



**DR. BABASAHEB AMBEDKAR
OPEN UNIVERSITY**

BCA

BACHELOR OF COMPUTER APPLICATION



BCAR-203

Digital Electronics & Computer Organisation

DIGITAL ELECTRONICS AND COMPUTER ORGANIZATION



**DR. BABASAHEB AMBEDKAR OPEN UNIVERSITY
AHMEDABAD**

Editorial Panel

Authors : Mr. Parimal Patel
I/C Director,
Kyati School of Computer Application,
Ahmedabad

Ms. Pooja Gandhi
Assistant Professor
L. J. Institute of Computer Application,
Ahmedabad

Editor : Dr. Darshna Patel
Assistant Professor
Shri M. M. Patel Institute of Sciences and
Research, (M.Sc. IT Department),
Kadi Sarva Vishwavidyalaya,
Gandhinagar

Language Editor : Dr. Jagdish Vinayakrao Anerao
Associate Professor,
Smt A. P. Patel Arts And,
N. P. Patel Commerce College,
Ahmedabad.

ISBN 978-81-949223-8-4

Edition : 2020

Copyright © 2020 Knowledge Management and Research Organisation.

All rights reserved. No part of this book may be reproduced, transmitted or utilized in any form or by a means, electronic or mechanical, including photocopying, recording or by any information storage or retrieval system without written permission from us.

Acknowledgment

Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.

ROLE OF SELF-INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self-instructional material is completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self-instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual-skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behaviour should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminate interactive communication cues, such as pauses, intonation and gestures, associated with the face-to-face method of teaching. This is

particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self-Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self-instructional materials, be they print or otherwise. These materials are designed to achieve certain pre-determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included many tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

DIGITAL ELECTRONICS AND COMPUTER ORGANIZATION

Contents

BLOCK 1 : NUMBER SYSTEM

Unit 1 **NUMBER SYSTEM**

Introduction, Number System, Non-positional Number Systems, Positional Number Systems, Binary Numbers, Octal Numbers, Hexadecimal Numbers, Number System Conversions

Unit 2 **COMPUTER ARITHMETIC**

Introduction, Fractional Numbers, 9's and 10's Complement, 1's and 2's Complement, Representation of Negative Numbers

Unit 3 **CODES FOR CHARACTER REPRESENTATION**

Introduction, Binary Coded Decimal, Excess 3 Code, Gray Code

BLOCK 2 : BOOLEAN ALGEBRA

Unit 4 **LOGIC GATES**

Introduction, OR GATE, AND GATE, XOR GATE, NOT GATE, NAND GATE, NOR GATE, XNOR GATE

Unit 5 **INTRODUCTION BOOLEAN ALGEBRA**

Introduction, Boolean Laws and Theorems of Boolean Algebra, Boolean Identities, Boolean Algebraic Properties

Unit 6 **SIMPLIFICATION OF BOOLEAN ALGEBRA - I**

Introduction, De Morgan's Law

Unit 7 **SIMPLIFICATION OF BOOLEAN ALGEBRA - II**

Introduction, Truth Tables, Simplification of Boolean Equation using K-Map

BLOCK 3 : DIGITAL COMPONENT

Unit 8 ARITHMETIC LOGIC UNIT

Introduction, Construction of ALU, Adder, Binary Half Adder, Binary Full Adder, Parallel Binary Adder, Binary Adder–Subtractor, Addition in 1's and 2's Complement System

Unit 9 DIGITAL COMPONENT

Introduction, Integrated Circuits, Decoders and its Expansion, Encoders, Multiplexer and its Expansion, Memory Unit

Unit 10 ADDRESS, DATA & CONTROL BUS

Introduction, Address, Data & Control Bus, Bus System for 4–Bit Register, Three–State Bus Buffer

BLOCK 4 : INPUT/OUTPUT DEVICES AND FLIP FLOPS

Unit 11 ADDRESS, DATA & CONTROL BUS

Introduction, Input/Output Devices, Key Board, Mouse, Display Unit, Printer (Types), Scanner, OCR, OMR, MICR

Unit 12 INPUT/OUTPUT INTERFACE and DATA TRANSFER

Introduction, Input/Output Interface, Asynchronous Data Transfer and Mode of Data Transfer, Concept of Programmed I/O, DMA

Unit 13 MEMORY

Introduction, Memory Hierarchy, Primary Memory, RAM and Types of RAM, ROM and Types of ROM, Secondary Memory, Magnetic Disk, Magnetic Tape, Optical Memory (CDROM), Concept of Virtual Memory, Concept of Cache and Their Need

Unit 14 FLIP-FLOPS

Introduction, (SR, JK, D, T) its Truth-Tables, Applications of Flip-Flops, Clocks, 3-4-bit Registers, Shift Register, Synchronous/Asynchronous Binary Counters

Unit 15 CPU

Introduction, Functions of CPU, Register Classification and Organization, Instruction Cycle, Instruction Formats, Addressing Modes



Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-203/
DCAR-203

Digital Electronics and **Computer Organization**

BLOCK 1 : NUMBER SYSTEM

UNIT 1 INTRODUCTION TO NUMBER SYSTEM

UNIT 2 COMPUTER ARITHMETIC

UNIT 3 CODES FOR CHARACTER REPRESENTATION

NUMBER SYSTEM

Block Introduction :

The number system that is used in our day-to-day life is called The Decimal number system. In this system, the base is equal to 10 because there are altogether ten symbols or digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) used in this system. In the decimal system, the successive positions to the left of the decimal point represent units, tens, hundreds, thousands, etc.

The positional number system includes only the few symbols which are called as digits and these symbols represent different values depending on the position they occupy in the number. The value of every digit in such the number system is determined by three considerations.

There are two characteristics of all number systems that are recommended by the value of the base. In all the number systems, the value of the base represents the total number of different symbols or digits accessible in the number system. The first of these choices is constantly zero. The second attribute is that the highest value of the single digit is always equal to one less than the value of the base.

Maths rules are always based on the defining limits we place on the exacting numerical quantities dealt with. When we say that $2 + 2 = 4$ or $5 + 4 = 9$, we imply the use of integer quantities: The same types of numbers we all learn to count in basic education. Most people believe to be self-evident rules of arithmetic – valid at all times and for all purposes – It actually depend on what we define the number to be.

Block Objectives :

After learning this Block, you will be able to :

- Understand The number system
- Perform number conversion addition and subtraction
- Understand Fractional number representation in computer system
- Understand Negative number representation in computer system.

Block Structure :

Unit 1 : Introduction to Number System

Unit 2 : Computer Arithmetic

Unit 3 : Codes for Character Representation

UNIT STRUCTURE

- 1.0 Learning Objectives
- 1.1 Introduction
- 1.2 Number System
 - 1.2.1 Non-positional Number Systems
 - 1.2.2 Positional Number Systems
- 1.3 Binary Numbers
- 1.4 Octal Numbers
- 1.5 Hexadecimal Numbers
- 1.6 Number System Conversions
- 1.7 Let Us Sum Up
- 1.8 Suggested Answer for Check Your Progress
- 1.9 Glossary
- 1.10 Assignment
- 1.11 Activities
- 1.12 Case Study
- 1.13 Further Readings

1.0 Learning Objectives :

After learning this unit, you will be able to;

- Understand number system used in computer
- Understand conversion from one number system to another

1.1 Introduction :

In the computer system, data is stored in the format that cannot be simply understood by human beings. This is the cause why input and output (I/O) interfaces are essential. Each computer stores letters, numbers and the special characters in the coded form. Before going into the details of these codes, it is essential to have the basic understanding of the number system. It also introduces some of the commonly used number systems by computer professionals and the relationship between them.

This chapter deals with the basic fundamentals of number system, computer arithmetic and binary codes. Data in the computer is stored in the series of bits which are often interpreted in blocks known as bytes [8 bits] or words. The ALU (Arithmetic and Logic Unit) performs arithmetic operations on data and the control unit needs to 'decode' instructions. Both of these jobs are performed by logic circuits.

1.2 Number System :

There are two types of Number systems: non-positional and positional.

1.2.1 Non – Positional Number Systems :

In very early days humans used to count fingers and ten toes there were insufficient, stones, pebbles or sticks were used to indicate values. An additive approach of counting scheme or non-positional number system uses

In this system, the symbols for 2 second, 3 for third, 4 IIII, so for 5 IIIII, I for one are used as such. Each I for 1, II for 2, III for 3, IIII for 4 etc. are used. Symbol represents the same price regardless of their position in the number and value of symbols to explore the number of demands are added. With the number system that is very difficult to perform arithmetic, positional number system was developed as the centuries passed.

1.2.2 Positional Number Systems :

In the positional number system, there are only the few symbols called digits and these symbols stand for dissimilar values depending on the position they occupy in the number. The value of each digit in such the number system is resolute by three considerations.

1. The digit itself
2. The position of The digit in The number
3. The base of The number system (where base is defined as The total number of digits available in The number system)

The number system that is used in our everyday life is called The Decimal number system. In this system, The base is equivalent to 10 because There are in general ten symbols or digits (0,1,2,3,4,5,6,7,8,9) used in this system. In The decimal system, the successive positions to the left of the decimal point represent units, tens, hundreds, thousands, etc.

It may also be observed that the same digit signifies different values depending upon the position it occupies in the number. For example,

In 2586_{10} the digit 6 signifies $6 * 10^0 = 6$

In 2568_{10} the digit 6 signifies $6 * 10^1 = 60$

In 2658_{10} the digit 6 signifies $6 * 10^2 = 600$

In 6258_{10} the digit 6 signifies $6 * 10^3 = 6000$

Thus, using existing digits and arranging them in the variety of positions can represent any number. The principles that relate to the decimal system relate in any additional positional number system.

The price suggested by the number of systems that all have two properties. In all systems, the base number of the price offered in the system represents the total number of different symbols or marks. The first of these options is always zero.

❖ Decimal Number System :

The conventional number system used currently is the decimal number system. The decimal number system has the digits from 0 to 9 and the numbers have the base ten. The smallest number in this system is 0 and the largest number is 9. The decimal number fifty-eight is represented as $(58)_{10}$

Any decimal number can be converted to another base by dividing the given decimal number by the base to be converted to.

□ Check Your Progress – 1 :

1. Explain 'positional' and 'non-positional' number system ?

.....

2. Give example of Positional and Non-Positional Numbers.

.....

1.3 Binary Number System :

The binary number system can be taken just like the decimal system excluding that the base is 2 instead of 10. There are only two digits (0 and 1) that can be used in this number system. We should note the largest single digit is 1 (one less than the base). Here again, each position in the binary number represents the power of The base (2). In this system, the rightmost position is the units (2^0) position, the second position from the right is the 2's (2^1) position and scheduled in this way the third position is the 4's (2^2) position, the fourth position is the 8's (2^3) position and so on. Thus, the decimal correspondent of the binary number 10101 (written as 10101_2) is

$$(1 * 2^4) + (0 * 2^3) + (1 * 2^2) + (0 * 2^1) + (1 * 2^0)$$

Or $16 + 0 + 4 + 0 + 1$

Or 21

In order to be specific about which system is referred to, it is common practice to indicate the base as the subscript. Thus, the numbers are represented as:

$$10101_2 = 21_{10}$$

The common abbreviation bit often refers to 'binary digit'. Thus, the "bit" in computer technology means either the 0 or the 1. The binary number consisting of n bits is called an n-bit number. Table 1.1 lists all the 3-bit numbers along with their decimal equivalent. Remember that there are only two digits, 0 and 1, in the binary system and hence the binary equivalent of the decimal number 2 has to be stated as 10 (read as one, zero).

Another important point to note is that with 3 bits (positions), only 8 (2^3) different patterns of 0's and 1's are possible and from Table 1.1 it may be seen that the 3-bit number can have one of the 8 values in the range 0 to 7. In fact, it can be shown that any decimal number in the range 0 to 2^n-1 can be represented in the binary form as an n-bit number.

Table 1.1 : 3-bit Numbers with their Decimal Values

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

□ Check Your Progress – 2 :

1. Explain Decimal and Binary number system ?

.....

.....

.....

.....

.....

2. List all 4 bits binary numbers.

.....

.....

.....

.....

.....

1.4 Octal Number System :

Octal number system, based on symbols or marks are only 8, 8. The system is thus : (does not exist in the system 8 and 9) 0, 1, 2, 3, 4, 5, 6 and 7. Here, The largest single digit 7 (base of less). Then, in each case based on the actual number (8) represents the power. So, (written as $(2057)_8$) of the decimal equivalent of the octal number is 2057 :

$$(2 * 8^3) + (0 * 8^2) + (5 * 8^1) + (7 * 8^0)$$

Or $1024 + 0 + 40 + 7$

Or 1071

Therefore, $2057_8 = 1071_{10}$

Observe that since there are only 8 digits in the octal number system, thus 3 bits ($2^3 = 8$) are sufficient to represent any octal number in binary.

One more example of octal number is $(325)_8$:

$$(3 * 8^2) + (2 * 8^1) + (5 * 8^0)$$

Or $192 + 16 + 5$

Or $(213)_{10}$

Therefore, $2057_8 = 213_{10}$

Another example of octal number is 123 :

$$(1 * 8^2) + (2 * 8^1) + (3 * 8^0)$$

Or $64 + 16 + 3$

Or 83

Therefore, $123_8 = 83_{10}$

□ Check Your Progress – 3 :

1. Explain Octal Number System.

.....

.....

.....

.....

.....

1.5 Hexadecimal Number System :

In the hexadecimal number system, the base is 16. The base of 16 suggests choices of 16 single-character digits or symbols. The first 10 digits are the digits of the decimal system 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. The remaining six digits are represented by A, B, C, D, E and F representing the decimal values of 10, 11, 12, 13, 14 and 15 correspondingly.

In the hexadecimal number system, thus, the letters, A to F are used for digits 10 to 15 are number digits. The number that has the decimal equal value of 10; B has the value of 11 and A is the number that has decimal equal value of 10 so on. Thus, the largest single digit is F or 15 (one less than the base). Again, each position in the hexadecimal system represents the power of the base (16). Thus, the decimal equal of the hexadecimal number 1AF is 431.

$$(1 * 16^2) + (A * 16^1) + (F * 16^0)$$

Or $(1 * 256) + (10 * 16) + (15 * 1)$

Or $256 + 160 + 15$

Or 431

Thus, $1AF_{16} = 431_{10}$

Observe that since there are only 16 digits in the hexadecimal number system, 4 bits ($2^4 = 16$) are sufficient to represent any hexadecimal number in binary.

□ Check Your Progress – 4 :

1. Explain Hexadecimal Number System.

.....

.....

.....

.....

.....

1.6 Number System Conversions :

Numbers expressed in decimal are much more meaningful than are values uttered in any other number system. Decimal numbers are used in daily life because of the fact that normally occurs. The value of the number, any number of systems can be identified in any other number system.

Input and the final output values are in decimal, decimal number system and other computer professionals often vice versa are required to change the number. To change the number from one to the other can be used by several methods or techniques.

Convert decimal to another base

The following three phases of the 10 base price of any other number systems are used to convert

Step 1 : Each issue of the column (positional) price (in this case the number of points, depending on the system) Locate.

Step 2 : To be successful with points in columns (Step 1) get into the column values.

Step 3 : Step 2 equivalent value in decimal is the total amount of the proposed products

Example – 1 :

$$11001_2 = ?_{10}$$

Solution :

Step 1 : Determine column values

Column Number Column Value

(From right)

1. $2^0 = 1$
2. $2^1 = 2$
3. $2^2 = 4$
4. $2^3 = 8$
5. $2^4 = 16$

Step 2 : Multiply column values by corresponding column digits

16	8	4	2	1
*1	*1	*0	*0	*1
16	8	0	0	1

Step 3 : Sum the products

$$16 + 8 + 0 + 0 + 1 = 25$$

Hence, $11001_2 = 25_{10}$

Example – 2 :

$$4706_8 = ?_{10}$$

Solution :

Step 1 :

Column Number Column Value

(From right)

1. $8^0 = 1$
2. $8^1 = 8$
3. $8^2 = 64$
4. $8^3 = 512$

Step 2 :

512	64	8	1
*4	*7	*0	*6
2048	448	0	6

Step 3 :

$$2048 + 448 + 0 + 6 = 2502$$

Hence, $4706_8 = 2502_{10}$

Example – 3 :

$$1AC_{16} = ?_{10}$$

Solution :

$$\begin{aligned} 1AC_{16} &= 1 * 16^2 + A * 16^1 + C * 16^0 \\ &= 1 * 256 + 10 * 16 + 12 * 1 \\ &= 256 + 160 + 12 \\ &= 428_{10} \end{aligned}$$

Converting from the Base 10 to the New Base

This method is also called as division – remainder technique. The following four steps are used to convert the number from base 10 to the new base.

Step 1 : Divide the decimal number can be converted to the new base price.

Step 2 : Divide the quotient of the division by the new base.

Step 3 : New to The base number (left) step 3 as the next issue remaining records.

Repeat steps 3 and 4, recording remainders from right to left, until the quotient becomes zero in step 3. Note the last remainder therefore obtained will be the most significant digit (MSD) of the new base number.

Example – 4 :

$$25_{10} = ?_2$$

Solution :

Steps 1 & 2 : $25/2 = 12$ and remainder 1

Steps 3 & 4 : $12/2 = 6$ and remainder 0

Steps 5 & 6 : $6/2 = 3$ and remainder 0

Steps 7 & 8 : $3/2 = 1$ and remainder 1

Steps 9 & 10 : $1/2 = 0$ and remainder 1

**Digital Electronics and
Computer Organization**

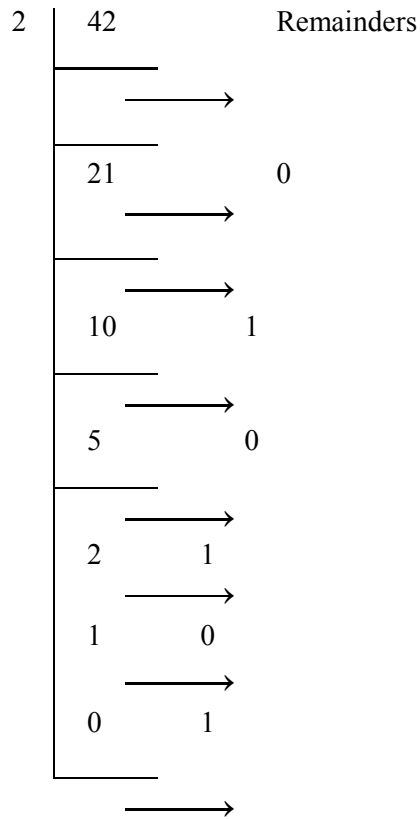
As mentioned in Steps 2 & 4, the remainders have to be in order in the reverse order so that the first remainder becomes the Least Significant Digit (LSD) and the last remainder becomes The Most Significant Digit (MSD).

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

Example – 5 :

$42_{10} = ?_2$

Solution :

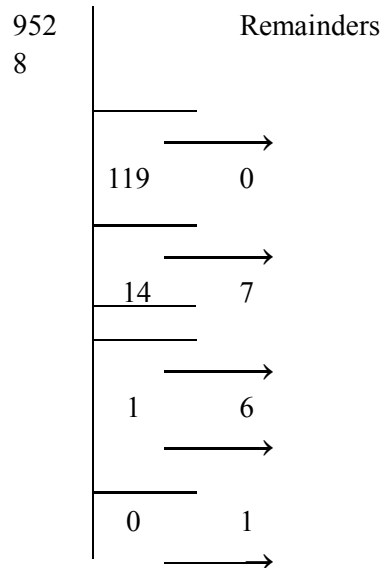


Hence, $42_{10} = 101010_2$

Example – 6 :

$952_{10} = ?_8$

Solution :



Hence, $952_{10} = 1670_8$

Example – 7 :

$$428_{10} = ?_{16}$$

Solution :

16	428	Remainders in hexadecimal
	26	12 = C
	1	10 = A
	0	1 = 1

Hence, $428_{10} = 1AC_{16}$

Converting from the base other than 10 to the base other than 10

The subsequent two steps are used to convert the number from the base other than **10 to the base other than 10**.

Step1 : Convert the original number to the decimal number (base 10).

Step2 : Convert the decimal number so obtained to the new base.

Example – 8 :

$$101110_2 = ?_8$$

Solution :

Step1 : Convert 101110_2 to base 10

$$\begin{aligned} 101110_2 &= 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0 \\ &= 32 + 0 + 8 + 4 + 2 + 0 \\ &= 46_{10} \end{aligned}$$

Step2 : Convert 46_{10} to base 8.

8	46	Remainders
	5	6
	0	5

Hence, $46_{10} = 56_8$

So, $101110_2 = 46_{10} = 56_8$

Thus, $101110_2 = 56_8$

Shortcut methods for conversions

There are shortcut methods of conversion from one base to another. The octal number can be converted to hexadecimal format without being converted to decimal and then being divided by 16. Each digit of the octal number are split and represented by equivalent binary numbers of 3 bits each. And the binary number can then be grouped into 4 bits each for representing the hexadecimal number. The shortcut methods of conversion are as follows:

Binary to Octal

The subsequent steps are used in this method :

Step 1 : Divide the binary digits into groups of three (initial from the right).

Step 2 : Convert each group of three binary digits into one octal digit.

Since decimal digits 0 to 7 are equal to octal digits 0 to 7 so binary to decimal conversion can be used in this step.

Example – 9 :

$$101110_2 = ?_8$$

Solution :

Step 1 : Divide the binary digits into groups of 3 starting from right (LSD)

$$\underline{101} \quad \underline{110}$$

Step 2 : Convert each group into one digit of octal (use binary-to-decimal conversion)

$$\begin{aligned} 101_2 &= 1 * 2^2 + 0 * 2^1 + 1 * 2^0 \\ &= 4 + 0 + 1 \\ &= 5_8 \end{aligned}$$

$$\begin{aligned} 110_2 &= 1 * 2^2 + 1 * 2^1 + 0 * 2^0 \\ &= 4 + 2 + 0 \\ &= 6_8 \end{aligned}$$

Therefore, $101110_2 = 56_8$

Octal to Binary

The subsequent steps are used in this method:

Step 1 : Each octal digit to convert the three –digit binary number (octal digits for the conversion can be represented as decimal)

Step 2 : The out coming binary groups combined in the single binary number (3 points each) are covered

Example – 10 :

$$562_8 = ?_2$$

Solution :

Step 1 : Convert each octal digit to 3 binary digits.

$$5_8 = 101_2$$

$$6_8 = 110_2$$

$$2_8 = 010_2$$

Step 2 : Combine the binary groups.

$$562_8 = \underline{101110010}$$

5 6 2

Hence, $562_8 = 101110010_2$

Example – 11 :

$$6751_8 = ?_2$$

Solution :

$$\begin{aligned} 6751_8 &= \underline{110111101001} \\ &\quad \quad \quad 6 \quad 7 \quad 5 \quad 1 \\ &= 110111101001_2 \end{aligned}$$

Hence, $6751_8 = 110111101001_2$

Binary to Hexadecimal

The following steps are followed under this method :

Step 1 : (Starting from right) Divide in groups of four binary digits

Step 2 : Each group of four binary digits in the hexadecimal digits converted. (0–9 hexadecimal digits hexadecimal digits 0 to 9 and decimal points are equal. F. for 10 to 15 F for the decimal must be represented as hexadecimal values are equivalent to 10 to 15 decimal digits).

Example – 12 :

$$11010011_2 = ?_{16}$$

Solution :

Step 1 : Divide the binary digits into groups of 4.

$$\underline{1101 \ 0011}$$

Step 2 : Convert each group of 4 binary digits to 1 hexadecimal digit.

$$\begin{aligned} 1101_2 &= 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 \\ &= 8 + 4 + 0 + 1 \\ &= 13_{10} \\ &= D_{16} \end{aligned}$$

$$\begin{aligned} 0011_2 &= 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 \\ &= 0 + 0 + 2 + 1 \\ &= 3_{16} \end{aligned}$$

Hence, $11010011_2 = D3_{16}$

Example – 13 :

$$10110101100_2 = ?_{16}$$

Solution :

$$\begin{aligned} 10110101100_2 &= \underline{010110101100} \\ &\quad \text{(Group 4 digits from right)} \\ &= 5AC \\ &\quad \text{(Convert each group to the hexadecimal digit)} \end{aligned}$$

Hence, $10110101100_2 = 5AC_{16}$

Hexadecimal to Binary

The following steps are used in this method:

Step 1 : Convert the decimal equivalent of hexadecimal digit to 4 bit binary number.

Step 2 : Combine all the out coming binary groups (of 4 digits each) into single binary number.

Example – 14 :

$$2AB_{16} = ?_2$$

Solution :

Step 1 : Convert the decimal equivalent of each hexadecimal digit into 4 binary digits.

$$2_{16} = 2_{10} = 0010_2$$

$$A_{16} = 10_{10} = 1010_2$$

$$B_{16} = 11_{10} = 1011_2$$

Step 2 : Combine the binary groups

$$2AB_{16} = \underline{0010101011}$$

2 the B

$$\text{Hence, } 2AB_{16} = 0010101011_2$$

Example – 15 :

$$ABC_{16} = ?_2$$

Solution :

$$ABC_{16} = \underline{101010111100}$$

the B C

$$= 101010111100_2$$

$$\text{Hence, } ABC_{16} = 101010111100_2$$

**Table 1.2 : Relationship between Decimal, Hexadecimal,
Binary and Octal Number Systems**

Decimal	Hexa Decimal	Binary	Octal
0	0	0	0
1	1	1	1
2	2	10	2
3	3	11	3
4	4	100	4
5	5	101	5
6	6	110	6
7	7	111	7
8	8	1000	10
9	9	1001	11
10	A	1010	12
11	B	1011	13
12	C	1100	14
13	D	1101	15
14	E	1110	16
15	F	1111	17

Finally, the above Table 1.2 summarises the relationship between the decimal, binary, hexadecimal and octal number systems. Note that the maximum value for the single digit of octal (7) is equal to the maximum value of three digits of binary. The value range of one digit of octal duplicates the value range of three digits of binary. If octal digits are substituted for binary digits, the substitution is on the one-to-three basis.

Similarly, note that the maximum value of one digit in hexadecimal is equal to the maximum value of four digits in binary. Thus, the value range of one digit of hexadecimal is equivalent to the value range of four digits of binary. Therefore, hexadecimal shortcut notation is the one-to-four reduction in the space and time required for memory dump.

❑ Check Your Progress – 5 :

1. Explain Number system conversion from binary to decimal, hexadecimal and octal with example.

.....
.....
.....
.....
.....

2. The decimal equivalent of 1000 is _____
(a) 2 (b) 4 (c) 6 (d) 8
3. The binary number 1100 is equal to _____ decimal number.
(a) 9 (b) 10 (c) 12 (d) 0
4. Convert in to decimal : $(214)_8 = ?$
(a) 140 (b) 141 (c) 142 (d) 130
5. The octal number equivalent of 110110 is _____
(a) 66 (b) 88 (c) 77 (d) 55
6. The binary number 11001010 is equal to _____ hexadecimal number.
(a) CA (b) AB (c) 1A (d) B1
7. The Hexadecimal representation of 1110 is _____
(a) 0111 (b) E (c) 15 (d) 14

1.7 Let Us Sum Up :

Number systems are of two types–non positional and positional. In the non–positional number system, every symbol determines the same value regardless of its position in The number and to find the value of the number, one has to count the number of symbols present in The number.

Some positional number system that are used in computer design and by computer professionals are binary, octal and hexadecimal.

1.8 Answer for Check Your Progress :

- ❑ **Check Your Progress 1 :**
See Section 1.2.2
- ❑ **Check Your Progress 2 :**
See Section 1.3
- ❑ **Check Your Progress 3 :**
See Section 1.4
- ❑ **Check Your Progress 4 :**
See Section 1.5

❑ **Check Your Progress 5 :**

- | | | |
|---------------------|-------|-------|
| 1 : See Section 1.6 | 2 : D | 3 : C |
| 4 : D | 5 : A | 6 : A |
| 7 : A | | |

1.9 Glossary :

1. **Non-Positional Number Systems** – In early days, human beings counted on fingers. when ten fingers were not adequate, he made use of pebbles, stones etc. to indicate values. This method of counting uses and preservative approach or the non-positional number system.
2. **Positional Number Systems** – In the positional number system, called points are only the few symbols and the symbols they correspond to different values depending on the state in the number. The value of each digit in such the number system is determined by three considerations.
3. **Decimal Number System** – The conventional number system used currently is the decimal number system. The decimal number system has the digits from 0 to 9 and the numbers have the base ten.

1.10 Assignment :

1. What is the bit in computer terminology ? How many different patterns of bits are possible with it ?
5 bits
6 bits
7 bits
8 bits
2. What will be the total number of different symbols or digits and the maximum value of the single digit for the following number systems ?
Number system with base 5 Number system with base 20 Number system with base 9 Number system with base 12

1.11 Activities :

What is the difference between positional and non-positional number system.

Write examples of both types of number system.

1.12 Case Study :

What is the base of the number system? Write examples to illustrate the role of base in positional number systems

1.13 Further Reading :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar

UNIT STRUCTURE

- 2.0 Learning Objectives
- 2.1 Introduction
- 2.2 Fractional Numbers
- 2.3 9's and 10's Complement
- 2.4 1's and 2's Complement
- 2.5 Representation of Negative Numbers
- 2.6 Let Us Sum Up
- 2.7 Suggested Answer for Check Your Progress
- 2.8 Glossary
- 2.9 Assignment
- 2.10 Activities
- 2.11 Case Study
- 2.12 Further Readings

2.0 Learning Objectives :

After learning this unit, you will be able to :

- Perform Fractional number representation
- Represent negative numbers
- Perform subtraction
- Explain 1's complement
- Discuss 2's complement

2.1 Introduction :

In the computer system, data is stored in the format that cannot be simply understood by human beings. This is the cause why input and output (I/O) interfaces are essential. Each computer stores letters, numbers and the special characters in the coded form. Before going into the details of these codes, it is essential to have the basic understanding of the number system. It also introduces some of the commonly used number systems by computer professionals and the relationship between them.

This chapter deals with the basic fundamentals of number system, computer arithmetic and binary codes. Data in the computer is stored in the series of bits which are often interpreted in blocks known as bytes [8 bits] or words. The ALU (Arithmetic and Logic Unit) performs arithmetic operations on data and the control unit needs to 'decode' instructions. Both of these jobs are performed by logic circuits.

2.2 Fractional Numbers :

In binary number system, fractional numbers are formed in the same general way as in the decimal system. The fraction in decimal number system can be represented as :

$$0.235 = (2 \times 10^{-1}) + (3 \times 10^{-2}) + (5 \times 10^{-3}) \text{ And}$$

$$68.53 = (6 \times 10^1) + (8 \times 10^0) + (5 \times 10^{-1}) + (3 \times 10^{-2})$$

Similarly in the binary system,

$$0.101 = (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$$

And

$$10.01 = (1 \times 2^1) + (0 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2})$$

Therefore, the binary point as the decimal point serves the same purpose.

Some of positional values in binary systems are accurate.

Binary Point

Position	4	3	2	1	0	-1	-2	-3	-4
Value	2⁴	2³	2²	2¹	2⁰	2⁻¹	2⁻²	2⁻³	2⁻⁴
Represented	16	8	4	2	1	1/2	1/4	1/8	1/16

In common, the number in the number system with base b would be written as :

$$a_n \ a_{n-1} \ \dots \ a_1 \ a_0 \ a_{-1} \ a_{-2} \ \dots \ a_{-m}$$

and would be interpreted to mean

$$a_n \times b^n + a_{n-1} \times b^{n-1} + \dots + a_0 \times b^0 + a_{-1} \times b^{-1} + a_{-2} \times b^{-2} + \dots + a_{-m} \times b^{-m}$$

The symbols $a_n \ a_{n-1}, \dots, a_{-m}$ used in The above representation should be one of The symbols allowed in The number system. Thus, as per the above mentioned general rule,

$$46.328 = (4 \times 8^1) + (6 \times 8^0) + (3 \times 8^{-1}) + (2 \times 8^{-2})$$

and

$$5A.3C_{16} = (5 \times 16^1) + (A \times 16^0) + (3 \times 16^{-1}) + (C \times 16^{-2})$$

Example – 1 :

Find the decimal equivalent of the binary number 110.101

Solution :

$$\begin{aligned} 110.101_2 &= 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} \\ &\qquad\qquad\qquad + 1 \times 2^{-3} \\ &= 4 + 2 + 0 + .5 + 0 + .125 \\ &= 6 + 0.5 + 0.125 \\ &= 6.625_{10} \end{aligned}$$

Example – 2 :

Find the decimal equivalent of the octal number 127.54

Solution :

$$\begin{aligned}
 127.548 &= 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 5 \times 8^{-1} + 4 \times 8^{-2} \\
 &= 64 + 16 + 7 + 5/8 + 4/64 \\
 &= 87 + 0.625 + 0.0625 \\
 &= 87.6875_{10}
 \end{aligned}$$

Example – 3 :

Find the decimal equivalent of the hexadecimal number 2B.C4

Solution :

$$\begin{aligned}
 2B.C4_{16} &= 2 \times 16^1 + B \times 16^0 + C \times 16^{-1} + 4 \times 16^{-2} \\
 &= 32 + 11 + C/16 + 4/256 \\
 &= 43 + 0.75 + 0.015625 \\
 &= 43.765625_{10}
 \end{aligned}$$

□ Check Your Progress – 1 :

1. Explain 'Fractional Numbers'

.....

.....

.....

.....

.....

2.3 9's and 10's Complement :

The 9's complement is used to find the subtraction of the decimal numbers. The 9's complement of a number is calculated by subtracting each digit of the number by 9. 9's and 10's complement in the decimal number system are dealt with for better understanding.

9's complement

To obtain 9's complement of the decimal number each digit of the number is subtracted from 9. For example, 9's complement of 45 is $(99 - 45) = 54$

9's complement of 523 = $(999 - 523) = 476$

The 10's complement is also used to find the subtraction of the decimal numbers. The 10's complement of a number is calculated by subtracting each digit by 9 and then adding 1 to the result. Simply, by adding 1 to its 9's complement we can get its 10's complement value.

10's complement

10's complement of the decimal number = its 9's complement + 1.

10's complement of 45 = $54 + 1 = 55$

10's complement of 523 = $476 + 1 = 477$

On adding the number 45 and its 10's complement, the sum becomes zero (omitting the carry over)

Example – 4 :

$$\begin{array}{r} 4 \quad \quad 5 \quad \text{(decimal number)} \\ + \quad 5 \quad 5 \quad \text{(its 10's complement)} \\ \hline 0 \quad \quad 0 \end{array}$$

Carry = 1

In this case the decimal number is of two digits. If the sum of the number and its 10's complement only up to two digits are considered, the sum becomes zero. Similarly,

Example – 5 :

$$\begin{array}{r} \quad 5 \quad 2 \quad 3 \quad \text{(decimal number)} \\ + \quad 4 \quad 7 \quad 7 \quad \text{(its 10's complement)} \\ \hline 0 \quad 0 \quad 0 \end{array}$$

Carry = 1

The decimal number 523 is of three digits. Considering The sum of the number and its 10's complement only up to three digits, the sum becomes zero. Thus it is concluded that 10's complement gives the negative value of the number

10's complement of the decimal number = – decimal number 1's Complement

□ Check Your Progress – 2 :

- 1. Explain 9's and 10's complement

.....
.....
.....
.....
.....

- 2. Find 9's Complement of 67.

.....
.....
.....
.....
.....

- 3. Find 10's Complement of 67.

.....
.....
.....
.....
.....

2.4 1's and 2's Complement :

1's complement in the binary number system is similar to The 9's complement in the decimal system. 1's complement of the binary number is obtained by subtracting each bit of the number from 1. 1's complement of 01 is 10. 1's complement of 111 is 000. Thus, 1's complement of the binary number can be obtained by simply changing bit 1 to 0 and 0 to 1.

Example – 6 :

Find 1's complement of 100110

$$1's \text{ complement of } 100110 = 011001$$

Example – 7 :

Find 1's complement of 0000

$$1's \text{ complement of } 0000 = 1111$$

Example – 8 :

Find 1's complement of 11111

$$1's \text{ complement of } 11111 = 00000$$

2's Complement.

2's complement in the binary number system is similar to 10's complement in the decimal number system. 2's complement of binary number = its 1's complement + 1.

Example – 9 :

Find 2's complement of 10011

$$2's \text{ complement of } 10011 = 01100 + 1 = 01101$$

Example – 10 :

Find 2's complement of 111.

$$2's \text{ complement of } 111 = 000 + 1 = 001$$

Example – 11 :

Find 2's complement of 0000

$$2's \text{ complement of } 0000 = 1111 + 1 = 0000$$

Consider the case of adding the binary number to its 2's complement.

Example – 12 :

Binary number = 1001

Its 1's complement = 0110

Its 2's complement = 0110 + 1 = 0111.

$$\begin{array}{rcccc}
 \text{Number + its 2's complement} & = & 1 & 0 & 0 & 1 \\
 + & & 0 & 1 & 1 & 1 \\
 \hline
 & & 0 & 0 & 0 & 0
 \end{array}$$

Carry =1

The last carry is lost if the processor is of 4 bits or only 4 bits sum is considered. For 8-bit processor the number and 2's complement will be written in 8 bits as given below :

$$\begin{array}{rcl}
 \text{Number} & = & 00001001 \\
 \text{1's complement} & = & 11110110 \\
 \text{2's complement} & = & 11110110 + 1 \\
 & = & 11110111 \\
 \text{Number} & = & 00001001 \\
 + \text{ 2's complement} & = & 11110111 \\
 \hline
 & & 00000000 \\
 & \nearrow & \text{Carry} = 1
 \end{array}$$

The last carry will be neglected, if The 8 bit sum is considered.

Example – 13 :

$$\begin{array}{rcl}
 + 4 \text{ (decimal)} & = & 00000100 \quad \text{(binary)} \\
 -4 = \text{Its 2's complement} & = & 11111011 + 1 \\
 & = & 11111100 \\
 +4 & = & 00000100 \\
 -4 & = & 11111100 \\
 \hline
 & & 00000000
 \end{array}$$

Example – 14 :

Addition

Add + 5 and -7

$$\begin{array}{rcl}
 7 & = & 00000111 \\
 -7 & = & 11111000 + 1 \\
 & = & 11111001 \\
 +5 & = & 00000101 \\
 -7 & = & 11111001 \\
 \hline
 -2 & = & 11111110 \\
 \text{Check : } 2 & = & 00000010 \\
 -2 & = & 11111101 + 1 \\
 & = & 11111110
 \end{array}$$

Thus, 2's complement of the binary number represents its negative.

Binary Subtraction Using 2's Complement

Addition of 2's complement of the number is equivalent to the subtraction of the number.

Suppose, 0010 (2 decimal) has to be subtracted from 0101 (5 decimal). If the 2's complement of 0010 (2 decimal) is added to 0101 (5 decimal) the sum will be 0011 (3 decimal). It is equal to 0101 (5 decimal) – 0010 (2 decimal) = 0011 (3 decimal).

Ordinary binary subtraction :

$$\begin{array}{rcl}
 0101 & (5 \text{ decimal}) & \\
 - 0010 & (-2 \text{ decimal}) & \\
 \hline
 0011 & (3 \text{ decimal}) &
 \end{array}$$

Subtraction using 2's complement :

$$\begin{array}{r}
 1's \text{ complement of } 0010 \text{ (2 decimal)} = 1101 \\
 2's \text{ complement of } 0010 = 1101 + 1 = 1110 \\
 \quad 0101 \text{ (5 decimal)} \\
 + \quad 1110 \text{ (+ 2's complement of 2)} \\
 \hline
 \quad 0011 \text{ (3 decimal)}
 \end{array}$$

The carry of the last stage is neglected.

□ Check Your Progress – 3 :

1. Explain 2's complement using example.

.....

.....

.....

.....

.....

2.5 Representation of Negative Numbers :

2's complement is used to represent the negative of the binary number.

For example to represent -4, First write binary representation of 4, so it is 00000100. Its 1's complement is 11111011. Then add 1 to resultant number to find 2's complement.

$$11111011 + 1 = 11111100$$

Example – 15 :

$$\begin{array}{r}
 + 4 \text{ (decimal)} = 00000100 \text{ (binary)} \\
 -4 = \text{Its 2's complement} = 11111011 + 1 \\
 \quad \quad \quad \quad \quad = 11111100 \\
 +4 \quad \quad \quad \quad \quad = 00000100 \\
 -4 \quad \quad \quad \quad \quad = 11111100 \\
 \hline
 \quad \quad \quad \quad \quad 00000000
 \end{array}$$

Example – 16 :

Addition

Add + 5 and -7

$$\begin{array}{r}
 7 = 00000111 \\
 -7 = 11111000 + 1 \\
 \quad = 11111001 \\
 +5 = 00000101 \\
 -7 = 11111001 \\
 \hline
 -2 = 11111110 \\
 \text{Check : } 2 = 00000010 \\
 \quad -2 = 11111101 + 1 \\
 \quad \quad = 11111110
 \end{array}$$

Thus, 2's complement of the binary number represents its negative.

Binary Subtraction Using 2's Complement

Addition of 2's complement of the number is equivalent to the subtraction of the number. Suppose, 0010 (2 decimal) has to be subtracted from 0101 (5 decimal). If the 2's complement of 0010 (2 decimal) is added to 0101 (5 decimal) the sum will be 0011 (3 decimal). It is equal to 0101 (5 decimal) - 0010 (2 decimal) = 0011 (3 decimal).

Ordinary binary subtraction :

$$\begin{array}{r} 0101 \quad (5 \text{ decimal}) \\ -0010 \quad (-2 \text{ decimal}) \\ \hline 0011 \quad (3 \text{ decimal}) \end{array}$$

Subtraction using 2's complement:

$$\begin{array}{r} 1's \text{ complement of } 0010 \text{ (2 decimal)} = 1101 \\ 2's \text{ complement of } 0010 = 1101 + 1 = 1110 \\ \begin{array}{r} 0101 \text{ (5 decimal)} \\ + 1110 \text{ (+ 2's complement of 2)} \\ \hline 0011 \text{ (3 decimal)} \end{array} \end{array}$$

The carry of the last stage is neglected.

□ Check Your Progress – 4 :

1. Explain how to represent negative numbers in binary system ?
.....
.....
.....
.....
.....
2. Subtract 0011 from 1110 using ordinary and binary subtraction.
.....
.....
.....
.....
3. 1's complement of 1011101 is ?
(a) 0101110 (b) 1001101 (c) 0100010 (d) 1100101
4. 2's complement of 11001011 is _____ ?
(a) 01010111 (b) 11010100 (c) 00110101 (d) 11100010
5. 9's complement of 33 is _____ ?
(a) 99 (b) 66 (c) 44 (d) 33
6. 2's complement of 11 is _____ ?
(a) 00 (b) 01 (c) 11 (d) 10

7. 10's complement of 3336 is _____ ?
 (a) 6664 (b) 6663 (c) 6666 (d) 3336

2.6 Let Us Sum Up :

Fractional numbers are formed in the same way as in decimal number system. In general the number in the number system having base b would be express as : $a_n a_{n-1} \dots a_1 a_0 a_{-1} a_{-2} \dots a_{-m}$

And would be interpreted to mean

$$a_n \times b^n + a_{n-1} \times b^{n-1} + \dots + a_0 \times b^0 + a_{-1} \times b^{-1} + a_{-2} \times b^{-2} \dots + a_{-m} \times b^{-m}$$

The symbols $a_n a_{n-1}, \dots, a_{-m}$ used in the above depiction should be one of the symbols approved in the number system. Thus, as per the above mentioned general rule the techniques for following types of conversions to convert numbers from one base to another base should be known.

Converting from another base to decimal

Converting from decimal to another base

The negative numbers in binary system are represented using 2's complement.

2.7 Answer for Check Your Progress :

Check Your Progress 1 :

See Section 2.2

Check Your Progress 2 :

See Section 2.3

Check Your Progress 3 :

See Section 2.4

Check Your Progress 4 :

- | | |
|---------------------|------------------------|
| 1 : See Section 2.5 | 2 : See Section 2.5 |
| 3 : C | 4 : C 5 : B |
| 6 : B | 7 : A |

2.8 Glossary :

- 1's complement :** It is used for binary system. In this 0 is replaced by 1 and 1 is replaced by 0.
- 2's complement :** It is used to represent negative numbers in binary system. If we add 1 to 1's complement of any number it gives 2's complement.

2.9 Assignment :

- Find 9's complement of 56.
- Find 10's complement of 56.
- Find 1's complement of 0001
- Find 2's complement of 0001
- Compare 10's complement with 2's complement.

2.10 Activities :

In which case we can apply 1's complement ? Any alternative method exists which can use to represent 2's complement?

2.11 Case Study :

List all applications of 2's complement

2.12 Further Reading :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar

UNIT STRUCTURE

- 3.0 Learning Objective
- 3.1 Introduction
- 3.2 Binary Coded Decimal
- 3.3 Excess 3 Code
- 3.4 Gray Code
- 3.5 Let Us Sum Up
- 3.6 Answers for Check Your Progress
- 3.7 Glossary
- 3.8 Assignment
- 3.9 Activities
- 3.10 Case studies
- 3.11 Further Readings

3.0 Learning Objectives :

After learning this unit, you will be able to :

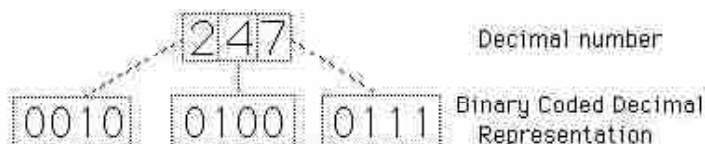
- Define BCD
- Define Excess-3 Code
- Define Gray code

3.1 Introduction :

Electronic systems use signals have two distinct values and circuit element have two stable states. There is a direct analogy among signals and binary digits. In this section we will focus on some different representation of binary system like BCD, Excess 3 code and Gray code which used in computer system architecture.

3.2 Binary Coded Decimal :

One of The most widely used representations of numerical data is the binary coded decimal (BCD) form, in which the 4-bit binary number represents each integer of the decimal number. It is particularly useful for the driving of display devices where the decimal output is desired. BCD usually refers to such coding in which binary digits have their normal values, that is, 8421. Sometimes it is written '8421 BCD' to clearly distinguish it from other binary codes such as The 4221 Code, but when BCD is used without qualification, The 8421 version is assumed.



Binary Coded Decimal

The following table gives the decimal number along with Their BCD codes.

Table 3.1 : Decimal to Standard BCD Conversion

Decimal Number	Standard BCD Number
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	00010000
11	00010001
12	00010010
13	00010011
14	00010100
30	00110000
45	01000101
64	01100100
73	01110011
98	10011000

❖ **Alphanumeric Coding :**

For the inherently binary world of the computer, it is necessary to put all symbols, letters, numbers, etc. into binary form. The most commonly used alphanumeric code is the ASCII code, with others like the EBCDIC code being applied in some communication applications.

□ **Check Your Progress – 1 :**

1. Explain Binary Coded Decimal with example.

.....

.....

.....

.....

.....

3.3 Excess-3 Code :

The excess-3 code (or XS3) is a non-weighted code used to express code used to direct decimal numbers. It is a self-complementary binary coded decimal (BCD) code which has biased representation. It is particularly significant for arithmetic operations as it overcomes inadequacy encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9.

❖ **Representation of Excess-3 Code :**

Excess-3 codes are unweighted and can be obtained by adding 3 to each decimal digit then it can be represented by using 4 bit binary number for each digit. An Excess-3 equivalent of a given binary number is obtained using the following steps:

- Find the decimal equivalent of the given binary number.
- Add +3 to each digit of decimal number.
- Convert the newly obtained decimal number back to binary number to get required excess-3 equivalent.

You can add 0011 to each four-bit group in binary coded decimal number (BCD) to get desired excess-3 equivalent.

These are following excess-3 codes for decimal digits –

Decimal	BCD Code	Excess 3 Code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0010	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Note : codes 0000 and 1111 are not used for any digit.

Example – 1 : Convert decimal number 23 to Excess-3 code.

So, according to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

$= 23 + 34 = 56 = 01010110$ which is required excess-3 code for given decimal number 23.

Example – 2 : Convert decimal number 15.46 into Excess-3 code.

According to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

$= 15.46 + 33.33 = 48.79 = 0100 1000.0111 1001$ which is required excess-3 code for given decimal number 15.46.

□ **Check Your Progress – 2 :**

1. Explain Excess-3 code on detail. Explain the issues with BCD which can be overcome with Excess-3 code.

.....

.....

.....

.....

.....

3.4 Gray Code :

The reflected binary or Gray Code is shown in Table 3.2. In this code only one bit changes in the transition from one number to the next higher number. The Gray code is used in shaft encoder, which is to indicate the angular position of the shaft. The use of Gray code reduces errors.

Suppose that the present position of the shaft is indicated by Gray code 0100, which is for 7. If the position changes to 8, the Gray code will be 1100. If the detector does not pick up the encoder the new change in binary bit will show the previous position that is 7. However, in case of ordinary binary code, 7 is represented by 0111 and 8 by 1000.

Now suppose the detector picks up the least significant bits, that is, 000, but fails to pick up the most significant bit 1, the output will be 0000 instead of 1000. So there is the large error if the simple binary code is used for the shaft encoder.

Table 3.2 : Gray Code

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

□ Check Your Progress – 3 :

1. Explain Gray Code.

.....
.....
.....
.....
.....

2. Why do we use gray codes ?

- (a) To count the no of bits changes
- (b) To rotate a shaft
- (c) Error correction
- (d) To count the no of bits changes and To rotate a shaft
- (e) None of the Mentioned

3. The binary representation of BCD number 00101001 (decimal 29) is _____

- (a) 00011101 (b) 0110101 (c) 0000000 (d) 11111110
- (e) None of the Mentioned

4. A binary digit is called a _____

- (a) bit (b) byte (c) number (d) All

5. The gray code equivalent of (1011) is _____

- (a) 1101 (b) 1010 (c) 1110 (d) 1111
- (e) None of above

6. 85 in BCD code is _____

- (a) 1000 1100 (b) 1101 1010 (c) 1000 0101 (d) 1101 1001
- (e) all of above

7. The binary equivalent of gray code 1110 is _____

- (a) 1101 (b) 1011 (c) 1110 (d) 1111

8. The binary equivalent of gray code 1111 is _____

- (a) 1101 (b) 1011 (c) 1110 (d) 1010

3.5 Let Us Sum Up :

The different character representation codes we have reviewed in this section. They are very useful in computer architecture.

Machines can understand binary number system, some machines understand hexadecimal number system. But in case of signal and digital circuit binary number system is suitable as they have the same scenario like on –off ,0–1 and true–false. When dealing with binary number system, machines sometimes faces some issues. To overcome these some more codes are introduced, they are using only binary system that is 0 and 1 but representation is quite different.

3.6 Answer for Check Your Progress :

❑ **Check Your Progress 1 :**

See Section 3.2

❑ **Check Your Progress 2 :**

See Section 3.3

❑ **Check Your Progress 3 :**

1 : See Section 3.4	2 : C	3 : A
4 : A	5 : C	6 : C
7 : B	8 : D	

3.7 Glossary :

1. **BCD** : The binary-coded decimal (BCD) is an encoding for decimal numbers in which each digit is represented by its own binary sequence.
2. **Gray Code** : Gray code is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit)

3.8 Assignment :

1. Explain the difference among BCD, Excess 3 code and Gray code with example.
2. Generate Gray code for $n = 1$ bit, $n = 2$ bits and $n = 3$ bits.

3.9 Activities :

1. Where is BCD code used ?
2. Where Gray Codes are used ?
3. Where is Excess 3 code used ?

3.10 Case Study :

Refer the format of date and time in BIOS of Some PCs and list the names of those PCs.

3.11 Further Reading :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar

BLOCK SUMMARY :

Number systems are of two types—non positional and positional. In the non-positional number system, every symbol represents the same value apart from of its position in the number and to find the value of the number, one has to count the number of symbols present in the number. It is enormously difficult to perform arithmetic with such the number system. In the positional number system there are only the little symbols called digits. These symbols represent different values depending on the position they occupy in the number. The value of every digit in such the number is determined by three considerations.

- a. The digit itself
- b. The position of the digit in the number
- c. The base of the number system

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Explain 2's complement ?
2. What is BCD ?
3. What is Gray code ? Which are applications of gray code ?
4. Explain Hexadecimal number system .List applications of it.

❖ **Long Questions :**

1. What do you understand by number system, explain in brief ?
2. Explain the examples of 'Fractional Numbers'.
3. Write difference between Positional and non positional number system.
4. How negative fractional number are represented in Binary ?

❖ **Enrolment No. :**

1. How many hours did you need for studying the units ?

Unit No.	1	2	3
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Language and Style	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Illustration used (Diagram, tables etc)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Conceptual Clarity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Check your progress Quest	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Feed back to CYP Question	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....



Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-203/
DCAR-203

Digital Electronics and **Computer Organization**

BLOCK 2 : BOOLEAN ALGEBRA

UNIT 4 LOGIC GATES

UNIT 5 INTRODUCTION BOOLEAN ALGEBRA

UNIT 6 SIMPLIFICATION OF BOOLEAN ALGEBRA – I

UNIT 7 SIMPLIFICATION OF BOOLEAN ALGEBRA – II

BOOLEAN ALGEBRA

Block Introduction :

The Boolean algebra has found one of the most sensible use in the generalization of logic circuits. If we interpret the logic circuit's job into symbolic (Boolean) form and apply certain algebraic rules to the out coming equation to reduce the number of terms and/or arithmetic operations, the simplified equation may be interpreted back into circuit form for the logic circuit performing the same job with less components. If equivalent job may be achieved with less components, the outcome will be increased reliability and decreased cost of manufacture.

Logic gates are electronic components which are used to are able to perform complex logical and arithmetic operations.

K-Map is a tool which is used to represent Boolean function and also useful to minimize Boolean equation.

Block Objectives :

After learning this Block, you will be able to :

- Understand The Boolean algebra
- Understand the different types of logic gates
- Understand how K map is used to represent Boolean equation
- Understand how k map is used to minimize Boolean equations

Block Structure :

Unit 4 : Logic Gates

Unit 5 : Introduction Boolean Algebra

Unit 6 : Simplification of Boolean Algebra – I

Unit 7 : Simplification of Boolean Algebra – II

UNIT STRUCTURE

- 4.0 Learning Objectives
- 4.1 Introduction
- 4.2 OR GATE
- 4.3 AND GATE
- 4.4 XOR GATE
- 4.5 NOT GATE
- 4.6 NAND GATE
- 4.7 NOR GATE
- 4.8 XNOR GATE
- 4.9 Let Us Sum Up
- 4.10 Suggested Answer for Check Your Progress
- 4.11 Glossary
- 4.12 Assignment
- 4.13 Activities
- 4.14 Case Study
- 4.15 Further Readings

4.0 Learning Objectives :

After learning this unit, you will be able to understand :

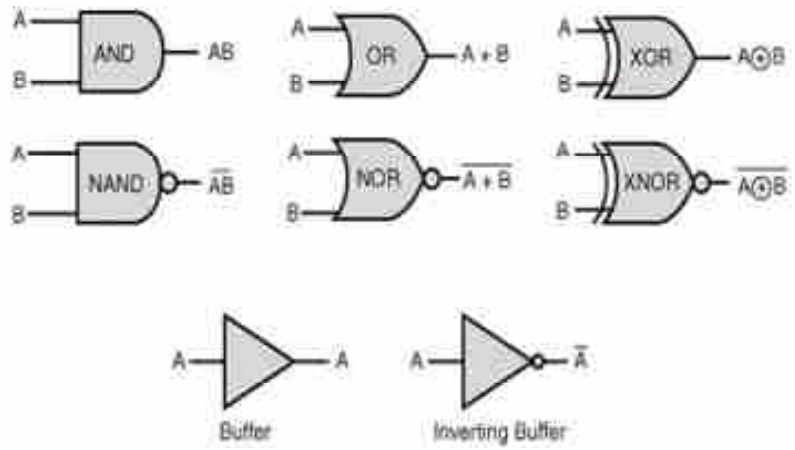
- State Boolean laws
- List Boolean identities
- Outline Boolean algebra properties
- Specify Boolean rules for simplification
- Explain DeMorgan's Theorem
- Implement equations

4.1 Introduction :

The term 'Gate' is used to describe the members of the set of basic electronic components which, when combined with each other, are able to perform complex logical and arithmetic operations. 'Gates' are the physical realisation of the simple Boolean expressions. Logic gates can be designed with only the few electronic components.

❖ Types of Logic Gates :

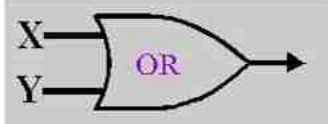
The transistors are the main components in designing circuits. There are various types of logic gates.



Basic Gates

4.2 'OR' Gate :

If any input is logic 1 then the output of an OR gate will be logic 1.



OR GATE

The output value is 1 when at least one input value is 1

Truth Table

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Check Your Progress – 1 :

1. Explain OR gate with digital circuit.

.....

.....

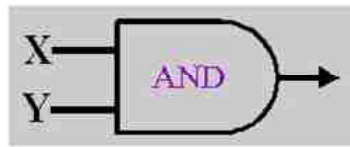
.....

.....

.....

4.3 'AND' Gate :

The AND gate requires both inputs to be logic 1 for an output of logic 1



AND GATE

Truth Table

X	Y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

The output is 1 only when both inputs are 1

❑ **Check Your Progress – 2 :**

1. Explain AND gate with digital circuit.

.....

.....

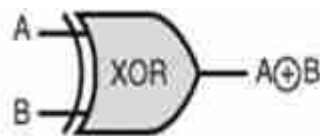
.....

.....

.....

4.4 XOR Gate :

If both The inputs are different the output is high.



XOR GATE

Truth Table

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

❑ **Check Your Progress – 3 :**

1. Explain XOR gate with digital circuit.

.....

.....

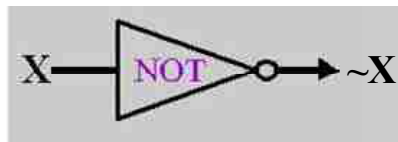
.....

.....

.....

4.5 'NOT' Gate (Inverter Gate) :

Any input is inverted. Logic 1 input yields logic 0 as output and vice versa.



NOT GATE

Truth Table

X	XNOTY
0	1
1	0

Not Gate takes only one input and is written as X or $\sim X$ in Boolean expressions.

Check Your Progress – 4 :

1. Explain NOT gate with digital circuit.

.....

.....

.....

.....

.....

4.6 NAND Gate :

It is the combination of AND gate and the NOT gate.



NAND GATE

Truth Table

X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

If both The inputs are high the output is low.

Check Your Progress – 5 :

1. Explain NAND gate with digital circuit.

.....

.....

.....

.....

.....

4.7 NOR Gate :

It is the combination of OR gate and NOT gate.



NOR GATE

Truth Table

X	Y	XNORY
0	0	1
0	1	0
1	0	0
1	1	0

□ Check Your Progress – 6 :

1. Explain NOR gate with digital circuit.

.....

.....

.....

.....

.....

4.8 XNOR Gate :

If both The inputs are same the output is high.



XNOR GATE

Truth Table

X	Y	X XNOR Y
0	0	1
0	1	0
1	0	0
1	1	1

For two binary variables (taking values 0 and 1) there are 16 possible jobs. The jobs involve only three operations, which make up Boolean algebra: AND, OR and COMPLEMENT. These operations are like ordinary algebraic operations in that they are commutative, associative and distributive.

AND: $A \cdot B$ OR: $A + B$ COMPLEMENT: $\bar{A} = \text{NOT } A$

❑ Check Your Progress 7 :


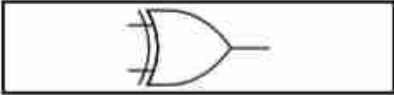

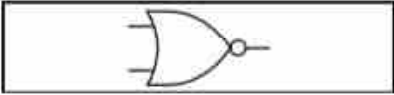
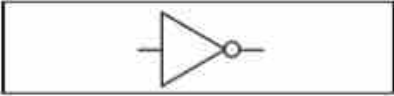
- 1 : See Section 4.2 2 : A 3 : B
 4 : C 5 : C 6 : A

4.11 Glossary :

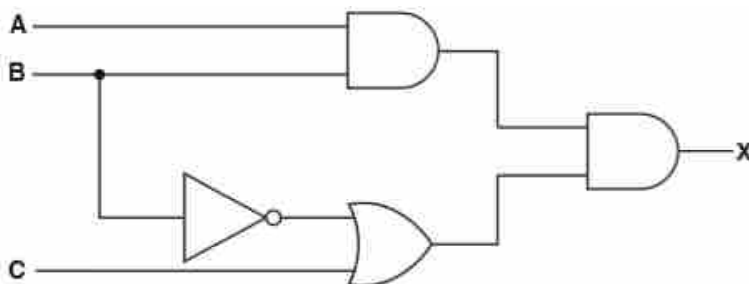
1. **Logic gate :** Boolean functions may be practically implemented by using Logic (Electronic) gates

4.12 Assignment :

1. Create Link between following two columns.

Logic Gate Symbol	Name
	AND
	NOT
	NOR
	XOR
	NAND

2. Write a logic statement for following.

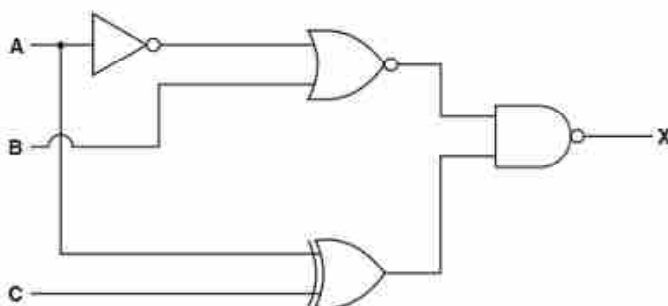


4.13 Activities :

Draw the Diagram of XOR gate and explain its functionalities.

4.14 Case Study :

A logic circuit is shown;



(a) Complete the truth table for the given logic circuit.

A	B	C	Working space	X
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

4.15 Further Reading :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar
3. <https://www.allaboutcircuit.com/textbook>

INTRODUCTION BOOLEAN ALGEBRA

UNIT STRUCTURE

- 5.0 Learning Objectives
- 5.1 Introduction
- 5.2 Boolean Laws and Theorems of Boolean Algebra
- 5.3 Boolean Identities
- 5.4 Boolean Algebraic Properties
- 5.5 Let Us Sum Up
- 5.6 Suggested Answer for Check Your Progress
- 5.7 Glossary
- 5.8 Assignment
- 5.9 Activities
- 5.10 Case Study
- 5.11 Further Readings

5.0 Learning Objectives :

After learning this unit, you will be able to understand :

- State Boolean laws
- List Boolean identities
- Outline Boolean algebra properties
- Specify Boolean rules for simplification
- Explain DeMorgan's Theorem
- Implement equations

5.1 Introduction :

In this chapter, you will find the lot of similarities between "normal algebra " and "Boolean algebra". Simply define the number of Boolean algebra system that is fully keep in mind is limited in terms of scope and only two possible values for any Boolean variable: 1 or 0. Subsequently, the 'Laws' of Boolean algebra frequently vary from the 'Laws' of real–number algebra, making promising such statements as $1 + 1 = 1$, which would normally be considered preposterous.

Once you recognize the basis of all quantities in Boolean algebra being limited to the two possibilities of 1 and 0 and the general philosophical principle of Laws depending on quantitative definitions, the "nonsense" of Boolean algebra disappears.

5.2 Boolean Laws and Theorems of Boolean Algebra :	
Identity	Dual
Identities for single variable	
Operations with 0 and 1	
1. $X + 0 = X$ (Identity)	2. $X.1 = X$
3. $X + 1 = 1$ (NULL element)	4. $X.0 = 0$
Idem Potent Theorem	
5. $X + X = X$	6. $X.X = X$
Complementarily	
7. $X + X' = 1$	8. $X.X' = 0$
Involution Theorem	
9. $(X')' = X$	
Identities for multiple variables	
Commutative	
10. $X + Y = Y + X$	11. $X.Y = Y.X$
Associative	
12. $X + (Y + Z) = (X + Y) + Z$	13. $X.(Y.Z) = (X.Y).Z$
Distributive	
14. $X.(Y + Z) = X.Y + X.Z$	15. $X+(YZ) = (X + Y)(X + Z)$
DeMorgan's Theorn	
16. $(X + Y + Z)' = X' Y' Z'$	17. $(X Y Z)' = X' + Y' + Z'$
Simplification Theorems	
18. $XY + XY' = X$ (uniting)	19. $(X + Y)(X + Y') = X$
20. $X + XY = X$ (absorption)	21. $X(X + Y) = X$
22. $(X + Y')Y = XY$ (absorption)	23. $XY' + Y = X + Y$
Consensus Theorem	
24. $XY + X'Z + YZ = XY + X'Z$	25. $(X + Y)(X' + Z)(Y + Z) = (X + Y)(X' + Z)$
Duality	
26. $(X + Y + Z + ...) D = XYZ...$	27. $(XYZ...)D = X + Y + Z +$

Note : in above table $X' = \sim X$

□ **Check Your Progress – 1 :**

1. Explain Boolean Laws.

.....

2. Explain Theorems governing Boolean Algebra

.....

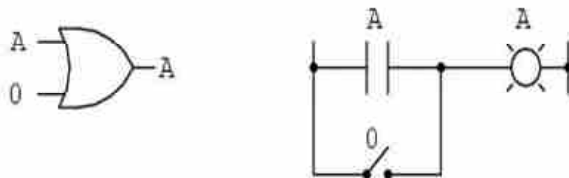
5.3 Boolean Identities :

In mathematics, For all operations there is some identity value. For example $X + 0 = X$ algebraic identity does not matter anything X may be what price, X all added to zero tells us that the original 'anything' equals. Like ordinary algebra, Boolean algebra Boolean variable based on the bivalent state has its own unique identity.

Boolean first identification and origin of zero sum 'anything' is the same as.

The identity of the reporter is changing its real algebraic number

$$A + 0 = A$$



No matter the cost, what one would consistently produce: When the input is 1, The output will be One when; When the input is 0, The output will be zero.

The following identity is different than any seen in the general algebra. Here we have another one that shows the amount of :

$$A + 1 = 1$$

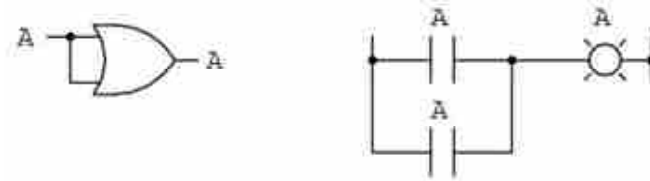


No matter the cost, what one would consistently produce : When the input is 1, the output will be One when; When the input is 0, the output will be zero.

The following identity is different than any seen in the general algebra. Here we have another one that shows the amount of what is consistently Tpadn : the = 1, the output will be even when; When the = 0, the output will be zero.

The following identity is different than any seen in the general algebra. Here we have another one that shows the amount :

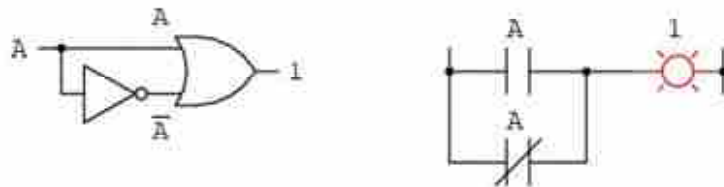
$$A + A = A$$



Only 1 and 0 in The algebra of real numbers, The amount of two basic variables for The same variables ($x + x = 2x$) twice, but in The world of mathematics Boolean '2' is The notion that in mind, so we cannot say that $A + A = 2A$. $0 + 0 = 0$ and $1 + 1 = 1$: We then add the Boolean quantity, when the amount is equal to the original volume.

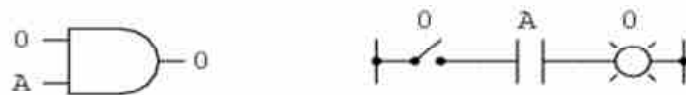
Introduces the concept of complementary specific Boolean and additive identity, we find the charming effect. Since There must be the single variable and its complement is between the amount of any Boolean quantity and the '1' value 1, is the sum of the variable and its complement must be :

$$A + \bar{A} = 1$$



Identification of four Boolean additive ($A + 0$, $A + 1$, $A + A$ and $A + A'$) as there are identified four factors: $Ax0$, $Ax1$, AXA and AXA . The first two of these expressions different from their counterparts in regular algebra are :

$$0A = 0$$



$$1A = A$$

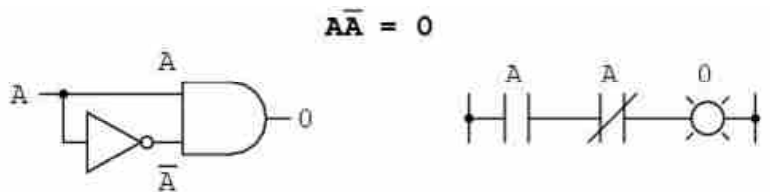


The third factor is multiplied by itself the boolean identity expresses volume outcomes. In general algebra, is the product of the variable and the variable itself ($3 \times 3 = 3^2 = 9$) of the class. However, the "class" has no meaning in the concept of the second volume of Boolean algebra, mean, so we cannot say that $X = A^2$. As an alternative, we have the Boolean product of volume, finding unique content that $x \cdot 0 = 0$ and 1 since $0 \times 1 = 1$:

$$AA = A$$



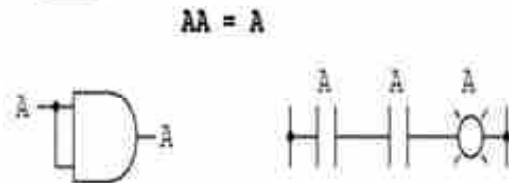
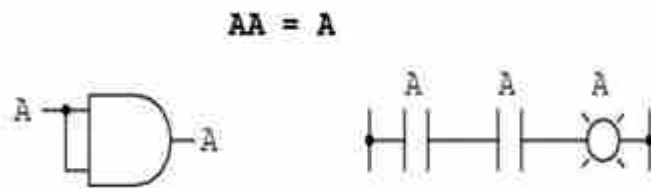
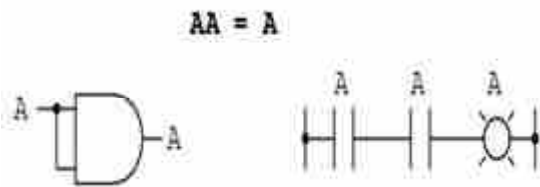
The fourth multiplicative identity has no equal in regular algebra because it uses the complement of the variable, the concept exclusive to Boolean mathematics. As, There must be one "0" Between any variable and its complement any Boolean product of volume and the zero value of 0, the variable product and its complement must be zero



To review, then, we have four fundamental Boolean identities for addition and four for multiplication.

Basic Boolean algebraic identities

Additive	Multiplicative
$A + 0 = A$	$0A = 0$
$A + 1 = 1$	$1A = A$
$A + \bar{A} = 1$	$\bar{A}A = 0$
$A + A = A$	$\bar{A}\bar{A} = \bar{A}$



Conversely, the variable twice: complementation is going to do with the identity of the double complement. Unique Boolean value twice the variable (or any number of times) outcomes complement. The actual number of algebra (multiplied by -1) negating concerns: negations original price Cancel to leave an even number of.

□ Check Your Progress – 2 :

- List all Additive and multiplicative Boolean identities.

.....

.....

.....

.....

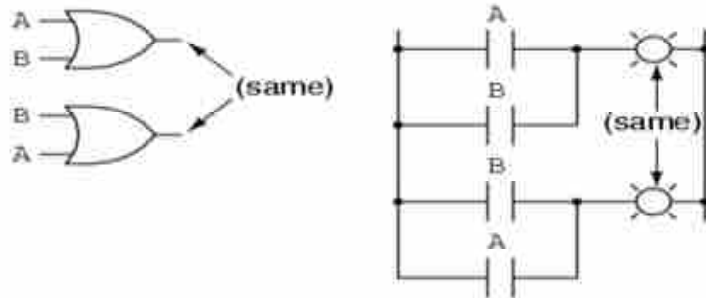
.....

5.4 Boolean Algebraic Properties :

Conversely, the variable twice: complementation is going to do with the identity of the double complement. Unique Boolean value twice the variable (or any number of times) outcomes complement. The actual number of algebra (multiplied by - 1) negating concerns: negations original price Cancel to leave an even number :

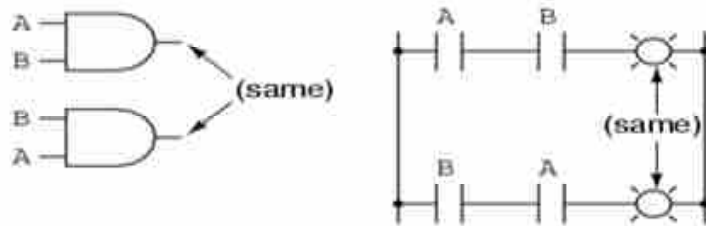
Commutative property of addition

$$A + B = B + A$$



Commutative property of multiplication

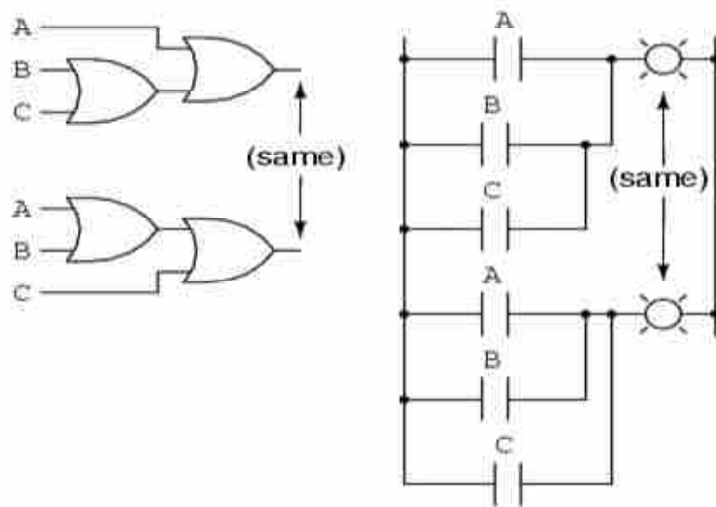
$$AB = BA$$



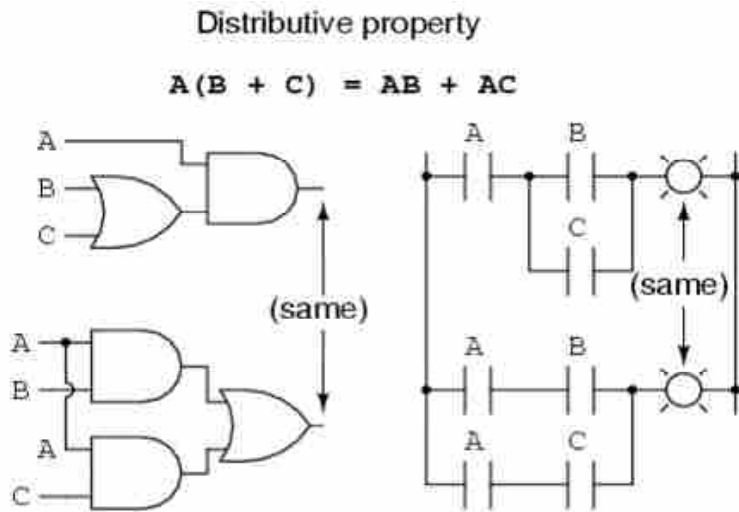
Along with the commutative properties of addition and multiplication, we have the associative property, over again applying equally well to addition and multiplication. This property shows us we can associate groups of added or multiplied variables together with parentheses without altering the truth of the equations.

Associative property of addition

$$A + (B + C) = (A + B) + C$$



Reference : <https://www.allaboutcicuits.com/textbook/dogital/chpt-7/boolean-algebraic-properties/>



Finally, we have the distributive property, illustrating how to enlarge the Boolean expression shaped by the product of the sum and in reverse shows us how terms may be factored out of Boolean sums-of-products:

To review, here are the three basic properties: commutative, associative and distributive.

Basic Boolean algebraic properties

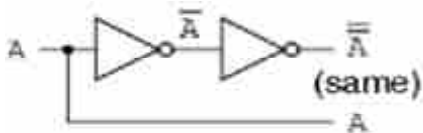
Additive

$$A + B = B + A$$

$$A + (B + C) = (A + B) + C$$

$$A(B + C) = AB + AC$$

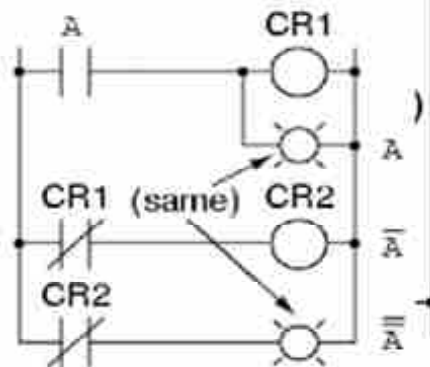
$$\overline{\overline{A}} = A$$



Multiplicative

$$AB = BA$$

$$A(BC) = (AB)C$$



Basic Boolean algebraic properties

Additive

$$A + B = B + A$$

$$A + (B + C) = (A + B) + C$$

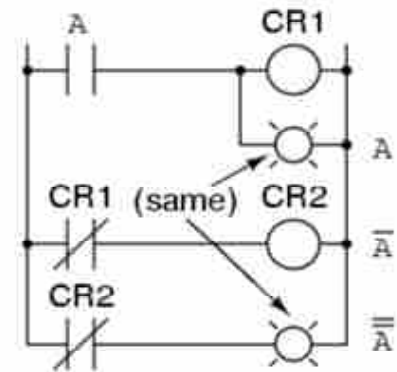
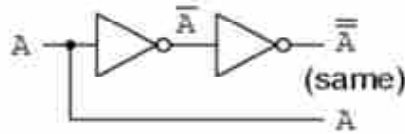
$$A(B + C) = AB + AC$$

Multiplicative

$$AB = BA$$

$$A(BC) = (AB)C$$

$$\overline{\overline{A}} = A$$



□ **Check Your Progress – 3 :**

1. What are The Boolean Algebraic Properties ?

.....

.....

.....

.....

.....

2. $A + 0 = \underline{\hspace{2cm}}$?

- (a) A (b) 0 (c) 1 (d) None

3. $X + XY = \underline{\hspace{2cm}}$ using $\underline{\hspace{2cm}}$ law.

- (a) . Y, ABSORPTION LAW (b) Y, associative law
(c) X, absorption law (d) None of above

4. What is dual of (P Q R S) ?

- (a) P + Q + R + S (b) P' + Q' + R' + S'
(c) P' * Q' * R' * S' (d) P Q R S

5. $AA = A$ is $\underline{\hspace{2cm}}$ law

- (a) Associative (b) Commutative (c) Idempotent (d) Absorption

6. $A + A' = \underline{\hspace{2cm}}$

- (a) 0 (b) A (c) 1 (d) A'

7. Which of following is Commutative property ?

- (a) $a + b = b + a$ (b) $a * b = b * a$
(c) Both a and b (d) None of above

5.5 Let Us Sum Up :

Like normal algebra, Boolean algebra has a number of suitable identities. An "identity" is merely a relation that is always true, regardless of the values that any variables involved might take on. Many of these are very equivalent to normal multiplication and addition, particularly when the symbols {0, 1} are used for {FALSE, TRUE}. But while this can be useful, there are certain identities that are dissimilar and that cause misperception.

5.6 Answer for Check Your Progress :

□ Check Your Progress 1 :

See Section 5.2

□ Check Your Progress 2 :

See Section 5.3

□ Check Your Progress 3 :

1 : See Section 5.4

2 : a

3 : c

4 : a

5 : c

6 : c

7 : c

5.7 Glossary :

1. **Algebraic properties :** Algebraic properties are used to solve algebra.

5.8 Assignment :

1. Draw the digital circuit for Associative law [Addition].

5.9 Activities :

List application of Boolean laws.

5.10 Case Study :

Using Boolean laws and identities simplify $(A + B)(A + C)$ to $A.(B + C)$

5.11 Further Reading :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar
3. <https://www.allaboutcircuit.com/textbook>

UNIT STRUCTURE

- 6.0 Learning Objectives
- 6.1 Introduction
- 6.2 De Morgan's Law
- 6.3 Let Us Sum Up
- 6.4 Suggested Answer for Check Your Progress
- 6.5 Glossary
- 6.6 Assignment
- 6.7 Activities
- 6.8 Case Study
- 6.9 Further Readings

6.0 Learning Objectives :

After Completing of this unit you will able to :

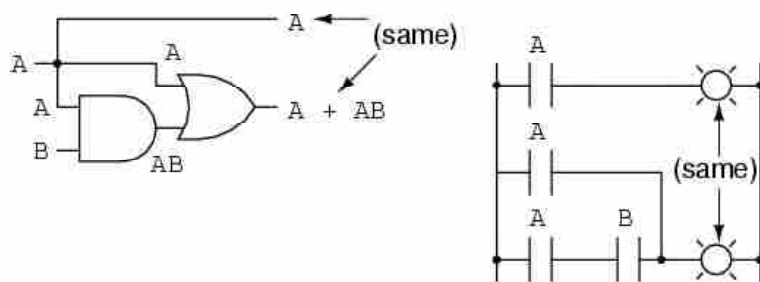
- Simplify the Boolean expression
- Use DE Morgan's law to simplify Boolean expression

6.1 Introduction :

Boolean algebra finds its mainly practical use in the generalization of logic circuits. If we interpret the logic circuit's job into symbolic (Boolean) form and apply certain algebraic rules to the consequential equation to reduce the number of terms and/or arithmetic operations, the simplified equation may be interpreted back into circuit form for the logic circuit performing the same job with smaller amount components. If corresponding job may be achieved with fewer components, the outcome will be improved consistency and decreased cost of manufacture.

To this end, they are numerous rules of Boolean algebra presented in this section for use in reducing expressions to their simplest forms. The identities and properties previously reviewed in this chapter are very useful in Boolean generalization and for the most part bear similarity to many identities and properties of 'normal' algebra. However, the rules shown in this section are all exclusive to Boolean mathematics.

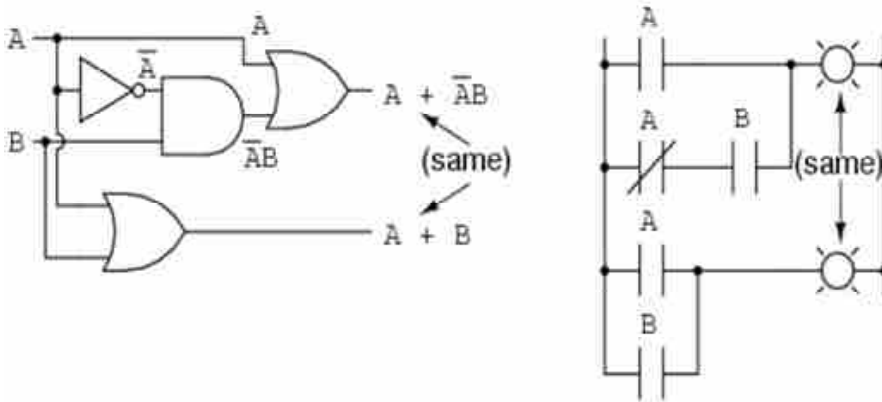
$$A + AB = A$$



$$\begin{aligned}
 & \bar{A} + \bar{A}B \\
 & \downarrow \text{Factoring } \bar{A} \text{ out of both terms} \\
 & \bar{A}(1 + B) \\
 & \downarrow \text{Applying identity } \bar{A} + 1 = 1 \\
 & \bar{A}(1) \\
 & \downarrow \text{Applying identity } 1\bar{A} = \bar{A} \\
 & \bar{A}
 \end{aligned}$$

This rule may be established symbolically by factoring an 'A' out of the two terms, Then applying The rules of $A + 1 = 1$ and $1A = A$ to achieve The final outcome :

$$A + \bar{A}B = A + B$$



$$\begin{aligned}
 & A + \bar{A}B \\
 & \downarrow \text{Applying the previous rule to expand } \bar{A} \text{ term} \\
 & A + AB + \bar{A}B \\
 & \downarrow \text{Factoring } B \text{ out of 2}^{\text{nd}} \text{ and 3}^{\text{rd}} \text{ terms} \\
 & A + B(A + \bar{A}) \\
 & \downarrow \text{Applying identity } A + \bar{A} = 1 \\
 & A + B(1) \\
 & \downarrow \text{Applying identity } 1A = A \\
 & A + B
 \end{aligned}$$

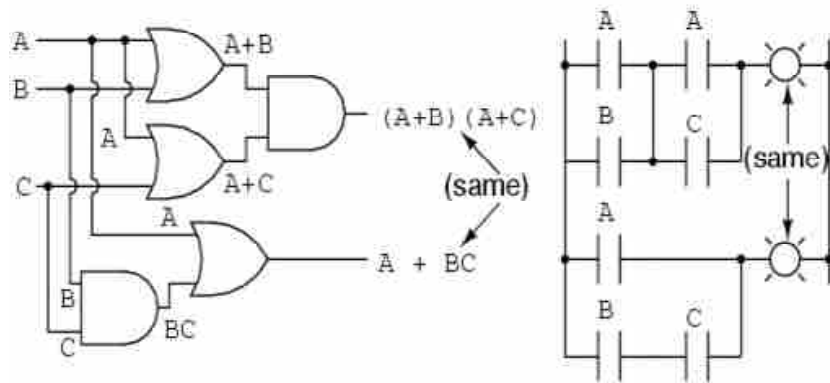
Please note how the rule $A + 1 = 1$ was used to decrease the $(B + 1)$ term to 1. When the rule like ' $A + 1 = 1$ ' is uttered using the letter 'A', it doesn't mean it only applies to expressions containing 'A'. What the 'A' stands for in the rule like $A + 1 = 1$ is any Boolean variable or collection of variables. This is possibly the hard concept for new students to master in Boolean simplification: applying standardised identities, properties and rules to expressions not in standard form.

For example, The Boolean expression $ABC + 1$ also reduces to 1 by means of the ' $A + 1 = 1$ ' identity. In this case, we know that the 'A' term in the identity's standard form can signify the entire 'ABC' term in the original expression.

The next rule looks like to the first one shown in this section, but is really relatively different and requires the more clever proof :

Note how the last rule ($A + AB = A$) is used to 'un-simplify' the first 'A' term in the expression, changing the 'A' into 'A + AB'. Whereas this may seem like the backward step, it definitely helps to reduce the expression to something simpler. Infrequently in mathematics we must take 'backward' steps to achieve the most elegant solution. Knowing when to take such the step and when not to, is part of the art-form of algebra, just as the victory in the game of chess almost always requires calculated sacrifices. Another rule involves the simplification of the product-of-sums expression.

$$(A + B)(A + C) = A + BC$$



$$\begin{aligned}
 &(A + B)(A + C) \\
 &\quad \downarrow \text{Distributing terms} \\
 &AA + AC + AB + BC \\
 &\quad \downarrow \text{Applying identity } AA = A \\
 &A + AC + AB + BC \\
 &\quad \downarrow \text{Applying rule } A + AB = A \\
 &\quad \quad \text{to the } A + AC \text{ term} \\
 &A + AB + BC \\
 &\quad \downarrow \text{Applying rule } A + AB = A \\
 &\quad \quad \text{to the } A + AB \text{ term} \\
 &A + BC
 \end{aligned}$$

Useful Boolean rules for simplification

$$A + AB = A$$

$$A + \bar{A}B = A + B$$

$$(A + B)(A + C) = A + BC$$

□ **Check Your Progress – 1 :**

1. State simplified rules of Boolean Algebra.

.....

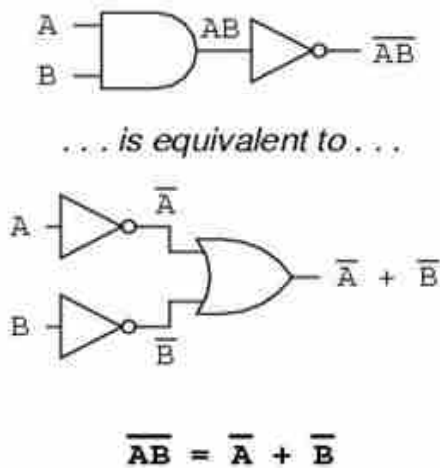
.....

.....

6.2 De Morgan's Theorem :

A mathematician named De Morgan developed the pair of significant rules concerning group complementation in Boolean algebra. Group complementation, refers to the complement of the group of terms, denoted by the long bar over more than one variable.

You should recall from the chapter on logic gates that inverting all inputs to the gate reverses that gate's significant job from AND to OR, or vice versa and also inverts The output. So, an OR gate with all inputs inverted (a Negative-OR gate) behaves the same as the NAND gate and an AND gate with all inputs inverted (a Negative-AND gate) behaves the same as the NOR gate. De Morgan's Theorems state the same equivalence in 'backward' form : that inverting the output of any gate outcomes in the same job as the opposite type of gate (AND vs. OR) with inverted inputs

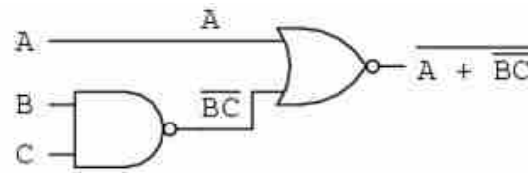


A long bar extending over the term AB acts as the grouping symbol and as such is completely different from the product of the and B separately inverted. In other words, (AB)' is not equal to A'B'. Because the 'prime' symbol (') cannot be stretched over two variables like the bar can, we are forced to use parentheses to make it apply to the whole term AB in The previous sentence. The bar, however, acts as its own grouping symbol when stretched over more than one variable. This has introspective impact on how Boolean expressions are evaluated and condensed, as we shall see DeMorgan's Theorem may be thought of in terms of breaking the long bar symbol. When the long bar is broken, the operation straight underside the break changes from addition to multiplication, or vice versa and the broken bar pieces remain over the individual variables. To demonstrate :

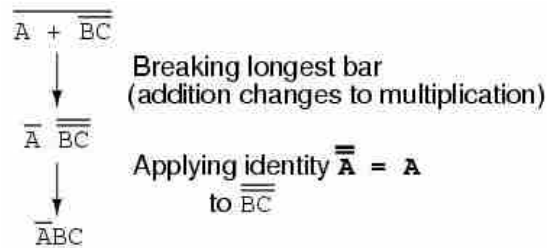
DeMorgan's Theorems



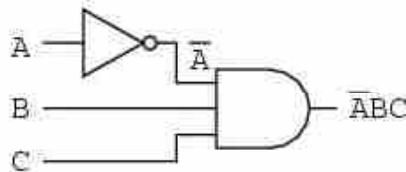
When multiple 'layers' of bars exist in an expression, you may only break one bar at the time and it is typically easier to begin oversimplification by breaking the longest (uppermost) bar first. To illustrate, let's take the expression $(A + (BC))'$ and reduce it using DeMorgan's Theorems :



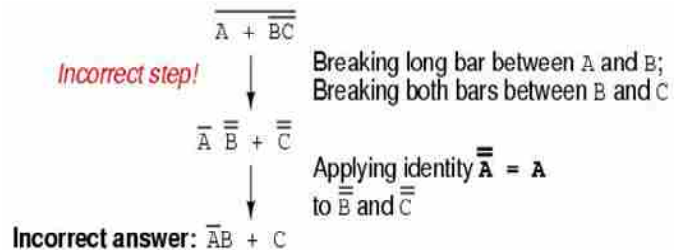
Following The recommendation of breaking the longest (uppermost) bar first, let us begin by breaking the bar covering the whole expression as the first step :



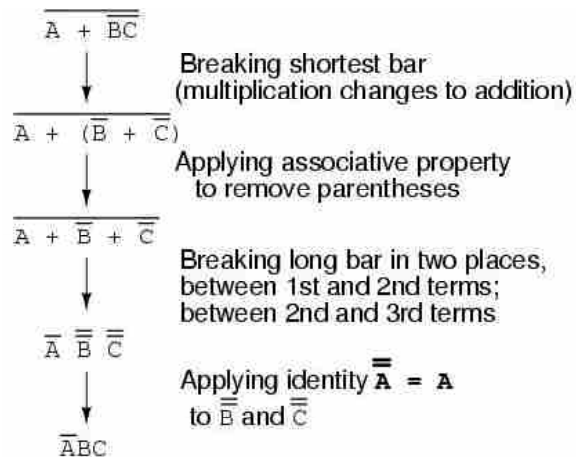
As the outcome, The unique circuit is condensed to the three-input AND gate with the input inverted :



You should never split more than one bar in the single step, as illustrated here :

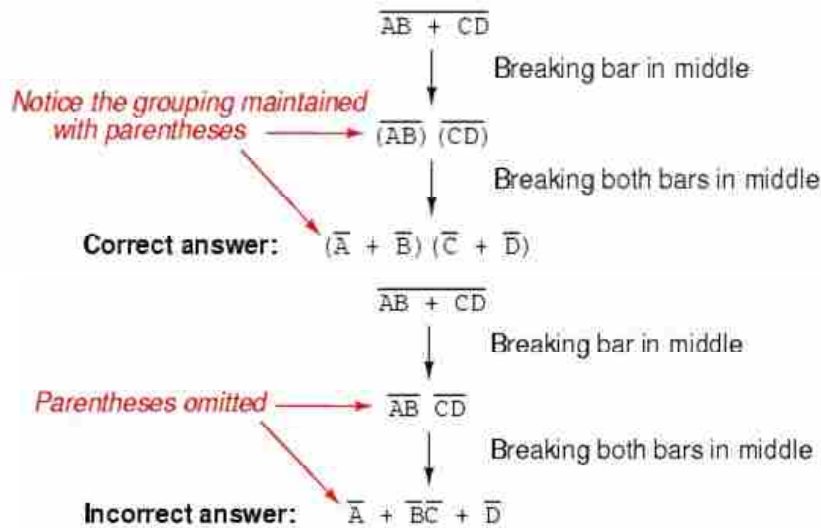


As tempting as it may be to preserve steps and break more than one bar at the time, it often leads to an incorrect outcome. It is practicable to properly reduce this expression by breaking the short bar first, rather than the long bar first :

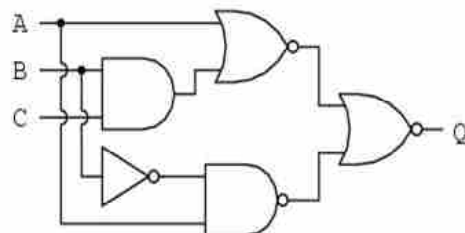


The end outcome is the similar, but more steps are necessary compared to using the first method, where the longest bar was broken first. Note how in the third step we broke the long bar in two places. This is the sensible mathematical operation and not the same as breaking two bars in one step. The prohibition alongside breaking more than one bar in one step is not the prohibition against breaking the bar in more than one place. Breaking in more than one place in the single step is okay; breaking more than one bar in the single step is not.

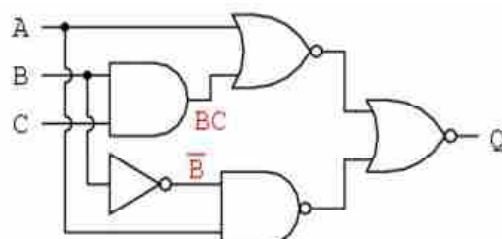
You might be surprised why parentheses were placed around the sub-expression $B' + C'$, when they were removed them in the next step. This done to lay emphasis on an important but simply ignored aspect of DeMorgan's theorem. Since the long bar jobs as the grouping symbol, the variables formerly grouped by the broken bar must remain grouped lest proper preference (order of operation) be lost. In this example, it really wouldn't matter if you forgot to put parentheses in after breaking the short bar, but in other cases it might. Consider this example, starting with the dissimilar expression :



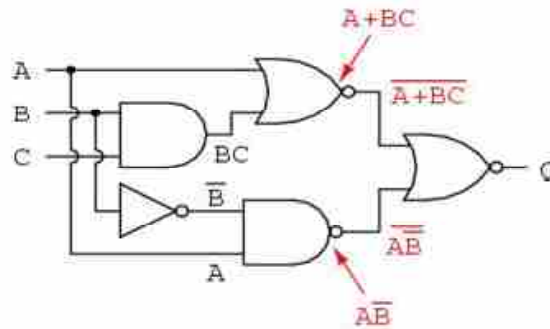
As you can see, maintaining the grouping implicit by the complementation bars for this expression is crucial to obtaining the correct answer. Let's apply the principles of De Morgan's theorems to the generalization of the gate circuit :



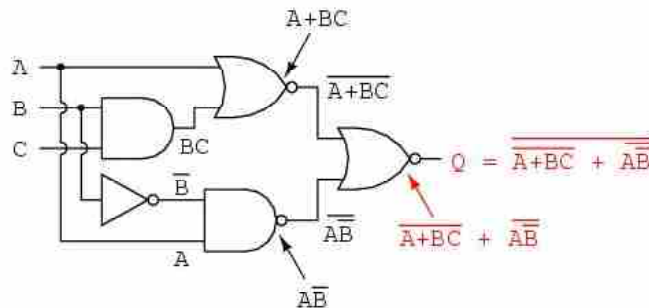
As always, our first step in simplifying this circuit must be to make an equivalent Boolean expression. We can do this by placing the sub-expression label at the output of each gate, as the inputs become known. Here's the first step in this process :



Next, we can label the outputs of the first NOR gate and the NAND gate. When dealing with inverted-output gates, I find it easier to write an expression for the gate's output without the concluding inversion, with an arrow pointing to just before the inversion bubble. Then, at the wire leading out of the gate (after the bubble), write the full, complemented expression. This ensures you don't forget the complementing bar in the sub-expression, by splitting the expression-writing task into two steps :



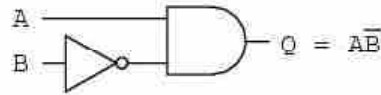
At last, we write an expression (pair of expressions) for the last NOR gate :



Now, we condense this expression using the identities, properties, rules and Theorems (DeMorgan's) of Boolean algebra :

$$\begin{aligned}
 & \overline{\overline{A + BC + A\overline{B}}} \\
 & \downarrow \text{Breaking longest bar} \\
 & \overline{(\overline{A + BC}) (\overline{A\overline{B}})} \\
 & \downarrow \text{Applying identity } \overline{\overline{A}} = A \text{ wherever double bars of equal length are found} \\
 & (A + BC) (A\overline{B}) \\
 & \downarrow \text{Distributive property} \\
 & AA\overline{B} + BCA\overline{B} \\
 & \downarrow \text{Applying identity } AA = A \text{ to left term; applying identity } A\overline{A} = 0 \text{ to B and } \overline{B} \text{ in right term} \\
 & A\overline{B} + 0 \\
 & \downarrow \text{Applying identity } A + 0 = A \\
 & A\overline{B}
 \end{aligned}$$

The equivalent gate circuit for this much-simplified expression is as follows :



Reference : <https://www.allaboutcircuit.com/textbook/digital/chpt-7>

□ Check Your Progress – 2 :

1. Describe DeMorgan's Theorems.

.....

2. The compound statement $A \vee \sim (A \wedge B) = 1$.

- (a) True (b) False

3. After applying simplification _____ Is output of $B + BA$

- (a) B (b) A (c) AB (d) None

4. D'Morgan's law is _____ and _____

- (a) $A + B = B + A$ and $AB = BA$
 (b) $(A + B)' = A' B'$ and $(AB)' = A' + B'$
 (c) $(A + B)' = AB$ and $(AB)' = A + B$
 (d) All of above

5. $(A + B)(A + C) =$ _____

- (a) A (b) $(A + AC)$ (c) $(A + BC)$ (d) All of above

6. $A + A'B =$ _____

- (a) B (b) A (c) $A + B$ (d) $A' + B$

6.3 Let Us Sum Up :

The **Demorgan's** theorem defines the uniformity between the gate with the same inverted input and output. It is used for implementing the basic gate operation likes NAND gate and NOR gate. The **Demorgan's** theorem mostly used in **digital** programming and for making **digital** circuit diagrams

6.4 Answer for Check Your Progress :

□ Check Your Progress 1 :

See Section 6.1

□ Check Your Progress 2 :

- | | | |
|---------------------|-------|-------|
| 1 : See Section 6.2 | 2 : a | 3 : a |
| 4 : b | 5 : c | 6 : c |

6.5 Glossary :

1. **Algebraic properties :** Algebraic properties are used to solve algebra.

6.6 Assignment :

1. Draw the digital circuit for Associative law [Addition].

6.7 Activities :

List application of Boolean laws.

6.8 Case Study :

Using Boolean laws and identities simplify $(A + B)(A + C)$ to $A.(B + C)$

6.9 Further Reading :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar
3. <https://www.allaboutcircuit.com/textbook>

UNIT STRUCTURE

- 7.0 Learning Objectives
- 7.1 Introduction
- 7.2 Truth Tables
- 7.3 Simplification of Boolean Equation using K-Map
- 7.4 Let Us Sum Up
- 7.5 Suggested Answer for Check Your Progress
- 7.6 Glossary
- 7.7 Assignment
- 7.8 Activities
- 7.9 Case Study
- 7.10 Further Readings

7.0 Learning Objectives :

After Completing of this unit you will able to :

- Use K-map to minimize Long Boolean expression
- Use K-map to simplify Long Boolean expression

7.1 Introduction :

The **Karnaugh map (K-map)** is a method of simplifying Boolean algebra expressions.

Karnaugh map reduces the need for extensive calculations by taking advantage of humans' pattern-recognition capability.

The Boolean results are transferred from a truth table onto a two-dimensional grid where, in Karnaugh maps, the cells are ordered in Gray code and each cell position represent one combination of input conditions cells which are also known as minterms, while each cell value represents the corresponding output value of the boolean function. Optimal groups of 1s or 0s are identified, which represent the terms of a canonical form of the logic in the original truth table. These terms can be used to write a minimal Boolean expression representing the required logic

Declarative statements Example:

- a. Tomorrow is Saturday.
- b. The month of June has thirty days.
- c. The moon is made of red cheese.
- d. New Haven is the city in Connecticut.

The following are not statements :

- a. Good bye dear.
- b. Come to our party !
- c. Close the door when you leave.
- d. Is your homework done ?

Those are not fine statements because they cannot be measured true or false. The basic type of sentence in logic is called the simple statement. The simple statement is one that has only one consideration with no connecting word.

Simple statements Example

- a. Five is the counting number.
- b. Bob is early for class

If we take the simple statement and join them with the connecting word such as, if, or, and.... Then, not, if and only if, we outline the new sentence called the complex or compound statement.

Compound statements are created from the combination of two or more simple statements.

Example :

Bob is early for class and he has his note books.

Five is counting number and is also an odd number.

Types of compound statements and their connectives

- 1. A negation: formed when we negate the simple statement by 'not'.
 - a. Example : Simple statement : Today is Monday
 - b. Compound statement : Negation: Today is not Monday
 - c. The sentence "Today is not Monday" is the compound statement called the negation.
- 1. When we connect two effortless statements using 'and', the outcome is the compound statement called the conjunction.
- 2. If the simple statements are joined by or the out coming compound statement is called the disjunction.
- 3. The '*if Then*' connector is used in compound statements called conditionals.
- 4. The '*if and only if*' connector is used to form compound statements called bi-conditionals.
 - a. We are familiar with using letters as replacements in algebra; in logic we can also use letters to replace statements. The common letters used to change statements are P, Q, and R: but any letters can be used.

Examples :

P = Today is Tuesday.

Q = I passed my test.

But P and Q would read Today is Tuesday and I passed my test. It is also common practice to use symbols for the connective words (or the connectors)

CONNECTORS	SYMBOLS
(a) not	\sim
(b) and	\wedge
(c) or	(figure available in print form)
(d) if ... Then	\Rightarrow
(e) if and only if	\Leftrightarrow

7.2 Truth Tables :

Since the statement in logic is either true or false, we should be able to decide the truth or falsity of the given statement. [*Logic is very accurate. There should be no worry about ambiguity*] Let P be the statement; Then $\sim P$ means 'not P' or the negation of P. The reversal of P is true whenever the statement P is false and false if P is true. These situations are confusing to write; thus we can record these statements in the truth table.

Example 1 : Let P = it is the hard course.

$\sim P$ = it is not the hard course.

Truth Table [T = True and F = False]

P	$\sim P$
T	F
F	T

In The first column, there are two possibilities of P; P is either true or false. Each line in the table represents the case that must be considered. In this case, there are only two cases. The truth table shows us the truth value of p in every case.

Truth Tables with the Connective \wedge

The Connective \wedge may be positioned between any two statements P and Q to form the compound statement $p \wedge q$

Let P = Today is Friday.

Q = I have the Hindi class.

Truth Table

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

In The compound statements, the individual statements are called components. In the compound statement with two components such as $p \wedge q$ There are four possibilities. These are called logical possibilities.

The possibilities are describing as :

1. p is true and q is true
2. p is true and q is false

- 3. p is false and q is true
- 4. p is false and q is false.

The four possibilities are covered in the four rows of the truth table. The last column gives values of $p \wedge q$. This is only true when both p and q are true. Using the examples given, truth tables of the more complex nature can be built.

Let us assume the situation $p \vee q$

Example 2 : P = Today is Friday. Q = I have the Hindi class

$P \vee Q$ = Today is Friday or I have the Hindi class

P	Q	$P \vee Q$
T	T	T
F	T	T
T	F	T
F	F	F

Example 3 : Find The Truth – Table of $[(p) \wedge (q)]$

INPUT		
P	Q	$(P \wedge Q)$
F	F	F
F	T	F
T	F	F
T	T	T

Example 4 : Find the truth table of $(\sim P \wedge \sim Q)$

		INPUT		
P	Q	$\sim P$	$\sim Q$	$(\sim P \wedge \sim Q)$
T	T	F	F	F
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

The given $[(\sim p) \wedge (\sim q)]$ uses parentheses and brackets to point out the order in which the connectives apply. Expressions can be simplified by removing some of the parentheses, thus $(p) \wedge (q)$ can be written as $\sim p \wedge \sim q$. It can be noticed from The Truth Table in examples 2 and 3 that the last columns are the same. Thus, we say that these statements are logically equivalent and can be written $P = Q$ and $P \vee Q = (\sim p \wedge \sim q)$.

The conditional and The Bi-conditional Statements

If the connectors \Leftrightarrow is used between any two statements P and Q to form the compound statement $P \rightarrow Q$ (reads if P Then Q), the statement is called the conditional statement.

Let P = you passed English.

Q = you will graduate

Truth Table

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

The statement $P \leftrightarrow Q$ reads if you pass English Then you will graduate. This statement is false only when you pass English (true) but you will not graduate. Thus, the final column will be true in every position but the second.

The connective \leftrightarrow is called the bi-conditional and may be placed between any two statements to form the compound statement $P \leftrightarrow Q$ (reads P if and only if Q).

The Truth - Table for $(P \rightarrow Q) \wedge (Q \rightarrow P)$

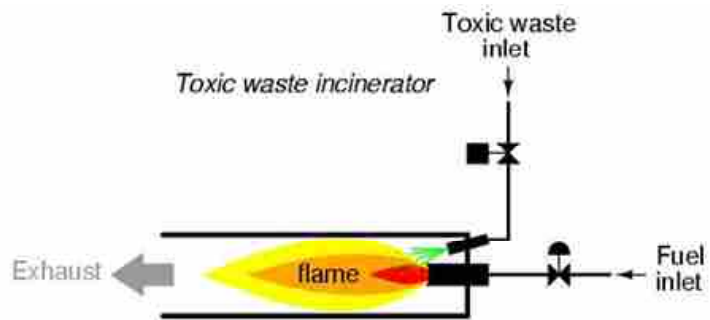
P	Q	$P \rightarrow Q$	$Q \rightarrow P$
T	T	T	T
T	F	F	T
F	T	T	F
F	F	T	T

In designing digital circuits, the designer frequently begins with the truth table describing what the circuit should do. The design task is mainly to resolve what type of circuit will perform the job described in the truth table. While some people seem to have the natural capability to look at the truth table and instantly envision the necessary logic gate or relay logic circuitry for The task, There are procedural techniques accessible for The rest of us. Here, Boolean algebra proves its utility in the most dramatic way.

Truth Table for $P \leftrightarrow Q$

P	Q	$P \leftrightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

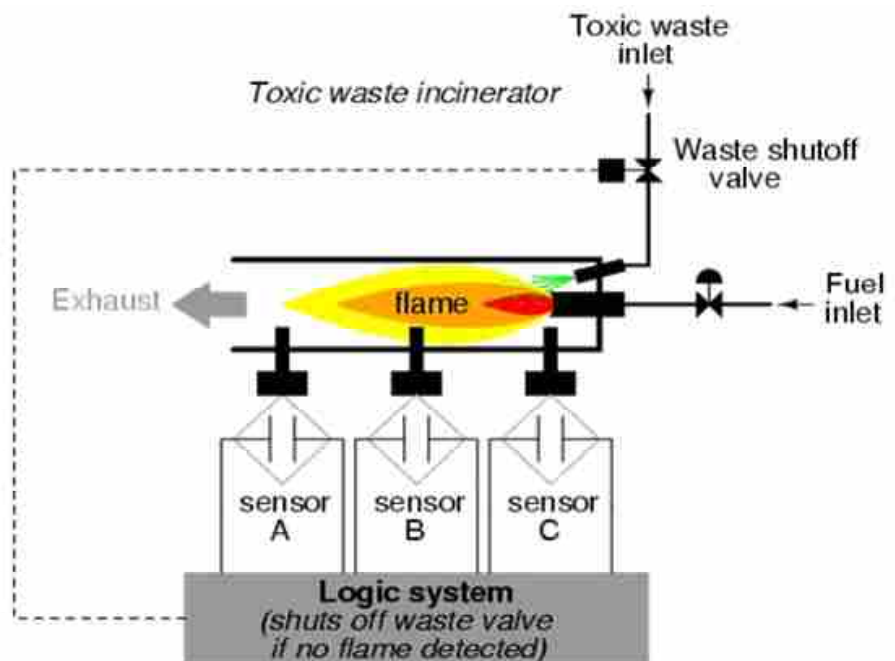
To express this procedural method, we should begin with the realistic design problem. Suppose we were given the task of designing the flame detection circuit for the toxic waste incinerator. The intense heat of the fire is proposed to neutralize the toxicity of the waste introduced into the incinerator. Such combustion-based techniques are usually used to neutralize medical waste, which may be infected with deadly viruses or bacteria :



So long as the flame is maintained in the incinerator, it is safe to inject waste into it to be neutralised. If the flame were to be extinguished, though, it would be unsafe to continue to inject waste into the combustion chamber, as it would exit the exhaust un-neutralised and pose the health threat to anyone in close proximity to the exhaust. What we require in this system is the sure way of detecting the presence of the flame and permitting waste to be injected only if the flame is 'proven' by The flame detection system.

Some different flame-detection technologies exist: optical (detection of light), Thermal (detection of high temperature) and electrical conduction (detection of ionized particles in the flame path), each one with its unique pros and cons.

Suppose that due to the high degree of hazard concerned with potentially passing un-neutralized waste out the exhaust of this incinerator, it is decided that the flame detection system be made redundant (multiple sensors), so that failure of the single sensor does not lead to an emission of toxins out the exhaust. Each sensor comes equipped with the normally-open contact (open if no flame, closed if flame detected) which we will use to inspire the inputs of the logic system :



Our task, now, is to design the circuitry of the logic system to open the waste valve if and only if there is good flame proven by the sensors. First, though, we must decide what the logical behaviour of this control system should be.

Do we want the valve to be opened if only one out of the three sensors detect flame? Probably not, since, this would defeat the purpose of having multiple sensors. If any one of the sensors were to fail in such the way as to falsely specify the presence of flame when There was none, the logic system based on The principle of 'any one out of three sensors showing flame' would give The similar output that the single–sensor system would with The similar failure.

A far better solution would be to design the system so that the valve is commanded to open if and only if all three sensors detect the good flame. This way, any single, failed sensor falsely presentation flame could not keep the valve in the open position; rather, it would need all three sensors to be failed in the same manner –the highly improbable scenario – for this dangerous condition to occur.

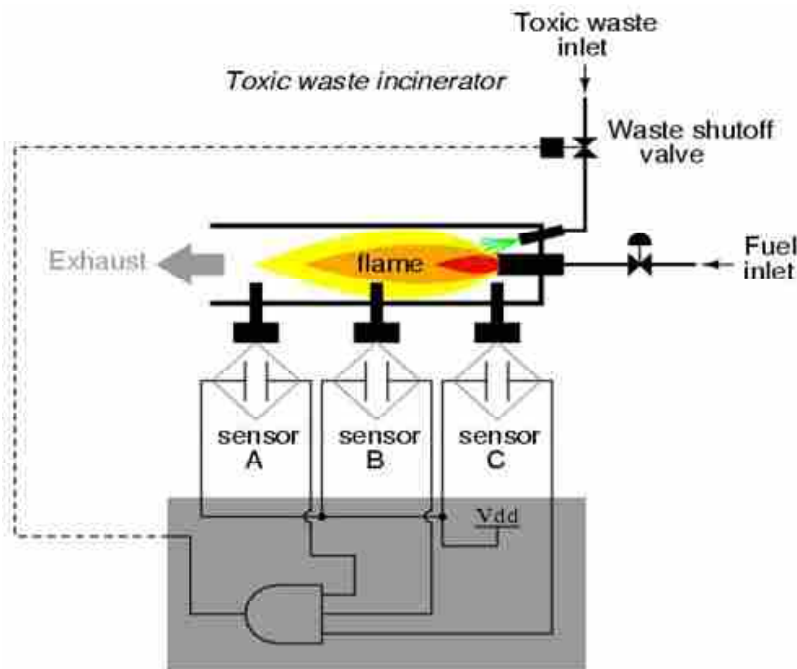
Therefore, our truth table would look like this :

sensor inputs			Output
A	B	C	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Output = 0 (close valve)

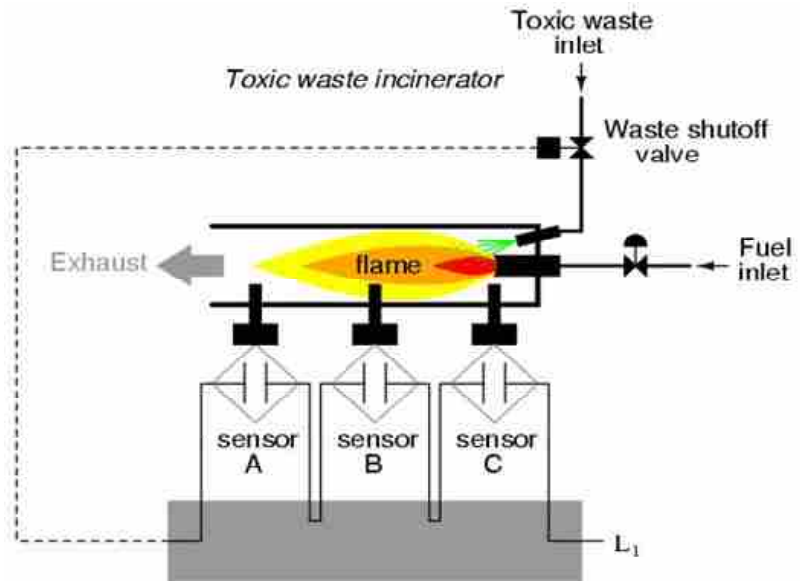
Output = 1 (open valve)

It does not require much insight to distinguish that this jobality could be generated with the three–input AND gate : The output of the circuit will be 'high' if and only if input the 'AND' input B 'AND' input C are all 'high'.



If using relay circuitry, we could create this and job by wiring three relay contacts in series, or basically by wiring the three sensor contacts in series, so that the only way electrical power could be sent to open the waste valve is if all three sensors specify flame :

❖ "Using Sum of Products"



While this design strategy maximises safety, it makes the system very susceptible to sensor failures of the opposite kind. Assume that one of the three sensors were to fail in such the way that it indicated no flame when There really was the good flame in the incinerator's combustion chamber. That single failure would shut off the waste valve unnecessarily, out coming in lost production time and wasted fuel (feeding the fire that wasn't being used to incinerate waste).

It would be nice to have the logic system that accepted for this kind of failure without shutting the system down needlessly, yet still provide sensor redundancy so as to maintain safety in the event that any single sensor failed 'high' (showing flame at all times, whether or not there was one to detect). The strategy that would meet both needs would be the 'two out of three' sensor logic, whereby the waste valve is opened if at least two out of the three sensors express good flame. The truth table for such the system would look like this :

sensor inputs			Output
A	B	C	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Output = 0 (close valve)

Output = 1 (open valve)

Here, it is not basically obvious what kind of logic circuit would satisfy the truth table. However, the simple method for designing such the circuit is found in the standard form of Boolean expression called the Sum-Of-Products, or SOP, form. As you might suspect, the Sum-Of-Products Boolean expression is accurately the set of Boolean terms added (summed) together, each term being the multiplicative (product) combination of Boolean variables.

An example of an SOP expression would be rather like this : $ABC + BC + DF$, The sum of products 'ABC', 'BC', and 'DF'.

Sum-of-products expressions are simple to generate from truth tables. All we have to do is observe the truth table for any rows where the output is "high" (1) and write the Boolean product term that would corresponding the value of 1 given those input conditions. For example, in the fourth row down in the truth table for our two-out-of-three logic system, where $A = 0$, $B = 1$ and $C = 1$, the product term would be $\bar{A}BC$, since that term would have the value of 1 if and only if $A = 0$, $B = 1$ and $C = 1$:

sensor inputs			
A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$\bar{A}BC = 1$

Three other rows of the truth table have an output value of 1, so those rows also require Boolean product expressions to represent them :

sensor inputs			
A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$\bar{A}BC = 1$

$A\bar{B}C = 1$

$AB\bar{C} = 1$

$ABC = 1$

Ultimately, we join these four Boolean product expressions jointly by addition, to create the single Boolean expression describing the truth table as the whole :

sensor inputs			
A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$\bar{A}BC = 1$

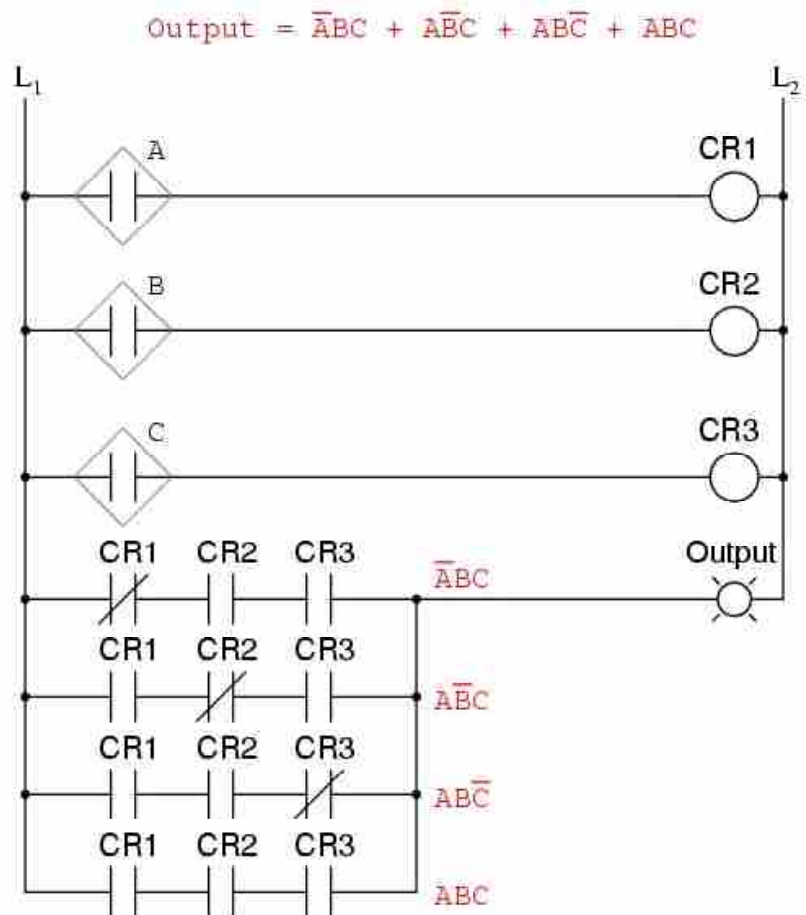
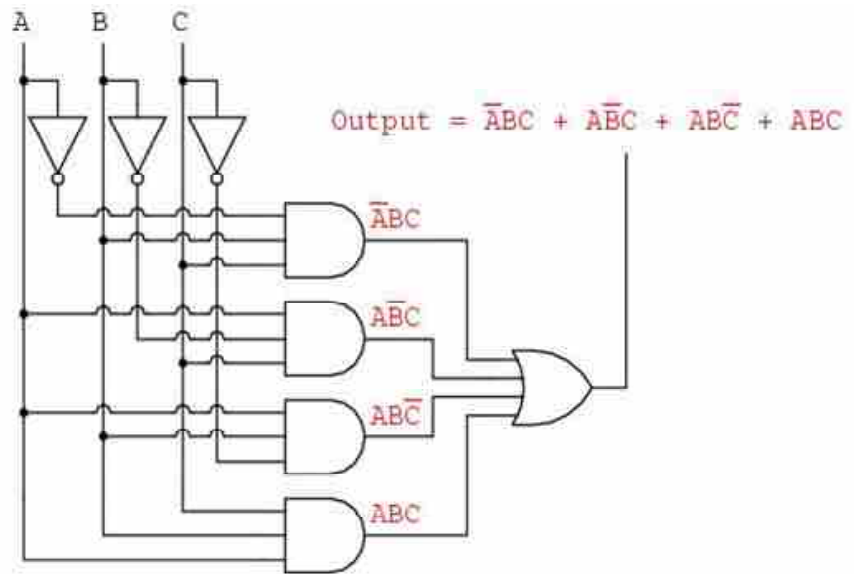
$A\bar{B}C = 1$

$AB\bar{C} = 1$

$ABC = 1$

$Output = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

Now that we have the Boolean sum-of-products expression for the truth table's job, we can simply design the logic gate or relay logic circuit based on that expression :

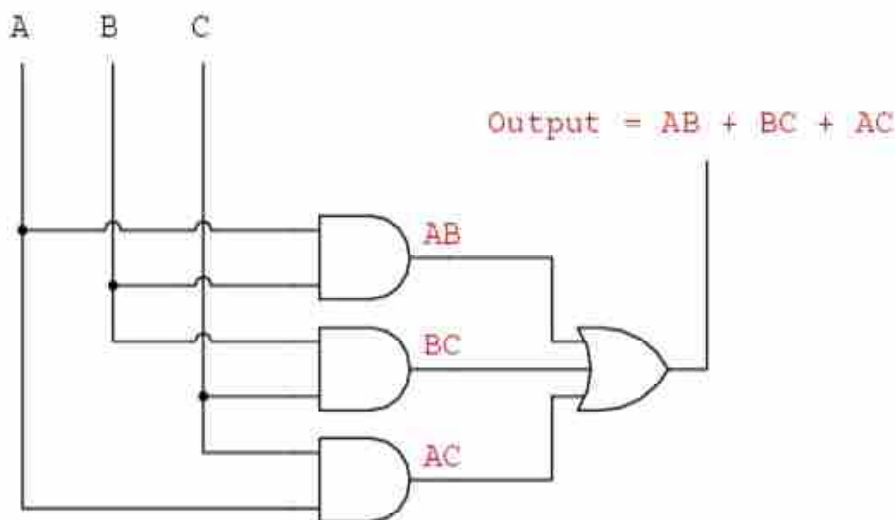


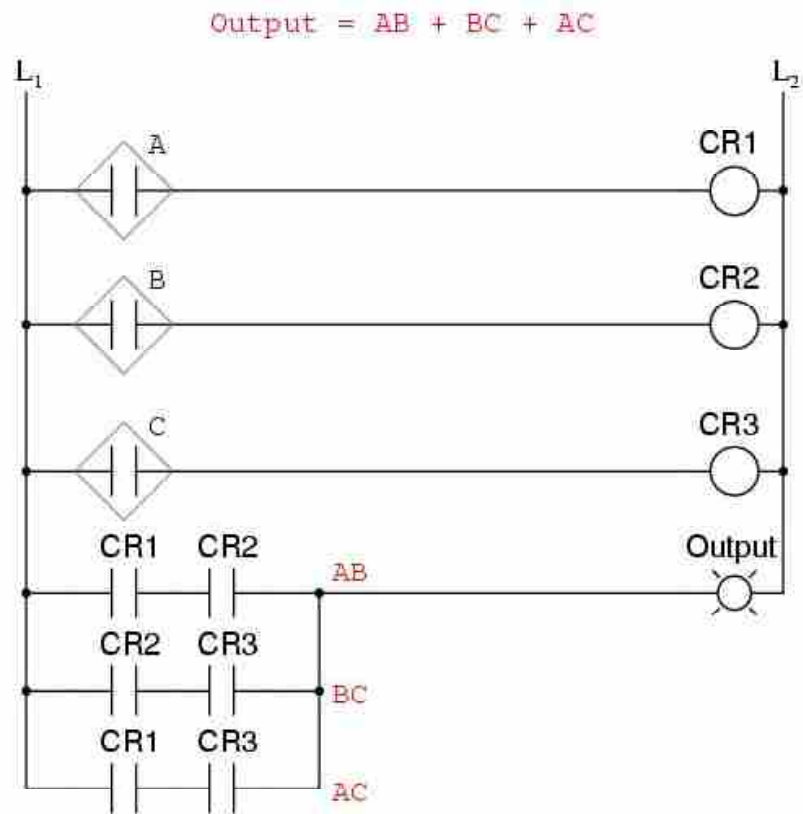
Unluckily, both of these circuits are quite complex and could benefit from simplification. Using Boolean algebra techniques, the expression may be considerably simplified :

Simplification of Boolean Algebra - II

$$\begin{aligned}
 & \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC \\
 & \downarrow \text{Factoring } BC \text{ out of 1}^{\text{st}} \text{ and 4}^{\text{th}} \text{ terms} \\
 & BC(\overline{A} + A) + A\overline{B}C + AB\overline{C} \\
 & \downarrow \text{Applying identity } A + \overline{A} = 1 \\
 & BC(1) + A\overline{B}C + AB\overline{C} \\
 & \downarrow \text{Applying identity } 1A = A \\
 & BC + A\overline{B}C + AB\overline{C} \\
 & \downarrow \text{Factoring } B \text{ out of 1}^{\text{st}} \text{ and 3}^{\text{rd}} \text{ terms} \\
 & B(C + A\overline{C}) + A\overline{B}C \\
 & \downarrow \text{Applying rule } A + \overline{A}B = A + B \text{ to the } C + A\overline{C} \text{ term} \\
 & B(C + A) + A\overline{B}C \\
 & \downarrow \text{Distributing terms} \\
 & BC + AB + A\overline{B}C \\
 & \downarrow \text{Factoring } A \text{ out of 2}^{\text{nd}} \text{ and 3}^{\text{rd}} \text{ terms} \\
 & BC + A(B + \overline{B}C) \\
 & \downarrow \text{Applying rule } A + \overline{A}B = A + B \text{ to the } B + \overline{B}C \text{ term} \\
 & BC + A(B + C) \\
 & \downarrow \text{Distributing terms} \\
 & BC + AB + AC \\
 & \text{or} \\
 & \text{Simplified result} \\
 & AB + BC + AC
 \end{aligned}$$

As the outcome of the generalization, we can now build much simpler logic circuits performing the same job, in either gate or relay form :





Either one of these circuits will adequately perform the task of operating the incinerator waste valve based on the flame verification from two out of the three flame sensors. At minimum, this is what we need to have the safe incinerator system. We can, however, expand the functionality of the system by adding to its logic circuitry designed to detect if any one of the sensors does not agree with the other two.

If all three sensors are operating correctly, they should detect flame with equal accuracy. Thus, they should either all register 'low' (000 : no flame) or all register 'high' (111 : good flame). Any other output combination (001, 010, 011, 100, 101, or 110) constitutes the discrepancy between sensors and may therefore serve as an indicator of the potential sensor failure.

If we add circuitry to detect any one of the six 'sensor disagreement' conditions, we could use the output of that circuitry to activate an alarm. Whoever is monitoring the incinerator would then exercise judgment in either progressing to operate with the possible failed sensor (inputs : 011, 101, or 110), or shut the incinerator down to be absolutely safe.

Also, if the incinerator is shut down (no flame) and one or more of the sensors still indicates flame (001, 010, 011, 100, 101, or 110) while the other(s) indicate(s) no flame, it will be known that the definite sensor problem exists.

The first step in designing this 'sensor disagreement' detection circuit is to write the truth table describing its behaviour. Since we already have the truth table describing the output of the 'good flame' logic circuit, we can simply add another output column to the table to represent the second circuit and make the table indicating the entire logic system :

Output = 0 (close valve) Output = 0 (sensors agree)
 Output = 1 (open valve) Output = 1 (sensors disagree)

sensor inputs **Good flame** **Sensor disagreement**

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

While it is possible to create the sum-of-products expression for this new truth table column, it would require six terms, of three variables each. Such the Boolean expression would involve many steps to simplify, with the large potential for making algebraic errors.

Output = 0 (close valve) Output = 0 (sensors agree)
 Output = 1 (open valve) Output = 1 (sensors disagree)

sensor inputs **Good flame** **Sensor disagreement**

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

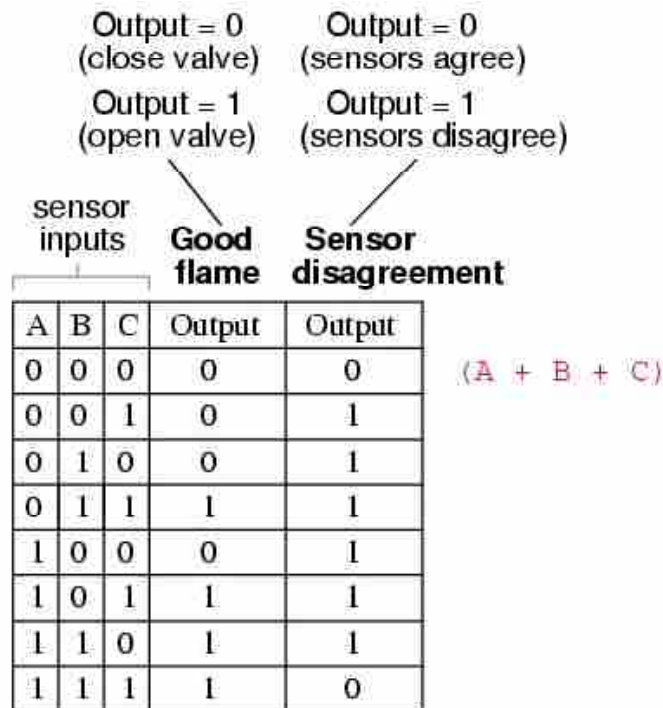
$$\text{Output} = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C + ABC$$

❖ "Product of Sums" :

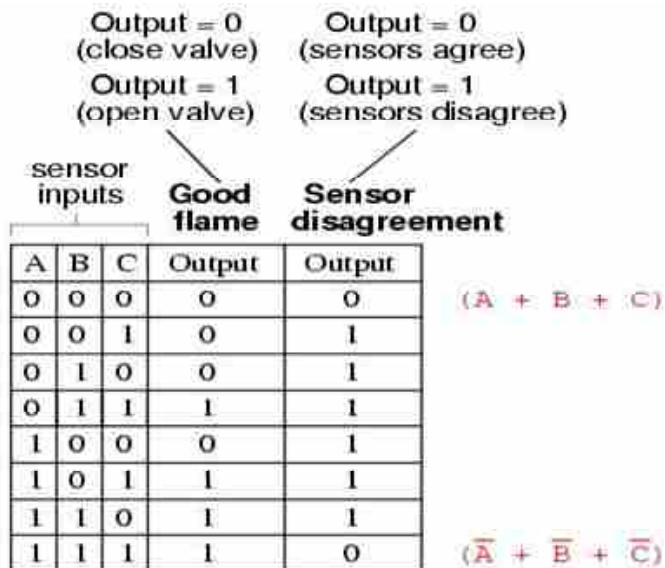
An alternative to generating the sum-of-products expression to account for all the 'high' (1) output conditions in the truth table is to generate the Product-Of-Sums, or POS, expression, to account for all the 'low' (0) output conditions instead. Being that there are much fewer instances of the 'low' output in the last truth table column, the consequential product-of-sums expression should contain fewer terms. As its name suggests, the product-of-sums expression is the set of added terms (sums), which are multiplied (product) together. An example of the POS expression would be (A + B) (C + D), the product of the sums 'A + B' and 'C + D'.

To begin, we identify which rows in the last truth table column have 'low' (0) outputs and write the Boolean sum term that would equal 0 for that

row's input conditions. For illustration, in The first row of the truth table, where $A = 0, B = 0$ and $C = 0$, The sum term would be $(A + B + C)$, since that term would have the value of 0 if and only if $A = 0, B = 0$ and $C = 0$.



Only one other row in the last truth table column has the 'low' (0) output, so all we need is one more sum term to complete our product-of-sums expression. This last sum term shows the 0 output for an input condition of $A = 1, B = 1$ and $C = 1$. Thus, the term must be written as $(\bar{A} + \bar{B} + \bar{C})$, because only the sum of the complemented input variables would equal 0 for that condition only :

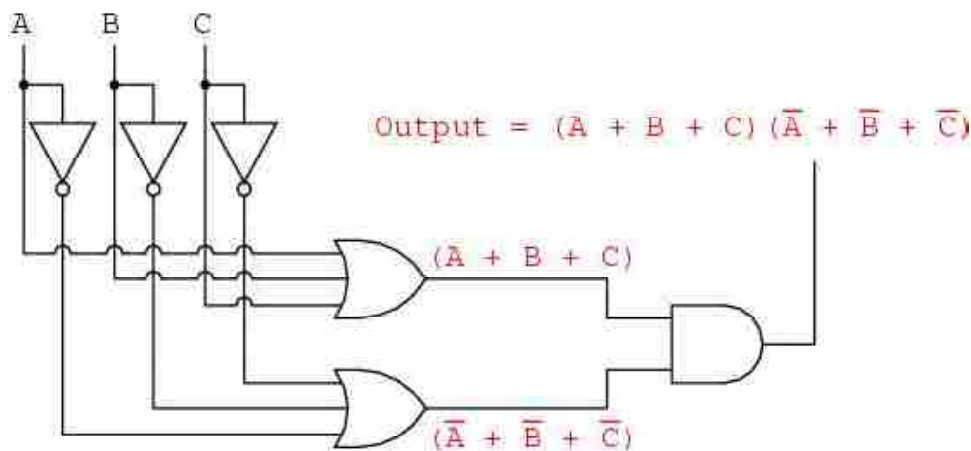


The completed product-of-sums expression, of course, is the multiplicative combination of these two sum terms :

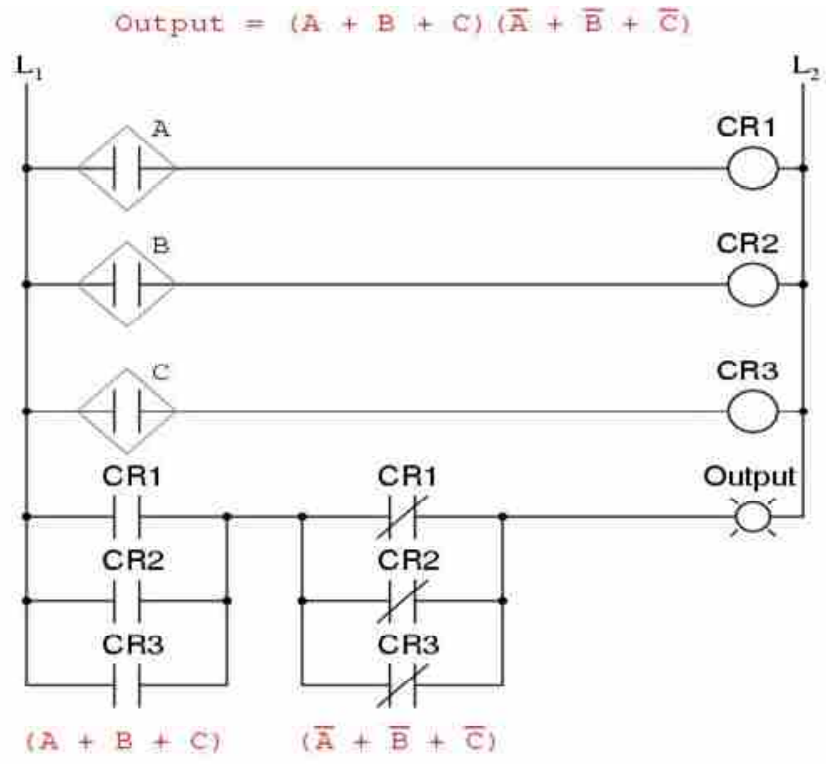
sensor inputs			Good flame	Sensor disagreement
A	B	C	Output	Output
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

$$\text{Output} = (A + B + C) (\bar{A} + \bar{B} + \bar{C})$$

Whereas the sum-of-products expression could be implemented in the form of the set of AND gates with their outputs connecting to the single OR gate, the product-of-sums expression can be implemented as the set of OR gates feeding into the single AND gate :

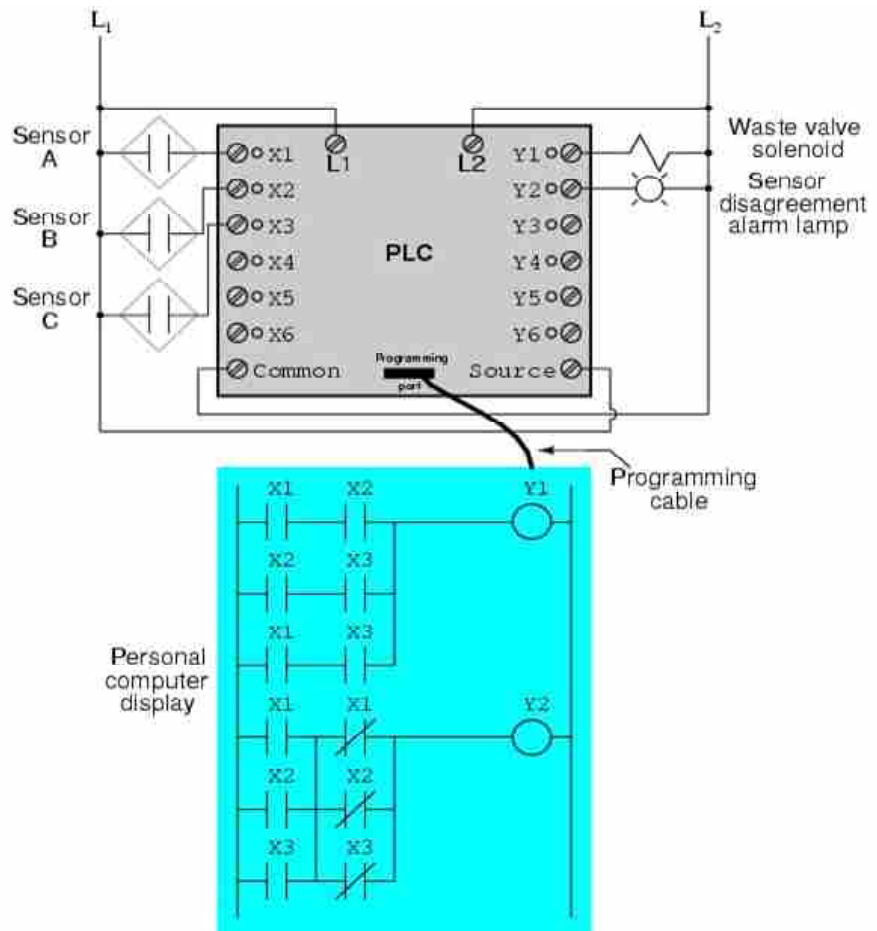


Correspondingly, whereas the sum-of-products expression could be implemented as the parallel collection of series-connected relay contacts, the product-of-sums expression can be implemented as the series collection of parallel-connected relay contacts.



The earlier two circuits represent different versions of the 'sensor disagreement' logic circuit only, not the 'good flame' detection circuit(s). The whole logic system would be the combination of both 'good flame' and 'sensor disagreement' circuits, shown on the same diagram.

Implemented in the Programmable Logic Controller (PLC), the entire logic system might resemble something like this :



As you can see, both the sum-of-products and products-of-sums standard Boolean forms are powerful tools when applied to truth tables. They allow us to derive the Boolean expression and finally, an actual logic circuit from nothing but the truth table, which is the written requirement for what we want the logic circuit to do.

To be able to go from the written requirement to an actual circuit using simple, deterministic procedures means that it is possible to automate the design process for the digital circuit. In other words, the computer could be programmed to design the custom logic circuit from the truth table requirement. The steps to take from the truth table to the final circuit are so unambiguous and direct that it requires little, if any, creativity or other original thought to execute them.

Bibliographic : <https://www.allaboutcircuit.com/textbook/digital/chpt-7>

□ Check Your Progress – 1 :

1. Explain Truth Table.

.....

2. Explain SOP and POS.

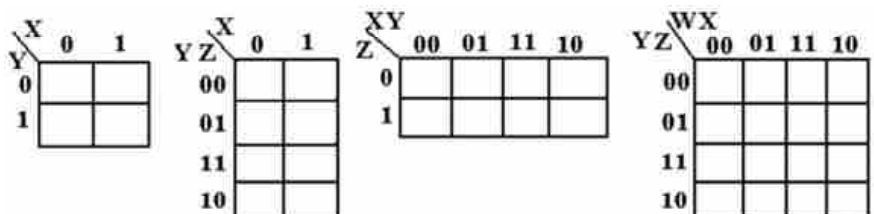
.....

7.3 Simplification of Boolean Equation Using K-Map :

The Karnaugh map, also recognized as the Veitch diagram (K-map or KV-map for short), was invented in 1950 by Maurice Karnaugh, the telecommunications engineer at Bell Labs. It is the very useful tool to facilitate Boolean algebraic expressions. Normally, general calculations are required to obtain the minimal expression of the Boolean job, but one can use the Karnaugh map instead.

Karnaugh maps make use of the human brain's excellent pattern-matching capability to decide which terms should be combined to get the simplest expression. In addition, K-maps permit rapid identification and elimination of potential race hazards, something that Boolean equations alone cannot do.

K Map for 1 2 3 4 variables



A Karnaugh map is an excellent aid for simplification of up to six variables but with more variables it becomes hard even for our brain to discern

optimal patterns. For expressions having more than 4 variables, the quine–Mccluskey algorithm, also called the method of prime implicants, should be used. This algorithm uses the deterministic approach to simplification of Boolean expressions.

Thus, following the steps of the algorithm ensures that the simplest expression can be found. This is especially useful for creating software programs that simplify any given Boolean expression. Karnaugh maps also help teach about Boolean jobs and minimisation.

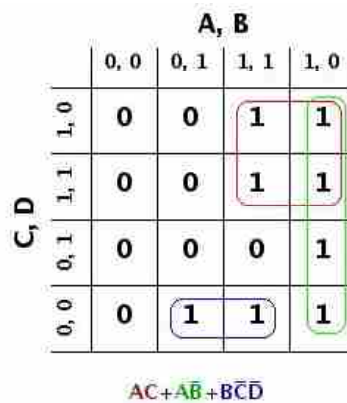
Consider the following job:

$$f(A, B, C, D) = E(4, 8, 9, 10, 11, 12, 14, 15)$$

This job has this truth table :

	<u>ABCDF</u>
0.	0000 0
1.	0001 0
2.	0010 0
3.	0011 0
4.	0100 1
5.	0101 0
6.	0110 0
7.	0111 0
8.	1000 1
9.	1001 1
10.	1010 1
11.	1011 1
12.	1100 1
13.	1101 0
14.	1110 1
15.	1111 1

The input variables can be combined in 16 different ways, so our Karnaugh map has to have 16 positions. The most suitable way to arrange this is in the 4 x 4 grid.



The binary digits in the map represent the job's output for any given combination of inputs. We write 0 in the upper leftmost corner of the map because $f = 0$ when $A = 0, B = 0, C = 1, D = 0$. Likewise, we mark the bottom right corner as 1 because $A = 1, B = 0, C = 0, D = 0$ gives $f = 1$.

After the Karnaugh map has been constructed, our next task is to find the minimal terms to use in the final expression. These terms are found by surrounding the 1's in the map. The encircling can only include 2^n fields, where n is an integer

≥ 0 (1, 2, 3, 4...). They should be as large as possible. The optimal encirclings in this map are distinct by the green, red and blue lines.

For each of these encircling, we find those variables that have the similar state in each of the fields in the encircling. For the first encircling (the red one) we find that:

- The variable maintains the same state (1) in the whole encircling; consequently, it should be included in the term for the red encircling.
- Variable B does not maintain the same state (it shifts from 1 to 0) and should therefore be excluded.
- Likewise, D changes but C does not.

The first term becomes AC .

For the green encircling we see that A and B maintains the similar state, C and D changes. But B is 0 and has to be invalid before it can be included.

The second term becomes AB' .

By working the blue encircling the same way we find the term $BC'D'$ and our ultimate expression for the job is ready : $AC + AB' + BC'D'$.

The converse of the job is solved in the same way by encircling the 0's instead.

It is worth mentioning that the number of product terms for an encircling P is :

$$P = 2\log(n/x)$$

Where, n is the number of variables in the Karnaugh map and x the number of fields encircled

□ Check Your Progress – 2 :

1. Define the term truth table. Using truth table how we can prove logical equivalence ? Explain with example.

.....

2. What is the difference between SOP and POS ? Explain with example.

.....

-
.....
.....
3. \Leftrightarrow is used for _____
 (a) Conditional statements (b) And gate
 (c) OR gate (d) Biconditional statements
 4. if there is a rain then use an umbrella is _____ type statement.
 (a) Conditional statements (b) And gate
 (c) OR gate (d) Biconditional statements
 5. P : I like ice cream.
 Q : I like Pizza
 P _____ Q
 (a) \wedge (b) \Leftrightarrow (c) \Rightarrow (d) none
 6. K Map is used to _____ boolean expression
 (a) Expand (b) Minimize (c) Sum (d) All of above
 7. Which highest value you can express using K map for 3 variables ?
 (a) 4 (b) 5 (c) 7 (d) 15
 8. Which highest value you can express using K map for 4 variables ?
 (a) 4 (b) 5 (c) 7 (d) 15
 9. Which highest value you can express using K map for 1 variables ?
 (a) 1 (b) 0 (c) 2 (d) 3

7.4 Let Us Sum Up :

Sum-of-products, or SOP, Boolean expressions may be generated from truth tables quite easily, by determining which rows of the table have an output of 1, writing one product term for each row and finally summing all the product terms. This creates the Boolean expression representing the truth table as the whole.

Sum-of-products expressions lend themselves well to implementation as the set of AND gates (products) feeding into the single OR gate (sum). Product-of-sums, or POS, Boolean expressions may also be generated from truth tables quite easily, by determining which rows of the table have an output of 0, writing one sum term for each row and finally multiplying all the sum terms. This creates the Boolean expression representing the truth table as the whole. Product-of-sums expressions lend themselves well to implementation as the set of OR gates (sums) feeding into the single AND gate (product).

A **Karnaugh map (K-map)** is a pictorial technique used to minimize Boolean expressions without having to use Boolean algebra theorems and equation controls. A **K-map** can be thought of as a special version of a truth table. Using a **K-map**, expressions with two to four variables are easily minimized.

7.5 Answer for Check Your Progress :

☐ **Check Your Progress 1 :**

See Section 7.2

☐ **Check Your Progress 2 :**

1 : See Section 7.2	2 : See Section 7.2	3 : d
4 : a	5 : a	6 : b
7 : c	8 : d	9 : a

7.6 Glossary :

1. **Karnaugh Map :** The **Karnaugh map** (KM or K-map) is a method of simplifying Boolean algebra expressions.
2. **SOP :** Sum of product
3. **POS :** Product of sum

7.7 Assignment :

1. Explain sum of product with example.
2. Explain product of sum with example ?
3. Explain Karnaugh map for 1, 2, 3 variables.

7.8 Activities :

Generate truth table for following.

- a. If the input file exists, then an error message is not generated.
- b. My program contains no bugs and it produces correct output

7.9 Case Study :

In multiple error correcting codes, How Karnaugh Map is useful ?

7.10 Further Reading :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar
3. <https://www.allaboutcircuit.com/textbook>

BLOCK SUMMARY :

In above units we have seen logic gates, Boolean identities, De'Morgan's Law, Karnaugh map, truth tables, logical connectives, Boolean equations etc.

Sum-of-products, or SOP, Boolean expressions may be generated from truth tables quite easily, by determining which rows of the table have an output of 1, writing one product term for each row and finally summing all the product terms. This creates the Boolean expression representing the truth table as the whole.

Sum-of-products expressions lend themselves well to implementation as the set of AND gates (products) feeding into the single OR gate (sum). Product-of-sums, or POS, Boolean expressions may also be generated from truth tables quite easily, by determining which rows of the table have an output of 0, writing one sum term for each row and finally multiplying all the sum terms. This creates the Boolean expression representing the truth table as the whole. Product-of-sums expressions lend themselves well to implementation as the set of OR gates (sums) feeding into the single AND gate (product).

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. What do you mean by basic gates ?
2. Describe the cumulative laws of Boolean Algebra.
3. What do you understand by Boolean rules ?
4. Describe idempotent property.

❖ **Long Questions :**

1. Explain types of universal gates in detail.
2. How K Map is used to represent Boolean expression ?
3. List all logical connectors of Boolean algebra with example.
4. Convert the following SOP to POS

$$AB'C + AB + A'BC$$

❖ **Enrolment No. :**

1. How many hours did you need for studying the units ?

Unit No.	4	5	6	7
No. of Hrs.				

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Language and Style	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Illustration used (Diagram, tables etc)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Conceptual Clarity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Check your progress Quest	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Feed back to CYP Question	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....



Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-203/
DCAR-203

Digital Electronics and Computer Organization

BLOCK 3 : DIGITAL COMPONENT

UNIT 8 ARITHMETIC LOGIC UNIT

UNIT 9 DIGITAL COMPONENTS

UNIT 10 ADDRESS, DATA & CONTROL BUS

DIGITAL COMPONENT

Block Introduction :

In this block, we will learn and study about basic of the CPU in which we will learn about sub unit of the CPU that is ALU and CU. The term 'ALU' is used to perform mathematical operation, logical operations and logical shifts operations. The CU interprets and executes instructions in memory.

In this block, we will learn and study about basic computer circuit and how that circuit is integrated with each other and all the task of the computer is carried on. Here, the student will be learned about integrated circuit and as a part of integrated circuit we will go through Half-Adder, Full-Adder, Parallel Binary Adder, Decoder, Encoder, and Multiplexer.

In this block, we will learn and study about the Bus system. Bus system plays an important role to move the data from one part to another part of the system. Each channel, called a bus, allows the various devices both inside and attached to the system unit to communicate with one another. Buses are used to transfer bits from input devices to memory, from memory to the processor, from the processor to memory, and from memory to output or storage devices.

Block Objectives :

After learning this block, you will be able to understand :

- Idea about ALU
- Half Adder & Full Adder
- Parallel Binary Adder
- Binary Adder-Subtractor
- 1's and 2's Complement System
- Idea about Integrated Circuit
- Decoders, Encoders and Multiplexers
- Memory Unit
- Idea about Bus
- Types of Bus

Block Structure :

Unit 8 : Arithmetic Logic Unit

Unit 9 : Digital Components

Unit 10 : Address, Data & Control Bus

UNIT STRUCTURE

- 8.0 Learning Objectives
- 8.1 Introduction
- 8.2 Construction of ALU
- 8.3 Adder
 - 8.3.1 Binary Half Adder
 - 8.3.2 Binary Full Adder
- 8.4 Parallel Binary Adder
- 8.5 Binary Adder–Subtractor
- 8.6 Addition in 1's and 2's Complement System
- 8.7 Let Us Sum Up
- 8.8 Suggested Answer for Check Your Progress
- 8.9 Glossary
- 8.10 Assignment
- 8.11 Activities
- 8.12 Case Study
- 8.13 Further Readings

8.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Construction of ALU
- Binary Half Adder Symbol, Truth–Table, Circuit
- Binary Full Adder Symbol, Truth–Table, Circuit
- Parallel Binary Adder
- Binary Adder–Subtractor
- Addition in 1's and 2's complement system

8.1 Introduction :

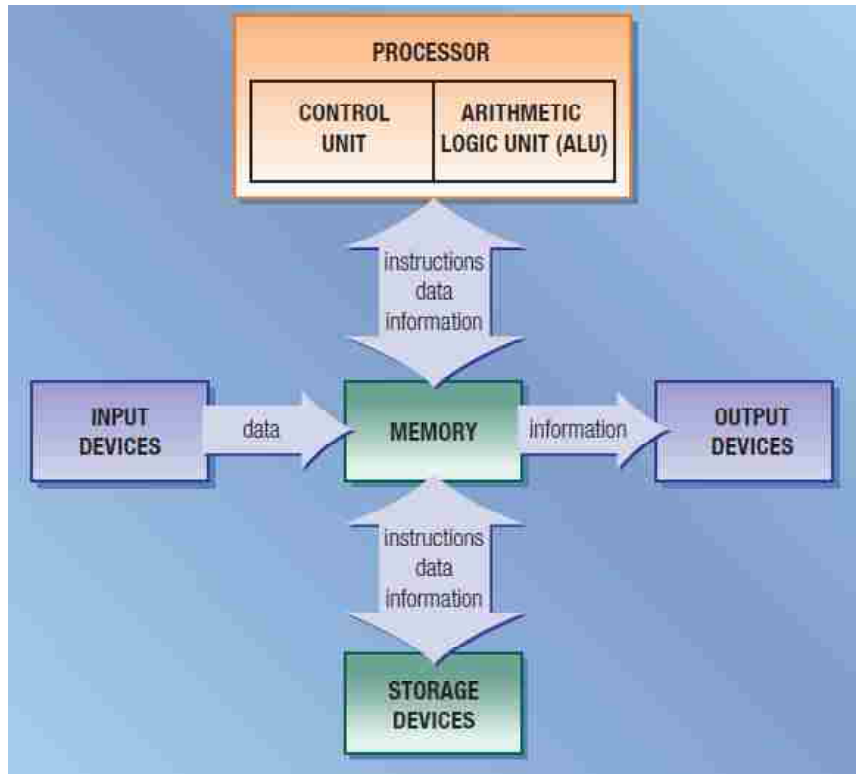
The term Central Processing Unit (CPU) is the brain of the computer system. CPU contains two sub unit that is Arithmetic Logic Unit (ALU) and Control Unit (CU). The term 'ALU' is used to perform mathematical operation, logical operations and logical shifts operations. The CU interprets and executes instructions in memory.

- The processor, also called the central processing unit (CPU), interprets and carries out the basic instructions that operate a computer.



Digital Electronics and Computer Organization

- On larger computers, such as mainframes and supercomputers, the various functions performed by the processor extend over many separate chips and often multiple circuit boards.
- On a personal computer, all functions of the processor usually are on a single chip.
- A multi-core processor is a single chip with two or more separate processor cores.

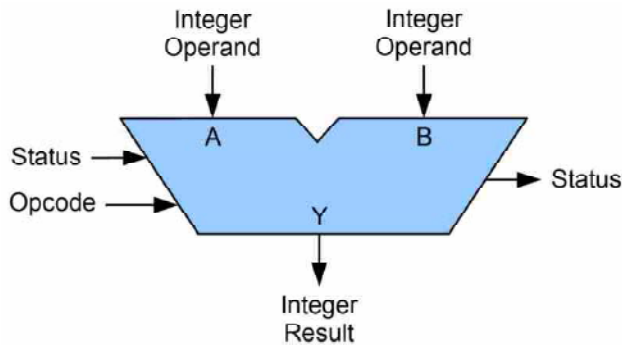


- Processors contain a control unit and an arithmetic logic unit (ALU). These two components work together to perform processing operations.
- The control unit interprets and executes instructions in memory, and the arithmetic logic unit performs calculations on the data in memory.
- Resulting information is stored in memory, from which it can be sent to an output device or a storage device for future access, as needed.

8.2 Construction of ALU :

- A sub unit within a computer's central processing unit.
- ALU full form is Arithmetic Logic Unit, takes the data from Memory registers.
- An arithmetic logic unit (ALU) is a major component of the central processing unit of a computer system.
- It does all processes related to arithmetic and logic operations that need to be done on instruction words.
- In some microprocessor architectures, the ALU is divided into the arithmetic unit (AU) and the logic unit (LU).
- An ALU can be designed by engineers to calculate many different operations.

- ALU contains the logical circuit to perform mathematical operations like subtraction, addition, multiplication, division, logical operations and logical shifts on the values held in the processors registers or its accumulator.



Arithmetic Logic Unit – ALU

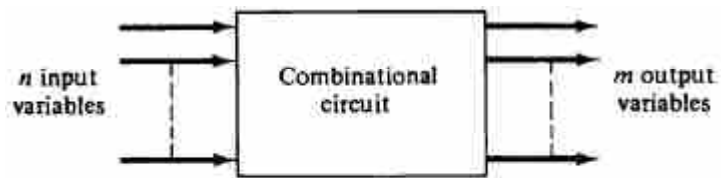
- ALU is also known as an Integer Unit (IU). The arithmetic logic unit is that part of the CPU that handles all the calculations the CPU may need. Most of these operations are logical in nature.
- Depending on how the ALU is designed, it can make the CPU more powerful, but it also consumes more energy and creates more heat.
- Therefore, there must be a balance between how powerful and complex the ALU is and how expensive the whole unit becomes.
- This is why faster CPUs are more expensive, consume more power and dissipate more heat.
- Different operation as carried out by ALU can be categorized as follows
 - ✓ **Logical operations** – These include operations like AND, OR, NOT, XOR, NOR, NAND, etc.
 - ✓ **Bit-Shifting Operations** – This pertains to shifting the positions of the bits by a certain number of places either towards the right or left, which is considered a multiplication or division operations.
 - ✓ **Arithmetic operations** – This refers to bit addition and subtraction. Although multiplication and division are sometimes used, these operations are more expensive to make. Multiplication and subtraction can also be done by repetitive additions and subtractions respectively.

❑ **Check Your Progress – 1 :**

1. ALU stands for _____.
 - (a) Arithmetic Logical Unit
 - (b) Arithmetic Logic Unit
 - (c) Arithmetic Legal Unit
 - (d) Arithmetic Large Unit
2. Logical circuit of ALU perform mathematical operation like _____.
 - (a) Addition
 - (b) Subtraction
 - (c) Multiplication
 - (d) All of Above

8.3 Adder :

- A combinational circuit is a connected arrangement of logic gates with a set of inputs and outputs.
- A block diagram of a combinational circuit is as below :



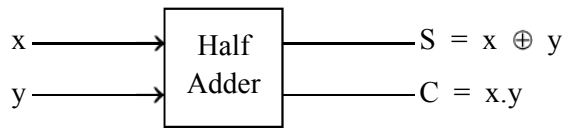
Block Diagram of Combinational Circuit

- The n binary input variables come from an external source, the m binary output variables go to an external destination, and in between there is an interconnection of logic gates.
 - A combinational circuit transforms binary information from the given input data to the required output data.
 - Combinational circuits are employed in digital computers for generating binary control decisions.
 - A combinational circuit can be described by a truth-table showing the binary relationship between the n input variables and the m output variables.
 - The design of combinational circuit procedure involves the following steps :
 1. The problem is stated.
 2. The input and output variables are assigned letter symbols.
 3. The truth-table that defines the relationship between inputs and outputs is derived.
 4. The simplified Boolean functions for each output are obtained.
 5. The logic diagram is drawn.
 - Examples of some important combinational circuits are :
 - Comparator
 - Multiplexer
 - Decoder/Demultiplexer
 - Shifter, Full-Adder
 - In electronics an Adder is a digital circuit that perform addition of number.
 - The most basic Arithmetic operation is the addition of two binary digits. The simple addition consists of four operation which are as follows :

$$\begin{array}{r} 0 + 0 = 0 \\ 0 + 1 = 1 \\ 1 + 0 = 1 \\ 1 + 1 = 1 \quad 0 \end{array}$$
 - There are two types of Adder which are as follows:
- 8.3.1 Binary Half Adder :**
- A half-adder is logic circuit that performs addition of two binary digit.
 - A half-adder produces a sum of two variables & a carry value which are both binary digits. A half adder can add two bits.
 - It has 2 inputs x & y and 2 outputs S & C. S stands for Sum and C stands for Carry.

- Here, S is two-bit EXOR of x & y so that is $S = x \oplus y$ and C is AND of x & y so that is $C = x.y$.

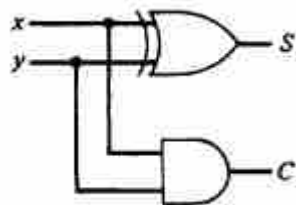
- **Symbol**



- **Truth-Table**

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

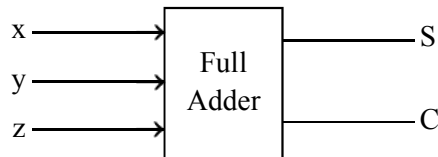
- **Circuit**



8.3.2 Binary Full Adder :

- Full-Adder is a combinational circuit that perform arithmetic sum. Full-Adder can add three bits.
- It has three inputs x, y, & z and two output S & C. S for sum and C for carry.
- Here, we have $S = (x \oplus y) \oplus C$ & $C = AB + (A \oplus B).C$

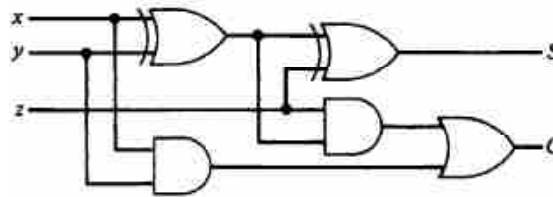
- **Symbol**



- **Truth-Table**

Inputs			Outputs	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

• **Circuit**



The output carry is designated as C-OUT and the normal output is represented as S which is 'SUM'.

With the above **full adder truth-table**, the implementation of a full adder circuit can be understood easily. The SUM 'S' is produced in two steps:

1. By XORing the provided inputs 'A' and 'B'
2. The result of A XOR B is then XORed with the C-IN

This generates SUM and C-OUT is true only when either two of three inputs are HIGH, then the C-OUT will be HIGH. So, we can implement a full adder circuit with the help of two half adder circuits. Initially, the half adder will be used to add A and B to produce a partial Sum and a second-half adder logic can be used to add C-IN to the Sum produced by the first half adder to get the final S output.

If any of the half adder logic produces a carry, there will be an output carry. So, C-OUT will be an OR function of the half-adder Carry outputs.

□ **Check Your Progress – 2 :**

1. Explain Half Adder with Truth-table and circuit.

.....

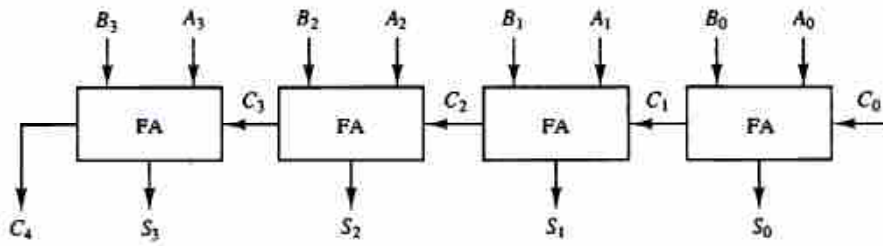
2. Write a note on Full Adder.

.....

8.4 Parallel Binary Adder :

- To implement the add microoperation with hardware, we need the registers that hold the data and the digital component that performs the arithmetic addition.
- The digital circuit that forms the arithmetic sum of two bits and a previous carry is called a full-adder.
- The digital circuit that generates the arithmetic sum of two binary numbers of any length is called a binary adder.
- The binary adder is constructed with full-adder circuits connected in cascade, with the output carry from one full-adder connected to the input carry of the next full-adder.

- Interconnections of four full-adders (FA) to provide a 4-bit binary adder as follows :



4-bit Binary Adder

- An n-bit binary adder requires n full-adders. The output carry from each full-adder is connected to the input carry of the next-high-order full-adder.
- The n data bits for the A inputs come from one register (such as R1), and then data bits for the B inputs come from another register (such as R2).
- The sum can be transferred to a third register or to one of the source registers (R1 or R2), replacing its previous content.

Check Your Progress – 3 :

1. Explain 4-bit Binary Adder with diagram.

.....

2. FA stands for _____.

- (a) Full Adder
- (b) Full Arithmetic
- (c) Full Addition
- (d) Full Assignment

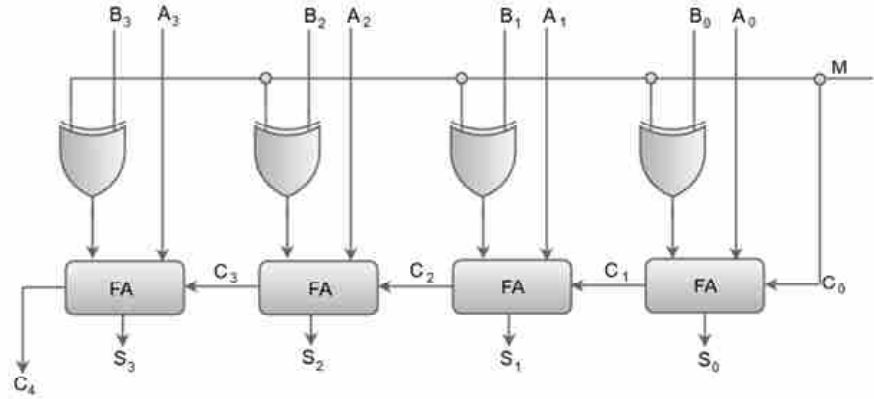
3. The Binary Adder is constructed by _____.

- (a) Half-Adder
- (b) Adder
- (c) Full-Adder
- (d) Complement

8.5 Binary Adder-Subtractor :

- The subtraction of binary numbers can be done most conveniently by means of complements.
- The subtraction $A - B$ can be done by taking the 2's complement of B and adding it to A.
- The 2's complement can be obtained by taking the 1's complement and adding one to the least significant pair of bits.
- The 1's complement can be implemented with inverters and a one can be added to the sum through the input carry.
- The addition and subtraction operations can be combined into one common circuit by including an exclusive-OR gate with each full-adder.

– A 4-bit adder–subtractor circuit is shown in below figure :



4–Bit Adder Subtractor

- The mode input M controls the operation. When M = 0 the circuit is an adder and when M = 1 the circuit becomes a subtractor.
- Each exclusive–OR gate receives input M and one of the inputs of B. When M = 0, we have $B \oplus 0 = B$.
- The full–adders receive the value of B, the input carry is 0, and the circuit performs A plus B. When M = 1, we have $B \oplus 1 = B'$ and $C_0 = 1$.

- The B inputs are all complemented and a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B.

- **Advantages of parallel Adder/Subtractor –**

- The parallel adder/subtractor performs the addition operation faster as compared to serial adder/subtractor.
- Time required for addition does not depend on the number of bits.
- The output is in parallel form i.e. all the bits are added/subtracted at the same time.
- It is less costly.

- **Disadvantages of parallel Adder/Subtractor –**

- Each adder has to wait for the carry which is to be generated from the previous adder in chain.
- The propagation delay (delay associated with the travelling of carry bit) is found to increase with the increase in the number of bits to be added.

□ **Check Your Progress – 4 :**

1. Explain 4–bit Adder–Subtractor with diagram.

.....

.....

.....

.....

.....

2. The subtraction of binary number can be done by _____.

- (a) Half–Adder (b) Full–Adder (c) Complement (d) Adder

8.6 Addition in 1's and 2's Complement System :
--

- Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation.
- There are two types of complements for each base r system: the $(r - 1)$'s complement and the (r) 's complement.
- When the value of the base r is substituted in the name, the two types are referred to as the 2's and 1's complement for binary numbers and the 10's and 9's complement for decimal numbers.

$(r - 1)$'s Complement

- Given a number N in base r having n digits, the $(r - 1)$'s complement of N is defined as $(r^n - 1) - N$.
- For decimal numbers $r = 10$ and $r - 1 = 9$, so the 9's complement of N is $(10^n - 1) - N$.
- Now, 10^n represents a number that consists of a single 1 followed by n 0's. $10^n - 1$ is a number represented by n 9's.
- For example, $(r^n - 1) - N = r = 10$, $n = 4$, $(10^4 - 1) - N = (10000 - 1) = 9999$
- Suppose $n = 4$ we have $10^4 = 10000$ and $10^4 - 1 = 9999$.
- It follows that the 9's complement of a decimal number is obtained by subtracting each digit from 9.
- For example :

$$\begin{array}{r}
 \text{9's complement of } 546700 \text{ is} = 999999 \\
 \phantom{\text{9's complement of } 546700 \text{ is}} - 546700 \\
 \phantom{\text{9's complement of } 546700 \text{ is}} 453299
 \end{array}$$

- For binary numbers, $r = 2$ and $r - 1 = 1$, so the 1's complement of N is $(2^n - 1) - N$.
- 2^n is represented by a binary number that consists of a 1 followed by n 0's. $2^n - 1$ is a binary number represented by n 1's.
- For example
- Suppose $n = 4$, we have $2^4 = (10000)_2$ and $2^4 - 1 = (1111)_2$. $2^4 = 16$ so, $16 - 1 = 15(1111)$.
- Thus, the 1's complement of a binary number is obtained by subtracting each digit from 1.
- However, the subtraction of a binary digit from 1 causes the bit to change from 0 to 1 or from 1 to 0.
- Therefore, the 1's complement of a binary number is formed by changing 1's into 0's and 0's into 1's.
- For example,

$$\begin{array}{r}
 \text{1's complement of } 1011001 \text{ is} = 1111111 \\
 \phantom{\text{1's complement of } 1011001 \text{ is}} - 1011001 \\
 \phantom{\text{1's complement of } 1011001 \text{ is}} 0100110
 \end{array}$$

8.9 Glossary :

1. **Adder :** In electronics an Adder is a digital circuit that perform addition of number.
2. **Half-Adder :** A half-adder is logic circuit that performs addition of two binary digit.
3. **Full-Adder :** Full-Adder is a combinational circuit that perform arithmetic sum. Full-Adder can add three bits.

8.10 Assignment :

1. Explain 1's complement and 2's complement with example.

8.11 Activities :

1. Take one example with minimum 4 variable and simplify it by half adder and full adder.

8.12 Case Study :

1. Find out in which areas we are using ALUs.

8.13 Further Readings :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar

UNIT STRUCTURE

- 9.0 Learning Objectives
- 9.1 Introduction
- 9.2 Integrated Circuits
- 9.3 Decoders and its Expansion
- 9.4 Encoders
- 9.5 Multiplexer and its Expansion
- 9.6 Memory Unit
- 9.7 Let Us Sum Up
- 9.8 Suggested Answer for Check Your Progress
- 9.9 Glossary
- 9.10 Assignment
- 9.11 Activities
- 9.12 Case Study
- 9.13 Further Readings

9.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Integrated Circuit
- Decoders and its expansion
- Encoder
- Multiplexer and its expansion
- Memory Unit

9.1 Introduction :

An Encoder is a device that converts the active data signal into a coded message format or it is a device that converts analogue signal to digital signals. A code may provide any of a digit of purposes such as compressing information for transmission or storage, encrypting or adding redundancies to an input code, or translating from one code to another. In digital electronics this would mean that an encoder is a numerous-input, numerous-output logic circuit ($2n - n$).

A decoder is a device that generates the original signal as output from the coded input signal and converts n lines of input into $2n$ lines of output. A decoder is a device which does a reverse of an encoder, undoing an encoding so that an original information can be retrieved. A same method makes use of d to encode is usually just reversed in order to decode. In digital electronics this would mean that a decoder is a numerous-input, numerous-output logic circuit ($n - 2n$).

9.2 Integrated Circuits :

- Digital circuits are constructed with integrated circuits.
- An integrated circuit (abbreviated IC) is a small silicon semiconductor crystal, called a chip. Chip containing the electronic components for the digital gates.
- The various gates are interconnected inside the chip to form the required circuit.
- The chip is mounted in a ceramic or plastic container, and connections are welded by thin gold wires to external pins to form the integrated circuit.
- The number of pins may range from 14 in a small IC package to 100 or more in a larger package.
- Each IC has a numeric designation printed on the surface of the package for identification.

❖ **Types of IC :**

- There are four types of IC which are as follows :

1. **SSI**
2. **MSI**
3. **LSI**
4. **VLSI**

1. SSI :

- SSI stands for Small–Scale Integration. SSI devices contain several independent gates in a single package.
- The inputs and outputs of the gates are connected directly to the pins in the package.
- The number of gates is usually less than 10 and is limited by the number of pins available in the IC.

2. MSI :

- MSI stands for Medium–Scale Integration.
- MSI devices have a complexity of approximately 10 to 200 gates in a single package.
- They usually perform specific elementary digital functions such as decoders, adders, and registers.

3. LSI :

- LSI stands for Large–Scale Integration.
- LSI devices contain between 200 and a few thousand gates in a single package.
- They include digital systems, such as processors, memory chips, and programmable modules.

4. VLSI :

- VLSI stands for Very Large–Scale Integration.
- VLSI devices contain thousands of gates within a single package.
- Examples are large memory arrays and complex microcomputer chips.

□ **Check Your Progress – 1 :**

1. SSI stands for _____.

(a) Small Scale Integration	(b) System Scale Integration
(c) Structure Scale Integration	(d) Service Scale Integration
2. MSI stands for _____.

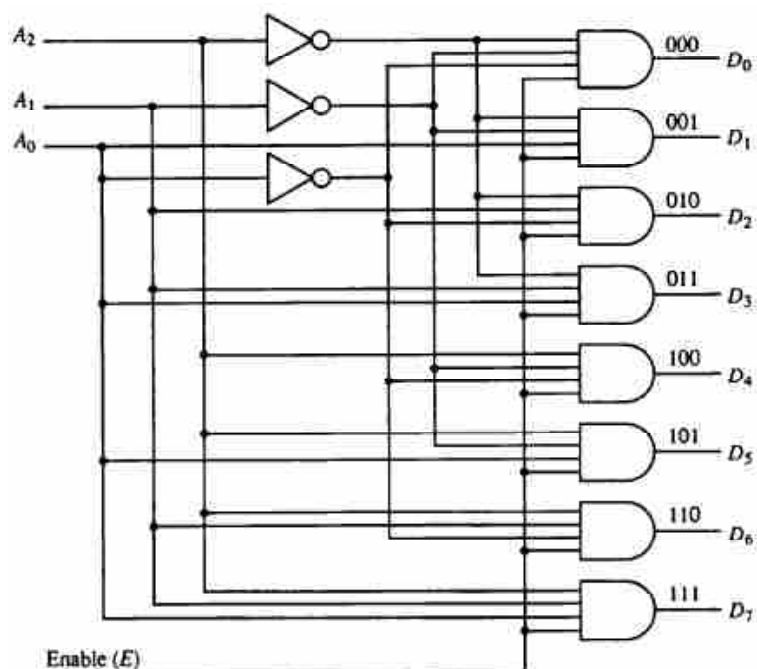
(a) Multiplexer Scale Integration	(b) Mode Scale Integration
(c) Multiple Scale Integration	(d) Medium Scale Integration
3. LSI stands for _____.

(a) Last Scale Integration	(b) Large Scale Integration
(c) Logical Scale Integration	(d) List Scale Integration
4. VLSI stands for _____.

(a) Very List–Scale Integration	(b) Very Logical–Scale Integration
(c) Very Large–Scale Integration	(d) Very Last–Scale Integration

9.3 Decoders and its Expansion :

- Discrete quantities of information are represented in digital computers with binary codes.
- A binary code of n bits is capable of representing up to 2^n distinct elements of the coded information.
- A decoder is a combinational circuit that converts binary information from the n coded inputs to a maximum of 2^n unique outputs.
- The decoders presented in this section are called n –to– m –line decoders, where $m \leq 2^n$. Their purpose is to generate the 2^n (or fewer) binary combinations of the n input variables.
- A decoder has n inputs and m outputs and is also referred to as an $n \times m$ decoder.
- The logic diagram of 3–to–8–line decoder is as follows :



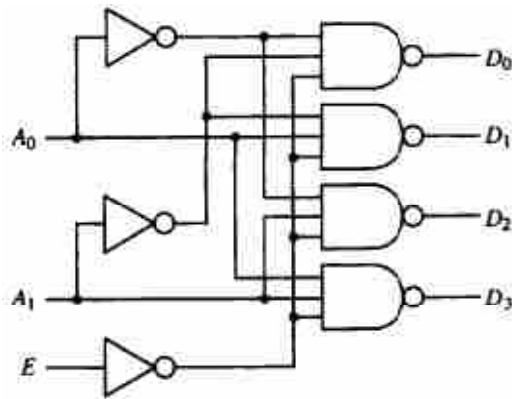
3–to–8 Line Decoder

- The three data inputs, A_0 , A_1 , and A_2 , are decoded into eight outputs, each output representing one of the combinations of the three binary input variables.
- The three inverters provide the complement of the inputs, and each of the eight AND gates generate one of the binary combinations.
- A particular application of this decoder is a binary-to-octal conversion.
- The input variables represent a binary number and the outputs represent the eight digits of the octal number system.
- However, a 3-to-8-line decoder can be used for decoding any 3-bit code to provide eight outputs, one for each combination of the binary code.
- Commercial decoders include one or more enable inputs to control the operation of the circuit.
- The decoder is enabled when E is equal to 1 and disabled when E is equal to 0.
- The operation of the decoder can be clarified using the truth table as follows :

Enable	Inputs			Outputs							
	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	×	×	×	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Truth-Table of 3-to-8 Line Decoder

- When the enable input, E is equal to 0, all the outputs are equal to 0 regardless of the values of the other three data inputs.
- The three Xs in the table designate don't-care conditions.
- When the enable input is equal to 1, the decoder operates in a normal fashion.
- **NAND Gate Decoder :**
- Some decoders are constructed with NAND instead of AND gates.
- Since a NAND gate produces the AND operation with an inverted output, it becomes more economical to generate the decoder outputs in their complement form.
- A 2-to-4-line decoder with an enable input constructed with NAND gates.



Logic Diagram

- The circuit operates with complemented outputs and a complemented enable input E.
- The decoder is enabled when E is equal to 0.

Enable	Inputs		Outputs			
	A ₁	A ₀	D ₀	D ₁	D ₂	D ₃
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	×	×	1	1	1	1

Truth-Table

- As indicated by the truth table, only one output is equal to 0 at any given time; the other three outputs are equal to 1.
- The circuit is disabled when E is equal to 1, regardless of the values of the other two inputs.
- When the circuit is disabled, none of the outputs are selected and all outputs are equal to 1.

□ Check Your Progress – 2 :

1. What is Decoder ? Explain 3-to-8 Line decoder.

.....

.....

.....

.....

.....

9.4 Encoders :

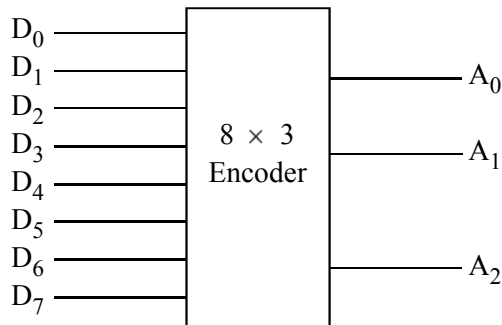
- An encoder is a digital circuit that performs the inverse operation of a decoder.
- An encoder has 2n (or less) input lines and n output lines.
- The output lines generate the binary code corresponding to the input value.

- An example of an encoder is the octal-to-binary encoder, it has eight inputs, one for each of the octal digits, and three outputs that generate the corresponding binary number.

➤ **8-to-3 Binary Encoder OR Octal-to-Binary Encoder**

• **Block Diagram of Encoder**

- In 8-to-3 binary encoder there are 8 inputs & 3 outputs.



Block Diagram of Encoder

• **Truth-Table :**

Inputs								Outputs		
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	A ₂	A ₁	A ₀
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Truth-Table of 8-to-3 OR Octal-to-Binary Encoder

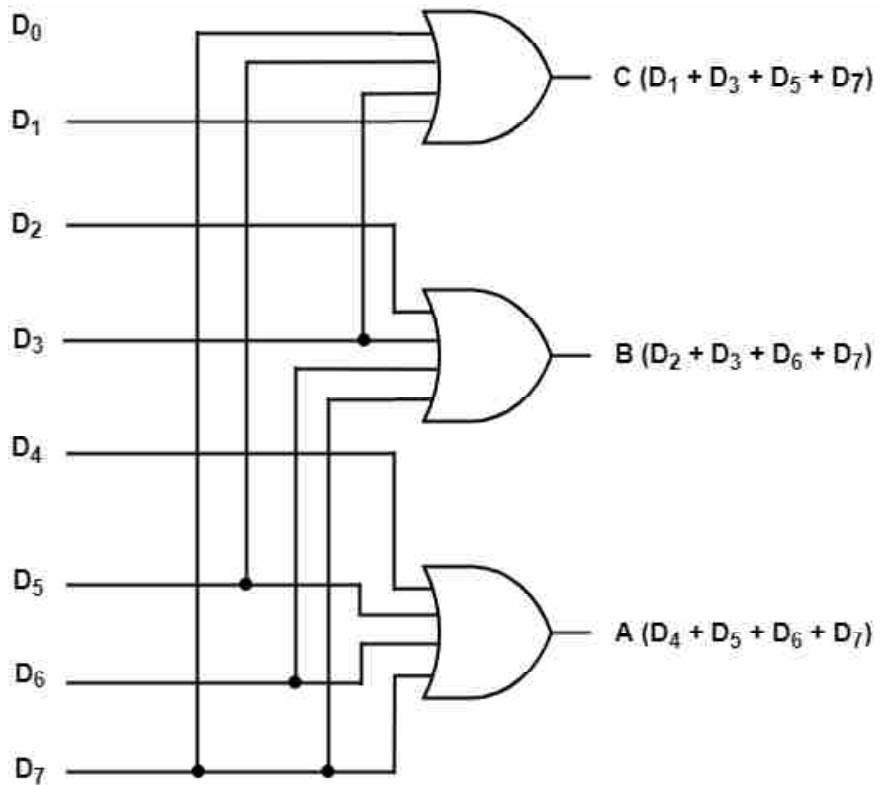
- It is assumed that only one input has a value of 1 at any given time; otherwise, the circuit has no meaning.
- The encoder can be implemented with OR gates whose inputs are determined directly from the truth table.
- Output A₀ = 1 if the input octal digit is 1 or 3 or 5 or 7. Similar conditions apply for the other two outputs.
- These conditions can be expressed by the following Boolean functions:

$$A_0 = D_1 + D_3 + D_5 + D_7$$

$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

• **8-to-3 OR Octal-to-Binary Encoder Circuit :**



□ **Check Your Progress – 3 :**

1. Explain 8-to-3 line encoder in detail.

.....

.....

.....

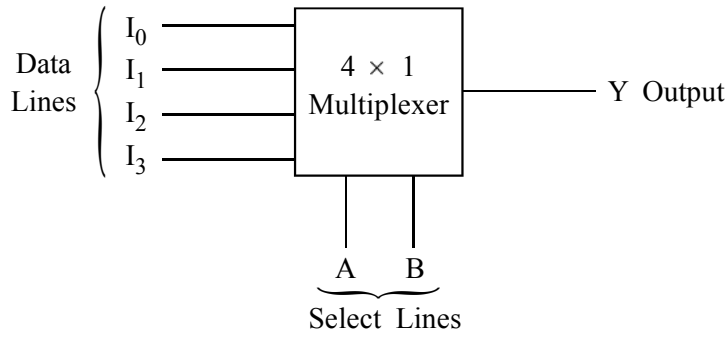
.....

.....

9.5 Multiplexer and its Expansion :

- Multiplexer means transmitting a large amount of information into a small number of lines.
 - A multiplexer is a combinational circuit that receives binary information from one of 2^n input data lines and directs it to a single output line.
 - The selection of a particular input data line for the output is determined by a set of selection inputs.
 - A 2^n -to-1 multiplexer has 2^n input data lines and n input selection lines whose bit combinations determine which input data are selected for the output.
- **4-line to 1-line Multiplexer :**
- In 4-line to 1-line we have 4 input data line $C_0, C_1, C_2,$ & C_3 .
 - We need 2 additional input or select input.
 - To select which of the input appear at the output it is also called 4-line to 1-line multiplexer with AND & OR gate.

- Symbol :

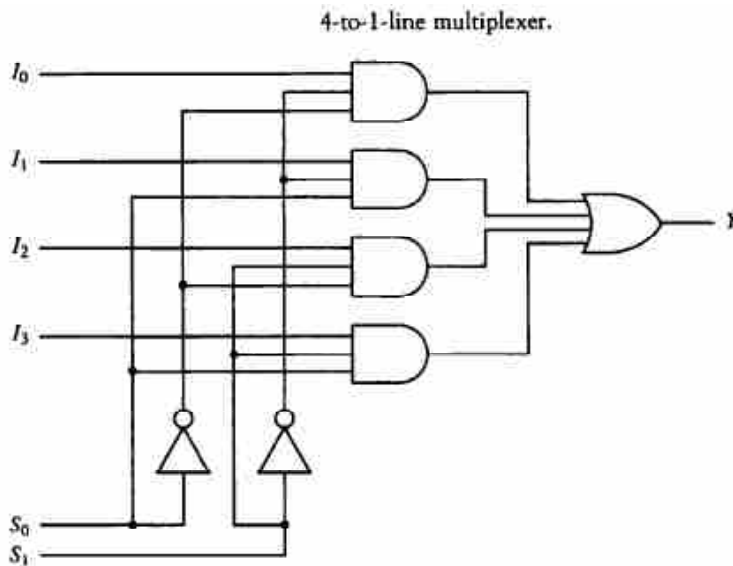


4-line to 1-line Multiplexer

- Function-Table :

Input		Output
S0(A)	S1(B)	Y
0	0	I ₀
0	1	I ₁
1	0	I ₂
1	1	I ₃

- Circuit of 4-line to 1-line Multiplexer :



□ **Check Your Progress – 4 :**

1. Write a note on Multiplexer.

.....

.....

.....

.....

.....

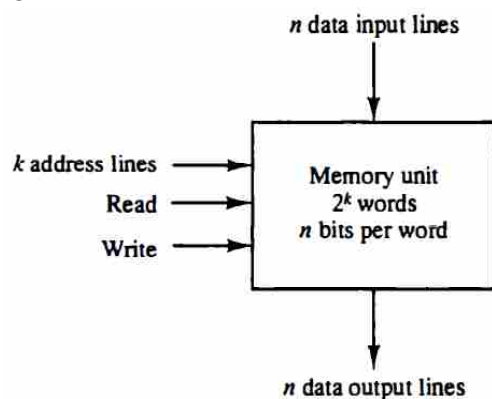
9.6 Memory Unit :

- A memory unit is a collection of storage cells together with associated circuits needed to transfer information in and out of storage.

- The memory stores binary information in groups of bits called words.
- A word in memory is an entity of bits that move in and out of storage as a unit.
- A memory word is a group of 1's and 0's and may represent a number, an instruction code, one or more alphanumeric characters, or any other binary-coded information.
- A group of eight bits is called a byte.
- Most computer memories use words whose number of bits is a multiple of 8. Thus a 16-bit word contains two bytes, and a 32-bit word is made up of four bytes.
- The capacity of memories in commercial computers is usually stated as the total number of bytes that can be stored.
- The internal structure of a memory unit is specified by the number of words it contains and the number of bits in each word.
- Special input lines called address lines select one particular word. Each word in memory is assigned an identification number, called an address.
- The selection of a specific word inside the memory is done by applying the k-bit binary address to the address lines.
- Two major types of memories are used in computer system:

- **Random-Access Memory :**

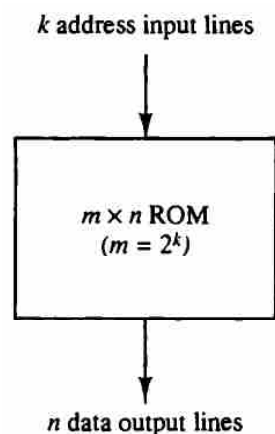
- In random-access memory (RAM) the memory cells can be accessed for information transfer from any desired random location.
- That is, the process of locating a word in memory is the same and requires an equal amount of time no matter where the cells are located physically in memory: thus, the name "random access."
- Communication between a memory and its environment is achieved through data input and output lines, address selection lines, and control lines that specify the direction of transfer.
- A block diagram of a RAM unit is as follows :



Block Diagram of Random-Access Memory (RAM)

- The n data input lines provide the information to be stored in memory, and the n data output lines supply the information coming out of memory.
- The k address lines provide a binary number of k bits that specify a particular word chosen among the 2^k available inside the memory.

- The two operations that a random-access memory can perform are the write and read operations.
- The write signal specifies a transfer-in operation and the read signal specifies a transfer-out operation.
- On accepting one of these control signals, the internal circuits inside the memory provide the desired function.
- The steps that must be taken for the purpose of transferring a new word to be stored into memory are as follows :
 1. Apply the binary address of the desired word into the address lines.
 2. Apply the data bits that must be stored in memory into the data input lines.
 3. Activate the write input.
- The memory unit will then take the bits presently available in the input data lines and store them in the word specified by the address lines.
- The steps that must be taken for the purpose of transferring a stored word out of memory are as follows :
 1. Apply the binary address of the desired word into the address lines.
 2. Activate the read input.
- The memory unit will then take the bits from the word that has been selected by the address and apply them into the output data lines.
- The content of the selected word does not change after reading.
- **Read-Only Memory :**
 - As the name implies, a read-only memory (ROM) is a memory unit that performs the read operation only; it does not have a write capability.
 - A RAM is a general-purpose device whose contents can be altered during the computational process; a ROM is restricted to reading words that are permanently stored within the unit.
 - ROMs come with special internal electronic fuses that can be "programmed" for a specific configuration. Once the pattern is established, it stays within the unit even when power is turned off and on again.
 - An $m \times n$ ROM is an array of binary cells organized into m words of n bits each.
 - A block diagram of ROM is as follows :



Block Diagram of Read-Only Memory (ROM)

- A ROM has k address input lines to select one of $2^k = m$ words of memory, and n output lines, one for each bit of the word.
- The ROM does not need a read-control line since at any given time, the output lines automatically provide the n bits of the word selected by the address value.

❖ **Types of ROMs :**

- There are basically two types of ROMs which are as follows:

1. **PROM (Programmed Read Only Memory)**
2. **EPROM (Erasable Programmable Read Only Memory)**

1. **PROM (Programmed Read Only Memory)**

- It is a field programmable device. As in ROM the contents cannot be modified after manufacturing.
- PROM gives the facility to the user to program ROM as per his requirements. But once programmed, the contents cannot be deleted.

2. **EPROM (Erasable Programmable Read Only Memory)**

- EPROM as the name suggests allows the contents of PROM to be deleted by using ultraviolet rays.
- Deleting one or more locations is not possible but the entire contents of EPROM are erased.
- Thus, contents of PROM which is used as ROM in a system can be erased and reprogrammed to suit the changed requirements.

❑ **Check Your Progress – 5 :**

1. RAM stands for _____.
(a) Read Access Memory (b) Random Access Memory
(c) Random Advance Memory (d) Read Advance Memory
2. ROM stands for _____.
(a) Read On Memory (b) Random Only Memory
(c) Read Only Memory (d) Random On Memory

9.7 Let Us Sum Up :

A multiplexer is a combinatorial circuit that is given a certain number (usually a power of two) data inputs, let us say $2n$, and n address inputs used as a binary number to select one of the data inputs. The multiplexer has a single output, which has the same value as the selected data input.

In other words, the multiplexer works like the input selector of a home music system. Only one input is selected at a time, and the selected input is transmitted to the single output. While on the music system, the selection of the input is made manually, the multiplexer chooses its input based on a binary number, the address input.

The demultiplexer is the inverse of the multiplexer, in that it takes a data input and n address inputs. It has $2n$ outputs. The address input determine which data output is going to have the same value as the data input. The other data outputs will have the value 0.

An encoder is the combinational circuit which performs a reverse function that of decoder. Encode inputs are decimal digits and/or alphabetic characters and outputs are coded representation of these inputs.

The basic function of a decoder is to detect the presence of a specified combination of bits on its inputs and to indicate that presence by a specified output level. A decoder has n input lines or handles n bits and from one to 2^n output lines to indicate the presence of one or more n – bit combinations.

9.8 Suggested Answer for Check Your Progress :

Check Your Progress 1 :

1 : a 2 : d 3 : b 4 : c

Check Your Progress 2 :

See Section 9.3

Check Your Progress 3 :

See Section 9.4

Check Your Progress 4 :

See Section 9.5

Check Your Progress 5 :

1 : b 2 : c

9.9 Glossary :

1. **Integrated Circuit** – An integrated circuit abbreviated 'IC' is a small silicon semiconductor crystal, called a chip. Chip containing the electronic components for the digital gates.
2. **Multiplexer** – A Multiplexer, abbreviated 'mux', is a device that has numerous inputs and one output.
3. **Encoders** – An encoder is a circuit that changes a set of signals into a code. Let's start making a 2-to-1 line encoder truth table by reversing a 1-to-2 decoder truth table.
4. **Decoders** – A decoder is a circuit that changes a code into a set of signals. It is called a decoder because it does a reverse of encoding, but we will start our study of encoders and decoders with decoders because they are easy to design.

9.10 Assignment :

1. Write a note on RAM.
2. What is ROM ? Explain types of ROM.

9.11 Activities :

Write down the advantages of 2-to-1 encoder in around 100 words.

9.12 Case Study :

Distinguish between 4-to-1 Multiplexer and 2-to-1 Multiplexer.

9.13 Further Readings :

1. Digital Electronics, Anand Kumar
2. Digital Logic and Computer Design, Morris Mano

UNIT STRUCTURE

- 10.0 Learning Objectives**
- 10.1 Introduction**
- 10.2 Address, Data & Control Bus**
- 10.3 Bus System for 4–Bit Register**
- 10.4 Three–State Bus Buffer**
- 10.5 Let Us Sum Up**
- 10.6 Suggested Answer for Check Your Progress**
- 10.7 Glossary**
- 10.8 Assignment**
- 10.9 Activities**
- 10.10 Case Study**
- 10.11 Further Readings**

10.0 Learning Objectives :

After learning this unit, you will be able to understand :

- What is Bus ?
- Address Bus
- Data Bus
- Control Bus
- Bus System for 4–Bit Register
- Three–State Bus Buffer

10.1 Introduction :

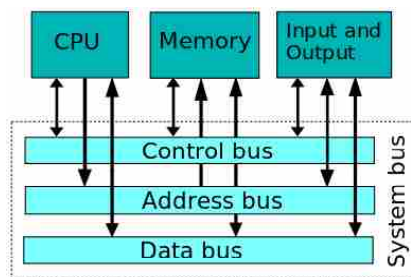
- A computer processes and stores data as a series of electronic bits. These bits transfer internally within the circuitry of the computer along electrical channels.
- Each channel, called a bus, allows the various devices both inside and attached to the system unit to communicate with one another.
- Buses are used to transfer bits from input devices to memory, from memory to the processor, from the processor to memory, and from memory to output or storage devices.
- The computer bus is communication link used in a computer system to send the data, addresses, control signals and the power to various components in a computer system.
- The computer buses are used to connect the various hardware components that are part of the computer system.
- In simple words, the computer buses are electrical wires which connect the various hardware components in a computer system. The computer

bus carries the data, control signals, memory addresses and the power supply to these components.

- The computer system makes use of different types of buses such as data bus, address bus and control bus.

10.2 Address, Data & Control Bus :

- A bus is a high-speed internal connection. Buses are used to send control signals and data between the processor and other components.
- A bus is a pathway for digital signals to rapidly move data. There are three internal buses associated with processors : the data bus, address bus, and control bus. Together, these three make up the "system bus."
- The system bus is an internal bus, intended to connect the processor with internal hardware devices, and is also called the "local" bus, Front Side Bus, or is sometimes loosely referred to as the "memory bus."
- Data moving in and out of the data bus is bi-directional, since the processor reads and writes data, however, the others are unidirectional, since the processor always determines when and what it will read from or write to.
- The address bus carries addressing signals from the processor to memory, I/O (or peripherals), and other addressable devices around the processor.
- Control signals move out of the processor, but not in to it.



Basic Diagram

- There are three types of bus which are as follows:

1. Address bus –

- Address bus carries memory addresses from the processor to other components such as primary storage and input/output devices. The address bus is unidirectional.
- It is a group of conducting wires which carries address only.
- Address bus is unidirectional because data flow in one direction, from microprocessor to memory or from microprocessor to Input/output devices (That is, Out of Microprocessor).
- Length of Address Bus of 8085 microprocessor is 16 Bit (That is, Four Hexadecimal Digits), ranging from 0000 H to FFFF H, (H denotes Hexadecimal).
- The microprocessor 8085 can transfer maximum 16-bit address which means it can address 65, 536 different memory location.
- The Length of the address bus determines the amount of memory a system can address.

- Such as a system with a 32-bit address bus can address 2^{32} memory locations.
- If each memory location holds one byte, the addressable memory space is 4 GB.
- However, the actual amount of memory that can be accessed is usually much less than this theoretical limit due to chipset and motherboard limitations.

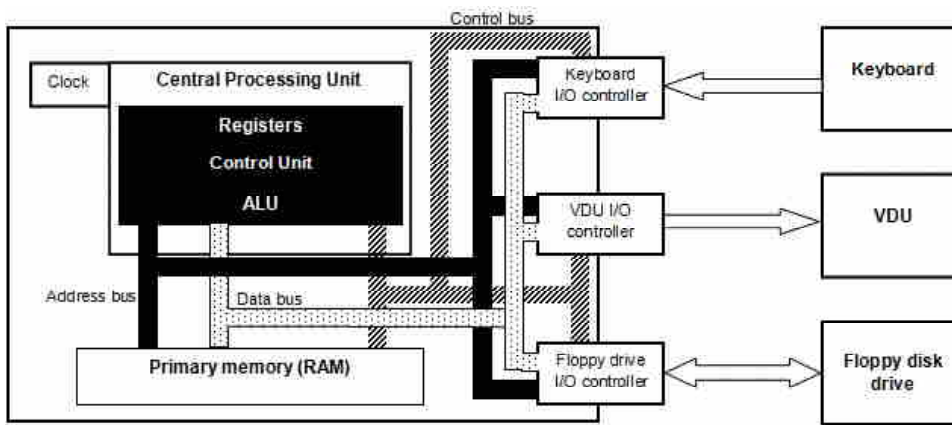
2. Data bus –

- Data bus carries the data between the processor and other components.
- It is a group of conducting wires which carries Data only. Data bus is bidirectional because data flow in both directions, from microprocessor to memory or Input/Output devices and from memory or Input/Output devices to microprocessor.
- Length of Data Bus of 8085 microprocessor is 8 Bit (That is, two Hexadecimal Digits), ranging from 00 H to FF H. (H denotes Hexadecimal).
- When it is write operation, the processor will put the data (to be written) on the data bus, when it is read operation, the memory controller will get the data from specific memory block and put it into the data bus.
- The width of the data bus is directly related to the largest number that the bus can carry, such as an 8-bit bus can represent 2 to the power of 8 unique values, this equates to the number 0 to 255. A 16-bit bus can carry 0 to 65535.

3. Control bus –

- Control bus carries control signals from the processor to other components. The control bus also carries the clock's pulses. The control bus is unidirectional.
- It is a group of conducting wires, which is used to generate timing and control signals to control all the associated peripherals, microprocessor uses control bus to process data that is what to do with selected memory location. Some control signals are :
 - ✓ Memory read
 - ✓ Memory write
 - ✓ I/O read
 - ✓ I/O Write
 - ✓ Opcode fetch
- If one line of control bus may be the read/write line. If the wire is low (no electricity flowing) then the memory is read, if the wire is high (electricity is flowing) then the memory is written.

Address, Data & Control Bus



A computer system showing the I/O controllers.

In general,

- The data bus "width" of a microcontroller is typically 8-, 16-, 32- or 64-bits, although microcontrollers of just a 4-bit data bus or greater than 64-bit width are possible.
- The width of the data bus reflects the maximum amount of data that can be processed and delivered at one time. A 64-bit processor has a 64-bit data bus and can communicate 64-bits of data at a time, and whether the data is read or written is determined by the control bus.
- The physical location of the data in memory is carried by the address bus. An internal hardware component, having received the address from the address bus and about to receive the data, enables a buffer to allow the flow of signals to or from the location that was designated by the address bus.
- The address bus carries only the information regarding the address, and is synchronized with the data bus to accomplish read/write tasks from the processor. The address bus is only as wide as is necessary to address all memory in the system.
- Other communication buses also communicate with the processor but are external to the system, such as Universal Serial Bus, RS-232, Controller Area Network (CAN), eSATA, and others. External peripherals may be set up to use the internal bus, and this was common with computers that used "expansion cards" to connect products to the internal bus. However, with one card per device this became untenable for the long term, and other bus communication systems such as USB were developed.
- A system bus can be "extended" to communicate with other computers via a chassis called a backplane. Internal buses have very rapid throughput and low latency. Several computers can be rack-mounted in a single backplane for very fast communication between computers.

☐ Check Your Progress – 1 :

1. What is BUS ? Explain Address, Data and Control Bus.

.....

.....

.....

.....

.....

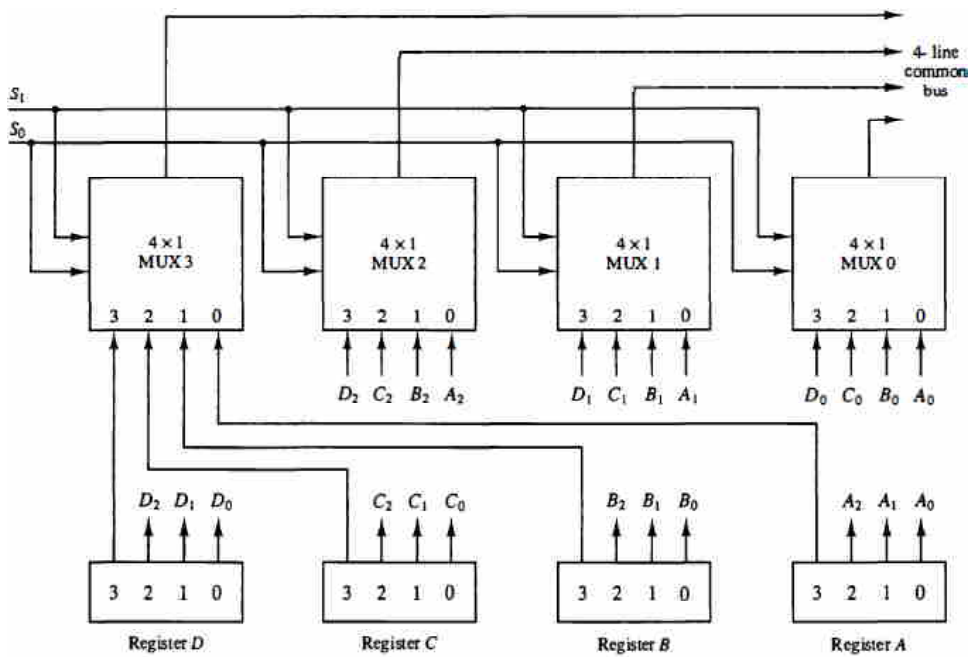
.....

□ **Check Your Progress – 2 :**

1. Which bus carry addresses _____ ?
(a) System bus (b) Address bus
(c) Control bus (d) Data bus
2. A 16-bit address bus can generate _____ addresses.
(a) 32767 (b) 25652 (c) 65536 (d) none of these
3. CPU can read & write data by using
(a) System bus (b) Address bus
(c) Control bus (d) Data bus
4. Which bus transfer signals from the CPU to external device and others that carry signals from external device to the CPU.
(a) System bus (b) Address bus
(c) Control bus (d) Data bus
5. When memory read or I/O read are active data is to the processor.
(a) Input (b) Output
(c) Processor (d) None of these
6. What is store by register ?
(a) memory (b) operands
(c) data (d) none of these
7. A set of register which contains are :
(a) data (b) memory addresses
(c) result (d) all of these

10.3 Bus System for 4-Bit Register :

- A typical digital computer has many registers, and paths must be provided to transfer information from one register to another.
- The number of wires will be unnecessary if separate lines are used between each register and all other registers in the system.
- For transferring information between registers in a multiple-register configuration is a common bus system.
- A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time.
- Control signals determine which register is selected by the bus during each particular register transfer.
- The multiplexers select the source register whose binary information is then placed on the bus.
- The construction of a bus system for four registers is shown in below figure :



Bus System for Four Register

- Each register has four bits, numbered 0 through 3.
- The bus consists of four 4 x 1 multiplexers each having four data inputs, 0 through 3, and two selection inputs, S1 and S0.
- The two selection lines S1 and S0 are connected to the selection inputs of all four multiplexers.
- The selection lines choose the four bits of one register and transfer them into the four-line common bus.
- When S1S0 = 00, the 0 data inputs of all four multiplexers are selected and applied to the outputs that form the bus.
- This causes the bus lines to receive the content of register A since the outputs of this register are connected to the 0 data inputs of the multiplexers.
- Similarly, register B is selected if S1S0 = 01, and so on.
- Function table for bus figure is as follows :

S1	S0	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

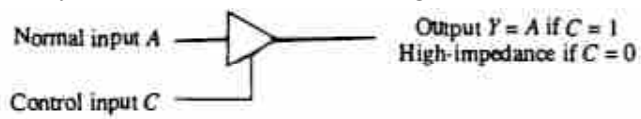
- The transfer of information from a bus into one of many destination registers can be accomplished by connecting the bus lines to the inputs of all destination registers and activating the load control of the particular destination register selected.
- When the bus is included in the statement, the register transfer is symbolized as follows :

$$\text{BUS} \leftarrow C, R1 \leftarrow \text{BUS}$$

- The content of register C is placed on the bus, and the content of the bus is loaded into register R1 by activating its load control input.

10.4 Three-State Bus Buffers :

- A bus system can be constructed with three-state gates instead of multiplexers.
- A three-state gate is a digital circuit that exhibits three states.
- Two of the states are signals equivalent to logic 1 and 0 as in a conventional gate. The third state is a high-impedance state.
- The high-impedance state behaves like an open circuit, which means that the output is disconnected and does not have a logic significance.
- One most commonly used in the design of a bus system is the buffer gate.
- The graphic symbol of a three-state buffer gate is shown in below figure :



- It is distinguished from a normal buffer by having both a normal input and a control input.
- The control input determines the output state.
- When the control input is equal to 1, the output is enabled and the gate behaves like any conventional buffer, with the output equal to the normal input.
- When the control input is 0, the output is disabled and the gate goes to a high-impedance state.
- The high-impedance state of a three-state gate provides a special feature not available in other gates. Because of this feature, a large number of three-state gate outputs can be connected with wires to form a common bus line without endangering loading effects.

□ Check Your Progress – 3 :

1. Explain 4-bit Bus Register.

.....

.....

.....

.....

.....

10.5 Let Us Sum Up :

A bus is simply the name given for links, channels or wires, along which you can send one of three types of information. These are:

- Data
- Addresses
- Control information.

It looks quite complicated at first. If you look closely, however, you can see the buses like,

The address bus – This bus, in black on the diagram, is usually a set of wires that links the CPU to the RAM (and to other places). If the CPU want to fetch an instruction from a particular address in RAM, or wants to write a piece of data to a particular address in RAM, it puts the address on the address bus.

The data bus – The dotted channels you can see in the diagram is the data bus. The CPU puts data it wants to transfer to RAM on this bus. If data is being fetched from RAM, then it is put on this bus as well.

The control bus – Signals need to be sent around the computer to control when things happen. These signals are sent along the control bus. The striped channel you can see in the diagram is the control bus.

As bus play an important role in transferring data and control information from one part of the system to another part. Bus is connected with many registers, and memory. A typical digital computer has many registers, and paths must be provided to transfer information from one register to another. The number of wires will be unnecessary if separate lines are used between each register and all other registers in the system. For transferring information between registers in a multiple-register configuration is a common bus system.

In simple words, the computer buses are electrical wires which connect the various hardware components in a computer system. The computer bus carries the data, control signals, memory addresses and the power supply to these components.

10.6 Suggested Answer for Check Your Progress :

Check Your Progress 1 :

See Section 10.1 & 10.2

Check Your Progress 2 :

1 : b 2 : c 3 : d 4 : c
5 : a 6 : c 7 : d

Check Your Progress 3 :

See Section 10.4

10.7 Glossary :

1. **Address bus** – Address bus carries memory addresses from the processor to other components such as primary storage and input/output devices.
2. **Data bus** – Data bus carries the data between the processor and other components.
3. **Control bus** – Control bus carries control signals from the processor to other components. The control bus also carries the clock's pulses.

10.8 Assignment :

1. Define data bus with its types.
2. What is role of data bus ?

10.9 Activities :

Note down how communication between bus actually done ?

10.10 Case Study :

1. Take one processor example and draw the diagram of how bus handle in it.

10.11 Further Readings :

1. Digital Electronics, Anand Kumar
2. Digital Logic and Computer Design, Morris Mano

BLOCK SUMMARY :

While studying this block, the user will achieve knowledge and understanding about CPU and the parts of the CPU. The Block explain about the ALU, Combinational Circuit, Half-Adder, Full-Adder, Parallel Binary Adder, and Binary Adder-Subtractor.

The block has given detail idea on the integrated circuit and types of integrated circuit, Decoder, Encoder, Multiplexer, and Memory Unit.

The block has given the knowledge about Bus as well as types of Bus like Data Bus, Address Bus and Control Bus. As we know that bus play an important role in moving data, detailed idea about implementation of 4-bit register, and three-state bus buffer.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. What is ALU ?
2. Explain Parallel Binary Adder.
3. Explain binary Adder-Subtractor.
4. What is Integrated Circuit ? Explain types of it.
5. What is Bus? Explain types of Bus.
6. Write a detailed note on Multiplexer.

❖ **Long Questions :**

1. What is Adder ? Explain Half-Adder and Full-Adder.
2. What is Decoder ? Explain 3-to-8-line decoder.
3. Explain Memory Unit in detail.
4. Explain Bus system for 4-bit register.
5. What is complement? Explain 1's and 2's complement with example.

❖ **Enrolment No. :**

1. How many hours did you need for studying the units ?

Unit No.	8	9	10
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Language and Style	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Illustration used (Diagram, tables etc)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Conceptual Clarity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Check your progress Quest	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Feed back to CYP Question	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....



Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-203/
DCAR-203

Digital Electronics and Computer Organization

BLOCK 4 : INPUT/OUTPUT DEVICES AND FLIP FLOPS

UNIT 11 INPUT/OUTPUT DEVICES

UNIT 12 INPUT/OUTPUT INTERFACE AND DATA TRANSFER

UNIT 13 MEMORY

UNIT 14 FLIP-FLOPS

UNIT 15 CPU

INPUT/OUTPUT DEVICES AND FLIP FLOPS

Block Introduction :

In this block, we will learn and study about basic of the Input and Output devices in which we will learn about Key Board, Mouse, Display Unit, Printer and types of the printer, Scanner, OCR (Optical Character Recognition), OMR (Optical Mark Recognition), MICR (Magnetic Ink Character Recognition). In computing, input/output (I/O, or informally io or IO) is the communication between an information processing system, such as a computer, and the outside world, possibly a human or another information processing system.

In this block, we will learn and study about the Input/output interface, Asynchronous data transfer, and modes of data transfer. Concepts of programmed I/O, DMA (Direct Memory Access). Input Output Interface provides a method for transferring information between internal storage and external I/O devices.

In this block, we will learn and study about the Memory and as a part of Memory we will learn about Memory Hierarchy, detailed study about Primary Memory – RAM, SRAM, DRAM, ROM, Types of ROM, Secondary Memory (Magnetic Disk, Magnetic Tape), Optical Memory (CDROM), Concept of Virtual Memory, Concept of Cache and their need.

In this block, we will learn and study about the Flip–Flop. Flip–Flop is two state device which offer basic memory cell for sequential logic operations. Flip–Flop is a term referring to an electronic circuit and are used for digital data storage and transfer. Students will also learn and study about Transfer circuit, Clocks, 3–4–bit registers, Shift register, Synchronous/ Asynchronous binary counters.

In this block, we will learn and study about the CPU. As a part of CPU, we will learn and study about Functions of CPU, register classification and organization, instruction cycle, instruction formats, addressing modes.

Block Objectives :

After learning this block, you will be able to understand :

- Idea about Input and Output Devices
- Input Output Interface and Data Transfer
- Modes of Data Transfer
- Detail about DMA (Direct Memory Access)
- Memory Hierarchy
- Primary Memory, Secondary Memory, and Optical Memory
- Concept of Cache Memory and Virtual Memory
- Detail about Flip-Flop and Types of Flip-Flop with truth-table
- Detail about Shift Registers and Counters
- Detail about CPU and Functions of CPU
- Classification and organization of register
- Instruction cycle
- Instruction formats
- Addressing modes.

Block Structure :

Unit 11 : Input/Output devices

Unit 12 : Input/Output Interface and Data Transfer

Unit 13 : Memory

Unit 14 : Flip-Flops

Unit 15 : CPU

ADDRESS, DATA & CONTROL BUS

UNIT STRUCTURE

- 11.0 Learning Objectives
- 11.1 Introduction
- 11.2 Input/Output Devices
 - 11.2.1 Key Board
 - 11.2.2 Mouse
 - 11.2.3 Display Unit
 - 11.2.4 Printer (Types)
 - 11.2.5 Scanner
 - 11.2.6 OCR
 - 11.2.7 OMR
 - 11.2.8 MICR
- 11.3 Let Us Sum Up
- 11.4 Suggested Answer for Check Your Progress
- 11.5 Glossary
- 11.6 Assignment
- 11.7 Activities
- 11.8 Case Study
- 11.9 Further Readings

11.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Input/Output Devices
- Input Device such as Key Board, Mouse, Scanner, OCR, OMR, MICR
- Output Device such as Display Unit, and Printer
- Types of Printers

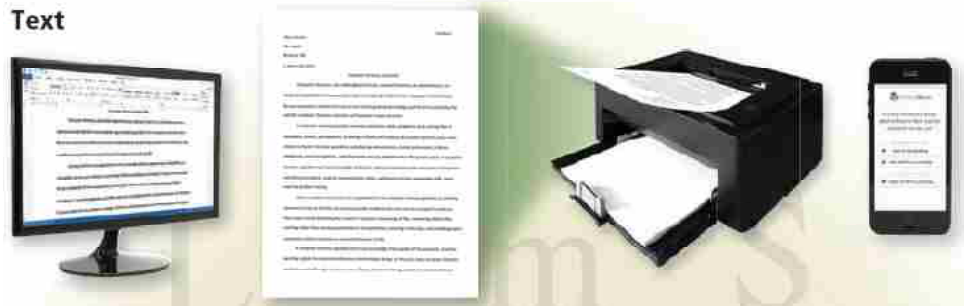
11.1 Introduction :

- In computing, input/output (I/O, or informally io or IO) is the communication between an information processing system, such as a computer, and the outside world, possibly a human or another information processing system.
- **Input** is any data and instructions entered into the memory of a computer.
- Data is a collection of unprocessed items, including text, numbers, images, audio, and video. Once data is in memory, a computer or mobile device interprets and executes instructions to process the data into information.

Digital Electronics and Computer Organization

- Instructions that a computer or mobile device processes can be in the form of software (programs and apps), commands, and user responses.
- **Software :**
 - Software is a series of related instructions, organized for a common purpose, that tells a computer or mobile device what tasks to perform and how to perform them.
- **Command :**
 - A command is an instruction that causes a program or app to perform a specific action.
 - Programs and apps respond to commands that a user issues.
 - Users issue commands by touching an area on a screen, pressing keys on the keyboard, clicking a mouse button to control a pointer on the screen, or speaking into a microphone.
- **User Response :**
 - A user response is an instruction a user issues by responding to a message displayed by a program or app.
 - A response to the message instructs the program or app to perform certain actions.
 - **Output** is data that has been processed into a useful form. Recall that computers process data (input) into information (output).
 - The form of output varies, depending on the hardware and software being used and the requirements of the user. Users view or watch output on a screen, print it, or hear it through speakers, headphones, or earbuds.
 - A user encounters four basic types of output: text, graphics, audio, and video.
- **Text :**
 - Examples of output that primarily contain text are text messages, Internet messages, memos, letters, press releases, reports, classified advertisements, envelopes, and mailing labels.
 - On the web, users read blogs, news and magazine articles, books, television show transcripts, stock quotes, speeches, and lectures.

Text



- **Graphics :**
 - Many forms of output include graphics to enhance visual appeal and convey information. Business letters have logos. Reports include charts. Newsletters use drawings, clip art, and photos.



• **Audio :**

- Users download their favorite songs and listen to the music. On the web, users listen to radio broadcasts, audio clips, podcasts, sporting events, news, music, and concerts.



• **Video :**

- As with audio, software and websites often include video clips and video blogs.
- Users watch news reports, movies, sporting events, weather conditions, and live performances on a computer or mobile device.



11.2 Input/Output Devices :

- An input device sends information to a computer system for processing, and an output device reproduces or displays the results of that processing.
- Input devices only allow for input of data to a computer and output devices only receive the output of data from another device.
- Input and output devices allow the computer system to interact with the outside world by moving data into and out of the system.
- **Input devices are :** Keyboard, Mouse, Scanner, OCR, OMR, MICR, Microphone, and Webcam.
- **Output devices are :** Display Unit, Printers, Projector, and Speaker.

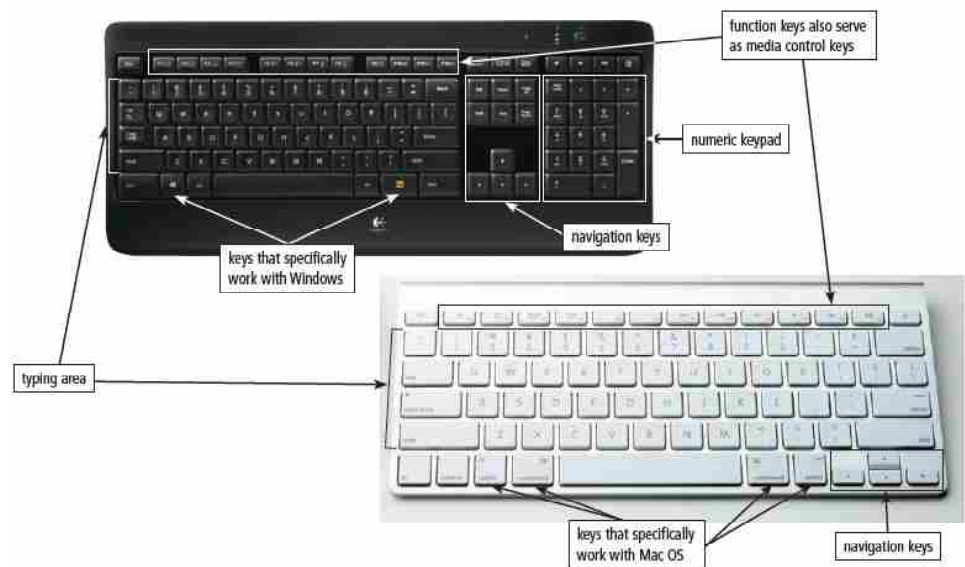
11.2.1 Key Board :

- A *keyboard* is an input device that contains keys you press to enter data and instructions into a computer or mobile device.
- Nearly all keyboards have a typing area, function keys, toggle keys, and navigation keys.

- Many also include media control buttons, Internet control buttons, and other special keys.
- The **typing area** includes letters of the alphabet, numbers, punctuation marks, and other basic keys.
- **Function keys**, which are labelled with the letter F followed by a number, are special keys programmed to issue commands to a computer. The command associated with a function key may vary, depending on the program you are using. F1–Help, F2–Rename, F3–Search, F4–Find, F5–Refresh, F7–Spell Check
- A **toggle key** is a key that switches between two states each time a user presses the key. Caps lock and num lock are examples of toggle keys.
- Users can press the **navigation keys**, such as arrow keys and page up/pg up and page down/pg dn on the keyboard, to move the insertion point in an application left, right, up, or down.
- A **keyboard shortcut** is one or more keyboard keys that you press to perform an operating system or application–related task.
- **Media control** buttons allow you to control a media player program, access the computer's optical disc drive, Pause, Play, Forward, Backward, Mute and adjust speaker volume.
- **Internet control** buttons allow you to run an email application, run a browser, and search the web. Wi–Fi On/Off.

• **Types of Keyboard :**

- Desktops include a standard keyboard.
- **Standard keyboards** typically have from 101 to 105 keys, which include function keys along the top and a numeric keypad on the right.
- Devices often use a compact keyboard, which is smaller than a standard keyboard and usually does not include the numeric keypad or navigation keys.



- Other compact keyboards are separate devices that communicate **wirelessly** or attach to the computer or device with a magnet, clip, or other mechanism.



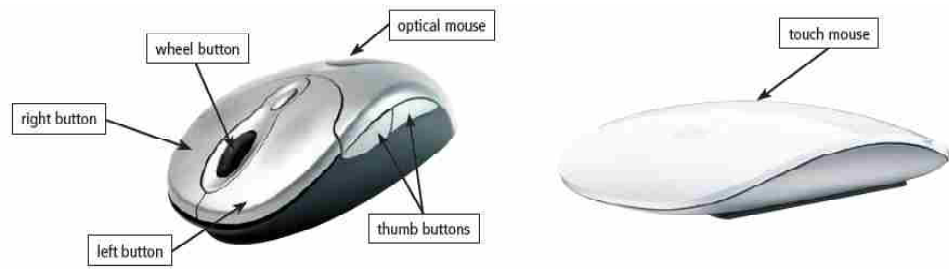
- Some users prefer to work with *on-screen or virtual keyboards* instead of a physical keyboard.
- An *ergonomic keyboard* has a design that reduces the chance of repetitive strain injuries (RSIs) of wrist and hand.
- Recall that the goal of ergonomics is to incorporate comfort, efficiency, and safety in the design of the workplace.



- A *gaming keyboard* is a keyboard designed specifically for users who enjoy playing games on the computer.
- Gaming keyboards typically include programmable keys so that gamers can customize the keyboard to the game being played.

11.2.2 Mouse :

- A *mouse* is a pointing device that fits under the palm of your hand comfortably.
- As you move a mouse, the pointer on the screen also moves.
- The bottom of a mouse is flat and contains a mechanism that detects movement of the mouse.
- An *optical mouse* uses optical sensors that emit and sense light to detect the mouse's movement. A *laser mouse* uses laser sensors that emit and sense light to detect the mouse's movement.
- The top and sides of an optical or laser mouse may have one to four buttons; some may also have a small wheel.



- A **touch mouse** is a touch-sensitive mouse that recognizes touch gestures, in addition to detecting movement of the mouse and traditional click and scroll operations.

11.2.3 Display Unit :

- A display device, or simply **display**, is an output device that visually conveys text, graphics, and video information.
- Sometimes called soft copy, information on a display exists electronically and appears for a temporary period.
- Displays consist of a screen and the components that produce the information on the screen.
- A **monitor** is a display that is packaged as a separate peripheral device.
- Monitor controls enable you to adjust the brightness, contrast, positioning, height, and width of images. Some have touch screens, integrated speakers, and/or a built-in webcam.
- Size of these displays varies depending on the mobile computer or device. Some mobile computers and many mobile devices have touch screens. Traditional laptops have a display that attaches with a hinge to the case.
- Tablets are available with two types of displays: one that attaches with a hinge and one built into the top of the case.
- Newer vehicles integrate a display in the dashboard, enabling drivers to control audio, video, navigation, temperature, and other settings.

11.2.4 Printer (Types) :

- A **printer** is an output device that produces text and graphics on a physical medium, such as paper.
- Printed information (hard copy) exists physically and is a more permanent form of output than that presented on a display (soft copy).
- A hard copy, also called a *printout*, is either in *portrait* or *landscape* orientation.
- A printout in *portrait* orientation is taller than it is wide, with information printed across the shorter width of the paper. Letters, reports, and books typically use portrait orientation.
- A printout in *landscape* orientation is wider than it is tall, with information printed across the widest part of the paper. Spreadsheets, slide shows, and graphics often use landscape orientation.
- **Nonimpact Printer :**
 - A **nonimpact printer** forms characters and graphics on a piece of paper without actually contacting the paper. Some spray ink, while others use heat or pressure to create images.

- Commonly used nonimpact printers are ink-jet printers, photo printers, laser printers, all-in-one printers, thermal printers, mobile printers, label printers, plotters, and large-format printers.

1. Ink-Jet Printers :

- An ink-jet printer spraying tiny drops of liquid ink onto a piece of paper.
- Ink-jet printers produce text and graphics in both black-and-white and color on a variety of paper types and sizes. Most ink-jet printers can print lab-quality photos.
- Ink-jet printers also print on other materials, such as envelopes, labels, index cards, greeting card paper (card stock), transparencies, and iron-on T-shirt transfers.
- The speed of an ink-jet printer is measured by the number of pages per minute (ppm) it can print. Graphics and colors print at a slower rate than text.



• Ink Cartridges :

- An ink-jet printer contains ink-filled cartridges. Each cartridge has fifty to several hundred small ink holes, or nozzles.
- The ink forces through any combination of the nozzles to form a character or image on the paper.
- When the cartridge runs out of ink, you simply replace the cartridge. Most ink-jet printers use two or more ink cartridges, one containing black ink and the other(s) containing colors.

2. Photo Printers :

- A photo printer is a color printer that produces lab-quality photos.



- Some photo printers print just one or two sizes of photos, for example, 3 * 5 inches and 4 * 6 inches. Others print up to 8 * 10 or even larger.
- Some even print panoramic photos. Generally, the more sizes the printer prints, the more expensive the printer.

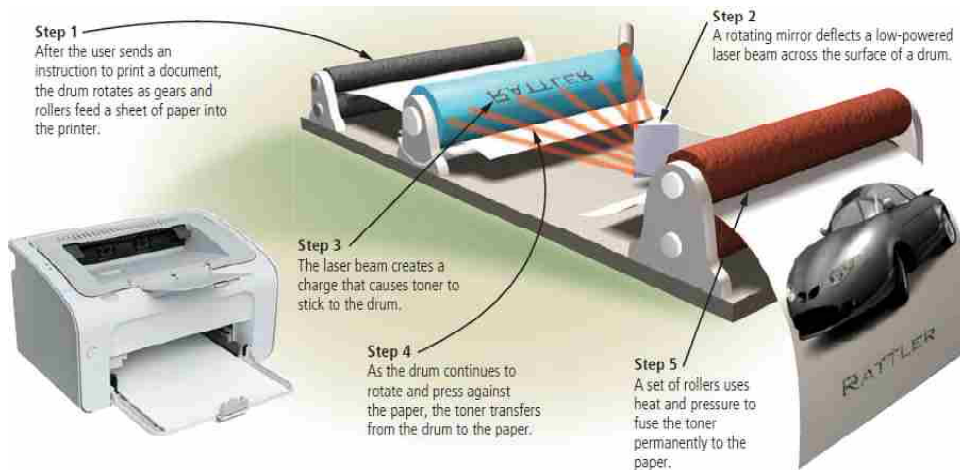
- Many photo printers use ink-jet technology. With models that can print letter-sized documents.
- Most photo printers are PictBridge enabled, so that you can print photos without a computer.
- PictBridge is a standard technology that allows you to print photos directly from a digital camera by connecting a cable from the digital camera to a USB port on the printer.
- Photo printers also usually have a built-in card slot(s) so that the printer can print digital photos directly from a memory card.

3. Laser Printers :

- A laser printer is a high-speed, high-quality nonimpact printer. Laser printers are available in both black-and-white and color models.
- A laser printer for personal computers ordinarily uses individual 8 1/2 * 11-inch sheets of paper.
- Laser printers print text and graphics in high-quality resolutions.
- While laser printers usually cost more than ink-jet printers. Laser printers usually print at faster speeds than ink-jet printers.
- Depending on the quality, speed, and type of laser printer, the cost ranges from a few hundred to a few thousand dollars for the home and small office user, and several hundred thousand dollars for the enterprise user.



- When printing a document, laser printers process and store the entire page before they actually print it. For this reason, laser printers sometimes are called page printers.
 - ✓ Operating in a similar to a copy machine, a laser printer creates images using a laser beam and powdered ink, called toner.
 - ✓ The laser beam produces an image on a drum inside the printer.
 - ✓ The light of the laser alters the electrical charge on the drum wherever it hits.
 - ✓ When this occurs, the toner sticks to the drum and then transfers to the paper through a combination of pressure and heat.
 - ✓ When the toner runs out, you replace the toner cartridge.



4. All-in-One Printers :

- An all-in-one printer, also called a *multifunction printer* (MFP), is a single device that looks like a printer or a copy machine but provides the functionality of a printer, scanner, copy machine, and perhaps a fax machine.
- Some use color ink-jet printer technology, while others use laser technology.



5. 3-D Printers :

- A 3-D printer uses a process called additive manufacturing to create an object by adding material to a three-dimensional object, one horizontal layer at a time.
- 3-D printers can print solid objects, such as clothing, prosthetics, eyewear, implants, toys, parts, prototypes, and more.



- Using a digital model created with CAD (Computer-Aided Design) software, 3-D printers begin creating an object at the bottom and add layers of material to the object until it is complete.
- Depending on the type of printer, the layers are built with liquid polymer, gel, or resin.

6. Thermal Printers :

- A thermal printer generates images by pushing electrically heated pins against heat-sensitive paper.
- Basic thermal printers are inexpensive, but the print quality is low, the images tend to fade over time, and thermal paper can be expensive.
- Self-service gas pumps often print gas receipts using a built-in, lower-quality thermal printer.
- Many point-of-sale terminals in retail and grocery stores also print purchase receipts on thermal paper.
- Some thermal printers have high print quality and can print at much faster rates than ink-jet and laser printers.
- A dye-sublimation printer, sometimes called a digital photo printer, uses heat to transfer colored dye to specially coated paper.



- Photography studios, medical labs, security identification systems, and other professional applications requiring high image quality use dye-sublimation printers that can cost thousands of dollars.

7. Mobile Printers :

- A mobile printer is a small, lightweight, battery-powered printer that allows a mobile user to print from a laptop, smartphone, or other mobile device while traveling.



- Mobile printers fit easily in a briefcase alongside a laptop.
- Mobile printers mainly use ink-jet or thermal technology.
- Many connect to a USB port. Others have a built-in wireless port through which they communicate with the computer.

8. Label Printers :

- A label printer is a small printer that prints on an adhesive-type material that can be placed on a variety of items, such as envelopes, packages, optical discs, photos, and file folders.
- Most label printers also print bar codes. Label printers typically use thermal technology.



9. Plotters and Large-Format Printers :

- Plotters are sophisticated printers used to produce high-quality drawings, such as blueprints, maps, and circuit diagrams.
- These printers are used in specialized fields such as engineering and drafting and usually are very costly.
- Current plotters use a row of charged wires (called styli) to draw an electrostatic pattern on specially coated paper and then fuse toner to the pattern.
- The printed image consists of a series of very small dots, which provides high-quality output.
- Using ink-jet printer technology, but on a much larger scale, a large-format printer creates photo-realistic-quality color prints.
- Graphic artists use these high-cost, high-performance printers for signs, posters, and other professional quality displays.

• Impact Printer :

- An impact printer forms characters and graphics on a piece of paper by striking a mechanism against an inked ribbon that physically contacts the paper.
- Impact printers characteristically are noisy because of this striking activity.



- Impact printers are ideal for printing multipart forms because they print through many layers of paper easily.
- Factories, warehouses, and retail counters may use impact printers because these printers withstand dusty environments, vibrations, and extreme temperatures.

11.2.5 Scanner :

- An optical scanner, usually called a scanner, is a light-sensing input device that reads printed text and graphics and then translates the results into a form the computer can process.
- A flatbed scanner works in a manner similar to a copy machine except it creates a file of the document in memory instead of a paper copy.

- The quality of a scanner is measured by its resolution, that is, the number of bits it stores in a pixel and the number of pixels per inch.
- Many scanners include OCR (optical character recognition) software, which can read and convert text documents into electronic files.
- OCR software converts a scanned image into a text file that can be edited, for example, with a word processing application.



11.2.6 OCR :

- OCR stands for Optical Character Recognition.
- OCR devices include a small optical scanner for reading characters and sophisticated software to analyze what is read.
- OCR is a technology that recognizes text within a digital image. It is commonly used to recognize text in scanned documents, but it serves many other purposes as well.
- OCR software processes a digital image by locating and recognizing characters, such as letters, numbers, and symbols.
- Some OCR software will simply export the text, while other programs can convert the characters to editable text directly in the image.
- Advanced OCR software can export the size and formatting of the text as well as the layout of the text found on a page.
- OCR technology can be used to convert a hard copy of a document into an electronic version or soft copy. For example, if you scan a multipage document into a digital image, such as a TIFF file, you can load the document into an OCR program, which will recognize the text and convert the document to an editable text file.
- While OCR technology was originally designed to recognize printed text, it can be used to recognize and verify handwritten text as well.
- For example, postal services such as USPS use OCR software to automatically process letters and packages based on the address.

11.2.7 OMR :

- OMR stands for Optical Mark Recognition.
- OMR devices read hand-drawn marks, such as small circles or rectangles. A person places these marks on a form, such as a test, survey, or questionnaire answer sheet.

11.2.8 MICR :

- MICR stands for Magnetic Ink Character Recognition.
- MICR devices read text printed with magnetized ink. An MICR reader converts MICR characters into a form the computer can process.
- The banking industry almost exclusively uses MICR for check processing. Each check in your checkbook has precoded MICR characters beginning at the lower-left edge.

- When a bank receives a check for payment, it uses an MICR inscriber to print the amount of the check in MICR characters in the lower-right corner.
- Each check is inserted in an MICR reader, which sends the check information – including the amount of the check – to a computer for processing.



☐ Check Your Progress – 1 :

1. Write a note on Mouse.

.....
.....
.....
.....
.....

2. Write a note on Display.

.....
.....
.....
.....
.....

3. Write a note on OCR, OMR, and MICR.

.....
.....
.....
.....
.....

☐ Check Your Progress – 2 :

1. OCR stands for _____.
- (a) Optical Character Recognition
 - (b) Original Character Recognition
 - (c) Organized Character Recognition
 - (d) Opposite Character Recognition
2. OMR stands for _____.
- (a) Original Mark Recognition
 - (b) Optical Memory Recognition
 - (c) Optical Mark Recognition
 - (d) Original Memory Recognition

3. MICR stands for _____.
 - (a) Memory Ink Character Recognition
 - (b) Magnetic Ink Character Recognition
 - (c) Master Ink Character Recognition
 - (d) Multiple Ink Character Recognition
4. _____ function key is used for getting help.
 - (a) F3
 - (b) F1
 - (c) F4
 - (d) F5
5. _____ are types of output.
 - (a) Text
 - (b) Graphics
 - (c) Audio and Video
 - (d) All of Above

11.3 Let Us Sum Up :

In this unit we learned about an input/output device can receive data from users, or another device (input), and send data to another device (output). We have also seen different types of output like Text, Graphics, Audio and Video. As a part of input/output device we have seen input devices like keyboard, mouse, display unit, printer with its types and scanner with its types.

We see that Keyboard, Mouse, Scanner, OCR, OMR, MICR are used as input device while Display Unit, and Printer are used as output device. We discussed about the all types of keys of keyboard as well as types of keyboards, printer and its types such as impact printer and non-impact printer.

Finally, we do understand about Input/Output device and working of all the devices of the computer system.

11.4 Suggested Answer for Check Your Progress :

☐ Check Your Progress 1 :

- 1 : See Section 11.2.2 2 : See Section 11.2.3
3 : See Section 11.2.6, 11.2.7, 11.2.8

☐ Check Your Progress 2 :

- 1 : a 2 : c 3 : b
4 : b 5 : d

11.5 Glossary :

1. **Input** : Input is any data and instructions entered into the memory of a computer.
2. **Software** : Software is a series of related instructions, organized for a common purpose, that tells a computer or mobile device what tasks to perform and how to perform them.
3. **Command** : A command is an instruction that causes a program or app to perform a specific action.
4. **Output** : Output is data that has been processed into a useful form.
5. **Keyboard** : A keyboard is an input device that contains keys you press to enter data and instructions into a computer or mobile device.

6. **Display** : A display device, or simply display, is an output device that visually conveys text, graphics, and video information.
7. **Printer** : A printer is an output device that produces text and graphics on a physical medium, such as paper.

11.6 Assignment :

1. Explain Keyboard in detail.
2. What is Printer ? Explain types of Printer

11.7 Activities :

1. Find out different input and output devices which are used in banking security sector.

11.8 Case Study :

1. Take a college management system and list out how many numbers of devices we can use at different levels ? Like admin, student, staff, etc...

11.9 Further Readings :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar

UNIT STRUCTURE

- 12.0 Learning Objectives**
- 12.1 Introduction**
- 12.2 Input/Output Interface**
- 12.3 Asynchronous Data Transfer and Mode of Data Transfer**
- 12.4 Concept of Programmed I/O**
- 12.5 DMA**
- 12.6 Let Us Sum Up**
- 12.7 Suggested Answer for Check Your Progress**
- 12.8 Glossary**
- 12.9 Assignment**
- 12.10 Activities**
- 12.11 Case Study**
- 12.12 Further Readings**

12.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Detail about Input/Output Interface
- Asynchronous Data Transfer
- Modes of Data Transfer
- Concept of Programmed I/O
- Detail about DMA (Direct Memory Access)

12.1 Introduction :

In this unit we are going to learn and study about the Input/Output interface as it is a method for transferring information between internal parts of the computer system. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit that communication link is provided by input/output Interface.

Asynchronous input output is a form of input output processing that allows others devices to do processing before the transmission or data transfer is done.

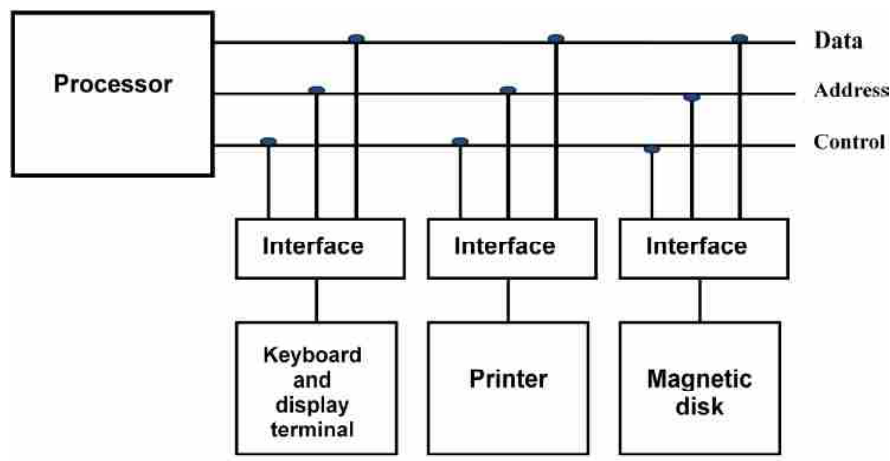
Direct memory access (DMA) is a means of having a peripheral device control a processor's memory bus directly. DMA permits the peripheral, such as a Universal Asynchronous Receiver–Transmitter (UART) to transfer data directly to or from memory without having each byte (or word) handled by the processor. Thus, DMA enables more efficient use of interrupts, increases data throughput, and potentially reduces hardware costs by eliminating the need for peripheral–specific FIFO buffers.

12.2 Input/Output Interface :

Input/Output Interface and Data Transfer

- Input/Output Interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit.
- The purpose of communication link is to resolve the differences that exist between the central computer and each peripheral.
- The Major Differences are :
 1. Peripherals are electromechanical and electromagnetic devices and CPU and memory are electronic devices. Therefore, a conversion of signal values may be needed.
 2. The data transfer rate of peripherals is usually slower than the transfer rate of CPU and consequently, a synchronization mechanism may be needed.
 3. Data codes and formats in the peripherals differ from the word format in the CPU and memory.
 4. The operating modes of peripherals are different from each other and must be controlled so as not to disturb the operation of other peripherals connected to the CPU.
- To resolve these differences, computer systems include special hardware components between the CPU and Peripherals to supervise and synchronize all input and out transfers.
- These components are called Interface Units because they interface between the processor bus and the peripheral devices.
- **I/O BUS and Interface Modules :**
 - It defines the typical link between the processor and several peripherals. The I/O Bus consists of data lines, address lines and control lines.
 - The I/O bus from the processor is attached to all peripherals interface.
 - To communicate with a particular device, the processor places a device address on address lines.
 - Each Interface decodes the address and control received from the I/O bus, interprets them for peripherals and provides signals for the peripheral controller.
 - It is also synchronizing the data flow and supervises the transfer between peripheral and processor.
 - Each peripheral has its own controller.
 - For example, the printer controller controls the paper motion, the print timing. The control lines are referred as I/O command.
 - The commands are as following:
 - o **Control command** – A control command is issued to activate the peripheral and to inform it what to do.
 - o **Status command** – A status command is used to test various status conditions in the interface and the peripheral.

- o **Data Output command** – A data output command causes the interface to respond by transferring data from the bus into one of its registers.
 - o **Data Input command** – The data input command is the opposite of the data output.
- In this case the interface receives an item of data from the peripheral and places it in its buffer register.



Connection of I/O bus to input-output devices

- **I/O Versus Memory Bus :**
 - To communicate with I/O, the processor must communicate with the memory unit. Like the I/O bus, the memory bus contains data, address and read/write control lines. There are 3 ways that computer buses can be used to communicate with memory and I/O:
 - ✓ Use two Separate buses, one for memory and other for I/O.
 - ✓ Use one common bus for both memory and I/O but separate control lines for each.
 - ✓ Use one common bus for memory and I/O with common control lines.
- **I/O Processor :**
 - In the first method, the computer has independent sets of data, address and control buses one for accessing memory and other for I/O. This is done in computers that provide a separate I/O processor (IOP).
 - The purpose of IOP is to provide an independent pathway for the transfer of information between external device and internal memory.

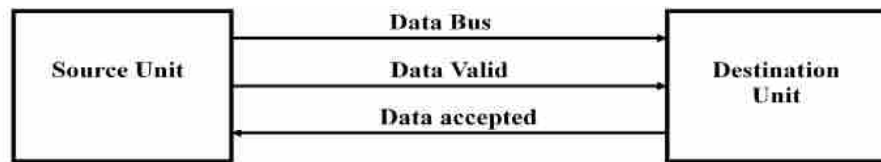
☐ **Check Your Progress – 1 :**

1. Explain Input/Output Interface.
-
-
-
-
-

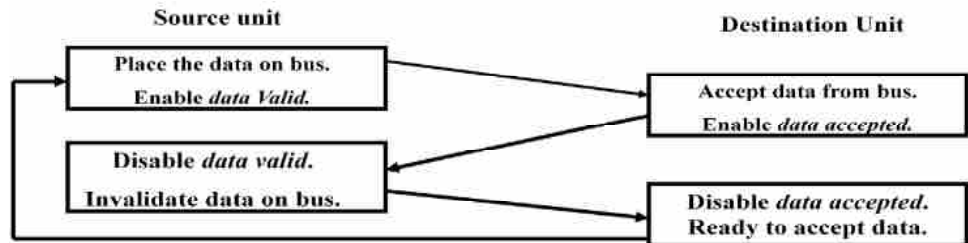
2. DMA Stands for _____.
- (a) Direct Memory Access (b) Direct Memory Avoid
(c) Direct Memory Arrange (d) Direct Memory Adjust
3. IOP Stands for _____.
- (a) Input Output Print (b) Input Output Processor
(c) Input Output Predication (d) Input Output Presence

12.3 Asynchronous Data Transfer and Mode of Data Transfer :

- This scheme is used when speeds of I/O devices do not match with microprocessor, and timing characteristics of I/O devices is not predictable.
- In this method, process initiates the device and checks its status. As a result, CPU has to wait till I/O device is ready to transfer data.
- When device is ready CPU issues instruction for I/O transfer.
- **Handshaking :**
 - Method commonly used to accompany each data item being transferred with a control signal that indicates the presence of data in the bus.
 - The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as handshaking.
- **Principle of Handshaking :**
 - The basic principle of the two – wire handshaking method of data transfer is as follow :
 - ✓ One control line is in the same direction as the data flows in the bus from the source to destination.
 - ✓ It is used by source unit to inform the destination unit whether there a valid data in the bus.
 - ✓ The other control line is in the other direction from the destination to the source.
 - ✓ It is used by the destination unit to inform the source whether it can accept the data.
 - ✓ The sequence of control during the transfer depends on the unit that initiates the transfer.
- **Source Initiated Transfer using Handshaking :**
 - The sequence of events shows four possible states that the system can be at any given time.
 - The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal.
 - The data accepted signal is activated by the destination unit after it accepts the data from the bus.
 - The source unit then disables its data accepted signal and the system goes into its initial state.

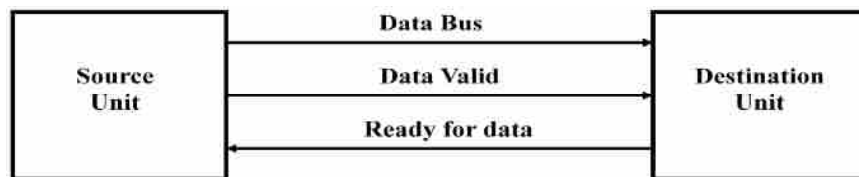


(a) Block Diagram

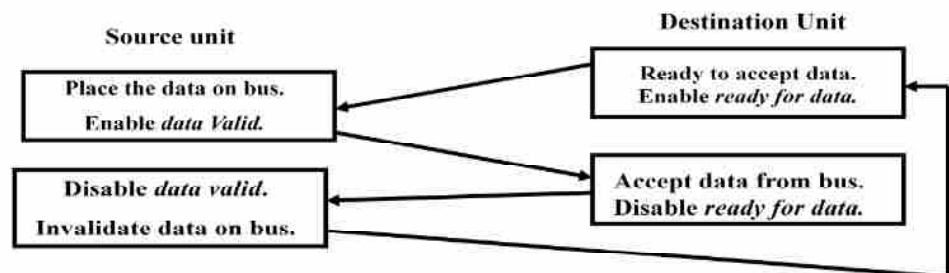


(b) Sequence of events

- **Destination Initiated Transfer Using Handshaking :**
 - The name of the signal generated by the destination unit has been changed to ready for data to reflect its new meaning.
 - The source unit in this case does not place data on the bus until after it receives the ready for data signal from the destination unit.
 - From there on, the handshaking procedure follows the same pattern as in the source-initiated case.
 - The only difference between the Source Initiated and the Destination Initiated transfer is in their choice of Initial state.



(a) Block Diagram



(b) Sequence of events

Destination-Initiated transfer using Handshaking

- **Advantage of the Handshaking method :**
 - The Handshaking scheme provides degree of flexibility and reliability because the successful completion of data transfer relies on active participation by both units.
 - If any of one unit is faulty, the data transfer will not be completed. Such an error can be detected by means of a Timeout mechanism which provides an alarm if the data is not completed within time.

❖ **Modes of data transfer :**

- Binary information received from an external device is usually stored in memory for later processing.
- Information transferred from the central computer into an external device originates in the memory unit.
- The CPU merely executes the I/O instructions and may accept the data temporarily, but the ultimate source or destination is the memory unit.
 1. Programmed I/O
 2. Interrupt–Initiated I/O
 3. Direct Memory Access (DMA)

1. Programmed I/O :

- Programmed I/O operations are the result of I/O instructions written in the computer program.
- Each data item transfer is initiated by an instruction in the program. Usually, the transfer is to and from a CPU register and peripheral.
- Other instructions are needed to transfer the data to and from CPU and memory.
- Transferring data under program control requires constant monitoring of the peripheral by the CPU.
- Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.

2. Interrupt–Initiated I/O :

- In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer.
- This is a time–consuming process since it keeps the processor busy needlessly.
- It can be avoided by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device.
- In the meantime, the CPU can proceed to execute another program.
- The interface meanwhile keeps monitoring the device. When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer.
- Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing.

3. Direct Memory Access (DMA) :

- Transfer of data under programmed I/O is between CPU and peripheral.
- In direct memory access (DMA), the interface transfers data into and out of the memory unit through the memory bus.
- The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred and then proceeds to execute other tasks.

- When the transfer is made, the DMA requests memory cycles through the memory bus.
- When the request is granted by the memory controller, the DMA transfers the data directly into memory.
- The CPU simply delays its memory access operation to allow the direct memory I/O transfer. Since peripheral speed is usually slower than processor speed, I/O-memory transfers are infrequent compared to processor access to memory.

(a) Interrupt-Initiated I/O :

- In this method an interrupt facility an interrupt command is used to inform the device about the start and end of transfer. In the meantime, the CPU executes another program.
- When the interface determines that the device is ready for data transfer it generates an Interrupt Request and sends it to the computer.
- When the CPU receives such a signal, it temporarily stops the execution of the program and branches to a service program to process the I/O transfer and after completing it returns back to task, what it was originally performing.
- In this type of IO, computer does not check the flag. It continues to perform its task.
- Whenever any device wants the attention, it sends the interrupt signal to the CPU.
- CPU then deviates from what it was doing, store the return address from PC and branch to the address of the subroutine.
- There are two ways of choosing the branch address:
 - Vectored Interrupt
 - Non-vectored Interrupt
- In vectored interrupt the source that interrupts the CPU provides the branch information. This information is called interrupt vectored.
- In non-vectored interrupt, the branch address is assigned to the fixed address in the memory.

□ Check Your Progress – 2 :

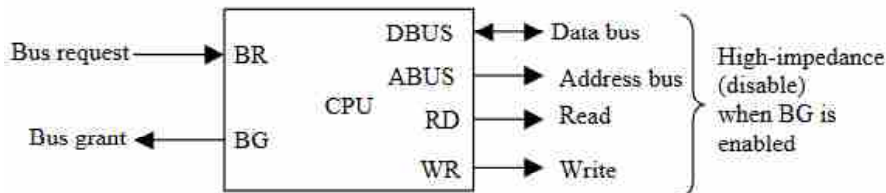
1. What is Asynchronous Data Transfer ? Explain Handshaking in detail.
.....
.....
.....
.....
.....
2. Write a note on mode of Data Transfer.
.....
.....
.....
.....
.....

12.4 Concept of Programmed I/O :

- Programmed I/O operations are the result of I/O instructions written in the computer program.
- Each data item transfer is initiated by an instruction in the program. Usually, the transfer is to and from a CPU register and peripheral.
- Other instructions are needed to transfer the data to and from CPU and memory.
- Transferring data under program control requires constant monitoring of the peripheral by the CPU.
- Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.

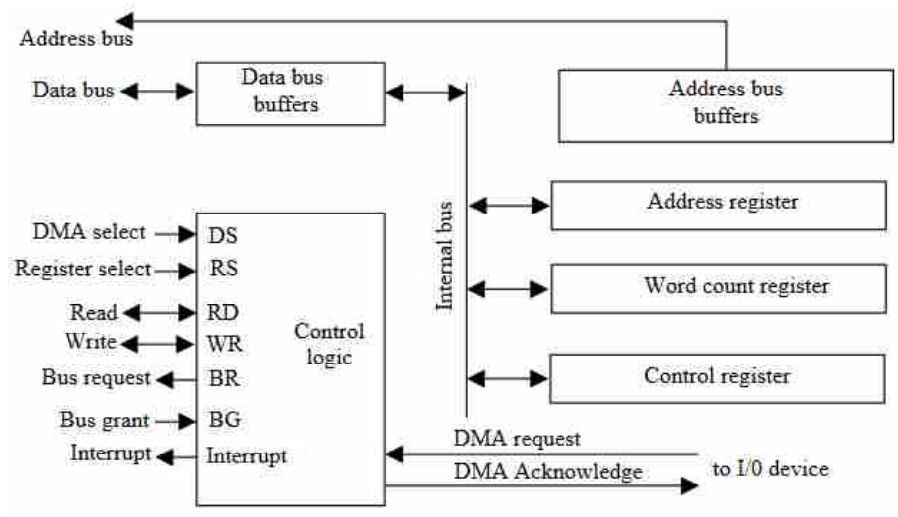
12.5 DMA :

- The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU.
- Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called direct memory access (DMA).
- During DMA transfer, the CPU is idle and has no control of the memory buses.
- A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.
- The CPU may be placed in an idle state in a variety of ways. One common method extensively used in microprocessors is to disable the buses through special control signals.



- Above figure shows two control signals in the CPU that facilitate the DMA transfer.
- The bus request (BR) input is used by the DMA controller to request the CPU to resign control of the buses. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and write lines into a high-impedance state behaves like an open circuit, which means that the output is disconnected and does not have a logic significance.
- The CPU activates the Bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state.
- The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor involvement.
- When the DMA terminates the transfer, it disables the bus request line.
- The CPU disables the bus grant, takes control of the buses, and returns to its normal operation.

- **DMA Controller :**
- The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device.
- In addition, it needs an address register, a word count register, and a set of address lines.
- The address register and address lines are used for direct communication with the memory. The word count register specifies the number of words that must be transferred.
- The data transfer may be done directly between the device and memory under control of the DMA.



- Above figure shows the block diagram of a typical DMA controller.
- The unit communicates with the CPU via the data bus and control lines.
- The registers in the DMA are selected by the CPU through the address bus by enabling the DS (DMA select) and RS (register select) inputs.
- The RD (read) and WR (write) inputs are bidirectional.
- When the BG (bus grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers.
- When BG = 1, the CPU has resigned the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.
- The DMA controller has three registers: an address register, a word count register, and a control register.
- The *address register* contains an address to specify the desired location in memory. The address bits go through bus buffers into the address bus. The address register is incremented after each word that is transferred to memory.
- The *word count* register is incremented after each word that is transferred to memory. The word count register holds the number of words to be transferred. This register is decremented by one after each word transfer and internally tested for zero.

- The *control register* specifies the mode of transfer. All registers in the DMA appear to the CPU as I/O interface registers. Thus, the CPU can read from or write into the DMA registers under program control via the data bus.

□ Check Your Progress – 3 :

1. Explain detailed note on DMA.

.....
.....
.....
.....
.....

2. _____ register is incremented after each word that is transferred to memory.

- | | |
|----------------------|-------------------|
| (a) Count Register | (b) Word Count |
| (c) Address Register | (d) None of Above |

3. RS Stands for _____.

- | | |
|---------------------|---------------------|
| (a) Register Select | (b) Register Start |
| (c) Register Stop | (d) Register Search |

4. BG Stands for _____.

- | | |
|---------------|------------------|
| (a) Bus Give | (b) Bus Graphics |
| (c) Bus Grant | (d) Bus Group |

12.6 Let Us Sum Up :

Then we also discuss input/ output interface and its working methods. We also seen asynchronous data transfer and different data transfer modes.

Finally, we have seen how we do programmed in input and output and DMA definition and works.

12.7 Suggested Answer for Check Your Progress :

□ Check Your Progress 1 :

- 1 : See Section 12.2 2 : a 3 : b

□ Check Your Progress 2 :

See Section 12.3

□ Check Your Progress 3 :

- 1 : See Section 12.5 2 : b
3 : a 4 : c

12.8 Glossary :

1. **Control command :** A control command is issued to activate the peripheral and to inform it what to do.
2. **Status command :** A status command is used to test various status conditions in the interface and the peripheral.

3. **Data Output command** : A data output command causes the interface to respond by transferring data from the bus into one of its registers.
4. **Data Input command** : The data input command is the opposite of the data output.

12.9 Assignment :

1. What is DMA ? Explain DMA Controller.
2. Explain Handshaking method.

12.10 Activities :

1. Find out the system where Direct Memory Access (DMA) is used.

12.11 Case Study :

1. Find out Input/Output interface in all available input and output device.

12.12 Further Readings :

1. Digital Logic and Computer Design, Morris Mano
2. Digital Electronics, Anand Kumar

UNIT STRUCTURE

- 13.0 Learning Objectives
- 13.1 Introduction
- 13.2 Memory Hierarchy
- 13.3 Primary Memory
 - 13.3.1 RAM and Types of RAM
 - 13.3.2 ROM and Types of ROM
- 13.4 Secondary Memory
 - 13.4.1 Magnetic Disk
 - 13.4.2 Magnetic Tape
- 13.5 Optical Memory (CDROM)
- 13.6 Concept of Virtual Memory
- 13.7 Concept of Cache and Their Need
- 13.8 Let Us Sum Up
- 13.9 Suggested Answer for Check Your Progress
- 13.10 Glossary
- 13.11 Assignment
- 13.12 Activities
- 13.13 Case Study
- 13.14 Further Readings

13.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Memory Hierarchy
- Primary Memory
- Secondary Memory
- Optical Memory
- Concept of Virtual Memory
- Concept of Cache Memory

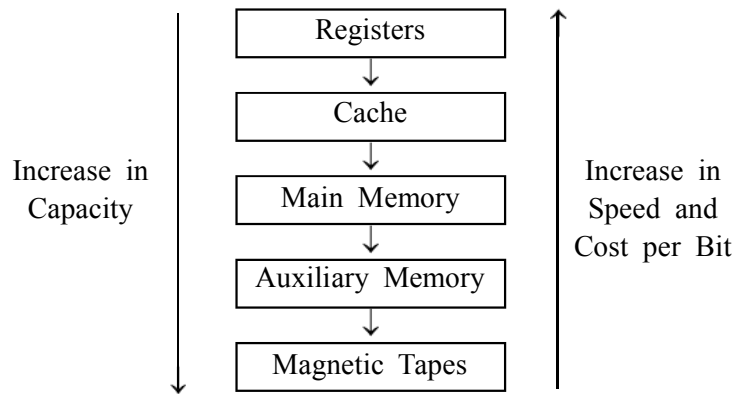
13.1 Introduction :

A memory is just like a human brain. It is used to store data and instructions. Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored. The memory is divided into large number of small parts called cells. Each location or cell has a unique address, which varies from zero to memory size minus one.

Memory can be either volatile or non-volatile memory. Volatile memory is memory that loses its contents when the computer or hardware device loses power. Computer RAM is an example of volatile memory. It is why if your computer freezes or reboots when working on a program, you lose anything that hasn't been saved. Non-volatile memory, sometimes abbreviated as NVRAM, is memory that keeps its contents even if the power is lost. EPROM is an example of non-volatile memory.

13.2 Memory Hierarchy :

- Memory plays a very vital role as it stores all the programs and data.
- There is not enough space in a single memory unit to accommodate all the programs and the data.



- Also, all the information stored is not needed by processor at the same time. Therefore, it is preferable to use low-cost memory storage devices to keep the programs and data that are not used by the processor.
- Generally, the main memory keeps the data that is currently used by the processor.
- The memory hierarchy consists of all the storage devices in the computer from the slow but high-capacity auxiliary memory to a fast but low-capacity cache memory.
- At the bottom of the hierarchy, we have the magnetic tape memory used to store removable files. The capacity of magnetic tape is large, but the speed is very slow.
- Magnetic disks are used to keep the programs and data in the computer. Also, the capacity is large but the speed is slow.
- The main memory interacts with CPU; thus, the data in auxiliary memory is brought in the main memory.
- Programs that are not currently in used are sent back to the auxiliary memory.
- A cache memory is used to increase the speed of processing by bringing the programs and data from the main memory into the cache memory. It works with the speed of the processor.

The processor is usually faster than main memory which results in slow processing speed. Thus, a memory which is fast and having low capacity is employed between the main memory and the processor, resulting in less access time and improving the performance of the processor.

□ Check Your Progress – 1 :

1. What is Memory ? Explain Memory Hierarchy.

2. _____ memory is used to increase the speed of processing.
 (a) Register (b) Cache (c) Auxiliary (d) Main
3. _____ memory interacts with CPU.
 (a) Main (b) Auxiliary (c) Magnetic (d) None of Above
4. _____ memory loses its contents when the computer or hardware device loses power.
 (a) Register (b) Auxiliary (c) Volatile (d) Non-Volatile

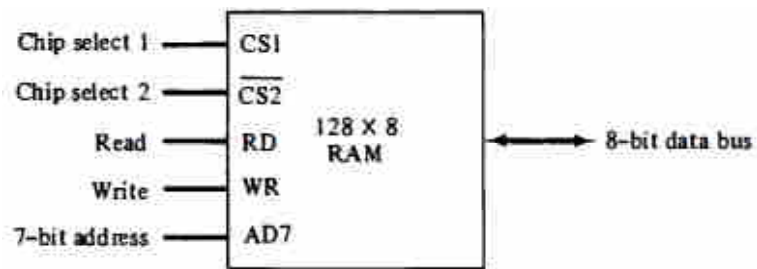
13.3 Primary Memory :

- Main memory is central storage of computer system. It is relatively large and fast memory to store the programs and data during computer operations.
- The average time required to read the contents of a location or write into a location in main memory is called *Access time*.
- Main memory has an equal access time for all its locations irrespective of its address. Thus, it is known as Random Access Memory
- After the access is over, main memory needs time to settle down for the next transfer to start. This is known as *settling time or recovery time*.
- The sum of access time and *settling time is known as cycle time*.

13.3.1 RAM and Types of RAM :

- The principal technology used for the main memory is based on semiconductor integrated circuits.
- Integrated circuit RAM chips are available in two possible operating modes, static and dynamic.
- **Types of RAM :**
 - There are two types of RAM which are as follows :
 - 1. Static RAM :**
 - The static RAM consists essentially of internal flip–flops that store the binary information. The stored information remains valid as long as power is applied to the unit.
 - The static RAM is easier to use and has shorter read and write cycles.
 - 2. Dynamic RAM :**
 - The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors.
 - The capacitors are provided inside the chip by MOS transistors.
 - The stored charge on the capacitors tend to discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory.
 - Refreshing is done by cycling through the words every few milliseconds to restore the decaying charge.
 - The dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip.

- A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip only when needed.
- Another common feature is a bidirectional data bus that allows the transfer of data either from memory to CPU during a read operation, or from CPU to memory during a write operation.
- A bidirectional bus can be constructed with three-state buffers.
- A three-state buffer output can be placed in one of three possible states: a signal equivalent to logic 1, a signal equivalent to logic 0, or a high impedance state.
- The logic 1 and 0 are normal digital signals.
- The high impedance state behaves like an open circuit, which means that the output does not carry a signal and has no logic significance.



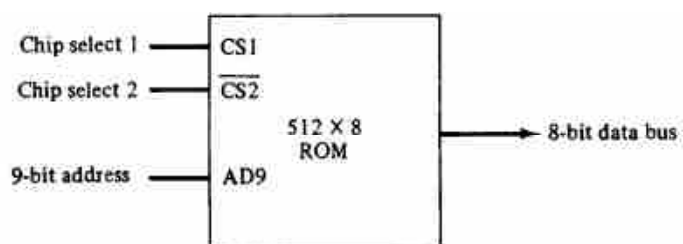
(A) Block Diagram

CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	x	x	Inhibit	High-impedance
0	1	x	x	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	x	Read	Output data from RAM
1	1	x	x	Inhibit	High-impedance

(B) Function Table

13.3.2 ROM and Types of ROM :

- Most of the main memory in a general-purpose computer is made up of RAM integrated circuit chips, but a portion of the memory may be constructed with ROM chips.
- Originally, RAM was used to refer to a random-access memory, but now it is used to designate a read/write memory to distinguish it from a read-only memory, although ROM is also random access.
- RAM is used for storing the bulk of the programs and data that are subject to change.
- ROM is used for storing programs that are permanently resident in the computer.



- A ROM chip is organized externally in a similar manner. However, since a ROM can only read, the data bus can only be in an output mode. The block diagram of a ROM chip is shown in above figure.

- The nine address lines in the ROM chip specify any one of the 512 bytes stored in it. The two chips select inputs must be $CS1 = 1$ and $CS2 = 0$ for the unit to operate.
- Otherwise, the data bus is in a high-impedance state. There is no need for a read or write control because the unit can only read. Thus, when the chip is enabled by the two select inputs, the byte selected by the address lines appears on the data bus.

- **Types of ROMs :**

- There are basically two types of ROMs which are as follows:

1. **PROM (Programmed Read Only Memory) :**

- It is a field programmable device. As in ROM the contents cannot be modified after manufacturing.
- PROM gives the facility to the user to program ROM as per his requirements. But once programmed, the contents cannot be deleted.

2. **EPROM (Erasable Programmable Read Only Memory) :**

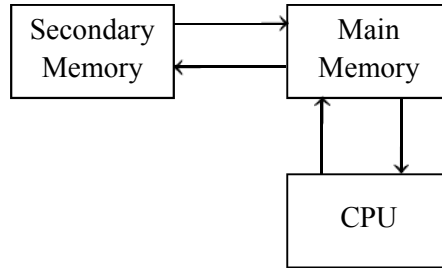
- EPROM as the name suggests allows the contents of PROM to be deleted by using ultraviolet rays.
- Deleting one or more locations is not possible but the entire contents of EPROM are erased.
- Thus, contents of PROM which is used as ROM in a system can be erased and reprogrammed to suit the changed requirements.

- **Check Your Progress – 2 :**

1. SRAM stands for _____.
 - (a) Small Random-Access Memory
 - (b) Static Random-Access Memory
 - (c) System Random-Access Memory
 - (d) Service Random-Access Memory
2. DRAM stands for _____.
 - (a) Dynamic Random-Access Memory
 - (b) Dual Random-Access Memory
 - (c) Direct Random-Access Memory
 - (d) Distinct Random-Access Memory
3. PROM stands for _____.
 - (a) Partial Read-Only Memory (b) Process Read-Only Memory
 - (c) Page Read-Only Memory (d) Programmed Read-Only Memory
4. EPROM stands for _____.
 - (a) Erasable Process Read-Only Memory
 - (b) Erasable Partial Read-Only Memory
 - (c) Erasable Programmable Read-Only Memory
 - (d) Erasable Page Read-Only Memory

13.4 Secondary Memory :

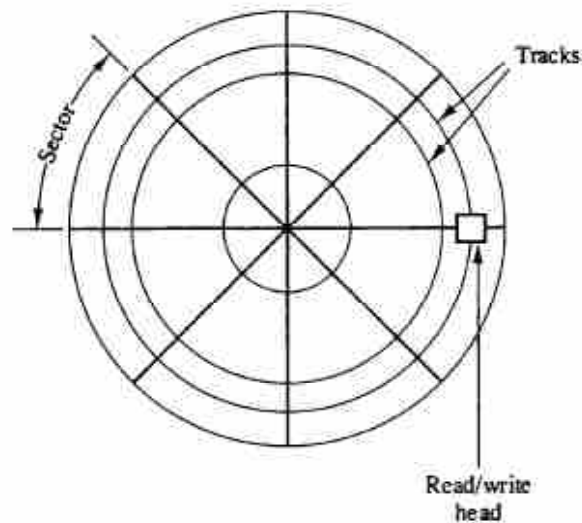
- It is also known as auxiliary memory.
- The most common secondary memory devices in our system are magnetic disks, tapes and optical disk. It stores a large amount of data and programs.
- As the capacity is more, they are slower and cheaper than main memory.



- The CPU generally brings a part of the program from the secondary memory into main memory and thus it treats secondary devices as peripheral devices.

13.4.1 Magnetic Disk :

- A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material.
- Often both sides of the disk are used and several disks may be stacked on one spindle with read/write heads available on each surface.



- All disks rotate together at high speed and are not stopped or started from access purposes.
- Bits are stored in the magnetized surface in spots along concentric circles called tracks.
- The tracks are commonly divided into sections called sectors.
- In most systems, the minimum quantity of information which can be transferred is a sector. The sub division of tone disk surface into tracks and sectors.

13.4.2 Magnetic Tape :

- A magnetic tape transport consists of the electrical, mechanical, and electronic components to provide the parts and control mechanism for a magnetic-tape unit.

- The tape itself is a strip of plastic coated with a magnetic recording medium.
- Bits are recorded as magnetic spots on the tape along several tracks.
- Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit.
- Read/write heads are mounted one in each track so that data can be recorded and read as a sequence of characters.
- Magnetic tape units can be stopped, started to move forward or in reverse, or can be rewound.

☐ **Check Your Progress – 3 :**

1. Explain Magnetic Disk and Magnetic Tape secondary storage device.

.....

.....

.....

.....

.....

13.5 Optical Memory (CDROM) :

- An **optical disc** is a type of storage medium that consists of a flat, round, portable disc made of metal, plastic, and lacquer that is written and read by a laser.
- Optical discs used in computers typically are 4.75 inches in diameter and less than 1/20 of an inch thick.
- Three widely used types of optical discs are **CDs** (compact discs), **DVDs** (digital versatile discs or sometimes digital video discs), and **Blu-ray Discs**.
- Many different formats of optical discs are available today. Some are **read only**, meaning users cannot write (save) on the media. Others are **read/write**, which allows users to save on the disc just as they save on a hard drive.
- With most discs, you can read and/or write on one side only.
- **What is the life span of an optical disc ?**
- Manufacturers claim that a properly cared for, high-quality optical disc will last 5 years but could last up to 100 years.
- Tips for proper care of optical discs include the following:
 - ✓ Never bend a disc; it may break.
 - ✓ Do not expose discs to extreme temperatures or humidity. The ideal temperature range for disc storage is 50 to 70 degrees Fahrenheit.
 - ✓ Stacking discs, touching the underside of discs, or exposing them to any type of contaminant may scratch a disc.
 - ✓ Place an optical disc in its protective case, called a jewel case, when you are finished using it, and store it in an upright (vertical) position.

- **CDs :**
 - CDs are available in three basic formats: read-only, recordable, and rewritable.
 - 1. A **CD-ROM** (CD-read-only memory) is a type of optical disc that users can read but not write on (record) or erase – hence, the name read-only.
 - 2. **CD-R** (CD-recordable) is an optical disc on which users can write once, but not erase, their own items, such as text, graphics, and audio. Because a CD-R can be written on only one time, the format of these discs sometimes is called *WORM* (write once, read many). Some CD-Rs are *multisession*, which means you can write on part of the disc at one time and another part at a later time – if the disc has free space.
 - 3. A CD-RW (CD-rewritable) is an erasable multisession disc user can write on multiple times.
- **DVDs :**
 - DVDs to have greater storage capacities and higher resolutions than CDs.
 - A more expensive DVD format is Blu-ray, which has a higher capacity and better quality than standard DVDs, especially for high-definition audio and video.
 - As with CDs, DVDs are available in three basic formats: read-only, recordable, and rewritable.
 - 1. A **DVD-ROM** (DVD-read-only memory) is a high-capacity optical disc that users can read but not write on or erase. DVD-ROMs store movies, music, music videos, huge databases, and applications you install on a computer.
 - 2. **DVD-R** and **DVD+R** are competing DVD-recordable WORM formats, on which users can write once but not erase their own items, including video, audio, photos, graphics, and text.
 - 3. **DVD-RW**, **DVD+RW**, and **DVD+RAM** are competing DVD-rewritable formats that users can write on multiple times.
- **Check Your Progress – 4 :**
 - 1. CD Stands for _____.
 - (a) Count Disc
 - (b) Compact Disc
 - (c) Computer Disc
 - (d) Common Disc
 - 2. DVD Stands for _____.
 - (a) Digital Versatile Disc
 - (b) Direct Versatile Disc
 - (c) Digital Various Disc
 - (d) Direct Various Disc

13.6 Concept of Virtual Memory :

- Virtual memory is a concept used to construct large programs through the physical memory has limited space.
- The operating system keeps the large program in the secondary memory and brings only a part of the program in main memory.
- The address generated by the CPU is known as virtual address. Each virtual address is mapped to physical address in main memory.

- The mapping or translation is handled by the operating system and the CPU hardware.

13.7 Concept of Cache and Their Need :

- Cache memory has small capacity compared to main memory, but is relatively faster and expensive.
- Analysis of large programs have shown that a particular point of time, references to memory tend to be limited within a few localized portions in memory. This is known as *locality of reference*.
- For example, in a program the function and subroutines are called frequently. The CPU refers repeatedly the function in memory. Every time the function or subroutine is called, the same set of instructions are fetched from the memory. Thus, they tend to localize the reference to memory.
- Similarly, the memory references to data also tend to be localized. The result of all these observations shows that only few of the localized portion of memory are referred repeatedly whereas the other areas are used very less.
- If these repeatedly used portions of program are placed in a small and fast memory than the access time can be reduced thus reducing the execution time of the program.
- Such small and fast memory is referred to as Cache memory. It is placed between the CPU and the main memory.
- Cache memory is the fastest in memory hierarchy and work nearly with speed of processor.
- The cache memory can be of two types :
 1. **Unified cache** : It is common cache memory that can store both instructions and data.
 2. **Split cache** : It has a separate cache to store instruction and data.
- The basic characteristic of cache memory is its fast access time. Therefore, very little or no time must be wasted when searching for words in the cache.
- The transformation of data from main memory to cache memory is referred to as a mapping process.

13.8 Let Us Sum Up :

Memory is the most essential element of a computing system because without its computer can't perform simple tasks. Computer memory is of two basic type – Primary memory (RAM and ROM) and Secondary memory (hard drive, CD, etc.). Random Access Memory (RAM) is primary–volatile memory and Read Only Memory (ROM) is primary–non–volatile memory.

Computer memory can vary is that some types are non–volatile, which means they can store data on a long–term basis even when there is no power. And some types are volatile, which are often faster, but which lose all the data stored on them as soon as the power is switched off.

13.9 Suggested Answer for Check Your Progress :

- ❑ **Check Your Progress 1 :**
1 : See Section 13.1 & 13.2 2 : b 3 : a
4 : c
- ❑ **Check Your Progress 2 :**
1 : b 2 : a 3 : d 4 : c
- ❑ **Check Your Progress 3 :**
See Section 13.4
- ❑ **Check Your Progress 4 :**
1 : b 2 : a

13.10 Glossary :

1. **Memory :** A memory is just like a human brain. It is used to store data and instructions. Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored.
2. **Volatile Memory :** *Volatile* memory is memory that loses its contents when the computer or hardware device loses power. Computer RAM is an example of volatile memory.
3. **Non-Volatile Memory :** *Non-volatile* memory is memory that keeps its contents even if the power is lost. EPROM is an example of non-volatile memory.
4. **Access Time :** The average time required to read the contents of a location or write into a location in main memory is called *Access time*.
5. **Setting Time / Recovery Time :** After the access is over, main memory needs time to settle down for the next transfer to start. This is known as *settling time or recovery time*.
6. **Cycle Time :** The sum of access time and settling time is known as *cycle time*.

13.11 Assignment :

1. Explain primary memory in detail.
2. Write a detailed note on Optical Memory.
3. What is Cache memory ? Explain need of Cache Memory.

13.12 Activities :

1. Find out which memory/ memories you are using in your desktop, laptop and mobile.

13.13 Case Study :

1. List out 3G mobiles, 4G mobiles and 5G mobiles. Then make a summary report for processor and memory for 3G, 4G and 4G mobiles.

13.14 Further Readings :

1. Digital Electronics, Anand Kumar
2. Digital Logic and Computer Design, Morris Mano

UNIT STRUCTURE

- 14.0 Learning Objectives
- 14.1 Introduction
- 14.2 (SR, JK, D, T) its Truth–Tables
- 14.3 Applications of Flip–Flops
- 14.4 Clocks
- 14.5 3–4–bit Registers
- 14.6 Shift Register
- 14.7 Synchronous/Asynchronous Binary Counters
- 14.8 Let Us Sum Up
- 14.9 Suggested Answer for Check Your Progress
- 14.10 Glossary
- 14.11 Assignment
- 14.12 Activities
- 14.13 Case Study
- 14.14 Further Readings

14.0 Learning Objectives :

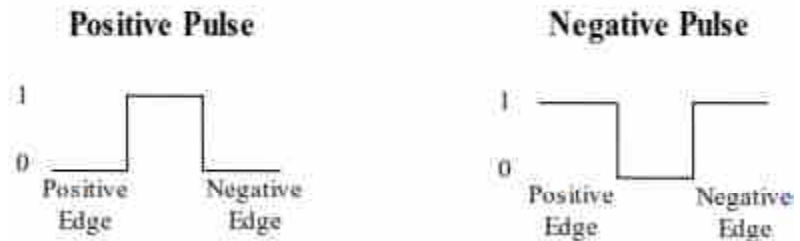
After learning this unit, you will be able to understand :

- Flip–Flops and its Truth–Table
- Transfer Circuit
- Clocks
- 3–4–bit Registers
- Shift Register
- Synchronous/Asynchronous Binary Counters

14.1 Introduction :

- Flip–Flop is two state device which offer basic memory cell for sequential logic operations.
- Flip–Flop is a term referring to an electronic circuit and are used for digital data storage and transfer.
- The name "Flip–Flop" comes from the circuit's nature of alternating between two states when a current is applied to the circuit as positive (1) voltage or negative (0) voltage.
- Flip–Flop will maintain its state until it receives an input pulse, called a trigger, which forces it to alternate its state.
- Once the circuit changes state it remains in that state until another trigger is received.

- **Triggering of Flip-Flop :**
- The state of Flip-Flop is changed by a changing the input signals, this change is called a trigger and the transition this cause is said to trigger the flip-flop.
- Clocked flip-flops are triggered by pulse.
- The clock pulse goes through two signal transition : from 0 to 1 and return from 1 to 0.

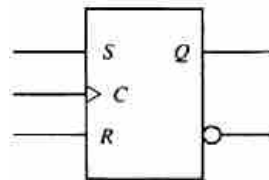


14.2 (SR, JK, D, T) its Truth-Tables :

- **Types of Flip-Flops :**
- There are four types of flip-flops which are as follows :
 1. **RS/SR/SR-Latch/Basic Flip-Flop**
 2. **D Flip-Flop**
 3. **JK Flip-Flop**
 4. **T Flip-Flop**

1. SR Flip-Flop :

- **Symbol :**



- It has three inputs, S for SET, R for RESET, and C for CLOCK.
- It has two output Q and Q', which is indicated by a small circle at the other output.
- An arrowhead shaped symbol in front of the letter C indicates a dynamic input.
- The dynamic indicator denotes the fact that the flip-flop responds to a positive transition (from 0 to 1) of the input clock signal.
- If there is no signal at the clock input C, the output of the circuit cannot change irrespective of the values at inputs S and R.
- Only when the clock signal changes from 0 to 1 the output be affected according to the values in inputs S and R.
- If S = 0 and R = 1 when C changes from 0 to 1, output Q is cleared to 0.
- If S = 1 and R = 0 when C changes from 0 to 1, output Q is set to 1.
- If both S and R are 0 during the clock transition, the output does not change.

- When both S and R are equal to 1, the output is unpredictable.

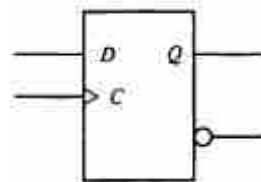
• **Characteristics Table :**

S	R	Q(t+1)	Action
0	0	Q(t)	No Change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	?	Indeterminate

- The S and R columns give the binary value of the two inputs.
- Q(t) is the binary state of the Q output at a given time (referred to as present state).
- Q(t+1) is the binary state of the output Q after the occurrence of a clock transition (referred to as next state).
- If S = R = 0, a clock transition produces no change of state.
- If S = 0 and R = 1, the flip-flop goes to the 0(Clear) state.
- If S = 1 and R = 0, the flip-flop goes to the 1(Set) state.
- The SR flip-flop should not be pulsed when S = R = 1 since it produces an indeterminate next state.
- This indeterminate condition makes the SR flip-flop difficult to manage and therefore it is rarely used in practice.

2. D Flip-Flop :

• **Symbol :**



- The D (Data) flip-flop is a slight modification of the SR flip-flop.
- An SR flip-flop is converted to D flip-flop by inserting an inverter between S and R and assigning the symbol D to the single input.
- The D input is sampled during the occurrence of a clock transition from 0 to 1.
- If D = 1, the output of the flip-flop goes to the 1 state, but if D = 0, the output of the flip-flop goes to the 0 state.

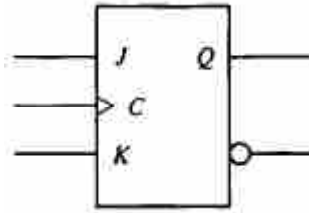
• **Characteristics Table :**

D	Q(t+1)	Action
0	0	Clear to 0
1	1	Set to 1

- We note that the next state Q(t+1) is determined from the D input.
- The relationship can be expressed by a characteristic equation : $Q(t+1) = D$. This means that the Q output of the flip-flop receives its value from D input every time that the clock signal goes through a transition from 0 to 1.

3. JK Flip-Flop :

• **Symbol :**



- A JK flip-flop is a refinement of the SR flip-flop in that the indeterminate condition of the SR type is defined in the JK type.
- Inputs J and K behave like inputs S and R to set and clear the flip-flop, respectively.
- When inputs J and K are both equal to 1, a clock transition switches the outputs of the flip-flop to their complement state.

• **Characteristics Table :**

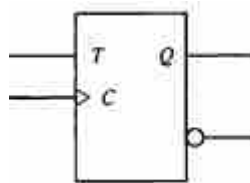
J	K	Q(t+1)	Action
0	0	Q(t)	No Change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	Q'(t)	Complement

- The J input is equivalent to the S (set) input of the SR flip-flop, and the K input is equivalent to the R (Clear) input.
- Instead of the indeterminate condition, the JK flip-flop has a complement condition $Q(t+1) = Q'(t)$ when both J and K are equal to 1.

4. T Flip-Flop :

- T stands for toggle flip-flop.
- T flip-flop is obtained from a JK type when inputs J and K are connected to provide a single input designated by T.

• **Symbol :**



• **Characteristics Table :**

T	Q(t+1)	Action
0	Q(t)	No Change
1	Q'(t)	Complement

- The T flip-flop therefore has only two conditions :
- When $T = 0$ ($J = K = 0$) a clock transition does not change the state of the flip-flop.
- When $T = 1$ ($J = K = 1$) a clock transition complements the state of the flip-flop.

□ Check Your Progress – 1 :

1. What is Flip-Flop ? Explain types of Flip-Flop.

.....

2. Flip-Flop is _____ state device.

- (a) Two (b) Three (c) Four (d) None of Above

3. The state of Flip-Flop is changed by a changing the input signals, this change is called a _____.

- (a) Voltage (b) Pulse (c) Trigger (d) All of Above

14.3 Applications of Flip-Flops :

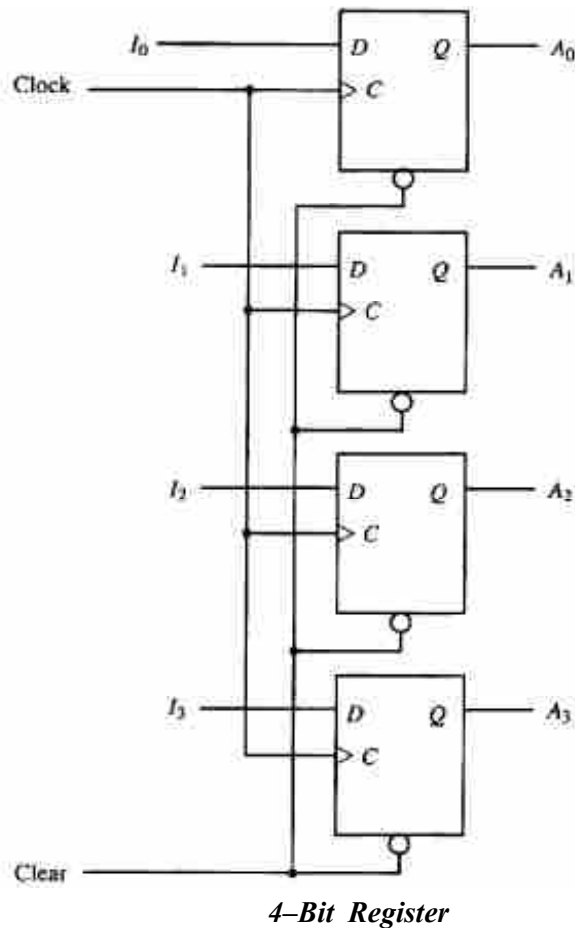
- These are the various types of flip-flops being used in digital electronic circuits and the applications of Flip-flops are as specified below.
 - Counters
 - Frequency Dividers
 - Shift Registers
 - Storage Registers

14.4 Clocks :

- The timing for all registers in the basic computer is controlled by a master clock generator.
- The clock pulses are applied to all flip-flops and registers in the system, including the flip-flops and registers in the control unit.
- The clock pulses do not change the state of a register unless the register is enabled by a control signal.
- The control signals are generated in the control unit and provide control inputs for the multiplexers in the common bus, control inputs in processor registers, and microoperations for the accumulator.

14.5 3-4-bit Registers :

- Its basic function is to store the information within a digital system. It is called buffer register or register.
- Register in which it is possible to give 4 inputs at a time and to get 4 output is called parallel load registers.
- A 4-bit register is shown in below figure. A clock transition applied the CP. Inputs of the register will load all four inputs L0 through L3 in parallel.



- A register constructed with 4 D-type flip-flop and a common clock pulse input.
- The clock pulse input CP, enables all flip-flop so that information presently available at the four inputs can be transferred into the 4-bit register.
- The four output can be sampled to obtain information presently stored in the register.
- Sometimes it may be possible that at a time all inputs are not available then clock is off, so when all inputs are available then the clock is on.
- So, in register error is not possible. That is the use of register in data storage.

□ Check Your Progress – 2 :

1. Explain 4-bit register in detail.

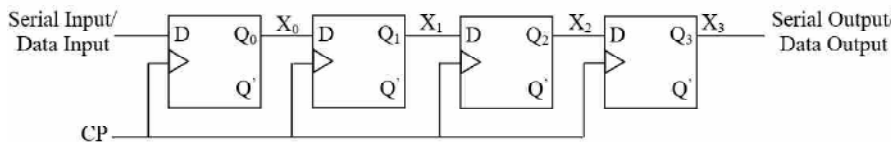
2. Register in which it is possible to give 4 inputs at a time and to get 4 output is called _____.
 (a) Parallel load registers (b) Buffer Register
 (c) Control Register (d) Data Register

14.6 Shift Register :

- A register capable of shifting its binary information in one, either to the right or to the left or in both directions is called a shift register.
- Shift register consists of a chain of flip-flops in cascade, where the output of one flip-flop connected to the input of the next flip-flop.
- All flip-flops receive common clock pulses that initiate the shift from one stage to the next.
- A *serial input* shift register has a single external input called the serial input entering in an out most flip-flop. Each remaining flip-flop uses the output of the previous flip-flop as its input; with the last flip-flop producing the external output called the *serial output*.
- A register capable of shifting in one direction is called a *unidirectional shift register*.
- A register that can shift in both directions is called *bi-directional shift register*.

• Shift Right Register :

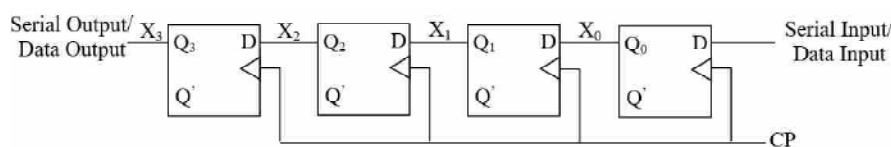
- A register capable of shifting its binary information to the right is called Shift Right Register.
- The logical configuration of a shift register consists of a chain of flip-flops connected, the output of one flip-flop connected to the input of the next flip-flop.
- A flip-flop receives a common clock pulse which causes the shift from one stage to the next stage.



- The Q output of a given flip-flop is connected to the D input of the flip-flop at its right.
- Each clock pulse send it signal to next flip-flop. This register shifts its contents to the right so it is called shift right register.

• Shift Left Register :

- A register capable of shifting its binary information to the left is called Shift Left Register.
- The logical configuration of a shift register consists of a chain of flip-flops connected, the output of one flip-flop connected to the input of the next flip-flop.
- A flip-flop receives a common clock pulse which causes the shift from one stage to the next stage.



- The Q output of a given flip-flop is connected to the D input of the flip-flop at its left.

- Each clock pulse send it signal to next flip-flop. This register shifts its contents to the left so it is called shift left register.

☐ Check Your Progress – 3 :

1. Write a note on Shift Register.
.....
.....
.....
.....
.....
2. A register capable of shifting in one direction is called a _____.
(a) Shift Left Register (b) Shift Right Register
(c) Unidirectional Shift Register (d) Bi-directional Shift Register
3. A register that can shift in both directions is called _____.
(a) Bi-directional Shift Register (b) Unidirectional Shift Register
(c) Buffer Register (d) Address Register

14.7 Synchronous/Asynchronous Binary Counters :

- **Introduction to Binary Counter :**
 - A counter is a mechanism, which calculate a step ahead and step behind current value.
 - In other words, it calculates any value on stepwise backward and forward.
 - A counter is a register, which goes through a pre-defined sequence of states when clock pulse is applied.
 - Clock pulse may originate from an external source. They may occur at uniform intervals of time or at random.
 - Counters are found in almost all equipment containing digital logic.
 - Counters are used for counting the number of occurrences of an event and are useful for generating timing signals to control the sequence of operations in digital computers.
 - The straight binary sequence is the simplest and most straightforward.
 - "A counter that follows the binary number sequence is called a binary counter".
 - An n-bit binary counter is a register of n flip-flops and associated gates that follows a sequence of states according to the binary count of n bits, from 0 to $2^n - 1$.
 - A simpler design procedure may be carried out from a direct inspection of the sequence of states that the register must undergo to achieve a straight binary count.
 - Going through a sequence of binary numbers such as 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111 and 1000.
 - ✓ Here, we note that the lower-order bit (Last Bit) is complemented after every count.

- ✓ Every other bit is complemented from one count to the next if and only if all its lower-order bits (Last Bits) are equal to 1.

– For example,

No	4-bit code	Description
0	0000	
1	0001	Last bit is complemented.
2	0010	Here, last bit is complemented and 2nd last bit is complemented because previous last bit = 1.
3	0011	Last bit is complemented.
4	0100	Here, last bit is complemented and 3rd last bit is complemented because 2nd last bit = 1.
5	0101	Last bit is complemented.
6	0110	Here, last bit is complemented and 2nd last bit is complemented because previous last bit = 1.
7	0111	Last bit is complemented.
8	1000	1000 obtained by:
		(a) Complementing the lower-bit order. (b) Complementing 2nd order-bit because the last bit of 0111 is 1. (c) Complementing 3rd order-bit because last two bits of 0111 are 1. (d) Complementing the 4th order-bit because the last three bits of 011 are 1.

- A counter circuit will usually employ flip-flops with complementing capabilities. Both T and JK flip-flops have this property.
- Remember that a JK flip-flop is complemented if both its J and K inputs are 1 and the clock goes through a positive transition.
- The output of the flip-flop does not change if $J = K = 0$.
- In addition, the counter may be controlled with an enable input that turns the counter on or off without removing the clock signal from the flip-flops.

• **Types of Counters :**

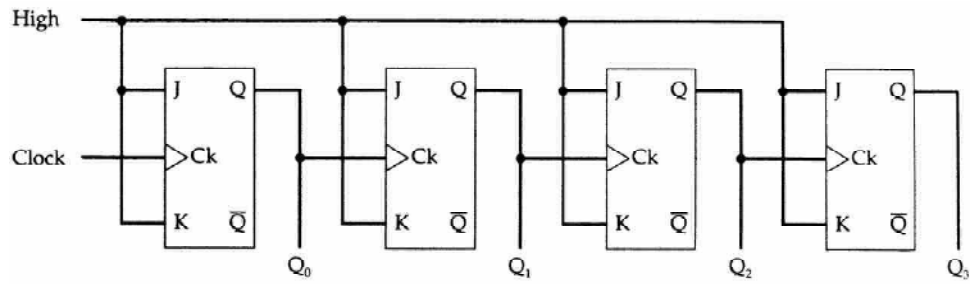
- There are basically two types of counters which are as follows :

1. Asynchronous or Ripple Counter

2. Synchronous Counter

1. Asynchronous or Ripple Counter :

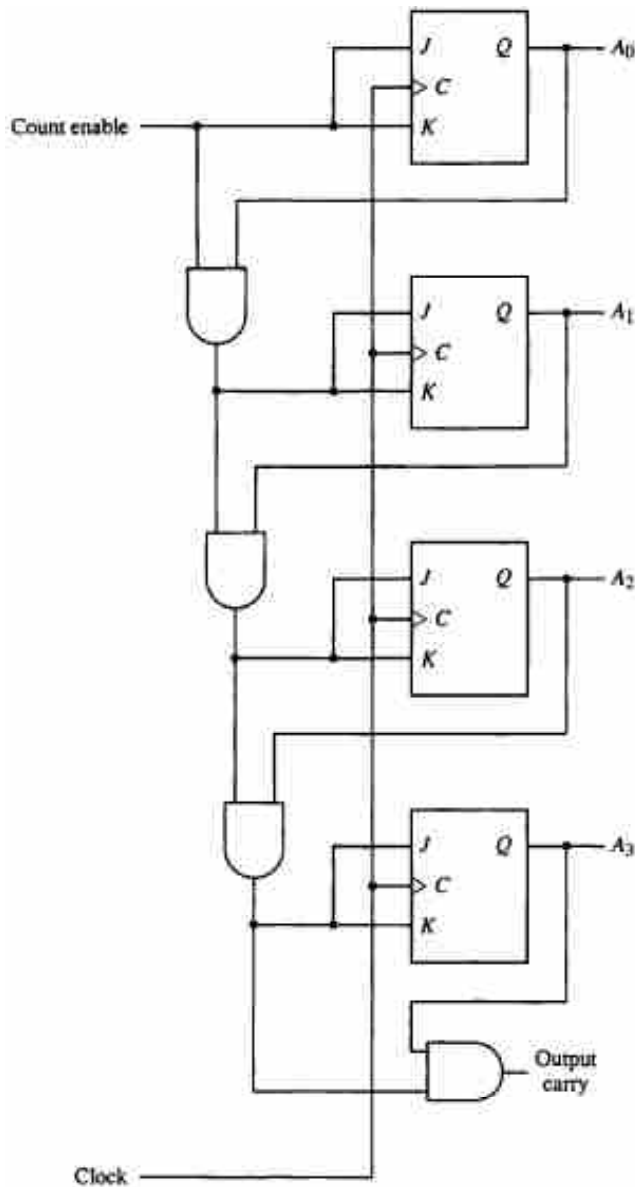
- In asynchronous counter, the change in state of one flip-flop trigger the other flip-flops.
- In other words, for these counters an external clock signal is applied to one flip-flop, and then the output of the previous flip-flop is connected to the clock of the next flip-flop.



- In an asynchronous counter or a ripple counter the clock input is applied only to the first flip-flop. The clock input to any subsequent flip-flop comes from the output of its immediately preceding flip-flop.
- The output of first flip-flop acts as the clock input to the second flip-flop, the output of second flip-flop feed the clock input of the third flip-flop and so on.

2. Synchronous Counter :

- Synchronous counters are relatively faster because the state of all flip-flop can be changed at the same time.
- Synchronous counter also known as a parallel counter, all the flip-flops in the counter change state at the same time with the input clock signal.
- Synchronous binary counters have regular pattern. The C inputs of all flip-flops receives the common clock.
- If the count enable is 0, all J and K inputs are maintained at 0 and the output of the counter does not change.
- The first stage A0 is complemented when the counter is enabled and the clock goes through a positive transition.
- Each of the other three flip-flops are complemented when all previous least significant flip-flops are equal to 1 and the count is enabled.
- The chain of AND gates generate the required logic for the J and K inputs.
- The output carry can be used to extend the counter to more stages, with each stage having an additional flip-flop and an AND gate.



4-bit Synchronous Binary Counter

□ **Check Your Progress – 4 :**

1. What is Counter ? Explain types of it.

.....

.....

.....

.....

.....

2. Lower-order bit is known as _____.

- (a) Last Bit
- (b) First Bit
- (c) Second Last Bit
- (d) None of Above

3. _____ is a register, which goes through a pre-defined sequence of states when clock pulse is applied.

- (a) Last Bit
- (b) Flip-Flop
- (c) Register
- (d) Counter

14.8 Let Us Sum Up :

A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs. Flip–flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems.

Flip–flops and latches are used as data storage elements. It is the basic storage element in sequential logic. But first, let's clarify the difference between a latch and a flip–flop.

14.9 Suggested Answer for Check Your Progress :

Check Your Progress 1 :

1 : See Section 14.1 & 14.2 2 : a 3 : c

Check Your Progress 2 :

1 : See Section 14.5 2 : a

Check Your Progress 3 :

1 : See Section 14.6 2 : c 3 : a

Check Your Progress 4 :

1 : See Section •• 2 : a 3 : d

14.10 Glossary :

1. **Flip–Flop** : Flip–Flop is a term referring to an electronic circuit and are used for digital data storage and transfer.
2. **Parallel Load Register** : Register in which it is possible to give 4 inputs at a time and to get 4 output is called parallel load registers.
3. **Counter** : A counter is a mechanism, which calculate a step ahead and step behind current value. A counter is a register, which goes through a pre–defined sequence of states when clock pulse is applied.

14.11 Assignment :

1. What is Counter ? Explain types of counter.

14.12 Activities :

Make a set–reset basic circuit with any flip–flop.

14.13 Case Study :

Take one printed circuit board and set and 3 flip–flop and make a power button with 3 switches.

14.14 Further Readings :

1. Digital Electronics, Anand Kumar
2. Digital Logic and Computer Design, Morris Mano

UNIT STRUCTURE

- 15.0 Learning Objectives
- 15.1 Introduction
- 15.2 Functions of CPU
- 15.3 Register Classification and Organization
- 15.4 Instruction Cycle
- 15.5 Instruction Formats
- 15.6 Addressing Modes
- 15.7 Let Us Sum Up
- 15.8 Suggested Answer for Check Your Progress
- 15.9 Glossary
- 15.10 Assignment
- 15.11 Activities
- 15.12 Case Study
- 15.13 Further Readings

15.0 Learning Objectives :

After learning this unit, you will be able to understand :

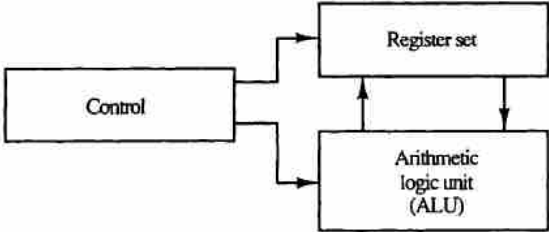
- Function of CPU
- Register Classification and Organization
- Instruction Cycle
- Instruction Formats
- Addressing Modes

15.1 Introduction :

- A central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions. The term has been used in the computer industry at least since the early 1960s.
- Processors contain a control unit and an arithmetic logic unit (ALU). These two components work together to perform processing operations.
- The control unit interprets and executes instructions in memory, and the arithmetic logic unit performs calculations on the data in memory.
- Resulting information is stored in memory, from which it can be sent to an output device or a storage device for future access, as needed.

15.2 Functions of CPU :

- The part of the computer that performs the bulk of data-processing operations is called the central processing unit and is referred to as the CPU.
- The CPU is made up of three major parts :



Major Components of CPU

- The register set stores intermediate data used during the execution of the instructions.
- The arithmetic logic unit (ALU) performs the required microoperations for executing the instructions.
- The control unit supervises the transfer of information among the registers and instructs the ALU as to which operation to perform.
- The CPU processes instructions it receives in the process of decoding data. In processing this data, the CPU performs four basic steps:

1. **Fetch** : Each instruction is stored in memory and has its own address. The processor takes this address number from the program counter, which is responsible for tracking which instructions the CPU should execute next.
2. **Decode** : All programs to be executed are translated into Assembly instructions. Assembly code must be decoded into binary instructions, which are understandable to your CPU. This step is called decoding.
3. **Execute** : While executing instructions, the CPU can do one of three things : Do calculations with its ALU, move data from one memory location to another, or jump to a different address.
4. **Store** : The CPU must give feedback after executing an instruction, and the output data is written to the memory.

☐ Check Your Progress – 1 :

1. What is CPU ? Explain function of CPU.

2. Which of following _____ performed by the CPU.
 (a) Fetch (b) Decode (c) Execute (d) All of Above

15.3 Register Classification and Organization :

- **Introduction to Register :**

- Register is a special purpose memory. This memory is vital for moving data in/out of the memory and process the data.
- A register is a group of flip–flops capable of storing one bit of information.
- An n–bit register has a group of n flip–flops and is capable of storing any binary information of n bits.
- The simplest possible register is one that consist of only flip–flops without any external gates.
- In addition to flip–flops, registers can have combinational gates that perform certain data processing tasks. The gates control how and when new information is transferred into the register.
- The transfer of new information into a register is referred to as a register load. If loading occurs simultaneously at a common clock pulse transition, we can say that the load is done in parallel.
- The load input in a register determines the action to be taken with each clock pulse.
- When the load input is 1, the data from the input lines is transferred into the register's flip–flops. When the load is 0, the data inputs are reserved and the flip–flop maintains its present state.

- **Types of Registers :**

- There are mainly two types of registers which are as follows:

1. **Register / General Purpose Register**

2. **CPU Register**

1. **Register / General Purpose Register :**

- Register receives the information, hold it temporarily and pass it on as directed by the control unit.
- There are three types of general–purpose register which are as follows:

- A. **Status Register :**

- A status register is a collection of status flag bits for a processor. It contains information about the state of the processor.
- There are three control flags : *Zero, Carry, and Overflow*.

- B. **Control Register :**

- Control register is a processor register which changes or controls the general behavior of a CPU or other digital device.
- Common tasks perform by control register includes interrupt control, switching and addressing mode, paging control etc.

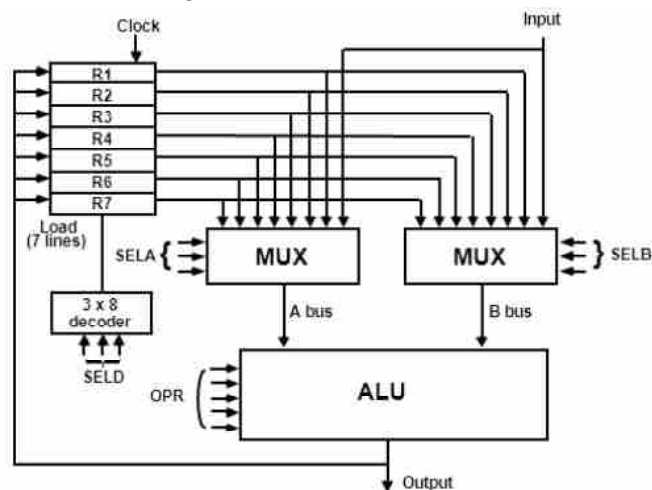
- C. **Data Input/output Register :**

- Data input/output register communicates with input/output devices.

2. **CPU Register :**

- There are various CPU registers which are as follows :

- A. Accumulator (AC) :**
 - It stores the results of the last processing steps of the ALU.
- B. Memory Address Register (MAR) :**
 - It connected to the address bus and specifies address for read/write operations.
- C. Memory Buffer Register (MBR) :**
 - It connected to data bus and hold data to write and last data read.
- D. Instruction Register (IR) :**
 - It holds last information for decoding and execution.
- E. Program Counter (PC) :**
 - It holds the address of the next instruction to be executed.
- **Register Organization :**
 - Generally, CPU has seven general register.
 - Register organization show how registers are selected and how data flow between register and ALU.
 - A decoder is used to select a particular register.
 - The output of each register is connected to two multiplexers (MUX) to form the two buses A and B.
 - The selection lines in each multiplexer select one register or the input data for the particular bus.
 - The A and B buses form the inputs to a common arithmetic logic unit (ALU).
 - The operation selected in the ALU determines the arithmetic or logic microoperation that is to be performed.
 - The result of the microoperation is available for output data and also goes into the inputs of all the registers.
 - The register that receives the information from the output bus is selected by a decoder.
 - The decoder activates one of the register load inputs, thus providing a transfer path between the data in the output bus and the inputs of the selected destination register.



Register Set with common ALU

Example :

- To perform the operation $R3 = R1 + R2$ we have to provide following binary selection variable to the select inputs.
 1. SEL A : 001 : To place the contents of R1 into bus A.
 2. SEL B : 010 : To place the contents of R2 into bus B.
 3. SEL OPR : 0010 : To perform the arithmetic addition A+B
 4. SEL REG or SEL D: 011 : To place the result available on output bus in R3
- Register and multiplexer input selection code

Binary Code	SEL A	SEL B	SEL D or SELREG
000	Input	Input	---
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

- Operation with symbol

Operation Selection Code	Operation	Symbol
0000	Transfer A	TSFA
0001	Increment A	INCA
0010	A+B	ADD
0011	A-B	SUB
0100	Decrement A	DECA
0101	A AND B	AND
0110	A OR B	OR
0111	A XOR B	XOR
1000	Complement A	COMA
1001	Shift Right A	SHRA
1010	Shift Right B	SHLA

- **Control Word :**

- The combined value of a binary selection inputs specifies the control word.
- It consists of four fields SELA, SELB, and SELD or SELREG contains three bits each and SELOPR field contains four bits thus the total bits in control word are 13-bits.

SEL A	SEL B	SELREG OR SELD	SELOPR
-------	-------	----------------	--------

1. Fetch an instruction from memory.
 2. Decode the instruction.
 3. Read the effective address from memory if the instruction has an indirect address.
 4. Execute the instruction.
- Upon the completion of step 4, the control goes back to step 1 to fetch, decode, and execute the next instruction.

• **Fetch and Decode :**

- Initially, the program counter PC is loaded with the address of the first instruction in the program.
- The sequence counter SC is cleared to 0, providing a decoded timing signal T_0 .
- After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2 , and so on.
- The microoperations for the fetch and decode phases can be specified by the following register transfer statements.

$T_0 : AR \leftarrow PC$

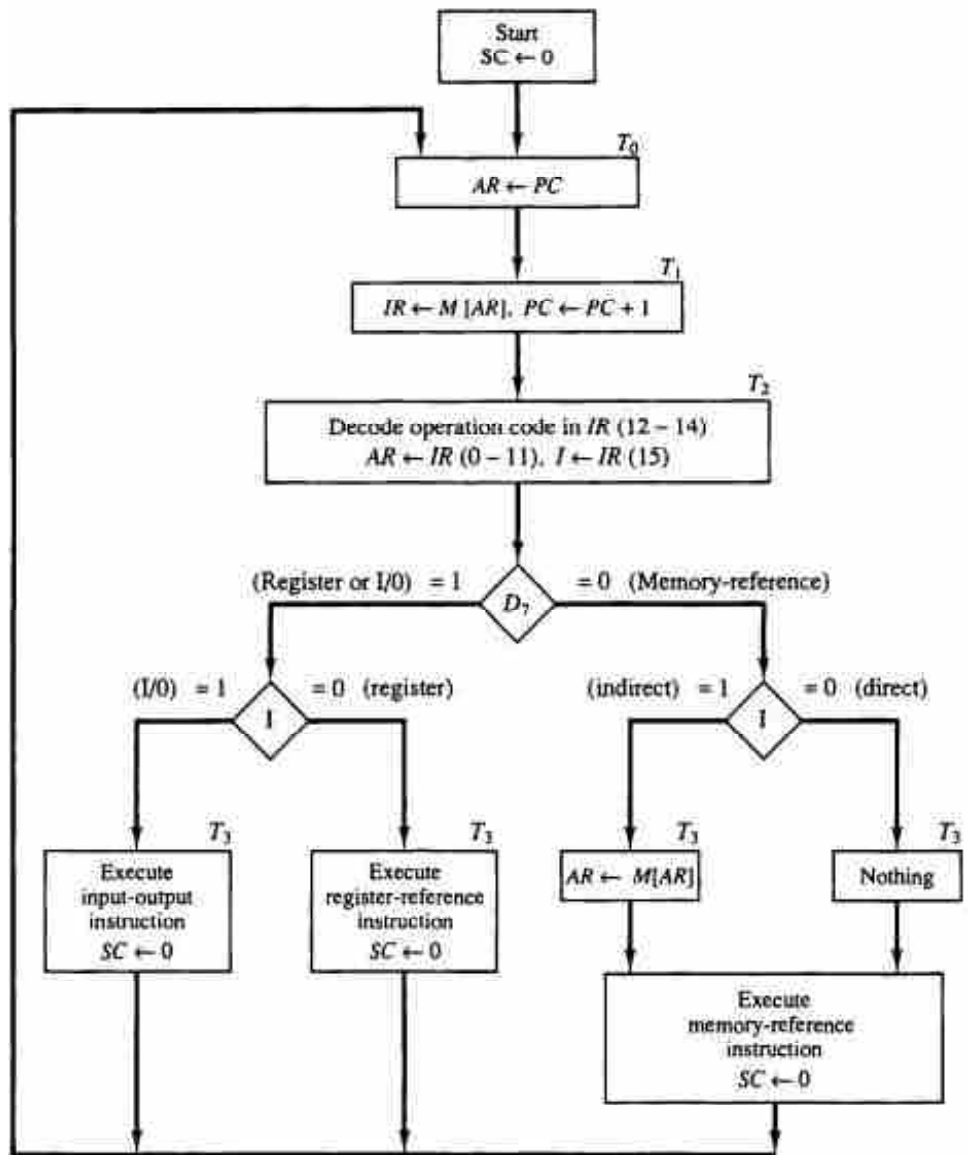
$T_1 : IR \leftarrow M[AR], PC \leftarrow PC + 1$

$T_2 : D_0, \dots, D_7 \leftarrow \text{Decode } IR(12 - 14), AR \leftarrow IR(0 - 11), I \leftarrow IR(15)$

- Since only AR is connected to the address inputs of memory, it is necessary to transfer the address from PC to AR during the clock transition associated with timing signal T_0 .
- The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing signal T_1 . At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program.
- At time T_2 , the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the instruction is transferred to AR.
- Note that SC is incremented after each clock pulse to produce the sequence T_0, T_1 , and T_2 .

• **Determine the Type of Instruction :**

- The timing signal that is active after the decoding is T_3 . During time T_3 , the control unit determines the type of instruction that was just read from memory.



Flowchart of Instruction Cycle

- The flowchart presents an initial configuration for the instruction cycle and shows how the control determines the instruction type after the decoding.
- The three possible instruction types available in the basic computer.
- Decoder output D_7 is equal to 1 if the operation code is equal to binary 111.
- If $D_7 = 1$, the instruction must be a register-reference or input-output type.
- If $D_7 = 0$, the operation code must be one of the other seven values 000 through 110, specifying a memory-reference instruction.

☐ **Check Your Progress – 3 :**

1. Explain Instruction Cycle in detail.

.....

.....

.....

15.5 Instruction Formats :

- The format of an instruction is usually depicted in a rectangular box symbolizing the bits of the instruction as they appear in memory words or in a control register.
- An instruction format defines the layout of the bits of an instruction, in terms of its basic part.
- The bits of the instruction are divided into groups called fields.
- The most common fields found in instruction formats are :
 1. An operation code field that specifies the operation to be performed.
 2. An address field that designates a memory address or a processor register.
 3. A mode field that specifies the way the operand or the effective address is determined.
- The operation code field of an instruction is a group of bits that define various processor operations, such as add, subtract, complement, and shift.
- Address fields contain either a memory address field or a register address.
- Mode fields offer a variety of ways in which an operand is chosen.
- There are mainly four types of instruction formats which are as follows :

1. Three Address Instructions :

- Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.
- The program in assembly language that evaluates $X = (A + B) * (C + D)$ is shown below, together with comments that explain the register transfer operation of each instruction.

ADD	R1, A, B	$R1 \leftarrow M[A] + M[B]$
ADD	R2, C, D	$R2 \leftarrow M[C] + M[D]$
MUL	X, R1, R2	$M[X] \leftarrow R1 * R2$

Opcode	Destination Address	Source Address	Source Address	Mode
--------	---------------------	----------------	----------------	------

- It is assumed that the computer has two processor registers, R1 and R2.
- The symbol $M[A]$ denotes the operand at memory address symbolized by A.
- 2. Two Address Instructions :**
- Two-address instructions are the most common in commercial computers.
- Here again each address field can specify either a processor register or a memory word.
- The program to evaluate $X = (A + B) * (C + D)$ is as follows :

MOV	R1, A	$R1 \leftarrow M[A]$
ADD	R1, B	$R1 \leftarrow R1 + M[B]$
MOV	R2, C	$R2 \leftarrow M[C]$
ADD	R2, D	$R2 \leftarrow R2 + M[D]$
MUL	R1, R2	$R1 \leftarrow R1 * R2$
MOV	X, R1	$M[X] \leftarrow R1$

Opcode	Destination Address	Source Address	Source Address	Mode
--------	---------------------	----------------	----------------	------

- The MOV instruction moves or transfers the operands to and from memory and processor registers.

3. One Address Instructions :

- One-address instructions use an implied accumulator (AC) register for all data manipulation.
- For multiplication and division there is a need for a second register.
- However, here we will neglect the second register and assume that the AC contains the result of all operations.
- The program to evaluate $X = (A + B) * (C + D)$ is

LOAD	A	$AC \leftarrow M[A]$
ADD	B	$AC \leftarrow AC + M[B]$
STORE	T	$M[T] \leftarrow AC$
LOAD	C	$AC \leftarrow M[C]$
ADD	D	$AC \leftarrow AC + M[D]$
MUL	T	$AC \leftarrow AC * M[T]$
STORE	X	$M[X] \leftarrow AC$

- All operations are done between the AC register and a memory operand.
- T is the address of a temporary memory location required for storing the intermediate result.

Opcode	Operand / Address of Operand	Mode
--------	------------------------------	------

4. Zero Address Instructions :

- A stack-organized computer does not use an address field for the instructions ADD and MUL.
- The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack.
- The following program shows how $X = (A + B) * (C + D)$ will be written for a stack organized computer. (TOS stands for top of stack.)

PUSH	A	TOS ← A
PUSH	B	TOS ← B
ADD		TOS ← (A + B)
PUSH	C	TOS ← C
PUSH	D	TOS ← D
ADD		TOS ← (C + D)
MUL		TOS ← (C + D) * (A + B)
POP	X	M[X] ← TOS

□ **Check Your Progress – 4 :**

1. Explain Instruction format in detail.

.....

.....

.....

.....

.....

2. Opcode stands for _____.

- (a) Operand Code
- (b) Operation Code
- (c) Other Code
- (d) Opposite Code

15.6 Addressing Modes :

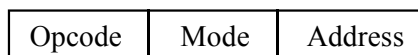
- The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in computer registers or memory words. The way the operands are chosen during program execution is dependent on the addressing mode of the instruction.
- The purpose of accommodating one or both of the following provisions :
 1. To give programming versatility to the user by providing such facilities as pointers to memory, counters for loop control, indexing of data, and program relocation.
 2. To reduce the number of bits in the addressing field of the instruction.

• **Types of Addressing Modes :**

- There are several types of addressing modes which are as follows :

1. Implied Mode :

- In this mode the operands are specified implicitly in the definition of the instruction.
- For example, the instruction "complement accumulator" is an implied-mode instruction because the operand in the accumulator register is implied in the definition of the instruction.



2. Immediate Mode :

- In this mode the operand is specified in the instruction itself.

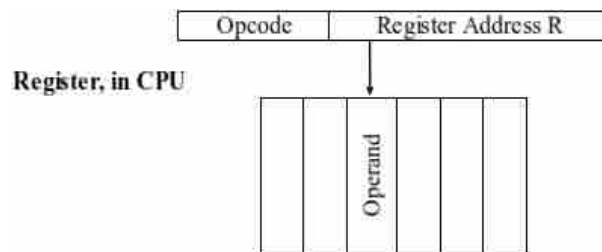
- In other words, an immediate-mode instruction has an operand field rather than an address field.
 - For example : ADD 7
- Here, ADD 7 means, ADD 7 to the content of accumulator. ADD is Opcode and 7 is operand.

Opcode	Operand
ADD	7

3. Register Mode :

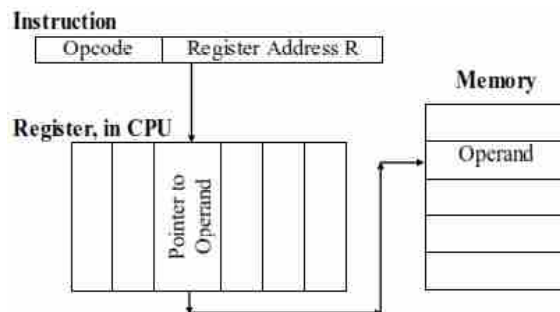
- In this mode the operands are in registers and this register is reside within the CPU.
- The instruction has the address of the register, where the operand is stored.

Instruction :



4. Register Indirect Mode :

- In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory.
- Thus, the register contains the address of operand rather than the operand itself.

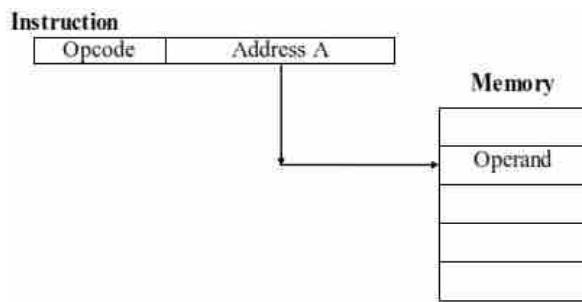


5. Auto Increment/Decrement Mode :

- This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory.
- When the address stored in the register and refers to a table of data in memory, it is necessary to increment or decrement the register after every access to the table.

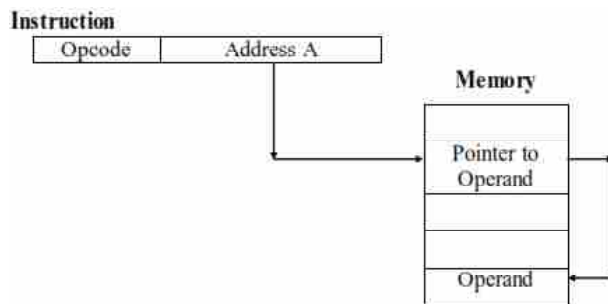
6. Direct Address Mode :

- In this mode, effective address of operand is present in instruction itself. Single memory reference to access data.
- No additional calculation to find the effective address of the operand.



7. Indirect Address Mode :

- In this mode, the address field of the instruction gives the address where the effective address is stored in memory.
- This slows down the execution, as this includes multiple memory lookups to find the operand.



8. Relative Addressing Mode :

- In this mode, the content of the program counter is added to the address part of the instruction in order to obtain the effective address.
- The address part of the instruction is usually a signed number in 2's complement representation which can be either positive or negative.
- When this number is added to the content of the program counter, the result produces an effective address whose position in memory is relative to the address of the next instruction.
- For example,
 - ✓ Assume that the program counter contains the number 825 and the address part of the instruction contains the number 24.
 - ✓ The instruction at location 825 is read from memory during the fetch phase and the program counter is then incremented by one to 826.
 - ✓ The effective address computation for the relative address mode is $826 + 24 = 850$.

9. Indexed Addressing Mode :

- In this mode the content of an index register is added to the address part of the instruction to obtain the effective address.
- The index register is a special CPU register that contains an index value.
- The address field of the instruction defines the beginning address of a data array in memory.
- Each operand in the array is stored in memory relative to the beginning address.
- The distance between the beginning address and the address of the operand is the index value stored in the index register.

- The index register can be incremented to facilitate access to consecutive operands.

10. Base Register Addressing Mode :

- In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.

The base register addressing mode is used in computers to facilitate the relocation of programs in memory.

❑ Check Your Progress – 5 :

1. In _____ mode operand is specified in the instruction itself.
(a) Implied (b) Register (c) Immediate (d) Direct Address

15.7 Let Us Sum Up :

The central processing unit (CPU) of a computer is a piece of hardware that carries out the instructions of a computer program.

It performs the basic arithmetical, logical, and input/output operations of a computer system. The CPU is like the brains of the computer – every instruction, no matter how simple, has to go through the CPU. So, let's say you press the letter 'P' on your keyboard and it appears on the screen – the CPU of your computer is what makes this possible. The CPU is sometimes also referred to as the central processor unit, or processor for short. So when you are looking at the specifications of a computer at your local electronics store, it typically refers to the CPU as the processor.

When we start to look at the various components of a CPU and how they function, remember that this is all about speed. When we use a computer, we want the instructions to be carried out very fast. As the instructions become more complicated (for example, creating a 3D animation or editing a video file), we demand more from the CPU. Thus, the technological advances we have seen in processor technology have largely been driven by the need for speed.

15.8 Suggested Answer for Check Your Progress :

- ❑ **Check Your Progress 1 :**
1 : See Section 15.1 & 15.2 2 : d
- ❑ **Check Your Progress 2 :**
1 : See Section 15.3 2 : b 3 : a
- ❑ **Check Your Progress 3 :**
See Section 15.4
- ❑ **Check Your Progress 4 :**
1 : See Section 15.5 2 : b
- ❑ **Check Your Progress 5 :**
1 : c

15.9 Glossary :

1. **Status Register :** A status register is a collection of status flag bits for a processor. It contains information about the state of the processor.

2. **Control Register :** Control register is a processor register which changes or controls the general behavior of a CPU or other digital device. Common tasks perform by control register includes interrupt control, switching and addressing mode, paging control etc.
3. **Data Input/output Register :** Data input/output register communicates with input/output devices.

15.10 Assignment :

1. Write a detailed note on Register Organization.
2. Explain all Addressing Mode in detail.
3. Full-Forms :
 - AC
 - MAR
 - MBR
 - IR
 - PC

15.11 Activities :

1. List down lower CPU and higher CPU. Difference of lower CPU performance and higher CPU performance.

15.12 Case Study :

1. Make a comparison study of Android, OxygenOS and iOS CPU.

15.13 Further Readings :

1. Digital Electronics, Anand Kumar
2. Digital Logic and Computer Design, Morris Mano

BLOCK SUMMARY :

While studying this block, the user will achieve knowledge and understanding about Input and Output devices. In computing, input/output (I/O, or informally io or IO) is the communication between an information processing system, such as a computer, and the outside world, possibly a human or another information processing system.

The block has given detail idea on the Key Board, Mouse, Display Unit, Printer and types of the printer, Scanner, OCR (Optical Character Recognition), OMR (Optical Mark Recognition), MICR (Magnetic Ink Character Recognition).

The block has given the knowledge about Input/output interface, Asynchronous data transfer, and modes of data transfer. Concepts of programmed I/O, DMA (Direct Memory Access). Input Output Interface provides a method for transferring information between internal storage and external I/O devices.

The block has given the knowledge about Memory, Flip-Flop, and CPU. As a part of Memory, we discussed about Primary Memory, Secondary Memory, Virtual Memory, and Cache Memory. Flip-Flop is two state device which offer basic memory cell for sequential logic operations. In CPU discussed about Functions of CPU, register classification and organization, instruction cycle, instruction formats, addressing modes.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Write a note on Mouse.
2. Write a note on Scanner.
3. What is Memory ? Explain Memory Hierarchy.
4. Explain Cache memory.
5. Explain Shift Register.
6. Explain Instruction Cycle.

❖ **Long Questions :**

1. Explain Keyboard in detail.
2. Explain Asynchronous Data Transfer in detail.
3. Write a detailed note on DMA.
4. What is Flip-Flop ? Explain types of Flip-Flop.
5. Explain 4-bit Register.
6. What is Counter ? Explain Asynchronous Binary Counter.
7. Write a detailed note on Instruction Formats.
8. Explain Addressing Mode.

❖ **Enrolment No. :**

1. How many hours did you need for studying the units ?

Unit No.	11	12	13	14	15
No. of Hrs.					

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Language and Style	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Illustration used (Diagram, tables etc)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Conceptual Clarity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Check your progress Quest	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
Feed back to CYP Question	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....



DR. BABASAHEB AMBEDKAR OPEN UNIVERSITY

'Jyotirmay' Parisar,
Sarkhej-Gandhinagar Highway, Chharodi, Ahmedabad-382 481.
Website : www.baou.edu.in