the operation is retained in the ACC.

**Control Unit:** The Program Counter (PC) in this unit holds the address of the instruction to be fetched whereas, the instructions are fetch into the Instruction Registrar (IR) from memory for execution. Other two included registers can be used to hold data and address values during computation.

**Input/output Unit:** This unit connects Input/output devices to the system. In this model, the I/O devises send and receive data into and from the ALU subsystem for simplicity. In

practice, the data exchange may also occur directly between I/O devices and memory without exploiting any register.

It is very necessary for you to note the following characteristics of the von Neumann that make it inefficient:

- i. Data and programs are stored in a single sequential memory, which create a single path memory access referred to as the ***Von Neumann Bottleneck***.
- ii. There is no distinction in representation of data and instruction in the memory.
- iii. In data representation, there is no distinction between a set of bits representing different data e.g. floating-point data and a character string.
- iv. High level language programming environments utilize several data structure and one dimensional single sequential memory requires linearization of such data structure

Consequently, the von Neumann model requires excessive mapping by compilers to generate the executable code from the programs written in high-level languages. Despite these deficiencies, practical structure of most digital computers follows the von Neumann model. However, the development of numerous compilers over years has significantly reduced the semantic gap problem created by the model.

The von Neumann architecture described to you so far, runs programs in what is known as von Neumann execution cycle which is also called *fetch-decode-execute cycle*. Section 3.1 presents the execution cycle.

## **1.2 Fetch-Decode-Execute Cycle**

The fetch-decode-execute cycle describes how the von Neumann machine operates. The following steps described one iteration of the cycle.

- a. The program counter determines the location of the instruction in the memory and the control unit *fetches* the next program instruction from the memory.
- b. The instruction fetched is *decoded* into a language that ALU can understand.
- c. All data operands necessary for execution of the instruction are fetched from the memory and booked into registers within the CPU.
- d. The ALU then *executes* the instruction and places the results in memory or register

The three main operations of the cycle are *fetch*, *decode*, and *execute*. The control unit fetches the instruction from the memory, the instruction is decoded into ALU language and then finally execute by the ALU. The next sub-section presents the basic machine organisation.

### 1.3 Basic Machine Organization

The main hardware components (CPU, Memory, and I/O) of most computers available in market today are interconnected according to the von Neumann model and therefore, described as von Neumann computers. However, the architecture of the model have been extended and rationalized into what is presently designated as “*system bus model*” which is shown in figure 3.1. The model shows a more generalized architecture of modern day computer system.

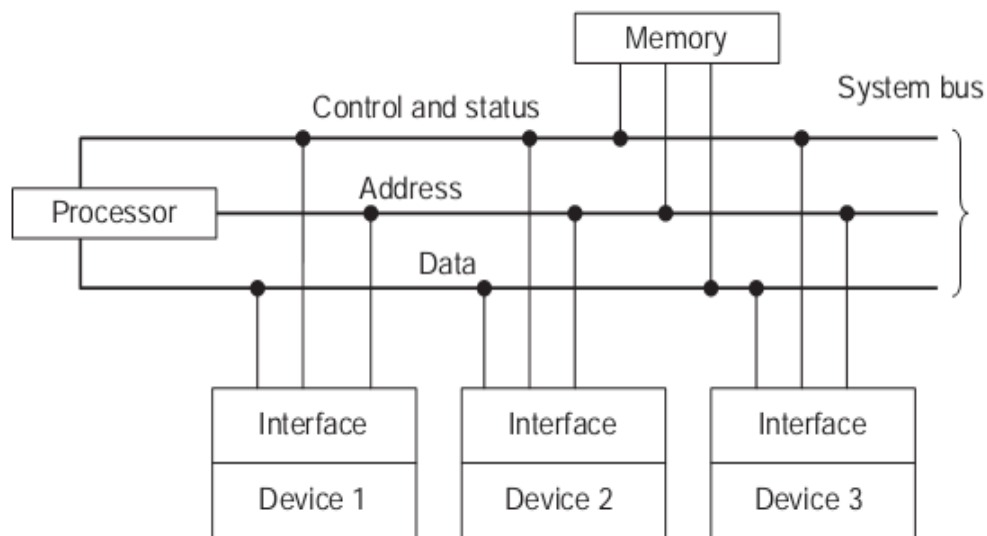


Figure 3.1: General Computer System Organization

The model architecture is made up of three major sub-systems (Processor, Memory and Input/output sub systems) which are all interconnected by the system bus. The system bus consists of three different buses which are: control bus, address bus and data bus whereas, the processor subsystem (CPU) comprises of the ALU, the Control unit and various processor registers.



Remember: The architecture described to you is a single-bus system architecture which may differ from practical system that may be configured around multiple buses to allow simultaneous operations on the buses. For example: in two bus structure, the processor may be connected to memory sub-system by memory bus whereas, the I/O bus interface the I/O devices to the processor. Although, the system complexity increases with multiple bus, there is possibility of higher throughput compared to single-bus architecture due to simultaneous operations on the buses. Consequently, a cost-speed trade-off is necessary to choose the system structure.

### **References and Further Reading**

The Essentials of Computer Organization and Architecture. Linda Null and Julia Labur.  
2003.

Computer Organization, Design, and Architecture. Fourth Edition. Sajjan G. Shiva  
CRS Press.

Stallings, W. Computer Organization and Architecture, 5th ed., New York: Macmillan  
Publishing Company, 2000.

Computer Organization and architecture. Patterson and Hennessy (1997)

Hennessy, J. L., & Patterson, D. A. Computer Architecture: A Quantitative Approach, 2nd  
ed., San

Gregg, John. Ones and Zeros: Understanding Boolean Algebra, Digital Circuits, and the  
Logic of Sets. New York: IEEE Press, 1998.

Tanenbaum, Andrew. Structured Computer Organization, 4th ed. Upper Saddle River, NJ:  
Prentice Hall, 1999.

# Module5

## Programming the Computer

Unit 1: Programming languages

Unit 2: Algorithms

Unit 3: Flowcharts

## Unit 1: Programming languages

### 1.0 Introduction

Historically, in the early period, computers are given instructions through the control buttons. However, the buttons are fixed and not flexible for executing different instructions. Therefore, this unit will expose you to the flexible way of instructing computer machines through what is called programming language. The unit will define the programming language and its purpose; the programming paradigms; and the programming translators that are used to convert one programming language to another.

### 1.1 Programming Languages

When we want to communicate with one another, a language is normally used either English language or any other language. Similarly, a programming language is a way of communication between a computer programmer and a computer itself. A programmer is the person that writes a set of instructions using a programming language to instruct the computer to perform a task. A program is a set of instructions that are logically related.

As we all know, every language comprises a collection of words and symbols, which the users of the language use for communication. The collection of words and symbols is known as vocabulary of a language.

However, programming languages use *syntax* instead of vocabulary. This *syntax* is a set of rules used by programmers to write programs that the computer understands.

Programming languages can be broadly categorized into three languages:

1. Machine language
2. Assembly language
3. High level language

If you recall a program written in any programming language involves a set of instructions that are logically related. These instructions have two parts, as shown in figure 1.1 below.

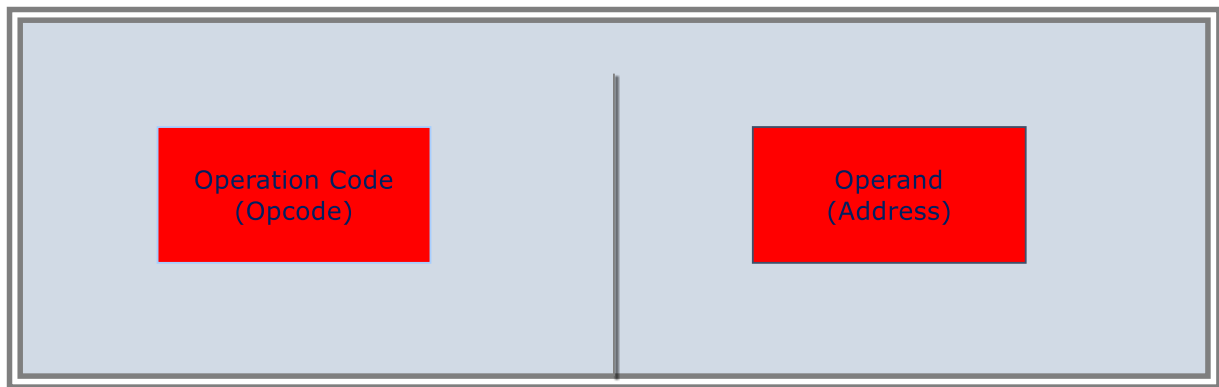


Figure 1.1 Format of a programming language instruction

The two parts of a programming language instruction are:

1. **Operation Code (Opcode):** This is the part that instructs a computer about the operation to be performed.
2. **Operand:** This is the part that instructs the computer about the location of the data on which the operation stated by the opcode is to be performed.

For example, look at this instruction Add A and B, “Add” is the Opcode while A & B are the operands.

Every computer has its own set of opcode. The following operations can be performed by the opcode in the instruction set. These operations include:

- i. **Logical Operations:** These operations are used for comparing data and the results can be either true or false.
- ii. **Arithmetic Operations:** these operations are used for mathematical calculations.
- iii. **Branch Operations:** These operations involve transfer of data to a memory location given in an operand field
- iv. **Data Movement Operations:** these operations involve the transfer of data between the memory and the processor or input/output devices and the computer.

### Machine Language

All computers use binary number system that comprise of 0 and 1 for performing internal operations.

The first language available for modern computers was the machine language and it is considered as the lowest level language. Machine language consists entirely of ONs and

OFFs denoted as 1s and 0s respectively and it varies from one type of machine to another. Since computers are designed to only understand 0s and 1s, it makes execution of machine language fast because it does not require any translator.

Although, programs can be written in a number of programming languages but computer only understands machine language. Therefore, the programs written in other languages need to be translated to machine language for execution.

The machine language is difficult to read and understand because it is written in binary digits which is difficult to learn. For example, a program to store a number might be

1001000101110001          1100101000111011

Machine language is the language of computer and it is also called a first-generation language.

Advantages of machine language

1. Program written in machine language executes fast
2. it does not require translator
3. it require a relatively small space

Disadvantages:

1. it uses binary or Hex code which are difficult for human to read
2. the programs of machine language are difficult to debug and maintain
3. The programmer needs to remember the memory locations of the data
4. Programs written in machine language are difficult to modify because of their complexity
5. The machine language code differs from one type of computer to another.

**Assembly Language**

*Assembly language* was introduced in 1952 in order to overcome the limitations of machine language. The assembly language uses alphanumeric notations instead of binary digits to represent instructions and memory addresses. For example, it uses ADD for addition and SUB for subtraction.

The alphanumeric code uses a set of letters and numbers to represent instruction. These alphanumeric code and symbols make the program shorter and easier to write than machine language. The possibility of errors is also reduced and the program can easily be modified.

An assembly language is also called a second generation language.

Below is a sample of a program written in assembly language to add two numbers:

1. LD A<sub>X</sub>, 8
2. LD B<sub>X</sub>, 9
3. ADD A<sub>X</sub>, B<sub>X</sub>
4. LD (100), A<sub>X</sub>
5. JMP B<sub>X</sub>
6. HLT

From here, A<sub>X</sub> and B<sub>X</sub> are registers. These registers are memory locations inside a microprocessor. Every instruction and data needs to be loaded from main memory to registers before the CPU can process it.

Each line of the above program is explained as follows:

1. Loads register A<sub>X</sub> with the value, 8.
2. Loads B<sub>X</sub> with the value, 9.
3. Adds the value of register B<sub>X</sub> to the value of register A<sub>X</sub>.
4. Stores the value of register A<sub>X</sub> in the main memory location, 100.
5. Uses JMP to jump to register B<sub>X</sub> transfer the control to register B<sub>X</sub>.

6. Stops the program execution

As we have mentioned earlier, a computer cannot understand any language except machine language. Because of this, a translation program is needed to interpret assembly language into machine language code. This translation program is called an assembler. A code written in any programming language other than machine language is known as source code. This source code requires translation for execution.

**Functioning of an Assembler**

The assembler takes each program statement in the source code of assembly language to generate a corresponding bit stream (a series of 1s and 0s of a given length). The set of instructions in a bit stream is the output of the assembler and it is called the *object code*. It is the object code that will now be executed whenever required. If errors are found in the source code, the assembler generates a list of errors instead of object code.

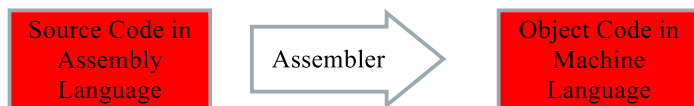


Figure 1.2 Functioning of an Assembler

**Advantages of Assembly Language:**

1. Assembly language programs are easier to write and understand than machine language
2. The programmer can easily remember the opcode and alphanumeric addresses of data and instruction
3. It saves time in developing and modifying programs in assembly language

### **Limitations of Assembly Language**

1. The execution of programs written in assembly language is slow compare to those written in machine language
2. Programs written in assembly language depend on processors. Therefore, it cannot be transported from one machine to another

### **High Level Language**

The machine and assembly languages are known as low-level languages because of their dependent on machine hardware. In order to create programs that are hardware independent, a new type of programming language called the *high level language* was developed by programmers. A high level language is close to human and it is user-friendly because it uses natural languages like English language to represent an instruction. It uses words such as PRINT to print and GOTO to jump to a particular line. Examples of high level language are Basic, COBOL, and Pascal.

High level language is also known as third-generation language. Each instruction in a high-level language is translated into multiple machine-level instructions.

### **Advantages of High-level languages**

1. High-level language is easier to learn than machine and assembly language
2. High-level language programs written in one type of computer can easily be used on another computer because it is machine independent
3. It makes reading, writing and maintaining of a program written on high-level language easy because high-level language vocabularies are similar to English language

### **1.2 Language Translators**

Program translator plays an important role in computer programming to translate high level language in to machine language. Any program written in high level language is called a source program which is the translated into machine code (i.e 1 & 0 codes) for machine to be able to execute the program. This conversion is done by a translator. There are two basic types of translators namely: compiler and interpreter. The following subsections will explain to you how compilers and interpreters work and also differ from each other.



## **Compiler**

For you to execute a program written in high-level language, a computer needs translation software called a compiler. Compiler is a program developed to convert the entire source program into machine code at once. A compiler is always language-specific, meaning that each high-level language has its own compiler. Compiler translates the source code of that language into the object code. For example, compiler for Pascal can only compile the programs written in Pascal. It cannot compile programs written in any other language, such as COBOL. Once the compiler creates the object program, it will never use the source program again unless modification is required in the source program. It is this object file that you normally copy from one computer to another computer for you to install and use.

## **Interpreter**

There are some high-level languages that use different type of translator program called an interpreter. Interpreter is a program that converts the source program into a machine code. Instead of creating an executable file, interpreter translates the source code line-by-line that is what makes it run slowly compared to other translators.

Debugging of errors is easier using an interpreter because error messages in the interpreter are usually specific and points out line on which the error occurs.

PERL, BASIC, and Visual Basic languages typically use interpreter.

## **Differences between a Compiler and an Interpreter**

The differences between a compiler and an interpreter are explained in the table below

S/N	Differences between a Compiler and an Interpreter	
	Compiler	Interpreter
1	A compiler translates the entire program first, and generates the object code, which can be later executed	An Interpreter translates and executes one line of source code at a time
2	The object code generated after compilation is saved for future use. The programmer does not need to compile the program again for object code generation. This makes the program execution fast	The translated lines of the code cannot be saved for future use. therefore, every time code needs to be translated before execution. This makes the program execution slow
3	A compiler detects and shows syntax errors that relate to the entire source code	An interpreter detects and shows errors that relate to a line of source code

Table 1.1 Differences between a Compiler and an Interpreter

### 1.3 Some Examples of High-level Languages

There are a number of high-level languages that have been developed since the first high-level language (FLOWMATIC) was developed in 1952. Some of the common high-level languages are Beginners All Purpose Symbolic Instruction Code (BASIC), Formula Translation (FORTRAN), Common Business Oriented Language (COBOL), Pascal, and C.

#### BASIC

Basic was developed in 1964 by Professor John Kenny and Thomas Kurtz at Darmouth College in the USA.

Basic is an interpreter-based language that is easy and simple to use. BASIC was one of the earliest high-level languages to be implemented on personal computers.

#### FORTRAN

FORTRAN is one of the oldest high-level languages. It was developed in 1957 by John Backus and his team at International Business Machines (IBM) Corporation. FORTRAN was designed to solve scientific and engineering problems. After the version was released, other versions evolved, such as FORTRAN II, FORTRAN IV, and FORTRAN 77. FORTRAN IV was the first language that was standardized by American National Standards Institute (ANSI).

ANSI is a non-profit making organization, which administers and coordinates standardization and conformity assessment system in the US.

## **COBOL**

COBOL was initially developed by the mathematician, Grace Hopper, in 1959. It was later modified in 1959-60 by a committee of the conference on Data Systems Languages (DODASYL). This committee defined the language specifications for COBOL.

COBOL was standardized by ANSI in 1968. Revised versions of this standard were published in 1974 and 1985, which are known as COBOL 74 and COBOL 85, respectively. Currently, the latest version of COBOL is known as COBOL 2002. COBOL was designed to solve business data processing problems.

## **Pascal**

Pascal was standardized by ANSI in 1971 by Professor Nicklaus Wirth of the Federal Institute of Technology in Zurich, Switzerland. He named it after the famous mathematician, Blasé Pascal.

Pascal was designed to help beginners to learn good problem solving and programming practices. Today, Pascal is recognized as a language that allows programmers to write structured and modular programs. It is suitable for scientific as well as business applications.

Pascal was standardized by ANSI in 1983. Since then, different versions of Pascal are also available.

## **C Language**

C was developed in 1972 by Dennis Ritchie and Brian Kernighan at AT& T's Bell Laboratories, USA. The C was designed in such a way that it has features of a high-level language along with efficiency of low-level language.

C was standardized by ANSI in 1989. C is a compiler-based language. Therefore, C programs can be easily transferred to other computers equipped with a C compiler. It is widely used for writing word-processing programs, compilers and operating systems. For example, UNIX operating system is written in C.

### **1.4 Programming Paradigms**

The processes involved in programming are usually accompanied by different school of thoughts called programming paradigms. They are the alternative approaches to programming processes and they also tell us how best we can program a computer. The most common important programming paradigms are procedural paradigm and the object-oriented paradigm

#### **Procedural Programming**

In Procedural programming, a list of instructions is used to tell the computer what to do step-by-step. Procedural programming relies on procedures, also called subroutines or routines. A procedure contains a series of computational steps to be carried out. Procedural programming is also referred to as imperative programming. Examples of procedural languages include FORTRAN, COBOL and C. If you want a computer to do something, you should provide step-by-step instructions on how to do it.

#### **Object-Oriented Programming (OOP)**

Object-oriented programming is another approach to problem-solving where all computations are carried out using objects. An **object** is a component of a program that knows how to perform certain actions and how to interact with other elements of the program. Objects are the basic units of object-oriented programming. A simple example of an object would be a person. Logically, you would expect a person to have a name. This would be considered a property of the person. You would also expect a person to be able to do something, such as

walking. This would be considered a method of the person. Examples of object-oriented programming are C++, Visual Basic etc.

## Unit 2: Algorithms

### 1.0 Introduction

Remember, you have learnt in the above unit that a programming language is designed to allow human and computer to communicate in order to solve a problem. However, before writing a problem's solution in programming language, the solution to the problem needs to be designed using a tool or technique generally known as *Algorithm*. Therefore, this unit will introduce you to the concept of problem-solving methods in relation to Algorithm. The unit will also tell you the properties of a good algorithm, various forms of representing, particularly pseudo-code method of representing algorithm.

### 1.1. Definition of Algorithm

An *algorithm* is a sequence of steps required to accomplish a task. The term algorithm has no root meaning. According to the *American Heritage Dictionary*, it is derived from the name of Muhammad ibn Musa al-Khwarizmi (780A.D-850A.D), a Muslim mathematician whose works introduced Arabic numerals and Algebraic concepts to Western mathematics. The word "algebra" stems from the title of his book *Kitab al jabrwa'l-muq'abala*. In your daily and professional activities, you also perform a sequence of steps to accomplish any given task. For example, to watch a football match in a cinema, you need to first pay for your ticket and then watch the football in the cinema. You cannot directly enter the cinema hall without buying the ticket.

The *concept* meaning of algorithm has come to be accepted as: set of step-by-step instructions for solving a well-defined problem which takes allowable inputs to produce desired output. These definitions require that an algorithm must consist of:

1. Set of steps that can be clearly followed (unambiguous), typically by a computer when implemented in computer language.
2. Executable means that the steps are doable
3. Terminating process means that the process must lead to an end
4. Algorithm takes acceptable data as input.
5. It produces desired output based on the input provided. If wrong input is provided, algorithm will not give correct output.

The concept of algorithm is depicted in figure 2.1 starting from Problem definition; algorithm transformed to computer program that computer executes acceptable inputs from user to give desired output.

The concept in figure 2.1 assumes the algorithm is implemented in a programming Language. One important concept in algorithm is that the problem solver must write explicitly all the instructions required to solve the problem without assumptions.

In fact, if a problem solver cannot express the concept clearly, the algorithm will not be useful to solve the problem.

Consider a problem to find the largest of three numbers. Almost anyone can immediately tell which is the largest, but many cannot explain the steps they followed to arrive at it. Most people will exclaim, "I just know it! However, an algorithm approach desires the steps to be clearly and unambiguously stated. This is what separates algorithm from ordinary human intuition.

The algorithm is stated below.

Step 1 Enter the three numbers say  $x, y, z$ .

Step 2 If  $x > y$  then go to step 3 else go to step 4

Step 3 is  $x > z$  then  $x$  is largest go to step 5 else  $z$  is largest go to step 5

Step 4 is  $y > z$  then  $y$  is the largest.

Step 5 stop

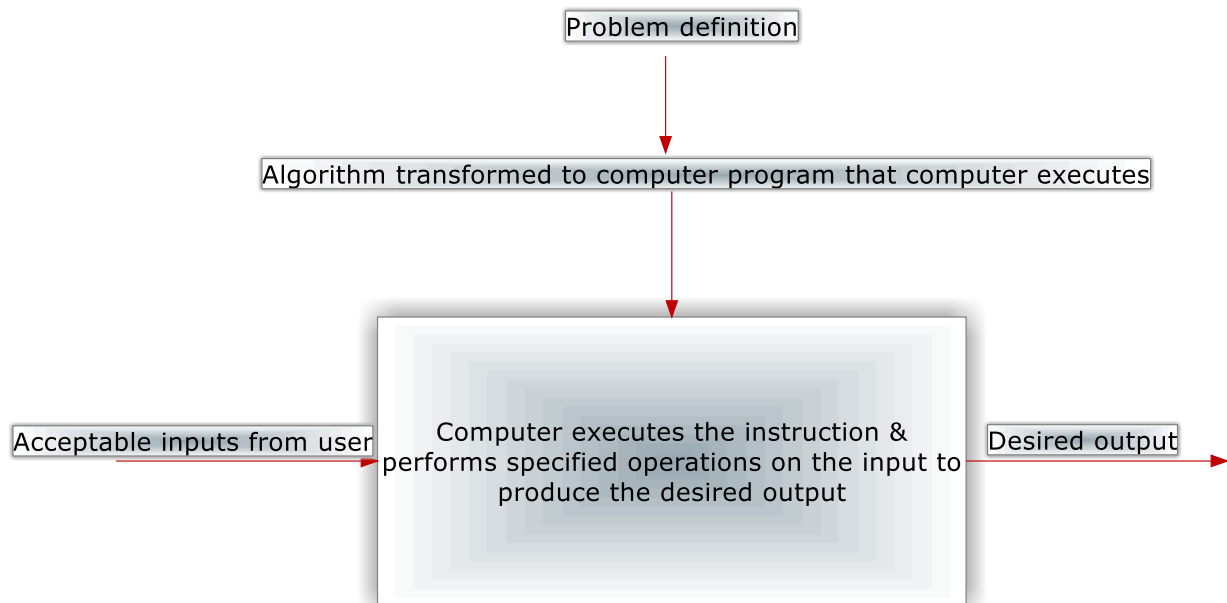


Figure 2.1: concept of algorithm

## 1.2 Characteristics of an Algorithm

1. An algorithm ends after a fixed number of steps
2. Each step in an algorithm clearly specifies the action to be performed
3. The steps in an algorithm specify basic operations. These operations could include mathematical calculations, input/output functions, and logical comparisons
4. The algorithm should accept input data, in a defined format, before it can be processed by the given instructions
5. An algorithm generates one or more outputs after the input is processed. The resultant information termed as output can be displayed or stored for future reference.

## 1.3 Representing an Algorithm

Algorithms can be represented in different ways, using flowcharts or pseudo codes.

**Flowcharts:** flowcharts are graphical representation of algorithms

**Pseudo-codes** represent algorithms by simple English language.

### Pseudo-Code

Pseudo-code represents an algorithm in the English language. It is used as an alternative approach to a flowchart. Pseudo-code uses simple statements that are written in a sequential manner.



Instead of using graphics or table to represent an algorithm, you can also use simple structure which consists of simple statements called pseudo-code. After you develop pseudo-code, you can now write it in a programming language. You can easily detect errors when pseudo-code.

The following are a number of key words used in pseudo-code.

- Begin..... end: these are used to start and finish pseudo-code. Begin is the first line and end is the last line of a pseudo-code.
- Accept: this keyword is used to receive an input from a user.
- Display: This keyword is used to present a result or an input.
- If ..... Else: These keywords are used in decision making.`

### **Examples of representing algorithm using pseudo-code**

1. The following pseudo-code accepts two numbers, calculates the sum of the numbers and display the result:

```
Begin
    Accept the first number
    Accept the second number
    Calculate the sum of the first number and second number
    Display
End
```

2: The following pseudo-code accepts two numbers and displays the greater number:

```
Begin:
    Accept the second number
    Accept the second number
    If    the first number is greater than the second number
        display the " the first number is greater"
    else
        display "the second number is greater"
end
```

**Other examples of algorithms we interact with in our day to day activities include but not limited to the following;**

- i. A cooking recipe booklet
- ii. The rules of how to play a game
- iii. DVD player instructions

- iv. Description of a martial arts technique
- v. Directional maps for driving from point A to B
- vi. A car repair manual

## **Unit 3: Flowcharts**

### **1.0 Introduction**

Before going into writing computer programs to solve a problem, it is highly necessary for the programmer to do an initial design by using proper tools for such a design. To avoid unnecessary errors in the programming, the programs should be expressed in an understandable form. Generally, programmers and analysts make use of charts and shapes as very important tools. In this unit, you will be introduced to the use of flowcharts for writing programming languages.

### **1.1 Definition of Flowchart**

A flowchart can be defined as a pictorial representation to solve a problem. In other words, it is a graphical representation of an algorithm for solving a problem. As you learnt in the above unit that for you to write a program to solve a problem, you need to express the steps of solving such a problem step-by-step called an algorithm. However, flowchart is used to represent the algorithm in which the steps of processes are shown in various kinds of boxes that are orderly connected with arrows. Flowchart makes it easy for you to understand the process of solving a problem. It makes use of standard symbols use to represent different processes. The most common symbols as approved by the American National Standards Institute (ANSI) are as shown in table 3.1 of section 3.2.

## 1.2 Flowchart Symbols and Their Meanings

The most common symbols used in flowcharting are shown in table 3.1

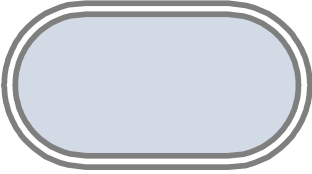


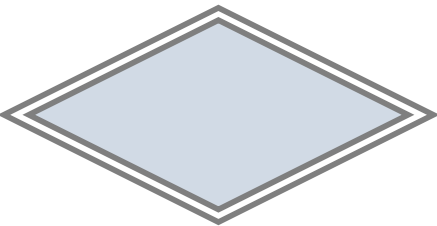
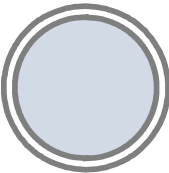


SYMBOLS	NAME	MEANING
	Oval	Terminal
	Rectangle	Process
	Parallelogram	Input/output
	Diamond	Decision
	Circle	Connector
	Arrow	Flow arrow
		Display

Table 3.1: common flowchart symbols and meaning

### Definition of Symbols

**Terminal:** This is used to indicate start and end of the chart

**Process:** This is the block that contain computational procedures and analyses

**Input/output:** This is used when input is read in and output is produced

**Decision:** This is the block where conditions that result in yes or no are put in

**Connector:** This is used to join segments of flowchart or where multiple arrow flows converge in a single point

**Arrow Flow:** This is used to indicate the sequence in which the steps described by the flowchart should be carried out.

**Display:** This is used to show the output of an algorithm

### 1.3 Applications of Flowchart

This section will introduce you to some simple examples of solving problems using flowcharts approach. You will first learn how to sum two numbers in a flowchart.

#### Flowchart for computing sum of two numbers

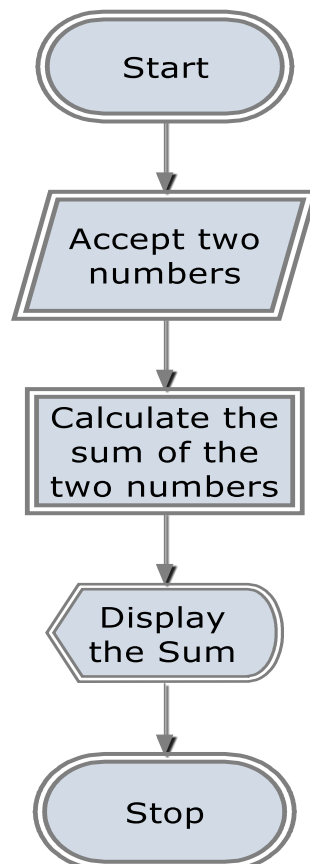


Figure 3.1 Flowchart that displays sum of two numbers

**Flowchart to find the greater of two numbers**

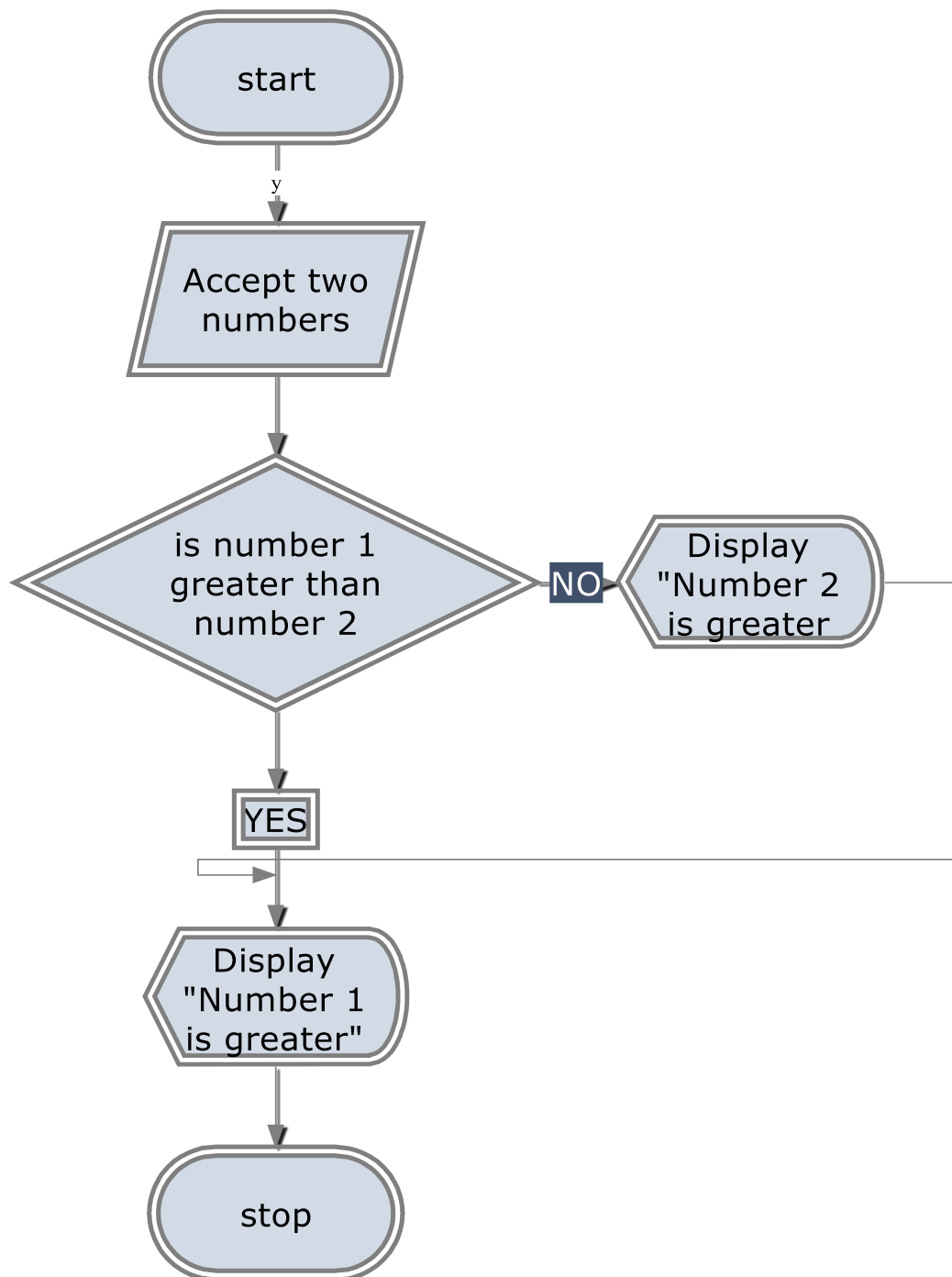


Figure 3.2: Flowchart to display the greater of two numbers

**Flowchart for computing the average of two numbers**

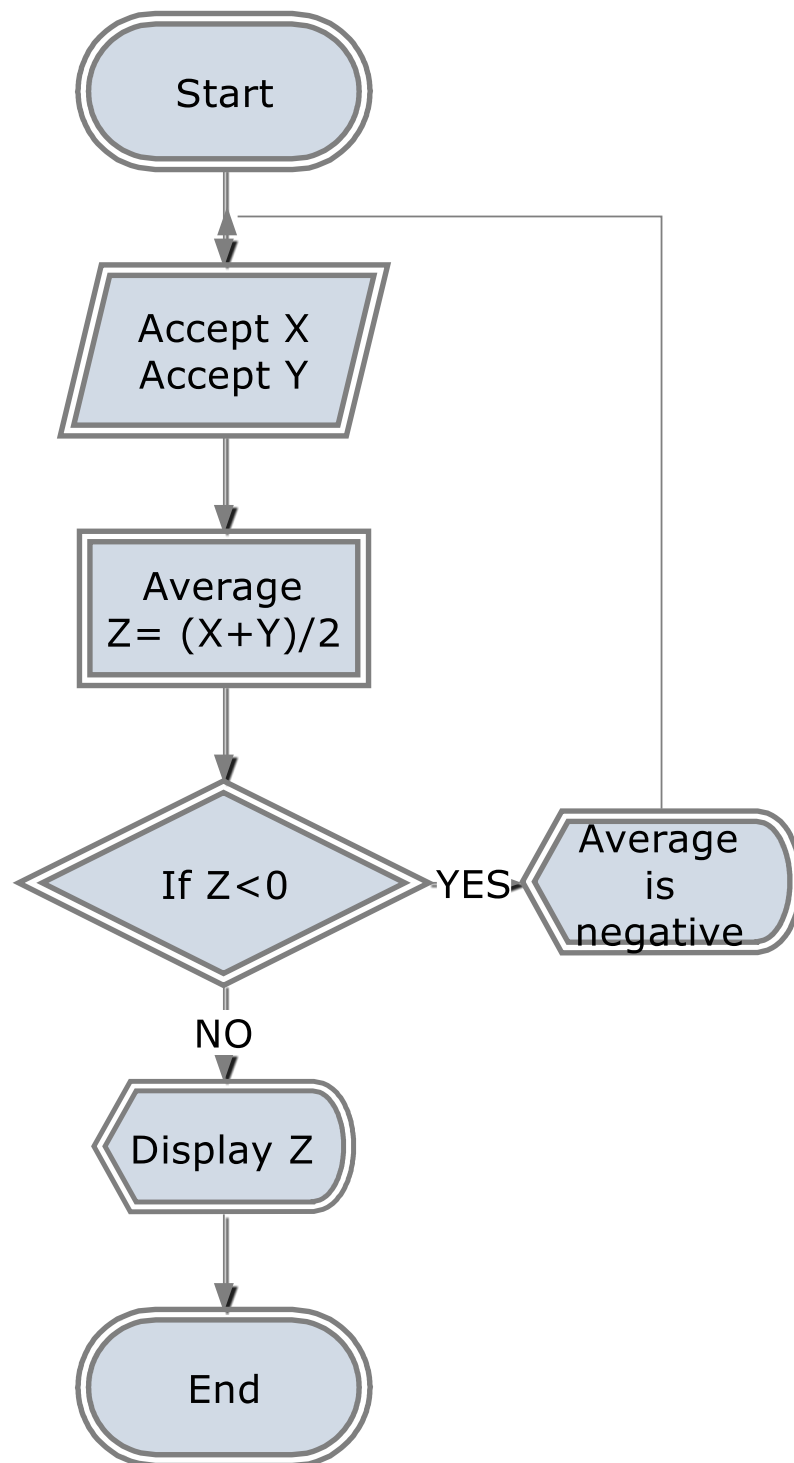


Figure 3.3: Flowchart that displays the average of two numbers

#### 1.4 Relationship between Pseudo-Code and Flowchart

The following table shows the pseudo-code keywords with corresponding flowchart symbols


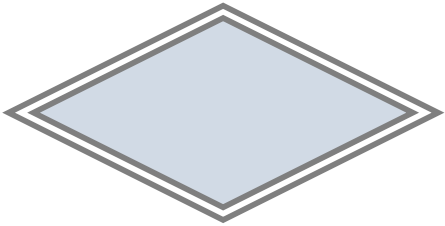
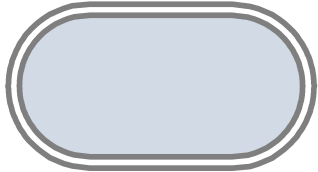
Pseudo-code Keywords	Flowchart Symbols
Accept	
If ..... else	
Begin .... End	

Table 3.2: Pseudo-code keywords with corresponding flowchart symbols

#### References and Further Reading

J. Glenn Brookshear “*Computer Science: An Overview*” Ninth Edition (2007)

NIIT Student Guide: Fundamentals of Programming, Mastermind Series

AnanyLevitin(2007) *Introduction-to-the-Design-Analysis-of-Algorithms: 2nd-Edition,*  
Pearson Addison Wesley.

I. S. Shehu “*CPT 223 Lecture note (2014): Introduction to Algorithms*”



# **Module6**

## **The Internet**

Unit 1: Introduction to the Internet

Unit 2: Concept of Web Technology

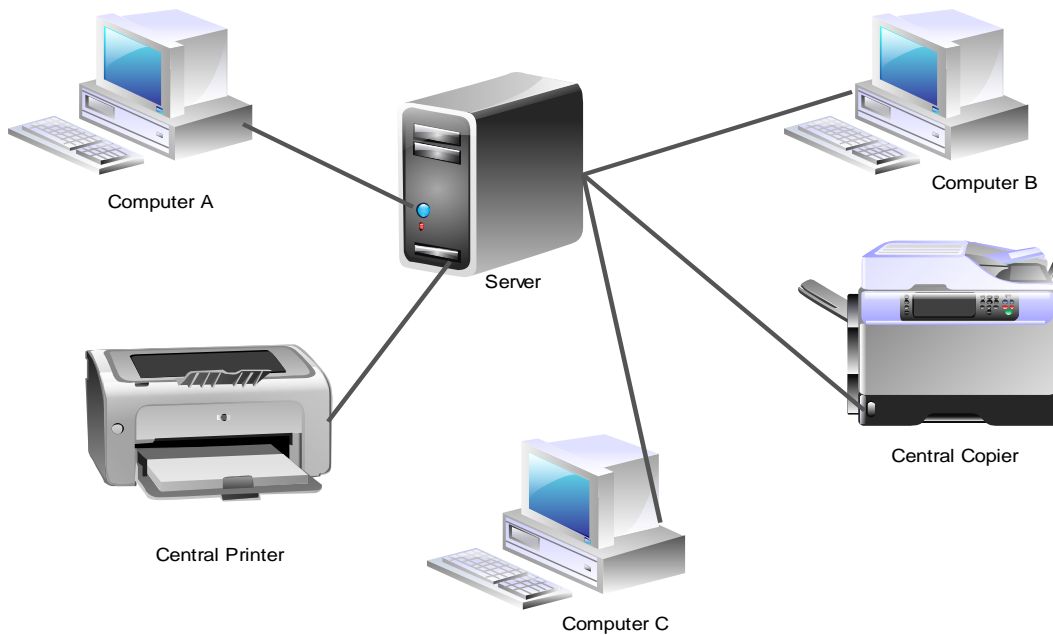
## **Unit 1: Introduction to the Internet**

### **1.0 Introduction**

Imagine being the desk officer of FUT CODEL and you need to monitor the activities of other representative of the unit in various part of the country so as to know the details of the number of people who came to inquire about the activities of the unit, those who show interest in one activity or the other and the number of people who have officially registered as students in a program undertaken by the unit, by the various representatives of the unit. A possible solution is for the desk officer to visit the various representatives of the units at their various locations so as to obtain that information which may be tasking, tedious and time consuming. An alternative solution is to connect the various representatives' computers at different locations together through a network which allows them to communicate, share resources and information. This will facilitate you (the desk officer) to have access to all those information from your seat enhancing your productivity. Wow! Imagine how great that would be. You are worried that what is a network and how can it be established. You will find out in a short while.

### **1.1 What is a Network?**

A network is a group of computers connected together--wired or wireless--by some special protocols so that they can communicate, share resources and enhance productivity. In a typical network, there are usually one or more dedicated computers that work as servers in storing, processing and routing data/information to other computers in the network. I.e. it provides services to other computers or clients in the network. Figure 1 illustrates the structure and interconnection of computers to form a network. It comprises of three computers named computer A, B and C respectively which are connected together via wired to a central server so as to communicate, exchange file, video, voice and data and also shared resources like the central copier and printer which any computer in the network can send documents to print.



Figure

1: An Illustration of a Typical Network

## 1.2 Types of Networks

Networks can be classified into four (4) main types based on their geographical coverage areas, equipment installation cost, as well as support tools and cost. The four types of networks are:

- i. **Local Area Network:** this is a network that comprises of different computers, printers, scanners and other devices connected together in a single building to enable sharing of resources, files voice and videos. Example of such network is the small network of computers, printers and other resources found in the second floor of Old SICT building of FUT,Minna,GidanKwano campus which houses the head office of the CODEL Unit. It is usually abbreviated as LAN and required relatively inexpensive devices and equipments to set up and maintain
- ii. **Campus Area Network:** this is a network that connects multiple local area networks. This type of network is larger than the LANs but smaller than the metropolitan area network. Example of such network is the type of network found in a university (FUT,Minna) which connects many LANs found in various schools and units together. It is abbreviated as CANs and requires more sophisticated equipment and cost to set up and maintain when compared to LAN.

- iii. **Metropolitan Area Network:** this is abbreviated as MAN, it is a high speed network that is larger than CAN and connects many LANs in a large and densely populated urban area to the internet.
- iv. **Wide Area Network:** this is a type of network that connects many LANs over very large group of cities through the internet or over the public switched telephone network. The MANs and WANs required more sophisticated networking equipment and cost to set-up and maintain. In addition, it requires on-site administrator and support staff.

Having discussed what a network is and the different types of network, I know you kept wondering that what are the importance or benefits of this so called networks and what an internet means. You need not worry, sit back and continue to read through this unit as you will find out soon yourself.

### **1.3 Importance of Networking**

- i. Facilitate data resource sharing: with the advent of computer networks, it makes information sharing between computers users on the network easy. Hence, making working together easy for network users and enhancing personal and collective productivity.
- ii. Enable users to share hardware resources: networking computers enables users to share hardware resources such as printers, copier and scanners. All users within the network can access these resources.
- iii. Sharing a high-speed Internet connection: all computers connected in a network can share high speed internet in communicating with other computers within the network and also connect to other outside networks via the internet.
- iv. Networks serve as a communication and support tools: computer networks serve as a communication tool in exchanging files and other information via the network. An example of such is the electronic mail (e-mail)

### **1.4 What is the Internet?**

The internet can simply be defined as a network of networks using a common set of rules that govern the transfer of information and communication over the network known as Internet protocol (IP). The internet give you access to numerous information, files, voice and videos all over the world. A good illustration of the importance of the internet is you accessing this course material titled "Introduction to Computer" written for CODEL online irrespective of your location or receiving lecture online just by logging in to the internet and accessing the unit website wherein you then log in with your username and password.

### **1.5 History of Networking and the Internet**

The origin of networking concept was dated as far back as 1962, when the concept of galactic network was proposed for social interaction by J.C.R Licklider of MIT who was also the first head of research computer program at Defense Advanced Research Projects Agency (DARPA), though then known as the Advanced Research Projects Agency (ARPA). The first major landmark towards networking computer was done in 1965 when computers were made to talk together by connecting a TX-2 computer in Mass to Q-32 in California which marks the first ever wide area network built using a telephone line but with a low speed dial-up.

He proceeded to further explore this network concept by proposing the ARPANET in 1967 which, theoretically, connections would not be lost even in the failure or absence of one or more of the network components. In 1969, ARPANET has metamorphosed as a network of four computer making it the first ever internet built by the US department of defense. However, ARPANET first public demonstration took place at international conference on computer communications in 1972.

The US internet supported about 11,000 servers in 1987 when they joined their Canadian counterpart. This figure increased drastically to about 200,000 servers by the end of 1989. Most countries' individual networks were linked together to form one worldwide network of networks by early 1990s and over 1million servers were on the internet as at the end of the year 1992. As of today no one can predict the exact figure of how many computers and network users are on the internet that reflects how large the internet has become. However, it was estimated that over one (1) billion users and 500 million computers are on the internet. The internet today serves as the largest avenue for information sharing, business, entertainment and social networking.

## **Unit 2: Concept of Web Technology**

### **1.0 Introduction**

Isn't it pretty cool? Having access to all kinds of information, be it data, voice, video and so on, at the snap of your fingers, via the internet. Imagine how tasking it would be for you if you have to come down all the way to CODEL at FUT,Minna to receive your lecture weekly irrespective of your location, or your daily schedules. Its hectic, right!? Thank God for the internet that is a network of networks that helps interconnect many computers around the World together. This averts the stress of having to travel tens or hundreds of kilometer down to received lecture at the main office. This was made possible by the interconnections of computers and other devices in such a way that they can communicate and share resources thereby forming what is known as a web through which you can access all lecture notes via the internet. I know your worry is "how would that be possible?" How the computer at CODEL office would know that I am requesting for a particular study guide, how would it transmit the study guide over the network to my computer? How would my computer retrieve it over the network right? Relax you will find out soon and lots more at the end of this unit.

### **1.1 Concept of Web Technologies**

Web technology can simply be described as the mechanism of interconnecting computers and other computer devices in such a way that enable it to share resources and communicate together over a network. This mechanism allows message to be send from a source to a receiver and facilitating the receiver to be able to decipher the message received and send feedback or acknowledgement back to the sender. In viewing internet resources, you need a browser which enables you to read or access information on the internet network. There are several types of browsers among which include the Internet Explorer, Google Chrome, Mozilla Firefox, Opera, and Netscape. This browser hooks-up to the internet via Modem or Internet Service Providers (ISPs), to a machine remotely requesting for documents and formatting such document retrieve to a format that can be view on the internet. In achieving this, the browser used the hyper text transfer protocols (HTTP) which is a special language designed for that purpose. It is a connection oriented network in the sense that an HTTP server is being run by the remote machine that contains the document. When a request is received by the server, it processes it and sends it back to the computer (sender) for viewing via the browser. The browser locates the corresponding server to send a request to, using the

associated uniform resource locator (URL) which help locate each document available on the web. The web document can take any form but the universally acceptable standard is the hyper text markup language (HTML) as discuss in the next section.

## **1.2 Hyper Text Markup Language (HTML) Protocol**

The first web page was created in November 1990 with little or no acceptable standards to follow. A group known as the World Wide Web Consortium (W3C) was then formed who were shouldered with the responsibility of developing a standard to be used for HTML in order to ensure uniformity of the protocol around the World. HTML is a language that is generally acceptable for encoding World Wide Web documents. It enables the browser to interpret and display the content of the documents retrieved from the web browsers. Furthermore, the HTML can be used in creating formatted text which remains in the same format when retrieved by a computer. HTML is made up of texts which show the content of the web page and tags that show the visual aspect and layout of the webpage. The most widely acceptable recent version of the HTML version is the HTML 4.0 which comprise of a cascading style sheets that enable repeated style features to be specify using an HTML Tags. Basically, you need a simple-text editor such as Notepad in coding your HTML so as to enable you make your own web page just like the one you view when you log on to CODEL Website.

## **1.3 Format of a Web Page**

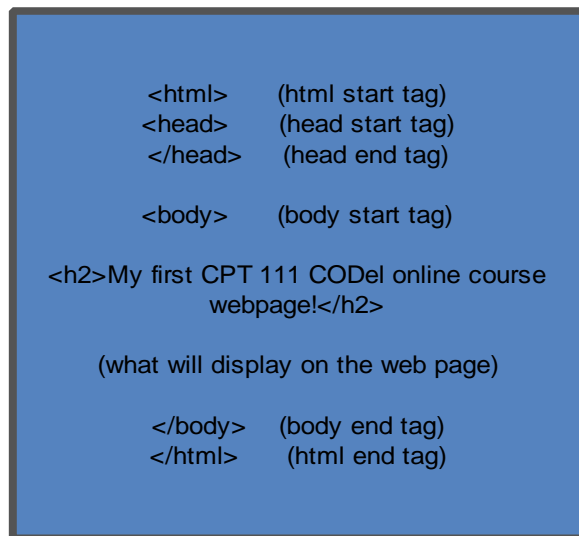
Having discussed the concept of HTML in section 2.2.2, this section will focus more on the structure of the HTML web page. All HTML page begins with a tag known as start tag and end with an end tag of html element respectively. Usually an angle bracket is used to enclose the HTML element “<>” these tags enable the computers to know that the document requested or being view is an HTML document. All other elements is then fit inside the start and end html tags.

The webpage can be classified further into two main classes namely, the Head and Body respectively as illustrated below in the HTML web page code showing My first CPT 111 CODEL online course webpage with an explanation of each line of codes done in a wall bracket. The Head contains instructions on how the web page will run its title and category while the body component of the HTML webpage contain all that will be displayed by the



web browser when logging on to the webpage after creation. All visible texts are placed outside any angle brackets

You can type in the html code shown in figure (2) in a notepad, then save it using all files option format. Having done that attempt to view your web page by opening a web browser and go to file, open and select browse so as to enable you navigate to the location you saved your file. When you find the file double click it... Wow! You have just viewed your first web page.



```
<html>      (html start tag)
<head>      (head start tag)
</head>     (head end tag)

<body>      (body start tag)

<h2>My first CPT 111 CODEl online course
        webpage!</h2>

        (what will display on the web page)

</body>     (body end tag)
</html>     (html end tag)
```

## 1.4 Support Tools for Website Creation

There are numerous support tools for website creation most of which are available freely online. Some of these tools will be highlighted as detailed explanation on how to use them in creating your web page will be discussed in other courses at higher level.

### Tools for Website Creation

- i. **BugHerd:** this is a tool that is used by web developer in handling and organizing feedback, fixes bug as well as feature requests.
- ii. **Fontello:** it helps in customizing the web page fonts
- iii. **Proto.io:** this is very useful in including animations to your webpage and also enables sharing such animations and commenting
- iv. **Foundation 3:** this serve as a template that can be used by web developer when creating their own web page

- v. Brackets assists web developer to cater for repetitive tasks such as searching a function, reloading a browser and so on, incorporated into the development process.

Other website creation tools include Dreamweaver CS6, Cloud9 IDE, Adobe Edge Inspect, Trello, Gridset, Microsoft WebMatrix 2, Firefox 18, Blogger, Google Sites, Jimdo, Kafafa, Wikispaces and so on

### **References for Further Reading**

Computer Networking, A Top-Down Approach 6<sup>th</sup> Edition by Kurose and Ross 2006.

Duyne, Douglas K. Van, James Landay, and Jason I. Hong. The design of sites: patterns, principles, and processes for crafting a customer-centered Web experience. Addison-Wesley Longman Publishing Co., Inc., 2002.

Bonk, Curtis J. The world is open: How web technology is revolutionizing education. John Wiley & Sons, 2009.

Hodgkins, R. (n.d.). Understanding web technology. Retrieved June 3, 2015, from <http://www.computerweekly.com/feature/Understanding-web-technology>