

Based on National Curriculum of Pakistan 2022-23

Textbook of
Computer Science
Grade
10

National Curriculum Council
Ministry of Federal Education and Professional Training



National Book Foundation
as
Federal Textbook Board
Islamabad

Government Approval

Approved by the National Curriculum Council (NCC), Ministry of Federal Education and Professional Training, Islamabad
vide letter No. F.No.1-2/2024/NBF-CS, dated December 09, 2024

© 2025 National Book Foundation (NBF) as Federal Textbook Board

All rights to this publication are strictly reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the prior express written consent of the publisher. Unauthorized use, duplication, or distribution is strictly prohibited and may result in legal action.

A Textbook of Computer Science for Grade 10
based on National Curriculum of Pakistan (NCP) 2022-23

Authors

Dr. Saleem Iqbal (Managing Author)
Mr. Mohammad Khalid, Dr. Faraz Ahsan, Ms. Lubna Kousar,
Mr. Mohammad Sajjad, Mr. Shah Islam, Dr. Hamid Hassan (Co-Authors)

Contributors/Collaborators

Muhammad Zeeshan Ullah (Designer)

Supervision of Curriculum and Review Process

Dr. Shafquat Ali Janjua
Director, National Curriculum Council (NCC)
Ministry of Federal Education and Professional Training, Government of Pakistan, Islamabad

NCC Review Committee Members

Ms. Sadaf Zehra Kazmi, Ms. Rozina Faheem, Mr. Muhammad Sameer Toor

Desk Officer

Zehra Khushal, (Assistant Education Advisor), National Curriculum Council (NCC)

NBF Textbooks Development Supervision

Murad Ali Mohmand
Managing Director, National Book Foundation (NBF)

In-Charge, NBF Textbooks Development

Mansoor Ahmad, Assistant Director

Printed in Pakistan

First Edition: First Impression: February 2025 | Pages: 262 | Quantity: 100000

Price: PKR 360/-, Code: STE-723, ISBN: 978-969-37-1725-9

Printer: Al-Waqas printers, Peshawar

**TEST
EDITION**

For details on additional publications from the National Book Foundation, please visit our website at www.nbf.org.pk
You can also reach us by phone at 051 9261125 or via email at books@nbf.org.pk
For feedback or corrections, kindly send your comments to 'nbftextbooks@gmail.com' and 'textbooks@snc.gov.pk'

Note

All illustrations, artwork, and images in this book are intended solely for educational and promotional purposes, benefiting the public interest.

PREFACE

This Textbook has been developed by NBF according to the National Curriculum of Pakistan 2022- 2023. The aim of this textbook is to enhance learning abilities through inculcation of logical thinking in learners, and to develop higher order thinking processes by systematically building upon the foundation of learning from the previous grades. A key emphasis of the present textbook is on creating real life linkages of the concepts and methods introduced. This approach was devised with the intent of enabling students to solve daily life problems as they go up the learning curve and for them to fully grasp the conceptual basis that will be built upon in subsequent grades.

After amalgamation of the efforts of experts and experienced authors, this book was reviewed and finalized after extensive reviews by professional educationists. Efforts were made to make the contents student friendly and to develop the concepts in interesting ways.

The National Book Foundation is always striving for improvement in the quality of its books. The present book features an improved design, better illustration and interesting activities relating to real life to make it attractive for young learners. However, there is always room for improvement and the suggestions and feedback of students, teachers and the community are most welcome for further enriching the subsequent editions of this book.

May Allah guide and help us (Ameen).

Murad Ali Mohmand
Managing Director

UTILITY OR PRACTICAL APPLICATIONS OF THE SUBJECT

Computer science is one of the important subjects that have an impact on our daily lives. It facilitates problem-solving, communication and team work. Students who study computer science in grade 10 are able to develop talents such as creativity, logical reasoning and problem-solving approach. Additionally, they pull together useful skills that will help them succeed in this digital age. As a result they will be better able to grasp and contribute to the rapidly evolving world of technology.

Domain A: Computer Systems lays introduces how computers work. An introduction to operations of computer system is provided. The foundational knowledge needed to represent data in a computer along with understanding of information processing and storage is taught such that students may enhance their troubleshooting skills and understanding how system software links various components of a computer. On one hand this knowledge makes it easier for students to develop, test and debug code at the technological side while on the other hand makes collaboration and working in a team more effective by preparing students for software development in the real world.

Domain B: Computational Thinking & Algorithms educate students how to address issues in a methodical and logical way. It assists them in decomposing complex issues into manageable segments and formulating detailed solutions. Students who are taught typical tasks such as sorting and searching are better able to reason clearly and handle problems systematically. This kind of thinking is helpful not just in computer science but also in daily spheres of life like scheduling, task organization, etc.

In **Domain C: Programming Fundamentals** covers all the aspects required for students to be knowledgeable for developing simple webpages using HTML. Later, how to add styles and animations within the webpage using CSS are included. Finally by incorporating JavaScript the websites become more responsive and handles user interactions like forms, etc. Last but not the least, familiarization of data structures like arrays and list; inclusion of complex algorithm in a program and identifying and removing of errors in a program through testing are also part of this unit. This way, students can map the logical reasoning and execution of tasks in their daily lives and how to recognize their mistakes and rectify to address relevant issues.

Domain D: Data and Analysis are essential features that everyone should be able to handle and understand nowadays. Data resides around us in digital form like handheld devices and smartphones. However, to extract meaningful information that is necessary corresponding analytical skills need to be developed among students. Additionally, in affairs of daily life students at times need to gather their own data, analyze it by studying and applying formulation or with the help of graphical images like charts and graphs. This way, students can identify patterns and trends for better decision-making.

Domain E: Applications of Computer Science like Internet of Things (IoT) and Blockchain, broaden the understanding of technological use and trends among students. As technology continues to change the world, this understanding equips students to make practical and moral judgments. Additionally, it helps them to blend in with the rapid changing world. IoT allows commonplace objects to interact with each other and with the internet, therefore simplifying our lives. Blockchain is a safe transaction recording method that is frequently used for cryptocurrencies but also contributes in supply-chain businesses for the sake of fairness. Students get an understanding of functions and social implications of these technologies by studying them.

Domain F: Impacts of Computing underline the wide range of technological aspects on our life. The use of technology and internet aligns with our present life style and comes up with risks associated like breaching of personal information. This unit provides the policy guidelines and good practices that are to be followed while using modern gadgets and how one can secure himself online keeping up with the moral values associated with our society.

Finally, **Domain H: Entrepreneurship in the Digital Age** guides students to the path of 'learning and earning' through skills using digital devices. With problem solving aptitude and skills like web development, keeping in view the current applications of computers and their impacts around us; students can vigilantly come up with newer solutions and products. Thus, the need arises of creating and managing a firm as per industrial standards which can acquire clients' demands and provide services.

Table of Contents

Unit 1: Computer Systems

1.1	Machine Level Representation of data	8
1.2	Numbering Systems	9
1.3	Common Coding Schemes (ASCII and Unicode)	24
1.4	Operating System	26
1.5	Computer Software.....	36

Unit 2: Computational Thinking & Algorithms

2.1	Computing Problems	54
2.2	Basics of Counting Problems	55
2.3	Basic Counting Principles	56
2.4	Algorithms and their Characteristics	59
2.5	Logical Reasoning	60
2.6	Standard Algorithms.....	62
2.7	Steps in algorithm to solve computational problems	63
2.8	Abstraction	64
2.9	Creating Generalized Solutions	65
2.10	Modular Design	66
2.11	Algorithm Dry Run	67
2.12	Errors	69

Unit 3: Programming Fundamentals

3.1	Website Development.....	75
3.2	Additional Features of HTML/CSS.....	76
3.3	Advanced Programming Constructs in JavaScript	92
3.4	Complex Algorithms in JavaScript.....	102
3.5	Advanced techniques for testing and debugging	106

Unit 4: Data and Analysis

4.1	Data Science	117
4.2	Data Visualization	126
4.3	Stages of the Data Science Life Cycle	134

Unit 5: Applications of Computer Science

5.1	Internet of Things (IoT)	148
-----	--------------------------------	-----

5.2	Foundational Components of IoT	148
5.3	Applications of IoT	151
5.4	Blockchain Technology	156
5.5	Applications of Blockchain	156
5.6	Cloud Computing	158
5.7	Speech Recognition and Voice Recognition.....	160
5.8	Natural Language Processing	163
5.9	Robotics	165
5.10	Bias and Unethical use of Artificial Intelligence.....	168
5.11	Responsibilities of AI System Designers	169
Unit 6: Impacts of Computing		
6.1	What is Technology Addiction	175
6.2	Internet.....	177
6.3	The Effects of Threats to Individual Privacy and Security.....	180
6.4	Cloud Computing	181
6.5	Hardware and Software Method to Protect Devices.....	183
6.6	Cybersecurity	185
6.7	Safety and Security Concept	187
6.8	Cyberbullying	188
6.9	The Environmental, Cultural and Human Impact of Computing and Assistive Technologies.....	190
6.10	Digital Divide.....	191
6.11	Inequitable Access to Information.....	192
Unit 7: Digital Literacy		
7.1	Information Collection.....	200
Unit 8: Entrepreneurship in Digital Age		
8.1	Qualitative vs. Quantitative Research Differences, Examples & Methods	221
8.2	The differences between quantitative and qualitative research	222
8.3	Questionnaire Design Methods, Question Types & Examples	226
8.4	What is Market Validation?	231
8.5	How to Use Research to Create Customer Profiles	234
8.6	Business Pitching: Identifying Problems and Creating Solutions	238



At the end of this unit students will be able to:

- understand and describe number systems and encoding schemes for data representation in computer systems.
- explain how system software controls the flow of information between hardware components used for input, output, storage, and processing.
- identify and learn common software tools such as translators, integrated development environments, online and offline computing platforms, code repositories, etc.

Computer Systems

Introduction

A computer system is an essential part of modern life. It has changed how we work, communicate, learn, and have fun. A computer system isn't just one device; it's a mix of hardware (physical parts) and software (programs) that work together to process information, solve problems, and do many tasks.

Understanding computer systems is very important today, whether you're just using it for basic things or working as a professional in technology. Knowing how they work helps us use computers for many purposes, like business, science, creativity, and entertainment.

1.1 Machine Level Representation of data

"Machine level representation of data" refers to how information is encoded and stored in a computer's memory, which is understood by the machine's hardware. At this level, data is represented using binary digits (bits), which are either 0s or 1s.

Binary Representation:

At the machine level, all data is represented using binary digits. A binary digit, or a bit, can have one of two states: 0 or 1. These bits are the fundamental units of information in a computer. For example:

- » The decimal number 5 is represented in binary as 101.
- » The ASCII character 'A' is represented as 01000001 in binary.
- » A color in an image might be represented as a combination of binary values for red, green, and blue components.

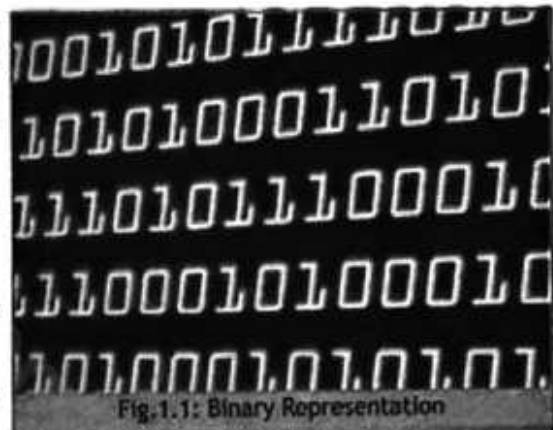


Fig.1.1: Binary Representation

Data Types:

Different types of data are represented in different ways at the machine level. Common data types include integers, floating-point numbers, characters, and Boolean values.

Integers: Integers are represented using a fixed number of bits, with the most significant bit (leftmost bit) indicating the sign (positive or negative). For example, an 8-bit integer can represent values from -128 to 127 (signed) or 0 to 255 (unsigned).

Floating-Point Numbers: A floating point number, is a positive or negative whole number with a decimal point. For example, 15.5, 0.45, and -203.345 are all floating point numbers.

Characters: Characters are represented using character encoding schemes such as ASCII or Unicode, where each character is assigned a unique binary code. For example, the ASCII code for the letter 'A' is 65 (01000001 in binary).

Boolean Values: Boolean values, representing true or false, can be represented using a single bit, where 0 typically represents false and 1 represents true.

1.2 Numbering Systems

Number System:

A number system is a way of writing and showing numbers. It uses specific digits or symbols to represent numbers. It helps us organize and understand numbers within a certain group. Number systems also make it easier to do basic math operations like addition, subtraction, multiplication, and division.

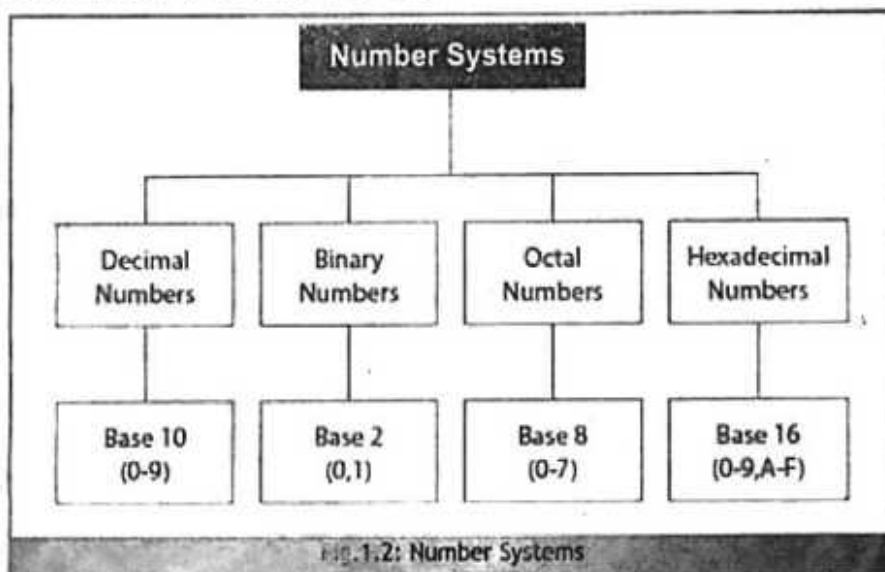
A Number:

A number is a mathematical value used for counting or measuring or labelling objects. Numbers are used to perform arithmetic calculations. Examples include: 20, 45, -10, 3.4, 11.5, -75.6, etc.

1.2.1 Types of Number Systems

There are various types of number systems in mathematics. The four most common number system types are:

- »»» Decimal number system (Base- 10)
- »»» Binary number system (Base- 2)
- »»» Octal number system (Base-8)
- »»» Hexadecimal number system (Base- 16)



Decimal Number System

The decimal (also called Denary) number system is composed of the 10 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. Using these symbols we can express any quantity. The decimal number system is called base 10 system because it has 10 digits. The decimal number system is a positional-value system, in which the value of a digit depends on its position in the number. For example, consider the decimal number 964. Here:

9 represents 9 hundreds

6 represents 6 tens

4 represents 4 units

The 9 carries the most weight. It is called the Most Significant Digit (MSD). The 4 carries the least weight and is called the Least Significant Digit (LSD). Each digit of a number carries weight that can be expressed as powers of 10. This is shown below, where the number 2745.214 is represented. The decimal point separates the positive powers of 10 from the negative powers.

$$\begin{aligned}(2745.214)_{10} &= 2 \times 10^3 + 7 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 + 2 \times 10^{-1} + 1 \times 10^{-2} + 4 \times 10^{-3} \\ &= 2000 + 700 + 40 + 5 + 0.2 + 0.01 + 0.004 \\ &= 2745.214\end{aligned}$$

In general, any number is simply the sum of the products of each digit value times its positional value.

Binary Number System

In binary number system there are only two symbols or digits, 0 and 1. The base of binary number system is 2. All the statements made earlier about the decimal number system are equally applicable to the binary number system as well. The binary system is also a positional-value system, wherein each bit has its own value or weight expressed as power of 2.

$$\begin{aligned}(1011.101)_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125 \\ &= (11.625)_{10}\end{aligned}$$

The binary number $(1011.101)_2$ is represented above and its equivalent decimal value can be found by taking the sum of the products of each bit value (0 or 1) times its positional value. The left most bit carries the most weight and it is called the Most Significant Bit (MSB). The positions to the right of the binary point are negative powers of 2. The right most bit carries the least weight and is referred to as the least significant bit (LSB).

Octal Number System

The Octal number system has a base of 8, meaning it has eight possible digits; 0, 1, 2, 3, 4, 5, 6, and 7. Thus, each digit of an octal number system can have any value from 0 to 7. The digit positions in an octal number system have weight with powers of 8. An octal number can be easily converted to its decimal equivalent by multiplying each digit by its positional weight.

For example,

$$(672)_8 = 6 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 = 384 + 56 + 2 = (442)_{10}$$

Another example:

$$(25.6)_8 = 2 \times 8^1 + 5 \times 8^0 + 6 \times 8^{-1} = 16 + 5 + 0.75 = (21.75)_{10}$$

Hexadecimal Number System

The hexadecimal number system uses base 16. Thus it has 16 possible digits. It uses the digits 0 - 9 and letters A, B, C, D, E and F as the 16 digit symbols. The table shows the relationship among hexadecimal and decimal numbers. It is important to remember that hexadecimal digits A through F are equivalent to the decimal values 10 through 15.

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

A hexadecimal number can be converted to its decimal equivalent by using the fact that each digit position has a weight that is power of 16. The LSD has a weight of $16^0 = 1$ the next higher digit has a weight of $16^1 = 16$, the next higher digit has a weight of $16^2 = 256$, and so on.

The conversion process is shown below.

$$\begin{aligned}(3AF)_{16} &= 3 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 = 768 + 160 + 15 \\ &= (943)_{10}\end{aligned}$$

1.2.2 Conversions from one Number system to the other

Conversion from any Base to Base-10(Decimal)

To convert a number from any base to base-10 (decimal), you need to convert each digit of the number to its decimal equivalent. Then, you use base-10 arithmetic to expand the number by multiplying each digit by the base raised to the power of its position (starting from 0 on the right). Finally, add the results together to get the decimal value.

i. Conversion from Binary to Decimal

Example1: $(10111)_2$ to (?)₁₀

$$\begin{aligned}(10111)_2 &= (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) \\ &= 16 + 0 + 4 + 2 + 1 \\ &= (23)_{10}\end{aligned}$$

Example2: $(100101.1011)_2$ to (?)₁₀

$$\begin{aligned}(100101.1011)_2 &= (1 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) \\ &\quad (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) + (1 \times 2^{-4}) \\ &= 32 + 0 + 0 + 4 + 0 + 1 + (1/2) + (0) + (1/8) + (1/16) \\ &= 37 + (11/16) \\ &= 37 + 0.6875 \\ &= (37.6875)_{10}\end{aligned}$$

ii. Conversion from Octal to Decimal

Example: $(437)_8$ to $(?)_{10}$

$$\begin{aligned}(437)_8 &= (4 \times 8^2) + (3 \times 8^1) + (7 \times 8^0) \\ &= (4 \times 64) + (3 \times 8) + (7 \times 1) \\ &= 256 + 24 + 7 \\ &= (287)_{10}\end{aligned}$$

iii. Conversion from Hexadecimal to Decimal

Example: $(1BE8)_{16}$ to $(?)_{10}$

$$\begin{aligned}(1BE8)_{16} &= (1 \times 16^3) + (B \times 16^2) + (E \times 16^1) + (8 \times 16^0) \\ &= (1 \times 16^3) + (11 \times 16^2) + (14 \times 16^1) + (8 \times 16^0) \\ &= (1 \times 4096) + (11 \times 256) + (14 \times 16) + (8 \times 1) \\ &= 4096 + 2816 + 224 + 8 \\ &= (7144)_{10}\end{aligned}$$



Activity-1

a. Convert the following Binary numbers to Decimal

- i. 11100011 ii. 101010101 iii. 11001.1001 iv. 10001.111

b. Convert the following Octal numbers to Decimal

- i. 3452 ii. 1256 iii. 7454 iv. 2743

c. Convert the following Hexadecimal numbers to Decimal

- i. AB01F ii. FE162B iii. 9C17D iv. 8D00A

Conversion from Base-10(Decimal) to any other Base

To convert a decimal number to any other number:

- Divide the decimal number to be converted by the value of the new base and record the remainder as the LSD of the new base number.
- Divide the quotient of the previous division by the new base again and record the remainder as the next digit to the left of the new base number.
- Repeat this process, until the quotient becomes zero or not divisible by the base.
- Note that the last remainder obtained will be the MSD of the new base number.

The following examples will demonstrate the above method by converting the decimal numbers to their binary, octal and hexadecimal equivalents, respectively.

i. From Decimal to Binary

Converting the Integer Part:

- Divide the integer part by 2.
- Record the remainder (0 or 1).
- Update the integer part to the quotient.
- Repeat steps 1-3 until the integer part is 0.
- The binary representation is the remainders read in reverse order.

Converting the Fractional Part:

- Multiply the decimal part by 2.
- Record the integer part (0 or 1). This becomes the next binary digit.
- Remove the integer part from the product, leaving only the decimal part.
- Repeat the process with the new decimal part.
- Continue until the decimal part becomes 0 or until you have reached the desired precision.
- The binary representation is the sequence of recorded integers.

Example1: $(179.257)_{10}$ to $(\quad?)_2$

Step1: Converting integer part 179 to binary			Step2: Converting fractional part 0.257 to binary	
2	179	Remainder		Binary
2	89	1	$0.257 \times 2 =$	0.514
2	44	1	$0.514 \times 2 =$	1.028
2	22	0	$0.028 \times 2 =$	0.056
2	11	0	$0.056 \times 2 =$	0.112
2	5	1	$0.112 \times 2 =$	0.224
2	2	1	$0.224 \times 2 =$	0.448
1	1	0	$0.448 \times 2 =$	0.896
			$0.896 \times 2 =$	1.792

$(179)_{10} = (10110011)_2$

$(0.257)_{10} = (0.0100001)_2$

So, $(179.257)_{10} = (10110011.0100001)_2$

Example1: $(179.257)_{10}$ to $(\quad?)_2$

Step1: Converting integer part 179 to binary			Step2: Converting fractional part 0.257 to binary	
2	179	Remainder		Binary
2	89	1	$0.257 \times 2 =$	0.514
2	44	1	$0.514 \times 2 =$	1.028
2	22	0	$0.028 \times 2 =$	0.056
2	11	0	$0.056 \times 2 =$	0.112
2	5	1	$0.112 \times 2 =$	0.224
2	2	1	$0.224 \times 2 =$	0.448
1	1	0	$0.448 \times 2 =$	0.896
			$0.896 \times 2 =$	1.792

$(179)_{10} = (10110011)_2$

$(0.257)_{10} = (0.0100001)_2$

So, $(179.257)_{10} = (10110011.0100001)_2$

ii. From Decimal to Octal

Example 1: $(165)_{10}$ to $(?)_8$

8	165	Remainder
8	20	5
8	2	4

$$(165)_{10} = (245)_8$$

Example 2: $(331)_{10}$ to $(?)_8$

8	331	Remainder
8	41	3
8	5	1

$$(331)_{10} = (513)_8$$

iii. From Decimal to Hexadecimal

Example 1: $(739)_{10}$ to $(?)_{16}$

16	739	Remainder
16	46	3
16	2	14

2 14 3
2 E 3

$$(739)_{10} = (2E3)_{16}$$

Example 2: $(3501)_{10}$ to $(?)_{16}$

16	3501	Remainder
16	218	13
16	13	10

13 10 13
D A D

$$(3501)_{10} = (DAD)_{16}$$



Activity-2

a. Convert the following Decimal numbers to Binary

- i. 786.345 ii. 943.095 iii. 6300.586 iv. 4732.7834

b. Convert the following Decimal numbers to Octal

- i. 759 ii. 1256 iii. 9454 iv. 2743

c. Convert the following Decimal to Hexadecimal numbers

- i. 7802 ii. 4923 iii. 3678 iv. 6401

Conversion from Octal to Binary and vice versa

The conversion from Octal to Binary and vice versa is performed by converting each octal digit to its 3-bit binary equivalent. The eight possible digits are converted as follows:

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

Using these conversions, any octal number can be converted to binary and vice versa.

i. From Octal to Binary

Example: $(7301)_8$ to $(\quad?)_2$

7	3	0	1
↓	↓	↓	↓
111	011	000	001

$(7301)_8 = (111011000001)_2$

ii. From Binary to Octal

Example: $(1100111000101)_2$ to $(\quad?)_8$

- Take group of 3 binaries from the right side.

1100111000101
 $\swarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \searrow$
 1 100 111 000 101

- If any one or two bits are left alone, add zeros to the left to make them group of 3.

001 100 111 000 101

- Convert each binary group to its equivalent Octal digit.

001	100	111	000	101
↓	↓	↓	↓	↓
1	4	7	0	5

$(1100111000101)_2 = (14705)_8$



Activity-3

a. Convert the following Octal numbers to Binary

i. 765

ii. 1430

iii. 63005

iv. 4232

b. Convert the following Binary numbers to Octal

i. 10011110

ii. 11000010110

iii. 1000001101

iv. 1110001001101010

Conversion from Hexadecimal to Binary and vice versa

The conversion from Hexadecimal to Binary and vice versa is performed by converting each octal digit to its 4-bit binary equivalent. The eight possible digits are converted as follows:

Hexadecimal	Binary
0	0000
1	0001
2	0010

3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A (10)	1010
B (11)	1011
C (12)	1100
D (13)	1101
E (14)	1110
F (15)	1111

Using these conversions, any Hexadecimal number can be converted to binary and vice versa.

i. From Hexadecimal to Binary

Example: $(D73A1)_{16}$ to (?)₂

D	7	3	A	1
↓	↓	↓	↓	↓
13	7	3	10	1
1101	0111	0011	1010	0001

$(D73A1)_{16}$ to $(1101011100110100001)_2$

ii. From Binary to Hexadecimal

Example: $(110011100100110101)_2$ to (?)₁₆

- Take group of 4 binaries from the right side.

110011100100110101
 $\swarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \searrow$
11 0011 1001 0011 0101

- If any one or two or three bits are left alone, add zeros to the left to make them group of 4.

0011 0011 1001 0011 0101

- Convert each binary group to its equivalent Hexadecimal digit.

0011	0011	1001	0011	0101
↓	↓	↓	↓	↓
3	3	9	3	5
3	3	9	3	5

$(110011100100110101)_2$ to $(33935)_{16}$



Activity-4

- a. Convert the following Hexadecimal numbers to Binary
 i. AF765 ii. D14C0 iii. B300E iv. C42F2
- b. Convert the following Binary numbers to Hexadecimal
 i. 1001100001000110 ii. 11011100010110 iii. 10001100001101
 iv. 111000100000110101110

Conversion from Octal to Hexadecimal and vice versa

i. From Octal to Hexadecimal

Method 1:

- Convert Octal to Decimal number
- Then convert the decimal number to Hexadecimal

Method 2:

- Convert Octal to Binary number
- Then convert the binary number to Hexadecimal

Example: $(6457)_8$ to (?)₁₆

Method 1: $(6457)_8 = (3375)_{10}$ { Method is already Explained }

$(3375)_{10} = (D2F)_{16}$ { Method is already Explained }

So, $(6457)_8$ to $(D2F)_{16}$

Method 2: $(6457)_8 = (110100101111)_2$ { Method is already Explained }

$(110100101111)_2 = (D2F)_{16}$ { Method is already Explained }

So, $(6457)_8$ to $(D2F)_{16}$

ii. From Hexadecimal to Octal

Method 1:

- Convert Hexadecimal to Decimal number
- Then convert the decimal number to Octal

Method 2:

- Convert Hexadecimal to Binary number
- Then convert the binary number to Octal

Example: $(A09D)_{16}$ to (?)₈

Method 1: $(A09D)_{16} = (41117)_{10}$ { Method is already Explained }

$(41117)_{10} = (120235)_8$ { Method is already Explained }

So, $(A09D)_{16}$ to $(120235)_8$

Method 2: $(A09D)_{16} = (1010000010011101)_2$ { Method is already Explained }

$(1010000010011101)_2 = (120235)_8$ { Method is already Explained }

So, $(A09D)_{16}$ to $(120235)_8$



Activity-5

- a. Convert the following Hexadecimal numbers to Octal
 i. AF65 ii. D1C0 iii. B30E iv. C2F2
- b. Convert the following Octal numbers to Hexadecimal
 i. 765 ii. 1430 iii. 63005 iv. 4232

1.2.3 Binary Arithmetic

Computers use binary arithmetic as their fundamental way of performing arithmetic operations because digital electronic circuits inherently operate in binary (base-2) rather than in decimal (base-10) like humans do. In binary arithmetic, there are only two digits: 0 and 1.

Basic Arithmetic Operations: Computers perform addition, subtraction, multiplication, and division in binary just like in decimal, but with binary digits (bits) instead of decimal digits.

Binary Addition:

➤ Binary addition is performed much like decimal addition, but with only two possible digits: 0 and 1.

➤ When adding two binary digits (bits), the possible outcomes are:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10 \text{ (carry 1 to the next higher bit)}$$

Some Examples of Binary Addition:

Example1: 10010 + 111	Example2: 11101010 + 10011011
$ \begin{array}{r} \textcircled{1}\textcircled{1} \text{ (Carry)} \\ 10010 \longrightarrow 18 \text{ (Decimal)} \\ + \quad 111 \longrightarrow + 7 \text{ (Decimal)} \\ \hline 11001 \longrightarrow 25 \end{array} $	$ \begin{array}{r} \textcircled{1}\textcircled{1}\textcircled{1}\textcircled{1} \text{ (Carry)} \\ 11101010 \\ + 10011011 \\ \hline 110000101 \end{array} $

HINT!

To verify your answer you can convert the binary number to decimal and perform decimal arithmetic, like in Example 1 (Addition).

Binary Subtraction:

➤ Binary subtraction is performed much like decimal subtraction, but with only two possible digits: 0 and 1.

➤ Borrowing is used when subtracting a larger number from a smaller one.

➤ When subtracting two binary digits (bits), the possible outcomes are:

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 = 1 \text{ with borrowing (borrow 1 from the next higher bit)}$$

Some Examples of Binary Subtraction:

Example 1: $10110 - 101$	Example 2: $11000101 - 101111$
$ \begin{array}{r} 10110 \\ - 101 \\ \hline 10001 \end{array} $ (Borrow) $\rightarrow (10)_2 = 2$	$ \begin{array}{r} 11000101 \\ - 101111 \\ \hline 10010110 \end{array} $



Activity-6

a. Perform the following Binary Additions

i. $11100001 + 10110001$

ii. $10100111 + 11110101$

iii. $11111101 + 10110111$

iv. $10101011 + 10110001$

b. Perform the following Binary Subtractions

i. $11100001 - 1011001$

ii. $10100111 - 1000101$

iii. $11111101 - 101101$

iv. $10101011 - 10111$

1.2.4 Overflow and Underflow

In computers, overflow and underflow are problems that occur when calculations produce results that are too large or too small to be represented within the allowed number of bits. These conditions are important because they can cause errors in programs and lead to unexpected behavior or bugs. Overflow happens when a value is too large to fit in the available space, while underflow occurs when a value is too small to be represented properly. Understanding these issues is important to prevent problems in software. Overflow is being explained as follows at that level.

Overflow: Overflow happens when a calculation produces a result that is larger than the maximum value that can be stored or represented with the available number of digits or bits in a system. This causes the system to be unable to properly show or handle the result.

Example: 8-bit Unsigned Integer Overflow

An 8-bit unsigned integer can represent values from 0 to 255. If user attempts to add 1 to 255, the result is 256, which cannot be represented with just 8 bits.

$$\begin{array}{r}
 11111111 \text{ (255 in binary)} \\
 + 1 \\
 \hline
 \text{Overflow} \rightarrow 100000000 \\
 100000000 \text{ (256 in binary)}
 \end{array}$$

Since an 8-bit register can only store 8 bits, the ninth bit (1) is the overflow bit, which will be lost and the overall result becomes wrong. This situation can be overcome by introducing 16-bit register.

Handling Overflow and Underflow:

To handle overflow and underflow, programmers can:

- **Use Larger Data Types:** Switching to a larger data type (e.g., from an 8-bit integer to a 16-bit integer) can help avoid overflow and underflow for larger ranges of values.
- **Perform Range Checking:** Before performing an arithmetic operation, check if the operation will result in a value outside the representable range and handle it appropriately.
- **Use Libraries or Built-in Functions:** Some programming languages provide built-in functions or libraries that can detect and handle overflow and underflow conditions.

1.2.5 Compliments (1's and 2's)

Complements:

In digital logic, 'Complements' are used to perform logical operations, especially the subtraction operation. The binary number system contains two types of complements, i.e., 1's complement and 2's complement.

1's Complements

The binary numbers can be easily converted into the 1's complement with the help of a simple technique. According to this algorithm, if we toggle or invert all bits of a binary number, the generated binary number will become the 1's complement of that binary number. That means we have to transform 1 bit into the 0 bit and 0 bit into the 1 bit in the 1's complement.

Example 1: 1's complement of binary number 01101

0 1 1 0 1	(Number)
1 0 0 1 0	(1's Complement)

Example 2: 1's complement of binary number 00011101

0 0 0 1 1 1 0 1	(Number)
1 1 1 0 0 0 1 0	(1's Complement)

2's Complements

The binary numbers can be converted into the 2's complement with the help of a very simple technique.

- Take the 1's complement of the binary number
- Add 1 to the LSB (least significant bit)

Example 1: 2's complement of binary number 01010001

0 1 0 1 0 0 0 1	(Number)
1 0 1 0 1 1 1 0	(1's Complement)
+ 1	(Adding 1)
1 0 1 0 1 1 1 1	(2's Complement)

So the 2's Complement of 01010001 is 10101111

Example 2: 2's complement of binary number 10101110

1 0 1 0 1 1 1 0	(Number)
0 1 0 1 0 0 0 1	(1's Complement)
+ 1	(Adding 1)
0 1 0 1 0 0 1 0	(2's Complement)

So the 2's Complement of 10101110 is 01010010



Activity-7

a. Convert the following Binary numbers to their 1's Complement forms.

- i. 00111001 ii. 11001100 iii. 11100011 iv. 01010101

b. Convert the following Binary numbers to their 2's Complement forms.

- i. 01111001 ii. 10001101 iii. 11101011 iv. 01011101

Binary Subtraction using 1's Complement

The following steps are used to perform binary subtraction using 1's Complement.

- Make the number of bits in both minuend and subtrahend the same by adding additional zeros(0s) to left side.
- Convert the subtrahend to the 1's complement.
- Add the 1's complement of the subtrahend with the minuend.
- If the result has a carry, then add that carry in the least significant bit.
- If there is no carry, then take the 1's complement of the resultant to find the answer. In this case the answer will be negative(-ve).

HINT!

You can verify your answer by doing simple subtraction.

Example1: Subtract 100110 from 111000 using 1's complement

- Number of bits are the same.

$$\begin{array}{r} 111000 \text{ (Minuend)} \\ 100110 \text{ (Subtrahend)} \end{array}$$
- Convert the subtrahend to the 1's complement.

$$011001$$
- Add the 1's complement of the subtrahend with the minuend.

$$\begin{array}{r} 111000 \\ + 011001 \\ \hline 1 - 010001 \end{array}$$
- Carry is generated, add it to the LSB of the result

$$\begin{array}{r} 010001 \\ + 1 \\ \hline 010010 \end{array}$$
- So the final answer is 010010

Example2: Subtract 10101001 from 11100011 using 1's complement

- $$\begin{array}{r} 11100011 \text{ (Minuend)} \\ 10101001 \text{ (Subtrahend)} \end{array}$$
- Convert the subtrahend to the 1's complement.

$$01011110$$
- Add the 1's complement of the subtrahend with the minuend.

$$\begin{array}{r} 11100011 \\ + 01011110 \\ \hline 1 - 00111001 \end{array}$$
- Carry is generated, add it to the LSB of the result

$$\begin{array}{r} 00111001 \\ + 1 \\ \hline 00111010 \end{array}$$
- So the final answer is 00111010

Binary Subtraction using 2's Complement

The following steps are used to perform binary subtraction using 2's Complement.

- Make the number of bits in both minuend and subtrahend the same by adding additional zeros(0s) to left side.
- Convert the subtrahend to the 2's complement.
- Add the 2's complement of the subtrahend with the minuend.
- If the result has a carry, ignore it and take the remaining part as answer.
- If there is no carry, then take the 2's complement of the resultant to find the answer. In this case the answer will be negative (-ve).

Example: Subtract 100110 from 111000 using 2's Complement

- Number of bits are the same.
111000 (Minuend)
100110 (Subtrahend)
- Convert the subtrahend to the 2's complement.
011001
+ 1

011010
- Add the 2's complement of the subtrahend with the minuend.
111000
+ 011010

1 - 010010
- Ignore the Carry and take 010010 as the answer
- So the final answer is 010010



Activity-8

- a. Convert the following Binary numbers to their 1's Complement forms.
i. 00111001 ii. 11001100 iii. 11100011 iv. 01010101
- b. Convert the following Binary numbers to their 2's Complement forms.
i. 01111001 ii. 10001101 iii. 11101011 iv. 01011101

PRACTICE TIME!

Subtract 10101001 from
11100011 using 2's
complement

1.2.6 Signed and Unsigned Numbers Representation

Binary numbers can be represented in signed and unsigned way. Unsigned binary numbers do not have sign bit, whereas signed binary numbers uses signed bit as well or these can be distinguishable between positive and negative numbers.

Unsigned Numbers Representation:

Unsigned numbers don't have any sign, these can contain only magnitude of the number. So, representation of unsigned binary numbers are all positive numbers only. For example, representation of positive decimal numbers are positive by default. We always assume that there is a positive sign symbol in front of every number. The range of unsigned binary number is from 0 to $(2^n - 1)$, where n is the number of bits. For example in 8-bits form the range will be from 0 to 255 (decimal) and 00000000 to 11111111 (binary)

Example: Represent decimal number 105 in unsigned binary number.

Simply convert 105 into Binary number.

$$105 = (01101001)_2$$

Signed Numbers Representation:

Signed numbers can represent both positive and negative integers. There are several methods to

represent signed numbers, but the most common one used in modern computers is Two's Complement.

Two's Complement Representation:

To get the 2's complement of a number, flip all the bits and then add 1 to the least significant bit (LSB). We use regular binary for positive numbers and 2's complement for negative numbers.

If the sign bit is 0, the number is positive and we can just use its regular binary form.

If the sign bit is 1, the number is negative, and we need to use the 2's complement of the binary number to get its value.

Example1: Represent -13 in a computer using 8-bit 2's complement.

- Convert 13 to binary: 13 in 8-bit binary form is 00001101
- Invert all the bits: 11110010
- Add 1 to get 2's complement: $11110010 + 1 = 11110011$
- So, -13 is represented as 11110011 using 2's Complement

Example2: Represent -55 in a computer using 8-bit 2's complement.

- Convert 55 to binary: 55 in 8-bit binary form is 00110111
- Invert all the bits: 11001000
- Add 1 to get 2's complement: $11001000 + 1 = 11001001$
- So, -55 is represented as 11001001 using 2's Complement



Activity-9

Represent the following numbers using 8-bit 2's complement.

i. -90

ii. -122

iii. -150

iv. -200

v. -240

Floating-point Numbers Representation:

Floating point numbers are a way to represent real numbers that can have fractional parts. They are used in computers to handle very large and very small numbers efficiently, as well as to perform arithmetic operations on them.

Representation of Floating Point Numbers:

Floating point numbers are represented in a computer using a standard format defined by the IEEE (Institute of Electrical and Electronics Engineers). The most common format is IEEE 754, which specifies how floating point numbers should be stored in binary.

Components of a Floating Point Number

A floating point number in the IEEE 754 format is divided into three parts:

Sign Bit: Indicates whether the number is positive or negative.

Exponent: Represents the power of 2 by which the fraction (or mantissa) is multiplied.

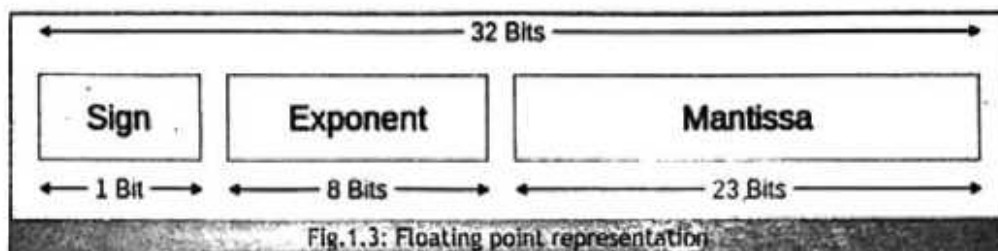
Mantissa (or Significand): Represents the significant digits of the number.

For example, in a 32-bit (single precision) floating point representation is shown in Figure.

1 bit for the sign

8 bits for the exponent

23 bits for the mantissa



Note: Further details about how floating point numbers are represented are beyond the scope of grade 10 students.

1.3 Common Coding Schemes (ASCII and Unicode)

Coding schemes are standards that computers use to convert and represent data in a format they can understand, such as binary digits (1's and 0's). Since computers cannot interpret human-readable characters like letters, numbers, and symbols directly, they need to translate this data into a machine-readable format. Different coding schemes are used for this purpose, among them ASCII and Unicode are common.

Why Coding Schemes are used

Data Representation: Computers use binary (0s and 1s) to represent all data. Coding schemes map human-readable characters to binary codes, making it possible for computers to store and manipulate text and other data.

Standardization: Coding schemes provide a consistent way to represent characters across different platforms and devices, ensuring compatibility and accurate data exchange.

Efficiency: They optimize the storage and processing of data by providing a fixed-length binary representation for each character.

Communication: They enable different computer systems to communicate with each other by using a common representation for characters and symbols.

ASCII Coding Scheme

ASCII stands for American Standard Code for Information Interchange. It is a standard data coding scheme in computers. It assigns standard numeric values to alphabets, numerals, punctuation marks and other characters used in computers. Before ASCII was developed, different makes and models of computers could not communicate with one another. Each computer manufacturer represented alphabets, numerals and other characters in its own way.

On June 17, 1963 ASCII was approved as the American standard code. However, it did not gain wide acceptance because IBM chose to use EBCDIC (Extended binary coded decimal interchange

code) in its computers. Initially ASCII used 7-bit codes of various combinations of 0's and 1's to represent 128 different characters ($2^7 = 128$). ASCII-7 could not represent newly developed characters so it underwent further developments and revisions and ASCII-8 was developed.

Extended ASCII Code (ASCII-8 bit Code):

ASCII was extended to an 8-bit code that can represent ($2^8 = 256$) different characters. ASCII-8 became popular in 1981 when it was used first time in IBM's Personal Computer (PC) and soon it became industry standard for personal computers. Many Windows systems use an 8-bit ASCII encoding and this Microsoft specific encoding is known as ANSI code. ANSI stands for American National Standards Institute. In extended ASCII-8, 32 code combinations are used for machine and control commands, such as "start of text," "carriage return," and "form feed", etc. Control commands do not represent

printable information, but rather they help control devices, such as printers. Table 1 (Annexure 1) shows non-printable ASCII control codes. Table 2 (Annexure 2) shows printable ASCII characters and Table 3 (Annexure 3) shows extended ASCII printable characters.

Just to Remember!	
Characters	ASCII-8 Value in Decimal
0-9	48-57
A-Z	65-90
a-z	97-122

Unicode Coding Scheme

Even extended ASCII (ASCII-8) does not include enough code combinations to support all written languages. Asian languages, for instance, require thousands of characters. This limitation gave rise to new encoding standard known as **Unicode (Universal coding system)** that can support all the written languages. It was published by Unicode Consortium (a group of multilingual software manufacturers). The first version of Unicode was introduced in 1991.

Unicode is an international character-encoding system designed to facilitate the electronic exchange, processing, and display of written texts from diverse modern and classical languages. The Unicode Standard encompasses letters, digits, accents, punctuation marks, technical symbols for the world's major written languages, as well as emoji and other symbols, all using a uniform encoding scheme. The latest version of Unicode includes over 100,000 characters. Unicode assigns a unique number to each character that remains consistent across all systems that support Unicode.

Unicode defines multiple encodings of its single character set: UTF-7, UTF-8, UTF-16, and UTF-32. UTF stands for Unicode Transformation Format. Conversion of data among these encodings is lossless, means do not lose the quality. Unicode was originally a 2-byte (16-bits) character set. Unicode version 3, however, is a 4-byte (32-bits) code and is fully compatible with ASCII. These all support encoding the same set of characters. Table 4 (Annexure 4) shows a part of Unicode character set after ASCII-8.

1.4 Operating System

An operating system (OS) is essential software that manages a computer's memory, processes, hardware, and software, enabling user interaction. It loads into RAM when the computer is turned on and is necessary to run any programs. OS has evolved since the first computer generation and is present in all devices like desktops, tablets, and smartphones, with common examples being Windows, Mac OS, Linux, iOS, and Android.

1.4.1 Main Tasks/Functions of an Operating System

Operating system performs the following main tasks or functions.

- Process Management
- Memory Management
- File Management
- Device Management
- Network Management
- Security management

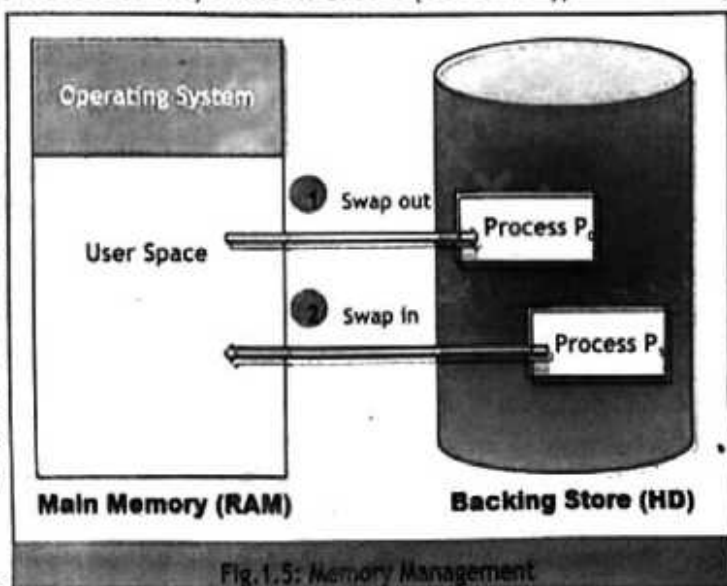
Process Management

A process is a program in execution, and process management in an operating system allocates resources like CPU time to various processes in memory. For example, with three processes (A, B, and C), each having different CPU cycles ($A = 5$ ms, $B = 2$ ms, $C = 1$ ms), the OS manages execution times. In Case 1 (ABC order), the total time is 6.67 ms, while in Case 2 (BCA order), it's 4.33 ms, showing that Case 2 is more efficient.

Memory Management

Memory management is the part of operating system that controls and manages the operation of main memory during the operation of computer. It allocates space to programs that are loaded in main memory for execution. It keeps track of freed memory when a program is closed and updates the memory status.

Example: In this example the OS is



managing memory for two processes P0 and P1. P1 is being loaded (swap in) and P0 is being taken out (swap out) from the main memory (RAM). The whole process is shown in Fig 1.5.

File Management

File management is the part of operating system that manages files and folders on storage devices such as hard disk, USB flash drive and DVD. It allows computer user to perform operations such as creating, copying, moving, renaming, deleting, and searching files and folders. It also allows the user to perform read, write, open and close operations on files and folders. Fig 1.6 shows the management of files in various folders by OS.

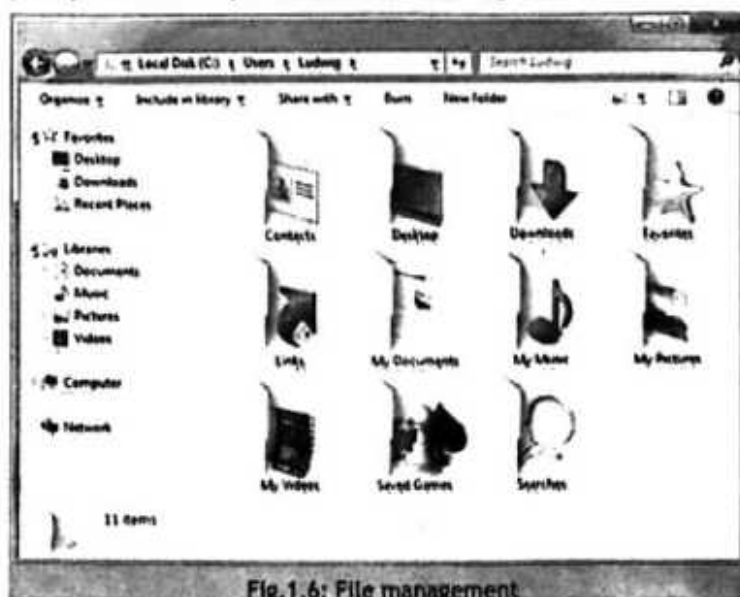


Fig.1.6: File management

Device Management

Device management is the part of operating system that controls input and output devices, such as keyboards, mouse, printers, and network interfaces. Efficient I/O management improves the performance of the computer. As shown in Fig. 1.7.

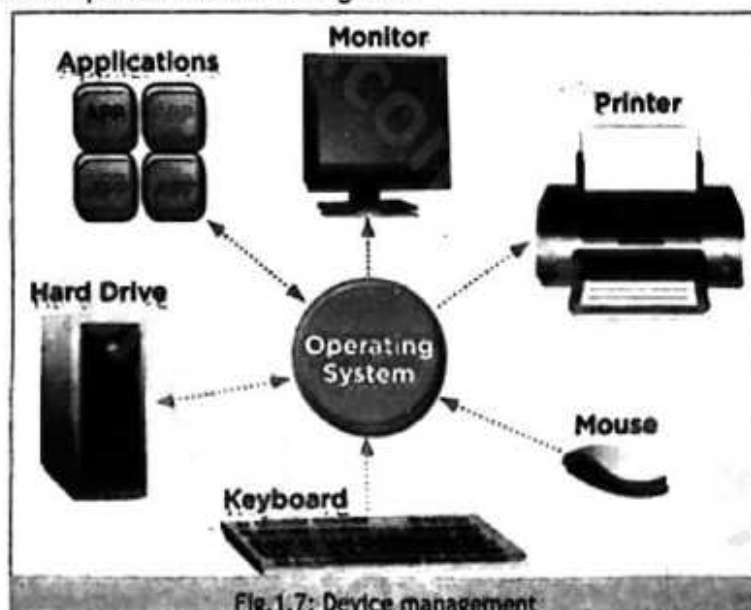
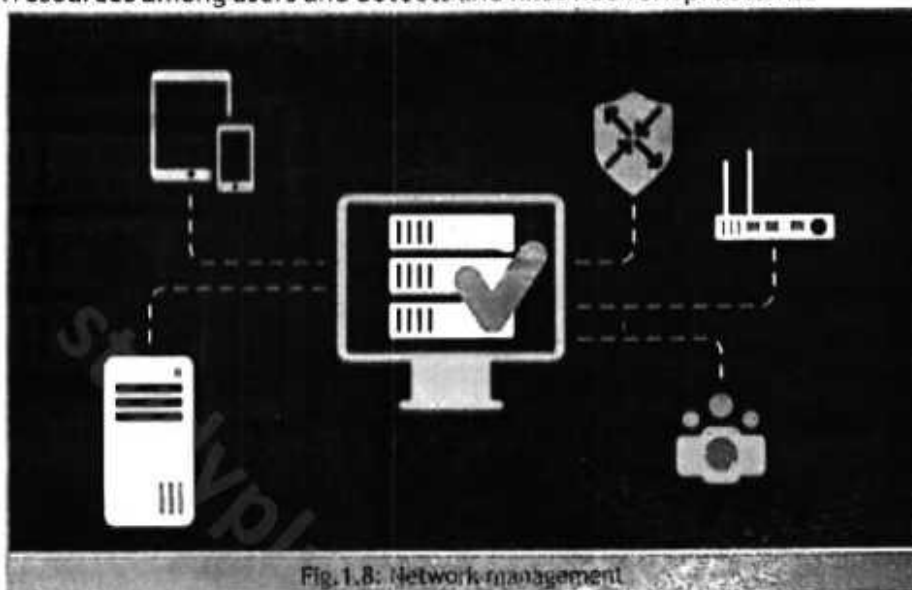


Fig.1.7: Device management

Example: There are three programs A, B and C which are using the printer. Now the OS will decide which program to use the printer first. A queue will be set by the OS and each program will get the printer by its turn.

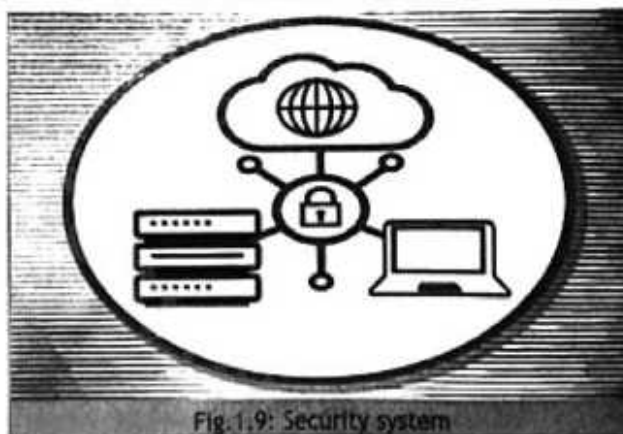
Network Management

Network management is the part of network operating system that monitors and manages the resources of a network. It allows to create user groups and assigns privileges to them. It shares the network resources among users and detects and fixes network problems.



Security management

Security management in an operating system ensures resources are used according to user privileges set by the system administrator. It creates user accounts, enforces security policies, controls access to resources, and protects the system from unauthorized access and malware.

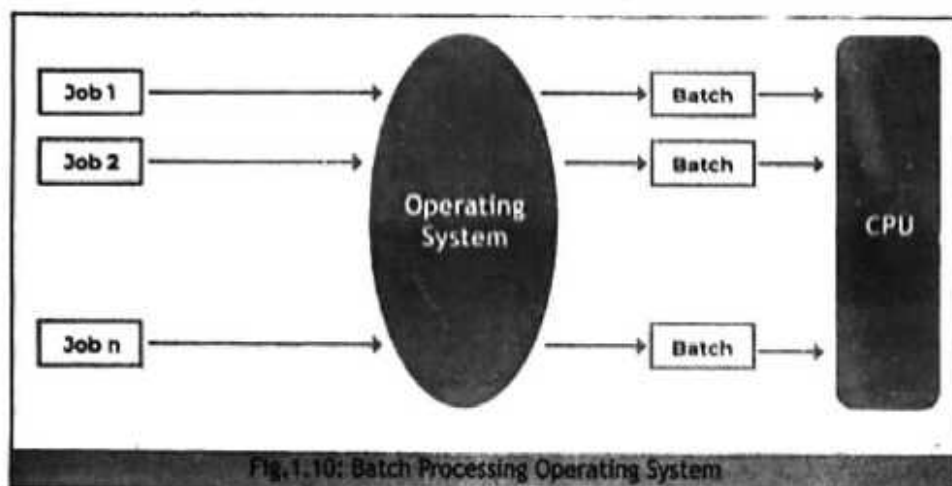


1.4.2 Types of Operating Systems

The following are the important types of operating systems that are commonly used on various computer systems.

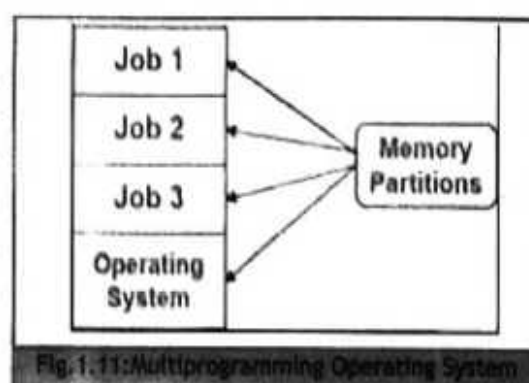
Batch Processing Operating System

A batch processing operating system groups similar tasks into batches and executes them sequentially. It is ideal for repetitive tasks, such as payroll processing or generating bank statements, where the same operation is applied to multiple users or jobs. While this system provides efficiency in handling large volumes of similar tasks, it may introduce delays as jobs are processed one after another.



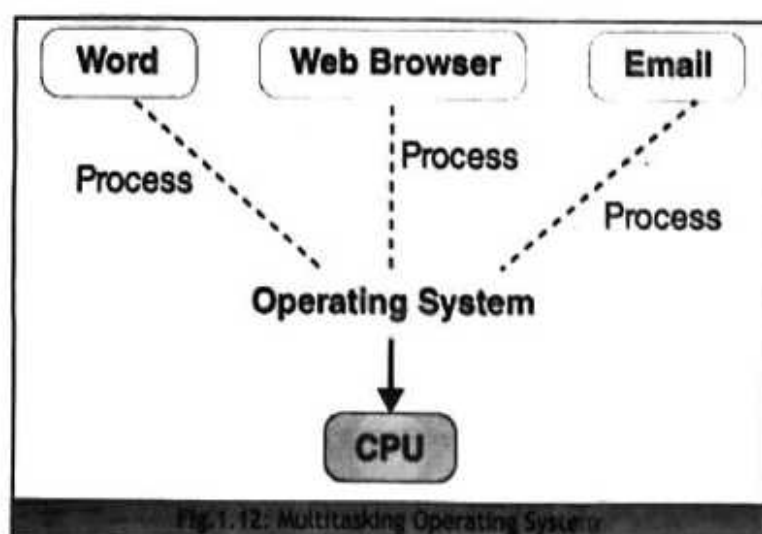
Multiprogramming Operating System (RTOS)

A multiprogramming operating system allows multiple programs to be loaded into the main memory simultaneously. Although the CPU can execute only one program at a time, it switches between programs when one is waiting for input or output, thus maximizing CPU utilization. This system is designed to improve the efficiency of the computer by ensuring that the CPU is rarely idle.



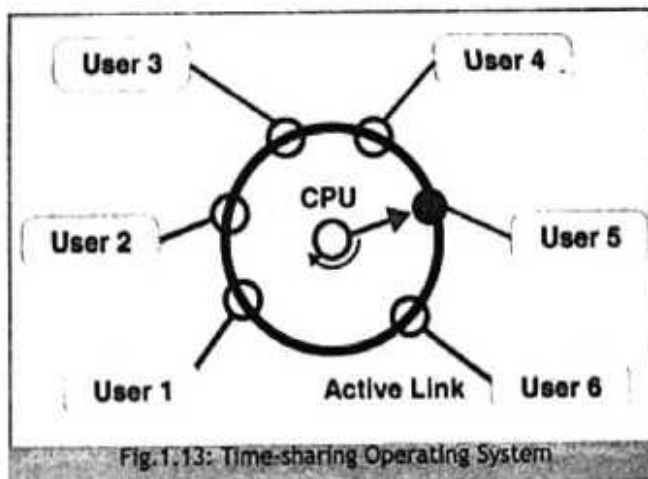
Multitasking Operating System

A multitasking operating system enables multiple tasks to be performed concurrently on a single CPU. The system rapidly switches between tasks, creating the impression that all tasks are running simultaneously. This capability is essential for modern personal computers and mobile devices, where users often run several programs at once.



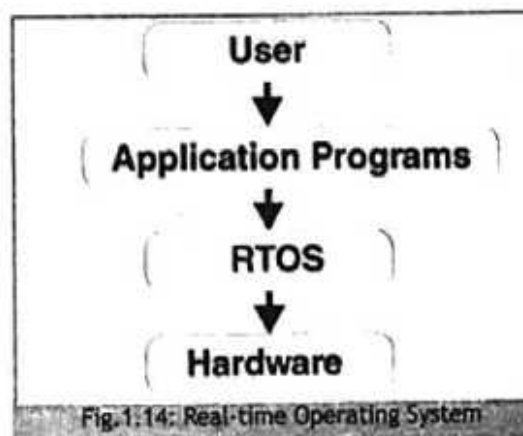
Time-sharing Operating System

A time-sharing operating system divides CPU time into short intervals, known as time slices, and allocates each program a time slice in turn. This method allows multiple users or programs to interact with the system simultaneously, giving the appearance that all are being processed at the same time. Time-sharing systems are commonly used in environments such as banks, universities, and large organizations that require multiple users to access the system concurrently.



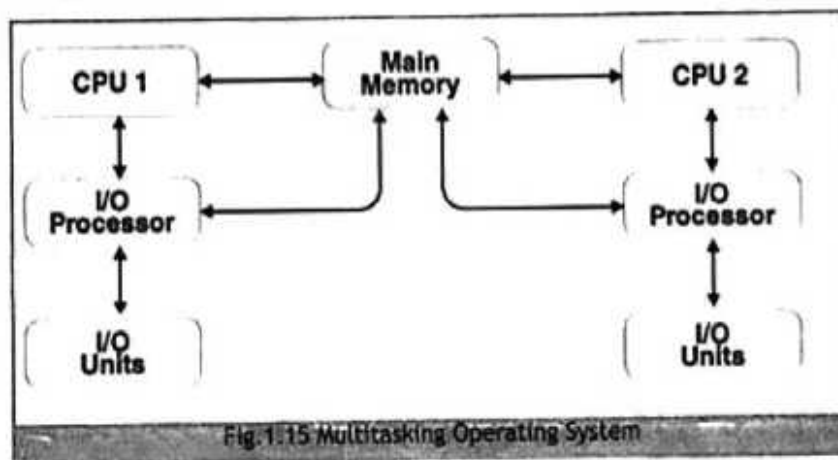
Real-time Operating System (RTOS)

A real-time operating system is designed to process data and provide immediate responses. It is used in environments where timing is critical, such as medical equipment, traffic control systems, and industrial automation. Real-time systems must operate within strict time constraints to ensure that tasks are completed within a specified deadline, making them highly reliable and responsive.



Multiprocessor Operating System

A multiprocessor operating system controls the operation of multiple CPUs within a single computer system. By distributing tasks among several processors, the system can handle large amounts of data more efficiently and increase the overall processing speed. Multiprocessor systems are used in environments requiring high performance, such as data centers, servers, and scientific research applications.



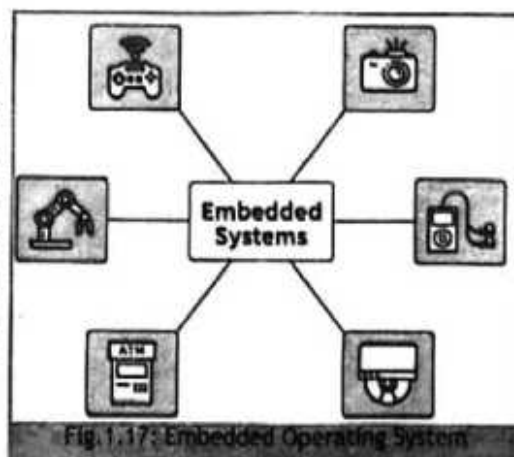
Distributed Operating System

A distributed operating system manages a network of computers, allowing programs to run on multiple machines simultaneously. In a distributed system, the operating system distributes tasks across different computers in the network, balancing the load and providing fast execution of application software. Users do not need to know which machine is processing their tasks, as the distributed OS manages this transparently. This system is widely used in cloud computing and other networked environments.



Embedded Operating System

An embedded operating system is integrated into the hardware of a specific device, such as a microwave oven, TV, or camera. It is designed to perform specialized tasks and operates automatically when the device is turned on. Embedded operating systems are optimized for the specific functions of the device and provide reliable performance with minimal resource requirements. They are commonly used in consumer electronics and industrial machines.



1.4.3. How OS manages to run Applications?

Application programs run on top of operating systems by utilizing the resources and services provided by the operating system. The operating system serves as an intermediary between the hardware and the application programs, managing tasks such as memory allocation, process scheduling, and device management.

For example, when a user opens a word processing program such as Microsoft Word, the operating system allocates memory for the program to run, manages input/output operations such as reading and writing files, and schedules the program to run on the CPU. The operating system also provides access to system resources such as printers and network connections that the application program may need to use.

Overall, the operating system acts as a platform on which application programs can run, providing a layer of abstraction that allows programs to interact with the hardware without needing to know the specific details of the underlying hardware. As shown in Fig 1.18.

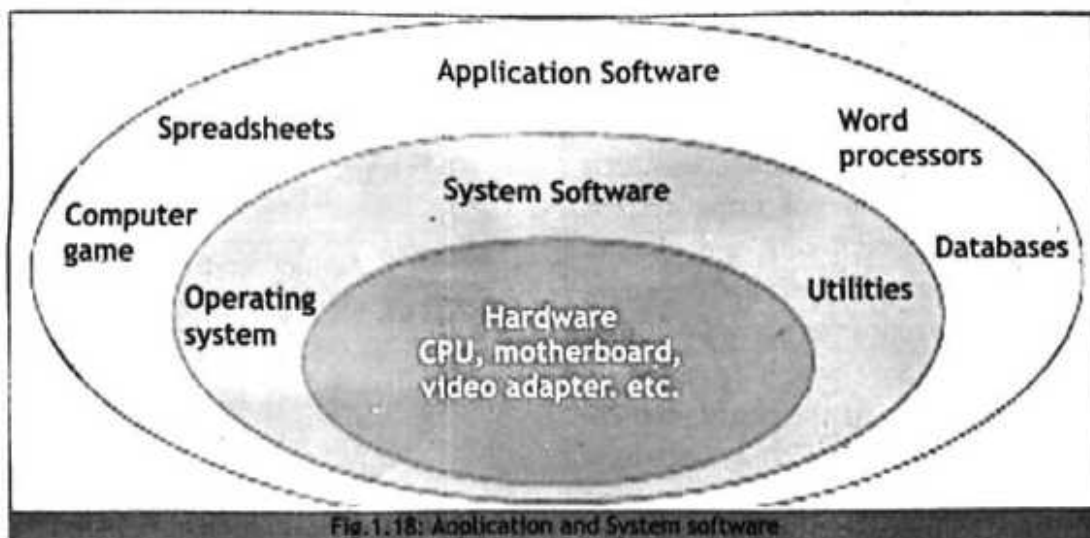


Fig.1.18: Application and System software

1.4.4 Process Management

Process management is an important task of operating system. It allocates systems resources to various processes so that they can run efficiently.

Process:

A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates a binary code. The original code and Binary code, both are programs. When we actually run the binary code, it becomes a process. Process is a part of program under execution that is scheduled and controlled by

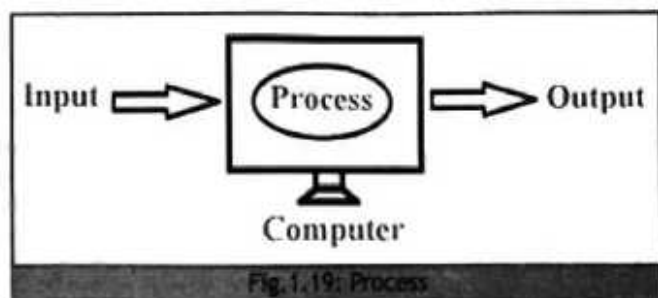


Fig.1.19: Process

operating system. When a program is loaded in memory for execution, it becomes a process. A program is an executable code that is stored in disk as a text file whereas a process is a dynamic instance of a program during its execution in RAM. It represents basic unit of work. It uses various resources of computer such as CPU time, files, I/O devices, memory, etc.

Various States of a Process:

There are five states of a process which are start, ready, running, waiting and terminated as shown in Figure.

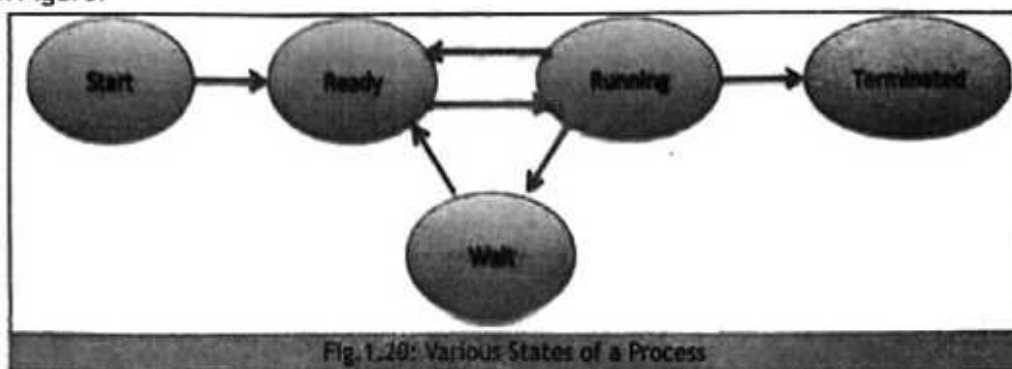


Fig.1.20: Various States of a Process

Start/New State: This is the first state of a process when it is created. Any new operation or service that is requested by a program for execution by the processor is known as new state of process.

Ready State: A process is said to be in ready state when it is ready for execution but it is waiting to be assigned to the processor by the operating system.

Running State: A process is said to be in running state when it is being executed by the processor. A process is assigned to a processor for execution by operating system.

Blocked State/Waiting State: A process is in blocked or waiting state when it is not under execution. It is waiting for a resource to become available.

Terminated State: A process is in terminated state when it completes its execution.

Thread and Process:

In programming, there are two basic units of execution: processes and threads. They both execute a series of instructions. A Process is an instance of a program that is being executed. A process may be made up of multiple threads. A Thread is a basic ordered sequence of instructions within a process that cannot be executed independently. The threads are made of and exist within a process; every process has at least one thread. Multiple threads can also exist in a process and share resources.

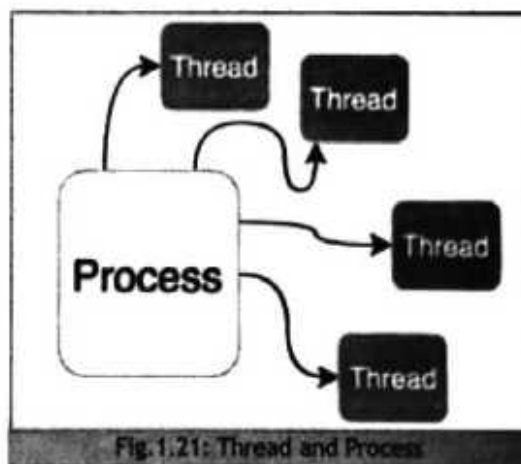


Fig.1.21: Thread and Process

Comparison between Process and Thread

	Process	Thread
1	An executing instance of a program is called a process.	A thread is a subset of the process.
2	It has its own copy of the data segment of the parent process.	It has direct access to the data segment of its process.
3	Any change in the process does not affect other processes.	Any change in the thread may affect the behavior of the other threads of the process.
4	Processes run in separate memory spaces.	Threads run in shared memory spaces.
5	Process is controlled by the operating system.	Threads are controlled by programmer in a program.
6	Processes are independent.	Threads are dependent.

Process Scheduler:

A process scheduler is part of the operating system that manages the execution order of processes by the CPU. It ensures efficient and fair use of CPU time by allocating processing time based on specific scheduling algorithms.

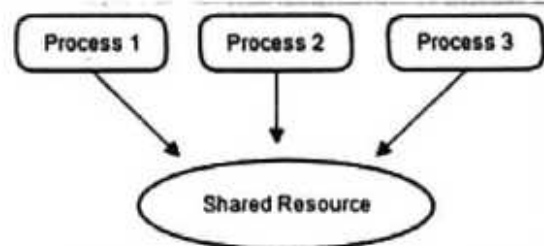


Fig.1.22: Process Synchronization

Process Synchronization:

Process synchronization is a method used to coordinate multiple processes or threads that share resources. It prevents issues like race conditions, which occur when processes try to access the same memory at the same time. The operating system controls the execution order to avoid conflicts.

Interrupts:

Interrupts are signals that alert the CPU to events that need immediate attention. When an interrupt occurs, the CPU stops its current task, saves its state, and runs a special program called an **Interrupt Service Routine (ISR)** to handle the event. For example, when you receive an email, the system interrupts the CPU to process the email quickly.

Deadlock:

A deadlock happens when two or more processes are stuck, each waiting for a resource held by the other. For example, as shown in Fig. 1.23, if Process P1 needs Resource R2 held by Process P2, while P2 needs Resource R1 held by P1, neither can proceed. The operating system must detect and resolve these deadlocks to keep processes running smoothly.

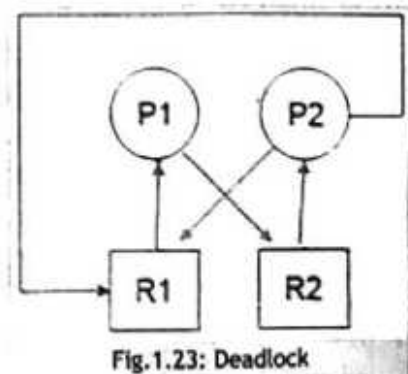


Fig.1.23: Deadlock

1.4.5. Computer System Resources managed by an Operating System

The four main computer system resources managed by an operating system are:

CPU (Central Processing Unit) Management:

The operating system manages the CPU by scheduling processes, ensuring efficient utilization of the CPU, and prioritizing tasks. This involves process scheduling algorithms, deadlock resolution, and handling interrupts.

Memory Management:

The operating system is responsible for managing the system's memory, which includes RAM. This involves allocation and deallocation of memory spaces to various programs, managing memory hierarchy, and ensuring that each process has adequate memory while optimizing the use of available memory.

Storage Management:

The operating system manages data storage, including hard drives, SSDs, and other storage

devices. This involves file system management, managing read/write operations, ensuring data integrity, and providing an interface for file and directory management.

I/O (Input/Output) Management:

The operating system manages input and output devices such as keyboards, mice, printers, and network interfaces. This includes managing device communication, buffering, and handling device drivers to ensure that the I/O operations are carried out smoothly and efficiently.

14.3 File Organization of the File System

The file system is a critical component of an operating system, responsible for organizing, storing, retrieving, and managing data on storage devices like hard drives, SSDs, and USB drives. Here's a detailed explanation of the structure and organization of a file system:

File System Components

A file system is composed of several key components:

Files: The smallest unit of data storage, representing a collection of bytes stored on a disk. Files can be of various types, such as text files, executables, images, etc.

Directories/Folders: Also known as folders, directories are containers that hold files and other directories, providing a hierarchical structure.

Metadata: Information about files and directories, such as their names, sizes, types, permissions, and timestamps (creation, modification, and access times).

Directory/Folder Structure

The directory/folder structure can be organized in different ways, commonly used directory/folder structures are shown in Fig. 1.24.

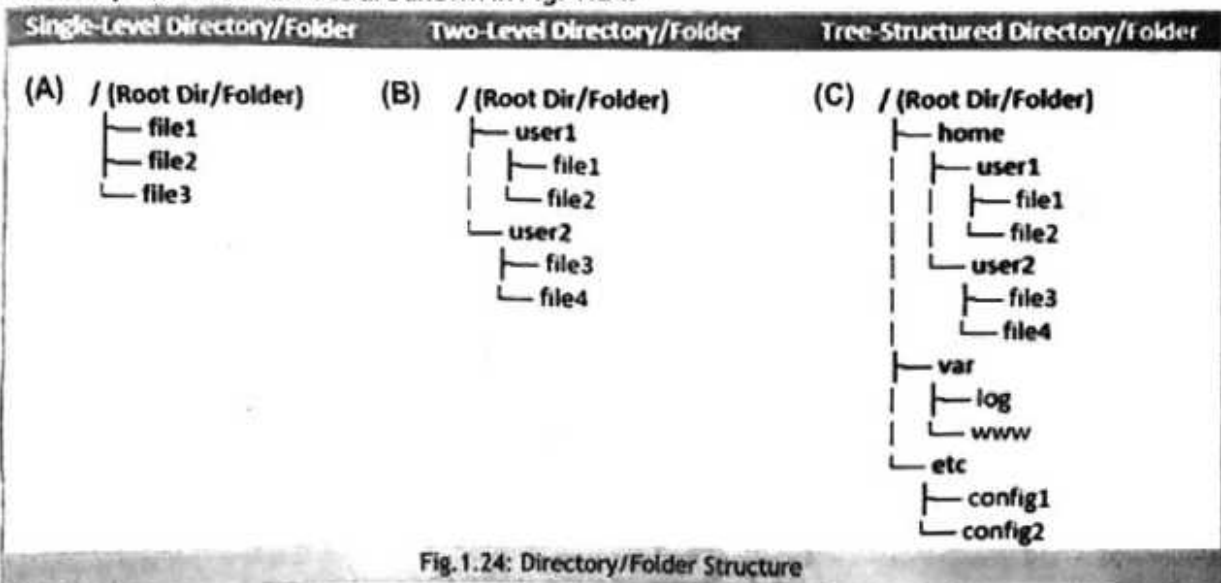


Fig. 1.24: Directory/Folder Structure

Single-Level Directory/Folder: All files are contained in a single directory/folder. This structure is simple but becomes inefficient as the number of files grows. Figure A shows single level directory/folder structure.

Two-Level Directory/Folder: Each user has their own directory under the root directory, solving some of the limitations of the single-level structure. Figure B shows two level directory/folder structure.

Tree-Structured Directory/Folder: The most common structure, allowing a hierarchy of directories and subdirectories, resembling a tree. This structure is flexible and scalable. Figure C shows tree-structured directory/folder structure.

File Allocation Methods

Files can be stored in different ways:

- **Contiguous Allocation:** Files occupy consecutive blocks, which speeds up access but can cause fragmentation.
- **Linked Allocation:** Files are stored in linked lists, allowing flexibility but possibly slower access due to pointer traversal.
- **Indexed Allocation:** Uses an index to point to the actual data blocks, allowing faster access while balancing storage efficiency.

File System Types

Various file systems are designed to meet different needs and have unique features. Some common file systems include:

- **FAT (File Allocation Table):** An older file system used by DOS and Windows. Simple but limited in features and scalability.
- **NTFS (New Technology File System):** Used by Windows NT and later versions. Supports large files, security features, and disk quotas.
- **ext3/ext4 (Extended File System 3/4):** Common in Linux environments. ext4 offers improved performance and features like journaling, which helps recover from crashes.

File Operations

File systems provide various operations to manage files and directories. Common file operations are:

Create: Making new files and directories.

Open: Accessing files for reading or writing.

Read: Retrieving data from files.

Write: Adding or modifying data in files.

Delete: Removing files and directories.

Rename: Changing the names of files and directories.

1.5 Computer Software

Computer software, often referred to simply as “software,” is a collection of programs, data, and instructions that tell a computer how to perform specific tasks or functions. It is an important component of any computer system, enabling it to process data, run applications, and interact

with users. Software is typically categorized into the following main types:

- System Software
- Application Software
- Programming software
- Driver software

1.5.1 Comparing Different Software Types

Software Type	Purpose & Function	Examples
System Software	➤ Manages hardware and system resources. ➤ Controls overall system operations and provides a platform for applications to run. ➤ Essential for the computer's overall functioning.	Windows, macOS, Linux, Android
Programming Software	➤ Provides tools for writing, testing, and debugging code. ➤ Translates programming languages into machine-readable code, helping developers create software.	Eclipse for Java, Coda for Mac, Visual Studio for multiple languages, GitHub for source code
Application Software	➤ Performs specific tasks for the user, like writing, streaming, or browsing. ➤ Designed to meet user needs for specific activities.	Microsoft Word, Spotify, Google Chrome
Driver Software	➤ Acts as a bridge between the operating system and hardware devices. ➤ Ensures proper communication and functionality between the system and external devices.	Printer drivers, NVIDIA graphics drivers, Wi-Fi drivers

1.5.2 Offline and Online Applications

Offline Applications

Offline applications are programs that can run without an internet connection. They rely entirely on the device's local resources, such as storage and processing power, to function. These apps are useful when there's no access to the internet, but they can still perform tasks.

Examples:

- Microsoft Word - used for writing and editing documents.
- Adobe Photoshop - used for image editing and graphic design.
- VLC Media Player - plays media files like videos and music stored locally on the device.

Online Applications

Online applications require an internet connection to function because they access resources, data, or services hosted on remote servers. These apps often provide real-time data and collaborative features.

Examples:

- Google Docs - for creating and sharing documents in real-time.
- Spotify - for streaming music.
- Gmail - for sending and receiving emails.

1.5.3 Uses of Common Productivity Application Software

Software Type	Examples	Uses
Word Processors	Microsoft Word, Google Docs	Writing essays, reports, letters, resumes, and professional documents.
Spreadsheets	Microsoft Excel, Google Sheets	Budgeting, financial analysis, tracking inventory, data charts.
Presentation Software	Microsoft PowerPoint, Google Slides	Business presentations, educational lectures, and project proposals.
Database Management Systems (DBMS)	Microsoft Access, MySQL	Managing customer data, inventory, employee records, and large datasets.
Email Clients	Microsoft Outlook, Mozilla Thunderbird	Organizing emails, scheduling appointments, managing contacts.
Note-Taking Applications	Evernote, Microsoft OneNote	Keeping track of tasks, lecture notes, organizing projects and ideas.
Calendar/Task Management	Google Calendar, Microsoft To-Do	Managing time, setting reminders, planning meetings, tracking tasks.
PDF Editors	Adobe Acrobat, Foxit Reader	Editing PDFs, signing documents, converting documents to/from PDF.
Graphic Design Software	Adobe Photoshop, Canva	Designing posters, flyers, social media content, and marketing materials.
Graphic Design Software	Microsoft Project, Trello	Organizing tasks, tracking project timelines, team collaboration.
Video Conferencing Tools	Zoom, Microsoft Teams	Conducting meetings, webinars, virtual classrooms, and online collaboration.

1.5.4 Application Patch

An application patch is a piece of software designed to update, fix, or improve an existing program or its supporting data. Patches address issues such as security vulnerabilities, bugs, or performance problems and can also add new features or functionalities to the application.

Key Functions of Application Patches:

Fix bugs: Resolve software glitches or errors.

Improve security: Patch vulnerabilities to prevent attacks.

Enhance performance: Optimize the program to run more smoothly.

Add features: Introduce new functionalities or improvements to the existing software.

Examples:

Security Patch for Windows: Microsoft regularly releases security patches to fix vulnerabilities in its operating systems to prevent malware attacks.

Game Updates: Games like World of Warcraft and Call of Duty often release patches that fix bugs, balance gameplay, and add new content.

Adobe Acrobat Patch: Adobe frequently provides patches to fix issues or add security updates to its PDF editing software.

1.5.5 Software Hosting

Software hosting refers to the process of deploying, managing, and providing access to software applications on servers. These servers can be located on-premises, in data centers, or in the cloud. The primary objective of software hosting is to make applications accessible to users over a network, typically the internet.

Some common types of Software Hosting are:

On-Premises Hosting: On-premises hosting involves installing and running software on servers located

within an organization's physical premises. This setup offers complete control over the hardware, software, and data security, making it ideal for organizations with strict compliance and security requirements. However, it comes with high upfront costs for purchasing and setting up the hardware, as well as ongoing maintenance and operational expenses. As shown in Fig. 1.25.

Pros:

- You have complete control over hardware and data security.
- It is customizable to specific needs and compliance requirements.

Cons:

- It has high initial and ongoing costs.

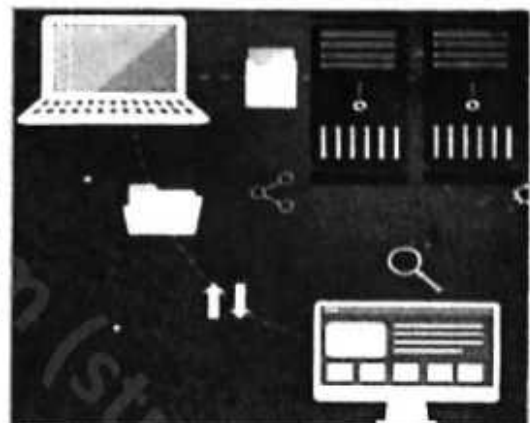


Fig. 1.25: On-Premises Hosting

- It requires in-house technical expertise. It has limited scalability.

Shared Hosting:

Shared hosting involves multiple websites or applications sharing the same server resources provided by a hosting company. This is a cost-effective solution, making it suitable for small websites and applications with moderate traffic. The hosting provider handles server maintenance, updates, and security. However, shared resources can lead to performance issues, and there are potential security risks due to neighboring sites on the same server. As shown in Fig. 1.26.



Fig.1.26: Shared hosting

Pros:

- It is cost-effective.
- It is easy to set up and manage.

Cons:

- You have limited control and customization. There are potential performance issues.
- There are security risks from other sites on the server.

Dedicated Hosting:

Dedicated hosting provides an entire physical server for the exclusive use of a single user or organization. This setup ensures high performance, reliability, and security, making it suitable for resource-intensive applications and high-traffic websites. Users have full control over the server configuration. However, dedicated hosting is costly and requires significant technical expertise for server management and maintenance. As shown in Fig. 1.27.



Fig.1.27: Dedicated Hosting

Pros:

- It provides high performance and reliability.
- You have complete control over server configuration.
- It offers enhanced security.

Cons:

- It has a high cost.
- It requires technical expertise for management.

Cloud Hosting:

Cloud hosting uses a network of virtual and physical servers to provide scalable and flexible resources on demand. Managed by cloud providers like AWS, Google Cloud, and Microsoft Azure, it offers high scalability, cost-efficiency with a pay-as-you-go model, and excellent reliability. Cloud hosting is ideal for applications with variable or unpredictable traffic. However, it depends on internet connectivity, and there may be concerns over data security and compliance. As shown in Fig. 1.28.



Fig. 1.28: Cloud Hosting

Pros:

It is highly scalable and flexible.

It is cost-efficient with pay-as-you-go pricing. It is reliable and supports global reach.

Cons:

- It is dependent on internet connectivity.
- There are potential data security and compliance concerns.

1.5.6 Programming Software

Programming software, often referred to as Integrated Development Environments (IDEs), text editors, and compilers, helps programmers write, debug, and compile code efficiently. The following is a detailed explanation of how these tools assist programmers in the coding and compilation process.

Components of Programming Software:

Text Editors: Text editors are basic tools that allow programmers to write and edit code. Examples include Notepad++, Sublime Text, and Visual Studio Code.

Integrated Development Environments (IDEs): IDEs are comprehensive tools that combine a text editor, debugger, and compiler in one interface. Examples include IntelliJ IDEA, Eclipse, and Visual Studio.

Compilers: Compilers translate code written in a high-level programming language into machine code that the computer can execute. Examples include GCC (GNU Compiler Collection) for C/C++ and javac for Java.

How Programming Software Helps:

Writing Code: Text Editors and IDEs provide a platform for writing code with features like syntax highlighting, which colors the code according to its syntax, making it easier to read and understand. For example, keywords in a programming language might appear in blue, while comments are in green.

Example: In Visual Studio Code, typing `if` in a Python file automatically highlights it and suggests possible completions, making coding faster and reducing errors.

Code Completion: IDEs offer advanced code completion, where the software predicts and suggests the next part of the code you are typing. This feature helps in writing code faster and with fewer errors.

Debugging: IDEs come with integrated debuggers that allow programmers to run their code step-by-step, inspect variables, and understand the flow of execution to find and fix bugs (errors). Fig.1.29 shows Python programming language IDE.



Fig.1.29: Python programming language IDE

Error Checking and Syntax Highlighting:

Text Editors and IDEs provide real-time error checking, underlining syntax errors and other issues as you type. This immediate feedback helps in quickly identifying and correcting mistakes. Fig 1.30 shows syntax error in a program.

Example: In Visual Studio Code, if you forget a semicolon in a JavaScript file, it highlights the error and provides suggestions for fixing it.

Compiling Code: Compilers take the code written by programmers and translate it into executable machine code. They also perform various optimizations to make the code run more efficiently.



Fig.1.30: Syntax error

Project Management:

IDEs offer project management features that help organize code files, manage dependencies, and streamline the build process.

Example: In Visual Studio Code, creating a new project sets up a directory structure with folders for source files, libraries, and dependencies, making it easier to manage complex applications.

Learning Activity

"Hello World! Project"

By following these detailed steps, you can create and run a simple "Hello World!" project using HTML, CSS, and JavaScript directly in Visual Studio Code using your default browser, like Google Chrome. This setup allows you to see real-time updates as you make changes to your code.

Setting Up Visual Studio Code for HTML, CSS, and JavaScript

Before you begin, ensure you have [Visual Studio Code installed](#).

Step-by-Step Guide

1. Open Visual Studio Code program. As Shown in Fig 1.31.

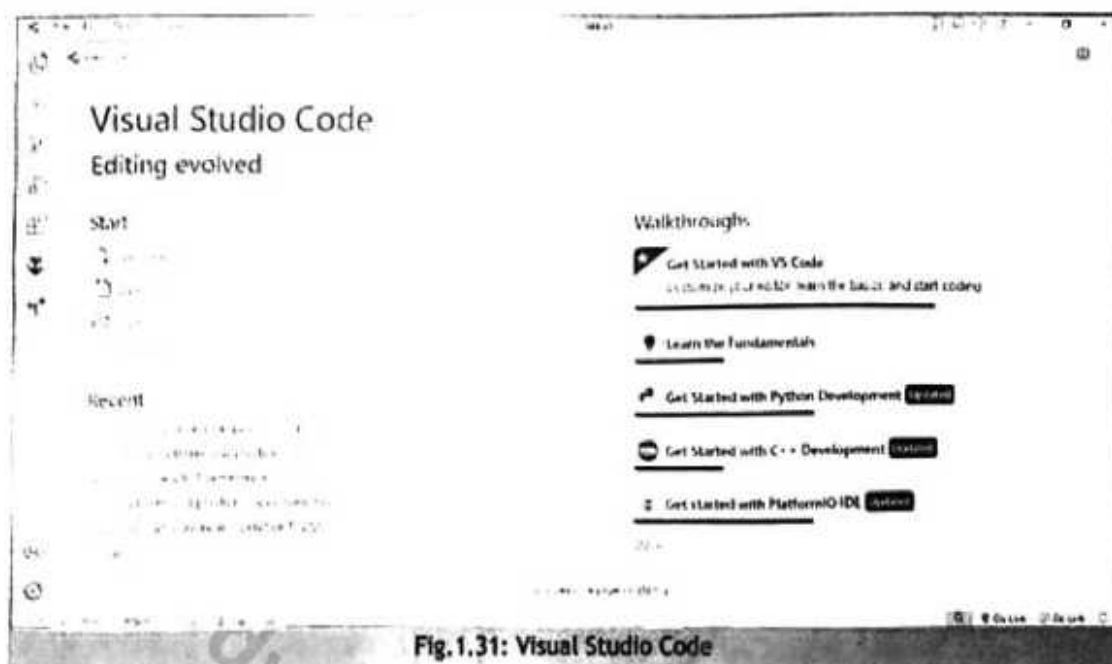


Fig.1.31: Visual Studio Code

2. Create the Project Files:

Create a new folder on your computer and name it HelloWorldProject. As shown in Fig 1.32.

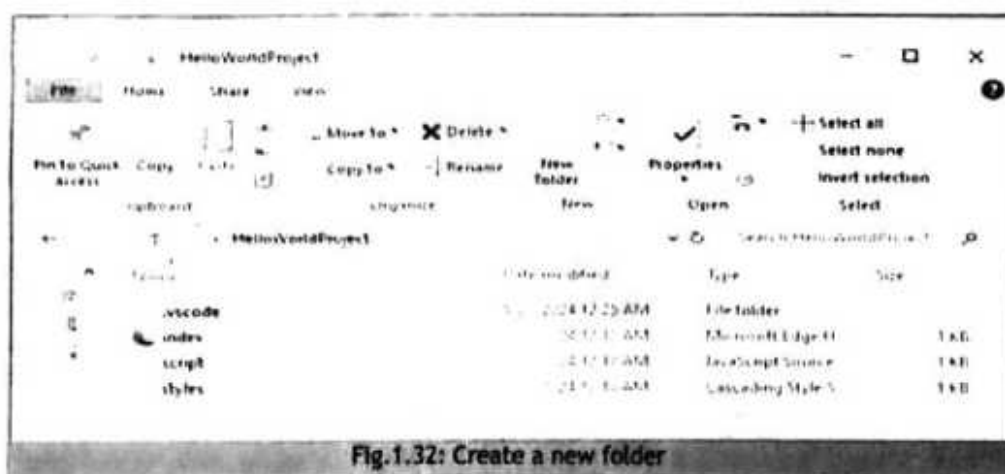


Fig.1.32: Create a new folder

3. Open the Project Folder in Visual Studio Code:

- In Visual Studio Code, click on File > Open Folder....
- Navigate to and select your HelloWorldProject folder.

4. Create HTML, CSS, and JavaScript Files:

- In the Explorer pane on the left side of VS Code, right-click on the folder name (HelloWorldProject).
- Select New File and create the following files:

index.html

styles.css

script.js

5. Add the Code to the Files

HTML (index.html):

- Open index.html and type the following code. As Shown in Fig 1.33.

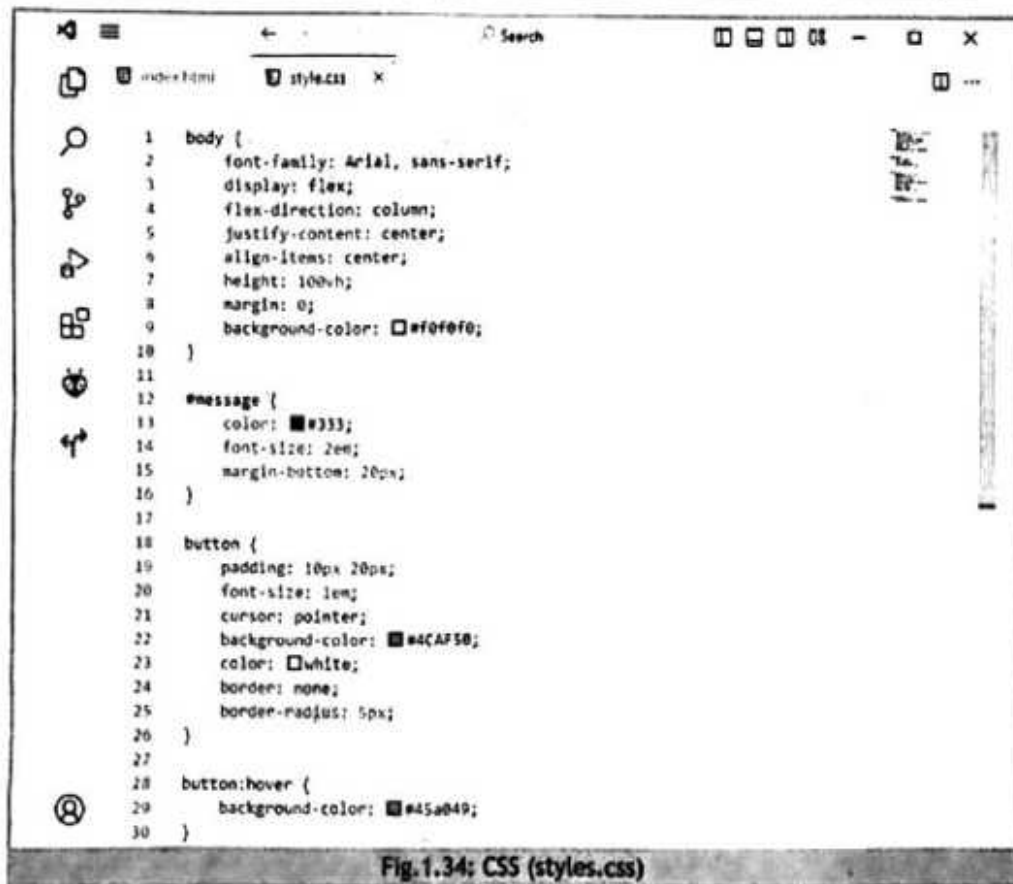


Fig.1.33: Index.html

- This is the main HTML file that will structure your webpage.

CSS (styles.css):

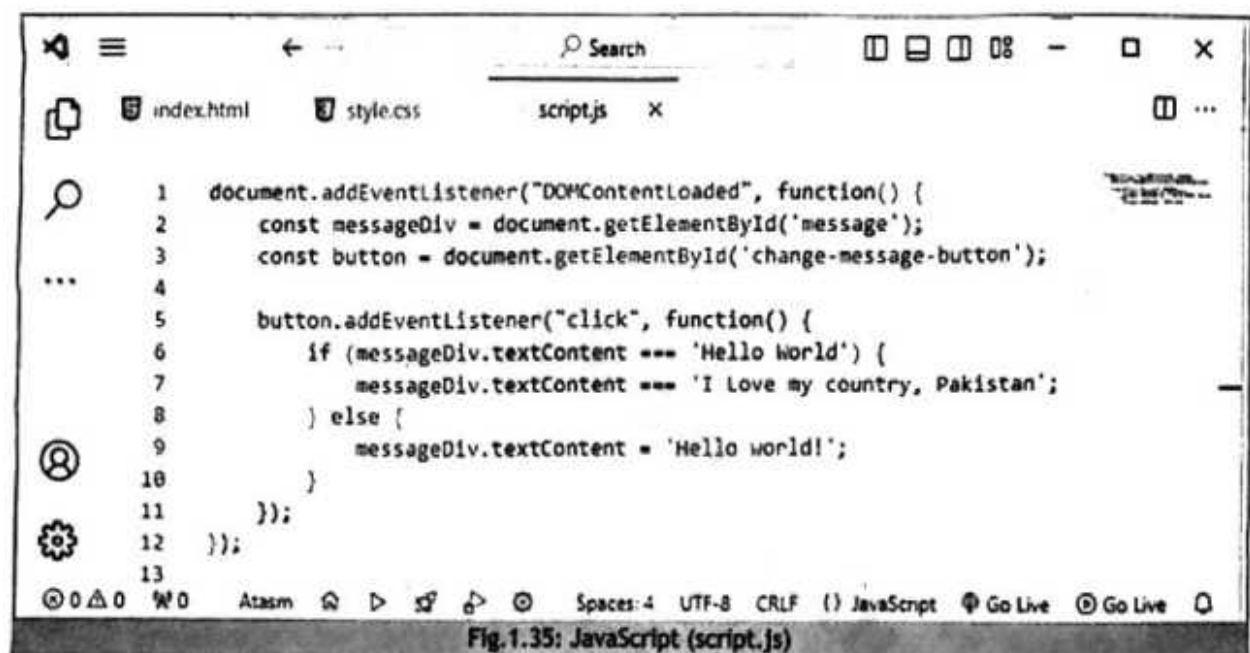
- Open styles.css and type the following code. As Shown in Fig 1.34.
- This file contains the CSS styles for your HTML elements.



JavaScript (script.js):

Open script.js and type the following code. As Shown in Fig 1.35.

This file contains the JavaScript code that adds interactivity to your webpage.



Running the Example

View in Browser:

- Your default browser (like Google Chrome) should open a new tab displaying your HTML file. You should see a "Hello, World!" heading and a "Change Message" button. As Shown in Fig 1.36.



Fig.1.36: Running the Example

- Clicking the Change Message button will activate a JavaScript alert saying "I Love my country, Pakistan". As Shown in Fig 1.37.

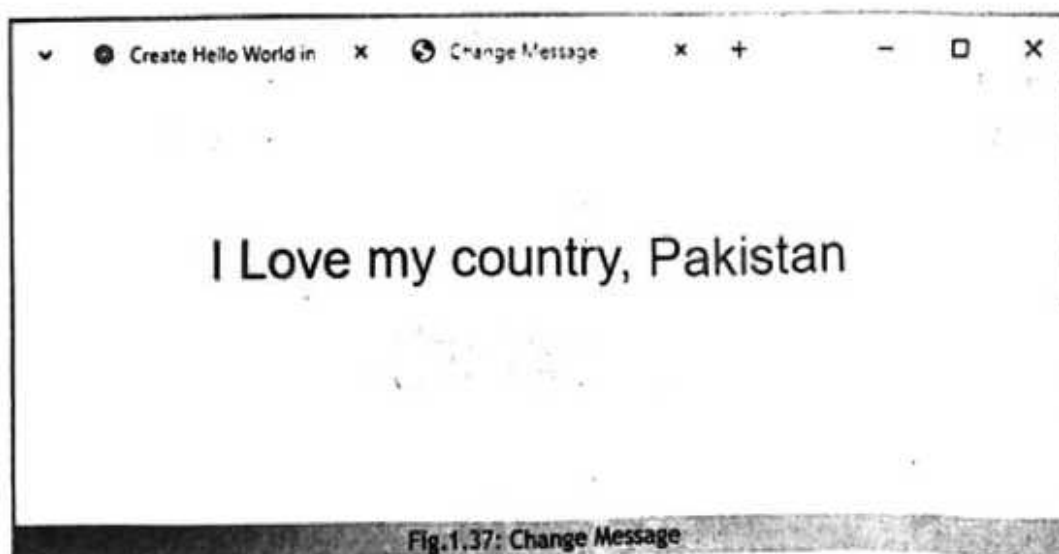


Fig.1.37: Change Message

Unit Summary

- **Machine-Level Representation of Data:** Data is stored in binary (bits) and represented in various data types.
- **Numbering Systems:** Include decimal, binary, octal, and hexadecimal systems for number representation and conversions.
- **Decimal System:** Uses base 10 with symbols 0-9.
- **Binary System:** Base 2, using 0 and 1 for data representation.
- **Octal System:** Base 8, with symbols 0-7.
- **Hexadecimal System:** Base 16, using symbols 0-9 and A-F for compact data representation.
- **Number System Conversions:** Methods to convert between decimal, binary, octal, and hexadecimal.
- **Binary Arithmetic:** Computers perform addition, subtraction, multiplication, and division in binary.
- **Overflow/Underflow:** Occurs when a calculation exceeds or falls below the representable range.
- **Complements (1's and 2's):** Used in binary operations, especially for subtraction.
- **Binary Subtraction:** Done using 1's or 2's complement to simplify operations.
- **Signed/Unsigned Numbers:** Binary numbers can be represented with or without a sign bit to indicate positive or negative values.
- **Floating-Point Representation:** Uses the IEEE 754 standard for storing real numbers, dividing them into sign, exponent, and mantissa.
- **Common Coding Schemes (ASCII/Unicode):** Convert characters into binary; ASCII uses 7 or 8-bit codes, Unicode supports global languages.
- **Operating System (OS) Overview:** Manages memory, processes, devices, and software, enabling user interaction.
- **Main OS Tasks:** Includes managing processes, memory, files, devices, networks, and security.
- **Process Management:** Allocates CPU time and optimizes resource use.
- **Memory Management:** Allocates and manages memory for programs.
- **File Management:** Handles file creation, deletion, and organization.
- **Device Management:** Controls and optimizes input/output devices.
- **Network Management:** Manages network resources and resolves network issues.
- **Security Management:** Enforces security policies and manages user accounts.

- **Batch Processing OS:** Processes tasks in batches, improving efficiency.
- **Multiprogramming OS:** Executes multiple programs simultaneously in memory.
- **Multitasking OS:** Performs multiple tasks at once by quickly switching between programs.
- **Time-sharing OS:** Shares CPU time across programs, simulating simultaneous processing.
- **Real-time OS:** Executes tasks within strict timing constraints, often used in industrial processes.
- **Multiprocessor OS:** Manages multiple CPUs to improve performance.
- **Parallel Processing OS:** Runs multiple processes at once using several processors.
- **Distributed OS:** Manages a distributed system, balancing load for fast execution.
- **Embedded OS:** Built into devices to manage hardware and perform specific tasks automatically.
- **Application Management:** OS manages resources for running applications efficiently.
- **Process Scheduler:** Manages CPU time for process execution, ensuring fairness and efficiency.
- **Process Synchronization:** Coordinates access to shared resources, preventing conflicts.
- **Scheduling Algorithms:** Includes FCFS, SJN/SJF, priority scheduling, and round-robin.
- **Interrupts:** Signals to the CPU that require immediate action, handled by interrupt routines.
- **Deadlock:** Occurs when processes block each other indefinitely over shared resources.
- **System Resources Managed by OS:** CPU, memory, storage, and I/O devices.
- **OS Design:** Includes kernel, process management, memory management, file systems, device management, security, networking, and user interface.
- **File System Structure:** Files, directories, and metadata organized into single-level, two-level, or tree structures.
- **File Allocation Methods:** Contiguous, linked, and indexed methods for storing files.
- **File System Types:** Examples include FAT, NTFS, and ext3/ext4, each with unique features.
- **File Operations:** Include creating, reading, writing, deleting, and renaming files.
- **Software Hosting:** Involves deploying software on servers for network accessibility.
- **On-Premises Hosting:** Software hosted on internal servers, offering control but with high cost.

- **Shared Hosting:** Multiple websites share a single server, offering low cost but potential performance issues.
- **Dedicated Hosting:** A server dedicated to one user, providing high performance and security at a higher cost.
- **Cloud Hosting:** Uses cloud servers for scalability and cost-efficiency, though dependent on connectivity.
- **Programming Software:** IDEs, text editors, and compilers help write, debug, and compile code.
- **Components of Programming Software:** Text editors for writing, IDEs for a full coding environment, and compilers for translating code.
- **Writing Code:** Tools like syntax highlighting and code completion enhance coding efficiency.
- **Debugging:** IDEs provide tools for finding and fixing bugs by stepping through code.
- **Error Checking:** Real-time error checking and syntax highlighting help avoid coding mistakes.
- **System Software:** Manages hardware and system resources, controls operations, and provides a platform for applications (e.g., Windows, macOS).
- **Programming Software:** Offers tools for writing, testing, and debugging code; translates code into machine language (e.g., Eclipse, Visual Studio).
- **Application Software:** Performs specific user tasks such as writing, streaming, or browsing (e.g., Microsoft Word, Spotify).
- **Driver Software:** Enables communication between the operating system and hardware devices (e.g., printer drivers, Wi-Fi drivers).
- **Offline Applications:** Run without internet, rely on local resources (e.g., Microsoft Word, Adobe Photoshop).
- **Online Applications:** Require an internet connection to access data or services (e.g., Google Docs, Spotify, Gmail).
- **Word Processors:** Used for writing documents (e.g., Microsoft Word, Google Docs).
- **Spreadsheets:** For data analysis, budgeting (e.g., Excel, Google Sheets).
- **Presentation Software:** Create presentations for business or education (e.g., PowerPoint, Google Slides).
- **Database Management:** Organize large datasets (e.g., Microsoft Access, MySQL).
- **Email Clients:** Manage emails and appointments (e.g., Outlook, Thunderbird).
- **Note-Taking Apps:** Organize notes and tasks (e.g., Evernote, OneNote).

Exercise

Q1. Select the best answer for the following MCQs.

1. What is the binary representation of the decimal number -43 using 8-bit two's complement notation?
a) 10101011 b) 11010101 c) 11010101 d) 11101011
2. Which of the following binary numbers corresponds to the hexadecimal number 3FA7?
a) 0011 1111 1010 0111 b) 1100 0011 1101 1000
c) 1011 1100 0011 0111 d) 0111 1011 1011 1100
3. How does the ASCII code for the character 'Z' (90 in decimal) look in binary?
a) 01011010 b) 01001110 c) 01011000 d) 01101001
4. In an 8-bit signed integer representation, what is the range of values that can be represented?
a) -127 to 128 b) -128 to 127 c) 0 to 255 d) -256 to 255
5. How many different values can be represented with a 10-bit binary number?
a) 256 b) 1024 c) 512 d) 2048
6. Which of the following is NOT a task performed by an operating system?
a) Process Management b) Memory Management
c) Email Management d) Device Management
7. Which type of operating system allows multiple tasks to be performed at the same time?
a) Multiprogramming b) Batch Processing c) Real-time d) Single-tasking
8. Which state of a process represents that it is ready for execution but waiting for CPU time?
a) Start State b) Running State c) Ready State d) Blocked State
9. What is the main purpose of a process scheduler?
a) To manage file operations b) To manage CPU utilization
c) To manage memory allocation d) To manage network resources
10. Which memory management approach involves dividing memory into fixed-sized blocks?
a) Paging b) Segmentation
c) Virtual Memory d) Contiguous Allocation
11. What is the main purpose of interrupts in an operating system?
a) To speed up CPU processing b) To slow down CPU processing
c) To handle immediate events d) To manage network connections

-
12. Which scheduling algorithm assigns CPU time in fixed time slices to each process in a cyclic order?
- a) First-Come First-Served (FCFS)
 - b) Shortest Job Next (SJN)
 - c) Priority Scheduling
 - d) Round Robin (RR)
13. Which memory management technique divides memory into fixed-sized blocks?
- a) Segmentation
 - b) Paging
 - c) Fixed Partitioning
 - d) Dynamic Partitioning
14. Which type of hosting involves installing and running software on servers located within an organization's physical premises?
- a) Shared Hosting
 - b) Dedicated Hosting
 - c) On-Premises Hosting
 - d) Cloud Hosting
15. Which component of programming software translates code written in a high-level programming language into machine code?
- a) Text Editor
 - b) Compiler
 - c) Debugger
 - d) IDE

Q2. Write answers of the following short response questions.

1. What is the significance of the Most Significant Bit (MSB) in signed binary numbers?
2. What is a binary digit, and why is it fundamental in computer systems?
3. Why are binary numbers more efficient for computer calculations than decimal numbers?
4. How do ASCII and Unicode differ in character representation?
5. What is the importance of positional value in number systems? Give two examples.
6. What is the process to convert a binary number to its hexadecimal equivalent?
7. What is the purpose of machine code in computer operations?
8. Why coding scheme is used in computer? Give three reasons.
9. State five differences between a process and a thread.
10. What is memory management and how does it work in an operating system?
11. What is a real-time operating system and where is it commonly used?
12. Differentiate between multiprogramming and multitasking operating systems.
13. List two pros and two cons of on-premises hosting.
14. What is an application patch? Give key functions of it.
15. Differentiate between offline and online applications. Give one example of each.

Q3. Write answers of the following extended response questions.

1. Explain decimal, binary, octal and hexadecimal number systems with examples.

-
2. What is 2's complement, and why it is used? Give examples.
 3. Define the terms Overflow and Underflow. Explain Overflow in number system with one example.
 4. What is the significance of ASCII and Unicode character coding schemes?
 5. What is an operating system? Explain any five tasks/functions of OS.
 6. Explain any four types of operating systems.
 7. How operating system manages applications? Explain five states of a process with diagram.
 8. Explain the role of Integrated Development Environments (IDEs) in the software development process, including specific features that enhance programming efficiency.
 9. What is software hosting? Explain On-premises and Cloud hosting. Also give two pros and cons of each.