Computer Programming

As per JNTU Kakinada R16 syllabus

Reema Thareja

Assistant Professor
Department of Computer Science
Shyama Prasad Mukherji College for Women
University of Delhi

OXFORD
UNIVERSITY PRESS

OXFORD
UNIVERSITY PRESS

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide. Oxford is a registered trade mark of
Oxford University Press in the UK and in certain other countries.

Published in India by
Oxford University Press
YMCA Library Building, 1 Jai Singh Road, New Delhi 110001, India

First published in 2016

ISBN-13: 978-0-19-947415-8
ISBN-10: 0-19-947415-X

Typeset in Times New Roman
by Pee-Gee Graphics, New Delhi
Printed in India by Magic International (P) Ltd, Greater Noida

I dedicate this book to my family and my uncle Mr B.L. Theraja, who is a well-known author

Preface

Computers are so widely used in our day-to-day lives that imagining a life without them has become almost impossible. They are not only used by professionals but also by children for interactively learning lessons, playing games, and doing their homework. Applications of the computer and its users are increasing by the day.

Learning computer fundamentals is a stepping stone to having an insight into how these machines work. Once the reader is aware of the basic terminology that is commonly used in computer science, he/she can then go on to develop useful computer programs that may help solve a user's problems.

Since computers cannot understand human languages, special programming languages are designed for this purpose. C is one such programming language. Being the most popular and successful programming language, it is used in several different software platforms such as system software and application software.

Many of the popular cross-platform programming languages, such as C++, Java, Python, Objective-C, Perl, and Ruby, and scripting languages, such as PHP, Lua, and Bash, borrow syntaxes, and functions from C.

Hence, mastering the C language is a prerequisite for learning such languages.

ABOUT THE BOOK

This book is designed as a textbook to meet the requirements of the latest syllabus of the first year computer programming course offered by the Jawaharlal Nehru Technological University (JNTU), Kakinada. The objective of this book is to introduce the students to the fundamentals of computers and the concepts of C programming language, and enable them to apply these concepts for solving real-world problems.

The book provides comprehensive coverage of all the relevant topics using simple language. It also contains useful annexures to various chapters including writing compiling and executing programs in Unix/ Ubuntu/ Linux, user-defined header files, and pointer declarations for additional information. Case studies, ASCII chart, important library functions, preprocessors, inline functions and bit-level programming are also provided to supplement the text.

Programming skill is best developed by rigorous practice. Keeping this in mind, the book provides a number of programming examples that would help the reader learn how to write efficient programs. To further enhance the understanding of the subject, there are numerous chapter-end exercises provided in the form of objective type questions, review questions, and programming problems.

ORGANIZATION OF THE BOOK

The book is organized into 9 chapters, 3 annexures, 2 case studies, and 5 appendices.

Chapter 1 provides an introduction to computer hardware and software. It introduces the concept of memory and its storage units such as bits and bytes. The chapter provides an insight into different programming languages and the generation through which these languages have evolved. Topics such as application software, system software, algorithms, flowcharts and the software development process are also presented in this chapter.

Chapter 2 covers the building blocks of the C programming language. The chapter discusses about data types, identifiers, constants, variables, and operators supported by the language. The chapter also discusses expressions, type conversion and casting, basic library functions, and I/O modifiers.

Annexure 1 demonstrates the steps to write, compile, and execute a C program in Unix, Linux, and Ubuntu environments.

KEY FEATURES OF THE BOOK AND THEIR BENEFITS

Features	Benefits
Topical coverage: Topics are arranged as per the latest **R16** syllabus of JNTU Kakinada.	Completely fulfils the syllabus requirements.
Presentation: Simple and lucid presentation of concepts supported with numerous illustrations.	This makes the text easier to comprehend for the readers.
Programming Examples: Plenty of programming examples along with their output and descriptions are provided in support of text.	These features will help readers understand the concept, logic, and syntaxes used while developing a program, thereby transforming them into efficient programmers.
Notes and Programming Tips: Every chapter includes 'notes' mentioning the important concepts to remember and 'programming tips' stating the do's and don'ts while developing a program.	This will help readers keep the critical things in mind and develop error-free programs.
Points to Remember and Glossary: A summary of important concepts and key terms with definitions are given at the end of each chapter.	This will help readers in quick recapitulation of the concepts discussed in that particular chapter.
Chapter-end Practice Questions: Chapter-end exercises include objective-type questions (fill in the blanks, true/false and multiple choice questions), review questions, programming exercises, and find the output and error questions. *Answers to the objective-type questions are given in Appendix E.*	These are provided to test readers' knowledge of the concepts, improve their programming skills as also help them apply and assimilate the concepts learnt.
Model Question Papers: 3 solved and 2 unsolved model papers are provided	This will help students prepare for their university examinations.

Chapter 3 explains important types of statements such as decision statements, iterative statements, control statement, break statement, and jump statement.

Case Study 1 includes two programs which harness the concepts learnt in Chapters 2 and 3.

Chapter 4 deals with declaring, defining and calling functions. The chapter also discusses the storage classes as well as variable scope in C. It ends with an important concept of recursive functions and a discussion on Tower of Hanoi problem.

Annexure 2 discusses how to create user-defined header files.

Chapter 5 provides a detailed explanation of arrays that includes one-dimensional, two- dimensional and multi-dimensional arrays. Finally, the operations that can be performed on such arrays are also explained.

Chapter 6 unleashes the concept of strings, also known as character arrays. The chapter not only focuses on reading and writing strings but also explains various operations that can be used to manipulate the character arrays.

Chapter 7 introduces the concept of pointers including pointer variables, pointer arithmetic, null and generic pointers. The chapter relates the use of pointers with arrays, strings, and functions. It also includes drawbacks of pointers and dynamic memory allocation in C.

Annexure 3 explains the process of deciphering pointer declarations. ***Case Study 2*** includes a program which shows how pointers can be used to access and manipulate strings.

Chapter 8 introduces two derived or user-defined data types—structure and union. The chapter includes the use of structures and unions with pointers, arrays and functions so that the inter-connectivity between the programming

techniques can be well understood. Typedef, enumerated data type, and bit fields are also covered in this chapter.

Chapter 9 discusses how data can be stored in files. The chapter discusses the opening, processing, and closing of files using a C program. These files are handled in text mode as well as binary mode for better clarity of the concepts.

The book also provides 5 appendices, namely, ***Appendix A***: ASCII Codes of Characters, ***Appendix B***: ANSI C Library Functions, ***Appendix C:*** Preprocessors and Inline Functions, ***Appendix D:*** Bit-level Programming and Bitwise Shift operators, and ***Appendix E***: Answers to Objective-type Questions.

ONLINE RESOURCES

For the benefit of faculty and students reading this book, additional resources are available online at india.oup.com/orcs/ 9780199474158.

For Faculty

- Solutions Manual
- Chapter-wise PPTs

For Students

- Lab Exercises
- Test Generator
- Projects
- Solutions to unsolved model question papers given in the book
- Extra Reading Material – Versions of C, Sparse matrix, Case study on Sorting

Comments and suggestions for the improvement of the book are welcome. Please send them to me at reemathareja@gmail.com.

Reema Thareja

Acknowledgements

The writing of this textbook was a mammoth task for which a lot of help was required from many people. Fortunately, I have had wholehearted support of my family, friends, and fellow members of the teaching staff and students at Shyama Prasad Mukherji College, New Delhi.

My special thanks would always go to my parents, Mr Janak Raj Thareja and Mrs Usha Thareja, and my siblings, Pallav, Kimi, and Rashi, who were a source of abiding inspiration and divine blessings for me. I am especially thankful to my son, Goransh, who has been very patient and cooperative in letting me realize my dreams. My sincere thanks go to my uncle, Mr B.L. Theraja, for his inspiration and guidance in writing this book.

I would like to express my gratitude to the following reviewers for their valuable suggestions and constructive feedback that helped in improving the book.

Venkata Ram Kumar Bonam	BVC Institute of Technology and Science, Amalapuram, Andhra Pradesh
G. Jena	BVC Engineering College, Odalarevu , Andhra Pradesh
Mrinalini Nunna	Andhra Loyola Institute of Engineering and Technology, Vijayawada, Andhra Pradesh
Ashok Nutalapati	Dhanekula Institute of Engineering and Technology, Vijayawada, Andhra Pradesh
M. Venkateswerarao	Malineni Perumallu Educational Society's Group of Institutions, Guntur, Andhra Pradesh
M.V.R. Narasimharao	BVC College of Engineering, Rajahmundry, Andhra Pradesh
P. Srinivas	JNTU College of Engineering Campus, Guntur, Andhra Pradesh
Rama Satish Aravapalli	DVR & Dr. HS MIC College of Technology, Kanchikacherla, Andhra Pradesh
N. Leelavathy	Pragati Engineering College, Surampalem, Andhra Pradesh
Nekuri Naveen	Sasi Institute of Technology and Engineering, Tadepalligudem, Andhra Pradesh
D. Haritha	SRK Institute of Technology, Vijayawada, Andhra Pradesh
K. Narasimha Rao	Vishnu Institute of Technology, Bhimavaram, Andhra Pradesh
Sripada Rama Sree	Aditya Engineering College, Surampalem, Andhra Pradesh
V. Venkateswara Rao	Sri Vasavi Engineering College, Tadepalligudem, Andhra Pradesh
M. Gargi	Vignan's Lara Institute of Technology and Science, Guntur, Andhra Pradesh
P. Srinivasa Rao	University College of Engineering, Jawaharlal Nehru Technological University, Guntur, Andhra Pradesh
P. R. Krishna Prasad	Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh
A. Rama Rao	Lendi Institute of Engineering and Technology, Vizinagaram, Andhra Pradesh
M. Venkata Rao	Chirala Engineering College, Chirala, Andhra Pradesh
Bhanu Prakash Battula	Tirumala Engineering College, Guntur, Andhra Pradesh
B. Renuka Devi	Vignan's Nirula Institute of Technology and Science for Women, Guntur, Andhra Pradesh
K. Venkat Rao	Vignan Institute of Technology, Visakhapatnam, Andhra Pradesh
Ravi Sankar	Vignan's LARA Institute of Technology and Science, Guntur, Andhra Pradesh
P. Ravikiran Varma	MVGR College of Engineering, Vizianagaram, Andhra Pradesh
B. Bhanu Pratap Reddy	Universal Engineering College, Guntur, Andhra Pradesh
Gopala Reddy Kallam	Ramachandra College of Engineering, Eluru, Andhra Pradesh

Last but not the least, I would like to thank the editorial team at Oxford University Press, India for their help and support.

Reema Thareja

Detailed Contents

UNIT I

UNIT II

UNIT III

UNIT IV

UNIT V

UNIT VI

ROAD MAP TO THE SYLLABUS

Computer Programming

(As per JNTU Kakinada R16 syllabus)

Syllabus	Chapter/ Section no.
UNIT I	
History and Hardware– Computer hardware, bits and bytes, components. ***Programming Languages–*** Machine language, assembly language, low- and high-level languages, procedural and object-oriented languages. Application and system software, the development of C algorithms, the software development process.	1. Computer History, Hardware, Software, Programming Languages, and Algorithms
UNIT II	
Introduction to C Programming– Identifiers, the main() function, the printf() function. ***Programming Style–*** Indentation, comments, data types**,** arithmetic operations, expression types, variables and declarations, negation, operator precedence and associativity, declaration statements, initialization. ***Assignment–*** Implicit type conversions, explicit type conversions (casts), assignment variations, mathematical library functions, interactive input, formatted output, format modifiers.	2. Introduction to C Programming
UNIT III	
Control Flow, Relational Expressions, Logical Operators: ***Selection–*** if-else statement, nested if, examples, multi-way selection– switch, else-if, examples. ***Repetition–*** Basic loop structures, pretest and posttest loops, counter-controlled and condition-controlled loops, the while statement, the for statement, nested loops, the do-while statement.	3. Conditional Flow, Relational Expressions and Logical Operators
UNIT IV	
Modular Programming– Function and parameter declarations, returning a value, functions with empty parameter lists, variable scope, variable storage class, local variable storage classes, global variable storage classes, pass by reference, passing addresses to a function, storing addresses, using addresses, declaring and using pointers. Case study– Swapping values. Recursion– mathematical recursion, recursion.	4. Modular Programming– Functions (For storing addresses, using addresses, declaring and using pointers, passing addresses to a function – *Also refer to Sections 7.3 and 7.7*)
UNIT V	
Arrays– One-dimensional arrays, input and output of array values, array initialization, arrays as function arguments, two-dimensional arrays, larger dimensional arrays–matrices.	5. Arrays
Strings– String fundamentals, string input and output, string processing, library functions.	6. Strings
UNIT VI	
Pointers – Concept of a pointer, initialization of pointer variables, pointers as function arguments, passing by address, dangling memory, address arithmetic, character pointers and functions, pointers to pointers, dynamic memory management functions, command line arguments	7. Pointers
Structures– Derived types, structures declaration, initialization of structures, accessing structures, nested structures, arrays of structures, structures and functions, pointers to structures, self-referential structures, unions, typedef, bit-fields.	8. Structures, Unions and Bit Fields (For derived types – *Also refer Section 2.11*)
Data Files– Declaring, opening, and closing file streams, reading from and writing to text files, random file access.	9. Data Files

CHAPTER 1

Computer History, Hardware, Software, Programming Languages, and Algorithms

Takeaways

- Characteristic of computers
- Types of computers
- Components of a computer
- CPU
- System software
- Generations of programming languages
- Pseudocodes
- History of computer
- Applications of computers
- Computer hardware and software
- Input and Output devices
- Translators
- Programming paradigms
- Algorithms
- Types of errors
- Stored program concept
- Basic architecture and organization of computer
- Computer memory
- Application software
- Software development process
- Flowcharts
- Testing and debugging

1.1 INTRODUCTION

We all have seen computers in our homes, schools, or colleges. In fact, in today's scenario we find computers in most aspects of daily life, and for some it is hard to even imagine a world without them. A computer is basically a machine that takes instructions and performs computations based on those instructions.

Nowadays computers come in different sizes. Their size may vary from very small to very large. In the past, computers were extremely large in size and required an entire room for installation. These computers consumed enormous amounts of power and were too expensive to be used for commercial applications. Therefore, they were used only for limited tasks, such as computing trajectories for astronomical or military applications. However, with technological advancements, the size of computers became smaller and their energy requirements lowered immensely. This opened the way for adoption of computers for commercial purposes.

These days, computers have become so prevalent in the market that all interactive devices such as cellular phones, global positioning system (GPS) units, portable organizers, automated teller machines (ATMs), and gas pumps work with computers.

1.2 WHAT IS A COMPUTER?

A computer is an electronic machine that takes instructions and performs computations based on those instructions. Before going into details, let us learn some key terms that are frequently used in computers.

Data Data is a collection of raw facts or figures.

Information Information comprises processed data to provide answers to *'who'*, *'what'*, *'where'*, and *'when'* type of questions.

Knowledge Knowledge is the application of data and information to answer *'how'* part of the question (Refer Figure 1.1).

Instructions Commands given to the computer that tells what it has to do are instructions.

Programs A set of instructions in computer language is called a program.

Software A set of programs is called software.

Hardware A computer and all its physical parts are known as hardware.

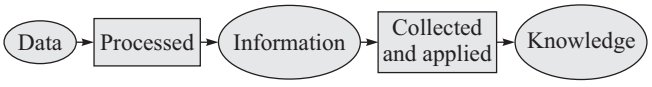


Figure 1.1 Data, information, and knowledge

1.3 HISTORY OF COMPUTERS

History of computers can be understood by looking into five generations. With each new generation of computers, there had been advancement in computer technology. The circuitry became smaller with enhanced speed, less consumption of power, and efficient memory.

Therefore, each generation of computer is characterized by a major technological development that has drastically changed the way in which computers operate.

First Generation (1942–1955)

Hardware Technology First generation computers were manufactured using thousands of vacuum tubes. Vacuum tube (as shown in Figure 1.2) is a device made of fragile glass.

Figure 1.2 Vacuum tube

Source: Vladyslav Danilin/ Shutterstock

Software Technology Programming was done in machine language or assembly language.

Used for Scientific applications

Examples ENIAC, EDVAC, EDSAC, UNIVAC I, IBM 701

Highlights

- They were the fastest calculating device of those times.
- Computers were too bulky and required a complete room for storage.
- Highly unreliable as vacuum tubes emitted a large amount of heat and burnt frequently.
- Required air-conditioned room for installation.
- Costly.
- Difficult to use.
- Required constant maintenance because vacuum tubes used filaments that had limited lifetime. Therefore, these computers were prone to frequent hardware failures.

Second Generation (1955–1964)

Hardware Technology Second generation computers were manufactured using transistors (as shown in Figure 1.3). Transistors were reliable, powerful, cheaper, smaller, and cooler than vacuum tubes.

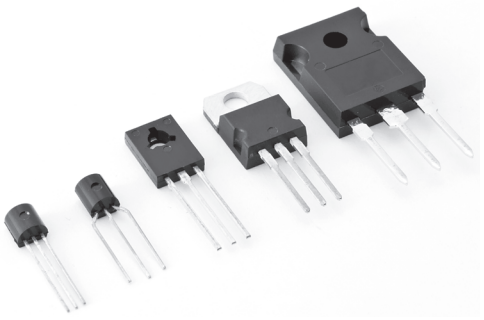


Figure 1.3 Transistors

Source: yurazaga/Shutterstock

Software Technology Programming was done in high level programming language.

Used for Scientific and commercial applications

Examples Honeywell 400, IBM 7030, CDC 1604, UNIVAC LARC

Highlights

- Faster, smaller, cheaper, reliable, and easier to use than the first generation computers.
- Consumed 1/10th the power consumed by first generation computers.

- Bulky in size and required a complete room for its installation.
- Dissipated less heat than first generation computers but still required air-conditioned room.
- Costly.
- Difficult to use.

Third Generation (1964–1975)

Hardware Technology Third generation computers were manufactured using integrated chips (ICs) as shown in Figure 1.4. ICs consist of several components such as transistors, capacitors, and resistors on a single chip to avoid wired interconnection between components. These computers used SSI and MSI technology. Minicomputers came into existence.

Note

Initially, ICs contained 10–20 components. This technology was called Small Scale Integration (SSI). Later it was enhanced to contain about 100 components. This was called MSI (Medium Scale Integration).

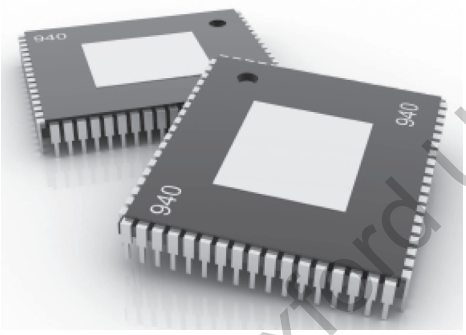


Figure 1.4 Integrated chip

Source: cooldesign/FreeDigitalPhotos.net

Software Technology Programming was done in high level programming language such as FORTRAN, COBOL, Pascal, and BASIC.

Used for Scientific, commercial, and interactive online applications.

Examples IBM 360/370, PDP–8, PADP-11, CDC6600

Highlights

- Faster, smaller, cheaper, reliable, and easier to use than the second generation computers.
- They consumed less power than second generation computers.
- Bulky in size and required a complete room for its installation.
- Dissipated less heat than second generation computers but still required air-conditioned room.
- Costly.
- Easier to use and upgrade.

Fourth Generation (1975–1989)

Hardware Technology Fourth generation computers were manufactured using ICs with LSI (Large Scale Integrated) and later with VLSI (Very Large Scale Integrated) technology as shown in Figure 1.5. Microcomputers came into existence, and use of personal computers became widespread during this period. High speed computer networks in the form of LANs, WANs, and MANs started growing. Besides mainframes, supercomputers were also used.

Figure 1.5 VLSI

Note

LSI contained 30,000 components on a single chip and VLSI technology had about 1 million electronic components on a single chip.

Software Technology Programming was done in high level programming language like C, C++. Graphical user interface (GUI) based operating system (like Windows) was introduced. It had icons and menus among other features to allow computers to be used as a general purpose machine by all users.

Used for Scientific, commercial, interactive online, and network applications.

Examples IBM PC, Apple II, TRS–80, VAX 9000, CRAY–1, CRAY–2, CRAY–X/MP

Highlights Faster, smaller, cheaper, powerful, reliable, and easier to use than the previous generation computers.

Fifth Generation (1989–Present)

Hardware Technology Fifth generation computers were manufactured using ICs with ULSI (Ultra Large Scale

Integrated) technology as shown in Figure 1.6. Use of Internet became widespread. Very powerful mainframes, desktops, portable laptops, smartphones are being used commonly. Super computers use parallel processing techniques.

Note

ULSI contained about 10 million electronic components on a single chip.

Software Technology Programming was done in high level programming language such as Java, Python, C#.

Figure 1.6 ULSI

Used for Scientific, commercial, interactive online, multimedia (graphics, audio, video), and network applications.

Examples IBM notebooks, Pentium PCs, SUM workstations, IBM SP/2, Param supercomputer.

Highlights

- Faster, smaller, cheaper, powerful, reliable, and easier to use than the previous generation computers.
- Speed of microprocessors and the size of memory are growing rapidly.
- High-end features available on mainframe computers in the fourth generation are now available on the microprocessors.
- Consume less power than computers of prior generations.
- Air-conditioned rooms required for mainframes and supercomputers but not for microprocessors.

1.4 CHARACTERISTICS OF COMPUTERS

The important characteristics of a computer (as shown in Figure 1.7) are given below:

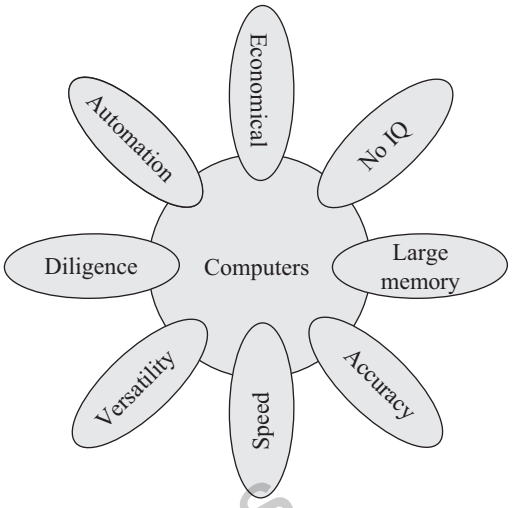


Figure 1.7 Characteristics of computer

Speed Computers can perform millions of operations in a single second. This means that a computer can process the data in blink of an eye which otherwise may take multiple days to complete. The speed of the computer is usually given in *nano second* and *pico second*, where

1 nano second=$1*10^{-9}$ second

1 pico second=$1*10^{-12}$ second

Accuracy Computers are a reliable electronic device. It never makes mistakes. It always gives accurate results provided that correct data and set of instructions are input to it. So in the advent of an error, only the user who has fed the incorrect data/program is responsible. If the input data is wrong, then the output will also be erroneous. In computer terminology, it is known as garbage-in garbage-out (GIGO).

Automatic Besides being very fast and accurate, computers are automatic devices that can perform without any user intervention. The user just needs to assign the task to the computer after which the computer automatically controls different devices attached to it and executes the program instructions one by one.

Diligence Computers can never get tired as humans do. It can continually work for hours without creating any error. If a large number of executions have to be made then each and every execution will require the same amount of time and accuracy.

Versatile Versatile means flexible. Today, computers are being used in our daily lives in different fields. For example, they are used as personal computers (PCs) for home use, for business-oriented tasks, weather forecasting,

space explorations, teaching, railways, banking, medicine, etc. On the PC that you use at home, you may play a game, compose and send e-mails, listen to music, etc. Therefore, computers are versatile devices as they can perform multiple tasks of different nature at the same time.

Memory Similar to humans, computers also have memory. Human beings cannot store everything in their memory and need secondary media, such as a notebook to record certain important things. Similarly, computers have internal memory (storage space) as well as external or secondary memory. While the internal memory of computers is very expensive and limited in size, the secondary storage is cheaper and bigger in size.

The computer stores a large amount of data and programs in the secondary storage space. The stored data and programs can be used whenever required. Secondary memory devices include CD, DVD, hard disk, pen drives, etc.

Note

When data and programs have to be used they are copied from the secondary memory into the internal memory (often known as RAM).

No IQ Although the trend today is to make computers intelligent by inducing artificial intelligence (AI) in them, they do not have any decision-making abilities of their own, that is, their IQ level is zero. They need guidance to perform various tasks.

Economical Today, computers are considered as short-term investment for achieving long-term gain. Using computers also reduces manpower requirements and leads to an elegant and efficient way for doing tasks. Hence, computers save time, energy, and money. When compared to other systems, computers can do more work in lesser time. For example, using the conventional postal system to send an important document takes at least 2–3 days, whereas the same information when sent using the Internet (e-mail) will be delivered instantaneously.

1.5 CLASSIFICATION OF COMPUTERS

Computers can be broadly classified into four categories based on their speed, amount of data that they can process, and price (refer to Figure 1.8). These categories are as follows:

- Supercomputers
- Mainframe computers
- Minicomputers
- Microcomputers

Supercomputers

Among the four categories, the supercomputer is the fastest, most powerful, and most expensive computer. Supercomputers were first developed in the 1980s to process large amounts of data and to solve complex scientific prob-lems. Supercomputers use parallel processing technology and can perform more than one trillion calculations in a second.

A single supercomputer can support thousands of users at the same time. Such computers are mainly used for weather forecasting, nuclear energy research, aircraft design, automotive design, online banking, controlling industrial units, etc. Some examples of supercomputers are CRAY-1, CRAY-2, Control Data CYBER 205, and ETA A-10.

Mainframe Computers

Mainframe computers are large-scale computers (but smaller than supercomputers). These are very expensive and need a very large clean room with air conditioning, thereby making them very costly to deploy. As with supercomputers, mainframes can also support multiple processors. For example, the IBM S/390 mainframe can

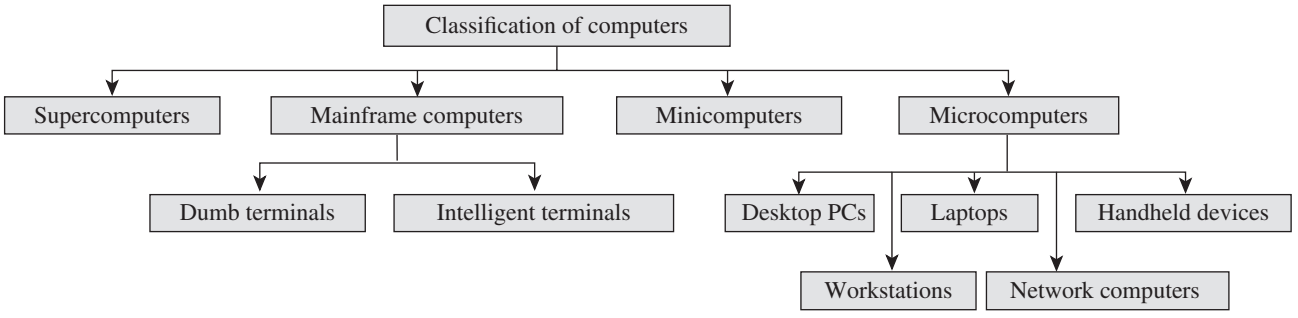


Figure 1.8 Classification of computers

support 50,000 users at the same time. Users can access mainframes by either using terminals or via PCs.

Mainframe computers are typically used as servers on the World Wide Web. They are also used in organizations such as banks, airline companies, and universities, where a large number of users frequently access the data stored in their databases. IBM is the major manufacturer of mainframe computers. Some examples of mainframe computers include IBM S/390, Control Data CYBER 176, and Amdahl 580.

Minicomputers

As the name suggests, minicomputers are smaller, cheaper, and slower than mainframes. They are called minicomputers because they were the smallest computer of their times. Also known as *midrange computers*, the capabilities of minicomputers fall between mainframe and personal computers.

Minicomputers are widely used in business, education, hospitals, government organizations, etc. While some minicomputers can be used only by a single user, others are specifically designed to handle multiple users simultaneously. Usually, single-user minicomputers are used for performing complex design tasks.

As with mainframes, minicomputers can also be used as servers in a networked environment, and hundreds of PCs can be connected to it.

The first minicomputer was introduced by Digital Equipment Corporation (DEC) in the mid-1960s. Other manufacturers of minicomputers include IBM Corporation (AS/400 computers), Data General Corporation, and Prime Computer.

Microcomputers

Microcomputers, commonly known as PCs, are very small and cheap. The first microcomputer was designed by IBM in 1981 and was named IBM-PC. Later on, many computer hardware companies copied this design and termed their microcomputers *PC-compatible*, which refers to any PC that is based on the original IBM PC design.

Another type of popular PC is designed by Apple. PCs designed by IBM and other PC-compatible computers have a different architecture from that of Apple computers. Moreover, PCs and PC-compatible computers commonly use the Windows operating system, while Apple computers use the Macintosh operating system (MacOS). PCs can be classified into the following categories:

Desktop PCs A desktop PC is the most popular model of PCs. The system unit of the desktop PC can be placed flat on a desk or table. It is widely used in homes and offices.

Laptops Laptops (Figure 1.9) are small microcomputers that can easily fit inside a briefcase. They are very handy and can easily be carried from one place to another. They may also be placed on the user's lap (thus the name). Hence, laptops are very useful, especially when going on long journeys. Laptops operate on a battery and do not always have to be plugged in like desktop computers.

Figure 1.9 Laptop

Source: You can more/Shutterstock

The memory and storage capacity of a laptop is almost equivalent to that of a desktop computer. As with desktop computers, laptops also have hard disk drives, USB drives, etc. For input, laptops have a built-in keyboard and a trackball/touchpad, which is used as a pointing device (as a mouse is used for a desktop PC).

Today, laptops have the same features and processing speed as the most powerful PCs. However, a drawback is that laptops are generally more expensive than desktop computers. These computers are very popular among business travellers.

Workstations Workstations are single-user computers that have the same features as PCs, but their processing speed matches that of a minicomputer or mainframe computer. Workstation computers have advanced processors, more RAM and storage capacity than PCs. Therefore, they are more expensive and powerful than a normal desktop computer.

Although workstation computers are widely used as powerful single-user computers by scientists, engineers, architects, and graphic designers, they can also be used as servers in a networked environment.

Network Computers Network computers have less processing power, memory, and storage than a desktop computer. These are specially designed to be used as terminals in a networked environment. For example, some

network computers are specifically designed to access data stored on a network (including the Internet and intranet).

Some network computers do not have any storage space and merely rely on the network's server for data storage and processing tasks. The concept of network computers had become popular in the mid-1990s when several variations of computers such as Windows terminals, NetPCs, and diskless workstations were widely used.

Network computers that are specifically designed to access only the Internet or intranet are often known as Internet PCs or Internet boxes. Some network computers used in homes do not even have a monitor. Such computers may be connected to a television, which serves as the output device. The most common example of a home-based network computer is Web TV, which enables the user to connect a television to the Internet. The other reason for the popularity of network computers is that they are cheaper to purchase and maintain than PCs.

Handheld Computers The mid-1990s witnessed a range of small personal computing devices that are commonly known as handheld computers, or mobile computers. These computers are called handheld computers because they can fit in one hand, while users can use the other hand to operate them. Handheld computers are very small in size, and hence they have small-sized screens and keyboards. These computers are preferred by business travellers and mobile employees whose jobs require them to move from place to place.

Some examples of handheld computers are as follows:

- Smartphones
- Tablet PCs

Smartphones These days, cellular phones are web-enabled telephones that have features of both analog and digital devices. Such phones are also known as smartphones because, in addition to basic phone capabilities, they also facilitate the users to access the Internet and send e-mails and faxes.

Tablet PCs A tablet PC (refer Figure 1.10) is a computing device that is smaller than a laptop, but bigger than a smartphone. Features such as user-friendly interface, portability, and touch screen have made them very popular in the last few years. These days, a wide range of high-performance tablets are available in the market. While all of them look similar from outside, they may differ in features such as operating system, speed of data connectivity, camera specifications, size of the screen, processing power, battery life, and storage capability.

Figure 1.10 Tablet

***Source*:** bloomua/Shutterstock/OUP Picture Bank

Some operating systems that are used in tablets are Android Jellybean (an open-source operating system built by Google), Windows 8, and iOS (developed by Apple).

While users can easily type directly on the surface of a tablet, some users prefer a wireless or bluetooth-connected keyboard. These days, tablets also offer an optional docking station with keyboards that transforms the tablet into a full-featured netbook.

Uses The following are the uses of tablet PCs:

- Viewing presentations
- Videoconferencing
- Reading e-books, e-newspaper
- Watching movies
- Playing games
- Sharing pictures, video, songs, documents, etc.
- Browsing the Internet
- Keeping in touch with friends and family on popular social networks, sending emails
- Business people use them to perform tasks such as editing documents, exchanging documents, taking notes, and giving presentations
- Tablets are best used in crowded places such as airports and coffee shops, where size and portability become more important.

Note

Tablets may replace laptops if users don't have to perform heavy processing tasks and do not require a CD or DVD player.

1.6 BASIC APPLICATIONS OF COMPUTERS

When the first computers were developed, they were used only in the fields of mathematics and science. In fact, the first effective utilization of computers was for decoding messages in military applications. Later on, computers were used in real-time control systems, such as for landing on the moon. However, with the advancement of technology, the cost of a computer and its maintenance declined. This opened the way for computers extensively being used in business and commercial sector for information processing. Today, computers are widely used in different fields as discussed below.

Communication Internet which connects computers all over the world. Internet gives you access to enormous amount of information much more than you could have in a library. Then using electronic mail you can communicate in seconds with a person who is thousands of miles away. The chat software enables you to chat with another person in real-time (irrespective of the physical location of that person). Then, video conferencing tools are becoming popular for conducting meetings with people who are unable to be present at a particular place.

Desktop Publishing Desktop publishing software enables you to create page layouts for entire books.

Government Computers are used to keep records on legislative actions, Internal Revenue Service records, etc.

Traffic Control It is used by governments for city planning and traffic control.

Legal System Computers are being used by lawyers to shorten the time required to conduct legal precedent and case research. Lawyers use computers to look through millions of individual cases and find whether similar or parallel cases were approved, denied, criticized, or overruled. This enables the lawyers to formulate strategies based on past case decisions. Moreover, computers are also used to keep track of appointments and prepare legal documents and briefs in time for filling cases.

Retail Business Computers are used in retail shops to enter the order, calculate the cost, and print a receipt. They are also used to keep an inventory of the products available and a complete description about them.

Sports In sports, computers are used to compile statistics, identify weak players and strong players by analysing statistics, sell tickets, create training programs and diets for athletes, and suggest game plan strategies based on the competitor's past performance. Computers are also used to generate most of the graphic art displays flashed on scoreboards.

Computers are used in the control room to display action replays and insert commercial breaks on schedule. Moreover, sports shoes manufacturing companies, like NIKE, use computers for designing footwears. They calculate stress points and then create the style and shape that offer maximum support for the foot.

Music Computers are used to generate a variety of sounds. Moreover, the background music in movies, television shows, and commercials are all generated electronically using computers.

Movies Computers are used to create sets, special effects, animations, cartoons, imaginary characters, videos, and commercials.

Travel and Tourism Computers are used to prepare ticket, monitor the train's or airplane's route, or guide the plane to a safe landing. They are also used to know about hotels in an area, reserve room, or rent a car.

Business and Industry In business and industry, computers are used mainly for entering and analysing data, pay roll processing, personnel record keeping, inventory management, etc.

Hospitals Hospitals use computers to record every information about a patient from the time of his admission till his exit. For example the date, time, reason of admit, the doctor being consulted, all prescribed medications, doctor visits, other hospital services, bill, etc. are all stored in computers. Moreover, computer-controlled devices are widely used to monitor pulse rate, blood pressure, and other vital signs of the patient and in an emergency situation an alarm is used to notify the nurses and other attendants.

Moreover, computers are used as an aid to physically handicapped people. For example, computers are used to develop more effective artificial limbs for amputees.

Simulation Computers enable the engineers to design aircraft models and simulate the effects that winds and other environmental forces might have on those designs. Even the astronauts at NASA are trained using computer-simulated problems that could be encountered during launch, in space, or upon return to Earth.

Geology Civil engineers use computers to evaluate the effects of an earthquake on the structure of buildings based on age, proximity to the fault, soil type, size, shape, and construction material.

Astronomy Spacecrafts are usually monitored using compu-ters which not only keep a continuous record of the voyage and the records of the speed, direction, fuel, temperature, and such performance but also suggests a corrective action if the vehicle makes any mistake. The remote stations on the earth compares all these quantities with the desired values and in case these values need to be modified to enhance the performance of the space craft, signals are immediately sent which set in motion the mechanics to rectify the situation. With the help of computers, these are done within a fraction of seconds.

Weather Forecasting When computers are fed with mathematical equations along with data about air pressure, temperature, humidity, and other values, the solution of these equations gives an accurate prediction of weather in a particular area. For example, a Crax XMP Supercomputer installed at Mausam Bhavan in New Delhi is used to predict weather and climatic changes in the Indian sub-continent.

Education A computer is a powerful teaching aid and acts as another teacher in the classroom. Teachers use computers to develop instructional material. They may use pictures, graphs, and graphical presentations to easily illustrate an otherwise difficult concept. Moreover, teachers at all levels can use computers to administer assignments and keep track of grades of the students. Besides teachers, students also prefer to learn from an E-learning software rather than learning from a book. Students can also give online exams and get instant results.

Online Banking The world today is moving towards a cashless society, where you need not have money in your pocket to purchase anything. You can just have your credit card or debit card with you. The ATM machines (Automated Teller Machine) provides a 24 × 7 service and allows you to draw cash, check the balance in your account, and order a product.

Industry and Engineering Computers are found in all kinds of industries like thermal power plant, oil refineries, chemical industries, etc. for process control, computer aided designing, and computer aided manufacturing.

Computerized process control (with or without human intervention) is used to enhance efficiency in applications such as production of various chemical products, oil refining, paper manufacture, rolling and cutting steel to customer requirements, etc.

In *Computer Aided Design (CAD)*, the computers are used for automating the design and drafting process. It helps an engineer to design a part, analyse its characteristics, and then subject it to simulated stresses. In case a part fails the stress test, its specifications can be modified on the computer and retested. The final design specifications are released for production only when the engineer is satisfied that the part meets strength and other quality considerations.

Computer-aided manufacturing (CAM) phase comes up where CAD leaves off. In this phase, the metal or other materials are manufactured while complying with their specification. For this computer-controlled manufacturing tools are used to produce high-quality products.

Robots Robots are computer-controlled machines mainly used in manufacturing process in extreme conditions where humans cannot work. For example, in high temperature, high pressure conditions or in processes that demand very high level of accuracy.

Decision Support Systems Computers help managers to analyse their organization's data to understand the present scenario of their business, view the trends in the market, and predict the future of their products. Managers also use decision support systems to analyse market research data, to size up the competition, and to plan effective strategies for penetrating their markets.

Expert System Expert systems are used to automate the decision-making process in a specific area like analysing the credit histories for loan approval and diagnosing a patient's condition for prescribing an appropriate treatment. Expert systems analyse the available data in depth to recommend a course of action. A medical expert system can provide the most likely diagnosis of a patient's condition.

Others Adding more to it, in today's scenario computers are used to find jobs on the Internet, find a suitable match for a boy or girl, read news and articles online, find one's batchmates, send and receive greetings pertaining to different occasions, etc.

1.7 STORED PROGRAM CONCEPT

All digital computers are based on the principle of stored program concept,which was introduced by Sir John von Neumann in the late 1940s. The following are the key characteristic features of this concept:

- Before any data is processed, instructions are read into memory.
- Instructions are stored in the computer's memory for execution.
- Instructions are stored in binary form (using binary numbers—only 0s and 1s).
- Processing starts with the first instruction in the program, which is copied into a control unit circuit. The control unit executes the instructions.
- Instructions written by the users are performed sequentially until there is a break in the current flow.
- Input/Output and processing operations are performed simultaneously. While data is being read/written, the central processing unit (CPU) executes another program in the memory that is ready for execution.

Note

A stored program architecture is a fundamental computer architecture wherein the computer executes the instructions that are stored in its memory.

A stored program architecture is a fundamental computer architecture wherein the computer executes the instructions that are stored in its memory.John W. Mauchly, an American physicist, and J. Presper Eckert, an American engineer, further contributed to the stored program concept to make digital computers much more flexible and powerful. As a result, engineers in England built the first stored-program computer, Manchester Mark I, in the year 1949. They were shortly followed by the Americans who designed EDVAC in the very same year.

Today, a CPU chip can handle billions of instructions per second. It executes instructions provided both the data and instructions are valid. In case either one of them or both are not valid, the computer stops the processing of instructions.

1.7.1 Types of Stored Program Computers

A computer with a Von Neumann architecture stores data and instructions in the same memory (refer Figure 1.11(a)). There is a serial machine in which data and instructions are selected one at a time. Data and instructions are transferred to and from memory through a shared data bus. Since there is a single bus to carry data and instructions, process execution becomes slower.

Later Harvard University proposed a stored program concept in which there was a separate memory to store data and instructions (refer Figure 1.11(b)). Instructions are selected serially from the instruction memory and executed in the processor. When an instruction needs data, it is selected from the data memory. Since there are separate memories, execution becomes faster.

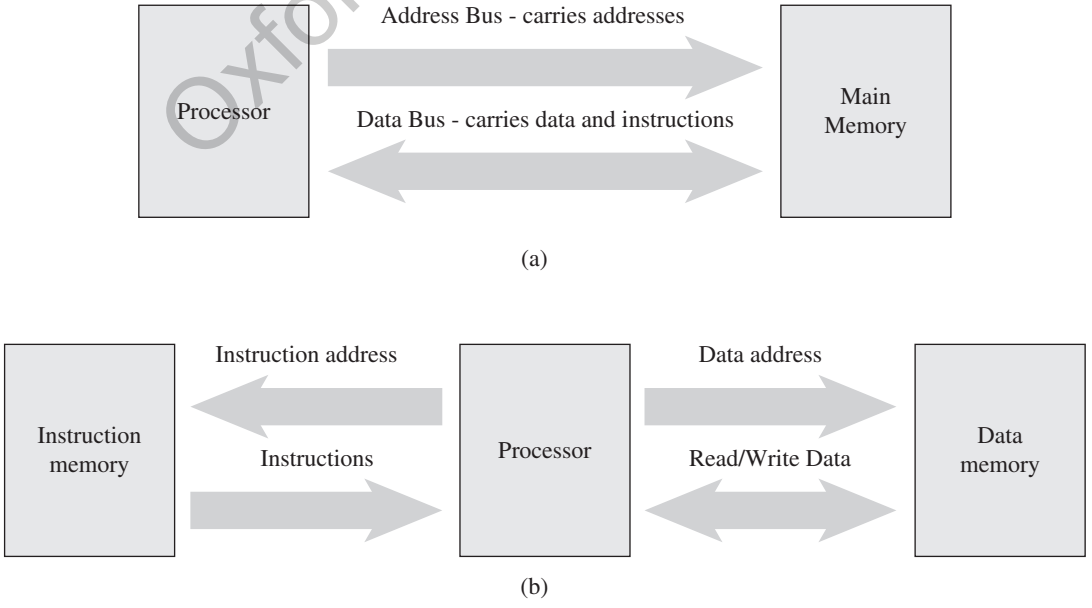


Figure 1.11 (a) Von Neumann architecture– Shared memory for instructions and data (b) Harvard architecture– Separate memories for instructions and data

1.8 COMPONENTS AND FUNCTIONS OF A COMPUTER SYSTEM

A computer is an electronic device which basically performs five major operations, which are as follows:

1. accepting data or instructions (input)
2. storing data
3. processing data
4. displaying results (output) and
5. controlling and coordinating all operations inside a computer

In this section, we will discuss all these functions and see how one component of a computer interacts with another unit to perform these operations using the block diagram of a computer as shown in Figure 1.12.

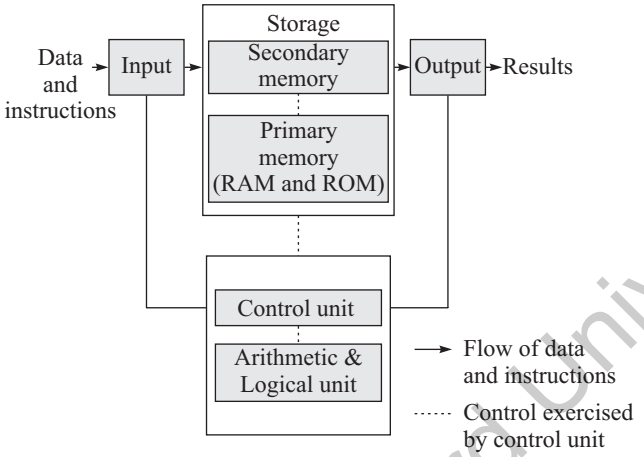


Figure 1.12 Block diagram of a computer

Input This is the process of entering data and instructions (also known as programs) into the computer system. The data and instructions can be entered into the computer system by using different input devices such as keyboard, mouse, scanner, trackball, etc.

Note

Computers understand binary language which consists of only two symbols (0s and 1s). Therefore, it is the responsibility of the input devices to convert the input data into binary codes.

Storage Storage is the process of saving data and instructions permanently in the computer so that it can be used for processing. The computer storage space stores not only the data and programs but also the intermediate results and the final results of processing. A computer has two types of storage areas:

Primary Storage Primary storage also known as the main memory is that storage area which is directly accessible by the CPU at a very fast speed. It is used to store the data and program, the intermediate results of processing and the recently generated results. The primary storage is very expensive and therefore limited in capacity. Another drawback of main memory is that it is volatile in nature, that is, as soon as the computer is switched off, the information stored in it gets erased. Hence, it cannot be used as a permanent storage of useful data and programs for future use. For example, RAM (Random Access Memory).

Secondary Storage Also known as the secondary memory or auxiliary memory is just the opposite of primary memory. It basically overcomes all the drawbacks of the primary storage. It is cheaper, non-volatile and used to permanently store data and programs of those jobs which are not being currently executed by the CPU. Secondary memory supplements the limited storage capacity of the primary memory. For example, magnetic disk you store your data in C drive, D drive, etc. for future use.

Processing The process of performing operations on the data as per the instructions specified by the user (program) is called processing. Data processing is an activity that involves handling or manipulating data in some way to assign meaning to it. The main aim of processing is to transform data into information. Data and instructions are taken from the primary memory and are transferred to the Arithmetic and Logical Unit (ALU), a part of CPU, which performs all sorts of calculations. When the processing completes, the final result is transferred to the main memory.

Output Output is the reverse of input. It is the process of giving the result of data processing to the outside world (external to the computer system). The results are given through output devices like monitor, printer, etc. Now that the computer accepts data only in binary form and the result of processing is also in the binary form, the result cannot be directly given to the user. The output devices therefore convert the results available in binary codes into a human-readable language before displaying it to the user.

Controlling The function of managing, coordinating, and controlling all the components of the computer system is handled by the control unit, a part of CPU. The control unit decides the manner in which the instructions will be executed and the operations will be performed.

1.9 CONCEPT OF HARDWARE AND SOFTWARE

You have a TV at home. When you purchase a TV, it is a box like device. A TV can be used only when it is able to display different programs. You can touch a TV but you cannot touch a program. Same is the concept in a computer. A computer system is made up of two parts—hardware and software.

1.9.1 Hardware

All the physical parts that can be touched are called hardware (refer Figure 1.13). For example, all input and output devices, memory devices form the hardware part of the computer.

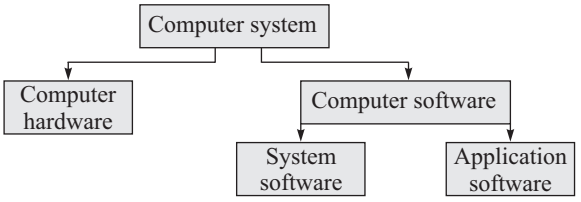


Figure 1.13 Parts of a computer system

If we think of computer as a living being, then the hardware would be the body that does things like seeing with eyes, lifting objects, and filling the lungs with air; the software would be the intelligence that helps in interpreting the images that come through the eyes, instructing the arms how to lift objects, and forcing the body to fill the lungs with air.

Since the computer hardware is a part of a machine, it can only understand two basic concepts: 'on' and 'off'. The 'on' and 'off' concept is called binary. Computer software was developed to tell the computer hardware what to do.

1.9.2 Software

The computer hardware cannot think and make decisions on its own. So, it cannot be used to analyse a given set of data and find a solution on its own. The hardware needs a software (a set of programs) to instruct what has to be done. A program is a set of instructions that is arranged in a sequence to guide a computer to find a solution for the given problem. The process of writing a program is called *programming*.

Let us now discuss the CPU and the other hardware components of a computer system in detail.

1.10 CENTRAL PROCESSING UNIT (CPU) : BASIC ARCHITECTURE

Central Processing Unit can be called the brain of the computer system because the entire processing of data and execution of instructions is done here. It is made up of one or more than one microprocessors which consist of two main parts—arithmetic and logical unit (ALU) and control unit (CU). It also contains registers and a bus interface unit (BIU) of shown in Figure 1.14.

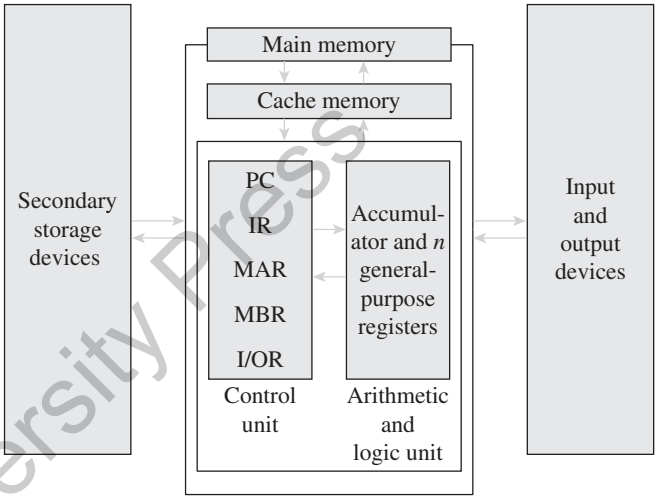


Figure 1.14 Basic computer organization

Arithmetic and Logical Unit

The ALU performs all kinds of calculations, such as arithmetic (add, subtract, multiply, divide, etc.), comparison (less than, greater than, or equal to), and other operations. The intermediate results of processing may be stored in the main memory, as they might be required again. When the processing completes, the final result is then transferred to the main memory. Hence, the data may move from main memory to the ALU multiple times before the processing is over.

Control Unit

The main function of the CU is to direct and coordinate the computer operations. It interprets the instructions (program) and initiates action to execute them. The CU controls the flow of data through the computer system and directs the ALU, input/output (I/O) devices, and other units. It is, therefore, called the central nervous system of the computer system. In addition, the CU is responsible for fetching, decoding, executing instructions, and storing results.

Registers

A processor register is a computer memory that provides quick access to the data currently being used for processing. The ALU stores all temporary results and the final result in the processor registers. As mentioned earlier, registers are at the top of memory hierarchy and are always preferred to speed up program execution.

Registers are also used to store the instructions of the program currently being executed. There are different types of registers, each with a specific storage function.

Accumulator and general-purpose registers These are frequently used to store the data brought from the main memory and the intermediate results during program execution. The number of general-purpose registers present varies from processor to processor. When program execution is complete, the result of processing is transferred from the accumulator to the memory through the memory buffer register (MBR).

Special-purpose registers These include the following:

- The memory address register (MAR) stores the address of the data or instruction to be fetched from the main memory. The value stored in the MAR is copied from the program counter.
- The MBR stores the data or instruction fetched from the main memory (Figure 1.15). If an instruction is fetched from the memory, then the contents of the MBR are copied into the instruction register (IR). If a data is fetched from the memory, the contents are either transferred to the accumulator or to the I/O register. The MBR is also used while writing contents in the main memory. In this case, the processor first transfers the contents to the MBR, which then writes them into the memory.

Figure 1.15 Data to and from memory comes from and to processor through the MBR

- The IR stores the instructions currently being executed. In general, an instruction consists of two parts—operation and address of the data on which the operation has to be performed. When the IR is loaded with an instruction, the address of the data is transferred to the MAR and the operation part is given to the CU, which interprets it and executes it.
- The I/O register is used to transfer data or instructions to or from an I/O device. An input device transfers data to the I/O register for processing. Correspondingly, any data to be sent to the output device is written in this register.
- The program counter stores the address of the next instruction to be executed.

The size of a register is usually specified by the number of bits it can store. For example, a register can be of 8 bits, 16 bits, 32 bits, or 64 bits. Higher the register size, more the data that can be stored in it.

Instruction cycle To execute an instruction, a processor normally follows a set of basic operations that are together known as an instruction cycle (Figure 1.16). The operations performed in an instruction cycle involve the following:

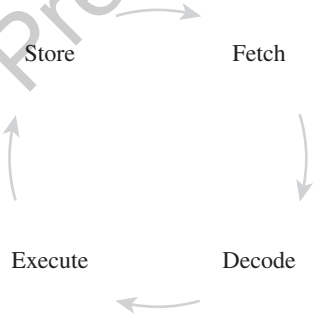


Figure 1.16 Instruction cycle

Fetch Retrieving an instruction or a data from memory.

Decode Interpreting the instruction.

Execute Running the corresponding commands to process the data.

Store Writing the results of processing into memory.

This instruction cycle is repeated continuously until the power is turned off.

Bus Interface Unit

The BIU provides functions for transferring data between the execution unit of the CPU and other components of the computer system that lie outside the CPU. Every computer system has three different types of busses to carry information from one part to the other. These are the data bus, control bus, and address bus (Figure 1.17).

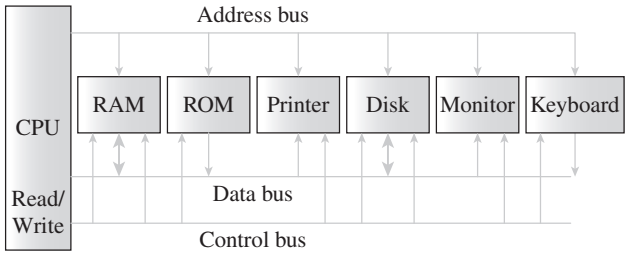


Figure 1.17 Buses with a computer system

The BIU puts the contents of the program counter on the address bus. Note that the content of the program counter is the address of the next instruction to be executed. Once the memory receives an address from the BIU, it places the contents at that address on the data bus, which is then transferred to the IR of the processor through the MBR. At this time, the contents of the program counter are modified (e.g., incremented by 1) so that it now stores the address of the next instruction.

1.11 Input and Output devices

An input device is used to feed data and instructions into the computer. In the absence of an *input device*, a computer would have only been a display device. Correspondingly, any device that outputs/gives information from a computer is called an *output device*. Refer to Figure 1.18 which shows some basic I/O devices that are generally connected with our computer system.

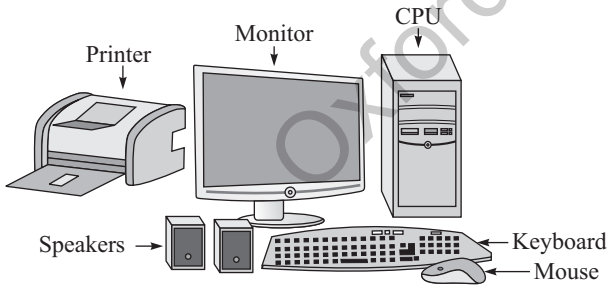


Figure 1.18 Basic I/O device computer system

Input Devices

Some of the input devices that are widely used by computer users to feed data or instruction to the computer are discussed as follows.

Keyboard Keyboard is the main input device which looks very similar to the keyboards of typewriters, with some additional keys. With a keyboard (Figure 1.19), the user can type a document, use keystroke shortcuts, access menus, play games, and perform numerous other tasks. Most keyboards have between 80 and 110 keys, which include typing keys, numeric keys, function keys, control keys, and arrow keys.

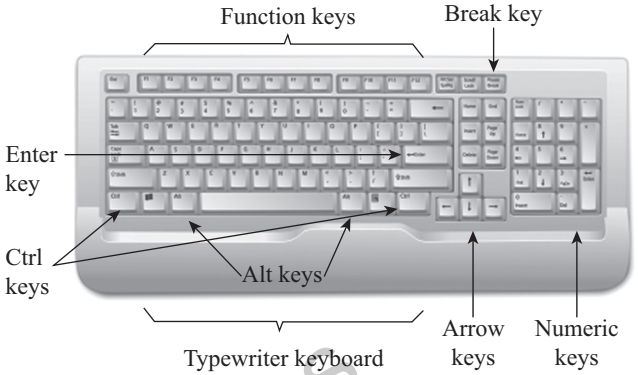


Figure 1.19 Keyboard

Source: digital art/FreeDigitalPhotos.net

The *typing keys* include the letters of the alphabet. The *numeric keys* include a set of 17 keys to speed up data entry of numbers. When the NUM LOCK is switched On, the user can type numbers, dot or use /, *, –, +. When the NUM LOCK is switched Off, the numeric keys can be used to move the cursor on the screen.

Function keys are used by applications and operating systems to perform specific commands. They are placed on the top of the keyboard in a single row. Function keys can be programmed so that their functionality can vary from one program to another.

Control keys are used to control the cursor and screen. Four *arrow* keys arranged in an inverted *T*-type fashion in between the typing keys and numeric keys are used to move the cursor on the screen in small increments. In addition to the arrow keys, there are other cursor keys (or navigational keys) such as:

- Home and End to move the cursor to the beginning and end of the current line, respectively
- Page Up and Page Down to see the previous page and the next page, respectively
- Insert to enter a character between two existing characters
- Delete to delete a character at the cursor position

Other common control keys on the keyboard include Control (Ctrl), Alternate (Alt), Escape (Esc), Print Screen key, Pause key, Windows or Start key (Microsoft Windows logo), and a shortcut key. The shortcut key has a menu with mouse pointer printed on it and is used to access the options available by pressing the right mouse button. Esc cancels the selected option and Pause key pauses a

command in progress. Finally, the Print Screen captures everything on the screen as an image. The image can be pasted into any document.

Note

Keys like Shift, Ctrl, and Alt are called *modifier keys* because they are used to modify the normal function of a key. For example, Shift + Alphabet key makes the computer to print a capital alphabet.

Mouse Mouse is an input device that is used in a graphical user interface (GUI). The users can use mouse to handle the pointer easily on the screen to perform various functions like opening a program or file. With mouse, the users no longer need to memorize commands, which was earlier a necessity when working with text-based command line environment such as MS-DOS. Mouse is specially used to create graphics such as lines, curves, and freehand shape on the screen. It is connected to a serial port or USB port on the system unit.

The mouse has two buttons and a scroller (refer Figure 1.20). Mouse can be held in hand and easily moved without lifting it along a hard flat surface to move the cursor to the desired location whether up, down, left, or right. Once the mouse is placed at the appropriate position, the user may perform the following operations:

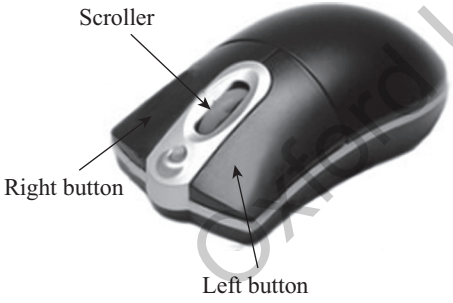


Figure 1.20 Mouse

Point Place the mouse pointer over the word or the object on the screen by moving the mouse on the desk.

Click Pressing either the left or the right button of the mouse is called clicking. When you move the mouse pointer over an icon of an application say Internet Explorer, and double click on it, it opens that application for you.

Drag Drag means moving an object to the desired position by pressing the left button.

Scroll The scroll wheel, which is placed in between the left and the right button of the mouse, is used to vertically scroll through long documents.

Trackball A trackball is a pointing device which is used to control the position of the cursor on the screen. Trackballs are usually used in notebook and laptop computers where it is placed on the keyboard as shown in Figure 1.21.

Figure 1.21 Trackball on keyboard

Source: Eugene Sergeev/Shutterstock

The trackball is nothing but an upside–down mouse that rotates in place within a socket. The user rolls the ball to position the cursor at an appropriate position on the screen and then clicks one of the buttons (identical to mouse buttons) near the trackball either to select objects or position the cursor for text entry.

Joystick Joystick (refer Figure 1.22) is a cursor control device widely used in computer games and CAD/CAM applications. A joystick has one or more push-buttons, called switches, whose position can also be read by the computer.

Figure 1.22 Joystick

Source: Viktor Kunz/Shutterstock

The lever of the joystick moves in all directions to control the movement of the pointer on the computer screen. Though a joystick is similar to a mouse, but with the mouse, the cursor stops moving as soon as you stop moving the mouse. However, in case of a joystick, the pointer continues moving in the direction the joystick is pointing. To stop the pointer, the user must return the joystick to its upright position.

Stylus (Pen) A stylus (shown in Figure 1.23) is a pen-shaped input device used to enter information or write on the touch screen of a phone. Stylus

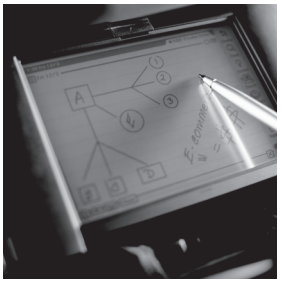


Figure 1.23 Stylus

Source: Photodisc/OUP Picture Bank

is a small stick that can also be used to draw lines on a surface as input to a computer, choose an option from a menu, move the cursor to another location on the screen, take notes, and create short messages. The stylus usually slides into a slot built into the smartphone for that purpose.

Touch Screen A touch screen shown in Figure 1.24 is a display screen which can identify the occurrence and position of a touch inside the display region. The user can touch the screen either by his finger or by using a stylus. The touch screen facilitates the users to interact with what is displayed on the screen in a direct way, rather than in an indirect way by using a mouse or touchpad. Such touch screen displays can be connected to computers, laptops, PDAs, cell phones, etc.

Figure 1.24 Touch screen

Source: Gareth Boden/OUP Picture Bank

Barcode Reader A barcode reader (or price scanner or point-of-sale scanner), shown in Figure 1.25, is a hand-held input device used to capture and read information stored in a barcode. The barcode reader merely captures and translates the barcode into numbers and/or letters. To use the captured information, it must be connected to a computer for further processing. These days, bar codes and readers are widely used in the following areas:

Figure 1.25 Barcode reader

Source: Image courtesy of Vectorlie at FreeDigitalPhotos.net

- in supermarkets and retail stores as point-of-sale devices
- to take inventory in retail stores
- to check out books from a library
- to track manufacturing and shipping movement
- to sign in on a job
- to identify hospital patients
- to tabulate the results of direct mail marketing returns
- to tag honey bees used in research.

Scanner A scanner (shown in Figure 1.26) is a device that captures images, printed text, handwriting from different sources (such as photographic prints, posters, magazines, etc.) and converts it into a digital image for computer editing and display.

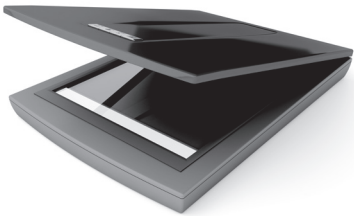


Figure 1.26 Flatbed image scanner

Source: Mile Atanasov/Shutterstock

Scanner has enabled users to store text documents as text files. Hence, the text files occupy much less storage space and can be easily edited. These days, they are used in the following areas:

- in libraries to digitize and preserve documents
- to process checks and credit card slips
- to sort letters for speeding up mail delivery.

Optical Character Recognition (OCR) Device Optical character recognition is the process of converting printed materials into text or word processing files that can be easily edited and stored. The steps involved in OCR include:

- Scanning the text character by character
- Analysing the scanned image to translate the character images into character codes (e.g., ASCII)

In OCR processing, the analysis of the scanned images is done to detect light and dark areas so as to identify each letter or numeral. When a character is recognized, it is converted into an ASCII code.

OCR has facilitated users to store text documents as text files (rather than as images, as in case of scanners). Hence, the text files occupy much less storage space and can be easily edited. These days, OCR is widely used in the following areas:

- Digitize and preserve documents in libraries
- Process checks and credit card slips
- Sort letters for speeding up mail delivery

Let us take a real-world example to understand the power of OCR. The police department usually has all the criminal records stored in large file cabinets. Scanning millions of

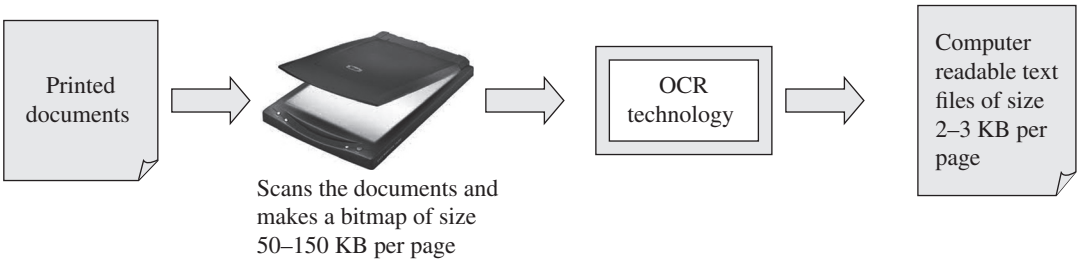


Figure 1.27 OCR technology

pages to find a particular record is not only tedious and error-prone but also an expensive process. However, if OCR is used to convert the pages into computer-readable text, the police can easily search through the entire history in a few seconds. OCR technology can be easily understood from Figure 1.27.

Advantages

- Printed documents can be converted into text files.
- Advanced OCR can recognize handwritten text and convert it into computer-readable text files.

Disadvantages

- OCR cannot recognize all types of fonts.
- Documents that are poorly typed or have strikeover cannot be recognized.
- Very old documents when passed through OCR may not convert into an exact copy of the text file. This is because some characters may not have been recognized properly. In such cases, the user has to manually edit the file.

Optical Mark Recognition (OMR) OMR is the process of electronically extracting data from marked fields, such as checkboxes and fill-infields, on printed forms. The optical mark reader is fed with an OMR sheet that has a pen or pencil mark in pre-defined positions to indicate each selected response (like answers for multiple choice questions in an entrance examination).

The OMR sheet is scanned by the optical mark reader (Figure 1.28) to detect the presence of a mark by measuring reflected light levels. The error rate for OMR technology is less than 1%. For this reason, OMR is widely used for applications in which large numbers of hand-filled forms have to be quickly processed with great accuracy, such as surveys, reply cards, questionnaires, ballots, or sheets for multiple-choice questions.

MICR MICR is used to verify the legitimacy or originality of paper documents, especially cheques. MICR consists of magnetic ink printed characters which can be recognized by high speed MICR devices (refer Figure 1.29). The printed characters provide important information (like cheque number, bank routing number, checking account number, and in some cases the amount of the cheque) for processing to the receiving party.

Figure 1.28 OMR reader

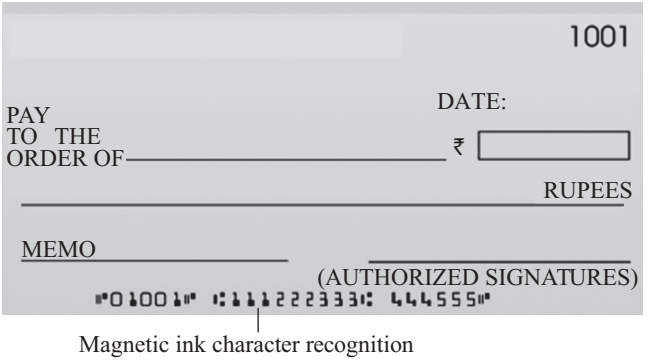


Figure 1.29 A check containing magnetic ink characters printed on it

MICR is widely used to enhance security, speed up the sorting of documents, and minimize the exposure to check fraud. Let us take a real-world problem to understand how MICR reduces the risk of fraud. If a person gives a cheque produced using a colour photocopying machine,

Figure 1.30 MICR reader

the magnetic-ink line will either not respond to magnetic fields, or will produce an incorrect code when scanned using a MICR reader (Figure 1.30). The MICR device even rejects the cheques issued by an owner of the account who has a history of writing bad cheques.

Audio Devices Audio devices are used to either capture or create sound. They enable computers to accept music, speech or sound effects for recording and/or editing. Microphones and CD players are examples of two widely used audio input devices.

A microphone feeds audio input to the computer. However, the audio input must be converted into digital data before storing it in the computer. For this, the computer must have a sound card. The sound card is a hardware unit that converts analog signals generated through microphones into digital data so that it can be stored in the computer. When the user wants to hear the pre-recorded audio input, the sound card converts the digital data into equivalent analog signals and sends it to the speakers. This process is depicted in Figure 1.31.

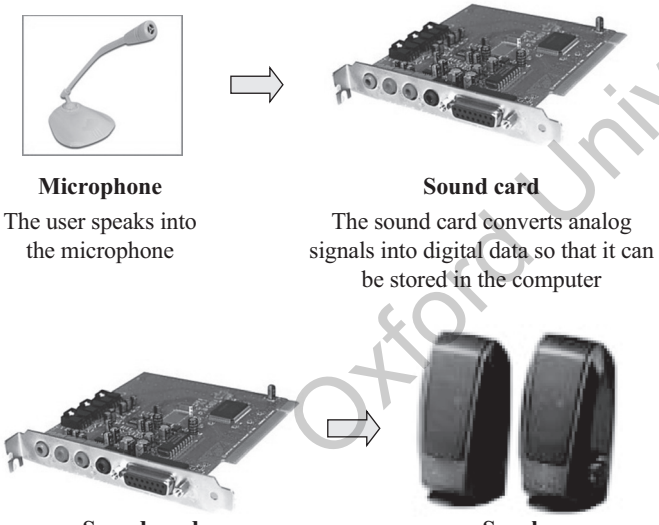


Figure 1.31 Recording and retrieving audio data

Digital Camera A computer with a microphone and speakers can be used to make telephone calls and do video-conferencing over the Internet.

Video Input Devices Video input devices are used to capture video from the outside world into the computer. Here, the term video means moving picture along with sound (as in television). Like we have sound cards to convert analog signals into digital data and digital data to analog signals, we also have video cards to convert analog signals to digital data to store it in the computer (and vice versa). Digital camera and web camera are popular examples of video input devices.

Digital camera (shown in Figure 1.32(a)) is a hand-held and easily portable device used to capture images or videos. The data can then be transferred to the computer using a cable which connects the computer to the digital camera. Once the images or videos are transferred to the computer, they can be easily edited, printed, or transmitted (through e-mails).

Figure 1.32 Video input devices (a) Digital camera (b) Web camera

Like digital camera, *web cameras* (shown in Figure 1.32(b)) also capture videos which can be transferred via Internet in real time. Web cameras are widely used for videoconferencing. Webcams are also used as security cameras as PC-connected cameras can be used to watch for movement and sound, recording both when they are detected. These recordings can then be saved in the computer and used to detect theft or any other crime.

Output Devices

We can classify the output devices in two categories as shown in Figure 1.33.

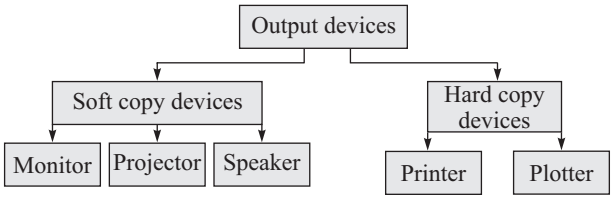


Figure 1.33 Classification of output devices

Soft copy output devices are those output devices which produce an electronic version of an output. For example, a file which is stored on hard disk, CD, pen drive, etc. and is displayed on the computer screen (monitor). Features of a soft copy output include:

- The output can be viewed only when the computer is switched On.
- The user can easily edit the soft copy output.
- Soft copy cannot be used by people who do not have a computer.
- Searching data in a soft copy is easy and fast.

- Electronic distribution of a soft copy is cheaper. It can be done easily and quickly.

Monitor The monitor, also known as visual display unit (VDU), is an output device. It looks similar to a television screen but displays information from the computer at a much higher quality. The monitor is connected to either a VGA or DVI port on a video card (in the mother board).

Monitors come in three variants—CRT, LCD, and Plasma (refer Figure 1.34). While CRT monitors look much like traditional televisions as they also have deep backs, LCD monitors on the other hand are thinner offering equivalent graphics quality. But these days LCD monitors are replacing CRT monitors as they are cheaper and occupies less space on the desk. Most monitors range in size from 15″ to 21″ or more (where size is defined as a diagonal measurement from one corner of the screen to the other).

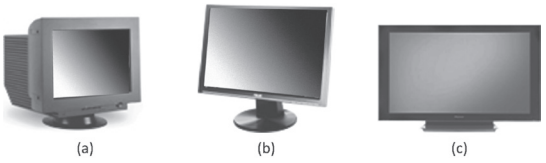


Figure 1.34 (a) CRT monitor (b) LCD monitor (c) Plasma monitor

Projector A projector (Figure 1.35) is a device that takes an image from a video source and projects it onto a screen or other surface. These days, projectors are used for a wide range of applications varying from home theater systems for projecting movies and television programs onto a screen much larger than even the biggest TV available to organizations for projecting information and presentations onto screens large enough for rooms filled with people to see. Projectors also allow users to change/adjust some features of the image like brightness, sharpness, and color settings of the image, in the same way a standard television would.

Figure 1.35 Projector
Source: olegbush/Shutterstock

Speaker With speakers, users can enjoy music, movie, or a game and the voice will be spread through the entire room. However, in case the user wants to enjoy loud music without disturbing the people around him, he can use a headphone. Headphones are small devices that fit in or on the ear, and give nearly the same quality and power of the sound only to the listener.

Users often use headphones to chat with people over the Internet. Another device called headset allows users to talk and listen at the same time while chatting over the Internet. See these devices in Figure 1.36.

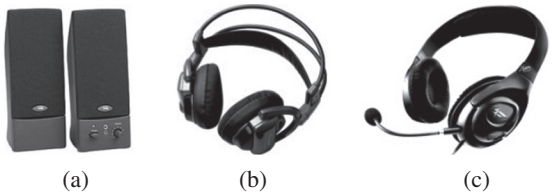


Figure 1.36 (a) Speakers (b) Headphones (c) Headset

Hard copy output devices are those output devices which produce a physical form of output. For example, the content of a file printed on a paper is a form of hard copy output. Features of a hard copy output include:

- Computer is not needed to see the output.
- Editing the hard copy is difficult.
- Hard copy output can be easily distributed to people who do not have a computer.
- Searching data in a hard copy is a tiring and difficult job.
- Distribution of a hard copy is not only costly but also slower.

Printer Printer is a device that outputs text and graphics information obtained from the computer and prints it on to a paper. Printers can be broadly classified into two groups: *impact* printers and *non-impact* printers. Refer Figure 1.37.

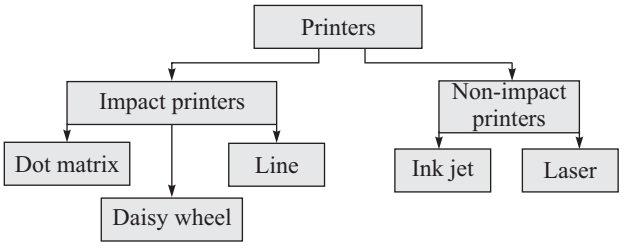


Figure 1.37 Classification of printers

Impact Printer Impact printers create characters by striking an inked ribbon against the paper. Examples of impact printers include: dot-matrix printers, daisywheel printers, and most types of line printer. Refer to Table 1.1 for its advantages and disadvantages.

Table 1.1 Advantages and disadvantages of impact printer

Advantages	Disadvantages
• Enables the user to produce carbon copies • Cheap	• Slow • Poor print quality • Noisy • Graphics formed are of very poor quality • Prints only using the standard font

Non-impact Printer Non-impact printers are much quieter than impact printers as their printing heads do not strike the paper. They offer better print quality, faster printing and the ability to create prints that contain sophisticated graphics. Non-impact printers use either solid or liquid cartridge-based ink which is either sprayed, dripped, or electrostatically drawn onto the page. The main types of non-impact printer are: inkjet printer, laser printer, and thermal printer. Refer to Table 1.2 for its advantages and disadvantages.

Table 1.2 Advantages and disadvantages of non-impact printer

Advantages	Disadvantages
• Good print quality • Noiseless • Good print quality of graphics • Fast • Can print text in different fonts	• Expensive • Ink cartridges are also costly

Dot matrix printer A dot matrix printer (shown in Figure 1.38) prints characters and images of all types as a pattern of dots (hence the name). This printer has a printhead (or hammer) that consists of pins representing the character or image. The printhead runs back and forth, or in an up-and-down motion on the page and prints by striking an ink-soaked cloth ribbon against the paper, much like the print mechanism of a typewriter.

From the 1970s to 1990s, dot matrix impact printers were the most common type of printers used with PCs.

Several dot matrix printer manufacturers implemented colour printing through a multi-colour ribbon. Colour was obtained through a multi-pass composite printing process. In each pass, the printhead struck a different section of the ribbon (one primary colour). However, because of poor colour quality and increased operating expense, colour dot matrix printers could never replace their monochrome counterparts. Moreover, the black ink section would gradually contaminate the other three colours (RGB), thereby changing the consistency of printouts over the life of the ribbon. The colour dot matrix printer was therefore suitable only for abstract illustrations and pie charts, but not for photo-realistic reproduction.

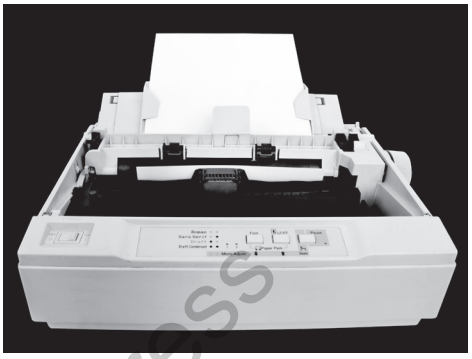


Figure 1.38 Dot matrix printer

Source: burnel1/Shutterstock

The speed of dot matrix printers varies in the range of 50–500 cps (characters per second).

Advantages

- The dot matrix printer can produce carbon copies.
- It offers the lowest printing cost per page.
- It is widely used for bulk printing where the quality of the print is not of much importance.
- It is inexpensive.
- When the ink is about to be exhausted, the printout gradually fades rather than suddenly stopping partway through a job.
- It can use continuous paper rather than individual sheets, making them useful for data logging.

Disadvantages

- This type of printer creates a lot of noise when the pins strike the ribbon against the paper.
- It can only print lower resolution graphics, with limited quality.
- It is very slow.
- It has poor print quality.

Daisy wheel printer Daisy wheel printers use an impact printing technology to generate high quality output comparable to typewriters, and are three times faster. However, today, daisy wheel technology is found only in some electronic typewriters.

The printhead of a daisy wheel printer is a circular wheel, about 3 inches in diameter with arms or spokes. The shape of the printer wheel resembles the petals of a daisy flower, and hence its name. The characters are embossed at the outer ends of the arms.

To print a character, the wheel is rotated in such a way that the character to be printed is positioned just in front of the printer ribbon. The spoke containing the required character is then hit by a hammer, thereby striking the ribbon to leave an impression on the paper placed behind the ribbon. The movement of all these parts is controlled by a microprocessor in the printer.

The key benefit of using a daisy wheel printer is that the print quality is high, as the exact shape of the character hits the ribbon to leave an impression on the paper.

Line printer A line printer is a high-speed impact printer in which one typed line is printed at a time. The speed of a line printer usually varies from 600 to 1200 lines per minute, or approximately 10–20 pages per minute. Because of their high speed, line printers are widely used in data centers and in industrial environments. Band printer is a commonly used variant of line printers.

Band printer A band printer (loop printer), is an impact printer with a printing mechanism that uses a metal loop or band to produce typed characters. The set of characters are permanently embossed on the band, and this set cannot be changed unless the band is replaced. The band itself revolves around hammers that push the paper against the ribbon, allowing the desired character to be produced on the paper.

The main advantage of using a band printer is its high speed. This type of printer can print 2000 lines per minute, and is, therefore, perfect for high volume printing in businesses, schools, and other organizations. Band printers are normally attached to mainframes and used for industrial printing.

However, band printers cannot be used for any graphics printing, as the characters are predetermined and cannot be changed unless the band is changed. Band printers were very popular in the 1970s and 1980s; however, today, laser printers have replaced band printers.

Note

Band printers are often confused with band printing. Band printing is the process of sending output to a printer and is not associated with this type of printers

Inkjet printer Inkjet printers, shown in Figure 1.39, came in the market in the 1980s, but it was only in the 1990s that their prices reduced enough to bring the technology to the high street. Inkjet printers have made rapid technological advances in recent years. The colour inkjet printers have succeeded in making colour printing an affordable option even for home users.

Figure 1.39 Inkjet printer

Source: Iakov Filimonov/Shutterstock

The printhead of inkjet printers has several tiny nozzles, also called jets. As the paper moves past the printhead, the nozzles spray ink onto it, forming characters and images. If you observe a printout that has just come out from an inkjet printer, you will see that the dots are extremely small (usually between 50 and 60 microns in diameter) and are positioned very precisely, with resolutions of up to 1440 × 720 dpi. To create a coloured image, the dots can have different colours combined together.

An inkjet printer can produce from 100 to several hundred pages (depending on the nature of the hard copy), before the ink cartridges must be replaced. There is usually one black ink cartridge and one colour cartridge containing ink in primary pigments (cyan, magenta, and yellow).

While inkjet printers are cheaper than laser printers, they are more expensive to maintain. The cartridges of inkjet printers have to be changed more frequently, and the special coated paper required to produce high quality output is very expensive. Hence, the cost per page of inkjet printers becomes ten times more than laser printers. Therefore, inkjet printers are not well suited for high volume print jobs.

Laser Printer A laser printer shown in Figure 1.40 is a non-impact printer that works at a very high speed and produces high-quality text and graphics. It uses the photocopier technology.

Figure 1.40 Laser printer

Source: restyler/Shutterstock

When a document is sent to the printer, the following steps take place:

- A laser beam 'draws' the document on a drum (which is coated with a photo-conductive material) using electrical charges.
- After the drum is charged, it is rolled in toner (a dry powder type of ink).
- The toner sticks to the charged image on the drum.
- The toner is transferred onto a piece of paper and fused to the paper with heat and pressure.
- After the document is printed, the electrical charge is removed from the drum and the excess toner is collected.

While colour laser printers are also available in the market, users prefer only monochrome printers because a color laser printer is up to 10 times more expensive than a monochrome laser printer.

Plotter A plotter is a printing device which is usually used to print vector graphics with a high print quality. It is widely used to draw maps, in scientific applications, and in applications such as computer-aided design, computer-aided-manufacturing, and computer-aided-engineering. Architects use plotters to draw blueprints of the structures they are working on.

A plotter is basically a printer that interprets commands from a computer to make line drawings on paper with one or more automated pens. Since plotters are much more expensive than printers, they are used only for specialized applications. Hewlett-Packard is the leading vendor of plotters worldwide. There are two different types of plotters—drum and flatbed (refer Figure 1.41).

Figure 1.41 Plotter

Source: Michal Vitek/Shutterstock

1.12 COMPUTER MEMORY

Computer memory is an internal storage area in the computer used to store data and programs either temporarily or permanently. No processing is done in the computer memory. A computer memory can be broadly divided into two groups: primary (main) memory and secondary memory. While the main memory holds instructions and data when a program is executing, the secondary memory holds data and programs not currently in use and provides long-term storage. Refer to Table 1.3 to understand the key differences between primary and secondary memory.

Table 1.3 Differences between primary and secondary memory

Primary memory	Secondary memory
It is more expensive.	It is cheaper.
It is faster and more efficient than secondary memory.	It is slower and less efficient than secondary memory.
Directly accessed by the CPU.	Cannot be accessed directly by the CPU.
It is volatile in nature.	It is non-volatile in nature.
Storage capacity is limited.	It has large storage capacity.
It has no moving parts.	It has moving parts.
The memory is power dependent.	The memory is power independent.
The memory is integrated circuit based.	The memory is magnetic or optical based.
It consumes less power.	It consumes more power.
It stores data temporarily.	It stores data permanently.

1.12.1 Memory Hierarchy

In contemporary usage, *memory* usually refers to random-access memory, typically DRAM (Dynamic-RAM) but *memory* can also refer to other forms of data storage. In computer terminology, the term *storage* refers to storage devices that are not directly accessible by the CPU (secondary or tertiary storage). Examples of secondary storage include hard disk drives, optical disc drives, and other devices that are slower than RAM but are used to store data permanently.

These days, computers use different types of memory which can be organized in a hierarchy around the CPU, as a trade-off between performance and cost. The memory at a higher level in the storage hierarchy has less capacity to store data, is more expensive, and is fastest to access as shown in Figure 1.42.

CPU Registers

CPU registers are located inside the processor and are therefore directly accessed by the CPU. Registers are the fastest of all forms of computer data storage.

Cache Memory

Cache memory is an intermediate form of storage between registers and the primary memory. It is used to store instructions and data that are repeatedly required to execute programs thereby improving the overall system

speed and increase the performance of the computer. Keeping frequently accessed data and instructions in the cache avoids accessing the slower primary memory.

Working of the Cache Memory When a program is being executed and the CPU wants to read data or instructions, then the following steps will be performed:

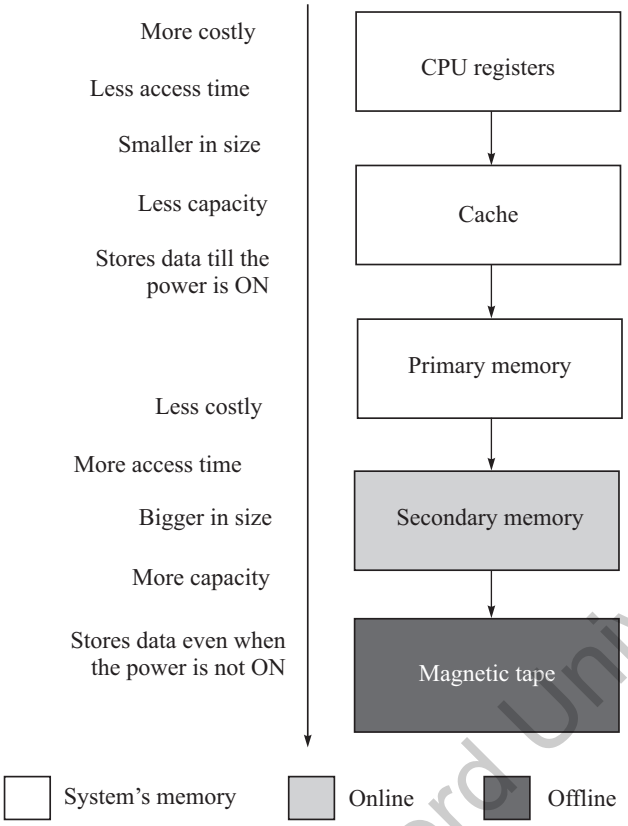


Figure 1.42 Memory hierarchy

CPU first checks whether the data or instruction is available in cache memory. If it is not present, the CPU reads the data or instructions from the main memory into the processor registers. The CPU also copies it into the cache memory. When the same piece of data/instruction is needed, the CPU reads it from the cache memory instead of the main memory.

1.12.2 Primary Memory

Primary memory (or main memory or internal memory) can be directly accessed by the CPU. The CPU continuously reads instructions stored in the primary memory and executes them. Any data that has to be operated by the CPU is also stored there. There are two types of primary memory: RAM and ROM, which are discussed as follows.

Random Access Memory (RAM)

RAM is a volatile (stores data only when the power is On) storage area within the computer typically used to store data temporarily so that it can be accessed by the CPU. The information stored in RAM is loaded from the computer's hard disk, and includes data related to the operating system and applications that are currently being executed by the processor.

RAM is considered *random access* because any memory cell can be directly accessed if its address is known. When the RAM gets full, the computer system operates at a slow speed. When multiple applications are being executed simultaneously and the RAM gets fully occupied by the application's data, it is searched to identify memory portions that have not been utilized. The contents of those locations are then copied onto the hard drive. This action frees up RAM space and enables the system to load other pieces of required data.

These days, the applications' and operating system's demand for system RAM has drastically increased. For example, in the year 2000, a personal computer (PC) had only 128 MB of RAM, but today PCs have 1–2 GB of RAM installed, and may include graphics cards with their own additional 512 MB or more of RAM. As discussed earlier, there are two types of RAM—static RAM (SRAM) and dynamic RAM (DRAM).

Static RAM This is a type of RAM that holds data without an external refresh as long as it is powered. This is in striking contrast with the DRAM which must be refreshed multiple times in a second to hold its data contents. SRAM is made of D flip-flops in which the memory cells flip-flop between 0 and 1 without the use of capacitors. Therefore, there is no need for an external refresh process to be carried out.

The limitation of SRAM is that it occupies more space and is more expensive than DRAM. While each transistor on a DRAM chip can store one bit of information, the SRAM chip, on the other hand, requires four to six transistors to store a bit. This means that a DRAM chip can hold at least four times as much data as an SRAM chip of the same size, thereby making SRAM much more expensive.

However, SRAM is faster, more reliable than DRAM, and is often used as cache memory. SRAM chips are also used in cars, household appliances, and handheld electronic devices.

Dynamic RAM This is the most common type of memory used in personal computers, workstations, and servers today. A DRAM chip contains millions of tiny memory cells. Each cell is made up of a transistor and a capacitor, and can contain 1 bit of information—0 or 1. To store a bit of information in a DRAM chip, a tiny amount of power is put into the cell to charge the capacitor. Hence, while reading a bit, the transistor checks for a charge in the capacitor. If a charge is present, then the reading is 1; if not, the reading is 0.

However, the problem with DRAM is that the capacitor leaks energy very quickly and can hold the charge for only a fraction of a second. Therefore, a refresh process is required to maintain the charge in the capacitor so that it can retain the information. This refreshing process is carried out multiple times in a second and requires that all cells be accessed, even if the information is not needed.

However, the advantage of DRAM over SRAM is that it is cheap, can hold more data per chip, and generates less heat than SRAM. DRAM is widely used to build the main memory. The following are the different types of DRAM:

Synchronous DRAM (SDRAM) SDRAM synchronizes itself with the clock speed of the microprocessor to enable faster access to memory.

Enhanced SDRAM (ESDRAM) This version of SDRAM, though not widely used, includes a small SRAM cache to reduce delays in data access and speed up operations.

Double data rate SDRAM (DDR) DDR allows data transfers on both the rising and falling edges of the clock cycle, which doubles the data throughput. DDR SDRAM chips are available in capacities of 128 MB to 1 GB. Although DDR memory is very common, the technology is becoming outdated and is being replaced by DDR2.

DDR2 These chips are the next generation of DDR SDRAM memory. It can hold 256 MB to 2 GB of memory and can operate at higher bus speeds. Although DDR2 has twice the latency (data access delays) of DDR, it delivers data at twice the speed, thereby performing at the same level.

Rambus DRAM (RDRAM) It is a proprietary, protocol-based, high-speed memory technology developed by Rambus Inc. RDRAM can operate at extremely high frequencies as compared to other types of DRAMs.

Synchronous link dynamic RAM (SLDRAM) This version of SDRAM, not used widely, was basically designed as a royalty-free, open-industry standard design alternative to RDRAM.

Read Only Memory (ROM)

ROM refers to computer memory chips containing permanent data. Unlike RAM, ROM is non-volatile, that is, the data is retained in it even when the computer is turned Off. Refer Table 1.4 to understand the key differences between RAM and ROM.

Table 1.4 Differences between RAM and ROM

RAM	ROM
Data can be read as well as written.	Data can only be read.
Data is stored temporarily.	Data is stored permanently.
Data is stored while the computer is being used by users to hold their data.	Data is stored during the time of fabrication.
It is required while computer is being used by users to run their applications.	It is required for starting the computer, and storing important programs.

Most computers contain a small amount of ROM that stores critical programs which are used to start the computer when it is turned On. Originally, ROM was actually read-only. So, in order to update the programs stored in ROM, the ROM chip had to be removed and physically replaced by the ROM chip that has a new version of the program. However, today ROM chips are not literally *read only*, as updates to the ROM chip are possible. The process of updating a ROM chip is a bit slower as memory must be erased in large portions before it can be re-written. Rewritable ROM chips include PROMs, EPROMs, and EEPROMs.

- *Programmable read-only memory (PROM)* also called one-time programmable ROM can be written to or programmed using a special device called a PROM programmer. The working of a PROM is similar to that of a CD-ROM recorder which enables the users to write programs just once but the recorded data can be read multiple times. Programming a PROM is also called *burning*.
- *Erasable programmable read-only memory (EPROM)* is a type of ROM that can be erased and re-programmed. The EPROM can be erased by exposing the chip to strong ultraviolet light typically for 10 minutes or longer and then rewritten with a process that again needs higher than usual voltage applied.
- *Electrically erasable programmable read-only memory (EEPROM)* allows its entire or selected contents to be electrically erased, then rewritten electrically. The process of writing an EEPROM is also known as flashing.

1.12.3 Secondary Storage Devices

Secondary storage (also known as external memory or auxiliary storage) differs from main memory in that it is not directly accessible by the CPU. The secondary storage devices hold data even when the computer is switched off. An example of such a device is the hard disk.

The computer usually uses its input/output channels to access data from the secondary storage devices to transfer the data to an intermediate area in the main memory. Secondary storage devices are non-volatile in nature, cheaper than the primary memory, and thus can be used to store huge amounts of data. While the CPU can read the data stored in the main memory in nanoseconds, the data from the secondary storage devices can be accessed in milliseconds.

The secondary storage devices are basically formatted according to a file system that organizes the data into files and directories. The file system also provides additional information to describe the owner of a certain file, the access time, the access permissions, and other information.

Some of the secondary storage devices such as magnetic tape, hard disks, compact disks, USB flash drive, memory card, and blue-ray disc are discussed in this section.

Magnetic Tape Magnetic tape (shown in Figure 1.43) is primarily used as an offline storage device on which the data is recorded and then physically removed or disconnected (off the computer, hence the name). In order to access data from an offline storage device, it must first be inserted in a computer.

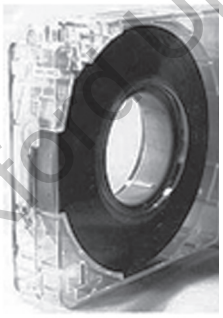


Figure 1.43 Magnetic tape

Off-line storage devices are widely used to keep a backup of important data. For example, if in case of a disaster, the original data gets destroyed, the data can be recovered from the offline devices which are usually stored in another distant place.

Hard Disk The hard drive is a part of the computer that stores all the programs and files. If the drive is damaged for some reason, all the data stored on the computer is lost. The hard disk provides relatively quick access to large amounts of data stored on an electromagnetically charged surface or a set of surfaces. The personal computers today come with a hard disk that can store gigabytes of data.

A hard disk is basically a set of disks stacked together like phonograph records, that has data recorded electromagnetically in concentric circles also known as tracks as shown in Figure 1.44.

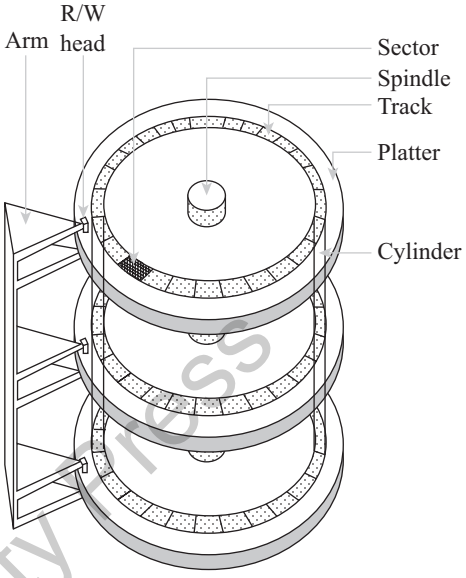


Figure 1.44 Hard disk

Compact Disk (CD) It uses laser technology to read and write data on the disc. A single CD (shown in Figure 1.45) can store a large amount of data. They are easily portable from one computer to another and are therefore used to transfer data from one computer to another. The storage capacity of CD-ROM varies from 650 MB to 1 GB. Nowadays, most of the software products (like Microsoft Office, Windows Operating System, etc.) are available on the CDs.

Figure 1.45 CD

Source: Italianphoto/Shutterstock/OUP Picture Bank

Digital Video Disk or Digital Versatile Disc (DVD) It is an extremely high capacity optical disc with storage capacity from 4.7 GB to 17 GB. DVDs are widely used to store large databases, movies, music, complex software, etc.

Most of the DVDs are double-sided discs as they can store data on both the sides of the disc. Although a DVD resembles a CD in size and shape, it stores information in different manner. When data is recorded on a DVD, the laser starts on the inside of the disk and moves outward. The laser beam has a smaller wavelength and can be focused on two different layers on the disk.

USB Flash Drive (Pen Drive) USB flash drives (shown in Figure 1.46) are removable, rewritable, and physically much smaller drives weighing even less than 30 g. The storage capacity of USB flash drives was as large as 256 GB. Such devices are a good substitute of floppy disks and CD-ROMs as they are smaller, faster, have thousands of times more capacity, and are more durable and reliable. Until approximately 2005, most desktop and laptop computers had a floppy disk drive, but nowadays floppy disk drives have been abandoned in favour of USB ports.

Figure 1.46 USB flash drive

Source: Coprid/Shutterstock/OUP Picture Bank

Memory Cards A memory card (sometimes called a *flash memory card* or a *storage card*) is a small device that can store a wide range of files as shown in Figure 1.47. They are easily portable from one place to another. A user can take a memory card, insert it into a computer, store files (such as text documents, pictures, audio files, and video files), and then remove the card and bring it to another computer where it can be again inserted to copy the files on the local disk of that computer.

Figure 1.47 Memory card

Source: IlyaAkinshin/ Shutterstock/ OUP Picture Bank

Memory cards are smaller, require less power, have higher storage capacity, are less prone to mechanical failures, allow immediate access to data and are portable among a greater number of devices. They are being widely used in the production of an increasing number of small, lightweight, and low-power devices. Although memory cards are far better than hard disks, they could not replace them because memory cards are quite expensive. For example, Compact Flash with a capacity of 192 MB typically costs more than a hard drive with a capacity of 40 GB.

Blue-ray Disc (BD) Blu-ray disk is a new optical disk developed by the Blu-ray Disc Association (BDA), which includes leading companies such as Apple, Dell, Hitachi, HP, JVC, LG, Mitsubishi, Panasonic, Pioneer, Philips, Samsung, Sharp, Sony, TDK, and Thomson. The format of a Blu-ray disk was specifi cally developed to bring forward a recordable, rewritable disk that can store large amount of data and display a high-defi nition video (HD).

Although a Blu-ray disk has the same size as that of a CD or a DVD, it can store much more data than a DVD.

A single-sided Blu-ray disk can store 25 GB of data and a dual-layer disk can store 50 GB of data. While CDs and DVDs use a red laser to read and write data, the Blu-ray disk on the other hand uses a blue-violet laser, hence the name Blu-ray. The advantage of using a blue laser with a shorter wavelength of 405 nm than the red laser (650 nm) is that it allows it to focus the laser spot with even greater precision. This means that data can be packed more tightly and therefore stored in less space. Moreover, the storage capacity of this disk is enough to store a continuous backup copy of most people's hard drives on a single disk.

A Blu-ray disk player is backwards compatible with CDs and DVDs and can therefore play a CD or a DVD despite the differences between the types of laser used. However, the Blu-ray disks will not play on CD and DVD players, because those players lack the blue-violet laser required to read the disks.

Blu-ray disks will soon replace the use of CDs and DVDs. They have already been supported by about 200 of the world's leading consumer electronics, personal computer, recording media, video game, and music companies. Besides this, they are also being supported by Hollywood studios and a number of smaller studios that have already announced that they will release new films on Blu-ray disks. Blu-ray disks are also being used in physical distribution of video games for PlayStation 3, WiiU, PlayStation 4, and Xbox One. Sony's Playstation 3 has a Blu-ray drive installed in it.

Note

Very soon a Blu-ray disc having 20 layers and storing 500 GB of data will be available in the market.

External Hard Disks As the name suggests, an external hard disk (see Figure 1.48) is a drive that is located outside the computer case in its own enclosure. It is used in addition to internal hard drives to store data. It has become quite popular because of its portability and high-storage capacity. It is connected to the computer system with a high-speed interface cable, usually with plug-and-play interfaces such as USB or FireWire, and may also contain a fan for cooling. While with USB connections, data can move at a rate of 12 to 480 Mbps (megabits per second), FireWire on the other hand can transfer data at speeds ranging from 400 to 800 Mbps. The external hard drive can also be connected to the computer wirelessly.

Figure 1.48 External hard disc

External drives allow users to save sensitive, confidential, or otherwise important data on them and kept at separate (away from the computer) secure locations. As external hard drives are lightweight portable devices, they can be easily carried anywhere and also be stored in a safe, secure location to protect the data from theft or disaster. Moreover, some external devices come with security features like fingerprint recognition to prohibit other people from gaining access to the stored data.

External hard drives have high storage capacities. External hard disks with a storage capacity of 2TB are very common these days (1 TB = 1000 GB). Therefore, they are often used to back up numerous computer files or serve as a network drive to store shared content. They are extensively used by people who do audio/video editing. These media files require high-quality settings, and therefore consume a large amount of disk space.

1.13 CLASSIFICATION OF COMPUTER SOFTWARE

Computer software is written by programmers using a programming language. The programmer writes a set of instructions (program) using a specific programming language. Such programs are known as the *source code*. Another computer program called a *compiler* is then used on the source code, to transform the instructions into a language that the computer can understand. The result is an executable computer program, which is another name for software.

Examples of computer software include the following:

- *Driver software*, which allows a computer to interact with hardware devices such as printers, scanners, and video cards.
- *Educational software*, which includes programs and games that help in teaching and providing drills to help memorize facts. Educational software can be used in diverse areas, from teaching computer-related activities like typing to subjects like chemistry.
- *Media players* and *media development software*, which are specifically designed to play and/or edit digital media files such as music and videos.
- *Productivity software*, which is an older term used to denote any program that allows the user to be more productive in a business sense. Examples of such software include word processors, database management utilities, and presentation software.
- *Operating systems software*, which helps in coordinating system resources and allows execution of other programs. Some examples of operating systems are Windows, Mac OS X, and Linux.
- *Computer games*, which are widely used as a form of entertainment software that has many genres.

Computer software can be broadly classified into two groups, namely application software and system software.

- *Application software* is designed for users to solve a particular problem. It is generally what we think of when we refer to a computer program. Examples of application software include spreadsheets, database systems, desktop publishing software, program development software, games, and web browsers. Simply put, application software represents programs that allow users to do something besides merely run the hardware.
- On the contrary, *system software*, provides a general programming environment in which programmers can create specific applications to suit their needs. This environment provides new functions that are not available at the hardware level and performs tasks related to executing the application program. System software represents programs that allow the hardware to run properly. It acts as an interface between the hardware of the computer and the application software that users need to run on the computer. Figure 1.49 illustrates the relationship between application software and system software.

Table 1.5 lists the differences between system and application softwares.

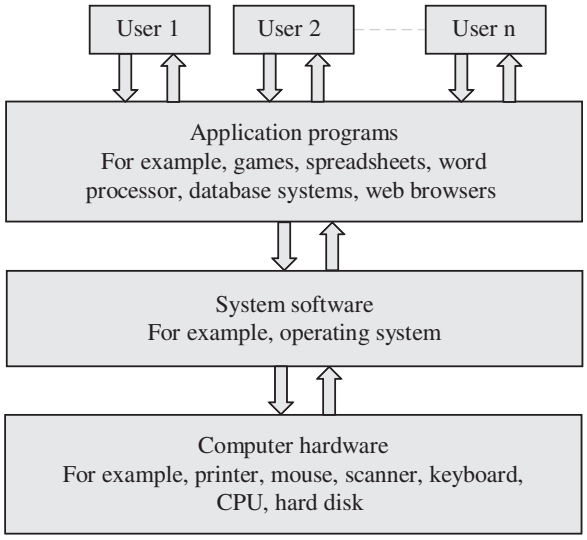


Figure 1.49 Relationship among hardware, system software, and application software

Table 1.5 Differences between system and application software

System software	Application software
It is a collection of programs that enable users to interact with hardware components efficiently.	It is a collection of programs written for a specific application, such as a library system, inventory control system, and so on.
It controls and manages the hardware.	It uses the services provided by the system software to interact with hardware components.
It is machine-dependent.	It is machine independent.
The programmer must understand the architecture of the machine and hardware details to write system software.	In most cases, the programmer ignores the architecture of the machine and hardware details to write application software.
It interacts with the hardware directly.	It interacts with the hardware indirectly through system calls provided by system software.
Writing system software is a complicated task.	Writing application programs is relatively easy.
Examples include compilers and operating systems.	Examples include Microsoft Word and Microsoft Paint.

1.13.1 System Software

System software is computer software designed to operate computer hardware and to provide and maintain a platform for running application software. Some of the most widely used system software are discussed in this section.

Computer BIOS and Device Drivers

Basic Input/Output System (BIOS) and device drivers provide basic functionality to operate and control the hardware connected to or built into the computer.

BIOS is built into the computer and is the first code run by the computer when it is switched on. The key role of BIOS is to load and start the operating system (OS).

When the computer starts, the first function that BIOS performs is to initialize and identify system devices such as the video display card, keyboard, mouse, hard disk, CD/DVD drive, and other hardware. In other words, the code in the BIOS chip runs a series of tests called POST, which stands for power on self test, to ensure that the system devices are working correctly.

The BIOS chip then locates the software held on a peripheral device such as a hard disk or a CD, and loads and executes that software, giving it control of the computer. This process is known as *booting*.

BIOS is stored on a ROM chip built into the system. It also has a user interface similar to a menu, which can be accessed by pressing a certain key on the keyboard when the PC starts. A BIOS screen is shown in Figure 1.50.

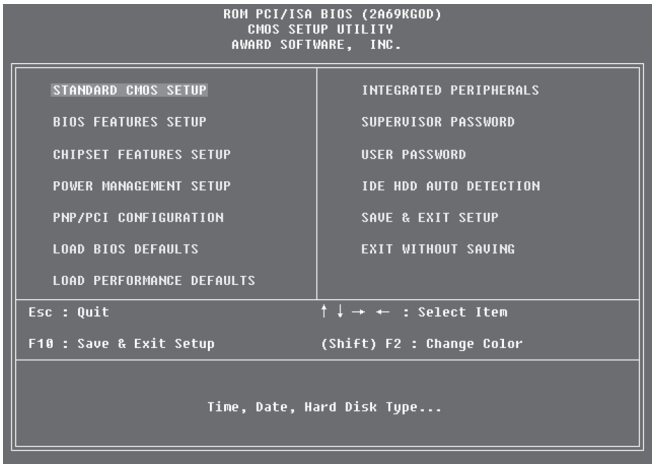


Figure 1.50 The BIOS menu

The BIOS menu enables the user to configure hardware, set the system clock, enable or disable system components, and, most importantly, select the devices which are eligible

to be a potential boot device and set various password prompts.

In summary, BIOS performs the following functions:

- Initializes system hardware
- Initializes system registers
- Initializes power management system
- Tests RAM
- Tests all the serial and parallel ports
- Initializes CD/DVD disk drive and hard disk controllers
- Displays system summary information

Operating System

The primary goal of an operating system is to make the computer system (or any other device in which it is installed, such as a cell phone) convenient and efficient to use. The operating system offers generic services to support user applications.

From the point of view of users, the primary consideration is always convenience. Users should find it easy to launch an application and work on it. For example, we use icons, which give us an idea about which application they launch. We have different icons for launching a web browser, an e-mail application, or even a document preparation application. In other words, it is the human–computer interface that helps to identify and launch an application. The interface hides a lot of details of the instructions that performs all these tasks.

Similarly, if we examine the programs that help us in using input devices such as the keyboard/mouse, all the complex details of the character-reading program are hidden from the user. We, as users, simply press buttons to perform the input operation regardless of the complexity of the details involved. The details are handled by the operating system.

An operating system ensures that system resources (such as CPU, memory, I/O devices, and so on) are utilized efficiently. For example, there may be many service requests on a web server, and each user request needs to be serviced. Similarly, there may be many programs residing in the main memory. The system needs to determine which programs are active and which need to wait for some I/O operation, since the programs that need to wait can be suspended temporarily from engaging the processor. Hence, it is important for an operating system to have a control policy and algorithm to allocate system resources.

Utility Software

Utility software is used to analyse, configure, optimize, and maintain the computer system. Utility programs may be requested by application programs during their execution for multiple purposes. Some examples of utility programs include the following:

- *Disk defragmenters* can be used to detect computer files whose contents are broken across several locations on the hard disk, and the fragments can be moved to one location in order to increase efficiency.
- *Disk checkers* can be used to scan the contents of a hard disk to find files or areas that are either corrupt in some way, or were not correctly saved, and eliminate/repair them in order to make the hard drive operate more efficiently.
- *Disk cleaners* can be used to locate files that are either not required for computer operation, or take up considerable amounts of space. Disk cleaners help the user to decide what to delete when their hard disk is full.
- *Disk space analysers* are used for visualizing disk space usage by obtaining the size of all folders (including subfolders) and files in a folder or drive.
- *Disk partitions* are used to divide an individual drive into multiple logical drives, each with its own file system. Each partition is then treated as an individual drive.
- *Backup* utilities can be used to make a copy of all information stored on a disk. In case a disk failure occurs, backup utilities can be used to restore the entire disk. Even if a file gets deleted accidentally, the backup utility can be used to restore the deleted file.
- *Disk compression* can be used to enhance the capacity of the disk by compressing/uncompressing the contents of a disk.
- *File managers* can be used to provide a convenient method of performing routine data management tasks, such as deleting, renaming, cataloguing, moving, copying, merging, generating, and modifying data sets.
- *System profilers* can be used to provide detailed information about the software installed and hardware attached to the computer.
- *Anti-virus* utilities are used to scan the computer for viruses.
- *Data compression* utilities are used to compress files to a smaller size.
- *Cryptographic* utilities are used to encrypt and decrypt files.

- *Launcher applications* are used as a convenient access point for application software.
- *Registry cleaners* are used to clean and optimize the Windows registry by deleting old registry keys that are no longer in use.
- *Network* utilities are used to analyse the computer's network connectivity, configure network settings, and check data transfer or log events.
- *Command line interface (CLI)* and *graphical user interface (GUI)* are used to interface the operating system with other software.

Translators

In this section we shall discuss the functions of translators which are computer programs used to translate a code written in one programming language to a code in another language that the computer understands.

Compiler

A compiler is a special type of program that transforms the source code written in a programming language (the *source language*) into machine language, which uses only two digits—0 and 1 (the *target language*). The resultant code in 0s and 1s is known as the object *code*. The object code is used to create an executable program.

Therefore, a compiler (Figure 1.51) is used to translate the source code from a high-level programming language to a lower-level language (e.g., assembly language or machine code). There is a one-to-one correspondence between the high-level language code and machine language code generated by the compiler.

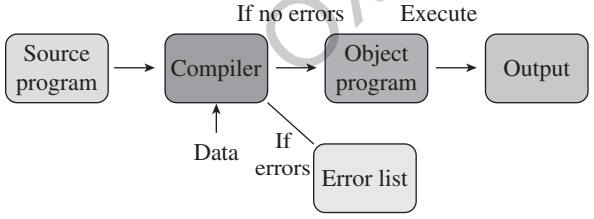


Figure 1.51 Compiler

If the source code contains errors, then the compiler will not be able to do its intended task. Errors that limit the compiler in understanding a program are called *syntax errors*. Examples of syntax errors are spelling mistakes, typing mistakes, illegal characters, and use of undefined variables. The other type of error is the logical error, which occurs when the program does not function accurately. Logical errors are much harder to locate and correct than syntax errors. Whenever errors are detected in the source code, the compiler generates a list of error messages indicating the type of error and the line in which the error has occurred. The programmer makes use of this error list to correct the source code.

The work of a compiler is only to translate the human-readable source code into a computer-executable machine code. It can locate syntax errors in the program (if any) but cannot fix it. Unless the syntactical error is rectified, the source code cannot be converted into the object code.

Each high-level language has a separate compiler. A compiler can translate a program in one particular high-level language into machine language. For a program written in some other programming language, a compiler for that specific language is needed.

Interpreter

Similar to the compiler, the *interpreter* also executes instructions written in a high-level language. Basically, a program written in a high-level language can be executed in any of the two ways— by compiling the program or by passing the program through an interpreter.

The compiler translates instructions written in a high-level programming language directly into machine language; the interpreter, on the other hand, translates the instructions into an intermediate form, which it then executes. The interpreter takes one statement of high-level code, translates it into the machine level code, executes it, and then takes the next statement and repeats the process until the entire program is translated.

HOW COMPILERS WORK

Compilers, like other programs, reside on the secondary storage. To translate a source code into its equivalent machine language code, the computer first loads the compiler and the source program from the secondary memory into the main memory. It then executes the compiler along with the source program as its input. The output of this execution is the object file, which is also stored in the secondary storage. Whenever the program is to be executed, the computer loads the object file into the memory and executes it. Thus, it is not necessary to compile the program every time it needs to be executed. Compilation will be needed again only if the source code is modified.

Note

An interpreter not only translates the code into machine language but also executes it.

Figure 1.52 shows an interpreter that takes a source program as its input and gives the output. This is in contrast with the compiler, which produces an object file as the output of the compilation process. Usually, a compiled program executes faster than an interpreted program. Moreover, since there is no object file saved for future use, users will have to reinterpret the entire program each time they want to execute the code.

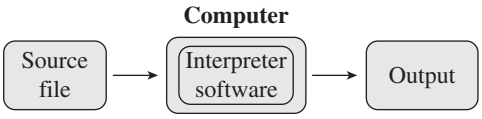


Figure 1.52 Interpreter

Overall, compilers and interpreters both achieve similar purposes, but they are inherently different as to how they achieve that purpose. The differences between compilers and interpreters are given in Table 1.6.

Table 1.6 Differences between compilers and interpreters

Compiler	Interpreter
• It translates the entire program in one go. • It generates error(s) after translating the entire program. • Execution of code is faster. • An object file is generated. • Code need not be recompiled every time it is executed. • It merely translates the code. • It requires more memory space (to save the object file).	• It interprets and executes one statement at a time. • It stops translation after getting the first error. • Execution of code is slower as every time reinterpretation of statements has to be done. • No object file is generated. • Code has to be reinterpreted every time it is executed. • It translates as well as executes the code. • It requires less memory space (no object file).

Assembler Since computers can execute only codes written in machine language, a special program, called the assembler, is required to convert the code written in assembly language into an equivalent code in machine language, which contains only 0s and 1s. The working of an assembler is shown in Figure 1.53; it can be seen that the assembler takes an assembly language program as input and gives a code in machine language (also called object program) as output. There is a one-to-one correspondence between the assembly language code and the machine language code. However, if there is an error, the assembler gives a list of errors. The object file is created only when the assembly language code is free from errors. The object file can be executed as and when required.

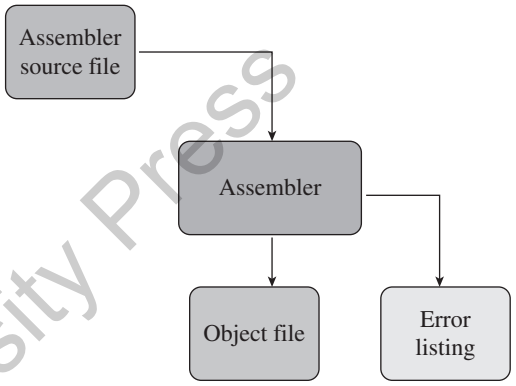


Figure 1.53 Assembler

Note

An assembler only translates an assembly program into machine language, the result of which is an object file that can be executed. However, the assembler itself does not execute the object file.

Linker

Software development in the real world usually follows a modular approach. In this approach, a program is divided into various (smaller) modules as it is easy to code, edit, debug, test, document, and maintain them. Moreover, a module written for one program can also be used for another program. When a module is compiled, an object file of that module is generated.

Once the modules are coded and tested, the object files of all the modules are combined together to form the final executable file. Therefore, a linker, also called a *link editor* or *binder*, is a program that combines the object modules to form an executable program (see Figure 1.54). Usually, the compiler automatically invokes the linker as the last step in compiling a program.

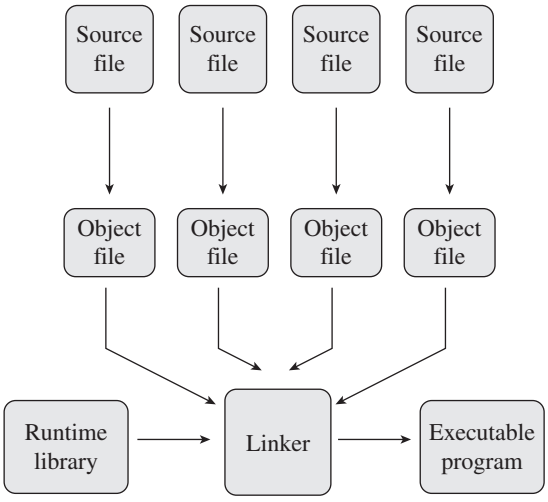


Figure 1.54 Linker

Loader

A *loader* is a special type of program that is part of an operating system and which copies programs from a storage device to the main memory, where they can be executed. Most loaders are transparent to the users.

Debugger

Debugging is a necessary step in software development process. Since it is very common for real world applications to have thousands of lines of code, the possibility of having errors in them cannot be ruled out. Therefore, identifying bugs (errors) and removing them as early as possible is very important.

Debugging tools, commonly known as *debugger*s, are used to identify coding errors at different stages of software (or program) development. These days, many programming language packages have a facility for checking the code for errors while it is being written.

A debugger is a program that runs other programs allowing users to exercise some degree of control over their programs so that they can examine them when things go wrong. A debugger helps the programmer to discover the following things:

- Which statement or expression was being executed when the error occurred?
- If an error occurred during the execution of a function, what parameters were passed to it while it was called?
- What is the value of variables at different lines in the program?
- What is the result of evaluating an expression?
- What is the sequence of statements actually executed in a program?

When a program crashes, debuggers show the position of the error in the program. Many debuggers allow programmers to run programs in a step-by-step mode. They also allow them to stop on specific points at which they can examine the value of certain variables.

1.13.2 Application Software

Application software is a type of computer software that employs the capabilities of a computer directly to perform a user-defined task. This is in contrast with system software, which is involved in integrating a computer's capabilities, but does not directly apply them in the performance of tasks that benefit the user.

To understand application software better, consider an analogy where hardware would depict the relationship of an electric light bulb (an application) to an electric power generation plant (a system).

The power plant merely generates electricity, which is not by itself of any real use until harnessed through an application such as the electric light, which performs a service that actually benefits the user.

Typical examples of software applications are word processors, spreadsheets, media players, education software, CAD, CAM, data communication software, statistical and operational research software, etc. Multiple applications bundled together as a package are sometimes referred to as an *application suite*.

Examples of Application Software These days, we have a number of application software packages available in the market for a wide range of applications. The range of these applications vary from simple applications such as word processing, inventory management to complex and scientific applications such as weather forecasting, oil and natural gas exploration. In this section we will discuss some popular application software.

Word Processing Software (MS Word) A word processor is a software package that enables its users to create, edit, print, and save documents for future retrieval and reference as shown in Figure 1.55. The key advantage of using a word processor is that it allows the users to make changes to a document without retyping the entire document. Microsoft Word is the world's leading word processing application. Users can create a variety of documents such as letters, memos, résumés, forms, or any other document that can be typed and printed.

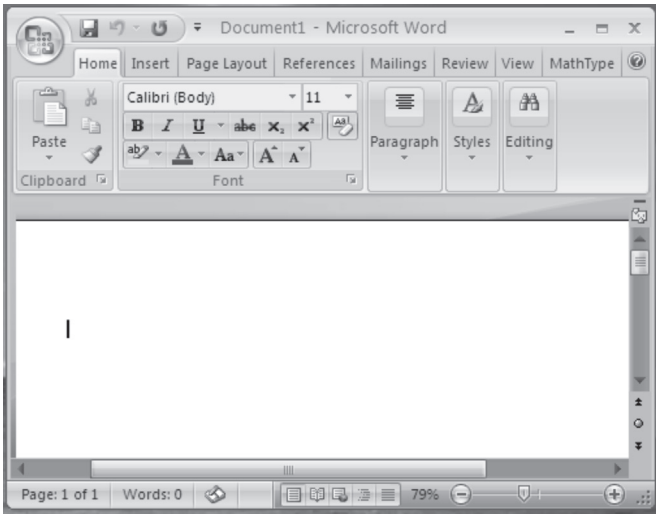


Figure 1.55 MS Word

Spreadsheet Program (Microsoft Excel) A spreadsheet software is the one in which data is stored into spreadsheet rows and columns, or 'cells' which can be formatted in various fonts or colours. Microsoft Excel is an example of a spreadsheet software (as shown in Figure 1.56) that is basically used to store, organize, and manipulate data. The stored data can also be converted into graphs for analysis.

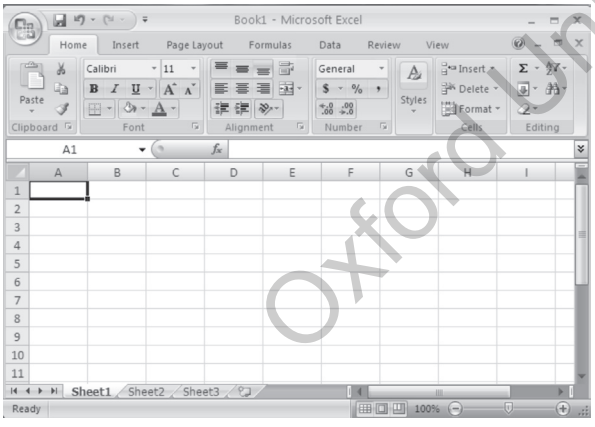


Figure 1.56 MS Excel

Microsoft Excel includes a number of simple as well as complex formulas and functions to calculate variables in the data. Excel is therefore widely used in finance to automatically calculate variables such as profit, loss, or expenditure.

Presentation Software (Microsoft PowerPoint) Microsoft PowerPoint (as shown in Figure 1.57) is used to create multimedia presentations and slide shows. When designing presentations on Microsoft PowerPoint, users can add effects on slide transitions, add sound clips, images, animations, or video clips to make the presentation even more interesting for the target audience.

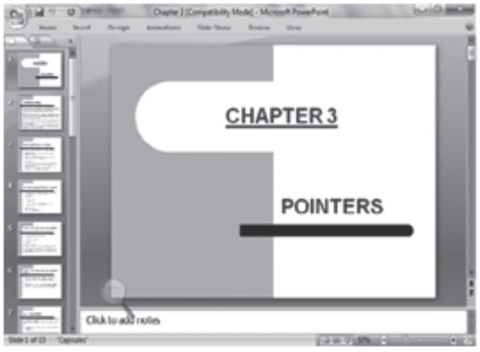


Figure 1.57 MS PowerPoint

In addition to slide shows, PowerPoint also offers printing options to facilitate the users to provide handouts and outlines for the audience as well as note pages for the speaker to refer to during the presentation.

All in all PowerPoint is a one-stop-shop for creating beautiful presentations for business and classrooms. It is also an effective tool for training purposes.

Database Software (Microsoft Access) Microsoft Access (as shown in Figure 1.58) is a database application which is used to store data for reporting, and analysis.

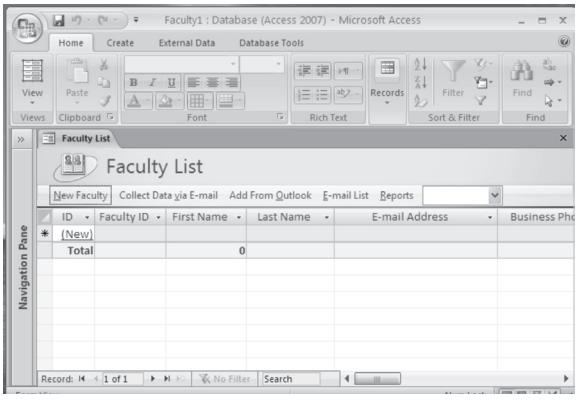


Figure 1.58 MS Access

Microsoft Access is equipped with query interface, forms to input and display data, and reports for printing. In addition to this, Access has features to automate repetitive tasks.

Microsoft Access is particularly appropriate for meeting end-user database needs and for rapid application development.

Graphics Software Graphics software or image editing software is a program that allows users to create and edit

digital images and illustrations. Examples of such software include Adobe Photoshop Illustrator, Paint Shop Pro, MS Paint, etc.

Most graphics programs have the ability to import and export one or more graphics file formats. Some of the graphics applications are given below:

Animation Software It simulates a movement by displaying a sequence of images in a fraction of a second.

CAD Software It is used by architects and engineers to create architectural drawings, product designs, landscaping plans, and engineering drawings. CAD software enables the designers to work much faster. The drawings that were created in several days can now be drawn in a few hours.

Desktop Publishing Software It facilitates users with a full set of word-processing features along with a fine control over placement of text and graphics. Using such an application, the users can easily create newsletters, advertisements, books, and other types of documents.

Multimedia Software *Multimedia* is a comprehensive term which means different types of media. It includes a combination of text, audio, still images, animation, video, and interactivity content forms.

Multimedia is used for creating exciting advertisements to grab and keep attention of the target audience. It is also used in business to design training programs. In the entertainment industry, multimedia is used to create special effects in movies and animations. It is also used in computer games and some video games that are a popular pastime.

Edutainment which combines education with multimedia entertainment is now emerging as a trend in school as well as higher education. This has made learning theories much simpler than ever before. Moreover, visually impaired or people with other kinds of disabilities can pursue their careers by using training programs specially designed for them.

Multimedia is used by engineers and researchers for modeling and simulation. For example, a scientist can look at a molecular model of a particular substance and manipulate it to arrive at a new substance. Even in medicines, doctors are now trained by looking at a virtual surgery.

Ability Media allows those with disabilities to gain qualifications in the multimedia field so they can pursue careers that give them access to a wide array of powerful communication forms.

1.14 REPRESENTATION OF DATA: BITS AND BYTES

We have seen that computers store and process data to retrieve information. Here,

- *Data* refers to anything that has some interest to the user, and
- *Information* is the result of data processing

The term *data representation* refers to the technique used to represent data internally stored in the computer.

These days, computers store massive amounts of a variety of data such as numbers, text, images, audio and video (as shown in Figure 1.59). Though all these types of data belong to a different class but internally they all are stored in the same simple format of 1s and 0s.

Computers are electronic machines which operate using binary logic. These devices use two different values to represent the two voltage levels (0 V for logic 0 and +5 V for logic 1). The two values 0 and 1, therefore, correspond to the two digits used by the binary number system.

The binary number system works like the decimal number system with the following exceptions:

- While the decimal number system uses a base 10, the binary number system on the other hand uses base 2.
- The decimal number system uses digits from 0 to 9 but the binary number system uses only two digits 0 and 1. Any other digit is considered to be invalid in this number system.

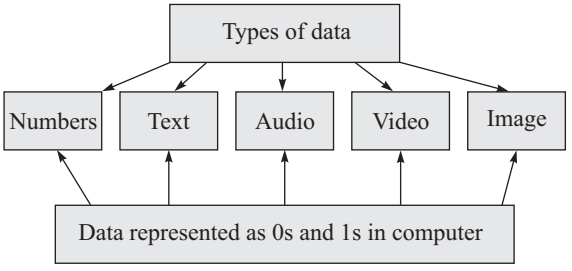


Figure 1.59 Different types of data

Some important terms in binary number system include (as shown in Table 1.7):

Table 1.7 Important terms in binary number system

Term	Size (bits)	Example
Bit	1	0
Nibble	4	1010
Byte	8	0101 1100
Word	16	0101 1100 0101 1100

Bit Bit is a short form of binary digit. It is the smallest possible unit of data. In computerized data a bit can either be 0 or 1.

Nibble Nibble is a group of four binary digits.

Byte Byte is a group of eight bits. A nibble is a half byte. Bits 0 through 3 are called the low order nibble, and bits 4 through 7 form the high order nibble as shown in Figure 1.60.

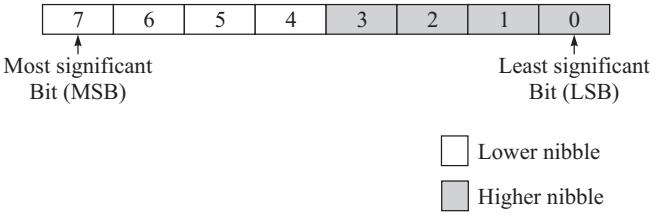


Figure 1.60 Lower and upper nibble

While a single bit can store two different values 2^0 (0 or 1), a byte comprised of 8 bits can store 2^8 or 256 different values.

Note

If a code has 128 different values, then it needs at least 7 bits to represent its values because $2^7 = 128$.

Besides bytes, data is also specified using the units shown in Table 1.8.

Table 1.8 Data units

Unit	Abbreviation	Equal to	Bytes	Power of 2
Byte	Byte	8 Bits	1	20 bytes
Kilobyte	KB	1024 Bytes	1024	210 bytes
Megabyte	MB	1024 KB	1048576	220 bytes
Gigabyte	GB	1024 MB	1073741824	230 bytes
Terabyte	TB	1024 GB	1099511627776	240 bytes

Word A group of two bytes is called a word. Bits 0 through 7 form the low order byte and bits 8 through 15 form the high order byte (refer Figure 1.61). However, computers today have redefined word as a group of 4 bytes (32 bits). With 16 bits, the computer can represent 216 (65536) different values.

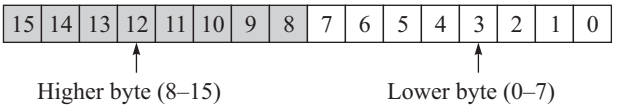


Figure 1.61 Lower and upper byte

Binary Representation with 2 Digits

We have seen that the number of values that can be encoded in binary depends on the number of binary digits. For example, if we have a single digit, we can represent only $2^1 = 2$ values 0 or 1. If we have two digits, we can represent $2^2 = 4$ values—00, 01, 10 and 11. Look at Tables 1.9 and 1.10, which summarize this concept.

Table 1.9 Data values using 2 bits

Number of digits—2 Data values that can be represented = $2^2 = 4$	
0	00
1	01
2	10
3	11

Table 1.10 Data values using 3 bits

Number of digits—3 Data values that can be represented = $2^3 = 8$	
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

1.15 PROGRAMMING LANGUAGES

A programming language is a language specifically designed to express computations that can be performed by a computer. Programming languages are used to create programs that control the behaviour of a system, to express algorithms, or as a mode of human communication.

Usually, programming languages have a vocabulary of syntax and semantics for instructing a computer to perform specific tasks. The term *programming language* refers to high-level languages such as BASIC (Beginners' All-purpose Symbolic Instruction Code), C, C++, COBOL

(COmmon Business Oriented Language), FORTRAN (FORmula TRANslator), Ada, and Pascal, to name a few. Each of these languages has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

Though high-level programming languages are easy for humans to read and understand, the computer can understand only machine language, which consists of only numbers. Each type of central processing unit (CPU) has its own unique machine language.

In between machine languages and high-level languages, there is another type of language known as assembly language. Assembly languages are similar to machine languages, but they are much easier to program because they allow a programmer to substitute names for numbers.

However, irrespective of the language that a programmer uses, a program written using any programming language has to be converted into machine language so that the computer can understand it. There are two ways to do this: *compile* the program or *interpret* the program.

The language chosen to write a program depends on the following factors:

- The type of computer on which the program is to be executed
- The type of program
- The expertise of the programmer

For example, FORTRAN is a particularly good language for processing numerical data, but it does not lend itself very well to organizing large programs. Pascal can be used for writing well-structured and readable programs, but it is not as flexible as the C programming language. C++ goes one step ahead of C by incorporating powerful object-oriented features, but it is complex and difficult to learn.

1.16 GENERATIONS OF PROGRAMMING LANGUAGES

We now know that programming languages are the primary tools for creating software. As of now, hundreds of programming languages exist in the market, some more used than others and each claiming to be the best. However, in the 1940s when computers were being developed, there was just one language—machine language.

The concept of generations of programming languages (also known as levels) is closely connected to the advances in technology. The five generations of programming languages include machine language, assembly language, high-level language (also known as the third generation language or 3GL), very high-level language (also known as the fourth generation language or 4GL), and fifth generation language that includes artificial intelligence.

1.16.1 First Generation: Machine Language

Machine language was used to program the first stored-program computer systems. This is the lowest level of programming language and is the only language that a computer understands. All the commands and data values are expressed using 0s and 1s, corresponding to the *off* and *on* electrical states in a computer.

In the 1950s, each computer had its own native language, and programmers had primitive systems for combining numbers to represent instructions such as *add* and *subtract*. Although there were similarities between each of the machine languages, a computer could not understand programs written in another machine language.

MACHINE LANGUAGE

This is an example of a machine language program that will add two numbers and find their average. It is in hexadecimal notation instead of binary notation because that is how the computer presented the code to the programmer. The program was run on a VAX/VMS computer, a product of the Digital Equipment Corporation.

```
                                    000 0000A   0000
                                    000 0000F   0008
                                    000 0000B   0008
                                                0008
                                                0058
                                                00Θ0
FF55   CF   FF54   CF   FF53   CF   C1          00A9
       FF24   CF   FF27   CF   D2   C7          00CC
                                                00E4
                                                010D
                                                013D
```

In machine language, all instructions, memory locations, numbers, and characters are represented in strings of 0s and 1s. Although machine language programs are typically displayed with the *binary* numbers represented in *octal* (base 8) or *hexadecimal* (base 16) number systems, these programs are not easy for humans to read, write, or debug.

The main advantage of machine language is that the execution of the code is very fast and efficient since it is directly executed by the CPU. However, on the downside, machine language is difficult to learn and is far more difficult to edit if errors occur. Moreover, if we want to store some instructions in the memory at some location, then all the instructions after the insertion point would have to be moved down to make room in the memory to accommodate the new instructions. In addition, the code written in machine language is not portable, and to transfer the code to a different computer, it needs to be completely rewritten since the machine language for one computer could be significantly different from that for another computer. Architectural considerations make portability a tough issue to resolve. Table 1.11 lists the advantages and disadvantages of machine language.

Table 1.11 Advantages and disadvantages of machine language

Advantages	Disadvantages
• Code can be directly executed by the computer. • Execution is fast and efficient. • Programs can be written to efficiently utilize memory.	• Code is difficult to write. • Code is difficult to understand by other people. • Code is difficult to maintain. • There is more possibility for errors to creep in. • It is difficult to detect and correct errors. • Code is machine dependent and thus non-portable.

1.16.2 Second Generation: Assembly Language

Second-generation programming languages (2GLs) comprise the assembly languages. Assembly languages are symbolic programming languages that use symbolic notations to represent machine language instructions. These languages are closely connected to machine language and the internal architecture of the computer system on which they are used. Since it is close to machine language, assembly language is also a low-level language. Nearly all computer systems have an assembly language available for use.

Assembly language developed in the mid-1950s was a great leap forward. It used symbolic codes, also known as *mnemonic* codes, which are easy-to-remember abbreviations, rather than numbers. Examples of these codes include ADD for add, CMP for compare, and MUL for multiply.

Assembly language programs consist of a series of individual statements or instructions to instruct the computer what to do. Basically, an assembly language statement consists of a label, an operation code, and one or more *operands*.

Labels are used to identify and refer instructions in the program. The operation code (opcode) is a mnemonic that specifies the operation to be performed, such as *move*, *add*, *subtract*, or *compare*. The operand specifies the register or the location in the main memory where the data to be processed is located.

However, like machine language, the statement or instruction in assembly language will vary from machine to machine, because the language is directly related to the internal architecture of the computer and is not designed to be machine independent. This makes the code written in assembly language less portable, as the code written to be executed on one machine will not run on machines from a different, or sometimes even the same manufacturer.

Nevertheless, the code written in assembly language will be very efficient in terms of execution time and main memory usage, as the language is similar to computer language.

Programs written in assembly language need a translator, often known as the assembler, to convert them into machine language. This is because the computer will understand only the language of 0s and 1s. It will not understand mnemonics such as ADD and SUB.

The following instructions are part of an assembly language code to illustrate addition of two numbers:

MOV AX,4	Stores the value 4 in the AX register of the CPU
MOV BX,6	Stores the value 6 in the BX register of the CPU
ADD AX,BX	Adds the contents of the AX and BX registers and stores the result in the AX register

Although it is much easier to work with assembly language than with machine language, it still requires the programmer to think on the machine's level. Even today, some programmers use assembly language to write those parts of applications where speed of execution is critical;

for example, video games, but most programmers have switched to 3GL or 4GL even to write such codes.

Table 1.12 lists the advantages and disadvantages of using assembly language.

Table 1.12 Advantages and disadvantages of assembly language

Advantages	Disadvantages
• It is easy to understand. • It is easier to write programs in assembly language than in machine language. • It is easy to detect and correct errors. • It is easy to modify. • It is less prone to errors.	• Code is machine dependent and thus non-portable. • Programmers must have a good knowledge of the hardware and internal architecture of the CPU. • The code cannot be directly executed by the computer.

1.16.3 Third Generation: High-level Language

Third-generation programming languages are a refinement of 2GLs. The second generation brought logical structure to software. The third generation was introduced to make the languages more programmer friendly.

The 3GLs spurred the great increase in data processing that occurred in the 1960s and 1970s. In these languages, the program statements are not closely related to the internal characteristics of the computer. Hence, these languages are often referred to as high-level languages.

In general, a statement written in a high-level programming language will expand into several machine language instructions. This is in contrast to assembly languages, where one statement would generate one machine language instruction. 3GLs made programming easier, efficient, and less prone to errors.

High-level languages fall somewhere between natural languages and machine languages. 3GLs include FORTRAN and COBOL, which made it possible for scientists and entrepreneurs to write programs using familiar terms instead of obscure machine instructions.

The widespread use of high-level languages in the early 1960s changed programming into something quite different from what it had been. Programs were written in languages that were more English-like, making them more convenient to use and giving the programmer more time to address a client's problems.

Although 3GLs relieve the programmer of demanding details, they do not provide the flexibility available in low-level languages. However, a few high-level languages such as C and FORTH combine some of the flexibility of assembly languages with the power of high-level languages, but these languages are not well suited to programmers at the beginner level.

Some high-level languages were specifically designed to serve a specific purpose (such as controlling industrial robots or creating graphics), whereas other languages were flexible and considered to be general purpose. Most programmers preferred to use general-purpose high-level languages such as BASIC, FORTRAN, Pascal, COBOL, C++, or Java to write the code for their applications.

Again, a translator is needed to translate the instructions written in a high-level language into the computer-executable machine language. Such translators are commonly known as interpreters and compilers. Each high-level language has many compilers, and there is one for each type of computer.

For example, the machine language generated by one computer's C compiler is not the same as the machine language of some other computer. Therefore, it is necessary to have a C compiler for each type of computer on which the C programs are to be executed.

The 3GLs make it easy to write and debug a program and give a programmer more time to think about its overall logic. Programs written in such languages are portable between machines. For example, a program written in standard C can be compiled and executed on any computer that has a standard C compiler.

Table 1.13 provides the advantages and disadvantages of 3GLs.

Table 1.13 Advantages and disadvantages of 3GLs

Advantages	Disadvantages
• The code is machine independent. • It is easy to learn and use the language. • There are few errors. • It is easy to document and understand the code. • It is easy to maintain the code. • It is easy to detect and correct errors.	• Code may not be optimized. • The code is less efficient. • It is difficult to write a code that controls the CPU, memory, and registers.

Note

Assemblers, linkers, compilers, loaders, and interpreters are all system software, which are discussed in Section 1.13.1.

1.16.4 Fourth Generation: Very High-level Languages

With each generation, programming languages started becoming easier to use and more similar to natural languages. 4GLs are a little different from their prior generation because they are non-procedural. While writing a code using a procedural language, the programmer has to tell the computer how a task is done—add this, compare that, do this if the condition is true, and so on—in a very specific step-by-step manner. In striking contrast, while using a non-procedural language, programmers define what they want the computer to do but they do not supply all the details of how it has to be done.

Although there is no standard rule that defines a 4GL, certain characteristics of such languages include the following:

- The instructions of the code are written in English-like sentences.
- They are non-procedural, so users concentrate on the 'what' instead of the 'how' aspect of the task.
- The code written in a 4GL is easy to maintain.
- The code written in a 4GL enhances the productivity of programmers, as they have to type fewer lines of code to get something done. A programmer supposedly becomes 10 times more productive when he/she writes the code using a 4GL than using a 3GL.

A typical example of a 4GL is the query language, which allows a user to request information from a database with precisely worded English-like sentences. A query language is used as a database user interface and hides the specific details of the database from the user. For example, when working with Structured Query Language (SQL), the programmer just needs to remember a few rules of *syntax* and *logic*, and therefore, it is easier to learn than COBOL or C.

Let us take an example in which a report needs to be generated. The report displays the total number of students enrolled in each class and in each semester. Using a 4GL, the request would look similar to the following:

```
TABLE FILE ENROLMENT
SUM STUDENTS BY SEMESTER BY CLASS
```

Thus, we see that a 4GL is very simple to learn and work with. The same task if written in C or any other 3GL would require multiple lines of code.

The 4GLs are still evolving, which makes it difficult to define or standardize them. The only downside of a 4GL is that it does not make efficient use of a machine's resources. However, the benefit of executing a program quickly and easily far outweighs the extra costs of running it.

1.16.5 Fifth-generation Programming Language

Fifth-generation programming languages (5GLs) are centred on solving problems using the constraints given to a program rather than using an algorithm written by a programmer. Most constraint-based and logic programming languages and some declarative languages form a part of the 5GLS. These languages are widely used in artificial intelligence research. Another aspect of a 5GL is that it contains visual tools to help develop a program. Typical examples of 5GLs include Prolog, OPS5, Mercury, and Visual Basic.

Thus, taking a forward leap, 5GLs are designed to make the computer solve a given problem without the programmer. While working with a 4GL, programmers have to write a specific code to do a work, but with a 5GL, they only have to worry about what problems need to be solved and what conditions need to be met, without worrying about how to implement a routine or an algorithm to solve them.

In general, 5GLs were generally built upon LISP, many originating on the LISP machine, such as ICAD. There are also many frame languages, such as KL-ONE.

In the 1990s, 5GLs were considered the wave of the future, and some predicted that they would replace all other languages for system development (except the low-level languages). During the period ranging from 1982 to 1993, Japan carried out extensive research on and invested a large amount of money into their fifth-generation computer systems project, hoping to design a massive computer network of machines using these tools. However, when large programs were built, the flaws of the approach became more apparent. Researchers began to observe that given a set of constraints defining a particular problem, deriving an efficient algorithm to solve it is itself a very difficult problem. All factors could not be automated and some still require the insight of a programmer.

However, today the fifth-generation languages are pursued as a possible level of computer language. Software vendors across the globe currently claim that their software meets the visual 'programming' requirements of the 5GL concept.

1.17 Programming Paradigms

A programming paradigm is a fundamental style of programming that defines how the structure and basic elements of a computer program will be built. The style of writing programs and the set of capabilities and limitations that a particular programming language has depends on the programming paradigm it supports. While some programming languages strictly follow a single paradigm, others may draw concepts from more than one. The sweeping trend in the evolution of high-level programming languages has resulted in a shift in programming paradigm. These paradigms, in sequence of their application, can be classified as follows:

- Monolithic programming—emphasizes on finding a solution
- Procedural programming—lays stress on algorithms
- Structured programming—focuses on modules
- Object-oriented programming—emphasizes on classes and objects
- Logic-oriented programming—focuses on goals usually expressed in predicate calculus
- Rule-oriented programming—makes use of 'if-then-else' rules for computation
- Constraint-oriented programming—utilizes invariant relationships to solve a problem

Each of these paradigms has its own strengths and weaknesses and no single paradigm can suit all applications. For example, for designing computation-intensive problems, procedure-oriented programming is preferred; for designing a knowledge base, rule-based programming would be the best option; and for hypothesis derivation, logic-oriented programming is used. In this book, we will discuss only the first four paradigms.

1.17.1 Monolithic Programming

Programs written using monolithic programming languages such as assembly language and BASIC consist of global data and sequential code. The global data can be accessed and modified (knowingly or mistakenly) from any part of the program, thereby posing a serious threat to its integrity. A sequential code is one in which all instructions are executed in the specified sequence. In order to change the sequence of instructions, jump statements or 'goto' statements are used. Figure 1.62 shows the structure of a monolithic program. As the name suggests, monolithic programs have just one program module as such programming languages do not support the concept of subroutines. Therefore, all the actions required to complete a particular task are embedded within the same application itself. This not only makes the size of the program large but also makes it difficult to debug and maintain. For all these reasons, monolithic programming language is used only for very small and simple applications where reusability is not a concern.

1.17.2 Procedural Programming

In procedural languages, a program is divided into subroutines that can access global data. To avoid repetition of code, each subroutine performs a well-defined task. A subroutine that needs the service provided by another subroutine can call that subroutine. Therefore, with 'jump', 'goto', and 'call' instructions, the sequence of execution of instructions can be altered. Figure 1.63 shows the structure of a procedural language. FORTRAN and COBOL are two popular procedural programming languages.

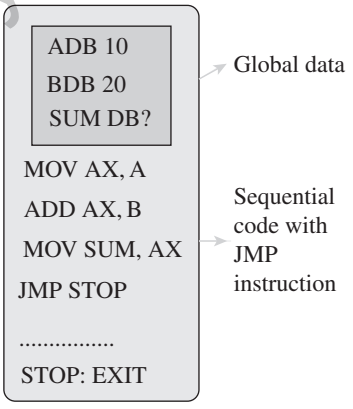


Figure 1.62 Structure of a monolithic program

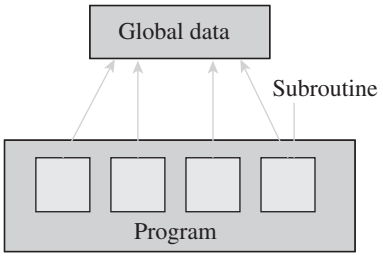


Figure 1.63 Structure of a procedural program

Advantages

- The only goal is to write correct programs
- Programs are easier to write as compared to monolithic programming

Disadvantages

- No concept of reusability
- Requires more time and effort to write programs
- Programs are difficult to maintain
- Global data is shared and therefore may get altered (mistakenly)

1.17.3 Structured Programming

Structured programming, also referred to as modular programming, was first suggested by mathematicians, Corrado Bohm and Guiseppe Jacopini in 1966. It was specifically designed to enforce a logical structure on the program to make it more efficient and easier to understand and modify. Structured programming was basically defined to be used in large programs that require large development team to develop different parts of the same program. Structured programming employs a top-down approach in which the overall program structure is broken down into separate modules. This allows the code to be loaded into memory more efficiently and also be reused in other programs. Modules are coded separately and once a module is written and tested individually, it is then integrated with other modules to form the overall program structure (refer to Figure 1.64). Structured programming is, therefore, based on modularization which groups related statements together into modules. Modularization makes it easier to write, debug, and understand the program. Ideally, modules should not be longer than a page. It is always easy to understand a series of 10 single-page modules than a single 10-page program. For large and complex programs, the overall program structure may further require the need to break the modules into subsidiary pieces. This process continues until an individual piece of code can be written easily. Almost every modern programming language similar to C, Pascal, etc., supports the concepts of structured programming. In addition to the techniques of structured programming for writing modules, it also focuses on structuring its data. In structured programming, the program flow follows a simple sequence and usually avoids the use of 'goto' statements. Besides sequential flow, structured programming also supports selection and repetition as mentioned here.

- Selection allows for choosing any one of a number of statements to execute, based on the current status of the program. Selection statements contain keywords such as 'if', 'then', 'end if', or 'switch' that help to identify the order as a logical executable.

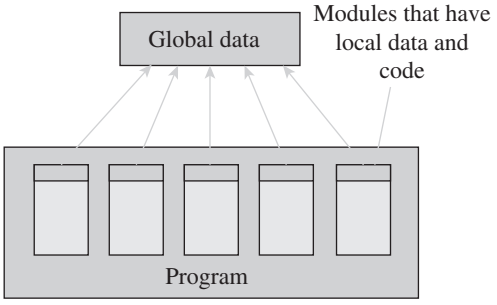


Figure 1.64 Structured program

- In repetition, a selected statement remains active until the program reaches a point where there is a need for some other action to take place. It includes keywords such as 'repeat', 'for', or 'do… until'. Essentially, repetition instructs the program as to how long it needs to continue the function before requesting further instructions.

Advantages

- The goal of structured programming is to write correct programs that are easy to understand and change.
- Modules enhance programmers' productivity by allowing them to look at the big picture first and focus on details later.
- With modules, many programmers can work on a single, large program, with each working on a different module.
- A structured program takes less time to be written than other programs. Modules or procedures written for one program can be reused in other programs as well.
- Each module performs a specific task.
- Each module has its own local data.
- A structured program is easy to debug because each procedure is specialized to perform just one task and every procedure can be checked individually for the presence of any error. In striking contrast, unstructured programs consist of a sequence of instructions that are not grouped for specific tasks. Their logic is cluttered with details and, therefore, difficult to follow.
- Individual procedures are easy to change as well as understand. In a structured program, every procedure has meaningful names and has clear documentation to identify the task performed by it. Moreover, a correctly written structured program is self-documenting and can be easily understood by another programmer.
- More emphasis is given on the code and the least importance is given to the data.

Disadvantages

- Not data-centred
- Global data is shared and therefore may get inadvertently modified
- Main focus is on functions

1.17.4 Object-oriented Programming (OOP)

With the increase in size and complexity of programs, there was a need for a new programming paradigm that could help to develop maintainable programs. To implement this, the flaws in previous paradigms had to be corrected. Consequently, OOP was developed. It treats data as a critical element in the program development and restricts its flow freely around the system. We have seen that monolithic, procedural, and structured programming paradigms are task-based as they focus on the actions the program should accomplish. However, the object-oriented paradigm is task-based and data-based. In this paradigm, all the relevant data and tasks are grouped together in entities known as objects (refer to Figure 1.65). For example, consider a list of numbers. The procedural or structured programming paradigm considers this list as merely a collection of data. Any program that accesses this list must have some procedures or functions to process this list. For example, to find the largest number or to sort the numbers in the list, we need specific procedures or functions to do the task. Therefore, the list is a passive entity as it is maintained by a controlling program rather than having the responsibility of maintaining itself. However, in the object-oriented paradigm, the list and the associated operations are treated as one entity known as an object. In this approach, the list is considered an object consisting of the list, along with a collection of routines for manipulating the list. In the list object, there may be routines for adding a number to the list, deleting a number from the list, sorting the list, etc. The major difference between OOP and traditional approaches is that the program accessing this list need not contain procedures for performing tasks; rather, it uses the routines provided in the object. In other words, instead of sorting the list as in the procedural paradigm, the program asks the list to sort itself. Therefore, we can conclude that the object-oriented paradigm is task-based (as it considers operations) as well as data-based (as these operations are grouped with the relevant data).

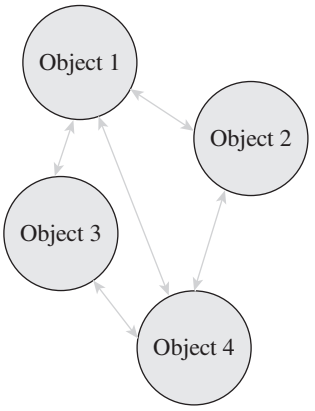


Figure 1.65 Object-oriented paradigm

The striking features of OOP include the following:

- Programs are data centred.
- Programs are divided in terms of objects and not procedures.
- Functions that operate on data are tied together with the data.
- Data is hidden and not accessible by external functions.
- New data and functions can be easily added as and when required.
- Follows a bottom-up approach for problem solving.

In the forthcoming chapters, we are going to study C programming language which supports both procedural as well as structured programming.

1.18 EXAMPLE OF A STRUCTURED PROGRAM

Imagine that your institute wants to create a program to manage the names and addresses of a list of students. For this, you would need to break down the program into the following modules:

- Enter new names and addresses
- Modify existing entries
- Sort entries
- Print the list

Now, each of these modules can be further broken down into smaller modules. For example, the first module can be subdivided into modules such as follows:

- Prompt the user to enter new data

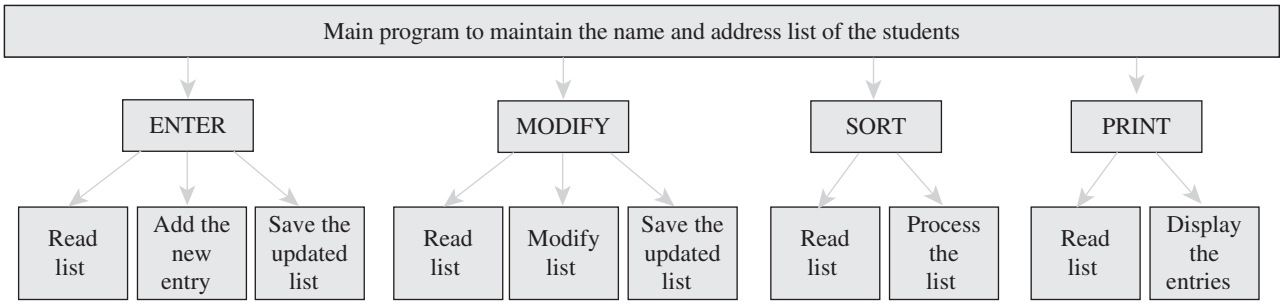


Figure 1.66 Layered program structure

- Read the existing list from the disk
- Add the name and address to the existing list
- Save the updated list to the disk

Similarly, 'Modify existing entries' can be further divided into modules such as follows:

- Read the existing list from disk
- Modify one or more entries
- Save the updated list to disk

Observe that the two sub-modules—'Read the existing list from disk' and 'Save the updated list to disk' are common to both the modules. Hence, once these sub-modules are written, they can be used in both the modules, which require the same tasks to be performed. The structured programming method results in a hierarchical or layered program structure, which is depicted in Figure 1.66.

1.19 SOFTWARE DEVELOPMENT PROCESS

The design and development of correct, efficient, and maintainable programs depend on the approach adopted by the programmer to perform various activities that need to be performed during the development process. The entire program or software (collection of programs) development process is divided into a number of phases, where each phase performs a well-defined task. Moreover, the output of one phase provides the input for its subsequent phase.

The phases in the software development life cycle (SDLC) process is shown in Figure 1.67.

The phases in the SDLC process can be summarized as follows:

Requirements analysis In this phase, the user's expectations are gathered to know why the program/software has to be built. Then, all the gathered requirements are analysed to arrive at the scope or the objective of the overall software product. The last activity in this phase includes documenting every identified requirement of the users in order to avoid any doubts or uncertainty regarding the functionality of the programs.

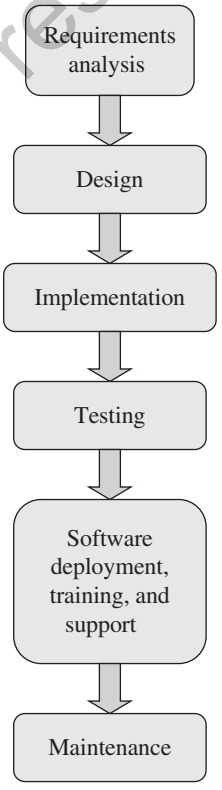


Figure 1.67 Phases in software development life cycle

The functionality, capability, performance, and availability of hardware and software components are all analysed in this phase.

Design The requirements documented in the previous phase acts as an input to the design phase. In the design phase, a plan of actions is made before the actual development process can start. This plan will be followed throughout the development process. Moreover, in the

design phase, the core structure of the software/program is broken down into modules. The solution of the program is then specified for each module in the form of algorithms or flowcharts. The design phase, therefore, specifies how the program/software will be built.

Implementation In this phase, the designed algorithms are converted into program code using any of the high-level languages. The particular choice of language will depend on the type of program, such as whether it is a system or an application program. While C is preferred for writing system programs, Visual Basic might be preferred for writing an application program. The program codes are tested by the programmer to ensure their correctness.

This phase is also called construction or code generation phase as the code of the software is generated in this phase. While constructing the code, the development team checks whether the software is compatible with the available hardware and other software components that were mentioned in the Requirements Specification Document created in the first phase .

Testing In this phase, all the modules are tested together to ensure that the overall system works well as a whole product. Although individual pieces of codes are already tested by the programmers in the implementation phase, there is always a chance for bugs to creep into the program when the individual modules are integrated to form the overall program structure. In this phase, the software is tested using a large number of varied inputs, also known as test data, to ensure that the software is working as expected by the user's requirements that were identified in the requirements analysis phase.

Software deployment, training, and support After the code is tested and the software or the program has been approved by the users, it is installed or deployed in the production environment. This is a crucial phase that is often ignored by most developers. Program designers and developers spend a lot of time to create software but if nobody in an organization knows how to use it or fix up certain problems, then no one would like to use it. Moreover, people are often resistant to change and avoid venturing into an unfamiliar area, so as a part of the deployment phase, it has become very crucial to have training classes for the users of the software.

Maintenance Maintenance and enhancements are ongoing activities that are done to cope with newly discovered problems or new requirements. Such activities may take a long time to complete as the requirement may call for the addition of new code that does not fit the original design or an extra piece of code, required to fix an unforeseen problem. As a general rule, if the cost of the maintenance phase exceeds 25% of the prior phase's cost, then it clearly indicates that the overall quality of at least one prior phase is poor. In such cases, it is better to re-build the software (or some modules) before the maintenance cost shoots out of control.

1.20 PROGRAM DESIGN TOOLS: ALGORITHMS, FLOWCHARTS, PSEUDOCODES

This section will deal with different tools, which are used to design solution(s) of a given problem at hand.

1.20.1 Algorithms

The typical meaning of an algorithm is a formally defined procedure for performing some calculation. If a procedure is formally defined, then it must be implemented using some formal language, and such languages are known as *programming languages*. The algorithm gives the logic of the program, that is, a step-by-step description of how to arrive at a solution.

In general terms, an algorithm provides a blueprint to writing a program to solve a particular problem. It is considered to be an effective procedure for solving a problem in a finite number of steps. That is, a well-defined algorithm always provides an answer, and is guaranteed to terminate.

Algorithms are mainly used to achieve *software reuse*. Once we have an idea or a blueprint of a solution, we can implement it in any high-level language, such as C, C++, Java, and so on. In order to qualify as an algorithm, a sequence of instructions must possess the following characteristics:

- Be precise
- Be unambiguous
- Not even a single instruction must be repeated infinitely.
- After the algorithm gets terminated, the desired result must be obtained.

Control Structures Used In Algorithms

An algorithm has a finite number of steps and some steps may involve decision-making and repetition. Broadly speaking, an algorithm may employ three control structures, namely, sequence, decision, and repetition.

Sequence Sequence means that each step of the algorithm is executed in the specified order. An algorithm to add two numbers is given in Figure 1.68. This algorithm performs the steps in a purely sequential order.

```
Step 1: Input first number as A
Step 2: Input second number as B
Step 3: Set Sum = A + B
Step 4: Print Sum
Step 5: End
```

Figure 1.68 Algorithm to add two numbers

Decision Decision statements are used when the outcome of the process depends on some condition. For example, `if x = y, then print "EQUAL"`. Hence, the general form of the `if` construct can be given as follows:

```
IF condition then process
```

A condition in this context is any statement that may evaluate either to a true value or a false value. In the preceding example, the variable *x* can either be equal or not equal to *y*. However, it cannot be both true and false. If the condition is true then the process is executed.

A decision statement can also be stated in the following manner:

```
IF condition
  then process1
ELSE process2
```

This form is commonly known as the if-else construct. Here, if the condition is true then `process1` is executed, else `process2` is executed. An algorithm to check the equality of two numbers is shown in Figure 1.69.

```
Step 1: Input first number as A
Step 2: Input second number as B
Step 3: IF A = B
           Print "Equal"
        ELSE
           Print "Not equal"
        [END of IF]
Step 4: End
```

Figure 1.69 Algorithm to test the equality of two numbers

Repetition Repetition, which involves executing one or more steps for a number of times, can be implemented using constructs such as the `while`, `do-while`, and `for loops`. These loops execute one or more steps until some condition is true. Figure 1.70 shows an algorithm that prints the first 10 natural numbers.

```
Step 1: [initialize] Set I = 1, N = 10
Step 2: Repeat Steps 3 and 4 while I <= N
Step 3: Print I
Step 4: SET I = I + 1
        [END OF LOOP]
Step 5: End
```

Figure 1.70 Algorithm to print the first 10 natural numbers

Example 1.1

Write an algorithm for interchanging/swapping two values.

Solution

```
Step 1: Input first number as A
Step 2: Input second number as B
Step 3: Set temp = A
Step 4: Set A = B
Step 5: Set B = temp
Step 6: Print A, B
Step 7: End
```

Example 1.2

Write an algorithm to find the larger of two numbers.

Solution

```
Step 1: Input first number as A
Step 2: Input second number as B
Step 3: IF A > B
          Print A
        ELSE IF A < B
          Print B
        ELSE
          Print "The numbers are equal"
        [END OF IF]
Step 4: End
```

Example 1.3

Write an algorithm to find whether a number is even or odd.

Solution

```
Step 1: Input number as A
Step 2: IF A % 2 = 0
          Print "Even"
        ELSE
          Print "Odd"
        [END OF IF]
Step 3: End
```

Example 1.4

Write an algorithm to print the grade obtained by a student using the following rules:

Marks	Grade
Above 75	O
60-75	A
50-60	B
40-50	C
Less than 40	D

Solution

```
Step 1: Enter the marks obtained as M
Step 2: IF M > 75
          Print "O"
Step 3: IF M >= 60 and M < 75
          Print "A"
Step 4: IF M >= 50 and M < 60
          Print "B"
Step 5: IF M >= 40 and M < 50
          Print "C"
        ELSE
          Print "D"
       [END OF IF]
Step 6: End
```

Example 1.5

Write an algorithm to find the sum of first N natural numbers.

Solution

```
Step 1: Input N
Step 2: Set I = 1, sum = 0
Step 3: Repeat Steps 4 and 5 while I <= N
Step 4: Set sum = sum + I
Step 5: Set I = I + 1
        [END OF LOOP]
Step 6: Print sum
Step 7: End
```

1.20.2 Flowcharts

A flowchart is a graphical or symbolic representation of a process. It is basically used to design and document virtually complex processes to help the viewers to visualize the logic of the process, so that they can gain a better understanding of the process and find flaws, bottlenecks, and other less obvious features within it.

When designing a flowchart, each step in the process is depicted by a different symbol and is associated with a short description. The symbols in the flowchart (refer Figure 1.71) are linked together with arrows to show the flow of logic in the process.

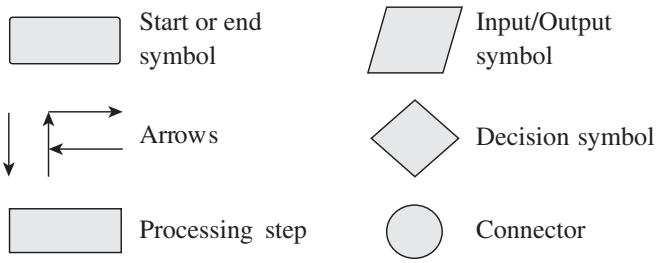


Figure 1.71 Symbols of flowchart

The symbols used in a flowchart include the following:

- *Start and end symbols* are also known as the terminal symbols and are represented as circles, ovals, or rounded rectangles. Terminal symbols are always the first and the last symbols in a flowchart.
- *Arrows* depict the flow of control of the program. They illustrate the exact sequence in which the instructions are executed.
- *Generic processing step,* also called as an activity, is represented using a rectangle. Activities include instructions such as `add a to b, save the result`. Therefore, a processing symbol represents arithmetic and data movement instructions. When more than one process has to be executed simultaneously, they can be placed in the same processing box. However, their execution will be carried out in the order of their appearance.
- *Input/Output symbols* are represented using a parallelogram and are used to get inputs from the users or display the results to them.
- A *conditional or decision symbol* is represented using a diamond. It is basically used to depict a Yes/No question or a True/False test. The two symbols coming out of it, one from the bottom point and the other from the right point, corresponds to Yes or True, and No or False, respectively. The arrows should always be labelled. A decision symbol in a flowchart can have more than two arrows, which indicates that a complex decision is being taken.
- *Labelled connectors* are represented by an identifying label inside a circle and are used in complex or multi-sheet diagrams to substitute for arrows. For each label, the 'outflow' connector must have one or more 'inflow' connectors. A pair of identically labelled connectors is used to indicate a continued flow when the use of lines becomes confusing.

Significance of Flowcharts

A flowchart is a diagrammatic representation that illustrates the sequence of steps that must be performed to solve a problem. It is usually drawn in the early stages of formulating computer solutions. It facilitates communication between programmers and users. Once a flowchart is drawn, programmers can make users understand the solution easily and clearly.

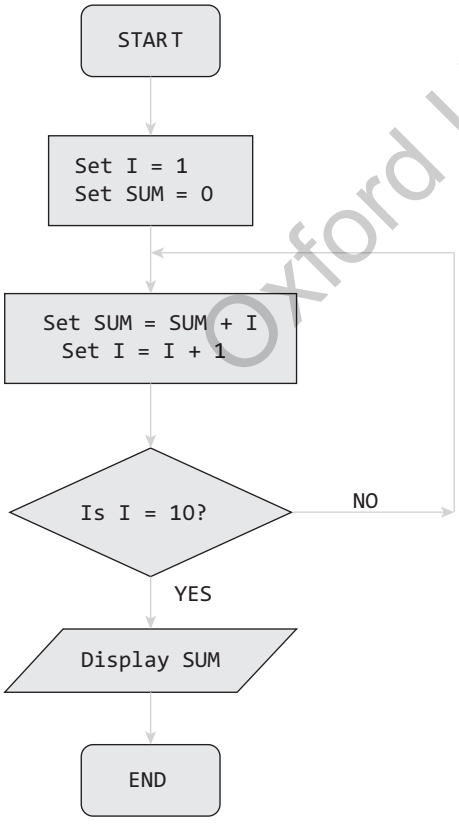
Flowcharts are very important in the programming of a problem as they help the programmers to understand the logic of complicated and lengthy problems. Once a flowchart is drawn, it becomes easy for the programmers to write the program in any high-level language. Hence, the flowchart has become a necessity for better documentation of complex programs.

A flowchart follows the top-down approach in solving problems.

Example 1.6

Draw a flowchart to calculate the sum of the first 10 natural numbers.

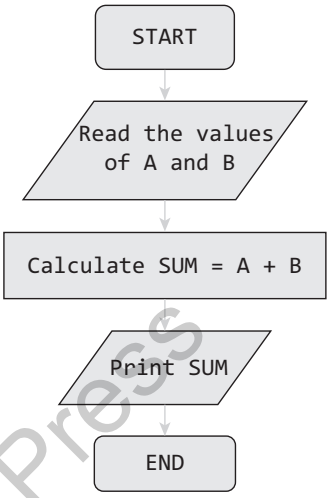
Solution



Example 1.7

Draw a flowchart to add two numbers.

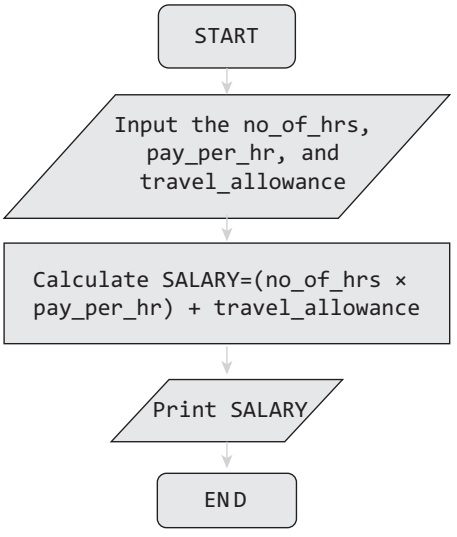
Solution



Example 1.8

Draw a flowchart to calculate the salary of a daily wager.

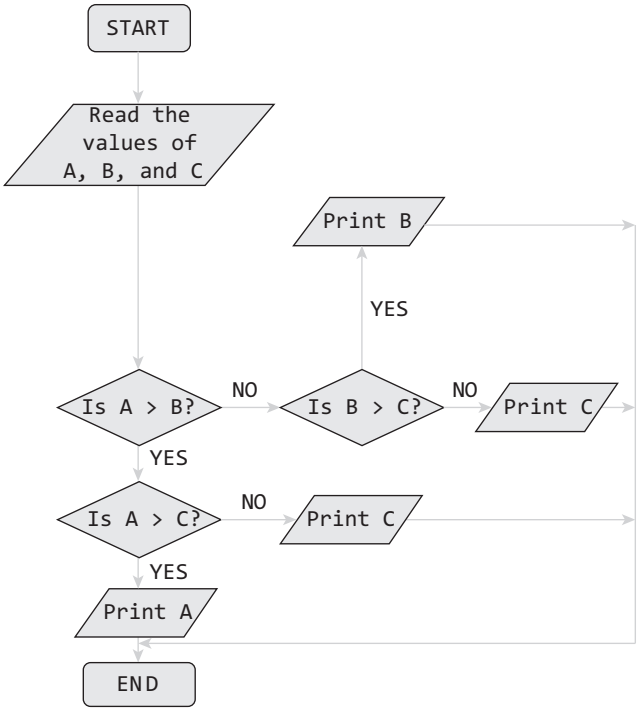
Solution



Example 1.9

Draw a flowchart to determine the largest of three numbers.

Solution



Advantages

- They are very good communication tools to explain the logic of a system to all concerned. They help to analyse the problem in a more effective manner.
- They are also used for program documentation. They are even more helpful in the case of complex programs.
- They act as a guide or blueprint for the programmers to code the solution in any programming language. They direct the programmers to go from the starting point of the program to the ending point without missing any step in between. This results in error-free programs.
- They can be used to debug programs that have error(s). They help the programmers to easily detect, locate, and remove mistakes in the program in a systematic manner.

Limitations

- Drawing flowcharts is a laborious and a time-consuming activity. Just imagine the effort required to draw a flowchart of a program having 50,000 statements in it!
- Many a times, the flowchart of a complex program becomes complex and clumsy.
- At times, a little bit of alteration in the solution may require complete redrawing of the flowchart.
- The essentials of what is done may get lost in the technical details of how it is done.
- There are no well-defined standards that limit the details that must be incorporated into a flowchart.

1.20.3 Pseudocodes

Pseudocode is a compact and informal high-level description of an algorithm that uses the structural conventions of a programming language. It facilitates designers to focus on the logic of the algorithm without getting bogged down by the details of language syntax. An ideal pseudocode must be complete, describing the entire logic of the algorithm, so that it can be translated straightaway into a programming language.

It is basically meant for human reading rather than machine reading, so it omits the details that are not essential for humans. Such details include variable declarations, system-specific code, and subroutines.

Pseudocodes are an outline of a program that can easily be converted into programming statements. They consist of short English phrases that explain specific tasks within a program's algorithm. They should not include keywords in any specific computer language.

The sole purpose of pseudocodes is to enhance human understandability of the solution. They are commonly used in textbooks and scientific publications for documenting algorithms, and for sketching out the program structure before the actual coding is done. This helps even non-programmers to understand the logic of the designed solution. There are no standards defined for writing a pseudocode, because a pseudocode is not an executable program. Flowcharts can be considered as graphical alternatives to pseudocodes, but require more space on paper.

Example 1.10

Write a pseudocode for calculating the price of a product after adding the sales tax to its original price.

Solution

```
1. Read the price of the product
2. Read the sales tax rate
3. Calculate sales tax = price of the item
   x; sales tax rate
4. Calculate total price = price of the
   product + sales tax
5. Print total price
6. End
Variables: price of the item, sales tax
   rate, sales tax, total price
```

Example 1.11

Write a pseudocode to calculate the weekly wages of an employee. The pay depends on wages per hour and the number of hours worked. Moreover, if the employee has worked for more than 30 hours, then he or she gets twice the wages per hour, for every extra hour that he or she has worked.

Solution

```
1. Read hours worked
2. Read wages per hour
3. Set overtime charges to 0
4. Set overtime hrs to 0
5. IF hours worked > 30 then
a. Calculate overtime hrs = hours worked –
   30
b. Calculate overtime charges = overtime hrs
   × (2 × wages per hour)
c. Set hours worked = hours worked –
   overtime hrs
   ENDIF
6. Calculate salary = (hours worked × wages
   per hour) + overtime charges
7. Display salary
8. End
Variables: hours worked, wages per hour,
   overtime charges, overtime hrs, salary
```

Example 1.12

Write a pseudocode to read the marks of 10 students. If marks is greater than 50, the student passes, else the student fails. Count the number of students passing and failing.

Solution

```
1. Set pass to 0
2. Set fail to 0
3. Set no of students to 1
4. WHILE no of students ≤ 10
    a. input the marks
    b. IF marks >= 50 then
         Set pass = pass + 1
       ELSE
         Set fail = fail + 1
       ENDIF
   ENDWHILE
5. Display pass
6. Display fail
7. End
Variables: pass, fail, no of students, marks
```

1.21 TYPES OF ERRORS

While writing programs, very often we get errors in our programs. These errors if not removed will either give erroneous output or will not let the compiler to compile the program. These errors are broadly classified under four groups as shown in Figure 1.72.

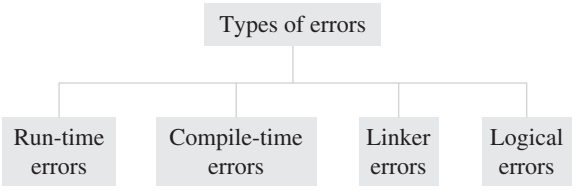


Figure 1.72 Types of Errors

Run-time Errors As the name suggests, run-time errors occur when the program is being run executed. Such errors occur when the program performs some illegal operations like

- dividing a number by zero
- opening a file that already exists
- lack of free memory space
- finding square or logarithm of negative numbers

Run-time errors may terminate program execution, so the code must be written in such a way that it handles all sorts of unexpected errors rather terminating it unexpectedly.

Compile-time Errors Again as the name implies, compile-time errors occur at the time of compilation of the program. Such errors can be further classified as follows:

Syntax Errors Syntax errors are generated when rules of a programming language are violated.

Semantic Errors Semantic errors are those errors which may comply with rules of the programming language but are not meaningful to the compiler.

Logical Errors Logical errors are errors in the program code that result in unexpected and undesirable output which is obviously not correct. Such errors are not detected by the compiler, and programmers must check their code line by line or use a debugger to locate and rectify the errors. Logical errors occur due to incorrect statements.

Linker Errors These errors occur when the linker is not able to find the function definition for a given prototype.

1.22 TESTING AND DEBUGGING APPROACHES

Testing is an activity that is performed to verify correct behaviour of a program. It is specifically carried out with an intent to find errors. Ideally testing should be conducted at all stages of program development. However, in the implementation stage, the following three types of tests can be conducted:

Unit Tests

Unit testing is applied only on a single unit or module to ensure whether it exhibits the expected behaviour.

Integration Tests These tests are a logical extension of unit tests. In this test, two units that have already been tested are combined into a component and the interface between them is tested. The guiding principle is to test combinations of pieces and then gradually expanding the component to include other modules as well. This process is repeated until all the modules are tested together. The main focus of integration testing is to identify errors that occur when the units are combined.

System Tests System testing checks the entire system. For example, if our program code consists of three modules then each of the module is tested individually using unit tests and then system test is applied to test this entire system as one system.

Debugging, on the other hand, is an activity that includes execution testing and code correction. The main aim of debugging is locating errors in the program code. Once the errors are located, they are then isolated and fixed to produce an error-free code. Different approaches applied for debugging a code includes:

Brute-Force Method In this technique, a printout of CPU registers and relevant memory locations is taken, studied, and documented. It is the least efficient way of debugging a program and is generally done when all the other methods fail.

Backtracking Method It is a popular technique that is widely used to debug small applications. It works by locating the first symptom of error and then tracing backward across the entire source code until the real cause of error is detected. However, the main drawback of this approach is that with increase in number of source code lines, the possible backward paths become too large to manage.

Cause Elimination In this approach, a list of all possible causes of an error is developed. Then relevant tests are carried out to eliminate each of them. If some tests indicate that a particular cause may be responsible for an error then the data are refined to isolate the error.

Example 1.13

Let us take a problem, collect its requirement, design the solution, implement it in C and then test our program.

Problem Statement To develop an automatic system that accepts marks of a student and generates his/her grade.

Requirements Analysis Ask the users to enlist the rules for assigning grades. These rules are:

Marks	Grade
Above 75	O
60-75	A
50-60	B
40-50	C
Less than 40	D

Design In this phase, write an algorithm that gives a solution to the problem.

```
Step 1: Enter the marks obtained as M
Step 2: If M > 75 then print "O"
Step 3: If M >= 60 and M < 75 then print "A"
Step 4: If M >= 50 and M < 60 then print "B"
Step 5: If M >= 40 and M < 50 then print "C"
        else
        print "D"
Step 6: End
```

Implementation Write the C program to implement the proposed algorithm.

```
#include <stdio.h>
#include <conio.h>
int main()
{ int marks;
  char grade;
  clrscr();
  printf("\n Enter the marks of the
   student:");
  scanf("%d", &marks);
  if (marks<0 || marks >100)
  {   printf("\n Not Possible");
      exit(1);
  }
```

```
    if(marks>=75)
      grade = 'O';
    else if(marks>=60 && marks<75)
      grade = 'A';
    else if(marks>=50 && marks<60)
      grade = 'B';
    else if(marks>=40 && marks<50)
      grade = 'C';
    else
      grade = 'D';
    printf("\n GRADE = %c", grade);
  }
```

Test The above program is then tested with different test data to ensure that the program gives correct output for all relevant and possible inputs. The test cases are shown in the table given below.

Test Case ID	Input	Expected Output	Actual Output
1	–12	Not Possible	Not Possible
2	112	Not Possible	Not Possible
3	32	D	D
4	46	C	C
5	54	B	B
6	68	A	A
7	91	O	O
8	40	C	C
9	50	B	B
10	60	A	A
11	75	O	O
12	100	O	O
13	0	D	D

Note in the above table, we have identified test cases for the following,

1. "Not Possible" Combinations
2. A middle value from each range
3. Boundary values for each range

POINTS TO REMEMBER

- A *computer* is an electronic machine that accepts data and instructions and performs computations on the data based on those instructions.
- Computers are used in all interactive devices, such as cellular telephones, GPS units, portable organizers, ATMs, and gas pumps.
- Modern-day computers are based on the principle of the stored program concept, which was introduced by Sir John von Neumann in the late 1940s.
- The speed of the computer is usually given in nanoseconds and picoseconds.
- The term *computer generation* refers to the different advancements of new computer technology.
- A computer has two parts—hardware, which does all the physical work computers are known for, and software, which tells the hardware what to do and how to do it.
- The CPU is a combination of the ALU and the CU. The CPU is known as the brain of the computer system.
- The CU is the central nervous system of the entire computer system. It manages and controls all the components of the computer system.
- An *input device* is used to feed data and instructions into the computer.
- *Output devices* are electromechanical devices that accept digital data from the computer and convert them into human understandable language.
- Computer memory is an internal storage area in the computer that is used to store data and programs either temporarily or permanently. It also stores the intermediate results and the final results of processing.
- While the main memory holds instructions and data when a program is being executed, the auxiliary or the secondary memory holds data and programs not currently in use and provides long-term storage.
- The primary memory is volatile, so the data can be retained in it only when the power is on. Moreover, it is very expensive and therefore limited in capacity.
- On the contrary, the secondary memory stores data or instructions permanently, even when the power is turned off. It is cheap and can store large volumes of data, which is highly portable.
- Processor registers are located inside the processor and are therefore directly accessed by the CPU. Each register stores a word of data (which is either 32 or 64 bits).
- *Cache memory* is an intermediate form of storage between the ultra-fast registers and the RAM.
- Computer software is written by computer programmers using a programming language.
- *Application software* is designed to solve a particular problem for users.
- *System software* represents programs that allow the hardware to run properly. System software acts as an

interface between the hardware of the computer and the application software that users need to run on the computer.

- Compilers and interpreters are special types of programs that convert source code written in a programming language (source language) into machine language comprising of just two digits—1s and 0s (target language).
- The number of unique digits used to form numbers within a number system is called radix of that system.

  Decimal number system has a radix of 10, binary has a radix of 2, octal has a radix of 8, and hexadecimal has a radix of 16.
- Every programming language has a vocabulary of syntax and semantics for instructing a computer to perform specific tasks.
- While high-level programming languages are easy for the humans to read and understand, the computer actually understands the machine language, which consists of only numbers.
- Second-generation programming languages comprise the assembly languages which use symbols to represent machine language instructions.
- An assembly language statement consists of a label, an operation code, and one or more operands. Labels are used to identify and refer instructions in the program. The operation code (opcode) is a mnemonic that specifies the operation that has to be performed, such as *move*, *add*, *subtract*, or *compare*. The operand specifies the register or the location in the main memory where the data to be processed is located.
- Once the modules are coded and tested, the object files of all the modules are combined together by the linker to form the final executable file.
- 3GLs (like FORTRAN, COBOL) made it possible for scientists and business people to write programs.
- While working with 4GLs, programmers define only what they want the computer to do, without supplying all the details of how it has to be done.
- 5GLs are centred on solving problems using the constraints given to the program rather than using an algorithm written by a programmer. They are widely used in artificial intelligence research.
- Programs written using monolithic programming languages such as assembly language and BASIC consist of global data and sequential code.
- In procedural languages, a program is divided into subroutines that can access global data.
- Structured programming employs a top-down approach in which the overall program structure is broken down into separate modules.
- In unstructured programming, programmers write small and simple programs consisting of only one main program.
- Object-oriented programming treats data as a critical element in the program development and restricts its flow freely around the system.
- The entire program or software (collection of programs) development process is divided into a number of phases, where each phase performs a well-defined task.
- During requirements analysis, users' expectations are gathered to know why the program/software has to be built.
- In the design phase, a plan of action is made.
- In the implementation phase, the designed algorithms are converted into program code using any of the high-level languages.
- In the testing phase, all the modules are tested together to ensure that the overall system works well as a whole product.
- After the code is tested and the software or the program has been approved by the users, it is then installed or deployed in the production environment.
- Maintenance and enhancements are on-going activities that are done to cope with newly discovered problems or new requirements.

GLOSSARY

Computer A machine that takes instructions and performs computations based on those instructions.

Data A collection of raw facts or figures.

Information Processed data that provide answers to 'who', 'what', 'where', and 'when' types of questions.

Knowledge The application of data and information to answer the 'how' part of the question.

Input The process of entering data and instructions into the computer system.

Output The process of giving the result of data processing to the outside world.

Processing The process of performing operations on the data as per the instructions specified by the user.

Storage The process of saving data and instructions permanently in the computer so that it can be used for processing.

Instructions Commands given to the computer that tell what it has to do.

Bit It is short form of binary digit. It is the smallest possible unit of data, which can either be 0 or 1.

Byte A group of eight bits.

Word A group of two bytes.

Program A set of instructions that are arranged in a sequence to guide a computer to find a solution for the given problem.

Programming The process of writing a program.

Hard copy output devices Output devices that produce a physical form of output.

Soft copy output devices Output devices that produce an electronic version of an output.

Memory An internal storage area in the computer used to store data and programs either temporarily or permanently.

DRAM A type of RAM that must be refreshed multiple times in a second to retain its data contents.

SRAM A type of RAM that holds data without an external refresh as long as it is powered.

Programmable read-only memory A type of ROM that can be programmed using high voltages.

Erasable programmable read only memory A type of ROM that can be erased and re-programmed. The EPROM can be erased by exposing the chip to strong ultraviolet light.

Flash memory A type of EEPROM in which the contents can be erased under software control. The most flexible type of ROM.

Software A set of programs.

Command line interface Command line interface (CLI) is a type of interface in which users interact with a program.

Graphical user interface Graphical user interface (GUI) is a type of user interface that enables users to interact with programs in more ways than typing. A GUI offers graphical icons and visual indicators to display the information and actions available to a user.

Basic input output system (BIOS) Program that tells the computer what to do when it starts up, e.g., running hardware diagnostics and loading the operating system into RAM.

Machine language The lowest level of programming that was used to program the first stored-program computer systems and is the only language that the computer understands.

Programming language A language specifically designed to express computations that can be performed by the computer.

Programming paradigm A fundamental style of programming that defines how the structure and basic elements of a computer program will be built.

Translator A computer program, which translates a code written in one programming language to a code in another language that the computer understands.

Assembler System software that converts the code written in assembly language into machine language.

Compiler/Interpreter System software that translates the source code from a high-level programming language to a lower-level language.

Loader System software that copies programs from a storage device to the main memory, where they can be executed.

Algorithm A formally defined procedure for performing some calculation and provides a blueprint to write a program that solves a particular problem.

Flowchart A graphical or symbolic representation of a process.

Pseudocode A compact and informal high-level description of an algorithm that uses the structural conventions of a programming language.

Compile-time errors These are errors that occur at the time of compilation of the program

Debugging An activity that includes execution testing and code correction. The main aim of debugging is to locate errors in the program code.

Runtime errors These are errors that occur when the program is being executed.

Testing An activity performed to verify the correct behaviour of a program. It is specifically carried out with the intent to find errors.

EXERCISES

Fill in the blanks

1. A program is the _____.
2. Computers operate on _____ based on _____.
3. The speed of computers is expressed in _____ or _____.
4. Raw facts or figures are called _____.
5. _____ and _____ are examples of first-generation computing devices.
6. Second-generation computers were first developed for the _____ industry.
7. _____ packages allow easy manipulation and analysis of data organized in rows and columns.

8. CRAY-1, CRAY-2, Control Data CYBER 205, and ETA A-10 are _____.
9. _____ concept was introduced by Sir John von Neumann in the late 1940s.
10. Android Jellybean, Windows, and iOS are all examples of popular operating systems used in _____ and _____.
11. _____ unit directs and coordinates the computer operations.
12. Intermediate results during program execution are stored in _____.
13. _____ stores the address of the data or instruction to be fetched from memory.
14. An instruction consists of _____ and _____.
15. The instruction cycle is repeated continuously until _____.
16. Buses in a computer system can carry _____, _____, and _____.
17. In an instruction, _____ specifies the computation to be performed.
18. Giga is _____ and tera is _____.
19. ____ instructs the hardware what to do and how to do it.
20. The hardware needs a _____ to instruct what has to be done.
21. _____ is used to feed data and instructions into the computer.
22. _____ captures everything on the screen as an image.
23. _____ is used to enter information or write on the touchscreen of a phone.
24. _____ technology is used for electronically extracting data from marked fields.
25. _____ converts analog signals generated through a microphone into digital data.
26. _____ capture videos that can be transferred via the Internet in real-time.
27. _____ allow the users to talk and listen at the same time.
28. The resolution of a printer means _____.
29. The _____ memory holds data and programs that are currently being executed by the CPU.
30. _____ memory is volatile.
31. _____ memory stores data or instructions permanently.
32. _____ are the fastest of all forms of computer data storage.
33. Static RAM is made of _____.
34. _____ is a one-time programmable ROM.
35. The process of writing data to an optical disk is called _____.
36. The process of writing a program is called _____.
37. ____ is used to write computer software.
38. ____ transforms source code into binary language.
39. ____ allows a computer to interact with additional hardware devices such as printers, scanners, and video cards.
40. _____helps in coordinating system resources and allows other programs to execute.
41. ____ provides a platform for running application software.
42. _____ is a software package that enables its users to create, edit, print, and save documents for future retrieval and reference.
43. Information from a database is extracted in the form of a _____.
44. Adobe Photoshop is an example of _____ software.
45. _____ is a good language for processing numerical data.
46. Assembly language statement consists of a _____, _____, and one or more _____.
47. _____ is used to convert assembly-level program into machine language.
48. _____ and _____ are used to translate the instructions written in high-level language into computer-executable machine language.
49. Fifth-general programming languages are widely used in _____.
50. The object file is created when _____.
51. _____ is extensively used for writing efficient codes for operating systems and compilers.
52. Programs written in _____ are robust, secure, and reliable.
53. _____ and _____ statements are used to change the sequence of execution of instructions.
54. _____ paradigm supports bottom-up approach of problem solving.
55. FORTRAN and COBOL are two popular _____ programming languages.
56. _____ is a formally defined procedure for performing some calculation.
57. _____ statements are used when the outcome of the process depends on some condition.
58. Repetition can be implemented using constructs such as _____, _____, and _____.
59. The _____ symbol is always the first and the last symbol in a flowchart.
60. _____ is a form of structured English that describes algorithms.
61. _____ is used to express algorithms and as a mode of human communication.

62. In the _____ phase, a plan of actions is made.
63. In the _____ phase, designed algorithms are converted into program code.
64. User's expectations are gathered in the _____ phase.

Multiple-choice Questions

1. A computer works on _____ given to it.
   (a) Computations (b) Instructions
   (c) Data (d) b and c
2. Computer is a _____ machine.
   (a) Electrical (b) Mechanical
   (c) Electronic (d) Physical
3. _____ comprises processed data.
   (a) Data (b) Information
   (c) Knowledge (d) Instructions
4. Commands given to the computer that tells what it has to do are _____ .
   (a) Data (b) Information
   (c) Knowledge (d) Instructions
5. Which generation of computers were used in the period 1955–1964?
   (a) First (b) Second
   (c) Third (d) Fourth
6. Which of the following were used for manufacturing first generation computers?
   (a) Vacuum tubes (b) Transistors
   (c) Integrated chips (d) ULSI
7. Select the computer(s) in the first generation of computers.
   (a) ENIAC (b) EDVAC
   (c) EDSAC (d) All of these
8. Which was the first commercial computer delivered to a business client?
   (a) UNIVAC (b) ENIAC
   (c) EDSAC (d) None of these
9. Which technology was used to manufacture second generation computers?
   (a) Vacuum tubes (b) Transistors
   (c) ICs (d) None of these
10. Select the computer(s) in the second generation of computers.
    (a) UNIVAC LARC (b) EDVAC
    (c) EDSAC (d) All of these
11. Which generation of computers were manufactured using ICs with LSI and later with VLSI technology?
    (a) First (b) Second
    (c) Third (d) Fourth
12. In which computer generation did microcomputers come into existence?
    (a) First (b) Second
    (c) Third (d) Fourth
13. Currently on which generation of computers are we working?
    (a) First (b) Fifth
    (c) Third (d) Fourth
14. Name an Indian supercomputer.
    (a) UNIVAC LARC (b) EDVAC
    (c) EDSAC (d) CRAY XMP
15. The process of entering data and instructions into the computer system is called _____.
    (a) Input (b) Output
    (c) Processing (d) Result
16. Computer understands only _____ language.
    (a) Assembly (b) Binary
    (c) High level (d) SQL
17. In which part of the CPU are all computations performed?
    (a) CU (b) MU
    (c) ALU (d) Registers
18. _____ is the main input device.
    (a) Mouse (b) Joystick
    (c) Keyboard (d) Touch screen
19. Name a cursor control device widely used in computer games and CAD/CAM applications.
    (a) Keyboard (b) Joystick
    (c) Mouse (d) Stylus
20. Which output device is used in home theatre systems?
    (a) Printer (b) Plotter
    (c) Projector (d) Speakers
21. A printer can print carbon copies of a document. Which printer are we talking about?
    (a) Laser (b) Inkjet
    (c) Dot matrix (d) Thermal
22. Which device will you use to draw maps?
    (a) Laser printer (b) Projector
    (c) Plotter (d) Dot matrix printer
23. _____ is the leading vendor of plotters worldwide.
    (a) Hewlett-Packard (b) Google
    (c) Microsoft (d) Nokia
24. Which memory holds data and programs not currently in use and provides long-term storage?
    (a) RAM (b) ROM
    (c) Primary memory (d) Secondary memory

25. Which storage device will you use to back up large amount of data?
    (a) Magnetic hard disk (b) ROM
    (c) RAM (d) Magnetic tape
26. Identify the storage device—an optical disc with storage capacity from 4.7 GB to 17 GB.
    (a) CD (b) Blu Ray
    (c) DVD (d) Hard disk
27. Which of the following enables the users to interact with hardware components efficiently?
    (a) Application software
    (b) Communication software
    (c) Presentation software
    (d) System software
28. Which of the following copies programs from a storage device to the main memory?
    (a) Compiler (b) Interpreter
    (c) Assembler (d) Loader
29. Code written in which language can be directly executed by the computer?
    (a) Compiled (b) Assembly
    (c) Binary (d) Interpreted
30. _____ is a mnemonic that specifies the operation that has to be performed.
    (a) Operand (b) Opcode
    (c) Label (d) Comment
31. A group of 4 binary digits is called a _____.
    (a) Bit (b) Nibble
    (c) Byte (d) Word
32. A group of 8 binary digits is called a _____.
    (a) Bit (b) Nibble
    (c) Byte (d) Word
33. The brain of the computer is the
    (a) control unit (b) ALU
    (c) CPU (d) All of these
34. The memory used by the CPU to store instructions and data that are repeatedly required to execute programs to improve overall system performance is
    (a) primary memory (b) auxiliary memory
    (c) cache memory (d) fl ash memory
35. Magnetic tapes, floppy disks, optical disks, flash memory, and hard disks are examples of
    (a) primary memory (b) auxiliary memory
    (c) cache memory (d) flash memory
36. The memory used in MP3 players, PDAs, laptops, and digital audio players is
    (a) primary memory (b) optical memory
    (c) cache memory (d) flash memory
37. The component of the processor that controls the flow of data through the computer system is
    (a) BIU (b) execution unit
    (c) CU (d) ALU
38. Which keys are used by applications and operating systems to perform specific commands?
    (a) Typing keys (b) Arrow keys
    (c) Control keys (d) Function keys
39. Select the optical devices from the following options:
    (a) MICR (b) Barcode reader
    (c) Scanner (d) All of these
40. Select the printer that uses impact printer technology from the following options:
    (a) Daisy wheel (b) Laser
    (c) Band (d) Inkjet
41. Which type of screen is used in gaming devices, clocks, watches, calculators, and telephones?
    (a) LCD (b) Plasma
    (c) CRT (d) All of these
42. BIOS is stored in
    (a) RAM (b) ROM
    (c) hard disk (d) none of these
43. Which of the following languages should not be used for organizing large programs?
    (a) C (b) C++
    (c) COBOL (d) FORTRAN
44. Which of the following languages is a symbolic language?
    (a) Machine language (b) C
    (c) Assembly language (d) All of these
45. Which of the following languages does not need any translator?
    (a) Machine language (b) 3GL
    (c) Assembly language (d) 4GL
46. Choose the odd one out from the following:
    (a) Compiler (b) Interpreter
    (c) Assembler (d) Linker
47. Windows Vista, Linux, and UNIX are examples of
    (a) operating systems (b) computer hardware
    (c) firmware (d) device drivers
48. Which among the following is an excellent analytical tool?
    (a) Microsoft Word (b) Microsoft Excel
    (c) Microsoft Access (d) Microsoft PowerPoint
49. Which interface makes use of the graphical components to allow users to easily interact with the computer system?

(a) CPU (b) CLI
(c) GUI (d) CUI

50. The language that is used to program the first stored-program computer systems is
(a) Machine language (b) Assembly language
(c) Pascal (d) Fortran
51. The register or location in main memory from where the data to be processed is located is specified by
(a) Label (b) Opcode
(c) Operand(s) (d) None of these
52. The generation to which COBOL belongs is:
(a) First generation (b) Second generation
(c) Third generation (d) Fourth generation
53. Of the following, a 5GL is
(a) Prolog (b) OPSS
(c) Mercury (d) LISP
54. The advantages of modularization are
(a) Reusability (b) Enhanced productivity
(c) Less time to develop (d) All of these
55. The code in 0s and 1s is
(a) Source code (b) Object code
(c) Executable code (d) None of these
56. The system software that creates the final executable file is
(a) Assembler (b) Compiler
(c) Loader (d) Linker
57. The type of high-level language that uses predicate logic is
(a) Unstructured (b) Procedure oriented
(c) Logic oriented (d) Object oriented
58. The high-level language that is used for numeric, scientific, statistical, and engineering computations is
(a) C (b) Basic
(c) Java (d) FORTRAN
59. The most portable language is
(a) C (b) Basic
(c) Java (d) FORTRAN
6o. Which among the following is an on-going activity in software development?
(a) Requirements analysis (b) Implementation
(c) User training (d) Maintenance
61. The functionality, capability, performance, availability of hardware and software components are all analysed in which phase?
(a) Requirements analysis (b) Design
(c) Implementation (d) Testing
62. In which phase are algorithms, flowcharts, and pseudocodes prepared?
(a) Requirements analysis (b) Design
(c) Implementation (d) Testing
63. Algorithms should be
(a) precise (b) unambiguous
(c) clear (d) all of these
64. To check whether a given number is even or odd, you will use which type of control structure?
(a) Sequence (b) Decision
(c) Repetition (d) All of these
65. Which one of the following is a graphical or symbolic representation of a process?
(a) Algorithm (b) Flowchart
(c) Pseudocode (d) Program
66. In a flowchart, which symbol is represented using a rectangle?
(a) Terminal (b) Decision
(c) Activity (d) Input/Output
67. Which of the following details are omitted in pseudocodes?
(a) Variable declaration (b) System specific code
(c) Subroutines (d) All of these
68. Trying to open a file that already exists, will result in which type of error?
(a) Run time (b) Compile time
(c) Linker error (d) Logical error
69. Which of the following errors is generated when rules of a programming language are violated?
(a) Syntax error (b) Semantic error
(c) Linker error (d) Logical error

State True or False

1. Computers work on the GIGO concept.
2. 1 nanosecond = 1×10^{-12} seconds.
3. Floppy disks and hard disks are examples of primary memory.
4. First-generation computers used a very large number of transistors.
5. First-generation computers could be programmed only in binary language.
6. Fifth-generation computers are based on AI.
7. Network computers have more processing power, memory, and storage than a desktop computer.

8. RAM stores the data and parts of program, the intermediate results of processing, and the recently generated results of jobs that are currently being worked on by the computer.
9. Computer hardware does all the physical work computers are known for.
10. The computer hardware cannot think and make decisions on its own.
11. The term software refers to a set of instructions arranged in a sequence to guide a computer to find a solution for the given problem.
12. BIOS defines the firmware interface.
13. Pascal cannot be used for writing well-structured programs.
14. Assembly language is a low-level programming language.
15. Microsoft DOS is a non-graphical command line operating system.
16. iOS is an open-source operating system.
17. C and Pascal can be used for writing well-structured and readable programs.
18. Code written in machine language is highly portable.
19. Home and End keys move the cursor to the previous and next page, respectively.
20. OCR is used to verify the legitimacy or originality of paper documents.
21. The monitor is a soft copy output device.
22. Non-impact printers create characters by striking an inked ribbon against the paper.
23. A laser printer uses the same technology used in photocopier machines.
24. A plotter is used to print vector graphics.
25. A mouse cannot be used with a laptop computer.
26. Soft copy output devices are those that produce a physical form of output.
27. Primary memory is faster than secondary memory.
28. Cache memory is made of DRAM.
29. Memory cards use flash memory to store data.
30. The ALU initiates action to execute the instructions.
31. The program counter stores the address of the next instruction to be executed.
32. The executing unit provides functions for data transfer.
33. A processor can perform all operations in a single clock tick.
34. A byte is a group of eight bits.
35. A computer can perform thousands of instructions in one second.
36. First generation of computers were used for commercial applications.
37. Knowledge is the application of data and information.
38. Computer and all its physical parts are known as software.
39. 1942–1955 marks the second generation of computers.
40. Machine/assembly language was used in first generation of computers.
41. SSI and MSI technology was used in fourth generation of computers.
42. Pascal, COBOL, FORTRAN, and BASIC are all low level programming languages.
43. Computer is a reliable machine.
44. If input data is wrong, then the output will also be erroneous.
45. When data and programs have to be used, they are copied from the primary memory into the secondary memory.
46. Robots cannot work in high temperature, high pressure conditions, or in processes which demand very high level of accuracy.
47. CPU can directly access primary memory.
48. Primary memory can be used for storing data permanently.
49. ALU manages and controls all the components of the computer system.
50. VDU is an input device.
51. Mouse can be used to create graphics such as lines, curves, and freehand shape on the screen.
52. Scanner is an input device.
53. OMR is used to verify the legitimacy or originality of paper documents.
54. Webcams are used for videoconferencing and as security cameras.
55. Searching data is faster in a soft copy output.
56. Dot-matrix and daisywheel printers are examples of impact printers.
57. Critical programs which are used to start the computer when it is turned on are stored in RAM.
58. Hard disk drive is an example of ROM.
59. A CD uses laser technology to read and write data on the disc.
60. CDs can store data on both sides of the disc.
61. Application software provides a general programming environment in which programmers can create specific applications to suit their needs.
62. Compiler and operating system is an example of application software.
63. MS Word and Paint are examples of application software.
64. Compiler translates one statement of high level language program into machine language and executes it.

65. Microsoft Excel is a word-processing software.
66. Labels are optional in assembly language.
67. Assembly language code is machine-dependant.
68. In monolithic paradigm, global data can be accessed and modified from any part of the program.
69. Monolithic programs are easy to debug and maintain.
70. Structured programming is based on modularization.
71. Object-oriented programming supports modularization.
72. Structured programming heavily uses goto statements.
73. Modules enhance the programmer's productivity.
74. A structured program takes more time to be written than other programs.
75. An algorithm solves a problem in a finite number of steps.
76. Flowcharts are drawn in the early stages of formulating computer solutions.
77. The main focus of pseudocodes is on the details of the language syntax.
78. In the deployment phase, all the modules are tested together to ensure that the overall system works well as a whole product.
79. Maintenance is an ongoing activity.
80. Algorithms are implemented using a programming language.
81. Logical errors are detected by the compiler.
82. Algorithm solves a problem in a finite number of steps.
83. Repetition means that each step of the algorithm is executed in a specified order.
84. Terminal symbol depicts the flow of control of the program.
85. Labelled connectors are square in shape.
86. Fourth-generation programming languages are non-procedural languages.
87. It takes less time to write a structured program than other programs.
88. Logic errors are much harder to locate and correct than syntax errors.
89. An interpreter translates the code and also executes it.

Review Questions

1. Define a computer.
2. Differentiate between data and information.
3. Differentiate between primary memory and secondary memory.
4. Write a short note on the characteristics of a computer.
5. Computers work on the garbage-in and garbage-out concept. Comment.
6. Explain the evolution of computers. Further, state how computers in one generation are better than their predecessors.
7. Broadly classify computers based on their speed, the amount of data that they can hold, and price.
8. Discuss the variants of microcomputers that are widely used today.
9. Explain the areas in which computers are being applied to carry out routine and highly-specialized tasks.
10. How does a keyboard work?
11. How is OCR technology better than an ordinary image scanner?
12. How does MICR technology help to detect fraud in cheque payments?
13. Web cameras can be used to check security in a bank. Comment.
14. How are projectors used to display information to a user?
15. How are headsets better than speakers and headphones?
16. Differentiate between impact and non-impact printers.
17. Why is a line printer preferred over a dot matrix printer? If you have an image to be printed, which out of the two will you use and why?
18. Under which situation, will you prefer to use an inkjet printer over a laser printer?
19. How is a plotter different from a printer?
20. What are input devices? Discuss the different types of input devices in detail.
21. Give a detailed note on different output devices.
22. Differentiate between a soft copy and a hard copy output.
23. What do you understand by computer memory?
24. Differentiate between primary memory and secondary memory.
25. Give the characteristics of the memory hierarchy chart.
26. Differentiate between static RAM and dynamic RAM.
27. Give the organization of computer memory. How does the CPU access a memory cell?
28. A DVD-ROM can store more data than a CD-ROM of the same size. Comment.
29. What is a USB flash drive?
30. Write a short note on memory cards.
31. Briefly discuss the importance of cache memory.
32. What do you understand by re-programmable ROM chips?
33. Draw and explain the basic architecture of a processor.
34. 'CPU is the brain of the computer.' Justify.

35. Broadly classify the computer system into two parts. In addition, make a comparison between a human body and the computer system, thereby explaining which part performs what function.
36. Differentiate between computer hardware and software.
37. Define programming.
38. What is booting?
39. What criteria are used to select the language in which a program will be written?
40. Explain the role of the operating system.
41. Why are compilers and interpreters used? Is there any difference between a compiler and an interpreter?
42. What is application software? Give examples.
43. Differentiate between syntax errors and logical errors.
44. Can a program written in a high-level language execute without a linker?
45. How is application software different from system software?
46. Write a short note on the different operating systems.
47. Define the term programming language. Give certain examples of such languages.
48. What is machine language? Do we still use it?
49. Code written in machine language is efficient and fast to execute. Comment.
50. How is a third generation programming language better than its predecessors?
51. 4GL code enhances the productivity of the programmers. Justify.
52. Define an algorithm. How is it useful in the context of software development?
53. Explain sequence, repetition, and decision statements. Also give the keywords used in each type of statement.
54. With the help of an example, explain the use of a flowchart.
55. How is a flowchart different from an algorithm? Do we need to have both of them for program development?
56. What do you understand by the term *pseudocode*?
57. Differentiate between algorithm and pseudocodes.
58. Define the term *programming language*. Give examples of such languages.
59. State the factors that a user should consider to choose a particular programming language.
60. What is machine language? Do we still use it?
61. Write a short note on assembly language.
62. What is an assembler?
63. Differentiate between an assembler and an interpreter.
64. A code written in machine language is efficient and fast to execute. Comment.
65. How is a third-generation programming language better than its predecessors?
66. A 4GL code enhances the productivity of the programmers. Justify.
67. Write a short note on structured programming.
68. What is modularization? Give its advantages.
69. How can you categorize high-level languages?
70. Differentiate between a procedural language and an object-oriented language.
71. Differentiate between a class and an object.
72. Explain the main features of an object-oriented programming language.
73. If given a program to write, how will you select the programming language to write the code?
74. What do you understand by the term 'programming paradigm'?
75. Briefly explain the phases in software development project.