Higher Secondary Course

Computer Science

Class - XI

Part - I

Government of Kerala
DEPARTMENT OF EDUCATION

State Council of Educational Research and Training (SCERT); Kerala
2016

THE NATIONAL ANTHEM

Jana-gana-mana adhinayaka, jaya he
Bharatha-bhagya-vidhata.
Punjab-Sindh-Gujarat-Maratha
Dravida-Utkala-Banga
Vindhya-Himachala-Yamuna-Ganga
Uchchala-Jaladhi-taranga
Tava subha name jage,
Tava subha asisa mage,
Gahe tava jaya gatha.
Jana-gana-mangala-dayaka jaya he
Bharatha-bhagya-vidhata.
Jaya he, jaya he, jaya he,
Jaya jaya jaya, jaya he!

PLEDGE

India is my country. All Indians are my brothers and sisters.

I love my country, and I am proud of its rich and varied heritage. I shall always strive to be worthy of it.

I shall give respect to my parents, teachers and all elders and treat everyone with courtesy.

I pledge my devotion to my country and my people. In their well-being and prosperity alone lies my happiness.

Prepared by :

State Council of Educational Research and Training (SCERT)
Poojappura, Thiruvananthapuram 695012, Kerala

Website : www.scertkerala.gov.in *e-mail* : scertkerala@gmail.com
Phone : 0471 - 2341883, Fax : 0471 - 2341869
Typesetting and Layout : SCERT

Dear students,

Computer Science, a subject belonging to the discipline of Science and of utmost contemporary relevance, needs continuous updating. The Higher Secondary Computer Science syllabus has been revised with a view to bringing out its real spirit and dimension. The constant and remarkable developments in the field of computing as well as the endless opportunities of research in the field of Computer Science and Technology have been included.

This textbook is prepared strictly in accordance with the revised syllabus for the academic year 2014 - 15. It begins with the historical developments in computing and familiarises the learner with the latest technological advancements in the field of computer hardware, software and network. The advancement in computer network, Internet technology, wireless and mobile communication are also dealt with extensively in the content. In addition to familiarising various services over the Internet, the need to be concerned about the factors that harness morality and the awareness to refrain from cyber security threats are also highlighted.

The major part of the textbook as well as the syllabus establishes a strong foundation to construct and enhance the problem solving and programming skills of the learner. The multi-paradigm programming language C++ is presented to develop programs which enable computers to manage different real life applications effectively. The concepts and constructs of the principles of programming are introduced in such a way that the learner can grasp the logic and implementation methods easily.

I hope this book will meet all the requirements for stepping to levels of higher education in Computer Science and pave your way to the peak of success.

Wish you all success.

Dr P. A. Fathima
Director, SCERT; Kerala

Textbook Development Team
COMPUTER SCIENCE

Joy John
HSST, St. Joseph's HSS
Thiruvananthapuram

Asees V.
HSST, GHSS Velliyode, Kozhikode

Roy John
HSST, St. Aloysius HSS
Elthuruth, Thrissur

Aboobacker P.
HSST, Govt. GHSS Chalappuram,
Kozhikode

Shajan Jos N.
HSST, St. Joseph's HSS, Pavaratty,
Thrissur

Afsal K. A.
HSST, GHSS Sivapuram,
Kariyathankare P.O., Kozhikode

Prasanth P. M.
HSST, St. Joseph's Boys' HSS,
Kozhikode

Vinod V.
HSST, NSS HSS, Prakkulam, Kollam

Rajamohan C.
HSST, Nava Mukunda HSS,
Thirunavaya,Malappuram

A. S. Ismael
HSST, Govt. HSS Palapetty,
Malappuram

Sunil Kariyatan
HSST, Govt. Brennen HSS,
Thalassery

Sai Prakash S.
HSST, St. Thomas HSS,
Poonthura, Thiruvananthapuram

Experts

Dr Lajish V. L.
Assistant Professor, Dept. of Computer Science, University of Calicut

Dr Madhu S. Nair
Assistant Professor, Dept. of Computer Science, University of Kerala

Madhu V. T.
Director, Computer Centre, University of Calicut

Dr Binu P. Chacko
Associate Professor, Dept. of Computer Science,
Prajyoti Niketan College, Pudukad

Dr Sushil Kumar R.
Associate Professor, Dept. of English, D.B. College, Sasthamcotta

Dr Vineeth K. Paleri
Professor, Dept. of Computer Science and Engineering, NIT, Kozhikode

Maheswaran Nair V.
Sub Divisional Engineer, Regional Telecom Training Centre, Thiruvananthapuram

Artist

Sudheer Y. **Vineeth V.**

Academic Co-ordinator

Dr Meena S.
Research Officer, SCERT

Jancy Rani A. K.
Research Officer, SCERT

Contents

Icons used in this textbook


Let us do


Check yourself


Information box


Lab activities


Learning outcomes

1

Key concepts

- **Computing milestones and machine evolution**
  - Counting and the evolution of the positional number system
  - Evolution of the computing machine
- **Generations of computers**
  - First generation computers
  - Second generation computers
  - Third generation computers
  - Fourth generation computers
  - Fifth generation computers
- **Evolution of computing**
  - Programming languages
  - Algorithm and computer programs
  - Theory of computing

The Discipline of Computing

Computers now play a major role in almost every aspect of life and influence our lives in one way or the other. Today, almost everyone is a computer user and many are computer programmers. Getting computers to do what you want them to do requires intensive hands-on experience. But computer science can be seen on a higher level, as a science of problem-solving. Computer scientists must be able to analyse problems and design solutions for real world problems. The Computer Science discipline covers a wide range of topics from theoretical aspects like design of algorithms to more practical aspects like application development and its implementation. The discipline of Computer Science involves the systematic study of algorithmic processes – their theory, analysis, design, efficiency, implementation and application – that describe and transform information.

The concept of computing has evolved from the abacus to super computers of today. This chapter discusses the evolution of computing devices and provides an overview of the different generations of computers. The evolution of programming languages and the contributions of Alan Turing to computer science are also discussed.

1.1 Computing milestones and machine evolution

In ancient times people used stones for counting. They made scratches on walls or tied knots in ropes to record information. Progressively many attempts had been made to replace these manual computing techniques with faster computing machines. In this section, we will have a look at the ancient methods of counting and the evolution of the positional number system.

1.1.1 Counting and the evolution of the positional number system

The idea of number and the process of counting goes back far before history began to be recorded. It is believed that even the earliest humans had some sense of 'more' or 'less'. As human beings differentiated into tribes and groups, it became necessary to be able to know the number of members in the group and in the enemy's camp. And it was important for them to know if the flock of sheep or other animals was increasing or decreasing in size. In order to count items, such as animals, 'sticks' were used, each stick representing one animal or object.

Let us now see how the number system evolved. It is important to note that the system that we use everyday is a product of thousands of years of progress and development. It represents contributions of many civilisations and cultures. The number system is a method in which we represent numbers. The chronological development of the number system throughout the history is discussed below.

To begin with let us see the Egyptian number system that emerged around 3000BC. It used 10 as a radix (base). They had unique symbols for 1 to 9, 10 to 90, 100 to 900 and 1000 to 9000. As the Egyptians write from right to left, the largest power of ten appears to the right of the other numerals.

Later on, the era of Sumerian/Babylonian number system began. It used 60 as its number base, known as the sexagesimal system. Numerals were written from left to right. It was the largest base that people ever used in number systems. They did not use any symbol for zero, but they used the idea of zero. When they wanted to express zero, they just left a blank space within the number they were writing.

The Chinese number system emerged around in 2500 BC. It was the simplest and the most efficient number system. The Chinese had numbers from 1 to 9. It had the base 10, very similar to the one we use today. They used small bamboo rods to represent the numbers 1 to 9.

Approximately in 500 BC, the Greek number system known as Ionian number system evolved. It was a decimal number system and the Greeks also did not have any symbol for zero.

The Romans started using mathematics for more practical purposes, such as in the construction of roads, bridges, etc. They used 7 letters (I, V, X, L, C, D and M) of the alphabet for representing numbers.

The Mayans used a number system with base 20. There is a simple logic behind this base 20. It is the sum of the number of fingers and toes. This number system could produce very accurate astronomical observations and make measurements with greater accuracy.

The Hindu-Arabic numeral system actually originated in India, around 1500 years ago. It was a positional decimal numeral system and had a symbol for zero. This invention can indeed be termed as one of India's greatest contributions to the world. Later on many of the countries adopted this numeral system. Now let us discuss the evolution of computing machines.

1.1.2 Evolution of the computing machine

During the period from 3000 BC to 1450 AD, human beings started communicating and sharing information with the aid of simple drawings and later through writings. The introduction of numbers led to the invention of Abacus, the first computing machine. In the following section, we will examine some important milestones in the evolution of computing machines.

a. Abacus

Abacus was discovered by the Mesopotamians around 3000 BC. The word 'abacus' means calculating board. An abacus consisted of beads on movable rods divided into two parts. The abacus may be considered the first computer for basic arithmetical calculations. An abacus is shown in Figure 1.1.

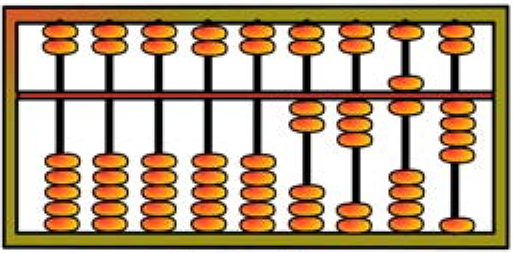


Fig. 1.1 : Abacus

The abacus is also called a counting frame, a calculating tool for performing arithmetic operations. The Chinese improved abacus as a frame holding vertical wires, with seven beads on each wire. A horizontal divider separates the top two beads from the bottom five. Addition and multiplication of numbers was done using the place value of digits and position of beads in an abacus.

The abacus works on the basis of the place value system. Reading it is almost like reading a written numeral. Each of the five beads below the bar has a value of 1. Each of the two beads above the bar has a value of 5. The beads which are pushed

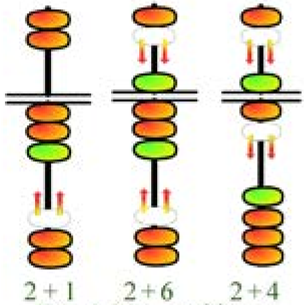


Fig. 1.2(a) : Addition using Abacus

against the bar represent the number. The number on the abacus given in Figure 1.1 is 2364.

Abacus is used even today by children to learn counting. A skilled abacus operation can be as fast as a hand held calculator. Figures 1.2(a) shows the addition of two single digit numbers. Figure 1.2(b) shows how two numbers (54 and 46) are added.

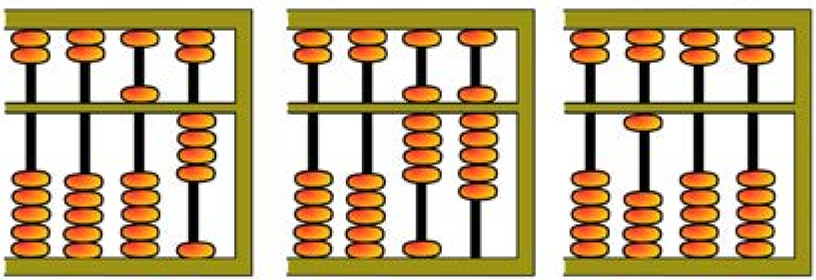


Fig. 1.2(b) : Addition using Abacus

b. Napier's bones

John Napier was a mathematician who devised a set of numbering rods known as Napier's bones in 1617 AD, by which a multiplication problem could be easily performed. These numbered rods could perform multiplication of any number by a number in the range of 2-9. There are 10 bones corresponding to the digits 0-9 and a special eleventh bone that is used to represent the multiplier. This device was known as Napier's bones. John Napier also invented logarithm in 1614, that reduced tedious multi-digit multiplications to addition problems. A representation of Napier's bones is given in Figure 1.3.

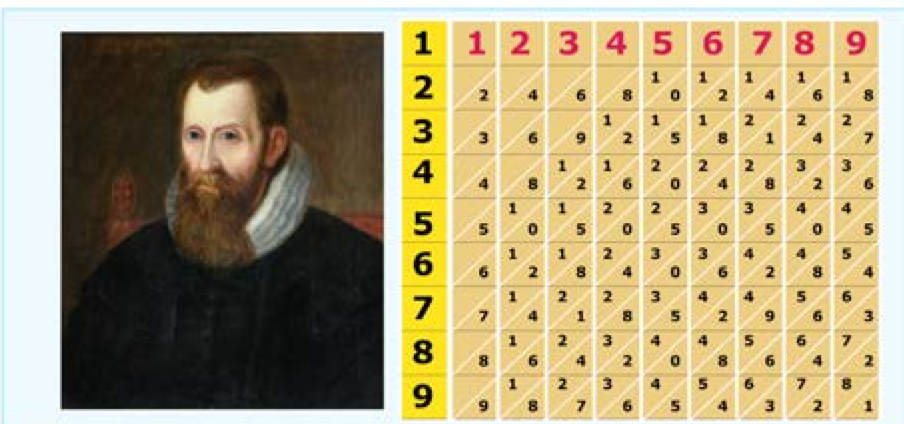


Fig. 1.3 : John Napier (1550 – 1617) and Napier's Bones

The strips of Napier's bones are the times tables. Each square gives 2 × number, 3 x number and so on, but the tens and units are written above and below a slanting line respectively. Napier's bones is good for multiplying a long number by a single digit number. Let us multiply 425928 by 7. First take the strips for 4, 2, 5, 9, 2 and 8, and fit them into the frame. Look at the squares next to the 7 on the side. It is coloured green in Figure 1.4. Now read the digits – any number within slanting lines must be added. So the answer is 2(8+1)(4+3)(5+6)(3+1)(4+5) 6 or 297(11)496. All the digits except 11 are in their position. 11 needs to have 10 carried to the left. This makes 29(7+1)1496 or 2981496, which is the correct answer.

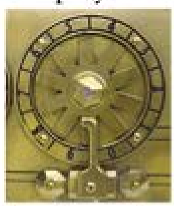
Fig. 1.4 : Multiplication using Napier's Bones

c. Pascaline

Blaise Pascal was a French mathematician and one of the first modern scientists to develop a calculator. In 1642, at the age of 19, he developed a computing machine that was capable of adding and subtracting two numbers directly and that could multiply and divide by repetition. Pascal invented this arithmetic calculator to assist his father in his work as a tax collection supervisor. This machine was operated by dialing a series of wheels, gears and cylinders. He called it 'Pascaline'. Figure 1.5 shows a Pascaline.

Fig. 1.5 : Blaise Pascal (1623 - 1662) and Pascaline

Consider adding the numbers 20 and 81 using Pascaline. Initially, the Pascaline will be set to 0 for all the six digits. To dial 20, you just have to put your finger into the space between the spokes next to digit 2 of the second wheel and rotate the wheel in clockwise direction until your finger strikes against the fixed stop on the bottom of the wheel. This rotation transmits the value of two into the second window from the right. Now the machines will display number 0020.



Pascaline wheel

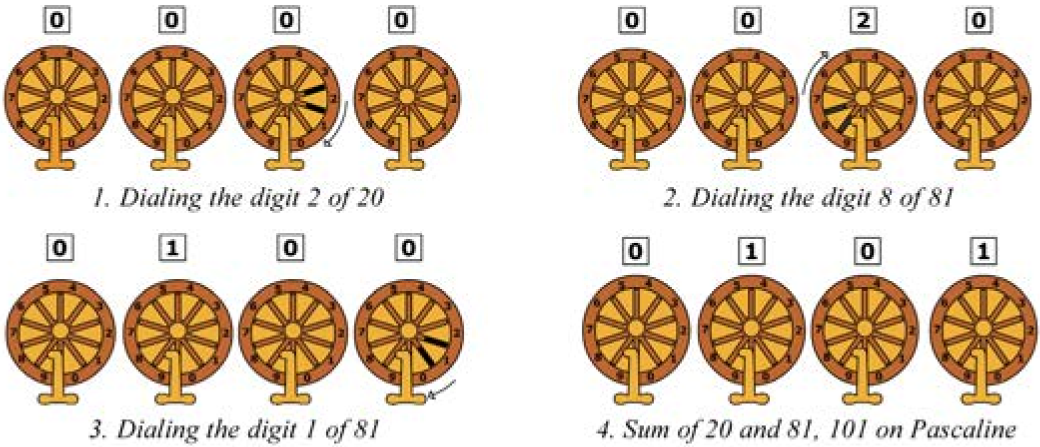


Fig. 1.6 : Adding using Pascaline

To dial 81, put your finger into the space between the spokes next to digit 8 of the second wheel and rotate it. After the second drum reaches number 9, the gears inside Pascaline will carry to the next drum one unit and the third drum of the machine will rotate by one tenth. So after the end of dialing number 8, the machine will display the number 100. Now put your finger into the space between the spokes next to digit 1 and rotate it in the same way you did before. Now the machine will display the number 0101, which is the final result of addition.

d. Leibniz's calculator

In 1673 the German mathematician-philosopher Gottfried Wilhelm von Leibniz designed a calculating machine called the Step Reckoner. The Step Reckoner expanded on Pascal's ideas and extended the capabilities to perform multiplication and division as well. Leibniz successfully introduced this calculator onto the market. His unique, drum-shaped gears formed the basis of many successful calculator designs in later years.

Fig. 1.7 : Gottfried Wilhelm von Leibniz (1646 - 1716) and Leibniz Calculator

e. Jacquard's loom

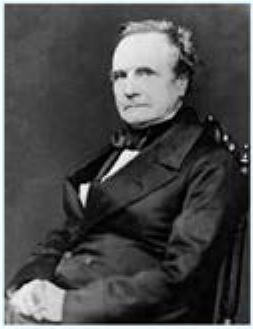
The Jacquard loom is a mechanical loom, invented by Joseph Marie Jacquard in 1801, that simplifies the process of manufacturing textiles with complex patterns. The loom is controlled by punched cards with punched holes, each row of which corresponds to one row of the design. Multiple rows of holes are punched on each card and the many cards that compose the design of the textile are joined together in order. The Jacquard loom not only reduced the amount of human labour, but also allowed to store patterns on cards to be utilised again to create the same product. These punched cards were innovative because the cards had the capability to store information on them. This ability to store information triggered the computer revolution. The punched card concept was adopted by Charles Babbage to control his Analytical Engine and later by Herman Hollerith.

Fig. 1.8 : Joseph Marie Jacquard (1752 - 1834) and Jacquard's Loom

f. Difference engine

The first step towards the creation of computers was made by a mathematics professor, Charles Babbage. He dreamed of removing the human element from the calculations. He realised that all mathematical calculations can be broken up into simple operations which are constantly repeated and these operations could be carried out by an automatic machine. Charles Babbage started working on a

Difference Engine that could perform arithmetic calculations and print results automatically. In 1822, Babbage invented the Difference Engine to compile mathematical tables. On completing it, he conceived the idea of a better machine that could perform not just one mathematical task but any kind of calculation.



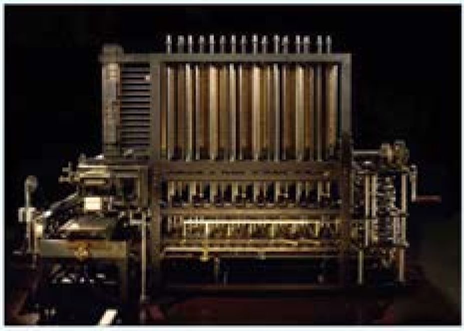


Fig. 1.9 : Charles Babbage (1791 - 1871) and Difference Engine

g. Analytical engine

In 1833, Charles Babbage started designing the Analytical Engine – the real predecessor of the modern day computer. Analytical Engine marks the development from arithmetic calculation to general-purpose computation. The Analytical Engine has many features found in the modern digital computer. The Engine had a 'Store' (memory) where numbers and intermediate results could be stored, and a separate 'Mill' (processor) where arithmetic processing could be performed. Its input/output devices were in the form of punched cards containing instructions. These instructions were written by Babbage's assistant, Agusta Ada King, the first programmer in the world. Owing to the lack of technology at that time, the Analytical Engine was never built, but Babbage established the basic principles on which today's modern computers work. Charles Babbage's great inventions – the Difference Engine and the Analytical Engine earned Charles Babbage the title 'Father of Computer'.

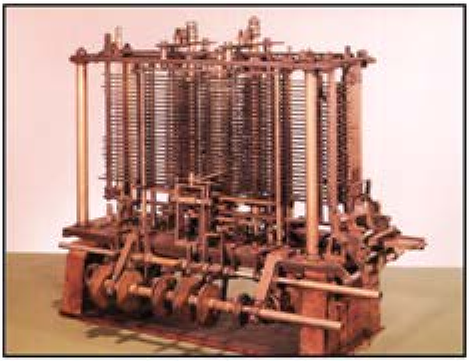


Fig. 1.10 : A model of Analytical Engine

h. Hollerith's machine

In 1887, an American named Herman Hollerith fabricated the first electromechanical punched card tabulator that used punched cards for input, output and instructions.

The card had holes on them in a particular pattern, having special meaning for each kind of data. In 1880's, the US Census Bureau had huge amounts of data to tabulate. It would take at least ten years to analyse population statistics manually. Herman Hollerith's greatest breakthrough was his use of electricity to read, count and sort punched cards whose holes represented data. His machines were able to accomplish the task in one year. In 1896, Hollerith started the Tabulating Machine Corporation which after a series of mergers, became International Business Machines (IBM) Corporation in 1924.


Fig. 1.11 : Herman Hollerith (1860 - 1929) and Hollerith Census Machine

i. Mark - I

In 1944, Howard Aiken, in collaboration with engineers at IBM, constructed a large automatic electromechanical computer. Aiken's machine, called the Harvard Mark I, based on Babbage's Analytical Engine, handled 23-decimal-place numbers and could perform all four arithmetic operations. It was preprogrammed to handle logarithms and trigonometric functions. Using Mark I, two numbers could be added in three to six seconds. For input and output, it used paper-tape readers, card readers, card punch and typewriters.

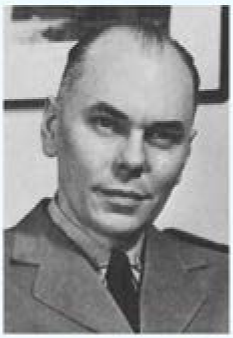



Fig. 1.12 : Howard Aiken (1900 - 1973) and Mark I computer

Check yourself

1. The Sumerian Number System is also known as _______.
2. What are the features of Hindu Arabic Number system?
3. How is the zero represented in the Babylonian Number System?
4. Who is the first programmer in the world?
5. The computing machine developed by Balaise Pascal is known as ______.

1.2 Generations of computers

The evolution of computer started from the 16^{th} century, resulting in today's modern machines. It is distinguished into five generations of computers from the first programmable computer to the ones based on artificial intelligence. Each generation of computers is characterised by a major technological development that fundamentally changed the way computers operate, resulting smaller, cheaper, more powerful, more efficient and reliable computing devices. Based on various stages of development, computers can be divided into different generations. They are:

- First Generation Computers (1940 – 1956)
- Second Generation Computers (1956 – 1963)
- Third Generation Computers (1964 – 1971)
- Fourth Generation Computers (1971 – Present)
- Fifth Generation Computers (Present and beyond)

1.2.1 First generation computers (1940 – 1956)

The first generation computers were built using vacuum tubes. This generation implemented the stored program concept. A vacuum tube is a device controlling electric current through a vacuum in a sealed container. This cylindrical shaped container is made of thin transparent glass. The input was based on punched cards and paper tapes and output was displayed on printouts.

The first general purpose programmable electronic computer, the Electronic Numerical Integrator and Calculator (ENIAC) belongs to this generation. ENIAC was built by J. Presper Eckert and John V. Mauchly. The ENIAC was 30-50 feet long, weighed 30 tons, contained 18,000 vacuum tubes, 70,000 registers, 10,000 capacitors and required 1,50,000 watts of electricity. First generation computers

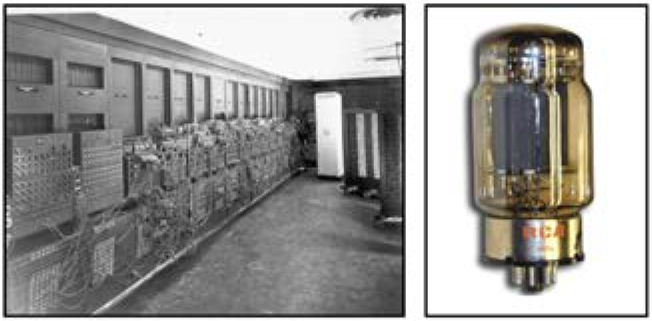


Fig. 1.13 : ENIAC and Vaccum tube

were too bulky in size, required a large room for installation and used to emit large amount of heat. Consequently, air-conditioner was a must for the proper working of computers.

Before ENIAC was completed, Von Neumann designed the Electronic Discrete Variable Automatic Computer (EDVAC) with a memory to hold both stored program as well as data. Eckert and Mauchly later developed the first commercially successful computer, the Universal Automatic Computer (UNIVAC), in 1952.

Von Neumann architecture

The mathematician John Von Neumann conceived a computer architecture which forms the core of nearly every computer system in use today. This architecture known as Von Neumann architecture consists of a central processing unit (CPU) containing arithmetic logic unit (ALU) and control unit (CU), input-output unit and a memory for storing data and instructions. This model implements the 'Stored Program Concept' in which the data and the instructions are stored in the memory.

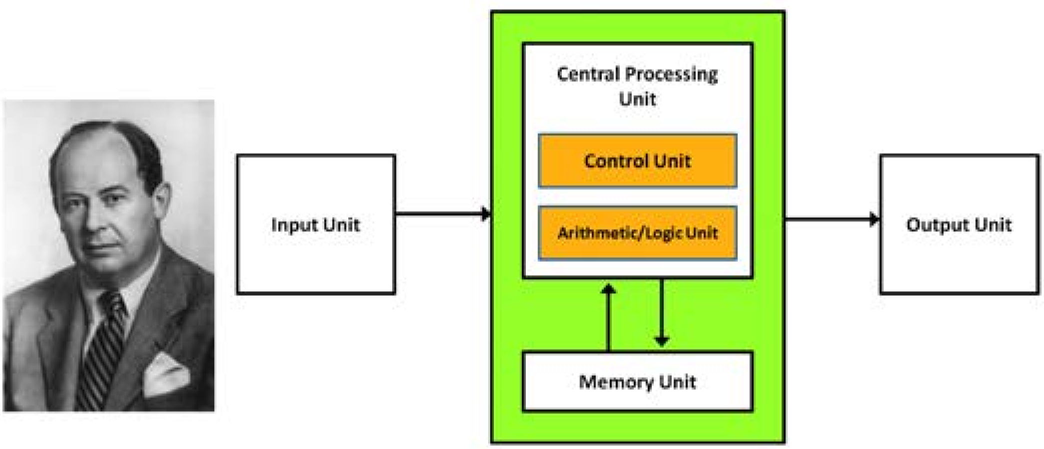
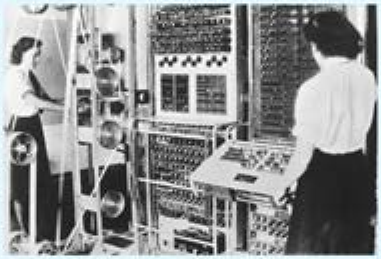


Fig. 1.14 : John Von Neumann (1903 - 1957) and Von Neumann architecture



In 1943, the British developed a secret code-breaking computer called Colossus to decode German messages. It was designed using vaccum tubes by the engineer Tommy Flowers. The Colossus's impact on the development of the computer industry was rather limited for two important reasons. First, Colossus was not a general-purpose computer; it was only designed to decode secret messages. Second, the existence of the machine was kept secret until 1970s. This deprived most of those involved with Colossus of credit for their pioneering advancements in electronic digital computing.

1.2.2 Second generation computers (1956 – 1963)

In second generation computers, vacuum tubes were replaced by transistors. It was developed at Bell Laboratories by John Bardeen, Walter Brattain and William Shockley in 1947. Replacing vacuum tubes with transistors, allowed computers to become smaller and more powerful and faster. They also became less expensive, required less electricity and emitted less heat. The manufacturing cost was also less.

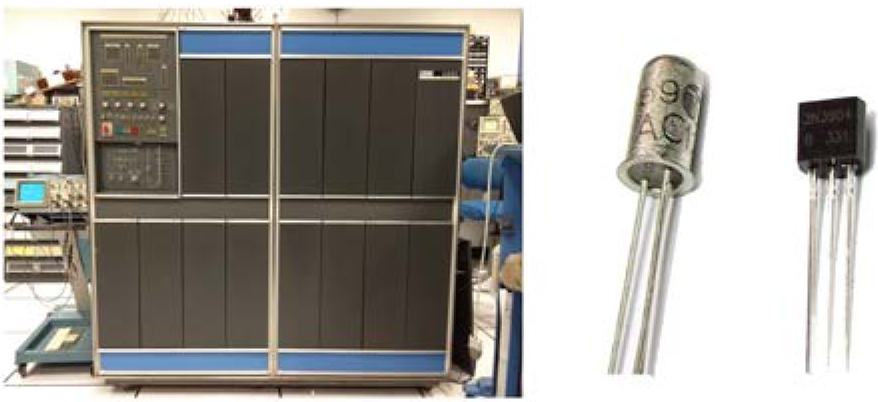


Fig. 1.15 : IBM 1401and Transistors

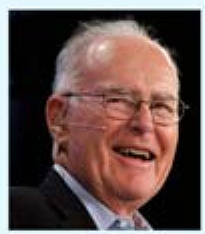
It is in the second generation that the concept of programming language was developed. This generation used magnetic core memory and magnetic disk memory for primary and secondary storage respectively. Second-generation computers moved from cryptic binary machine language to symbolic or assembly languages. During second generation, many high level programming languages like FORTRAN and COBOL were introduced that allowed programmers to specify instructions in English like words. IBM 1401 and IBM 1620 are popular computers in this generation.

1.2.3 Third generation computers (1964 – 1971)

Third generation computers are smaller in size due to the use of integrated circuits (IC's). IC's or silicon chips that contained miniaturised transistors were developed by Jack Kilby, an engineer with Texas Instruments. IC drastically reduced the size and increased the speed and efficiency of computing. Multilayered printed circuits were developed and core memory was replaced by faster, solid state memories with large capacity.

Fig. 1.16 : IBM 360 and Integrated Circuit

This generation of computers had better processing speed, consumed less power and was less costly. Integrated circuits, improved secondary storage devices and new input/output devices like keyboards and monitors were introduced. Arithmetic and logical operations were performed in microseconds or even nanoseconds. These computers could run many different programs at one time with a central program that monitored the memory. The high level language BASIC which made programming easy was developed during this period. Computers for the first time became accessible to a mass audience because they were smaller and cheaper than their predecessors. Some computers in this generation are IBM 360 and IBM 370.



Moore's Law states that the number of transistors on integrated circuits doubles approximately every two years. The law is named after Gordon E Moore, who described the trend in 1965. He noted that the number of components in IC had doubled every year from 1958 to 1965. He predicted that the trend would continue 'for at least ten years'. His prediction has proven to be accurate. Although this trend continued for more than half a century, Moore's Law is considered only as an observation and not a physical or natural law.

1.2.4 Fourth generation computers (1971 onwards)

The computers that we use today belong to this generation. These computers use microprocessors and are called microcomputers. Microprocessor is a single chip which contains Large Scale of Integration (LSI) of electronic components like transistors, capacitors, resistors, etc. Due to the development of microprocessor, it is possible to place computer's Central Processing Unit (CPU) on a single chip. Because of microprocessors, the fourth generation includes more data processing capacity than third generation computers. Later LSI circuits were replaced by Very Large Scale Integrated (VLSI) circuits which further increased the scale of integration. The fourth generation computers are smaller in size and have faster accessing and processing speeds.

The computer which occupied a very large room in earlier days could now fit in a palm. These computers were interconnected to form computer networks, which eventually led to the development of the Internet. As computers became less costly and more user-friendly, large number of people began buying them for personal use. Some computers in this generation are IBM PC and Apple II.

Fig. 1.17 : VLSI Chip

1.2.5 Fifth generation computers (future)

Fifth generation computers are based on Artificial Intelligence (AI). AI is the ability to simulate human intelligence. Such intelligent systems are still in the development stage, though there are some applications, such as speech recognition, face recognition and robotic vision and movement that are already available.

AI is the branch of computer science concerned with developing computer programs (intelligent systems) for solving complex problems (which are normally done by human beings without any effort) by the application of process that are analogues to human reasoning process. The two most common AI programming languages are LISP and Prolog. The fifth-generation computing also aims at developing computing machines that respond to natural language input and are capable of learning and self-organisation.

Table 1.1 shows comparative features of five generations of computers.

The first processor Intel 4004 was developed in 1971 by Intel Corporation and consisted of 2,300 transistors integrated into a single IC. Some popular microprocessors and the number of transistors integrated in them are given below.

Processor	Transistor Count
Intel 8086	29,000
Motorola 68000 (used in Apple)	68,000
Intel Pentium	31,00,000
AMD K7	2,20,00,000
Core i7	73,10,00,000

Criteria	Generation				
	First	Second	Third	Fourth	Fifth
Technology	Vacuum Tube	Transistor	Integrated Circuit	Microprocessor	Artificial Intelligence
Operating System	None	None	Yes	Yes	Yes
Language	Machine	Assembly	High Level	High Level	High Level
Period	1940-1956	1956-1963	1964-1971	1971-Present	Present and Yet to come

Table 1.1 : Comparative features of five generations of computers

Check yourself

1. UNIVAC belongs to ______ generation..
2. What is meant by stored program concept?
3. Say True or False "Computers can understand only machine languages".
4. First generation computers are characterised by the use of ______.
5. What is the major technological advancement in the fourth generation computers?

1.3 Evolution of computing

Computing machines are used for processing, storing, and displaying information. The processing is done according to the instructions given to it. Early computers built during 1940's were only capable of performing series of single tasks, like a calculator. They were special-purpose systems programmed by rows of mechanical switches or by jumper wires on plug. The implementation of branching/looping statements, subroutine calls, etc. was not possible or was difficult. Later computers solved this problem by implementing John Von Neumann's revolutionary innovation the 'Stored Program Concept', which suggested storing data and programs in memory. The set of detailed instructions given to computer for executing a specific task is called a program.

Agusta Ada Lowelace

Augusta Ada King, Countess of Lowelace commonly known as Ada Lowelace, was an English mathematician and writer known for her work on Charles Babbage's early mechanical general-purpose computer, the Analytical Engine. Her notes on the engine included the first algorithm intended to be carried out by a machine. Because of this, she is often described as the world's first computer programmer.

Fig.1.18 : Agusta Ada Lowelace (1815 - 1852)

1.3.1 Programming languages

A programming language is an artificial language designed to communicate instructions to a computer. Programming languages can be used to create programs that control the behavior of a machine and/or to express algorithms.

The first programming language developed for use in computers was called machine language. Machine language consisted of strings of the binary digits 0 and 1. Introduction of this language improved the overall speed and efficiency of the programming process. This language had many drawbacks like difficulty in finding and rectifying programming errors and its machine dependency. The programmer also needed to have a good knowledge of the computer architecture.

To make programming easier, a new language with instructions consisting of English-like words instead of 0's and 1's, was developed. This language was called assembly language. Electronic Delay Storage Automatic Calculator (EDSAC) built during 1949 was the first to use assembly language. Although this made writing programs easier, it had limitations. It is specific to a given machine and the programs written in this language are not transferable from one machine to another.

This led to the development of new languages called high level languages which are machine independent and which used simple English-like words and statements. It allowed people having less knowledge of computer architecture to develop easy-to-understand programs. A-0 programming language developed by Rear Admiral Dr. Grace Hopper, in 1952, for UNIVAC-I computer is the first language of this type. FORTRAN developed by the team led by John Backus at IBM for IBM 704 machine and 'Lisp' developed by Tim Hart and Mike Levin at MIT are other examples.

Fig. 1.19 : Dr. Grace Hopper (1906-1992)

1.3.2 Algorithm and computer programs

A programmer cannot write the instruction to be followed by a computer, unless he/she knows how to solve the problem manually. In order to ensure that the program instructions are appropriate for solving the given problem, and are in the correct sequence, program instructions are to be planned before they are written. An effective tool for planning a computer program is an algorithm. An algorithm provides a step by step solution for a given problem. These steps can then be converted to machine instructions using a programming language.

1.3.3 Theory of computing

The theory of computation is the branch that deals with how efficiently problems can be solved based on computation models and related algorithms. In order to perform a rigorous study of computation, computer scientists work with a mathematical abstraction of computers called a model of computation. There are several models in use, but the most commonly examined is the Turing Machine named after the famous computer scientist Alan Turing.

a. Contribution of Alan Turing

Alan M. Turing (1912 - 1954) was a British mathematician, logician, cryptographer and computer scientist. He made significant contributions to the development of computer science, by presenting the concepts of algorithm and computing with the help of his invention the Turing Machine, which is considered as a theoretical model of a general purpose computer. In 1950's, Alan Turing proposed to consider the question, 'Can machines think?' and later it turns out to be the foundation for the studies related to the computing machinery and intelligence. Turing proposed an imitation game which is later modified to Turing test and it is considered to be the test for determining a machine's intelligence. Considering these contributions he is regarded as the Father of Modern Computer Science as well as Artificial Intelligence.

b. Turing machine

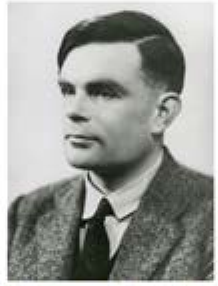


Fig. 1.20 : Alan Turing (1912 - 1954)

Turing machine is a model of a computer proposed by Alan Turing in 1936. It is conceived as an ideal model of 'computing'. Alan Turing reasoned that any computation that could be performed by a human involved writing down intermediate results, reading them back and carrying out actions that depend only on what has been read and the current state of things. Turing machine is a theoretical computing device that could print symbols on a paper tape in a manner that emulate a person following a series of logical instructions.

Fig. 1.21 : Turing machine

A Turing machine consists of an infinitely-long tape which acts like the memory in a computer. The cells on the tape are usually blank at the start and can be written with symbols. In this case, each cell can contain the symbols '0', '1' and ' ' (blank), and is thus said to be a 3-symbol Turing machine (refer Figure 1.22). At each step, the machine can read the symbol on the cell under the head, edit the symbol and move the tape left or right by one square so that the machine can read and edit the symbol in the neighbouring cells.

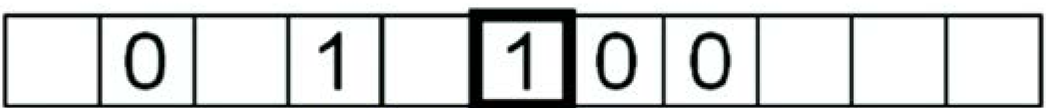


Fig. 1.22 : The head movement over tape

Any particular Turing machine is defined by a rule which specifies what the head should do at each step.The action of a Turing machine is determined by (a) the current state of the machine (b) the symbol in the cell currently being scanned by the head and (c) a table of transition rules, which serve as the 'program' for the machine.

In modern terms, the tape serves as the memory of the machine, while the read-write head is the memory bus through which data is accessed and updated by the machine. The initial arrangement of symbols of cells on the tape corresponds to the input given to the computer. The steps of the Turing machine correspond to the running of the computer. For a given input, each part of the rule specifies what 'operation' the machine should perform. Even though Turing machines are equivalent to modern electronic computers at a certain theoretical level, they differ in many details.

The Turing test

Alan Turing introduced the 'Turing test' in his paper titled 'Computing Machinery and Intelligence'. The 'Turing Test' involves a human interrogator and two contestants - a computer and a human. The interrogator converses with these contestants via computer terminals, without knowing the identity of the contestants. After a sufficiently long period of conversation, if the interrogator is unable to identify the computer, then the computer is said to have passed 'Turing test' and must be considered intelligent. Turing predicted that by 2000, computers would pass the test. There have been various programs that have demonstrated some amount of human like behaviour, but no computer has this far passed the Turing test.

Let us sum up

In this chapter, we briefly sketched the evolution of the number system and counting. We went through the development of computing machines and described the structure of the modern computing system. The evolution of computing was also discussed along with the different types of programming languages. The concepts of algorithms and computer programs were also discussed. The detailed description of generations of computers was also seen. Finally we discussed the theory of computation, in which the contributions of Alan Turing and the concept of Turing Machine were outlined.

Learning outcomes

After the completion of this chapter the learner will be able to

- categorize the basic concept of computing milestones in history.
- sketch the modern computing machine.

- recognise the impact of John Von Neumann's Architecture in today's world.
- identify the pioneers of Computer Science.
- list the characteristics of computers in each generation.
- explain the contribution of Alan Turing and the concept of Turing Machine.

Sample questions

Very short answer type

1. Which is the base of Mayan's Number System?
2. Greek Number System is known as ________.
3. Which was the first computer for basic arithmetic calculations?
4. Who invented logarithms?
5. What is the name of the machine developed by Blaise Pascal?
6. Who was the first programmer in the world?
7. Computing machines recognizes and operates in ___________ language.
8. What does EDVAC stand for?
9. Give the name for a simple kind of theoretical computing machine.

Short answer type

1. Discuss the developments of the number system from the Egyptian to the Chinese Era.
2. Discuss the impact of Hindu-Arabic numeral system in the world.
3. Compare the Roman Number system and Mayan's Number System.
4. Discuss the features of Abacus.
5. Compare the Analytical Engine and Difference Engine of Charles Babbage.
6. Bring out the significance of Hollerith's machine.
7. What are the developments in computing machines that took place during the Second World War?
8. Discuss the evolution of computer languages.
9. Discuss the working of Turing Machine.

Long answer type

1. List out and explain the various generations of Computers.
2. Prepare a seminar report on evolution of positional number system.
3. Discuss the various computing machines emerged till 1900's.

2

Key concepts

- **Number systems**
  - Decimal, binary, octal, hexadecimal
- **Number conversions**
- **Binary arithmetic**
  - Addition, subtraction
- **Data representation**
  - Representation of integers and floating point numbers
  - Character representation: ASCII, EBCDIC, ISCII, UNICODE
  - Representation of audio, image and video data
- **Introduction to Boolean algebra**
  - Logic operators and logic gates
- **Basic postulates**
- **Basic theorems**
- **Circuit designing for simple Boolean expressions**
- **Universal gates**

Data Representation and Boolean Algebra

Computer is a machine that can handle different types of data items. We feed data such as numbers, characters, images, videos and sounds to a computer for processing. We know that computer is an electronic device functioning on the basis of two electric states - ON and OFF. All electronic circuits have two states - open and closed. The two-state operation is called binary operation. Hence, the data given to computer should also be in binary form. In this chapter we will discuss various methods for representing data such as numbers, characters, images, videos and sounds.

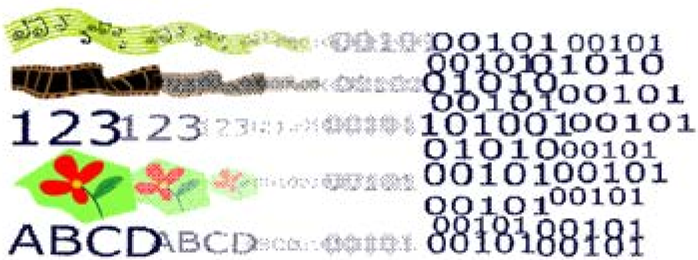


Fig. 2.1: External and internal form of data

Data representation is the method used internally to represent data in a computer.

Before discussing data representation of numbers, let us see what a number system is.

2.1 Number systems

A number is a mathematical object used to count, label and measure. A number system is a systematic way to represent numbers. The number system we use in our day to day life is the decimal number system that uses 10 symbols or digits. The number 289 is pronounced as two hundred and eighty nine and it consists of the symbols 2, 8 and 9. Similarly there are other number systems. Each has its own symbols and method for constructing a number. A number system has a unique base, which depends upon the number of symbols. The number of symbols used in a number system is called **base** or **radix** of a number system.

Let us discuss some of the number systems.

2.1.1 Decimal number system

The decimal number system involves ten symbols 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9 to form a number. Since there are ten symbols in this number system, its base is 10. Therefore, the decimal number system is also known as base-10 number system.

Consider two decimal numbers 743 and 347

743 → seven hundred + four tens+ three ones ($7\times10^2 + 4\times10^1 + 3\times10^0$)

347 → three hundreds + four tens + seven ones ($3\times10^2 + 4 \times10^1 + 7\times10^0$)

Here, place value (weight) of 7 in first number 743 is $10^2=100$. But weight of 7 in second number 347 is $10^0=1$. The weight of a digit depends on its relative position. Such a number system is known as ***positional number system.*** All positional number systems have a base and the place value of a digit is some power of this base.

Place value of each decimal digit is power of 10 (10^0, 10^1, 10^2, ...). Consider a decimal number 5876.

This number can be written in expanded form as

Weight	10^3	10^2	10^1	10^0
Decimal Number	5	8	7	6

$$\begin{aligned}
&= 5\times10^3 + 8\times10^2 + 7\times10^1 + 6\times10^0 \\
&= 5\times1000 + 8\times100 + 7\times10 + 6\times1 \\
&= 5000 + 800 + 70 + 6 \\
&= 5876
\end{aligned}$$

In the above example, the digit 5 has the maximum place value, $10^3=1000$ and 6 has the minimum place value, $10^0=1$. The digit with most weight is called Most Significant

Digit (MSD) and the digit with least weight is called Least Significant Digit (LSD). So in the above number MSD is 5 and LSD is 6.

Left most digit of a number is MSD and right most digit of a number is LSD

For fractional numbers weights are negative powers of 10 (10^{-1}, 10^{-2}, 10^{3}, …) for the digits to the right of decimal point. Consider another example 249.367

Weight	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}
Decimal Number	2	4	9	3	6	7

MSD (.) LSD

$= 2\times10^2 + 4\times10^1 + 9\times10^0 + 3\times10^{-1} + 6\times10^{-2} + 7\times10^{-3}$
$= 2\times100 + 4\times10 + 9\times1 + 3\times0.1 + 6\times0.01 + 7\times0.001$
$= 200 + 40 + 9 + 0.3 + 0.06 + 0.007$
$= 249.367$

So far we have discussed a number system which uses 10 symbols. Now let us see the construction of other number systems with different bases.

2.1.2 Binary number system

A number system which uses only two symbols 0 and 1 to form a number is called binary number system. Bi means two. Base of this number system is 2. So it is also called base-2 number system. We use the subscript 2 to indicate that the number is in binary.

e.g. $(1101)_2$, $(101010)_2$, $(1101.11)_2$

Each digit of a binary number is called bit. A **bit** stands for **binary digit**.

The binary number system is also a positional number system where place value of each binary digit is power of 2. Consider an example $(1101)_2$. This binary number can be written in expanded form as shown below:

Weight	2^3	2^2	2^1	2^0
Binary Number	1	1	0	1

MSB LSB

$= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$
$= 1\times8 + 1\times4 + 0\times2 + 1\times1$
$= 8 + 4 + 0 + 1$
$= 13$

The right most bit in a binary number is called Least significant Bit (**LSB**). The leftmost bit in a binary number is called Most significant Bit (**MSB**).

The binary number 1101 is equivalent to the decimal number 13. The number 1101 also exists in the decimal number system. But it is interpreted as one thousand one hundred and one. To avoid this confusion, base must be specified in all number systems other than decimal number system. The general format is

$$\textbf{(Number)}_{\textbf{base}}$$

This notation helps to differentiate numbers of different bases. So a binary number must be represented with base 2 as $(1101)_2$ and it is read as "one one zero one to the base two".

If no base is given in a number, it will be considered as decimal. In other words, specifying the base is not compulsory in decimal number.

For fractional numbers, weights are negative powers of 2 (2^{-1}, 2^{-2}, 2^{-3}, ...) for the digits to the right of the binary point. Consider an example $(111.011)_2$

Weight	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
Binary Number	1	1	1	0	1	1

MSB (.) LSB

$$= 1\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$$

$$= 1\times4 + 1\times2 + 1\times1 + 0\times\frac{1}{2} + 1\times\frac{1}{4} + 1\times\frac{1}{8}$$

$$= 4 + 2 + 1 + 0 + 0.25 + 0.125$$

$$= 7.375$$

Importance of binary numbers in computers

We have seen that binary number system is based on two digits 1 and 0. The electric state ON can be represented by 1 and the OFF state by 0 as in Figure 2.2. Because of this, computer uses binary number system as the basic number system for data representation.

Fig. 2.2 : Digital representation of ON and OFF

2.1.3 Octal number system

A number system which uses eight symbols 0, 1, 2, 3, 4, 5, 6 and 7 to form a number is called octal number system. Octa means eight, hence this number system is called octal. Base of this number system is 8 and hence it is also called base-8 number system. Consider an example $(236)_8$. Weight of each digit is power of 8 (8^0, 8^1, 8^2, 8^3, ...). The number $(236)_8$ can be written in expanded form as

Weight	8^2	8^1	8^0
Octal Number	2	3	6

$$= 2\times8^2 + 3\times8^1 + 6\times8^0$$
$$= 2\times64 + 3\times8 + 6\times1$$
$$= 128 + 24 + 6$$
$$= 158$$

For fractional numbers weights are negative powers of 8, i.e. (8^{-1}, 8^{-2}, 8^{-3}, …) for the digits to the right of the octal point. Consider an example $(172.4)_8$

Weight	8^2	8^1	8^0	8^{-1}
Octal Number	1	7	2	4

$$= 1\times8^2 + 7\times8^1 + 2\times8^0 + 4\times8^{-1}$$
$$= 64 + 56 + 2 + 4\times\frac{1}{8}$$
$$= 122 + 0.5$$
$$= 122.5$$

2.1.4 Hexadecimal number system

A number system which uses 16 symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F to form a number is called hexadecimal number system. Base of this number system is 16 as there are sixteen symbols in this number system. Hence this number system is also called base-16 number system.

In this system, the symbols A, B, C, D, E and F are used to represent the decimal numbers 10, 11, 12, 13, 14 and 15 respectively. The hexadecimal digit and their equivalent decimal numbers are shown below.

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

Consider a hexadecimal number $(12AF)_{16}$. Weights of each digit is power of 16 (16^0, 16^1, 16^2, ...).

This number can be written in expanded form as shown below:

Weight	16^3	16^2	16^1	16^0
Hexadecimal Number	1	2	A	F

$$= 1\times16^3 + 2\times16^2 + 10\times16^1 + 15\times16^0$$
$$= 1\times4096 + 2\times256 + 10\times16 + 15\times1$$
$$= 4096 + 512 + 160 + 15$$
$$= 4783$$

For fractional numbers, weights are some negative power of 16 (16^{-1}, 16^{-2}, 16^{-3}, ...) for the digits to the right of the hexadecimal point. Consider an example $(2D.4)_{16}$

Weight	16^1	16^0	16^{-1}
Hexadecimal	2	D	4

$$= 2\times16^1 + 13\times16^0 + 4\times\frac{1}{16}$$
$$= 32 + 13 + 0.25$$
$$= 45.25$$

Table 2.1 shows the base and symbols used in different number systems:

Number System	Base	Symbols used
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Table 2.1 : Number systems with base and symbols

Importance of octal and hexadecimal number systems

As we have discussed, digital hardware uses the binary number system for its operations and data. Representing numbers and operations in binary form requires too many bits and needs lot of effort. With octal, the bits are grouped in threes (because $2^3 = 8$) and with hexadecimal, the bits are grouped in four (because $2^4 = 16$) and these groups are replaced with the respective octal or hexadecimal symbol. This conversion processes of binary numbers to octal and hexadecimal number systems and vice versa are very easy. This short-hand notation is widely used in the design and operations of electronic circuits.

Check yourself

1. Number of symbols used in a number system is called ____.
2. Pick invalid numbers from the following
   i) $(10101)_8$　ii) $(123)_4$　iii) $(768)_8$　iv) $(ABC)_{16}$
3. Define the term 'bit'.
4. Find MSD in the decimal number 7854.25.
5. The base of hexadecimal number system is ________.

2.2 Number conversions

After having learnt the various number systems, let us now discuss how to convert the numbers of one base to the equivalent numbers in other bases. There are different types of number conversions like decimal to binary, binary to decimal, decimal to octal etc. This section discusses how to convert one number system to another.

2.2.1 Decimal to binary conversion

The method of converting decimal number to binary number is by repeated division. In this method the decimal number is successively divided by 2 and the remainders are recorded. The binary equivalent is obtained by grouping all the remainders, with the last remainder being the Most Significant Bit (MSB) and first remainder being the Least Significant Bit (LSB). In all these cases the remainders will be either 0 or 1 (binary digit).

Examples:

Find binary equivalent of decimal number 25.

		Remainders	
2	25		
2	12	1	↑ LSB
2	6	0	
2	3	0	
2	1	1	
	0	1	MSB

$(25)_{10} = (11001)_2$

Find binary equivalent of $(80)_{10}$.

		Remainders	
2	80		
2	40	0	↑ LSB
2	20	0	
2	10	0	
2	5	0	
2	2	1	
2	1	0	
	0	1	MSB

$(80)_{10} = (1010000)_2$

Hint: Binary equivalent of an odd decimal number ends with 1 and binary of even decimal number ends with zero.

Converting decimal fraction to binary

To convert a fractional decimal number to binary, we use the method of repeated multiplication by 2. At first the decimal fraction is multiplied by 2. The integer part of the answer will be the MSB of binary fraction. Again the fractional part of the answer is multiplied by 2 to obtain the next significant bit of binary fraction. The procedure is continued till the fractional part of product is zero or a desired precision is obtained.

Example: Convert 0.75 to binary.

	$0.75 \times 2 = 1.50$
1	$.50 \times 2 = 1.00$
1	$.00$

$(0.75)_{10} = (0.11)_2$

Example: Convert 0.625 to binary.

	$0.625 \times 2 = 1.25$
1	$.25 \times 2 = 0.50$
0	$.50 \times 2 = 1.00$
1	$.00$

$(0.625)_{10} = (0.101)_2$

Example: Convert 15.25 to binary.

Convert 15 to binary

		Remainders
2	15	
2	7	1
2	3	1
2	1	1
	0	1

Convert 0.25 to binary

	$0.25 \times 2 = 0.50$
0	$.50 \times 2 = 1.00$
1	$.00$

$(15.25)_{10} = (1111.01)_2$

2.2.2 Decimal to octal conversion

The method of converting decimal number to octal number is also by repeated division. In this method the number is successively divided by 8 and the remainders are recorded. The octal equivalent is obtained by grouping all the remainders, with the last remainder being the MSD and first remainder being the LSD. Remainders will be 0, 1, 2, 3, 4, 5, 6 or 7.

Example: Find octal equivalent of decimal number 125.

8	125	Remainders	
8	15	5	LSD
8	1	7	
	0	1	MSD

$(125)_{10} = (175)_8$

Example: Find octal equivalent of $(400)_{10}$.

8	400	Remainders
8	50	0
8	6	2
	0	6

$(400)_{10} = (620)_8$

2.2.3 Decimal to hexadecimal conversion

The method of converting decimal number to hexadecimal number is also by repeated division . In this method, the number is successively divided by 16 and the remainders are recorded. The hexadecimal equivalent is obtained by grouping all the remainders, with the last remainder being the Most Significant Digit (MSD) and first remainder being the Least Significant Digit(LSD). Remainders will be 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E or F.

Example: Find hexadecimal equivalent of decimal number 155.

16	155	Remainders	
16	9	11 (B)	→ LSD
	0	9	→ MSD

$(155)_{10} = (9B)_{16}$

Example: Find hexadecimal equivalent of 380.

16	380	Remainders
16	23	12 (C)
16	1	7
	0	1

$(380)_{10} = (17C)_{16}$

2.2.4 Binary to decimal conversion

A binary number can be converted into its decimal equivalent by summing up the product of each bit and its weight. Weights are some power of 2 ($2^0, 2^1, 2^2, 2^3, \ldots$).

Example: Convert $(11011)_2$ to decimal.

Weight	2^4	2^3	2^2	2^1	2^0
Bit	1	1	0	1	1

$$
\begin{aligned}
(11011)_2 &= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 \\
&= 16 + 8 + 2 + 1 \\
&= 27
\end{aligned}
$$

$(11011)_2 = (27)_{10}$

Example: Convert $(1100010)_2$ to decimal.

Weight	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Bit	1	1	0	0	0	1	0

$$
\begin{aligned}
(1100010)_2 &= 1\times2^6 + 1\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 \\
&= 64 + 32 + 2 \\
&= 98
\end{aligned}
$$

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

Table 2.2 may help us to find powers of 2.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1

Table 2.2 : Powers of 2

Converting binary fraction to decimal

A binary fraction number can be converted into its decimal equivalent by summing up the product of each bit and its weight. Weights of binary fractions are negative powers of 2 ($2^{-1}, 2^{-2}, 2^{-3}, \ldots$) for the digits after the binary point.

Example: Convert $(0.101)_2$ to decimal.

Weight	2^{-1}	2^{-2}	2^{-3}
Bit	1	0	1

$$
\begin{aligned}
(0.101)_2 &= 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} \\
&= 0.5 + 0 + 0.125 \\
&= 0.625
\end{aligned}
$$

$(0.101)_2 = (0.625)_{10}$

Example: Convert $(1010.11)_2$ to decimal.

Weight	2^3	2^2	2^1	2^0
Bit	1	0	1	0

$$
\begin{aligned}
(1010)_2 &= 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 \\
&= 8 + 0+2+0 \\
&= 10
\end{aligned}
$$

$(1010)_2 = (10)_{10}$

$(0.11)_2 = 1\times2^{-1} + 1\times2^{-2}$
$= 0.5 + 0.25$
$= 0.75$ $(0.11)_2 = (0.75)_{10}$

Weight	2^{-1}	2^{-2}
Bit	1	1

$(1010.11)_2 = (10.75)_{10}$

Table 2.3 shows some negative powers of 2.

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
0.5	0.25	0.125	0.0625	0.03125

Table 2.3 : Negative powers of 2

2.2.5 Octal to decimal conversion

An octal number can be converted into its decimal equivalent by summing up the product of each octal digit and its weight. Weights are some powers of 8 (8^0, 8^1, 8^2, 8^3, ...).

Example: Convert $(157)_8$ to decimal.

$(157)_8 = 1\times8^2 + 5\times8^1 + 7\times8^0$
$= 64 + 40 + 7$
$= 111$

Weight	8^2	8^1	8^0
Octal digit	1	5	7

$(157)_8 = (111)_{10}$

Example: Convert $(1005)_8$ to decimal.

$(1005)_8 = 1\times8^3 + 0\times8^2 + 0\times8^1 + 5\times8^0$
$= 512+5$
$= 517$

Weight	8^3	8^2	8^1	8^0
Octal digit	1	0	0	5

$(1005)_8 = (517)_{10}$

2.2.6 Hexadecimal to decimal conversion

An hexadecimal number can be converted into its decimal equivalent by summing up the product of each hexadecimal digit and its weight. Weights are powers of 16 (16^0, 16^1, 16^2, ...).

Example: Convert $(AB)_{16}$ to decimal.

$(AB)_{16} = 10\times16^1+11\times16^0$
$= 160 + 11$
$= 171$

Weight	16^1	16^0
Hexadecimal digit	A	B

A = 10 B = 11

$(AB)_{16} = (171)_{10}$

Example: Convert $(2D5)_{16}$ to decimal.

$(2D5)_{16} = 2\times16^2+13\times16^1+5\times16^0$

$= 512+208+5$

$= 725$

Weight	16^2	16^1	16^0
Hexadecimal digit	2	D	5

D = 13

$(2D5)_{16} = (725)_{10}$

2.2.7 Octal to binary conversion

An octal number can be converted into binary by converting each octal digit to its 3 bit binary equivalent. Eight possible octal digits and their binary equivalents are listed in Table 2.4.

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

Table 2.4 : Binary equivalent of octal digit

Example: Convert $(437)_8$ to binary.

3-bit binary equivalents of each octal digit are

$$\begin{array}{ccc} 4 & 3 & 7 \\ \downarrow & \downarrow & \downarrow \\ 100 & 011 & 111 \end{array}$$

$(437)_8=(100011111)_2$

Example: Convert $(7201)_8$ to binary.

3-bit binary equivalents of each octal digits are

$$\begin{array}{cccc} 7 & 2 & 0 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 111 & 010 & 000 & 001 \end{array}$$

$(7201)_8 = (111010000001)_2$

2.2.8 Hexadecimal to binary conversion

A hexadecimal number can be converted into binary by converting each hexadecimal digit to its 4 bit binary equivalent. Sixteen possible hexadecimal digits and their binary equivalents are listed in Table 2.5.

Example: Convert $(AB)_{16}$ to binary.

4-bit binary equivalents of each hexadecimal digit are

A   B
↓   ↓
1010   1011

$(AB)_{16} = (10101011)_2$

Example: Convert $(2F15)_{16}$ to binary.

4-bit binary equivalents of each hexadecimal digit are

2   F   1   5
↓   ↓   ↓   ↓
0010   1111   0001   0101

$(2F15)_{16} = (10111100010101)_2$

Hexa decimal	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Table 2.5 : Binary equivalent of hexadecimal digits

2.2.9 Binary to octal conversion

A binary number can be converted into its octal equivalent by grouping binary digits to group of 3 bits and then each group is converted to its octal equivalent. Start grouping from right to left.

Example: Convert $(101100111)_2$ to octal.

We can group above binary number 101100111 from right as shown below.

101   100   111
↓   ↓   ↓
5   4   7

$(101100111)_2 = (547)_8$

Example: Convert $(10011000011)_2$ to octal.

We can group above binary number 10011000011 from right as shown below.

010   011   000   011
↓   ↓   ↓   ↓
2   3   0   3

$(10011000011)_2 = (2303)_8$

After grouping, if the left most group has no 3 bits, then add leading zeros to form 3 bit binary.

2.2.10 Binary to hexadecimal conversion

A binary number can be converted into its hexadecimal equivalent by grouping binary digits to group of 4 bits and then each group is converted to its hexadecimal equivalent. Start grouping from right to left.

Example: Convert $(101100111010)_2$ to hexadecimal.

We can group the given binary number 101100111010 from right as shown below:

$$\begin{array}{ccc} 1011 & 0011 & 1010 \\ \downarrow & \downarrow & \downarrow \\ B & 3 & A \end{array}$$

$$\boxed{(101100111010)_2 = (B3A)_{16}}$$

Example: Convert $(110111100001100)_2$ to hexadecimal.

We can group the given binary number 110111100001100 from right as shown below:

After grouping, if the left most group has no 4 bits, then add leading zeros to form 4 bit binary.

$$\begin{array}{cccc} 0110 & 1111 & 0000 & 1100 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 6 & F & 0 & C \end{array}$$

$$\boxed{(110111100001100)_2 = (6F0C)_{16}}$$

2.2.11 Octal to hexadecimal conversion

Conversion of an octal number to hexadecimal number is a two step process. Octal number is first converted into binary. This binary equivalent is then converted into hexadecimal.

Example: Convert $(457)_8$ to hexadecimal equivalent.

First convert $(457)_8$ into binary.

$$\begin{array}{rccc} (457)_8 = & 4 & 5 & 7 \\ & \downarrow & \downarrow & \downarrow \\ & 100 & 101 & 111 \end{array}$$

$$= (100101111)_2$$

Then convert $(100101111)_2$ into hexadecimal as follows:

$$\begin{array}{rlcc} (100101111)_2 & = 0001 & 0010 & 1111 \\ & \quad\downarrow & \downarrow & \downarrow \\ & = \ 1 & 2 & F \\ & = (12F)_{16} & & \end{array}$$

$$\boxed{(457)_8 = (12F)_{16}}$$

2.2.12 Hexadecimal to octal conversion

Conversion of an hexadecimal to octal number is also a two step process. Hexadecimal number is first converted into binary. This binary equivalent is then converted into octal.

Example: Convert $(A2D)_{16}$ into octal equivalent.

First convert $(A2D)_{16}$ into binary.

$$(A2D)_{16} = \begin{array}{ccc} A & 2 & D \\ \downarrow & \downarrow & \downarrow \\ 1010 & 0010 & 1101 \end{array}$$

$$= (101000101101)_2$$

Then convert $(101000101101)_2$ into octal as follows:

$$(101000101101)_2 = \begin{array}{cccc} 101 & 000 & 101 & 101 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 5 & 0 & 5 & 5 \end{array}$$

$$= (5055)_8$$

$$\boxed{(A2D)_{16} = (5055)_8}$$

Table 2.6 shows procedures for various number conversions.

Conversion	Procedure
Decimal to Binary	Repeated division by 2 and grouping the remainders
Decimal to Octal	Repeated division by 8 and grouping the remainders
Decimal to Hexadecimal	Repeated division by 16 and grouping the remainders
Binary to Decimal	Multiply binary digit by place value(power of 2) and find their sum
Octal to Decimal	Multiply octal digit by place value (power of 8) and find their sum
Hexadecimal to Decimal	Multiply hexadecimal digit by place value (power of 16) and find their sum
Octal to Binary	Converting each octal digit to its 3 bit binary equivalent
Hexadecimal to Binary	Converting each hexadecimal digit to its 4 bit binary equivalent
Binary to Octal	Grouping binary digits to group of 3 bits from right to left
Binary to Hexadecimal	Grouping binary digits to group of 4 bits from right to left
Octal to Hexadecimal	Convert octal to binary and then binary to hexadecimal
Hexadecimal to Octal	Convert hexadecimal to binary and then binary to octal

Table 2.6 : Procedure for number conversions

Check yourself

1. Convert the decimal number 31 to binary.
2. Find decimal equivalent of $(10001)_2$
3. If $(x)_8 = (101011)_2$, then find x.
4. Fill in the blanks:

   a) $(\_\_\_\_\_\_)_2 \quad = \quad (AB)_{16}$

   b) $(\_\_D\_\_)_{16} \quad = \quad (1010\_\_\_\_1000)_2$

   c) $0.25_{10} \quad = \quad (\_\_\_\_\_\_)_2$
5. Find the largest number in the list

   (i) $(1001)_2$ (ii) $(A)_{16}$ (iii) $(10)_8$ (iv) $(11)_{10}$

2.3 Binary arithmetic

As in the case of decimal number system, arithmetic operations are performed in binary number system. When we give instruction to add two decimal numbers, the computer actually adds their binary equivalents. Let us see how binary addition and subtraction are carried out.

2.3.1 Binary addition

The rules for adding two bits are as follows:

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Note that a carry bit 1 is created only when two ones are added. If three ones are added (i.e. 1+1+1), then the sum bit is 1 with a carry bit 1.

Example: Find sum of binary numbers 1011 and 1001.

```
  1 0 1 1 +
  1 0 0 1
 ---------
1 0 1 0 0
```

Example: Find sum of binary numbers 110111 and 10011.

```
  1 1 0 1 1 1 +
  1 0 0 1 1 0
 -------------
1 0 1 1 1 0 1
```

2.3.2 Binary subtraction

The rules for subtracting a binary digit from another digit are as follows.

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Note that when 1 is subtracted from 0 the difference is 1, but 1 is borrowed from immediate left bit of first number. The above rules can be used only when a small binary number is subtracted from a large binary number.

Example: Subtract $(10101)_2$ from $(11111)_2$.

```
 1 1 1 1 1 −
 1 0 1 0 1
 ---------
 0 1 0 1 0
```

Example: Subtract $(10111)_2$ from $(101000)_2$.

```
 1 0 1 0 0 0 −
   1 0 1 1 1
   ---------
   1 0 0 0 1
```

2.4 Data representation

Computer uses a fixed number of bits to represent a piece of data which could be a number, a character, image, sound, video etc. Data representation is the method used internally to represent data in a computer. Let us see how various types of data can be represented in computer memory.

2.4.1 Representation of numbers

Numbers can be classified into integer numbers and floating point numbers. Integers are whole numbers or numbers without any fractional part. A floating point number or a real number is a number with fractional part. These two numbers are treated differently in computer memory. Let us see how integers are represented.

a. Representation of integers

There are three methods for representing an integer number in computer memory. They are

i) Sign and magnitude representation
ii) 1's complement representation
iii) 2's complement representation

The following data representation methods are based on 8 bit word length.

A **word** is basically a fixed-sized group of bits that are handled as a unit by a processor. Number of bits in a word is called **word length**. The word length is the choice of computer designer and some popular word lengths are 8, 16, 32 and 64.

i. Sign and magnitude representation

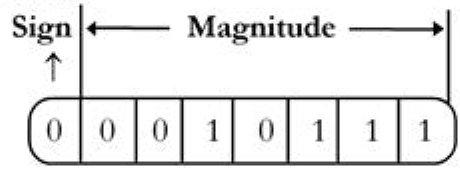
In this method, first bit from left (MSB) is used for representing sign of integer and remaining 7-bits are used for representing magnitude of integer. For negative integers sign bit is 1 and for positive integers sign bit is 0. Magnitude is represented as 7-bit binary equivalent of the integer.

Example: Represent + 23 in sign and magnitude form.

Number is positive, so first bit (MSB) is 0.

7 bit binary equivalent of 23 = $(0010111)_2$

So + 23 can be represented as $(00010111)_2$

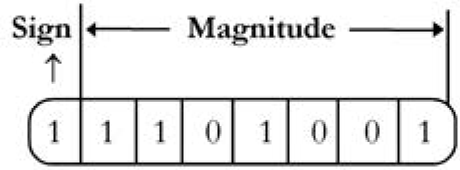


Example: Represent -105 in sign and magnitude form.

Number is negative, so first bit(MSB) is 1

7 bit binary equivalent of 105 = $(1101001)_2$

So -105 can be represented as $(11101001)_2$



Note: In this method an 8 bit word can represent 2^8-1 =255 numbers (i.e. -127 to +127). Similarly, a 16 bit word can represent 2^{16}-1 = 65535 numbers (i.e. -32767 to +32767). So, an *n*-bit word can represent 2^n-1 numbers i.e., $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$. The integer 0 can be represented in two ways: +0 = 00000000 and –0 = 10000000.

ii. 1's complement representation

In this method, first find binary equivalent of absolute value of integer. If number of digits in binary equivalent is less than 8, provide zero(s) at the left to make it 8-bit form. 1's complement of a binary number is obtained by replacing every 0 with 1 and every 1 with 0. Some binary numbers and the corresponding 1's compliments are given below:

Binary Number	1's Complement
11001	00110
10101	01010

If the number is negative it is represented as 1's complement of 8-bit form binary. If the number is positive, the 8-bit form binary equivalent itself is the 1's complement representation.

Example: Represent -119 in 1's complement form.

Binary of 119 in 8-bit form $= (01110111)_2$

-119 in 1's complement form $= (10001000)_2$

Example: Represent +119 in 1's complement form.

Binary of 119 in 8-bit form $= (01110111)_2$

+119 in 1's complement form $= (01110111)_2$

(No need to find 1's complement, since the number is positive)

Note: In this representation if first bit (MSB) is 0 then number is positive and if MSB is 1 then number is negative. So 8 bit word can represent integers from -127 (represented as 10000000) to +127 (represented as 01111111). Here also integer 0 can be represented in two ways: +0 = 00000000 and –0 = 11111111. An *n*-bit word can represent 2^n-1 numbers i.e. -(2^{n-1}-1) to +(2^{n-1}-1).

iii. 2's complement representation

In this method, first find binary equivalent of absolute value of integer and write it in 8-bit form. If the number is negative, it is represented as 2's complement of 8-bit form binary. If the number is positive, 8-bit form binary itself is the representation. 2's complement of a binary number is calculated by adding 1 to its 1's complement.

For example, let us find the 2's complement of $(10101)_2$.

1's complement of $(10101)_2$ $= (01010)_2$

So 2's complement of $(10101)_2$ $=$ 01010 +

1

$= (01011)_2$

Example: Represent -38 in 2's complement form.

Binary of 38 in 8-bit form $= (00100110)_2$

-38 in 2's complement form $=$ 11011001+

1

$= (11011010)_2$

Example: Represent +38 in 2's complement form.

Binary of 38 in 8-bit form $= (00100110)_2$

+38 in 2's complement form $= (00100110)_2$ (No need to find 2's complement)

Note: In this representation if first bit (MSB) is 0 then number is positive and if MSB is 1 then number is negative. Here integer 0 has only one way of representation and is 00000000. So an 8 bit word can represent integers from -128 (represented as 10000000) to +127(represented as 01111111). It is the most common integer representation. An *n*-bit word can represent 2^n numbers -(2^{n-1}) to + (2^{n-1}-1). Table 2.7 shows the comparison of different representation methods of integers in 8-bit word length.

Features	Sign & Magnitude	1's Complement	2's Complement	Remarks
Range	-127 to +127	-127 to +127	-128 to +127	Range is more in 2's complement
Total Numbers	255	255	256	
Representation of integer 0	Two ways of representation	Two ways of representation	Only one way of representation	In 2's complement there is no ambiguity in 0 representation
Representation of positive integers	Binary equivalent of integer in 8 bit form	Binary equivalent of integer in 8 bit form	Binary equivalent of integer in 8 bit form	All three forms are same.
Representation of negative integers	Sign bit 1 and magnitude is represented in 7 bit binary form	Find 1's complement of 8 bit form binary	Find 2's complement of 8 bit form binary	For all negative numbers MSB is 1

Table 2.7 : Comparison for representation of integers in 8-bit word length

Subtraction using complements

We have discussed how to subtract a binary number from another binary number. But to design and implement an electronic circuit for this method of subtraction is really complex and difficult. Circuitry for binary addition is simpler. So it is better if we can subtract through addition. For that we use the concept of complements. There are two methods of subtraction using complements.

Subtraction using 1's complement

The steps for subtracting a smaller binary number from a larger binary number are:

Step 1: Add 0s to the left of smaller number, if necessary, to make two numbers with same number of bits.

Step 2: Find 1's complement of subtrahend (Number to be subtracted, here small number)

Step 3: Add the complement with minuend (Number from which subtracting, here larger number)

Step 4: Add the carry 1 to the sum to get the answer.

Example: Subtract 100_2 from 1010_2 using 1's complement.

At first find 1's complement of 0100 and it is 1011

To compare the three types of representations let us consider the following table. For clarity and easy illustration, 4-bits are used to represent the numbers in this table .

Number	Sign & Magnitude	1's Complement	2's Complement
-8	Not possible	Not possible	1000
-7	1111	1000	1001
-6	1110	1001	1010
-5	1101	1010	1011
-4	1100	1011	1100
-3	1011	1100	1101
-2	1010	1101	1110
-1	1001	1110	1111
0	1000 or 0000	0000 or 1111	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111

From this table, it is clear that the MSB of a binary number indicates the sign of the corresponding decimal number irrespective of the representation. That is, if the MSB is 1, the number is negative and if it is 0, the number is positive. The table also shows that only 2's complement method can represent the maximum numbers for a given number of bits. This fact reveals that, a number below -7 and above +7 cannot be represented using 4-bits in sign & magnitude form and 1's complement form. So we go for 8-bit representation. Similarly in 2's complement method, if we want to handle numbers outside the range -8 to +7, eight bits are required.

In 8-bits implementation, the numbers from -128 to +127 can be represented in 2's complement method. The range will be -127 to +127 for the other two methods. For the numbers outside this range, we use 16 bits and so on for all the representations.

Add it with larger number, i.e.	1010	+
	1011	
	1 0101	
MSB	0101	+ MSB is removed and added with the result
	1	
The result is	0110	

Subtraction using 2's complement

To subtract a smaller binary number from a larger binary number the following are the steps.

Step 1: Add 0s to the left of smaller number, if necessary, to make the two numbers have the same number of bits.

Step 2: Find 2's complement of subtrahend (Number to be subtracted, here the smaller number).

Step 3: Add the 2's complement with minuend (Number from which subtracting, here the larger number).

Step 3: Ignore the carry.

Example: Subtract $(100)_2$ from $(1010)_2$ using 2's complement.

2's complement of 0100	1100	
Add it with larger number, i.e.	1010	+
	1100	
Ignore the carry to get the result	1 0110	
The result is	110	

b. Representation of floating point numbers

A floating point number / real number consists of an integer part and a fractional part. A real number can be written in a special notation called the floating point notation. Any number in this notation contains two parts, ***mantissa*** and ***exponent.***

For example, 25.45 can be written as 0.2545×10^2, where 0.2545 is the mantissa and the power 2 is the exponent. (In normalised floating point notation mantissa is between 0.1 and 1). Similarly -0.0035 can be written as $-0.35\text{x}10^{-2}$, where -0.35 is mantissa and -2 is exponent.

Let us see how a real number is represented in 32 bit word length computer. Here 24 bits are used for storing mantissa (among these the first bit is for sign) and 8 bits are used for storing exponent (first bit for sign) as in Figure 2.3. Assume that decimal point is to the right of the sign bit of mantissa. No separate space is reserved for storing decimal point.

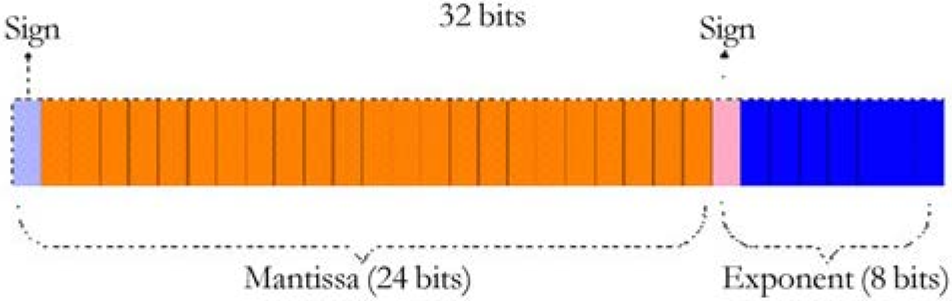


Fig 2.3: Representation of floating point numbers

Consider the real number 25.45 mentioned earlier, that can be written as 0.2545×10^2, where 0.2545 is the mantissa and 2 is the exponent. These numbers are converted into binary and stored in respective locations. Various standards are followed for representing mantissa and exponent. When word length changes, bits used for storing mantissa and exponents will change accordingly.

In real numbers, binary point keeps track of mantissa part and exponent part. Since the value of mantissa and exponent varies from number to number the binary point is not fixed. In other words it floats and hence such a representation is called floating point representation.

2.4.2 Representation of characters

We have discussed methods for representing numbers in computer memory. Similarly there are different methods to represent characters. Some of them are discussed below.

a. ASCII

The code called ASCII (pronounced "AS-key"), which stands for American Standard Code for Information Interchange, uses 7 bits to represent each character in computer memory. The ASCII representation has been adopted as a standard by the U.S. government and is widely accepted. A unique integer number is assigned to each character. This number called ASCII code of that character is converted into binary for storing in memory. For example, ASCII code of A is 65, its binary equivalent in 7-bit is 1000001. Since there are exactly 128 unique combinations of 7 bits, this 7-bit code can represent only128 characters.

Another version is ASCII-8, also called extended ASCII, which uses 8 bits for each character, can represent 256 different characters. For example, the letter A is represented by 01000001, B by 01000010 and so on. ASCII code is enough to represent all of the standard keyboard characters.

b. EBCDIC

It stands for Extended Binary Coded Decimal Interchange Code. This is similar to ASCII and is an 8 bit code used in computers manufactured by International Business Machine (IBM). It is capable of encoding 256 characters. If ASCII coded data is to be used in a computer which uses EBCDIC representation, it is necessary to transform ASCII code to EBCDIC code. Similarly if EBCDIC coded data is to be used in a ASCII computer, EBCDIC code has to be transformed to ASCII.

c. ISCII

ISCII stands for Indian Standard Code for Information Interchange or Indian Script Code for Information Interchange. It is an encoding scheme for representing various writing systems of India. ISCII uses 8-bits for data representation. It was evolved by a standardisation committee under the Department of Electronics during 1986-88, and adopted by the Bureau of Indian Standards (BIS). Nowadays ISCII has been replaced by Unicode.

d. Unicode

Using 8-bit ASCII we can represent only 256 characters. This cannot represent all characters of written languages of the world and other symbols. Unicode is developed to resolve this problem. It aims to provide a standard character encoding scheme, which is universal and efficient. It provides a unique number for every character, no matter what the language and platform be.

Unicode originally used 16 bits which can represent up to 65,536 characters. It is maintained by a non-profit organisation called the Unicode Consortium. The Consortium first published the version 1.0.0 in 1991 and continues to develop standards based on that original work. Nowadays Unicode uses more than 16 bits and hence it can represent more characters. Unicode can represent characters in almost all written languages of the world.

2.4.3 Representation of audio, image and video

In the previous sections we have discussed different data representation techniques and standards used for the computer representation of numbers and characters. While we attempt to solve real life problems with the aid of a digital computer, in most cases we may have to represent and process data other than numbers and characters. This may include audio data, images and videos. We can see that like numbers and characters, the audio, image and video data also carry information. In this section we will see different file formats for storing sound, image and video.

Digital audio, image and video file formats

Multimedia data such as audio, image and video are stored in different types of files. The variety of file formats is due to the fact that there are quite a few approaches to compressing the data and a number of different ways of packaging the data. For example an image is most popularly stored in Joint Picture Experts Group (JPEG) file format. An image file consists of two parts - header information and image data. Information such as name of the file, size, modified data, file format, etc. are stored in the header part. The intensity value of all pixels is stored in the data part of the file.

The data can be stored uncompressed or compressed to reduce the file size. Normally, the image data is stored in compressed form. Let us understand what compression is. Take a simple example of a pure black image of size 400×400 pixels. We can repeat the information black, black, ..., black in all 16,0000 (400×400) pixels. This is the uncompressed form, while in the compressed form black is stored only once and information to repeat it 1,60,000 times is also stored. Numerous such techniques are used to achieve compression. Depending on the application, images are stored in various file formats such as bitmap file format (BMP), Tagged Image File Format (TIFF), Graphics Interchange Format (GIF), Portable (Public) Network Graphic (PNG).

What we said about the header file information and compression is also applicable for audio and video files. Digital audio data can be stored in different file formats like WAV, MP3, MIDI, AIFF, etc. An audio file describes a format, sometimes referred to as the 'container format', for storing digital audio data. For example WAV file format typically contains uncompressed sound and MP3 files typically contain compressed audio data. The synthesised music data is stored in MIDI(Musical Instrument Digital Interface) files. Similarly video is also stored in different files such as AVI (Audio Video Interleave) - a file format designed to store both audio and video data in a standard package that allows synchronous audio with video playback, MP3, JPEG-2, WMV, etc.

Check yourself

1. Which is the MSB of representation of -80 in the sign and magnitude method?
2. Write 28.756 in mantissa exponent form.
3. ASCII stands for ______________.
4. Represent -60 in 1's complement form.
5. Define Unicode.
6. List any two image file formats.

2.5 Introduction to Boolean algebra

In many situations in our life we face questions that require 'Yes' or 'No' answers. Similarly much of our thinking process involves answering questions with 'Yes' or 'No'. The way of finding truth by answering such two-valued questions is known as human reasoning or logical reasoning. These values can be expressed as 'True' or 'False' and numerically 1 or 0. These values are known as binary values or Boolean values. Boolean algebra is the algebra of logic which is a part of mathematical algebra that deals with the operations on variables that represent the values 1 and 0. The name Boolean algebra is given to honour the British mathematician George Boole, as he was the person who established the link between logic and mathematics. His revolutionary paper 'An Investigation of the laws of thought' led to the development of Boolean algebra.

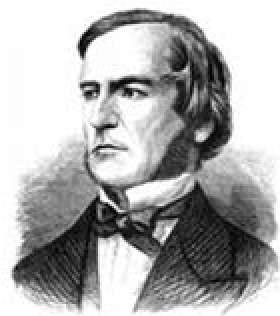


Fig. 2.4: George Boole (1815 - 1864)

2.5.1 Binary valued quantities

Let us consider the following:

1. Should I take an umbrella?
2. Will you give me your pen?
3. George Boole was a British mathematician.
4. Kerala is the biggest state in India.
5. Why were you absent yesterday?
6. What is your opinion about Boolean algebra?

1st and 2nd sentences are questions which can be answered as YES or NO. These cases are called binary decisions and the results are called binary values. The 3rd statement is TRUE and 4th statement is FALSE. But 5th and 6th sentences cannot be answered like the cases above. The sentences which can be determined to be TRUE or FALSE are called ***logical statements*** or truth functions and the results TRUE or FALSE are called binary values or logical constants. The **logical constants** are represented by 1 and 0, where 1 stands for TRUE and 0 for FALSE. The variables which can store (hold) logical constants 1 and 0 are called logical variables or ***Boolean variables***.

2.5.2 Boolean operators and logic gates

We have already seen that data fed to a computer must be converted into a combination of 1s and 0s. All data, information and operations are represented inside the computer using 0s and 1s. The operations performed on these Boolean values are called **Boolean operations**. As we know, operators are required to perform these operations. These operators are called Boolean operators or logical operators. There are three basic logical operators in Boolean algebra. These operators and their operations are as follows:

OR → Logical Addition
AND → Logical Multiplication
NOT → Logical Negation

The first two operators require two operands and the third requires only one operand. Here the operands are always Boolean variable or constants and the result will always be either True (1) or False (0).

Computers perform these operations with some electronic circuits, called logic circuits. A logic circuit is made up of individual units called gates, where a gate represents a Boolean operation. A **Logic gate** is a physical device that can perform logical operations on one or more logical inputs and produce a single logical output. Logic gates are primarily implemented using diodes or transistors acting as electronic switches. There are three basic logic gates and they represent the three basic Boolean operations. These gates are OR, AND and NOT.

a. The OR operator and OR gate

Let us consider a real life situation. When do you use an umbrella? When it rains, isn't it? And of course, if it is too sunny. We can combine these two situations using a compound statement like "If it is raining or if it is sunny, we use an umbrella". Note down the use of ***or*** in this statement. The interpretation of this statement can be shown as in Table 2.8. The logical reasoning of the use of umbrella in our example very much resembles the Boolean OR operation.

Raining	Sunny	Need Umbrella
No	No	No
No	Yes	Yes
Yes	No	Yes
Yes	Yes	Yes

Table 2.8: Logical OR operation

The OR operator performs logical addition and the symbol used for this operation is + (plus). The expression A + B is read as A OR B. Table 2.9 is the truth table that represents the OR operation. Assume that the variables A and B are the inputs (operands) and A + B is the output (result). It is clear from the truth table that, if any one of the inputs is 1 (True), the output will be 1 (True).

Truth Table is a table that shows Boolean operations and their results. It lists all possible inputs for the given operation and their corresponding output. Usually these operations consist of operand variables and operators. The operands and the operators together are called Boolean expression. Truth Table represents all possible values of the operands and the corresponding results (values) of the operation. A Boolean expression with m operands (variables) and n operators require 2^m rows and $m + n$ columns.

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Table 2.9 : Truth table of OR operation

While designing logic circuits, the logic gate used to implement logical OR operation is called logical **OR gate**. Figure 2.5 shows the OR gate symbol in Boolean algebra.

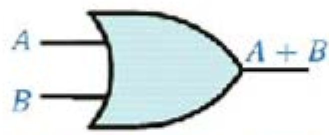


Fig. 2.5 : Logical OR gate

The working of this gate can be illustrated with an electronic circuit. Figure 2.6 illustrates the schematic circuit of parallel switches which shows the idea of an OR gate. Here A and B are two switches and Y is a bulb. Each switch and the bulb can take either close (ON) or open (OFF) state. Now let us relate the operation of the above circuit with the functioning of OR gate. Assume that OFF represents the logical LOW state (say 0) and ON represents the logical HIGH (say 1) state. If we consider the state of switches A and B as input to the OR gate and state of bulb as output of OR gate, then the truth table shown in Table 2.9 will describe the operation of an OR gate. Thus the Boolean expression for OR gate can be written as: $Y = A + B$

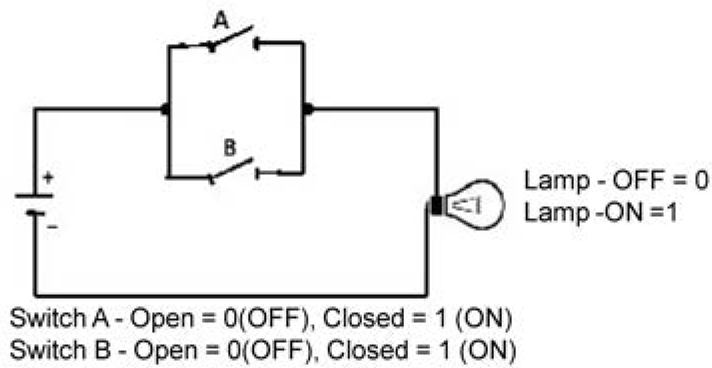


Fig. 2.6 : Circuit with two switches and a bulb for parallel connection

An OR gate can take more than two inputs. Let us see what will be the truth table, Boolean expression and logical symbol for the three input OR gate.

The truth table and the gate symbol shows that, the Boolean expression for the OR gate with three inputs is Y= A + B + C. Figure 2.7 shows the representation of OR gate with three inputs. From the truth tables 2.9 and 2.10, we can see that the output of OR gate is 1 if any input is 1; and output is 0 if and only if all inputs are 0.

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

Table 2.10 : Truth table for OR gate with 3 inputs

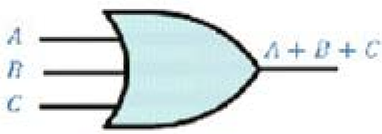


Fig 2.7 : OR gate with three inputs

b. The AND operator and AND gate

We will discuss another situation to understand the concept of AND Boolean operation. Suppose you are away from home and it is lunch time. You can have your food only if two conditions are satisfied – (i) there should be a hotel and (ii) you should have enough money. Here also, we can make a compound statement like "If there is a hotel and if we have money, we can have food". Note the use of ***and*** in this statement. Table 2.11 shows the logical reasoning of getting food and it very much resembles the Boolean AND operation.

Hotel	Money	Take Food
No	No	No
No	Yes	No
Yes	No	No
Yes	Yes	Yes

Table 2.11 : Logical AND operation

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

Table 2.12 : Truth table of AND operation

The AND operator performs logical multiplication and the symbol used for this operation is **.** (dot). The expression A **.** B is read as A AND B. Table 2.12 is the truth table that represents the AND operation. Assume that the variables A and B are the inputs (operands) and A **.** B is the output (result).

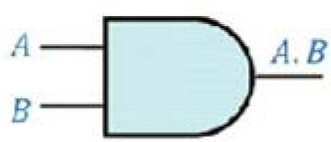


Fig. 2.8 : Logical AND gate

While designing logic circuits, the logic gate used to implement logical AND operation is called logical **AND gate**. Figure 2.8 shows the AND gate symbol in Boolean algebra.

The working of this gate can be illustrated with an electronic circuit shown in Figure 2.9. This schematic circuit has two serial switches which illustrates the idea of an AND gate. Here A and B are two switches and Y is a bulb. Each switch and the bulb can take either close (ON) or open (OFF) state. Now let us relate the operation of the above circuit with the functioning of AND gate. Assume that OFF represents the logical LOW state (say 0) and ON represents the logical HIGH state (say 1). If we consider the state of switches A and B as input to the AND gate and state of bulb as output of AND gate, then the Boolean expression for AND gate can be written as: Y = A . B

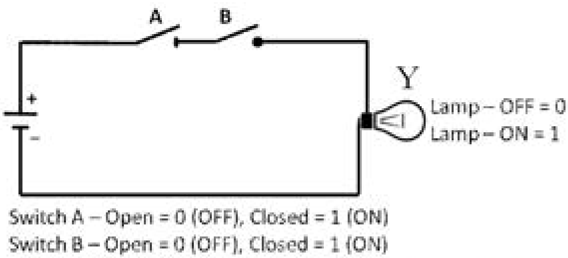


Fig. 2.9 : Circuit with two switches and a bulb for serial connection

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

Table 2.13 : Truth table for AND gate with 3 inputs

An AND gate can take more than two inputs. Let us see what will be the truth table, Boolean expression and logical symbol for three input AND gate. The truth table and the gate symbol shows that the Boolean expression for the AND gate with three inputs is Y = A . B . C

Figure 2.10 shows the representation of AND gate with three inputs. From Truth Tables 2.12 and 2.13, we can see that the output of AND gate is 0 if any input is 0; and output is 1 if and only if all inputs are 1.

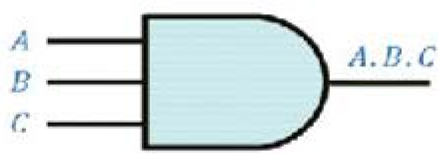


Fig. 2.10 : AND gate with three inputs

c. The NOT operator and NOT gate

Let us discuss another case to familiarise the Boolean NOT operation. Suppose you jog everyday in the morning. Can you do it every day? If it rains, can you jog in the morning? Table 2.14 shows all the possibilities of this situation. It is quite similar to Boolean NOT operation.

Raining	Jogging
No	Yes
Yes	No

Table 2.14: Logical NOT

It is a unary operator and hence it requires only one operand. The NOT operator performs logical negation and the symbol used for this operation is - (over-bar).

A	$\overline{A}$
0	1
1	0

Table 2.15 : Truth table of NOT operation

The expression $\overline{A}$ is read as A bar . It is also expressed as A' and read as A dash. Table 2.15 is the truth table that represents the NOT operation. Assume that the variable A is the input (operand) and $\overline{A}$ is the output (result). It is clear from the truth table that, the output will be the opposite value of the input. The logic gate used to implement NOT operation is NOT gate. Figure 2.11 shows the NOT gate symbol.

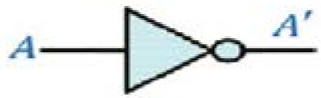


Fig. 2.11 : NOT gate

A NOT gate is also called ***inverter***. It has only one input and one output. The input is always changed into its opposite state. If input is 0 , the NOT gate will give its complement or opposite which is 1. If the input is 1, then the NOT gate will complement it to 0.

Check yourself

1. Define the term Boolean variable.
2. A logic circuit is made up of individual units called _______.
3. Name the logical operator/gate which gives high output if and only if all the inputs are high.
4. Define the term truth table.
5. An AND operation performs logical ________ and an OR operation performs logical _______.
6. Draw the logic symbol of OR gate.

2.6 Basic postulates of Boolean algebra

Boolean algebra being a system of mathematics, consists of certain fundamental laws. These fundamental laws are called postulates. They do not have proof, but are made to build solid framework for scientific principles. On the other hand, there are some theorems in Boolean algebra which can be proved based on these postulates and laws.

Postulate 1: Principles of 0 and 1

If $A \neq 0$, then $A = 1$ and if $A \neq 1$, then $A=0$

Postulate 2: OR Operation (Logical Addition)

$0 + 0 = 0$ $\quad$ $0 + 1 = 1$ $\quad$ $1 + 0 = 1$ $\quad$ $1 + 1 = 1$

Postulate 3: AND Operation (Logical Multiplication)

$0 . 0 = 0$ $\quad$ $0 . 1 = 0$ $\quad$ $1 . 0 = 0$ $\quad$ $1 . 1 = 1$

Postulate 4: NOT Operation (Logical Negation or Compliment Rule)

$$\bar{0} = 1 \qquad \bar{1} = 0$$

Principle of Duality

When Boolean variables and/or values are combined with Boolean operators, Boolean expressions are formed. X + Y and $\overline{A}$ + 1 are examples of Boolean expressions. The postulates 2, 3 and 4 are all Boolean statements. Consider the statements in postulate 2. If we change the value 0 by 1 and 1 by 0, and the operator OR (+) by AND (.), we will get the statements in postulate 3. Similarly, if we change the value 0 by 1 and 1 by 0, and the operator AND (.) by OR (+) in statements of postulate 3, we will get the statements of postulate 2. This concept is known as principle of duality.

The principle of duality states that for a Boolean statement, there exists its dual form, which can be derived by

(i) changing each OR sign (+) to AND sign (.)

(ii) changing each AND sign (.) to OR sign (+)

(iii) replacing each 0 by 1 and each 1 by 0

2.7 Basic theorems of Boolean algebra

There are some standard and accepted rules in every theory. The set of rules are known as ***axioms*** of the theory. A conclusion can be derived from a set of presumptions by using these axioms or postulates. This conclusion is called law or theorem. Theorems of Boolean algebra provide tools for simplification and manipulation of Boolean expressions. Let us discuss some of these laws or theorems. These laws or theorems can be proved using truth tables and Boolean laws that are already proved.

2.7.1 Identity law

If X is a Boolean variable, the law states that:

(i) $0 + X = X$ (ii) $1 + X = 1$

(iii) $0 . X = 0$ (iv) $1 . X = X$

Statements (i) and (ii) are known as additive identity law; and statements (iii) and (iv) are called multiplicative identity. Also note that, statement (iv) is the dual of (i) and vice versa. Similarly, statements (ii) and (iii) are dual forms. The truth tables shown in Tables 2.16(a), 2.16(b), 2.17(a) and 2.17 (b) prove these laws.

0	X	0 + X
0	0	0
0	1	1

Table 2.16 (a) : Additive Identity law

1	X	1 + X
1	0	1
1	1	1

Table 2.16 (b) : Additive Identity law

Table 2.16 (a) shows that columns 2 and 3 are the same and it is proved that $0 + X = X$. Similarly, columns 1 and 3 of table 12.16 (b) are the same and hence the statement $1+X=1$ is true.

0	X	0 . X
0	0	0
0	1	0

Table 2.17(a) : Multiplicative Identity law

1	X	1 . X
1	0	0
1	1	1

Table 2.17(b) : Multiplicative Identity law

Table 2.17 (a) shows that columns 1 and 3 are the same and it is proved that $0 . X = 0$. Similarly, columns 2 and 3 of Table 2.17 (b) are the same and hence the statement $1. X=X$ is true.

2.7.2 Idempotent law

The idempotent law states that: (i) $X + X = X$

and (ii) $X . X = X$

If the value of X is 0, the statements are true, because $0 + 0 = 0$ (*Postulate 2*) and $0 . 0 = 0$ (*Postulate 3*). Similarly the statements will be true when the value of X is 1. Truth Tables 2.18 (a) and 2.18 (b) shows the proof of these laws. Also note that the statements are dual to each other.

X	X	X + X
0	0	0
1	1	1

Table 2.18 (a) : Idempotent law

X	X	X . X
0	0	0
1	1	1

Table 2.18 (b) : Idempotent law

2.7.3 Involution law

This law states that: $\overline{\overline{X}} = X$

Let $X = 0$, then $\overline{X} = 1$ (*Postulate 4*); and if we take its compliment, $\overline{\overline{X}} = \overline{1} = 0$, which is same as X. The statement will also be true, when the value of X is 1.

Columns 1 and 3 of Table 2.19 show that $\overline{\overline{X}} = X$.

X	$\overline{X}$	$\overline{\overline{X}}$
0	1	0
1	0	1

Table 2.19 : Involution law

2.7.4 Complimentary law

The complimentary law states that: (i) $X + \overline{X} = 1$

and (ii) $X \cdot \overline{X} = 0$

If the value of X is 0, then $\overline{X}$ becomes 1. Hence, $X + \overline{X}$ becomes 0 + 1, which results into 1 (*Postulate 2*). Similarly when X is 1, $\overline{X}$ will be 0. The truth tables 2.20 (a) and 2.20 (b) show the proof of these laws taking all the possibilities. Also note that the statements are dual to each other.

X	$\overline{X}$	$X + \overline{X}$
0	1	1
1	0	1

Table 2.20 (a) : Complimentary law

X	$\overline{X}$	$X \cdot \overline{X}$
0	1	0
1	0	0

Table 2.20 (b) : Complimentary law

2.7.5 Commutative law

Commutative law allows to change the position of variables in OR and AND operations. If X and Y are two variables, the law states that:

(i) $X + Y = Y + X$

and (ii) $X \cdot Y = Y \cdot X$

The truth table shown in Tables 2.21 (a) and 2.21 (b) prove these statements.

X	Y	X + Y	Y + X
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Table 2.21 (a) : Commutative law

X	Y	X.Y	Y.X
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Table 2.21 (b) : Commutative law

The law ensures that order of the operands for OR and AND operations does not affect the output in each case.

2.7.6 Associative law

In the case of three operands for OR and AND operations, associative law allows grouping of operands differently. If X, Y and Z are three variables, the law states that:

(i) $X + (Y + Z) = (X + Y) + Z$

and (ii) $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$

The truth tables shown in Tables 2.22(a) and 2.22(b) prove these statements.

X	Y	Z	X + Y	Y + X	X+(Y + Z)	(X+Y)+Z
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	0	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Table 2.22(a) : Associative law 1

In Table 2.22(a), columns 6 and 7 show that $X + (Y + Z) = (X + Y) + Z$. Columns 6 and 7 of Table 2.22(b) show the validity of the experience $X . (Y . Z) = (X . Y) . Z$

X	Y	Z	X . Y	Y . Z	X . (Y . Z)	(X . Y) . Z
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Table 2.22(b) : Associative law 2

Associative law ensures that the order and combination of variables in OR (logical addition) or AND (logical multiplication) operations do not affect the final output.

2.7.7 Distributive law

Distributive law states that Boolean expression can be expanded by multiplying terms as in ordinary algebra. It also supports expansion of addition operation over multiplication. If X, Y and Z are variables, the law states that:

(i) $X . (Y + Z) = X . Y + X . Z$

and (ii) $X + Y . Z = (X + Y) . (X + Z)$

The following truth tables prove these statements:

X	Y	Z	Y + Z	X.(Y+Z)	X.Y	X.Z	X.Y + X.Z
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Table 2.23(a) : Distribution of Multiplication over Addition

Columns 5 and 8 of Table 2.23(a) show that $X . (Y + Z) = X . Y + X . Z$

X	Y	Z	Y . Z	X + Y . Z	X+Y	X+Z	(X+Y) . (X+Z)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Table 2.23(b) : Distribution of Addition over Multiplication

Columns 5 and 8 of Table 2.23(b) show that $X + Y . Z = (X + Y) . (X + Z)$

We are familiar with the first statement in ordinary algebra. To remember the second statement of this law, find the dual form of the first.

2.7.8 Absorption law

Absorption law is a kind of distributive law in which two variables are used and the result will be one of them. If X and Y are variables, the absorption law states that:

(i) $X + (X . Y) = X$

and (ii) $X . (X + Y) = X$

The truth tables shown Tables 2.24(a) and 2.24(b) prove the validity of the statements of absorption law.

X	Y	X . Y	X+(X.Y)
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Table 2.24 (a) : Absorption law

X	Y	X+Y	X.(X+Y)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Table 2.24 (b) : Absorption law

Columns 1 and 4 of Table 2.24(a) and columns 1 and 4 of Table 2.24(b) show that the laws are true.

Table 2.25 depicts all the Boolean laws we have discussed so far.

No.	Boolean Law	Statement 1	Statement 2
1	Additive Identity	$0 + X = X$	$1 + X = 1$
2	Multiplicative Identity	$0 . X = 0$	$1 . X = X$
3	Idempotent Law	$X + X = X$	$X . X = X$
4	Involution Law	$\overline{\overline{X}} = X$	
5	Complimentary Law	$X + \overline{X} = 1$	$X . \overline{X} = 0$
6	Commutative Law	$X + Y = Y + X$	$X . Y = Y . X$
7	Associative Law	$X + (Y + Z) = (X + Y) + Z$	$X . (Y . Z) = (X . Y) . Z$
8	Distributive Law	$X . (Y + Z) = X . Y + X . Z$	$X + (Y . Z) = (X+Y) . (X+Z)$
9	Absorption Law	$X + (X . Y) = X$	$X . (X + Y) = X$

Table 2.25 : Boolean laws

The laws we discussed have been proved using truth tables. Some of them can be proved by applying some other laws. This method of proof is called algebraic proof. Let us see some of them.

i. To prove that X . (X + Y) = X – Absorption law

LHS $= X . (X + Y)$

$= X . X + X . Y$ *(Distribution of multiplication over addition)*

$= X + X . Y$ *(Idempotent law)*

$= X . (1 + Y)$ *(Distribution of multiplication over addition)*

$= X . 1$ (*Additive identity*)

$= X$ (*Multiplicative identity*)

$= RHS$

ii. To prove that $X + (X . Y) = X$ – Absorption law

LHS $= X + (X . Y)$

$= X . 1 + X . Y$ (*Multiplicative identity*)

$= X . (1 + Y)$ (*Distribution of multiplication over addition*)

$= X . 1$ (*Additive identity*)

$= X$ (*Multiplicative identity*)

$= RHS$

iii. To prove that $X + (Y . Z) = (X+Y) . (X+Z)$ – Distributive Law

Let us take the expression on the RHS of this statement.

$(X+Y) . (X+Z)$

$= (X+Y) . X + (X+Y) . Z$ (*Distribution of multiplication over addition*)

$= X . (X+Y) + Z . (X +Y)$ (*Commutative law*)

$= X . X + X . Y + Z . X + Z . Y$ (*Distribution of multiplication over addition*)

$= X + X . Y + Z . X + Z . Y$ (*Idempotent law*)

$= X . 1 + X . Y + Z . X + Z . Y$ (*Multiplicative identity*)

$= X . (1 + Y) + Z . X + Z . Y$ (*Distribution of multiplication over addition*)

$= X . 1 + Z . X + Z . Y$ (*Additive identity*)

$= X . (1 + Z) + Z . Y$ (*Distribution of multiplication over addition*)

$= X . 1 + Z . Y$ (*Additive identity*)

$= X + Y . Z$ (*Multiplicative identity and Commutative law*)

$= LHS$

The expression obtained is the LHS of the given statement. Thus the theorem is proved.

2.8 De Morgan's theorems

Augustus De Morgan (1806 – 1871), a famous logician and mathematician of University College, London proposed two theorems to simplify complicated Boolean expressions. These theorems are known as De Morgan's theorems. The two theorems are:

(i) $\overline{X+Y} = \overline{X} \cdot \overline{Y}$

(ii) $\overline{X.Y} = \overline{X} + \overline{Y}$

Literally these theorems can be stated as

(i) "the complement of sum of Boolean variables is equal to product of their individual complements" and

(ii) "the complement of product of Boolean variables is equal to sum of their individual complements".

Algebraic proof of the first theorem

We have to prove that, $\overline{X+Y} = \overline{X} \cdot \overline{Y}$

Let us assume that, $Z = X + Y$ ______ (1)

Then, $\overline{Z} = \overline{X+Y}$ ______ (2)

We know that, by complimentary law, the equations (3) and (4) are true.

$Z + \overline{Z} = 1$ ______ (3)

$Z \,.\, \overline{Z} = 0$ ______ (4)

Substituting expressions (1) in (3) and (2) in (4), we will get equations (5) and (6).

$(X + Y) + (\overline{X+Y}) = 1$ ______ (5)

$(X + Y) \,.\, (\overline{X+Y}) = 0$ ______ (6)

For the time being let us assume that De Morgan's first theorem is true. If so, $(\overline{X+Y})$ in equations (5) and (6) can be substituted with $(\overline{X} \cdot \overline{Y})$. Thus equations (5) and (6) can be modified as follows:

$(X+ Y) + (\overline{X} \,.\, \overline{Y}) = 1$ ______ (7)

$(X + Y) \,.\, (\overline{X} \,.\, \overline{Y}) = 0$ ______ (8)

Now we will prove equations (7) and (8) separately. If they are correct, we can conclude that the assumptions we made to form those equations are also correct. That is, if equations (7) and (8) are true, De Morgan's theorem is also true.

Consider the LHS of equation (7),

$(X + Y) + (\overline{X} \,.\, \overline{Y}) = (X + Y + \overline{X}) \,.\, (X + Y + \overline{Y})$ (*Distributive Law*)

$= (X + \overline{X} + Y) \,.\, (X + Y + \overline{Y})$ (*Associative Law*)

$= (1 + Y) \,.\, (X + 1)$ (*Complimentary Law*)

$= 1 . 1$ (*Additive Identity*)

$= 1$

$= \text{RHS}$

Now, let us consider the LHS of equation (8),

$(X + Y) . (\overline{X} . \overline{Y})$ $= (X . \overline{X} . \overline{Y}) + (Y . \overline{X} . \overline{Y})$ (*Distributive Law*)

$= (X . \overline{X} . \overline{Y}) + (Y . \overline{Y} . \overline{X})$ (*Associative Law*)

$= (0 . \overline{Y}) + (0 . \overline{X})$ (*Complimentary Law*)

$= 0 + 0$ (*Multiplicative Identity*)

$= 0$

$= \text{RHS}$

We have algebraically proved equations (7) and (8), which mean that De Morgan's first theorem is proved. The theorem can also be proved using truth table, but it is left to you as an exercise.

Algebraic proof of the second theorem

We have to prove that, $\overline{X.Y} = \overline{X} + \overline{Y}$

Let us assume that, $Z = X . Y$ ______ (1)

Then, $\overline{Z} = \overline{X.Y}$ ______ (2)

We know that, by complimentary laws the equations (3) and (4) are true.

$Z + \overline{Z} = 1$ ______ (3)

$Z . \overline{Z} = 0$ ______ (4)

Substituting expressions (1) in (3) and (2) in (4), we will get the expressions (5) and (6).

$(X . Y) + (\overline{X.Y}) = 1$ ______ (5)

$(X . Y) . (\overline{X.Y}) = 0$ ______ (6)

For the time being let us assume that De Morgan's second theorem is true. If so, $(\overline{X.Y})$ in equations 5 and 6 can be substituted with $(\overline{X} + \overline{Y})$. Thus equations (5) and (6) can be modified as follows:

$(X . Y) + (\overline{X} + \overline{Y}) = 1$ ______ (7)

$(X . Y) . (\overline{X} + \overline{Y}) = 0$ ______ (8)

Now we will prove equations (7) and (8) separately. If they are correct, we can conclude that the assumptions we made to form those equations are also correct. That is, if equations (7) and (8) are true, De Morgan's theorem is also true.

Consider the LHS of equation (7),

$$
\begin{aligned}
(X \cdot Y) + (\overline{X} + \overline{Y}) &= (\overline{X} + \overline{Y}) + (X \cdot Y) && \text{(Commutative Law)} \\
&= (\overline{X} + \overline{Y} + X) \cdot (\overline{X} + \overline{Y} + Y) && \text{(Distributive Law)} \\
&= (\overline{X} + X + \overline{Y}) \cdot (\overline{X} + \overline{Y} + Y) && \text{(Associative Law)} \\
&= (1 + \overline{Y}) \cdot (\overline{X} + 1) && \text{(Complimentary Law)} \\
&= 1 \cdot 1 && \text{(Additive Identity)} \\
&= 1 \\
&= \text{RHS}
\end{aligned}
$$

Now, let us consider the LHS of equation (8),

$$
\begin{aligned}
(X \cdot Y) \cdot (\overline{X} + \overline{Y}) &= (X \cdot Y \cdot \overline{X}) + (X \cdot Y \cdot \overline{Y}) && \text{(Distributive Law)} \\
&= (X \cdot \overline{X} \cdot Y) + (X \cdot Y \cdot \overline{Y}) && \text{(Associative Law)} \\
&= (0 \cdot Y) + (X \cdot 0) && \text{(Complimentary Law)} \\
&= 0 + 0 && \text{(Multiplicative Identity)} \\
&= 0 \\
&= \text{RHS}
\end{aligned}
$$

We have algebraically proved equations (7) and (8), which mean that De Morgan's second theorem is proved. The theorem can also be proved using truth table, but it is left to you as an exercise.

We can extend Demorgan's theorem for any number of variables as shown below:

$$
\begin{aligned}
\overline{A + B + C + D + \ldots\ldots} &= \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \ldots\ldots \\
\overline{A.B.C.D\ldots\ldots} &= \overline{A} + \overline{B} + \overline{C} + \overline{D} + \ldots\ldots
\end{aligned}
$$

Although the identities above represent De Morgan's theorem, the transformation is more easily performed by following the steps given below:

(i) Complement the entire function

(ii) Change all the ANDs (.) to ORs (+) and all the ORs (+) to ANDs (.)

(iii) Complement each of the individual variables.

This process is called *demorganisation* and simply demorganisation is '*Break the line, change the sign*'.

Check yourself

1. Find the dual of Boolean expression A.B+B.C=1
2. Name the law which states that A+A=A.

   (a) Commutative law (b) Idempotent Law (c) Absorption Law
3. State De Morgan's theorems.

2.9 Circuit designing for simple Boolean expressions

By using basic gates, circuit diagrams can be designed for Boolean expressions. We have seen that the Boolean expressions A.B is represented using an AND gate, A+B is represented using an OR gate and $\overline{A}$ is represented using a NOT gate. Let us see how a circuit is designed for other Boolean expressions.

Consider a boolean expression $\overline{A}+B$, which is an OR operation with two input and first input is inverted. So circuit diagram can be drawn as shown in Figure 2.12.

Fig 2.12 : f (A, B) = $\overline{A}+B$

Example: Construct a logical circuit for Boolean expression $f(X, Y) = X.Y + \overline{Y}$

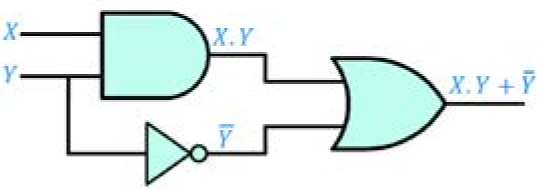


Fig 2.13 : f (X, Y) = X.Y + $\overline{Y}$

Example: Construct a logical expression for $f(a, b) = (a + b) . (\overline{a} + \overline{b})$

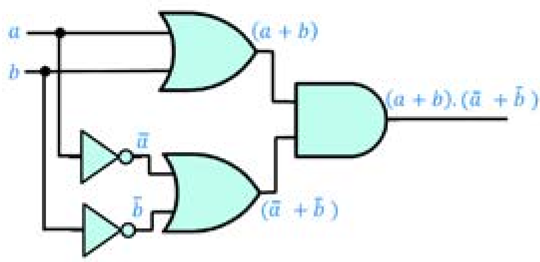


Fig 2.14 : f (a, b) = (a + b) . $(\overline{a} + \overline{b})$

Example: Construct logical circuit for the Boolean expression $\overline{a}.b + a.\overline{b}$

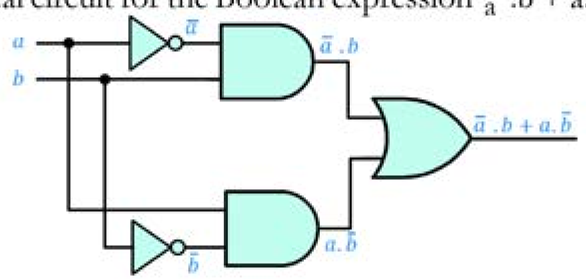


Fig 2.15 : f (a, b) = $\overline{a}.b + a.\overline{b}$

2.10 Universal gates

The NAND and NOR gates are called universal gates. A universal gate is a gate which can implement any Boolean function without using any other gate type. In practice, this is advantageous, since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in most of IC digital logic families.

2.10.1 NAND gate

This is an AND gate with its output inverted by a NOT gate. The logical circuit arrangement is shown in Figure 2.16.

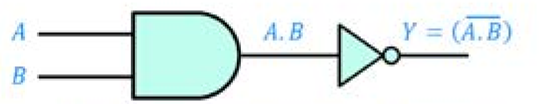


Fig. 2.16 : Circuit realisation of NAND gate

Note that A and B are the inputs of AND gate and its output is (A.B). The output of AND gate is inverted by an inverter (NOT gate) to get the resultant output Y as $(\overline{A.B})$. So the logical expression for a NAND gate is $(\overline{A.B})$. From the truth table shown as Table 2.26, we can see that output of a NAND gate is 1 if any one of the input is 0. It produces output 0 if and only if all inputs are 1. This is the inverse operation of an AND gate. So we can say that ***a NAND gate is an inverted AND gate.***

A	B	$Y = (\overline{A.B})$
0	0	1
0	1	1
1	0	1
1	1	0

Table 2.26 : NAND truth table

The logical symbol of NAND gate is shown in Figure 2.17. Note that the NAND symbol is an AND symbol with a small bubble at the output. The bubble is sometimes called an invert bubble.

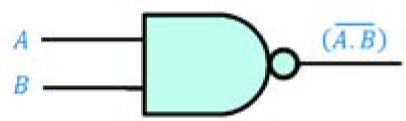


Fig. 2.17 : NAND gate

2.10.2 NOR gate

This is an OR gate with its output inverted by a NOT gate. The logical circuit arrangement is shown in Figure 2.18.

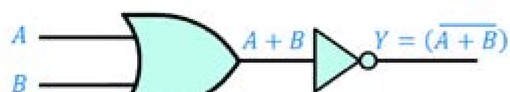


Fig. 2.18 : Circuit realisation of NOR gate

Note that A and B are the input of OR gate and its output is (A+B). The output of OR gate is inverted by an inverter (NOT gate) to get resultant output as $(\overline{A+B})$. So the logical expression for a NOR gate is $(\overline{A+B})$. Let us see the truth table of the two input NOR gate.

From the truth table shown on as Table 2.27, we can see that output of a NOR gate is 1 if and only if all inputs are 0. If any one of the inputs is 1 it produces an output 0. This is the inverse operation of an OR gate. So we can say that ***a NOR gate is an inverted OR gate.***

A	B	Y= $\overline{(A+B)}$
0	0	1
0	1	0
1	0	0
1	1	0

Table 2.27 : NOR truth table

The logical symbol of NOR gate is shown in Figure 2.19. Note that the NOR symbol is an OR symbol with a small bubble at the output.

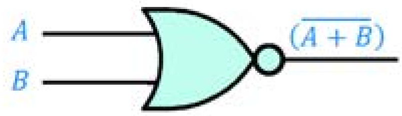


Fig. 2.19 : NOR gate

2.10.3 Implementation of basic gates using NAND and NOR

We can design all basic gates (AND, OR and NOT) using NAND or NOR gate alone. Let us see the implementation of basic gates using NAND gate.

NOT gate using NAND gate

We can implement a NOT gate (inverter) using a NAND by applying the same signal to both inputs of a NAND gate as shown in Figure 2.20.

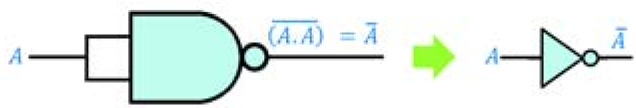


Fig. 2.20 : NOT gate using NAND gate

Proof:

A NAND A = $(\overline{A.A})$

= $\overline{A}$ Since A.A = A

The truth table shown as Table 2.28 is the proof for obtaining NOT gate using NAND gate.

A	AA	$(\overline{A.A})$	$\overline{A}$
0	0	1	1
1	1	0	0

Table 2.28 : Proof using truth table

AND gate using NAND gate

We can implement an AND gate by using a NAND gate followed by another NAND gate to invert the output as shown in Figure 2.21.

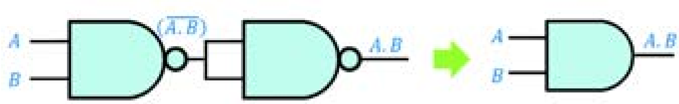


Fig. 2.21 : AND gate using NAND gate

Proof

We know that A NAND B $= \overline{(A.B)}$

(A NAND B) NAND (A NAND B) $= \overline{(A.B)}$ NAND $\overline{(A.B)}$

$= \overline{((\overline{AB}).(\overline{AB}))}$

$= (\ \overline{\overline{(AB)}}\ )$ Since A.A = A

$=$ A.B   Since $\overline{\overline{(A)}} = A$

Table 2.29 shows the proof for obtaining AND gate using NAND gate with the help of the truth table.

A	B	A.B	$\overline{(A.B)}$	$\overline{(A.B)} . \overline{(A.B)}$	$\overline{((\overline{AB}).(\overline{AB}))}$
0	0	0	1	1	0
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	0	0	1

Table 2.29 : Proof using truth table

OR gate using NAND gate

The OR gate is replaced by a NAND gate with all its inputs complemented by NAND gate inverters as shown in Figure 2.22.

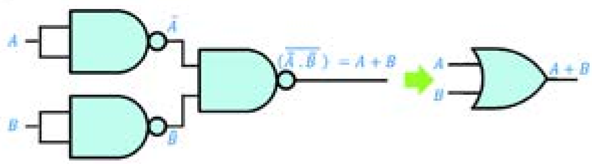


Fig. 2.22 : OR gate using NAND gate

Proof:

A NAND A $= \overline{(A.A)}$

$= \overline{A}$

Similarly, B NAND B $= \overline{B}$

Therefore, (A NAND A) NAND (B NAND B) $= \overline{A}$ NAND $\overline{B}$

$= \overline{(\overline{A}.\overline{B})}$

$= \overline{\overline{A}} + \overline{\overline{B}}$ Since $\overline{(A.B)} = \overline{A} + \overline{B}$

$= A + B$ Since $\overline{\overline{(A)}} = A$

Table 2.30 shows the proof for obtaining OR gate using NAND gate with the help of truth table.

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	$(\overline{\overline{A}.\overline{B}})$	A + B
0	0	1	1	1	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	0	1	1

Table 2.30 : Proof using truth table

Thus, the NAND gate is a universal gate since it can implement AND, OR and NOT operations. Now let us see implementation of basic gates by using another universal gate, the NOR gate.

NOT gate using NOR gate

We can implement a NOT gate (inverter) using a NOR by applying the same signal to both inputs of a NOR gate as shown in Figure 2.23.

Fig 2.23 : NOT gate using NOR gate

Proof:

$$\text{A NOR A} = (\overline{A+A})$$
$$= \overline{A} \quad \text{Since } A+A=A$$

Table 2.31 shows the proof for obtaining NOT gate using NOR gate with the help of truth table.

A	A+A	$(\overline{A+A})$	$\overline{A}$
0	0	1	1
1	1	0	0

Table 2.31 : Proof using truth table

OR gate using NOR gate

We can implement an OR gate by using a NOR gate followed by another NOR gate to invert the output as shown in Figure 2.24.

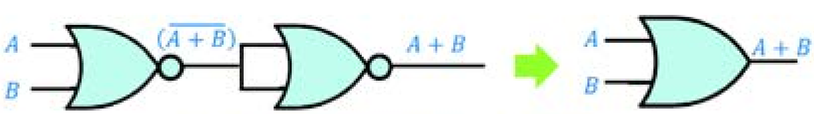


Fig 2.24: OR gate using NOR gate

Proof:

We know that A NOR B $= (\overline{A+B})$

(A NOR B) NOR (A NOR B) $= (\overline{A+B})$ NOR $(\overline{A+B})$

$$= (\overline{(\overline{A+B})+(\overline{A+B})})$$

$= ((\overline{\overline{A+B}}))$ Since A+A=A

$= A+B$ Since $(\overline{\overline{A}})=A$

Table 2.32 shows the proof of obtaining OR gate using NOR gate with the help of the truth table.

A	**B**	**A+B**	$(\overline{A+B})$	$(\overline{A+B}) + (\overline{A.B})$	$\overline{((\overline{A+B}).(\overline{A+B}))}$
0	0	0	1	1	0
0	1	1	0	0	1
1	0	1	0	0	1
1	1	1	0	0	1

Table 2.32 : Proof using truth table

AND gate using NOR gate

The AND gate is replaced by a NOR gate with all its inputs complemented by NOR gate inverters as shown in Figure 2.25.

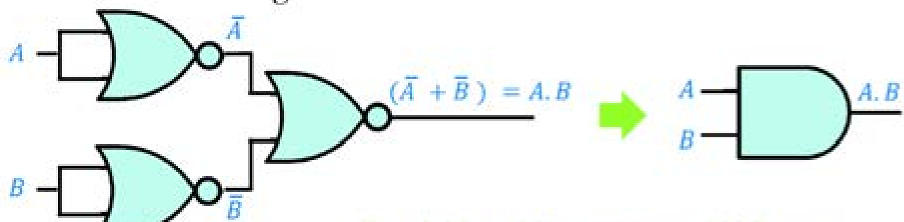


Fig. 2.25 : AND gate using NOR gate

Proof

A NOR A $= (\overline{A+A}) = \overline{A}$

Similarly, B NOR B $= (\overline{B+B}) = \overline{B}$

Therefore, (A NOR A) NOR (B NOR B) $= \overline{A}$ NOR $\overline{B}$

$= \overline{(\overline{A}+\overline{B})}$

$= (\overline{\overline{A}}).(\overline{\overline{B}})$ Since $(\overline{A+B}) = \overline{A}.\overline{B}$

$= A.B$ Since $\overline{\overline{A}} = A$

Thus, the NOR gate is also a universal gate since it can implement the AND, OR and NOT operations. Table 2.33 represents the proof for obtaining AND gate using NOR gate with the help of truth table.

A	**B**	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$	$\overline{(\overline{A}+\overline{B})}$	**A.B**
0	0	1	1	1	0	0
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	0	1	1

Table 2.33 : Proof using truth table

Check yourself

1. Draw logic circuits for the Boolean expression $X+\overline{Y}$.
2. Which gates are called universal gates?
3. ______ gate produces low (0) output if any one of the input is high(1)

   (a)OR (b) AND (c) NAND (d) NOR
4. A NAND B = ____________.

   (a) A+B (b) A.B (c) $(\overline{A+B})$ (d) $(\overline{A.B})$

Let us sum up

Different methods of data representation were discussed in this chapter. Before discussing data representation of numbers, we introduced different number systems and their conversions. After the discussion of integer and floating point number representation we have mentioned different methods for character, sound, image and sound data representation. We have also discussed the concept of Boolean algebra in detail including logical operators, logic gates and laws of Boolean algebra. We concluded the chapter by introducing methods for designing basic logic circuits and by discussing the importance of universal gates in circuit designing.

Learning outcomes

After the completion of this chapter the learner will be able to

- explain the characteristics of different number systems.
- convert one number system to another.
- perform binary arithmetic.
- represent numbers and characters in computer memory.
- list the formats of sound, image and video file formats.
- identify the concept of Boolean algebra.
- explain the working of logical operators and logic gates with the help of examples.
- state and prove basic postulates and laws of Boolean algebra.
- design circuits for simple Boolean expressions.
- implement basic gates using universal gates.

Sample questions

Very short answer type

1. What is the place value of 9 in $(296)_{10}$?
2. Find octal equivalent of the decimal number 55.
3. Find missing terms in the following series
   a) 101_2, 1010_2, 1111_2, ______, ______
   b) 15_8, 16_8, 17_8, ______, ______.
   c) 18_{16}, $1A_{16}$, $1C_{16}$, ______, ______.
4. If $(X)_2 - (1010)_2 = (1000)_2$ then find X.
5. Name the coding system that can represent almost all the characters used in the human languages in the world.
6. Find out the logical statement(s) from the following .
   a) Why are you late?
   b) Will you come with me to market ?
   c) India is my country.
   d) Go to class room.
7. List three basic logic gates.
8. Which gate is called inverter?
9. List two complimentarity Laws.
10. The Boolean expression $\overline{(A+B)}$ represents ______ gate.
    a) AND b) NOR c) OR d) NAND

Short answers type

1. Define the term data representation.
2. What do you mean by a number system? List any four number systems.
3. Convert the following numbers into the other three number systems:
   a) $(125)_8$ b) 98 c) $(101110)_2$ d) $(A2B)_{16}$
4. Find the equivalents of the given numbers in the other three number systems.
   a) $(7F.1)_{16}$ b) $(207.13)_8$ c) 93.25 d) $(10111011.1101)_2$
5. If $(X)_2 = (Y)_8 = (Z)_{16} = (28)_{10}$ Then find X, Y and Z.
6. Arrange the following numbers in descending order
   a) $(101)_{16}$ b) $(110)_{10}$ c) $(111000)_2$ d) $(251)_8$
7. Find X , if $(X)_2 = (10111)_2 + (11011)_2 - (11100)_2$

8. What are the methods of representing integers in computer memory?
9. Represent the following numbers in sign and magnitude method, 1's complement method and 2's complement method

   a) -19 b) +49 c) -97 d) -127
10. Find out the integer which is represented as $(10011001)_2$ in sign and magnitude method.
11. Explain the method of representing a floating point number in 32 bit computer.
12. What are the methods of representing characters in computer memory?
13. Briefly explain the significance of Unicode in character representation.
14. Match the following:

A		B	
i)	If any input is 1 output is 1	a)	NAND
ii)	If an input is 0 output is 0	b)	OR
iii)	If any input is 0 output is 1	c)	NOR
iv)	If any input is 1 output is 0	d)	AND

15. Find dual of following Boolean expressions

    a) $X.Y+Z$ b) $A.C+A.1+A.C$ c) $(A+0).(A.1.\overline{A})$
16. Find complement of following Boolean expressions

    a) $\overline{A}\,\overline{B}$ b) $\overline{A.B}+\overline{C.D}$
17. Construct logic circuit for the following Boolean expression.

    (i) $\bar{a}\bar{b}+c$ (ii) $ab+\bar{a}b+\overline{ab}$ (iii) $(a+\bar{b}).(\bar{a}+\bar{b})$
18. Why are NAND and NOR gates called universal gates? Justify with an example.

Long answer type

1. Briefly explain different methods for representing numbers in computer memory.
2. Briefly explain different methods for representing characters in computer memory.
3. What are the file formats for storing image, sound and video data?
4. Give logic symbol, Boolean expression and truth table for three input AND gate.
5. Prove that NOR gate is a universal gate by implementing all the basic gates.

3

Components of the Computer System

Key concepts

- **Data processing**
- **Functional units of a computer**
- **Hardware**
  - o Processors
  - o Motherboard
  - o Memory - primary storage, secondary storage
  - o Use of memory in computer
  - o Input output devices
- **e-Waste**
  - o Disposal methods
  - o Green computing
- **Software**
  - o System software (operating system, language processors, utility software)
  - o Application software (general purpose, specific purpose)
  - o Free software and open source software
  - o Freeware and shareware
  - o Proprietary software
- **Humanware / Liveware**

We are familiar with computers and their uses in today's world. Computer can be defined as a fast electronic device that accepts data, processes it as per stored instructions and produces information as output. This chapter presents an overview of the basic design of a computer system: how the different parts of a computer system are organised and various operations are performed to do a specific task. We know that a computer has two major components - hardware and software. Hardware refers to all physical components associated with a computer system while software is a set of instructions for the hardware to perform a specific task. When we use computers to solve any problem in real life situations, we define the tasks required to process data for generating information. This chapter presents the concepts of data processing at first and discusses how the functional units of a computer help data processing. Then various hardware components are presented followed by electronic waste, its disposal methods and the concept of green-computing. A detailed classification of software is listed along with different types of computer languages. We will also discuss the concepts of open source, freeware, shareware and proprietary software.

3.1 Data processing

Data processing refers to the operations or activities performed on data to generate information. Data denotes raw facts and figures such as numbers, words, amount, quantity, etc. that can be processed or manipulated. Information is a meaningful and processed form of data. It adds to our knowledge and helps in making decisions. Data processing proceeds through six stages as in Figure 3.1.

(a) Capturing data
(b) Input of data
(c) Storage of data
(d) Processing / manipulating data
(e) Output of information
(f) Distribution of information

Let us take a close look at these stages.

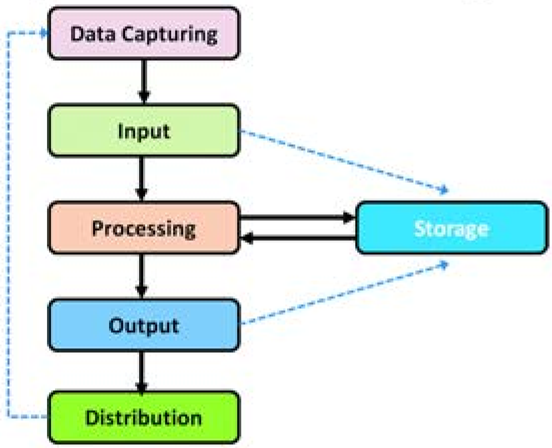


Fig. 3.1 : Data processing stages

Capturing Data: This is the first stage in data processing. Here a proforma, known as the source document, is designed to collect data in proper order and format. This document is used for data collection.

Input: In this stage, the data collected through the source documents is fed to the computer for processing. But now a days, in many cases, the data are directly fed into the computer without using source documents.

Storage: In data processing, the input data are stored before processing. The information obtained after processing may also be stored.

Process: The data stored in computers is retrieved for processing. Various operations like calculation, classification, comparison, sorting, filtering, summarising, etc. may be carried out as part of processing.

Output: The processed data is obtained in this stage in the form of information. The output may be stored for future reference as it may be used for generating some other information in another context.

Distribution of information: The information obtained from the output stage is distributed to the beneficiaries. They take decisions or solve problems according to the information.

We have seen the activities involved in data processing. Computers are designed in such a way that it can be involved in these activities. Let us see how the functional units of a computer are organised.

3.2 Functional units of a computer

Even though computers differ in size, shape, performance and cost over the years, the basic organisation of a computer is the same. As we discussed in Chapter 1, it is based on a model proposed by John Von Neumann, a mathematician and computer scientist. It consists of a few functional units namely, Input Unit, Central Processing Unit, Storage Unit and Output Unit as shown in Figure 3.2. Each of these units is assigned to perform a particular task. Let us discuss the functions of these units.

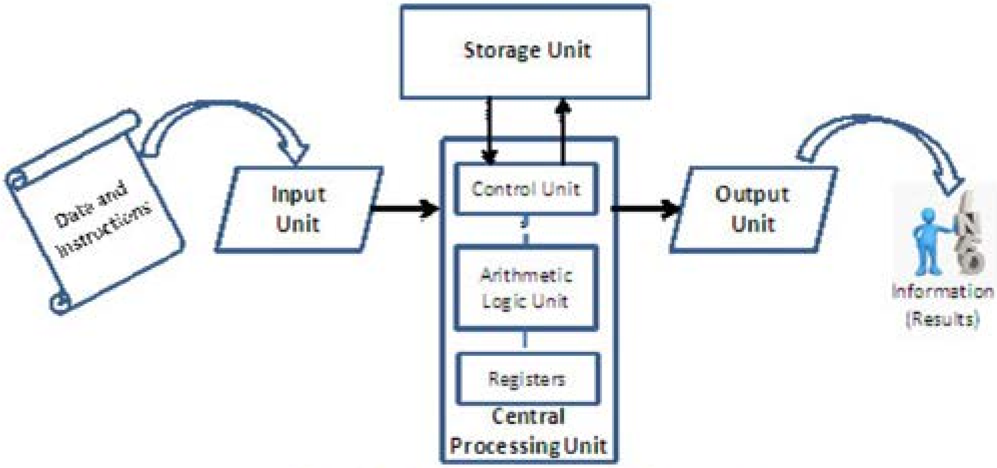


Fig. 3.2 : Basic organisation of computer

a. Input unit

The collected data and the instructions for their processing are entered the computer through the input unit. They are stored in the memory (storage unit). The data may be in different forms like number, text, image, audio, video, etc. A variety of devices are available to input the data depending on its nature. Keyboard, mouse, scanner, mic, digital camera, etc. are some commonly used input devices. In short, the functions performed by input unit are as follows:

1. Accepts instructions and data from the outside world.
2. Converts these instructions and data into a form acceptable to the computer.
3. Supplies the converted instructions and data to the computer for processing.

b. Central Processing Unit (CPU)

The CPU is the brain of the computer. In a human body, all major decisions are taken by the brain, and other parts of the body function as directed by the brain. Similarly, in a computer system, all major calculations and comparisons are made inside the CPU. It is also responsible for activating and controlling the operations of other units of the computer. The functions of CPU are performed by three components - Arithmetic Logic Unit (ALU), Control Unit (CU) and Registers.

i. Arithmetic Logic Unit (ALU)

The actual operations specified in the instructions are carried out in the Arithmetic Logic Unit (ALU). It performs calculations and logical operations such as comparisons and decision making. The data and instructions stored in the storage unit are transferred to the ALU and the processing takes place in it. Intermediate results produced by the ALU are temporarily transferred back to the storage and are retrieved later when needed for further processing. Thus there is a data flow between the storage and the ALU many times before the entire processing is completed.

ii. Control Unit (CU)

Each of the functional units has its own function, but none of these will perform the function until it is asked to. This task is assigned to the control unit. It invokes the other units to take charge of the operation they are associated with. It is the central nervous system that manages and coordinates all other units of the computer. It obtains instructions from the program stored in the memory, interprets the operation and issues signals to the unit concerned in the system to execute them.

iii. Registers

These are temporary storage elements that facilitate the functions of CPU. There are variety of registers; each is designated to store unique items like data, instruction, memory address, results, etc.

c. Storage unit

The data and instructions entered in the computer through input unit are stored inside the computer before actual processing starts. Similarly, the information or results produced after processing are also stored inside the computer, before transferring to the output unit. Moreover, the intermediate results, if any, must also be stored for further processing. The storage unit of a computer serves all these purposes. In short, the specific functions of storage unit are to hold or store:

1. Data and instructions required for processing.
2. Intermediate results for ongoing processing.
3. Final results of processing, before releasing to the output unit.

The storage unit comprises of two types of storages as detailed below:

i. **Primary storage:** It is also known as main memory. It is again divided into two – Random Access Memory (RAM) and Read Only Memory (ROM). RAM holds instructions, data and intermediate results of processing. It also holds the recently produced results of the job done by the computer. ROM contains instructions for the start up procedure of the computer. The Central Processing

Unit can directly access the main memory at a very high speed. But it has limited storage capacity.

ii. **Secondary storage:** It is also known as auxiliary storage and it takes care of the limitations of primary storage. It has huge storage capacity and the storage is permanent. Usually we store data, programs and information in the secondary storage, but we have to give instruction explicitly for this. Hard disk, CDs, DVDs, memory sticks, etc. are some examples.

d. Output unit

The information obtained after data processing is supplied to the outside world through the output unit in a human-readable form. Monitor and printer are the commonly used output devices. The functions performed by output unit can be concluded as follows:

1. Receives the results produced by the CPU in coded form.
2. Converts these coded results to human-readable form.
3. Supplies the results to the outside world.

3.3 Hardware

We know that a computer system consists of hardware and software. The term hardware represents the tangible and visible parts of a computer, which consists of some electro mechanical components. These hardware components are associated with the functional units of a computer. Let us discuss some of these components.

3.3.1 Processors

In the previous section, we learned that CPU (Central Processing Unit) is responsible for all computing and decision making operations and coordinates the working of a computer. The performance of a CPU determines the overall performance of the computer. Since CPU is an Integrated Circuit (IC) package which contains millions of transistors and other components fabricated into a single silicon chip, it is also referred as microprocessor. Figure 3.3 shows the processors developed by some manufacturers. A CPU is usually plugged into a large socket on the main circuit board (the motherboard) of the computer. Since heat is generated when the CPU works, a proper cooling system is provided with a heat sink and fan. The commonly used processors are Intel core i3, core i5, core i7, AMD Quadcore, etc.

Fig. 3.3 : Different Processors

Registers are storage locations inside the CPU, whose contents can be accessed more quickly by the CPU than

Every computer contains an internal clock that regulates the rate at which instructions are executed. The CPU requires a fixed number of clock ticks (or clock cycles) to execute each instruction. The faster the clock, the more the instructions the CPU can execute per second. Another factor is the architecture of the chip. The number of bits the processor can process at one time is called word size. Processors with many different word sizes exist: 8-bit, 16-bit, 32-bit, 64-bit, etc.

other memory. Registers are temporary storage areas for instructions or data. They are not a part of memory; rather they are special additional storage locations that offer the advantage of speed. Registers work under the direction of the control unit to accept, hold and transfer instructions or data and perform arithmetic or logical operations at high speed. It speeds up the execution of programs. Important registers inside a CPU are:

i. **Accumulator:** The accumulator is a part of the arithmetic/logic unit (ALU). This register is used to store intermediate result while performing arithmetic and logical operations. It is also called register A.

ii. **Memory Address Register (MAR)**: It stores the address of a memory location to which data is either to be read or written by the processor.

iii. **Memory Buffer Register (MBR)**: It holds the data, either to be written to or read from the memory by the processor.

iv. **Instruction Register (IR):** The instructions to be executed by the processor are stored in the Instruction Register.

v. **Program Counter (PC):** It holds the address of the next instruction to be executed by the processor.

3.3.2 Motherboard

A motherboard is a large Printed Circuit Board (PCB) to which all the major components including the processor are integrated. It also provides expansion slots for adding additional circuit boards like memory, graphics card, sound card, etc. (refer Figure 3.4). The motherboard must be compatible with the processor chosen.

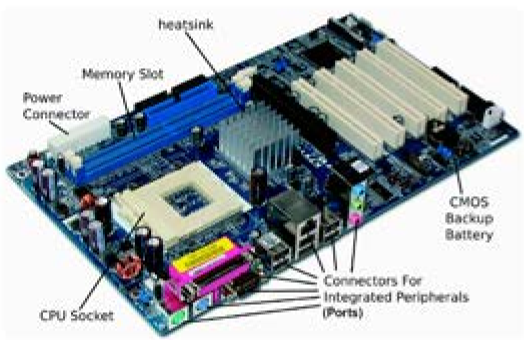


Fig. 3.4 : Motherboard

3.3.4 Peripherals and ports

Peripherals are devices that are attached to a computer system to enhance its capabilities. Ports on the motherboard are used to connect external devices. Peripherals include input and output devices, external storage and communication devices. These devices communicate with the motherboard through the ports like VGA, PS/2, USB, Ethernet, HDMI, etc. that are available on the motherboard. Figure 3.5 shows some ports used in personal computers.

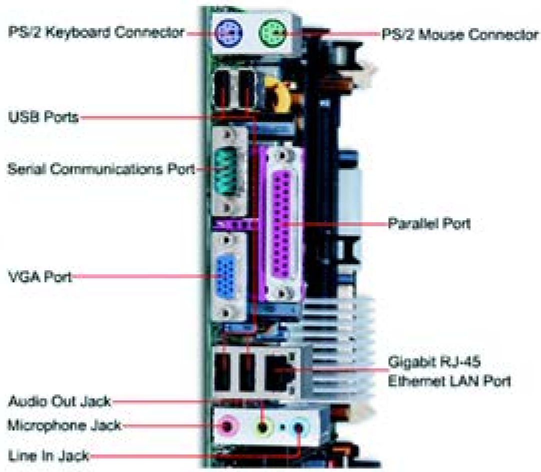


Fig. 3.5 : Ports

3.3.5 Memory

Memory is a place where we can store data, instructions and results temporarily or permanently. Memory can be classified into two - primary memory and secondary memory. Primary memory holds data, intermediate results and results of ongoing jobs temporarily. Secondary memory, on the other hand, holds data and information permanently. Before learning more about memory, let us discuss the different memory measuring units. The measuring units are:

Binary Digit = 1 Bit	1 MB (Mega Byte) = 1024 KB
1 Nibble = 4 Bits	1 GB (Giga Byte) = 1024 MB
1 Byte = 8 Bits	1 TB (Tera Byte) = 1024 GB
1 KB (Kilo Byte) = 1024 Bytes	1 PB (Peta Byte) = 1024 TB

a. Primary storage

Primary memory is a semiconductor memory that is accessed directly by the CPU. It is capable of sending and receiving data at high speed. This includes mainly three types of memory such as RAM, ROM and cache memory.

i. Random Access Memory (RAM)

RAM, shown in Figure 3.6 refers to the main memory that microprocessor can read from and write into. Data can be stored and retrieved at random from anywhere within the RAM, no matter where the data is. Data or instructions to be processed by the CPU must be placed in the RAM. The contents of RAM are lost when power is switched off. Therefore, RAM is a volatile memory. Storage capacity of RAM is 2 GB and above.

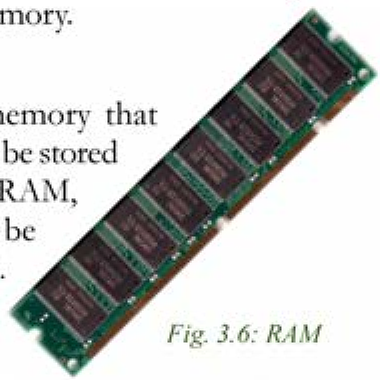


Fig. 3.6: RAM

The speed of a RAM refers to how fast the data in memory is accessed. It is measured in Mega Hertz (MHz). When a computer is in use, its RAM contains the following:

1. The operating system software.
2. The application software currently being used.
3. Any data that is being processed.

ii. Read Only Memory (ROM)

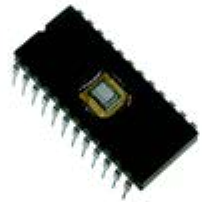


Fig.: 3.7 : ROM Chip

ROM is a permanent memory that can perform only read operations and its contents cannot be easily altered. ROM is non-volatile; the contents are retained even after the power is switched off. ROM, shown in Figure 3.7, is used in most computers to hold a small, special piece of 'boot up' program known as Basic Input Output System (BIOS). This software runs when the computer is switched on or 'boots up'. It checks the computer's hardware and then loads the operating system. There are some modified types of ROM that include:

1. PROM - Programmable ROM which can be programmed only once.
2. EPROM - Erasable programmable ROM that can be rewritten using ultra violet radiation.
3. EEPROM - Electrically Erasable Programmable ROM which can be rewritten electrically.

Table 3.1 shows the comparison between RAM and ROM.

RAM	ROM
• It is faster than ROM. • It stores the operating system, application programs and data when the computer is functioning. • It allows reading and writing. • It is volatile, i.e. its contents are lost when the device is powered off.	• It is a slower memory. • It stores the program required to boot the computer initially. • Usually allows reading only. • It is non-volatile, i.e. its contents are retained even when the device is powered off.

Table 3.1 : RAM - ROM comparison

iii. Cache memory

Cache memory is a small and fast memory between the processor and RAM (main memory). Frequently accessed data, instructions, intermediate results, etc. are stored in cache memory for quick access. When the processor needs to read from or write to a location in RAM, it first checks whether a copy of that data is in the cache. If so, the processor immediately reads the cache, which is much faster than reading from the RAM. Cache is more expensive than RAM, but it is worth getting a CPU and motherboard with built-in cache in order to maximise system performance.

b. Secondary or Auxiliary storage

Secondary memory is permanent in nature. Unlike the contents of RAM, the data stored in these devices does not vanish when power is turned off. Secondary memory is much larger in size than RAM, but is slower. It stores programs and data but the processor cannot access them directly. Secondary memory is also used for transferring data or programs from one computer to another. It can also act as a backup. The major categories of storage devices are magnetic, optical and semiconductor memory.

i. Magnetic storage devices

Magnetic storage devices use plastic tape or metal/plastic disks coated with magnetic materials. Data is recorded magnetically in these devices. Read/write heads are used to access data from these devices. Some of the popular magnetic storage devices are magnetic tapes, hard disks, etc.

ii. Optical storage devices

Optical disk is a data storage medium which uses low-powered laser beam to read and write data into it. It consists of an aluminum foil sandwiched between two circular plastic disks. Data is written on a single continuous spiral in the form of pits and lands. The laser beam reads this pits and lands as 0s and 1s. Optical disks are very cheap to produce in large quantities and are popular secondary storage media. The main types of optical disks are CD, DVD and Blu-Ray.

iii. Semi-conductor storage (Flash memory)

Flash drives use EEPROM chips for data storage. They do not contain any moving parts and hence they are shockproof. Flash memory is faster and durable when compared to other types of secondary memory. The drawback is that they are limited to a certain number of write cycles. The different variants of flash memories are USB flash drives and flash memory cards. Figure 3.8 shows different types of flash memories.

Fig. 3.8: Flash drive and memory cards

To see how registers, primary memory and secondary storage work together in a computer, let us use the analogy of making a salad in our kitchen. Suppose we have:

- A refrigerator where we store vegetables for the salad
- A counter where we place all vegetables before putting them on the cutting board for chopping.
- A cutting board on the counter where we chop vegetables.
- A recipe that details what vegetables to chop.
- The corners of the cutting board are kept free for partially chopped piles of vegetables that we intend to chop more or to mix with other partially chopped vegetables.
- A bowl on the counter where we mix and store the salad.
- Space in the refrigerator to put the mixed salad after it is made.

The process of making the salad is then: bring the vegetables from the fridge to the counter top; place some vegetables on the chopping board according to the recipe; chop the vegetables, possibly storing some partially chopped vegetables temporarily on the corners of the cutting board; place all the chopped vegetables in the bowl and keep it back in the fridge if not served on the dinner table.

In this context the refrigerator serves as secondary (hard disk) storage. It can store high volumes of vegetables for long periods of time. The counter top functions like the computer's motherboard - everything is done on the counter (inside the computer). The cutting board is the ALU - the work gets done there. The recipe is the control unit - it tells you what to do on the cutting board (ALU). Space on the counter top is the equivalent of RAM - all required vegetables must be brought from the fridge and placed on the counter top for fast access. Note that the counter top (RAM) is faster to access than the fridge (disk), but cannot hold as much, and cannot hold it for long periods of time. The corners of the cutting board where we temporarily store partially chopped vegetables are equivalent to the registers. The corners of the cutting board are very fast to access for chopping, but cannot hold much. The salad bowl is like a cache memory, it is for storing chopped vegetables to be temporarily removed from the corners of the cutting board (as there is too much) or the salad waiting to be taken back to the fridge (putting data back on a disk) or to the dinner table (outputting the data to an output device).

3.3.6 Role of memories in computers

Let us consider the case of a payroll program to calculate the salary of an employee. The data for all the employees is available in the hard disk. All the data about a particular employee is taken to the RAM and from there data related to salary calculation - bonuses, deductions, etc. is taken to the cache. The data representing the hours worked and the rate of pay is moved to their respective registers. Using data on the hours worked and the rate of pay, ALU makes calculations based on instructions from control unit. For further calculations, it moves the overtime hours, bonuses, etc. from cache to registers. As the CPU finishes calculations about one employee, the data about the next employee is brought from secondary storage into RAM, then cache and eventually into the registers.

Figure 3.9 depicts the hierarchy of different memories according to the storage capacity and access speed.

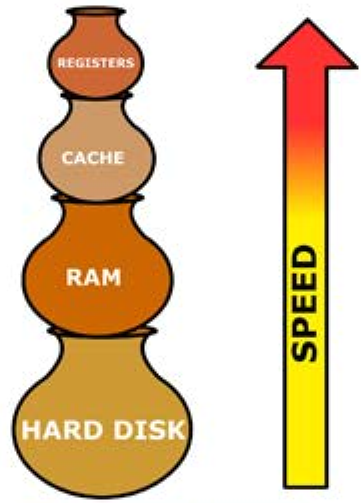


Fig. 3.9 : Memory hierarchy

Table 3.2 summarises the characteristics of various kinds of data storage in the storage hierarchy.

Storage	Speed	Capacity	Relative Cost	Volatile
Registers	Fastest	Lowest	Highest	Yes
Cache	More Fast	Low	Very High	Yes
RAM	Very Fast	Low/Moderate	High	Yes
Hard Disk	Moderate	Very High	Very Low	No

Table 3.2 : Comparison of different characteristics of various types of memories

Check yourself

1. The fastest memory in a computer is _______ .
2. Define data processing.
3. What is cache memory?
4. What is the use of program counter register?
5. What is the use of ALU?

3.3.7 Input/Output devices

The computer will be of no use unless it is able to communicate with the outside world. Input/output devices are required for users to communicate with the computer. In simple terms, input devices feed data and instructions into the computer and output devices presents information from a computer system. These input/output devices are connected to the CPU through various ports or with the help of wireless technology. Since they reside outside the CPU, they are called peripherals. The following table shows various **input devices** and their uses.

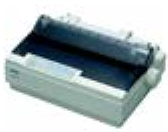
Device	Features / Uses
Keyboard 	Allows the user to input text data consisting of alphabets, numbers and other characters. Detects the key pressed and generates the corresponding ASCII code which can be recognised by the computer.Wired and wireless keyboards are available.
Mouse 	A small handheld device used to position the cursor or move the pointer on the computer screen by rolling it over a mouse pad / flat surface. Different types of mouse are ball, optical and laser mouse. Wireless mouse is also available.
Light pen 	A pointing device shaped like a pen. Has the advantage of 'drawing' directly onto the screen. Used by engineers, artists, fashion designers for Computer Aided Designing (CAD) and drawing purposes.
Touch screen 	Allows the user to operate/make selections by simply touching on the display screen. It can also be operated using a stylus which gives more precision.
Graphic tablet 	Consists of an electronic writing area and a special 'pen' that works with it.Allows artists to create graphical images with actions similar to traditional drawing tools.
Joystick 	Used to playing video games, control training simulators and robots. Has a vertical stick which can move in any direction and a button on top that is used to select the option pointed by the cursor.
Microphone 	Accepts sound which is analogue in nature as input and converts it to digital format. The digitised sound can be stored in the computer for later processing or playback.

Scanner 	Allows capturing of information, like pictures or text and converting it into a digital format that can be edited using a computer. Quality of the image depends on the resolution of the scanner. Different variants of scanners are flat bed, sheet feed and hand held scanner. Optical Character Recognition (OCR) software is used to recognise the text in an image scanned and convert it into text, which can be edited by a text editor.
Optical Mark Reader (OMR) 	Another scanning device that reads predefined positions and records where marks are made on the printed form. Useful for applications in which large numbers of hand-filled forms need to be processed quickly with great accuracy, such as objective type tests and questionnaires.
Barcode/Quick Response (QR) code reader 	A bar code is a set of vertical lines of different thicknesses and spacing that represent a number. Barcode readers are used to input data from such set of barcodes. Hand-held scanners, mobile phones with camera and special software are used as barcode readers. QR (Quick Response) code is similar to barcodes. Barcodes are single dimensional where as QR codes are two dimensional. The two dimensional way of storing data allows QR code to store more data than a standard barcode. This code can store website URLs, plain text, phone numbers, email addresses and any other alphanumeric data. The QR code can be read using a barcode reader or a mobile with a camera and special software installed.
Magnetic Ink Character Recognition (MICR) Reader 	MICR readers are used in banks for faster electronic clearing of cheques. The lower portion of a cheque contains cheque number, branch code, bank code, etc. printed in a special font using an ink containing iron oxide particles. Iron oxide has magnetic properties. MICR reader can easily recognise these characters by magnetically charging them while scanning. This MICR data along with the image of the cheque is send to the cheque drawer's (the person who issues the cheque) branch to transfer the amount. This reduces errors in data entry and speeds up money transfer.
Biometric sensor 	Identifies unique human physical features with high accuracy. It is an essential component of a biometric system which uses physical features like fingerprints, retina, iris patterns, etc. to identify, verify and authenticate the identity of the user.

Smart card reader 	Smart card is a plastic card that stores and transacts data. It may contain a memory or a micro processor. Used in banking, healthcare, telephone calling, electronic cash payments and other applications. These are used to access data in a smart card.
Digital camera 	Takes pictures and videos and converts it to the digital format. The images are stored in the memory and can be transferred to computer. Web camera is a compact and less expensive version of a digital camera. It is used in computers for video calling, video chatting, etc. It does not have an internal memory.

Table 3.3 : Input devices and their uses

Now let us see some **output devices** and their features. Table 3.4 shows various output devices and their uses.

Device	Features / Uses
Visual Display Unit (VDU) 	Display devices include CRT monitors, LCD monitors, TFT monitors, LED monitors, gas plasma monitors, Organic Light Emitting Diode (OLED) Monitors, etc. Information shown on a display device is called soft copy. The size of a monitor is measured diagonally across the screen, in inches.
LCD projector 	An LCD projector is a type of video projector for displaying video, images or computer data on a large screen or other flat surface. A beam of high-intensity light which travels through thousands of shifting pixels in an LCD is focused by a lens on the surface.
Printer  	Used to produce hardcopy output. The output printed on paper is known as hardcopy. Classified as Impact or Non-impact printers. Dot-matrix uses impact mechanism. It can print carbon copies with less printing cost. Speed is measured in number of characters printed in a unit of time and is represented as characters per second (cps), lines per minute (lpm) or pages per minute (ppm). These printers are slow and noisy. Inkjet printers are non-impact printers that form the image on the page by spraying tiny droplets of ink from the print head. Ink jet printers are inexpensive, but the cost of ink cartridges makes them costly to operate in the long run. Laser printers are non-impact printers that

 	produce good quality images. Monochrome and color laser printers are available. Color laser printers use multiple color toner cartridges to produce color output and are expensive. Laser printers are faster and their speed is rated in pages per minute (ppm). Thermal printer is a non-impact printer that produces a printed image by selectively heating heat sensitive thermal paper when it passes over the thermal print head. The coating turns black in the areas where it is heated, producing an image. It is popular as a portable printer.
Plotter 	A plotter is an output device used to produce hardcopies of graphs and designs on the paper. A plotter is typically used to print large-format graphs or maps such as construction maps, engineering drawings and big posters. Plotters are of two types: Drum plotters and Flatbed plotters. A drum plotter is also known as Roller plotter. A flatbed plotter is also known as Table plotter.
3D printer 	A 3D printer is an output device used to print 3D objects. It can produce different kinds of objects, in different materials, using the same printer. The 3D printing process turns the object to be printed, into thousands of tiny little slices. It then prints it from the bottom to top, slice by slice. Those tiny layers stick together to form a solid object.
Audio output device 	The audio output is the ability of the computer to produce sound. Speakers are the output devices that produces sound. It is connected to the computer through audio ports.

Table 3.4 : Output devices and their uses

We have seen different types of printers. Table 3.5 shows a comparison on various characteristics of these printers.

Features	**Laser Printers**	**Inkjet Printers**	**Thermal Printers**	**Dot Matrix Printers**
Printing material used	Ink powder	Liquid ink	Heat sensitive paper	Ink soaked ribbon
How it prints	It fuses the powder on the paper through heating.	It sprays liquid ink on paper through microscopic nozzles.	Thermal paper is passed over the thermal print head.	Pins are pushed against ribbon on paper.

Printing speed	20 pages per minute	6 pages per minute	150 mm per second	30-550 characters per second
Quality	Printing quality is good. Best for black and white.	Printing quality is good, especially for smaller fonts.	Poor quality printing of images. Good quality text printing.	Poor printing quality for images. In terms of text, printing is good.
Advantages	Quiet, prints faster, high print quality.	Quiet, high print quality, no warm up time, device cost is less.	Quiet, fast, smaller, lighter, consume less power and portable.	Cheaper to print as ribbon is cheap. Carbon copy possible.
Disadvantages	More susceptible to paper jams. Toner is expensive. Device itself is expensive.	Ink is expensive, ink is not waterproof and nozzle is prone to clogging.	Requires special thermal quality paper. Poor quality printing.	Initial purchase is expensive, prints are not fast, makes noise.

Table 3.5 : Comparison of printers

3.4 e-Waste

e-Waste refers to electronic products nearing the end of their 'useful life'. Electronic waste may be defined as discarded computers, office electronic equipment, entertainment devices, mobile phones, television sets and refrigerators. The used electronics which are destined for reuse, resale, salvage, recycling or disposal are also considered as e-waste.

Nowadays electronics is part of modern life – desktops, laptops, cell phones, refrigerators, TVs and a growing number of other gadgets. Every year we buy new, updated equipments to satisfy our needs. More than 300 million computers and one billion cell phones are produced every year. All of these electronics goods become obsolete or unwanted, often, within two or three years of purchase. This global mountain of waste is expected to continue growing at 8% per year.

Rapid changes in technology, changes in media, falling prices and planned obsolescence have resulted in a fast-growing surplus of electronic waste around the globe. It is estimated that 50 million tons of e-Waste are produced each year. Only 15-20% of e-Waste is recycled, the rest of these materials go directly into landfills and incinerators. Sale of electronic products in countries such as India and China and across continents such as Africa and Latin America are set to rise sharply over the next 10 years.

3.4.1 Why should we be concerned about e-Waste?

Electronic waste is not just waste. It contains some very toxic substances, such as mercury, lead, cadmium, brominated flame retardants, etc. The toxic materials can cause cancer, reproductive disorders and many other health problems, if not properly managed. It has been estimated that e-Waste may be responsible for up to 40% of the lead found in landfills. Important hazardous chemicals, their sources and consequences are listed in Table 3.6.

Chemical	Source	Consequence
Lead	Found as solder on printed circuit boards and in computer monitor glass.	Lead can cause damage to the central and peripheral nervous systems, blood systems and kidneys in humans.
Mercury	Found in printed circuit boards, LCD screen backlights.	Affect a baby's growing brain and nervous system. Adults can suffer organ damage, mental impairment and a variety of other symptoms.
Cadmium	Found in chip resistors and semiconductors.	Cause various types of cancer. Cadmium can also accumulate in the kidney and harm it.
BFRs-Brominated Flame Retardants	Found in printed circuit boards and some plastics.	These toxins may increase the risk of cancer.

Table 3.6 : Hazardous chemicals, its source and consequence

3.4.2 What happens to the e-Waste?

Unfortunately, an incredibly small percentage of e-waste is recycled. Even when we take it to a recycling center it is often not actually recycled – in the way most of us expect. CRTs have a relatively high concentration of lead and phosphors both of which are necessary for the display. The United States Environmental Protection Agency (EPA) includes discarded CRT monitors in its category of 'hazardous household waste'.

The majority of e-Waste is most often dumped or burned – either in formal landfills and incinerators or informally dumped or burned. These inappropriate disposal methods for electronic waste fail to reclaim valuable materials or manage the toxic materials safely. In effect, our soil, water and air are easily contaminated.

Fig. 3.10 : Defective and obsolete electronic items

e-Wastes should never be disposed with garbage and other household wastes. This should be segregated at the site and sold or donated to various organisations. Considering the severity of the e-Waste problem, it is necessary that certain management options be adopted by government, industries and the public to handle the bulk e-Wastes.

Realising the growing concern over e-Waste, Central Pollution Control Board (CPCB) of Government of India has formulated 'The e-Waste (Management & Handling) Rules, 2011' and are effective from 01-05-2012. These rules shall apply to every producer, consumer, collection centre, dismantler and recycler of e-Waste involved in the manufacture, sale and processing of electrical and electronic equipment or components. The implementation and monitoring of these guidelines shall be done by the State Pollution Control Boards concerned.

Government of Kerala has introduced strict measures for safe collection and disposal of e-Waste through a government order. The government has defined the role of manufacturers, local bodies and the Pollution Control Board (PCB) in safe disposal of e-Waste. Under the Extended Producer Responsibility, manufacturers of electrical and electronic goods will be required to take back used products from consumers directly or through agents or introduce buyback arrangement. They will also have to supply the e-Waste to authorised recycling units. Consumers have been directed to return used products of known brands to the manufacturers or deposit them at the collection centresset up by local bodies.The PCB will be required to identify agencies for recycling or disposal of e-Waste and organise awareness programmes on e-Waste disposal.

3.4.3 e-Waste disposal methods

The following methods can be used for disposing e-Waste.

a. **Reuse:** It refers to second-hand use or usage after the equipment has been upgraded or modified. Most of the old computers are passed on to relatives/ friends or returned to retailers for exchange or for money. Some computers are also passed on to charitable institutions, educational institutions, etc. Inkjet cartridges and laser toners are also used after refilling. This method reduces the volume of e-Waste generation.

b. **Incineration:** It is a controlled and complete combustion process in which the waste is burned in specially designed incinerators at a high temperature in the range of 900 to 1000 degree Celsius.

c. **Recycling of e-Waste:** Recycling is the process of making or manufacturing new products from a product that has originally served its purpose. Monitors, keyboards, laptops, modems, telephone boards, hard drives, compact disks, mobiles, fax machines, printers, CPUs, memory chips, connecting wires and cables can be recycled.

d. **Land filling:** It is one of the widely used but not recommended method for the disposal of e-Waste.

Role of students in e-Waste disposal

- Stop buying unnecessary electronic equipments.
- When electronic equipments get faulty try to repair it instead of buying a new one.
- Try to recycle electronic equipments by selling them or donating them to others extending their useful life and keeping them out of the waste stream.
- If you really need to buy new electronics, choose items with less hazardous substances, greater recycled content, higher energy efficiency, longer life span, and those that will produce less waste.
- Visit the manufacturer's website or call the dealer to find out if they have a take back programme or scheme for your discarded electronics.
- If the device is battery-operated, buy rechargeable instead of disposable batteries.
- Buy products with good warranty and take back policies.

3.4.4 Green computing or Green IT

Green computing is the study and practice of environmentally sustainable computing or IT. Green computing is the designing, manufacturing, using and disposing of computers and associated components such as monitors, printers, storage devices, etc., efficiently and effectively with minimal or no impact on the environment.

One of the earliest initiatives towards green computing was the voluntary labelling program known as 'Energy Star'. It was conceived by the Environmental Protection Agency (EPA) in 1992 to promote energy efficiency in hardware of all kinds. The Energy Star label has become a common sight, especially in notebook computers and displays. Similar programmes have been adopted in Europe and Asia. The commonly accepted Energy Star symbol is shown in Figure 3.11.

Fig. 3.11 : Energy Star label

Government regulation is only a part of an overall green computing idea. The work habits of computer users and business firms have to be modified to minimise adverse impact on the global environment. Here are some steps that can be taken:

- Turn off computer when not in use.
- Power-on the peripherals such as laser printers only when needed
- Use power saver mode.
- Use laptop computers rather than desktop computers whenever possible.
- Take printouts only if necessary.
- Use liquid crystal display (LCD) monitors rather than cathode ray tube (CRT) monitors.
- Use hardware/software with Energy Star label.
- Dispose e-Waste according to central, state and local regulations.
- Employ alternative energy sources like solar energy.

The environmentally responsible and eco-friendly use of computers and their resources is known as green computing.

How to make computers green?

The features that are important in making a computer greener include size, efficiency and materials. Smaller computers are greener because they use fewer materials and require less electricity to run. Efficient use of energy is also an important component of a green computer. Smaller computers such as laptops are more energy-efficient than bigger models and LCD screens use much less energy than the older CRT models. The use of hazardous materials such as lead and mercury should be minimised.

To promote green computing the following four complementary approaches are employed:

Green design: Designing energy-efficient and eco-friendly computers, servers, printers, projectors and other digital devices.

Green manufacturing: Minimising waste during the manufacturing of computers and other components to reduce the environmental impact of these activities.

Green use: Minimising the electricity consumption of computers and peripheral devices and using them in an eco-friendly manner.

Green disposal: Reconstructing used computers or appropriately disposing off or recycling unwanted electronic equipment.

Check yourself

1. The environmentally responsible and eco-friendly use of computers and their resources is known as ________.
2. The process of making or manufacturing new products from the product that has originally served its purpose is called ________.
3. The labelling programme to promote energy efficiency in computers and their resources is called ________.
4. List any two input and output devices each.

Let us do

1. *Conduct a survey in your locality to study the impact of e-Waste on the environment and health of the people and write a report.*
2. *Discuss the importance of green computing.*

3.5 Software

Software is a general term used to denote a set of programs that help us to use the computer system and other electronic devices efficiently and effectively. If hardware is said to form the body of a computer system, software is its mind or soul. There are two types of software:

- System software
- Application software

3.5.1 System software

It is a set of one or more programs designed to control the operations of a computer. They are general programs designed to assist humans in the use of computer system by performing tasks such as controlling the operations, move data into and out of a computer system and to do all the steps in executing application programs. In short, system software supports the running of other software, its communication with other peripheral devices. It helps users to use computer in an effective manner. It implies that system software helps to manage resources of the computer. Figure 3.12 depicts how system software interfaces between user and hardware.

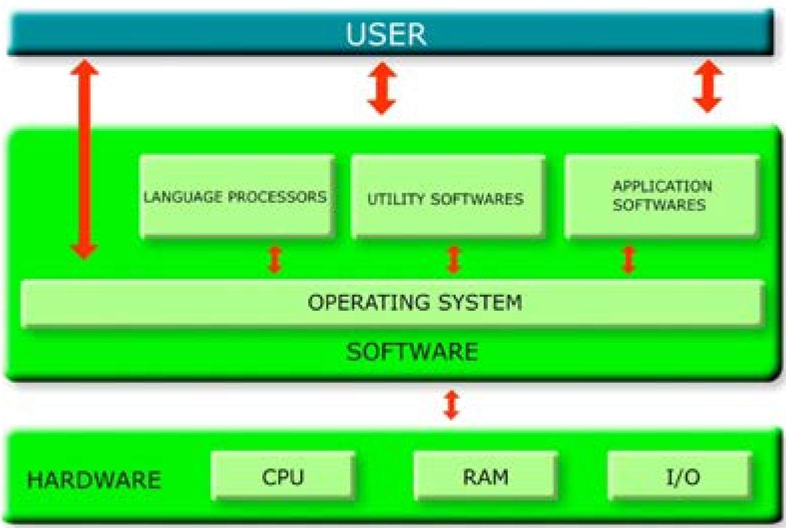


Fig.3.12: Software with user and hardware interface

System software is a set of system programs which aids in the execution of a general user's computational requirements on a computer system. The following are the components of system software:

a. Operating system
b. Language processors
c. Utility software

a. Operating system

Operating system is a set of programs that acts as an interface between the user and computer hardware. The primary objective of an operating system is to make the computer system convenient to use. Operating system provides an environment for user to execute programs. It also helps to use the computer hardware in an efficient manner.

Operating system controls and co-ordinates the operations of a computer. It acts as the resource manager of the computer system as shown in Figure 3.13. Operating system is the most important system software. It is the first program to be loaded from hard disk in the computer and it resides in the memory till the system is shut down. It tries to prevent errors and the improper use of computer.

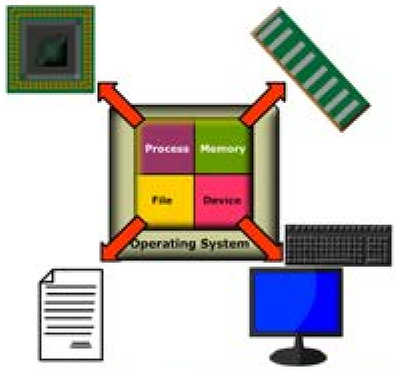


Fig. 3.13 : Operating System as a resource manager

The major functions of an operating system are process management, memory management, file management, security management and command interpretation.

i. Process management

By the term process we mean a program in execution. The process management module of an operating system takes care of the allocation and deallocation of processes and scheduling of various system resources to the different requesting processes.

ii. Memory management

Memory management is the functionality of an operating system which handles or manages primary memory. It keeps track of each and every memory location whether it is allocated to some process or it is free. It calculates how much memory is to be allocated to each process and allocates it. It de-allocates memory if it is not needed further.

iii. File management

The file management module of an operating system takes care of file related activities such as organising, naming, storing, retrieving, sharing, protection and recovery.

iv. Device management

Device management module of an operating system performs the management of devices attached to the computer. It handles the devices by combining both hardware and software techniques. The OS communicates with the hardware device via the device driver software. Examples of various operating systems are DOS, Windows, Unix, Linux, Mac OS X, etc.

b. Language processors

We know that natural languages are the medium of communication among human beings. Similarly, in order to communicate with the computer, the user also needs to have a language that should be understood by the computer. Computer languages may be broadly classified into low level languages and high level languages.

Low-level languages are described as machine-oriented languages. In these languages, programs are written using the memory and registers available on the computer. Since the architecture of computer differs from one machine to another, there is separate low level programming language for each type of computer. Machine language and assembly language are the different low level languages.

Machine language: We know that a computer can understand only special signals, which are represented by 1s and 0s. These two digits are called binary digits. The language, which uses binary digits, is called machine language. Writing a program in machine language is definitely very difficult. It is not possible to memorise a long string of 0s and 1s for every instruction.

Assembly language: Assembly language is an intermediate-level programming language. Assembly languages use mnemonics. Mnemonic is a symbolic name given to an operation. For example ADD for addition operation, SUB for subtraction operation, etc. It is easier to write computer programs in assembly language as compared to machine language. It is machine dependent and programmer requires knowledge of computer architecture.

High Level Languages (HLL): These are like English languages and are simpler to understand than the assembly language or machine language. High level language is not understandable to the computer. A computer program written in a high level language is to be converted into its equivalent machine language program. So these languages require a language translator (compilers or interpreters) for conversion. Examples of high-level programming languages are BASIC, C, C++, Java, etc.

Need for language processor

The programs consisting of instructions to the computer, written in assembly language or high level language are not understood by the computer. We need language processors to convert such programs into low level language, as computer can only understand machine language. Language processors are the system programs that translate programs written in high level language or assembly language into its equivalent machine language.

Types of language processors

- **Assembler:** Assembly languages require a translator known as assembler for translating the program code written in assembly language to machine language. Because computer can interpret only the machine code instruction, the program can be executed only after translating. An assembler is highly machine dependent.
- **Interpreter:** Interpreter is another kind of language processor that converts a HLL program into machine language line by line. If there is an error in one line, it reports and the execution of the program is terminated. It will continue the translation only after correcting the error. BASIC is an interpreted language.
- **Compiler:** Compiler is also a language processor that translates a program written in high level language into machine language. It scans the entire program in a single run. If there is any error in the program, the compiler provides a list of error messages along with the line number at the end of the compilation. If there are no syntax errors, the compiler will generate an object file. Translation using compiler is called compilation. After translation compilers are not required in memory to run the program. The programming languages that have a compiler are C, C++, Pascal, etc.

Figure 3.14 shows process involved in the translation of assembly language and high level language programs into machine language programs

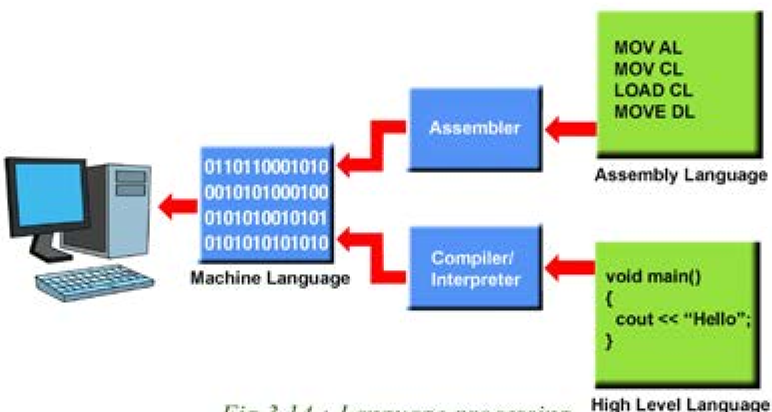


Fig.3.14 : Language processing

c. Utility software

Utility software is a set of programs which help users in system maintenance tasks and in performing tasks of routine nature. Some of the utility programs with their functions are listed below:

- **Compression tools:** Large files can be compressed so that they take less storage area. These compressed files can be decompressed into its original form when needed. Compression of files is known as zipping and decompression is called unzipping. WinZip, WinRAR, etc. are examples.

- **Disk defragmenter:** Disk defragmenter is a program that rearranges files on a computer hard disk. The files are arranged in such a way that they are no longer fragmented. This enables the computer to work faster and more efficiently.
- **Backup software:** Backup means duplicating the disk information so that in an event of disk failure or in an event of accidental deletion, this backup may be used. Backup utility programs facilitates the backing up of disk.
- **Antivirus software:** A computer virus is a program that causes abnormality in the functioning of a computer. Antivirus software is a utility program that scans the computer system for viruses and removes them. As new viruses are released frequently, we have to make sure that latest antivirus versions are installed on the computer. Most of the antivirus programs provide an auto-update feature which enables the user to download profiles of new viruses so as to identify and inactivate them. Norton Antivirus, Kaspersky, etc. are examples of antivirus programs.

3.5.2 Application software

Software developed for specific application is called application software. It includes general purpose software packages and specific purpose software. GIMP, Payroll System, Air line reservation System, Tally, etc. are examples of application software.

a. General purpose software packages

General purpose software are used to perform tasks in a particular application area. Such software is developed keeping in mind the various requirements of its users. They provide a vast number of features for its users. General purpose software is classified as word processors, spreadsheet software, presentation software, database software and multimedia software.

- **Word processing software:** Word Processing software is designed for creating and modifying documents. It helps to create, edit, format and print textual matters easily. Formatting features include different font settings, paragraph settings, bullets and numbering, alignments and more. In addition to this it can check spelling and grammar in the document, insertion of pictures, charts and tables. We can specify headers and footers for every page in the document. The most popular examples of this type of software are MS Word, Open Office Writer, Apple iWork Pages, etc.
- **Spreadsheet software:** Spreadsheet software allows users to perform calculations using spreadsheets. They simulate paper worksheets by displaying multiple cells that make up a grid. It also allows us to insert drawing objects in the worksheet and create different types of charts for graphical representation

of numerical data. Microsoft Excel, Open Office Calc, Lotus 1-2-3 and Apple iWork Numbers are some examples of spreadsheet software.

- **Presentation software:** The software that is used to display information in the form of a slide show is known as presentation software. Presentation software allows preparing slides containing pictures, text, animation, video and sound effects. Microsoft PowerPoint, Apple iWork Keynote and Open Office Impress are examples for presentation software.
- **Database software:** Database is an organised collection of data arranged in tabular form. Database Management System (DBMS) consists of a collection of interrelated data and a set of programs to access those data. The primary goal of a DBMS is to provide an environment that is both convenient and efficient to use in retrieving and storing database information. They provide privacy and security to data and enforce standards for data. Examples of DBMS software are Microsoft Access, Oracle, Postgres SQL, My SQL, etc.
- **Multimedia software:** Multimedia is the integration of multiple forms of media. This includes text, graphics, audio, video, etc. Multimedia software can process information in a number of media formats. It is capable of playing media files. Some multimedia software allows users to create and edit audio and video files. Audio converters, audio players and video editing software are some forms of multimedia software. Examples are VLC Player, Adobe Flash, Real Player, Media Player, etc.

b. Specific purpose software

Specific purpose software is a highly specialised software designed to handle particular tasks. These are tailor-made software to satisfy the needs of an organisation or institution. It is also known as customised software. Since custom software is developed for a single customer, it can accommodate that customer's particular preferences and expectations.

Some examples of specific purpose application software are listed in Table 3.7.

Application Software	Purpose
Payroll System	Payroll system maintains the details of employees of an organisation and keeps track of their salary details.
Inventory Management System	It is used for tracking inventory levels, orders, sales and deliveries in a business firm.
Human Resource Management System	It is used for managing human resource in an organisation.

Table 3.7 : Examples of application software

Let us do

- *Discuss the classification of software.*
- *Compare and contrast the features of Linux and Windows operating systems with the help of your teacher and prepare short notes (Lab Demonstration). Discuss the role of utility software.*
- *Write short notes on the following:*
  Language processors
  General purpose software packages

Check yourself

1. Define operating system.
2. Give two examples for OS.
3. A program in execution is called ________.
4. Mention any two functions of OS
5. Name the software that translates assembly language program into machine language program.
6. Name the two different language processors which translate high level language programs into machine language programs.
7. Differentiate between compiler and interpreter.
8. DBMS stands for ____________.
9. Give two examples for customized software.
10. Duplicating disk information is called ________.

3.5.3 Free and open source software

Free and open source software gives the user the freedom to use, copy, distribute, examine, change and improve the software. Nowadays free and open source software is widely used throughout the world because of adaptable functionality, less overall costs, vendor independency, adherence to open standards, interoperability and security.

The Free Software Foundation (FSF) defines the four freedoms for free and open source software:

Freedom 0 : The freedom to run program for any purpose.

Freedom 1 : The freedom to study how the program works and adapt it to your needs. Access to source code should be provided.

Freedom 2 : The freedom to distribute copies of the software.

Freedom 3 : The freedom to improve the program and release your improvements to the public, so that the whole community benefits.

The following are some of the examples of free and open source software:

GNU/Linux: GNU/Linux is a computer operating system assembled under the model of free and open source software development and distribution. It was organised in the GNU project introduced in 1983 by Richard Stallman in the FSF.

GIMP: It stands for GNU Image Manipulation Program. It is an image editing software. It can be used for retouching photographs, creating and editing images. It supports graphic files of different formats and allows converting from one format to another.

Mozilla Firefox: It is one of the most popular web browsers created by the Mozilla Corporation. It provides added security features for safe browsing.

OpenOffice.org: It is a complete office suite that contains word processor (Writer) to prepare and format documents, spreadsheets (Calc) and presentations (Impress). It works on both Linux and Windows platforms.

3.5.4 Freeware and Shareware

Freeware refers to copyrighted computer software which is made available for use, free of charge, for an unlimited period.

The term shareware refers to commercial software that is distributed on a trial basis. It is distributed without payment and with limited functionality. Shareware is commonly offered in a downloadable format on the Internet. The distribution of this kind of software aims at giving users a chance to analyse the software before purchasing it. Some shareware works for a limited period of time only. Table 3.8 highlights a comparison between freeware and shareware:

Freeware	Shareware
• Freeware refers to software that anyone can download from the Internet and use for free.	• Shareware gives users a chance to try the software before buying it.
• All the features are free.	• All features are not available. To use all the features of the software, user has to purchase it.
• Freeware programs can be distributed free of cost.	• Shareware may or may not be distributed freely. In many cases, author's permission is needed to distribute the shareware.

Table 3.8 : Comparison of Freeware and Shareware

3.5.5 Proprietary software

Proprietary software is a computer program that is an exclusive property of its developer or publisher and cannot be copied or distributed without licensing agreements. It is sold without any access to source code and is therefore cannot be changed for improved by the user. Some examples of proprietary software are Microsoft Windows operating system, MS Office, Mac OS, etc.

3.6 Humanware or Liveware

Humanware or liveware refers to humans who use computer. It was used in computer industry as early as 1966 to refer to computer users, often in humorous contexts by analogy with software and hardware. It refers to programmers, systems analysts, operating staff and other personnel working in a computer system. Table 3.9 shows various categories of humanware and their job description.

Humanware	Job Description
System Administrators	Upkeep, configuration and reliable operation of computer systems; especially multi-user computers such as servers.
System Managers	Ensure optimal level of customer services and maintain expertise in all business unit systems and develop professional relationships with all vendors and contractors.
System Analysts	Design new IT solutions to improve business efficiency and productivity.
Database Administrators	Create, monitor, analyse and implement database solutions.
Computer Engineers	Design either the hardware or software of a computer system.
Computer Programmers	Write the code that computers read in order to operate properly.
Computer Operators	Oversee the running of computer systems, ensuring that the machines are running, physically secured and free of any bugs.

Table 3.9 : Categories of humanware with job description

Check yourself

1. An example of free and open source software is ______.
2. The software that give users a chance to try it before buying is _____.
3. What do you mean by free and open source software?
4. Give an example for proprietary software.
5. Give two examples of humanware.

Let us sum up

Data processing is a series of activities by which data is converted into information. The limitations of manual data processing are overcome by electronic data processing and computer is the best electronic data processing machine. A computer has five functional units such as input unit, storage unit, arithmetic and logic unit, control unit and output unit. This chapter provided a general overall introduction to computer organisation. Input and output devices, e-waste and its disposal methods and the importance of green computing were introduced. The classification of software and the need of operating system in a computer with its major functions were discussed. Following this, the categories of computer languages were presented. The concepts of open source, freeware, shareware, free software and proprietary software were also discussed in detail. The chapter concluded outlining the concept of humanware.

Learning outcomes

After the completion of this chapter the learner will be able to

- distinguish between data and information.
- identify various stages in data processing.
- explain basic organisation of computer system.
- recognise the different types of input and output devices.
- distinguish between system software and application software.
- identify the importance of e-Waste disposal.
- identify the importance of green computing concept.
- classify the different types of software.
- recognise the functions of operating system.
- use word processor, electronic spreadsheets and presentation software.
- classify the different types of computer languages.
- list the different types of utility software.
- promote open source software.
- explain the term humanware or liveware.

Sample questions

Very short answer type

1. What is data?
2. Processed data is known as ________.
3. What are the components of a digital computer?
4. Write the main functions of central processing unit.
5. What are the different types of main memory?
6. What is the advantage of EEPROM over EPROM?
7. When do we use ROM?
8. What is an input device? List few commonly used input devices.
9. What do you mean by an output device? List few commonly used output devices.
10. What is a storage device? List few commonly used storage devices.
11. What is the role of ALU?
12. What is a control unit?
13. What are registers? Write and explain any two of them.
14. Differentiate hard copy and soft copy.
15. What is e-Waste?
16. What is operating system?
17. What is a language processor?
18. Mention the categories of computer languages.
19. What is disk defragmenter?
20. Why is OS considered as a 'resource manager'?
21. What is proprietary software?
24. What do you mean by open source software?

Short answer type

1. Distinguish between data and information.
2. The application form for Plus One admission contains your personal details and your choice of groups and schools.
   (a) Identify the data and information in the admission process.
   (b) Explain how the information helps the applicants and school authorities.
   (c) Write down the activities involved in the processing of the data.

3. Briefly explain any three input devices.
4. Compare CRT with LED monitor
5. Differentiate between RAM and ROM
6. List and explain e-waste disposal methods.
7. Enumerate the steps that can be taken for the implementation of green computing philosophy.
8. What do you mean by customised software? Give examples.
9. Distinguish between low level and high level languages.
10. Differentiate compiler and interpreter.
11. Describe the use of electronic spreadsheets.
12. What is utility software? Give two examples.
13. Categorise the software given below into operating system, application packages and utility programs. Linux, Tally, WinZip, MS-Word, Windows, MS-Excel
14. Differentiate between freeware and shareware.
15. What are the four freedoms which make up free and open source software?
16. What do you mean by human-ware? Give any two examples.

Long answer type

1. Taking the case of a real life example, briefly describe the activities involved in each stage of data processing.
2. With the help of a block diagram, explain the functional units of a computer.
3. Describe in detail the various units of the Central Processing Unit.
4. Briefly explain various types of memory.
5. Explain classification of printers.
6. "e-Waste is hazardous to our health and environment." Justify the statement. List and explain the methods commonly used for e-Waste disposal.
7. Define the term green computing. List and explain the approaches that you can adopt to promote green computing concepts at all possible levels.
8. List and explain various categories of software.
9. Describe the use of various utility software.
10. Define the term 'operating system'. List and explain the major functions of operating system.
11. List and explain general purpose application software with examples.
12. Compare freeware and shareware.