



Uttar Pradesh Rajarshi Tandon  
Open University

# Bachelor of Computer Application

## BCA-1.10 Computer Organization

---

### **Block-1 Electronics Components 3-92**

---

**Unit-1 : Electronics Components : Register, Capacitor  
and Inductors**

**Unit-2 : Diode**

**Unit-3 : Transistors**

**Unit-4 : Integrated Circuit**

---

### **Block-2 Building Blocks 93-182**

---

**Unit-5 : Basic Building Blocks of A Computer**

**Unit-6 : Basic Computer Organization and Design**

**Unit-7 : Instruction Cycle**

**Unit-8 : Design of Basic Computer**

**Unit-9 : Central Processing Unit**

**Unit-10 : Stack Organization**

**Unit-11 : Instruction Formats**

**Unit-12 : Addressing Modes**

---

### **Block-3 Memory & I/O 183-242**

---

**Unit-13 : Memory**

**Unit-14 : Peripheral Devices**

**Unit-15 : Introduction To 8085 Microprocessor**

---





॥ सरस्वती नः सुभगा मयस्कृत ॥

Uttar Pradesh Rajarshi Tandon  
Open University

# Bachelor of Computer Application

## BCA-1.10 Computer Organization

### BLOCK

# 1

## Electronics Components

UNIT 1	15-38
<b>Electronics Components : Register, Capacitor and Inductors</b>	
UNIT 2	39-60
<b>Diode</b>	
UNIT 3	61-78
<b>Transistors</b>	
UNIT 4	79-92
<b>Integrated Circuit</b>	

---

## Course Design Committee

---

**Dr. Ashutosh Gupta** **Chairman**  
Director (In-charge)  
School of Computer and Information Science, UPRTOU Allahabad

**Prof. R. S. Yadav** **Member**  
Department of Computer Science and Engineering  
MNNIT Allahabad

**Ms Marisha** **Member**  
Assistant Professor (Computer Science),  
School of Science, UPRTOU Allahabad

**Dr. C. K. Singh** **Member**  
Lecturer  
School of Computer and Information Science, UPRTOU Allahabad

---

## Course Preparation Committee

---

**Dr. Jitendra Pande** **Author**  
Associate Professor  
School of Computer Sciences & Information Technology  
Haldwani, Uttarakhand 263139

**Dr. Abhay Sexena** **Editor**  
Professor and Head, Department of Computer Science  
Dev Sanskriti Vishwavidyalya, Hardwar, Uttrakhand

**Dr. Ashutosh Gupta**  
Director (In-Charge)  
School of Computer & Information Sciences, UPRTOU Allahabad

**Mr. Manoj Kumar Balwant** **Coordinator**  
Assistant Professor (computer science),  
School of Sciences, UPRTOU Allahabad

---

© UPRTOU, Prayagraj. 2019

**ISBN :**

*All Rights are reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from the **Uttar Pradesh Rajarshi Tandon Open University, Prayagraj.***

Printed and Published by Dr. Arun Kumar Gupta Registrar, Uttar Pradesh Rajarshi Tandon Open University, 2018.

**Printed By :** Chandrakala Universal Pvt. Ltd. 42/7 Jawahar Lal Neharu Road, Prayagraj.

---

# COURSE INTRODUCTION

---

This course is on “Computer Organization”. This course is designed to give learners a clear understanding of components of a computer system.

Each major component is further described by decomposing into its subcomponents and describing their structure and function. The course comprises of fifteen units which are as follows:

- Unit 1** deals with the basic components of the electronics system i.e. Register, Capacitor and Inductor. It also discusses the effect of adding these components.
- Unit 2** discusses different type diodes including Rectifier Diodes, Rectifier Filters, Switching Diodes, Zener Diode and Optical Diodes.
- Unit 3** concentrates on transistors. This unit will discuss on NPN, PNP transistors, Bipolar Junction Transistors, Field Effect Transistors and Filed Effect Transistors. It also discusses the functioning of a transistor as a switch.
- Unit 4** is on Multi vibrators. It discusses the functioning of Astable Multi vibrator, Mon stable Multi vibrator and Bis table Multi vibrator. It is followed by a discussion on Counters and explanation of working of Asynchronous counters and Synchronous counters.
- Unit 5** deals with the basic components of the computer system. It also discusses how these components work together to perform the different functions a computer. It also discusses memory and its various types,
- Unit 6** focuses on Addressing Modes, Processor Registers, Basic Computer Instructions, working of Common Bus System and Control Unit.
- Unit 7** is on Instruction Cycle. In this unit various operations associated with Register Reference Instruction, Memory Reference Instructions and Input-Output Reference Instructions are discussed.
- Unit 8** concentrates on the design of a basic computer. This unit discusses Control Logic Gates, Control of Register & Memory and Control of Common Bus. Concept of design of Accumulator Logic is also covered in this unit.
- Unit 9** is on General Register Organization. It also discusses how the register communicates among each other and the ALU through bus. The operation of memory stack is also discussed in this unit.

- Unit 10** focuses on Memory Stack. In this unit operation of memory stack and evaluation of arithmetic expression using memory stack are discussed.
- Unit 11** deals with Instruction Formats. Representation of Three Address Instructions, Two Address Instructions, One Address Instructions, Zero Address Instructions and RISC Instructions are described in this unit.
- Unit 12** introduces you with some addressing modes like Implied Mode, Immediate Mode, Register Mode, Register Indirect Mode, Auto increment or Auto decrement Mode, Direct Address Mode, Indirect Address Mode, Relative addressing Mode, Indexed Addressing Mode and Base Addressing Mode.
- Unit 13** concentrates on memory. This unit discusses main memory, cache memory and virtual memory. It also describes various mapping techniques in detail.
- Unit 14** discusses different types of peripheral devices, Asynchronous Data Transfer and I/O cards of a personal computer.
- Unit 15** is the last unit of this course. This unit focuses on architecture of 8085 Microprocessor. Instruction set of 8085 Microprocessor is covered in this unit.

Each unit of this course includes SUMMARY of the unit at the end of the Unit. You will get “CHECK YOUR PROGRESS” questions. These have been designed to make you self-check your progress of study. It will be helpful for you if you solve the problems put in these boxes immediately after you go through the sections of the units and then match your answers with “ ANSWERS TO CHECK YOUR PROGRESS ”given at the end of each unit.

---

# COMPUTER ORGANIZATION

---

<b>1.0 Learning Objectives.....</b>	<b>016</b>
<b>1.1 Introduction.....</b>	<b>016</b>
<b>1.2 Resistor.....</b>	<b>017</b>
<b>1.2.1 Various types of Resistors.....</b>	<b>018</b>
<b>1.2.1.1 Wire wound Resister.....</b>	<b>018</b>
<b>1.2.1.2 Carbon Composition Resistor.....</b>	<b>019</b>
<b>1.2.1.3 Carbon Film Resistors.....</b>	<b>019</b>
<b>1.2.1.4 Variable Resistor.....</b>	<b>020</b>
<b>1.2.2 Comparison of various types of Resistors.....</b>	<b>021</b>
<b>1.2.3 Color Code.....</b>	<b>022</b>
<b>1.3 Capacitor.....</b>	<b>024</b>
<b>1.3.1 Various types of Capacitors.....</b>	<b>025</b>
<b>1.3.1.1 Paper Capacitor.....</b>	<b>025</b>
<b>1.3.1.2 Electrolyte Capacitor.....</b>	<b>026</b>
<b>1.3.1.3 Mica Capacitor.....</b>	<b>026</b>
<b>1.3.1.4 Ceramic Capacitor.....</b>	<b>027</b>
<b>1.3.2 Capacitor Values.....</b>	<b>029</b>
<b>1.4 Inductor.....</b>	<b>030</b>
<b>4.1 Various Types of Inductors.....</b>	<b>031</b>
<b>1.4.1 Various types of Inductors.....</b>	<b>031</b>
<b>1.4.1.1 Air Core Inductor.....</b>	<b>031</b>
<b>1.4.1.2 Ferrite Core inductor.....</b>	<b>031</b>
<b>1.4.1.3 Iron Core Inductor.....</b>	<b>031</b>
<b>1.4.1.4 Coupled Inductor 3.....</b>	<b>031</b>
<b>1.5 Effect of Addition.....</b>	<b>032</b>
<b>1.5.1 Adding resistors in series and parallel.....</b>	<b>032</b>
<b>1.5.2 Adding capacitors in series and parallel.....</b>	<b>033</b>
<b>1.5.3 Adding inductors in series and parallel.....</b>	<b>035</b>
<b>1.6 Summary.....</b>	<b>036</b>
<b>1.7 Answers to Check Your Progress.....</b>	<b>037</b>
<b>1.8 Terminal Questions.....</b>	<b>038</b>

<b>2.0</b>	<b>Learning Objectives.....</b>	<b>039</b>
<b>2.1</b>	<b>Introduction.....</b>	<b>040</b>
<b>2.2</b>	<b>What is a Diode?.....</b>	<b>040</b>
<b>2.3</b>	<b>How Diodes work? .....</b>	<b>042</b>
<b>2.4</b>	<b>Types of diodes.....</b>	<b>043</b>
<b>2.4.1</b>	<b>Rectifier diodes.....</b>	<b>043</b>
<b>2.4.1.1</b>	<b>Half-wave rectifier.....</b>	<b>043</b>
<b>2.4.1.2</b>	<b>Full-wave rectifier.....</b>	<b>044</b>
<b>2.4.2</b>	<b>Switching diodes.....</b>	<b>046</b>
<b>2.4.3.1</b>	<b>Clipping.....</b>	<b>046</b>
<b>2.3.3.2</b>	<b>Clamping.....</b>	<b>051</b>
<b>2.4.3</b>	<b>Zener Diodes.....</b>	<b>056</b>
<b>2.4.3.1</b>	<b>V-I Characteristics of Zener Diode.....</b>	<b>037</b>
<b>2.4.3.2</b>	<b>Application of Zener Diode.....</b>	<b>058</b>
<b>2.4.4</b>	<b>Optical Diodes.....</b>	<b>058</b>
<b>2.4.4.1</b>	<b>Light Emitting Diode.....</b>	<b>058</b>
<b>2.4.4.2</b>	<b>Photodiode.....</b>	<b>058</b>
<b>2.4.4.3</b>	<b>Optocoupler.....</b>	<b>058</b>
<b>2.5</b>	<b>Summary.....</b>	<b>059</b>
<b>2.6</b>	<b>Answers to Check Your Progress.....</b>	<b>060</b>
<b>2.7</b>	<b>Terminal Questions.....</b>	<b>060</b>
<b>3.0</b>	<b>Learning Objectives.....</b>	<b>061</b>
<b>3.1</b>	<b>Introduction.....</b>	<b>062</b>
<b>3.2</b>	<b>Transistors.....</b>	<b>062</b>
<b>3.2.1</b>	<b>NPN Transistor.....</b>	<b>063</b>
<b>3.2.2</b>	<b>PNP Transistor.....</b>	<b>064</b>
<b>3.3</b>	<b>Bipolar Junction Transistors (BJT) .....</b>	<b>065</b>
<b>3.3.1</b>	<b>NPN Bipolar Junction Transistor.....</b>	<b>066</b>
<b>3.3.2</b>	<b>PNP Bipolar Junction Transistor.....</b>	<b>067</b>
<b>3.3.3</b>	<b>Regions of Operation.....</b>	<b>068</b>
<b>3.3.4</b>	<b>Configurations.....</b>	<b>069</b>
<b>3.3.4.1</b>	<b>The Common Base Configuration.....</b>	<b>069</b>



3.3.4.2 The Common Emitter Configuration.....	070
3.3.4.3 The Common Collector Configuration.....	070
3.4 Field Effect Transistor (FET).....	071
3.4.1 Junction FET (JFET).....	072
3.4.2 Metal-oxide- semiconductor FET (MOSFET).....	073
3.5 Analog and digital electronics.....	074
3.6 Transistor as a switch.....	074
3.7 Summary.....	076
3.8 Answers to Check Your progress.....	077
4.0 Learning Objectives.....	079
4.1 Introduction.....	080
4.2 Integrated Circuit.....	080
4.3 Multivibrators.....	081
4.3.1 Astable Multivibrator.....	081
4.3.2 Monostable Multivibrator.....	084
4.3.3 Bistable Multivibrator.....	085
4.4 Counters.....	086
4.4.1 Asynchronous counters(or ripple counters).....	087
4.4.2 Synchronous Counters.....	088
4.4.2. 1 Synchronous Decade Counter.....	088
4.5 Summary.....	089
4.6 Answers to check your progress.....	090
4.7 Terminal Questions.....	091
5.0 Learning Objectives.....	095
5.1 Introduction.....	096
5.2 Basic Building Blocks of a Computer.....	096
5.3 Input Unit.....	097
5.4 OUTPUT Devices.....	100
5.5 Central Processing Unit.....	102
5.5.1 Arithmetic Logic Unit.....	103

5.5.2 Control Unit.....	104
5.5.3 Register Set.....	104
5.6 Memory.....	105
5.6.1 Random Access Memory (RAM) .....	106
5.6.2 Read Only Memory (ROM) .....	106
5.6.3 Units of Storage.....	107
5.7 Secondary Storage Devices.....	107
5.8 Summary.....	110
5.9 Answers to Check Your Progress.....	110
5.10 Terminal Questions.....	111
6.0 Learning Objectives.....	113
6.1 Introduction.....	114
6.2 The Basic Computer.....	114
6.2.1 Addressing Modes.....	115
6.2.3 Processor Register.....	116
6.3 Basic Computer Instructions.....	118
6.4 Common Bus System.....	121
6.5 Control Unit.....	123
6.6 Summary.....	125
6.7 Answers to Check Your Progress.....	125
6.8 Terminal Questions.....	126
7.0 Learning Objectives.....	127
7.1 Introduction.....	127
7.2 Instruction Cycles and Subcycles.....	128
7.2.1 Instruction Fetch from Memory.....	128
7.2.2 Instruction Decode.....	129
7.2.3 Instruction Execution.....	129
7.3 Register Reference Instructions.....	131
7.4 Memory Reference Instructions.....	132
7.5 Input-Output Reference Instructions.....	138

<b>7.6 Summary.....</b>	<b>140</b>
<b>7.7 Answers to Check Your Progress.....</b>	<b>140</b>
<b>7.8 Terminal Questions.....</b>	<b>141</b>
<b>8.0 Learning Objectives.....</b>	<b>143</b>
<b>8.1 Introduction.....</b>	<b>143</b>
<b>8.2 Basic Computer.....</b>	<b>144</b>
<b>8.2.1 Control Logic Gates.....</b>	<b>146</b>
<b>8.2.2 Control of Registers and Memory.....</b>	<b>147</b>
<b>8.2.3 Control of Single Flip-flops.....</b>	<b>149</b>
<b>8.2.4 Control of Common Bus.....</b>	<b>150</b>
<b>8.3 Design of Accumulator Logic.....</b>	<b>152</b>
<b>8.3.1 Control of AC Register.....</b>	<b>153</b>
<b>8.3.2 Adder and Logic Circuit.....</b>	<b>154</b>
<b>8.4 Summary.....</b>	<b>154</b>
<b>8.5 Answers to Check Your Progress.....</b>	<b>155</b>
<b>8.6 Terminal Questions.....</b>	<b>155</b>
<b>9.0 Earning Objectives.....</b>	<b>157</b>
<b>9.1 Introduction.....</b>	<b>157</b>
<b>9.2 Eeneral Register Organization.....</b>	<b>158</b>
<b>9.3 Summary.....</b>	<b>161</b>
<b>9.4 Answers to Check Your Progress.....</b>	<b>162</b>
<b>9.5 Terminal Questions.....</b>	<b>162</b>
<b>10.0 Learning Objectives.....</b>	<b>163</b>
<b>10.1 Introduction.....</b>	<b>163</b>
<b>10.2 Memory Stack.....</b>	<b>165</b>
<b>10.3 Evaluation of Arithmetic Expressions.....</b>	<b>166</b>
<b>10.4 Summary.....</b>	<b>168</b>
<b>10.5 Answers to Check Your Progress.....</b>	<b>168</b>

10.6 Terminal Questions.....	169
11.0 Learning Objectives.....	171
11.1 Introduction.....	171
11.2 Three Address Instructions.....	172
11.3 Two Address Instructions.....	173
11.4 One Address Instructions.....	173
11.5 Zero Address Instructions.....	174
11.6 RISC Instructions.....	175
11.7 Summary.....	175
11.8 Answers to Check Your progress.....	176
11.9 Terminal Questions.....	176
12.0 Learning Objectives.....	177
12.1 Introduction.....	177
12.2 Addressing Modes.....	178
12.2.1 Implied Mode.....	178
12.2.2 Immediate Mode.....	178
12.2.3 Register Mode.....	179
12.2.4 Register Indirect Mode.....	179
12.2.5 Auto-increment or Auto-decrement Mode.....	179
12.2.6 Direct Address Mode.....	179
12.2.7 Indirect Address Mode.....	180
12.2.8 Relative Address Mode.....	180
12.2.9 Indexed Addressing Mode.....	180
12.2.10 Base Register Addressing Mode.....	181
12.3 Summary.....	181
12.4 Answers to Check Your Progress.....	182
12.5 Terminal Questions.....	182
13.0 Learning Objectives.....	185
13.1 Introduction.....	186
13.2 Main Memory.....	187

13.2.1 RAM (Random access memory).....	187
13.2.2 ROM (Read Only Memory).....	188
13.2.3 Difference between RAM and ROM.....	189
13.2.4 Memory Address Table.....	190
13.3 Cache Memory.....	191
13.3.1 Associative Mapping.....	192
13.3.2 Direct Mapping.....	193
13.3.3 Set-Associative Mapping.....	194
13.3.4 Writing into Cache.....	195
13.3.5 Page Replacement Algorithm.....	196
13.4 Virtual memory.....	197
13.4.1 Address Space and Memory Space.....	197
13.4.2 Address mapping Using Pages.....	199
13.4.3 Associative Memory Page Table.....	200
13.5 Summary.....	202
13.6 Answers to Check Your Progress.....	202
13.7 Terminal Questions.....	203
14.0 Learning Objectives.....	205
14.1 Introduction.....	206
14.2 Peripheral Devices.....	210
14.2.1 Keyboard.....	210
14.2.2 Magnetic Tape.....	210
14.2.3 Display System.....	211
14.3 Input/Output Interface.....	211
14.4 Asynchronous Data Transfer.....	213
14.5 I/O cards in personal computers.....	214
14.5.1 Graphic card.....	214
14.5.2 Sound Card.....	215
14.5.3 Network Interface Card (NIC) .....	216

<b>14.6 Summary.....</b>	<b>217</b>
<b>14.7 Answers to Check Your Progress.....</b>	<b>217</b>
<b>14.8 Terminal Questions.....</b>	<b>217</b>
<b>15.0 Learning Objectives.....</b>	<b>219</b>
<b>15.1 Introduction.....</b>	<b>219</b>
<b>15.2 Architecture of Microprocessor.....</b>	<b>220</b>
<b>15.2.1 8085 Microprocessor.....</b>	<b>221</b>
<b>15.2.2 The Internal Architecture of 8085.....</b>	<b>225</b>
<b>15.3 Instruction Set of 8085 Microprocessor.....</b>	<b>228</b>
<b>15.3.1 Data Transfer Group.....</b>	<b>228</b>
<b>15.3.2 Arithmetic Group.....</b>	<b>230</b>
<b>15.3.3 Logical Group.....</b>	<b>232</b>
<b>15.3.4 Branch Control Group.....</b>	<b>234</b>
<b>15.3.5 I/O and Machine Control Group.....</b>	<b>235</b>
<b>15.4 Summary.....</b>	<b>236</b>
<b>15.5 Answers to Check Your Progress.....</b>	<b>237</b>
<b>15.6 Terminal Questions.....</b>	<b>237</b>
<b>References.....</b>	<b>238-242</b>

# UNIT-1

---

## ELECTRONICSCOMPONENTS REGISTER, CAPACITOR AND INDUCTORS

---

### Structure

- 1.0 Learning Objectives
- 1.1 Introduction
- 1.2 Resistor
  - 1.2.1 Various types of Resistors
    - 1.2.1.1 Wire wound Resister
    - 1.2.1.2 Carbon Composition Resistor
    - 1.2.1.3 Carbon Film Resistors
    - 1.2.1.4 Variable Resistor
  - 1.2.2 Comparison of various types of Resistors
  - 1.2.3 Color Code
- 1.3 Capacitor
  - 1.3.1 Various types of Capacitors
    - 1.3.1.1 Paper Capacitor
    - 1.3.1.2 Electrolyte Capacitor
    - 1.3.1.3 Mica Capacitor
    - 1.3.1.4 Ceramic Capacitor
  - 1.3.2 Capacitor Values
- 1.4 Inductor
  - 1.4.1 Various types of Inductors
    - 1.4.1.1 Air Core Inductor
    - 1.4.1.2 Ferrite Core inductor
    - 1.4.1.3 Iron Core Inductor
    - 1.4.1.4 Coupled Inductor 3
- 1.5 Effect of Addition
  - 1.5.1 Adding resistors in series and parallel

### 1.5.2 Adding capacitors in series and parallel

### 1.5.3 Adding inductors in series and parallel

## 1.6 Summary

## 1.7 Answers to Check Your Progress

## 1.8 Terminal Questions

---

# 1.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

Define register, capacitor and an inductor. Understand the principal on which the a register, capacitor and an inductor is based upon. Compare various types of registers and know its applications.

1. Compare various types of capacitors and know its applications.
2. Compare various types of inductors and know its applications.

---

# 1.1 INTRODUCTION

---

In this unit, we are going to discuss the basic electronics components like register, capacitor and inductors. These electronic components are the basic building block of any electronic equipment. The design of the equipment is finalized by the designer based on the requirement and then electronic circuit is designed using these electronic components. These components are connected to each other through conducting path to form the digital circuit. These components are present inside the devices you are using in your daily life like: Television, Mobile, MP3 Players, digital watches, washing machines, etc. The electronics components are broadly divided into two categories:

- a. *Active Components:* Active components are those components which requires an external power source for operation. Some of the examples of active components are transistors, ICs, LED, etc.
- b. *Passive Components:* Passive components are those components which do not require an external power source for operation. Some examples of passive components are registers, capacitors, inductors, etc.

Now let us discuss the above components viz. Register, Capacitor and Inductor in detail.



---

## 1.2 RESISTORS

---

The first and most common electronic component is the resistor. Resistor has a very important property: it resists the flow of charge, thus allowing us to control the current. Resistor obeys Ohm's law, which states that:

The potential difference (voltage) across an ideal conductor is proportional to the current through it. Ohm's Law is given by:

$$V = IR \quad (1.1)$$

Where,

V is the potential difference between two points and is measured in volts (V).

I is the current and is measured in ampere (A).

R is the constant of proportionality is called the "resistance", and is measured in ohms ( $\Omega$ ).

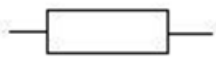
Thus, if the voltage and the current is known, the resistance is calculated by the formula:

$$R = \frac{V}{I} \quad (1.2)$$

From eq. (1.1) we can conclude that the voltage is directly proportional to current i.e. if we increase the voltage, the current will increase.

From eq. (1.2) we can conclude that resistance is inversely proportional to current i.e. if we increase the resistance, the current will reduce.

represents the symbols used for resistors in a circuit diagram.



Modern(European )  
representation



Old(American) representation

**Figure 1: Symbol for Resistor**

*Example 1:* Find the current in an electrical circuit that has resistance of 100 Ohms and voltage supply of 10 Volts.

Solution:  $V = 10V$   $R = 100\Omega$

$I = V / R = 10V / 100\Omega = 0.1A = 100mA$

*Example 2:* Find the resistance of an electrical circuit that has voltage supply of 10 Volts and current of 5mA.

Solution:  $V = 10V$        $I = 5mA = 0.005A$

$R = V / I = 10V / 0.005A = 2000\Omega = 2k\Omega$

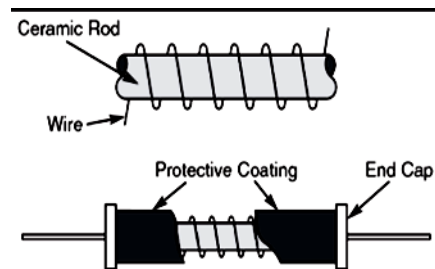
## 1.2.1 Various types of Resistors

Based on the type of material used for insulation, its ability to handle different amount of electrical power, response to the variation in the voltage, and ability to mechanically change the resistance, there are many variants of a resistor. Some important types of resistors are discussed below:

### 1.2.1.1 Wirewound Resistor

This type of resistor is made by wrapping a resistive wire, which is usually made up of nickel-chromium alloy, around a non-conductive rod, which is mostly made up of ceramic due to its desired heat properties since the current carrying conducting wires produces heat (

Figure 2)



**Figure 2: Wirewound Resistor**

End caps with leads attached were then placed over the ends of the rod making contact with the resistive wire, usually a nickel-chromium alloy. Wirewound resistor's response to the increase in temperature is flat but they respond sharply with the change in frequency.

#### ***Advantages:***

1. Very reliable & stable
2. Low power consumption
3. Very less noise.
4. Can operate in moist conditions also.

#### ***Disadvantages:***

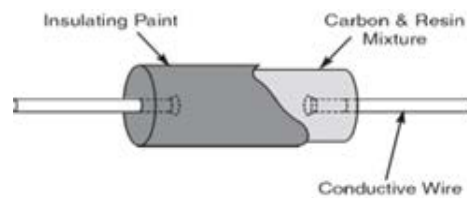
1. Does not operate well at high frequency.

2. Large size.

### 1.2.1.2 Carbon Composition Resistor

This type of resistors is formed by mixing finely grinding carbon with resin. The unit is heated in an oven after connecting the end of the unit with conductive leads. The body of the resistor is then painted by and insulating paints

Figure 3).



**Figure 3: Carbon Composition Resistor**

#### *Advantages*

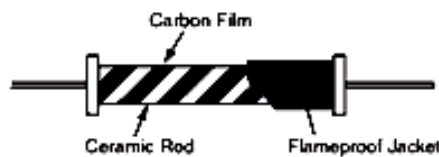
1. It can withstand high energy pulses.
2. Cheaper than wirewound resistors.
3. Higher resistance than wirewound resistors.

#### *Disadvantages*

1. Cannot withstand overload. Overload can decrease the resistance permanently.
2. Noise properties are not good as the resistor is made up of mixture of different materials.

### 1.2.1.3 Carbon Film Resistors

Carbon film resistors are made by depositing a very thin layer of carbon on a ceramic rod, which is then protected by flameproof jacket (Figure 4).



**Figure 4: Carbon Film resistor**

There are two variants of carbon film resistors:

- i. **Metal Oxide Resistors:** In this type of carbon film resistor, the film is made up of tin chloride. They can withstand very high temperatures.
- ii. **Metal Film Resistors:** In this type of carbon film resistor, a film of metal (like nickel alloy) is deposited over a ceramic rod. They can withstand high pressure and frequency.

**Advantages**

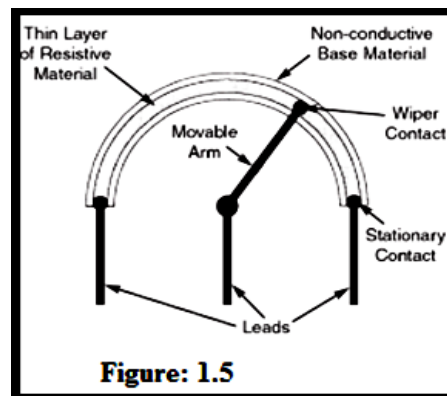
1. They produce less electrical noise.
2. Values are constant even at high frequencies

**Disadvantage**

1. Very costly

**1.2.1.4 Variable Resistor**

Variable resistors or potentiometers, as the name suggests, the value of resistance can be varied during the operation in the variable resistor. This type of resistor is made by depositing a resistive material on a non-conducting base and both the ends of the resistive material are connected by stationary contacts. A movable contact, also known as wiper, is attached to the resistor such that it can move along the resistive material and tap off the desired resistance (Figure 5).



**Figure 5: Variable Resistor**

**Advantages**

1. They are very stable.
2. They are very precise.
3. They have very low power ratings.
4. They have the ability to vary the resistance.

**Disadvantage**

1. Variable resistors, due to the rotary action, produce friction and leads to the generation of noise.

## 1.2.2 Comparison of various types of Resistors

The comparison of the different type of resistors on the basis of various characteristics likes reliability, cost, performance etc. is shown below in the Table 1. But before that, we will introduce some of the terminologies which are used throughout this chapter.

**i. Temperature coefficient :** It represents the effect of varying the temperature of a resistor on the resistance of a material. Typically the units are either resistance per temperature or 1/temperature depending on which equation is used for the calculations. For example, in copper the temperature coefficient of resistance is about 0.0039 per change in degrees Celsius. A positive temperature coefficient of resistance means that the resistance of the material will increase as temperature increases.

Let a conductor having a resistance of  $R_0$  at  $0^\circ\text{C}$  and  $R_t$  at  $t^\circ\text{C}$  respectively. From the equation of resistance variation with temperature we get

$$\frac{R_t}{R_0} = \frac{t_0 + t}{t_0 + 0} \quad (1.3)$$

$$\Rightarrow R_t = R_0 + \frac{R_0 \cdot t}{t_0} \quad (1.4)$$

or,

$$R_t - R_0 = \Delta R = \frac{1}{t_0} R_0 \cdot t = \alpha_0 R_0 \cdot t, \text{ where } \alpha_0 = \frac{1}{t_0} \quad (1.5)$$

This  $\alpha_0$  is called **temperature coefficient of resistance** of that substance at  $0^\circ\text{C}$ . Based on the above equations, the formal definition of resistance temperature coefficient can be given as:

*“Rise in temperature per unit initial resistance, when temperature is raised by one degree Celsius is called the resistance temperature coefficient.”*

**ii. Tolerance :** It represents the accuracy in the desired output. For a given resistor of 1000 ohms, if the tolerance is claimed to be 3%, then it means the output will vary from 997 ohms to 1003 ohms.

**iii. Stability:** It represents the ability to produce the output in the desired range even if the temperature and the humidity condition varies considerably.

**iv. Noise :** It is random, undesirable electrical energy that enters the communications system via the communicating medium and interferes with the transmitted message. However, some noise is also produced in the receiver. Noise negatively effect the output quality of the system.

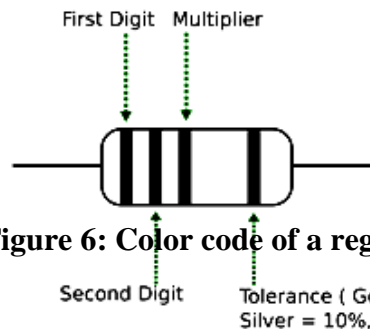
v. **Reliability** : It is a measure of the quality of a system which represents the performance of a system over a period of time. If any resistance changes its value very less after the maximum utilization then this type of resistor is called a reliable resistor.

**Table 1: Comparison of various resistors**

Types of Resistor	Stability	High Frequency Response	Reliability	Noise	Cost	Voltage Coefficient	Value	Temperature coefficient
1. Carbon Film Type	High	Good	Very Good	Low	Expensive	Low	1-20 Ohm	Low
2. Carbon Composition Type	Less Stable	Satisfactory	Good	High	Less Expensive	More	10-20 mega ohms	Large
3. Wirewound Type	Very High	Poor	Very Good	Negligible	More Expensive	Negligible	0.1-4 mega ohms	Medium

### 1.2.3 Color Code

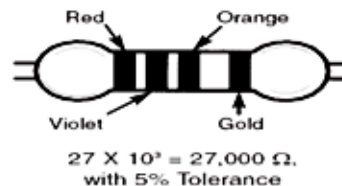
The resistance of a resistor is measured in Ohm( $\Omega$ ). Different color rings around the body of the resistor are used to represent the resistance of a resistor (Figure 6).



**Figure 6: Color code of a resistor**

1. Image adopted from: <https://commons.wikimedia.org/wiki/File:Resistor-Codes.svg>

2. Image adopted from: [https://commons.wikimedia.org/wiki/File:Types\\_of\\_capacitor.svg](https://commons.wikimedia.org/wiki/File:Types_of_capacitor.svg)



The first and the second color bands represent the first and the second digit of the resistor's value respectively. The third color band represents the multiplier and the fourth color band represents the tolerance. The color code chart is presented in Table 2.

**Table 2: Color code**

Color	1 <sup>st</sup> Figure	2 <sup>nd</sup> Figure	Multiplier	Tolerance
Black	0	0	1.0	
Brown	1	1	10	
Red	2	2	100	
Orange	3	3	1000	
Yellow	4	4	10000	
Green	5	5	100000	
Blue	6	6	1000000	
Violet	7	7	10000000	
Gray	8	8	100000000	
White	9	9	1000000000	
Gold			0.1	5%
Silver			0.01	10%
No color				20%

Now let us try to decode the resistance of resistor by observing its color bands.

**Example 3:** Suppose the pattern is as follows:

1 <sup>st</sup> Color	2 <sup>nd</sup> Color	3 <sup>rd</sup> Color	4 <sup>th</sup> Color
band	band	band	band
Red	Red	Green	Gold

By looking above in table 2, we can easily decode the values of these color bands as

$22 \times 100,000$  Ohm with 5% tolerance.

**Example 4:** Suppose the pattern is as follows:

1 <sup>st</sup> Color	2 <sup>nd</sup> Color	3 <sup>rd</sup> Color	4 <sup>th</sup> Color
band	band	band	band
Orange	Orange	Yellow	Silver

Again by looking at the table 2, the resistance of the resistor is decoded as:

$33 \times 10,000$  Ohm with 10% tolerance.

**Example 5:** Suppose the pattern is as follows:

1 <sup>st</sup> Color	2 <sup>nd</sup> Color	3 <sup>rd</sup> Color	4 <sup>th</sup> Color
band	band	band	band

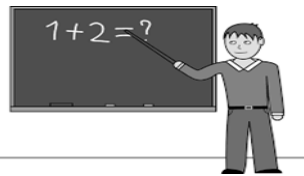
Yellow

Violet

Silver

No color

By using table 2, the color band is translated to  $47 \times 0.01$  or 0.47 ohms and no fourth band indicates a 20% tolerance.



## Check Your Progress

1. Resistor follows \_\_\_\_ law.
2. The potential difference (voltage) across an ideal conductor is proportional to the \_\_\_\_\_ through it.
3. Wire wound resistor's response to the increase in temperature is flat but they respond sharply with the change in \_\_\_\_\_.
4. \_\_\_\_\_ cannot withstand overload. Overload can decrease the resistance permanently.
5. In \_\_\_\_\_ the film is made up of tin chloride. They can withstand very high temperatures.
6. In \_\_\_\_\_, a film of metal (like nickel alloy) is deposited over a ceramic rod. They can withstand high pressure and frequency.
7. Variable resistor is also known as \_\_\_\_\_.
8. The resistance of a resistor is measured in \_\_\_\_\_.

---

## 1.3 CAPACITORS

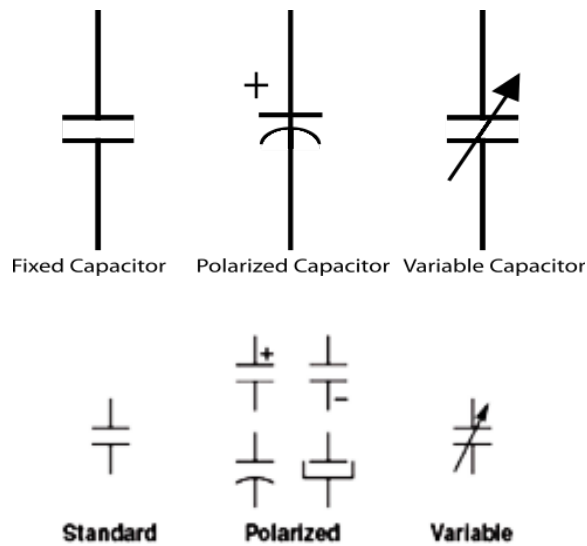
---

There are situations when it is required to store electrical charge in a circuit for a short duration of time. For this purpose, a passive electronic component, known as capacitor is used. Capacitors are based on the principle of electrostatic induction. Electrostatic Induction is a method to create static electricity on a material by bringing an electrically charged object near it. This causes the electrical charges to be redistributed in the material, resulting in one side having an excess of either positive (+) or negative (-) charges.

A capacitor can be either polarized or non-polarized, which means either that they have to be connected the right way round (like the LED) or you can connect them anyway you like (like the resistor). If a capacitor is polarized it has a slightly different symbol on the circuit diagram and it is marked on the case with a plus-sign (+) or a minus sign (-). shows the schematic symbols used to represent capacitors. The + symbol indicates that the capacitor is polarized and the lead marked with the + sign must always have a *higher* voltage than the other lead. The curved plate, plate with sides, and minus sign also indicate the capacitor is polarized and



these leads must always be at a *lower* voltage than the other lead. The arrow crossing through the capacitor indicates capacitance is variable.



**Figure 7: Symbols for Capacitors**

Now let us discuss how to find out the Capacity of charge storage inside a capacitor.

The capacity to store the charge inside the capacitor is given by the formula:

$$C = \frac{Q}{V} \quad (1.6)$$

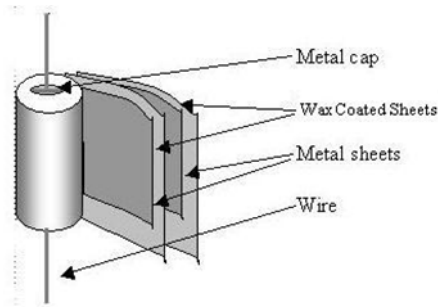
It can be calculated by dividing the charge (Q) present on the plates of conductor by the potential (V) of the conductor. The common word used in our daily life for capacitors is Condensers which we use in most of our electric appliances. Their main purpose is to provide the initial start to the device in which it is installed.

### 1.3.1 Various types of Capacitors

Capacitors are of many types. Some of the popular ones are:

#### 1.3.1.1 Paper Capacitor

The flat thin strips of metal foil conductors are separated by dielectric material, which in this case is a waxed paper. They are sealed with wax to prevent moisture. The whole unit is covered with outer covering which is made up of different materials like cardboard, plastic, etc. depending of the temperature range in which they are designed to operate.

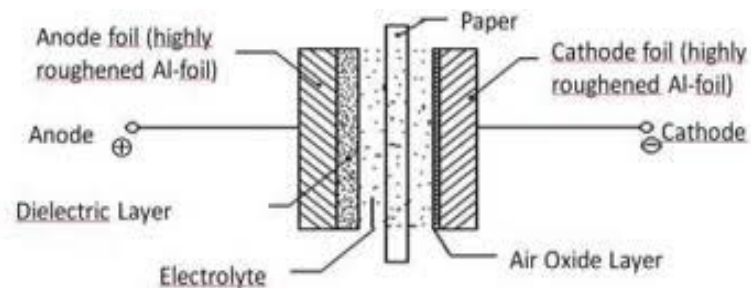


**Figure 8 : Paper Capacitor**

Paper capacitors usually operate in the range between 300 picofarads to about 4 microfarads and the working voltage of a paper capacitor rarely exceeds 600 volts.

### 1.3.1.2 Electrolyte Capacitor

Electrolyte capacitor has two plates, one of which is coated with a very thin layer of an oxide using electrolysis and the other plate is replaced by an electrolyte. They are small in size but have large capacitance.



**Figure 9: Electrolytic Capacitors**

Certain degree of caution is required to use electrolyte capacitors as they are polarized. In case the polarity is not matched, the capacitor will be permanently damaged.

### 1.3.1.3 Mica Capacitor

In case of Mica capacitors, Mica is used as dielectric material. They come in two variants viz. clamped mica capacitors and silver mica capacitors. It is made by sandwiching mica sheets coated with metal on both sides. The unit is then encased in epoxy in order to protect it from the environment.

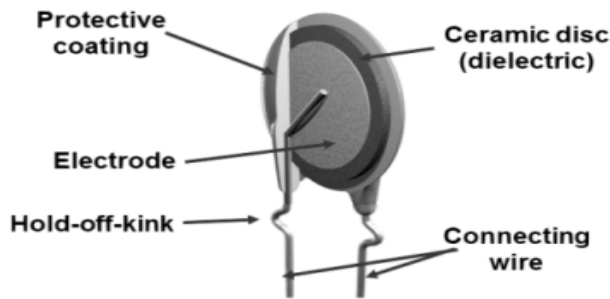


**Figure 10: Mica Capacitors**

Mica capacitors are ideal choice for working at high frequencies.

### 1.3.1.4 Ceramic Capacitor

In case of ceramic capacitors, ceramic is used as dielectric material which helps in improving capacitor power dissipation. They are made by coating two sides of a small porcelain or ceramic disc with silver and are then stacked together to make a capacitor.



**Figure 81: Ceramic or Disk Capacitor**

The amount of charge a capacitor can hold (capacitance) is measured in Farads. There are three factors that determine the capacitance that exist between two conductive plates:

- i. The capacitance is directly proportional to the plate area i.e. larger is the area of the plate, higher is the capacitance. Or we can say that:

Capacitance  $\propto$  Area of the plate

**C  $\propto$  A**

- ii The capacitance is inversely proportional to the distance between the two plates i.e. higher is the distance between the two plates, lower is the capacitance. Or we can say that

$$\text{Capacitance} \propto \frac{1}{\text{distance between two plates}}$$

$$C \propto \frac{1}{d}$$

- iii The capacitance is directly proportional to the value of dielectric constant i.e. larger value of dielectric constant leads to higher value of capacitance.

$$C \propto K$$

From the above factors, the formula for capacitance in Farads becomes:

$$C = 0.244K \frac{A(N - 1)}{d} \text{picofarads} \quad (1.7)$$

Where,

C = Capacitance in Picofarads (Farad x 10<sup>-12</sup>)

A = Area of one Plate in square inches

N = Number of Plates

d = Distance between plates in inches

K = Dielectric Constant, is the ratio by which the capacitance is increased when another dielectric replaces a vacuum between two plates. Table 2 shows the Dielectric Constant of various materials.

**Table 3: Value of K for different materials**

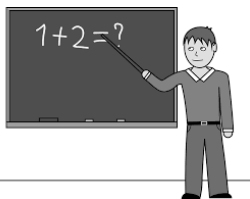
<b>Material</b>	<b>Value of Dielectric Constant</b>
Air, at normal pressure	1
Alcohol, ethyl(grain)	25
Beeswax	1.86
Caster oil	4.67
Glass flint density 4.5	10
Glycerin	56
Mica	7.5
Paper, manila	1.5
Paraffin wax	2.25
Porcelain	4.4
Quartz	2
Water, distilled	81

### 1.3.2 Capacitor Values

Few years back, the capacitors also had the color coding rings and dots similar to resistors but now the details like value, tolerance, temperature characteristics etc. are printed on the surface.

**Table 4: Standard values and different methods used for marking in capacitors**

Voltage 1	Code 2	Cap. value 3	Typical Markings 5	Tolerance(%) 6	Markings 7	
4	0G	100pF	100pF	101	±5%	J
5.5	0L	.001µF	.001	102	±10%	K
6.3	0J	.015 Mf	.015	152	±20%	M
10	1A	.002 µF	.002	202	-10% +30%	Q
16	1C	.0022 µF	.0022	222	-10% +50%	T
25	1E	.003 µF	.003	302	-20% +80%	Z
35	1V	.033 µF	.033	333	SPECIAL	A
50	1H	.047 µF	.047	473		
63	1J	.05 µF	.05	R05	<b>Temperature Markings</b>	
80	1K	.068 µF	.068	R068		
100	2A	.1 µF	.1	104	NP0(<10ppm/ °C)	
110	2Q	.15 µF	.15	154	NP100(<100ppm/ °C)	
125	2B	.2 µF	.2	204	NP220(<220ppm/ °C)	
160	2C	2.2 µF	2.2	2R2	NP820(<820ppm/ °C)	
180	2Z	22 µF	22	22	Y5F	
200	2D	100 µF	100	100	Y5T	
220	2P	220 µF	220	220	X5F	
250	2E	470 µF	470	470	Y5V	
315	2F	1000 µF	1000	1000	Z5U	



### Check Your Progress

- \_\_\_\_\_ is a measure of the quality of a system which represents the performance of a system over a period of time. Capacitors are based on the principle of\_\_\_\_\_.
- In case the \_\_\_\_\_ of electrolyte capacitor is not matched, the capacitor will be permanently damaged.

---

## 1.4 INDUCTOR

---

An inductor is a passive electronic component, which works on the principle of electromagnetic induction and is capable of storing electrical energy in the form of magnetic energy. Normally, inductor is made up of iron core, ferrite core, etc over which wire coil is wounded. Current flowing through the inductor creates a magnetic field which has an associated electromagnetic field which opposes the applied voltage.



**Figure 9: An Inductor**

The inductance of a conductor depends on four factors:

- i. The Inductance(L) of a conductor is directly proportional to the number of wounds around the core of the conductor i.e. more is the number of turns around the core, better is the inductance.
- ii. The Inductance(L) of a conductor is directly proportional to the area of cross-section i.e. it increases with an increase in the cross-section area of the coil and reduces with lesser area of cross-section.
- iii. The Inductance(L) is directly proportional to the distance between the coil winding i.e. it increases with the overlapping or narrowing of coils.
- iv. The Inductance (L) is directly proportional to the material used in the core of the inductor.

The inductance can be represented by the formula:

$$L = \frac{\mu \cdot k \cdot N^2 \cdot S}{l} \quad (1.8)$$

Where,

L= Inductance and its unit is Henry

$\mu$ = magnetic permeability (H/m)

k= Nagaoka coefficient

N= Number of turns of coil

S= Cross-sectional area of coil(m<sup>2</sup>)

$l$  = length of coil in axial direction(m)

The inductor is represented in the circuit as shown in **Error! Reference source not found.**:



**Figure 13 : Representation of an inductor in a circuit**

---

## **4.1 VARIOUS TYPES OF INDUCTORS**

---

Inductors are used in variety of electronic applications. They come in different form factors to suit the different industrial needs. Some of the common types of inductors are explained in the section below.

### **1.4.1.1 Air Core Inductor**

In air core inductors, the magnetic core is not used. The wire is wound around the plastic, ceramic or other nonmagnetic core to avoid the magnetic effect, hence they are suited for high frequency applications. Figure 13 is an example of an air core inductor.

### **1.4.1.2 Ferrite Core inductor**

In ferrite core inductor, the inner core of the inductor is made up of ferrite ore, which results in the increase in the value of inductance of an inductor by many folds.

### **1.4.1.3 Iron Core Inductor**

In iron core inductor, as the name suggest, the inner core is made up of iron, which is laminated to prevent heating. This type of inductor offers the highest inductance and is used in the application such as power supplies of audio frequency devices, switch mode power supplies (SMPS), etc.

### **1.4.1.4 Coupled Inductor**

Coupled inductors are based on the principle of mutual inductance and are used in transformers. These type of inductors that share a magnetic path and influence each other. The characteristics of an inductor are opposite to a capacitor. A capacitor blocks DC current and allows AC current. It stores energy in electric field. In contrast to this, an inductor blocks AC and allows DC to pass through it. And it stores the energy in magnetic field.

---

## 1.5 EFFECT OF ADDITION THE COMPONENTS

---

The components can be added in two fashions i.e. in serially fashion and in parallel fashion. Depending on the type of component used like capacitor, resistor or inductor, their values add differently.

### 1.5.1 Adding resistors in series and parallel

When a resistor is added in series, the resistance of all the resistors is added. For example:

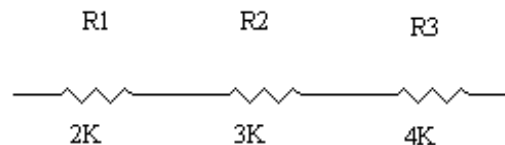
If there are three resistors with resistance  $R_1$ ,  $R_2$  and  $R_3$  respectively, the resulting resistance  $R$  after adding the three resistors serially will be:

$$R = R_1 + R_2 + R_3 \quad (1.9)$$

In case the resistors are added in parallel fashion, the resistance of each of the resistor is added in the reciprocal. i.e.

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} \quad (1.10)$$

*Example 6:* Suppose the resistance of register  $R_1$ ,  $R_2$  and  $R_3$  is 2K,3K and 4K respectively. If all the three registers are connected serially, find the resultant resistance of the circuit.



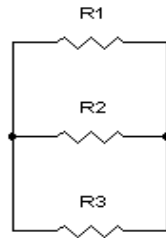
Solution: If the resistors are connected serially, the resultant resistance will be given by the formula:

$$R = R_1 + R_2 + R_3 = 2 + 3 + 4 = 9K$$

*Example 7:* Suppose the resistance of register  $R_1$ ,  $R_2$  and  $R_3$  is 2K,3K and 4K respectively. If all the three registers are connected in parallel, find the resultant resistance of the circuit.

Solution: If the resistors are connected in parallel, the resultant resistance will be given by the formula:





$$\frac{1}{R} = \frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3}$$

$$\frac{1}{R} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4}$$

$$\frac{1}{R} = \frac{13}{12}$$

$$R = 1.082K$$

**Special Case:**

In parallel connection, if all the resistors are of same value, then the resultant resistance of the total circuit is the common resistance value divided by the number of resistors being connected in parallel.

Example: Suppose the resistance of register R1, R2 and R3 is 6K for all the three resistors. If all the three registers are connected in parallel, find the resultant resistance of the circuit.

Solution: Since all the resistance are of the same value and the circuit is connected in parallel fashion, therefore, the final resistance will be 6/3=2K

**1.5.2 Adding capacitors in series and parallel**

When added in serial or parallel fashion, capacitor behaves just opposite to the behavior of resistor. The value is added normally when connected on parallel and adds in reciprocal when connected in series. Its behavior can be represented by the following formulas:

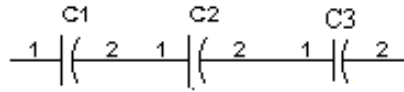
When three capacitors C1,C2 and C3 added in serial fashion, the total C value is given by:

$$\frac{1}{C} = \frac{1}{C1} + \frac{1}{C2} + \frac{1}{C3} \tag{1.11}$$

**Example 8 :** Suppose three capacitorsC1, C2 and C3 have values 0.2 uF, 0.3 uF and 0.4 uF. Find the total value, if they are connected serially.

Solution: When connected serially, the total value will be given by the formula:

$$\frac{1}{C} = \frac{1}{C_1} + \frac{1}{C_2} + \frac{1}{C_3}$$



$$\frac{1}{C} = \frac{1}{0.2} + \frac{1}{0.3} + \frac{1}{0.4}$$

$$\frac{1}{C} = \frac{1.3}{0.12}$$

$$C = 0.092 \mu\text{F}$$

**Special Case :**

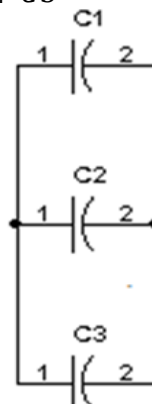
Note that the total series capacitance is always less than the smallest capacitance. Also note that the voltage rating of the series capacitors is equal to the sum of the voltage ratings of the individual capacitors *if* they are the same value. If the capacitors are of different values, and are used in an AC circuit, the voltage division will not be equal. There is a special series case where all the capacitors are the same value. In this case, the total capacitance is the common capacitance value divided by the number of capacitors being connected in series. For example, if three  $1\mu\text{F}$  capacitors are connected in series, the total capacitance is equal to  $1\mu\text{F}/3$ , or  $0.333\mu\text{F}$ . The reciprocal formula will also work, but this method is quicker.

When three capacitors  $C_1, C_2$  and  $C_3$  added in parallel fashion, the total  $C$  value is given by:

$$C = C_1 + C_2 + C_3 \quad (1.12)$$

**Example 9 :** Suppose three capacitors  $C_1, C_2$  and  $C_3$  have values  $0.2\mu\text{F}$ ,  $0.3\mu\text{F}$  and  $0.4\mu\text{F}$ . Find the total value, if they are connected serially.

Solution: When connected in parallel fashion, the total value will be given by the formula:  $C = C_1 + C_2 + C_3$



$$C = 0.2 + 0.3 + 0.4$$

$$C = 0.9 \mu\text{F}$$

### 1.5.3 Adding inductors in series and parallel

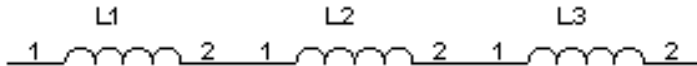
Inductor's behavior is same as a resistor, when added in parallel or serially. When inductors are connected serially, the value is added normally. For example, if three inductors with values L1, L2 and L3 respectively are added serially, the final value L is given by the formula:

$$L = L1 + L2 + L3 \quad (1.13)$$

**Example 10 :** Suppose three inductors L1, L2 and L3 have values 2mH, 3mH and 4mH. Find the total value, if they are connected serially.

Solution: Since the inductors are connected serially, the total value is given by the formula:

$$L = L1 + L2 + L3$$



$$L = 2 + 3 + 4$$

$$L = 9\text{mH}$$

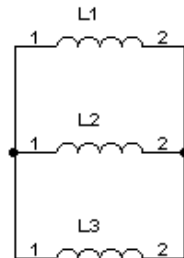
When Inductors are added in parallel fashion, they add in reciprocal. For example, if three inductors with values L1, L2 and L3 respectively are added serially, the final value L is given by the formula:

$$\frac{1}{L} = \frac{1}{L1} + \frac{1}{L2} + \frac{1}{L3} \quad (1.14)$$

**Example 11 :** Suppose the value of inductor L1, L2 and L3 is 2mH,3mH and 4mH respectively. If all the three inductors are connected in parallel, find the total value L of the circuit.

Solution: If the inductors are connected in parallel, the total value will be given by the formula:

$$\frac{1}{L} = \frac{1}{L1} + \frac{1}{L2} + \frac{1}{L3}$$

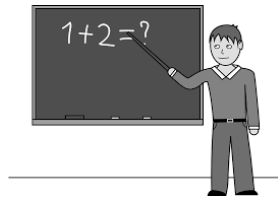


$$\frac{1}{L} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4}$$

$$\frac{1}{L} = \frac{13}{12}$$

$$L = 1.082mH$$

*Special Case:* There is a special parallel case where all the inductors are the same value. In this case, the total inductance is the common inductance value divided by the number of inductors being connected in parallel. For example, if six, 6mH inductors are connected in parallel, the total inductance is equal to 6mH/6, or 1uH.



## Check Your Progress

11. An inductor is a \_\_\_\_\_ electronic component.
12. Coupled inductors are based on the principle of \_\_\_\_\_ and are used in transformers.

---

## 1.6 SUMMARY

---

1. Active components are those components which require an external power source for operation.
2. Passive components are those components which do not require an external power source for operation.
3. Resistor has a very important property: it resists the flow of charge, thus allowing us to control the current.
4. Based on the type of material used for insulation, its ability to handle different amount of electrical power, response to the variation in the voltage, and ability to mechanically change the resistance, there are many variants of a resistor.
5. Wirewound registers are made by wrapping a resistive wire, which is usually made up of nickel-chromium alloy, around a non-conductive rod, which is mostly made up of ceramic due to its desired heat properties since the current carrying conducting wires produces heat.
6. Carbon Composition Resistors are formed by mixing finely grinding carbon with resin.

7. Carbon film resistors are made by depositing a very thin layer of carbon on a ceramic rod, which is then protected by flameproof jacket.
8. Variable resistors or potentiometers, as the name suggests, the value of resistance can be varied during the operation in the variable resistor.
9. Different color rings around the body of the resistor is used represent the resistance of a resistor.
10. There are situations when it is required to store electrical charge in a circuit for a short duration of time. For this purpose, a passive electronic component, known as capacitor is used.
11. Electrostatic Induction is a method to create static electricity on a material by bringing an electrically charged object near it.
12. A capacitor can be either polarized or non-polarized, which means either that they have to be connected the right way round (like the LED) or you can connect them anyway you like (like the resistor).
13. An inductor works on the principle of electromagnetic induction and is capable of storing electrical energy in the form of magnetic energy.
14. The Inductance(L) of a conductor is directly proportional to the number of wounds around the core of the conductor.
15. In air core inductors, the magnetic core is not used. The wire is wound around the plastic, ceramic or other nonmagnetic core to avoid the magnetic effect, hence they are suited for high frequency applications.
16. In ferrite core inductor, the inner core of the inductor is made up of ferrite ore, which results in the increase in the value of inductance of an inductor by many folds.
17. The components can be added in two fashions i.e. in serially fashion and in parallel fashion.

---

## **1.7 ANSWER TO CHECK YOUR PROGRESS**

---

1. Ohm's
2. Current
3. Frequency
4. Carbon Composition Resistors

5. Metal Oxide Resistors
6. Metal Film Resistors
7. Potentiometer
8. Ohm.
9. Reliability
10. Electrostatic Induction
11. Polarity
12. Passive
13. Mutual inductance

---

## 1.8 TERMINAL QUESTIONS CHECK YOUR PROGRESS

---

- i. What is Ohm's Laws?
- ii. How to calculate the three band Resistor color code? Define the steps.
- iii. Define different types of resistor and its characteristics?
- iv. What is Potentiometer?
- v. What is capacitance of a capacitor?
- vi. What are the factors for affecting the capacitance?
- vii. Define the followings-
  - a. Paper Capacitor
  - b. Electrolyte Capacitor
  - c. Mica Capacitor
  - d. Ceramic Capacitor
- viii. A capacitor stores a charge of  $3.0 \times 10^{-4}$  C when the p.d. across its terminals is 600 V.  
[Answer: 0.5  $\mu$ F]
- ix. A 30  $\mu$ F capacitor stores  $6.0 \times 10^{-3}$  C of charge. How much energy is stored in the capacitor? [Answer: 0.60 J]
- x. What is Inductance? How it is measured?
- xi. What is self Inductance? Explain differences between Magnetic field and Electric field?

# UNIT-2

---

## DIODE

---

### Structure

#### 2.0 Learning Objectives

#### 2.1 Introduction

#### 2.2 What is a Diode?

#### 2.3 How Diodes work?

#### 2.4 Types of diodes

##### 2.4.1 Rectifier diodes

###### 2.4.1.1 Half-wave rectifier

###### 2.4.1.2 Full-wave rectifier

##### 2.4.2 Switching diodes

###### 2.4.3.1 Clipping

###### 2.3.3.2 Clamping

##### 2.4.3 Zener Diodes

###### 2.4.3.1 V-I Characteristics of Zener Diode

###### 2.4.3.2 Application of Zener Diode

##### 2.4.4 Optical Diodes

###### 2.4.4.1 Light Emitting Diode

###### 2.4.4.2 Photodiode

###### 2.4.4.3 Optocoupler

#### 2.5 Summary

#### 2.6 Answers to Check Your Progress

#### 2.7 Terminal Questions

---

## 2.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to: Define a diode. Understand the working and characteristics of a diode. Compare the ideal and practical characteristics of a diode. Define various types of diodes.

Differentiate between clipping and clamping. Define Zener diode. Explain the VI characteristics of Zener diode.

---

## 2.1 INTRODUCTION

---

Diode is an electronic component that allows current to only flow in one direction. Before the advent of semiconductors, vacuum tube diodes were used. A diode is comprised of two types of semiconductor crystal (usually made from silicon or germanium) that are highly refined then doped with an impurity. Depending on the impurity, the crystal can either be called an “N-type” or “P-type”. When you put an N-doped region next to a P-doped region, a diode or PN junction is formed. In our diodes, the P-region is called the anode, and the N-region is called the cathode. As you can imagine, these properties are useful, allowing current to flow only in one direction.

In this unit, we will discuss what a diode is and how it works. We will also discuss its characteristics. We will also discuss types of diodes and its applications.

---

## 2.2 WHAT IS A DIODE?

---

A Diode is the simplest two-terminal unilateral semiconductor device. It allows current to flow only in one direction and blocks the current that flows in the opposite direction. The two terminals of the diode are called as anode and cathode. The symbol of diode is as shown in the Figure 10 below.



**Figure 10: Diode Symbol**

function of a diode is to allow an electric current to pass in one direction (called the diode's forward direction), while blocking current in the opposite direction (the reverse direction). Thus, the diode can be viewed as an electronic version of a check valve. This unidirectional behavior is called rectification, and is used to convert alternating current (AC) to direct current (DC), including extraction of modulation from radio signals in radio receivers—these diodes are forms of rectifiers.

However, diodes can have more complicated behavior than this simple on–off action, because of their nonlinear current-voltage characteristics. Semiconductor diodes begin conducting electricity only if a certain threshold voltage or cut-in voltage is present in the forward direction (a state in which the diode is said to be forward-biased). The voltage drop across a forward-biased diode varies only a little with the current, and is a function of temperature; this effect can be used as a temperature sensor or as a voltage reference.



A semiconductor diode's current–voltage characteristic can be tailored by selecting the semiconductor materials and the doping impurities introduced into the materials during manufacture. These techniques are used to create special-purpose diodes that perform many different functions. For example, diodes are used to regulate voltage (Zener diodes), to protect circuits from high voltage surges (avalanche diodes), to electronically tune radio and TV receivers (varactor diodes), to generate radio-frequency oscillations (tunnel diodes, Gunn diodes, IMPATT diodes), and to produce light (light-emitting diodes). Tunnel, Gunn and IMPATT diodes exhibit negative The characteristics of a diode closely match to that of a switch. An ideal switch when open does not conduct current in either directions.

Ideally, in one direction that is indicated by the arrow head diode must behave short circuited and in other one that opposite to that of the direction of arrow head must be open circuited. By ideal characteristics, the **diodes** is designed to meet these features theoretically but are not achieved practically. So the practical **diode characteristics** are only close to that of the desired. (Figure 15 and Figure 16).

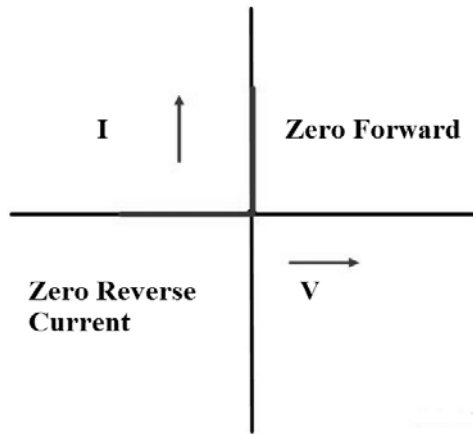


Figure 15 : Characteristic curve of a diode

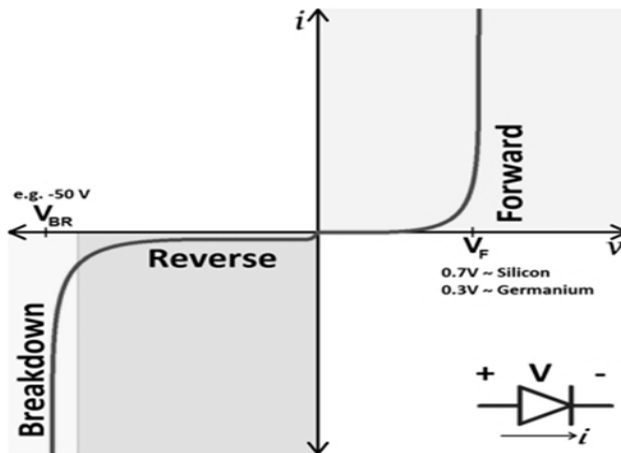
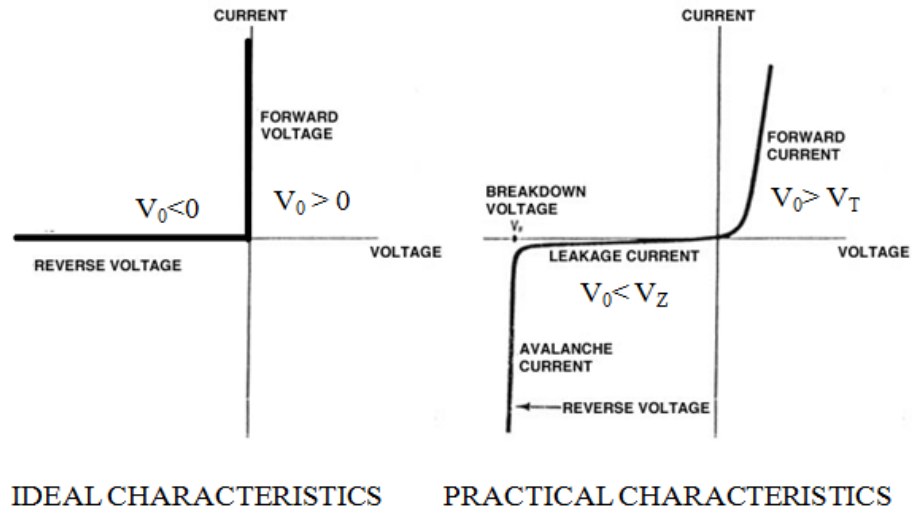


Figure 16 : Practical Diode Characteristics

Ideally, in one direction that is indicated by the arrow head diode must behave short circuited and in other one that opposite to that of the direction of arrow head must be open circuited. By ideal characteristics, the **diodes** is designed to meet these features theoretically but are not achieved practically. So the practical **diode characteristics** are only close to that of the desired.



**Figure 117: Ideal vs. Practical Diode Characteristics**

---

## 2.3 HOW DIODES WORK?

---

The **diode** operates when a voltage signal is applied across its terminals. The application of a DC voltage to make the diode operate in a circuit is called as 'Biasing'. As already mentioned above the diode resembles to that of a one way switch so it can either be in a state of conduction or in a state of non conduction. The 'ON' state of a diode is achieved by 'Forward biasing' which means that positive or higher potential is applied to the anode and negative or lower potential is applied at the cathode of the diode. In other words, the 'ON' state of diode has the applied current in the same direction of the arrow head. The 'OFF' state of a diode is achieved by 'Reverse biasing' which means that positive or higher potential is applied to the cathode and negative or lower potential is applied at the anode of the diode. In other words, the 'OFF' state of diode has the applied current in the opposite direction of the arrow head.

During 'ON' state, the practical *diode* offers a resistance called as the 'Forward resistance'. The diode requires a forward bias voltage to switch to the 'ON' condition which is called Cut-in-voltage. The diode starts conducting in reverse biased mode when the reverse bias voltage

exceeds its limit which is called as the Breakdown voltage. The diode remains in 'OFF' state when no voltage is applied across it.

A simple p-n junction diode is fabricated by doping p and n type layers on a silicon or germanium wafer. The germanium and silicon materials are preferred for diode fabrication because:

- They are available in high purity.
- Slight doping like one atom per ten million atoms of a desired impurity can change the conductivity to a considerable level.
- The properties of these materials change on applying heat and light and hence it is important in the development of heat and light sensitive devices.

---

## 2.4 TYPES OF DIODES

---

We can distinguish the following types of diodes:

- **Rectifier diodes** are typically used for power supply applications. Within the power supply, you will see diodes as elements that convert AC power to DC power;
- **Switching diodes** have lower power ratings than rectifier diodes, but can function better in high frequency application and in clipping and clamping operations that deal with short-duration pulse waveforms;
- **Zener diodes**, a special kind of diode that can recover from breakdown caused when the reverse-bias voltage exceeds the diode breakdown voltage. These diodes are commonly used as voltage-level regulators and protectors against high voltage surges;
- **Optical diodes**

### 2.4.1 Rectifier diodes

A rectifier is a dispositive that ideally transforms the AC input voltage into a DC voltage (voltage is always positive or zero). These diodes have the largest ratings and sometime can be quite big in volume. As a rule of thumb, the bigger the diode (more pn surface junction available for heat dissipation), the higher the ratings.

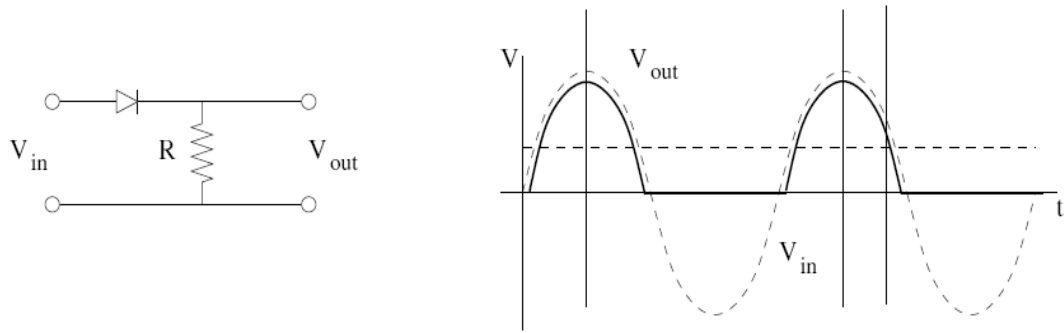
#### 2.4.1.1 Half-wave rectifier

A half-wave rectifier is composed of a single diode that connects an AC source to a load. In figure 3 the load is represented by a resistor. The diode conducts on AC voltage only when its anode is positive with respect to the cathode (i.e. greater than 0.7 V for a silicon diode).

The output has therefore only a positive component with an average value:

$$V_{ave} = \frac{1}{T} \int_0^{T/2} V_p \sin \omega t \, dt = \frac{V_p}{\pi}$$

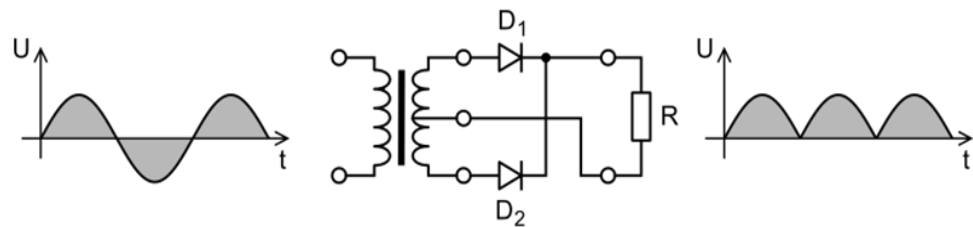
The output peak voltage is the AC source minus the voltage drop of the diode, that in most cases can be neglected.



**Figure 128: Half wave rectifier. Note the effect of the 0.7 V diode voltage drop. If  $V_p \gg 0.7V$ , it can be neglected**

#### 2.4.1.2 Full-wave rectifier

In half-wave rectifiers, half of the power provided by the source is not used. To solve this problem, we have to use full-wave rectifiers. The minimum full-wave rectifier is composed of two diodes, but it requires a center tapped transformer. Figure 18 shows a bridge rectifier, composed of four diodes that can use a “normal” transformer.

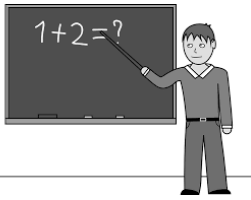


**Figure 13: Full wave rectifier with 4 diode configuration<sup>1</sup>**

The AC current, according to its direction, flows either in the top or in the bottom part of the bridge in each half-cycle. In the output voltage we will have a component for both negative and positive parts of the input voltage. In both case the current passes through two forward-biased diodes in series, what produces a voltage drop of 1.4V.

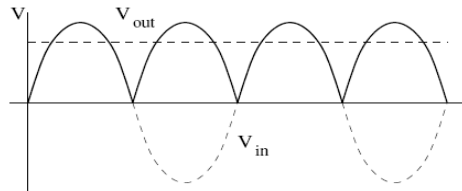
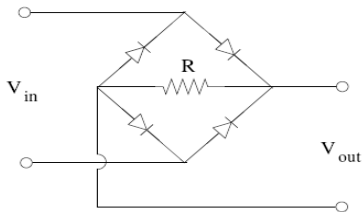
The average voltage of a full-wave rectifier is:

$$V_{ave} = \frac{1}{T/2} \int_0^{T/2} V_p \sin \omega t \, dt = \frac{2V_p}{\pi}$$



## Check Your Progress

1. A \_\_\_\_\_ is the simplest two-terminal unilateral semiconductor device.
2. The characteristics of a diode closely match to that of a \_\_\_\_\_.
3. The application of a DC voltage to make the diode operate in a circuit is called as \_\_\_\_\_.
4. \_\_\_\_\_ are typically used for power supply applications.
5. In \_\_\_\_\_ clippers, when diode is in the 'off condition, transmission of input signal should take place to output.
6. In \_\_\_\_\_ clippers, when the diode is in 'OFF' position, there will be no transmission of input signal to output.
7. When a portion of both positive and negative of each half cycle of the input voltage is to be clipped (or removed), \_\_\_\_\_ clipper is employed.



**Figure 20: Full wave rectifier. In this case the voltage drop, not shown in the graphic, will be 1.4 V because two diodes are crossed**

### Rectifier filters

The waveforms generated directly by the rectifiers described above, are not very useful, but can be smoothed to produce almost perfect DC. For that purpose we can use the inertia properties of capacitors and inductors.

Although the capacitor does a good job producing DC, a significant ripple voltage remains. During the discharging period, the capacitor voltage reduces exponentially. At the end of the discharge, the output voltage is:

$$V_{out} = V_p e^{-(t_2 - t_1) / R_L C}$$

The discharging time is  $t_2 - t_1 \sim T$ , so the ripple voltage is:

$$v_r = \Delta V = V_p - V_p e^{-T/R_L C} \simeq V_p \frac{T}{R_L C} = \frac{V_p}{\nu R_L C}$$

In case of a full-wave rectifier the ripple voltage is:

$$v_r = V_p \frac{T}{2R_L C} = \frac{V_p}{2\nu R_L C}$$

## 2.4.2 Switching diodes

A switching diode provides the same functionality as a switch. It has high resistance below the specified applied voltage similar to an open switch, whereas above that voltage it changes in a sudden way to the low resistance of a closed switch.

### 2.4.3.1 Clipping

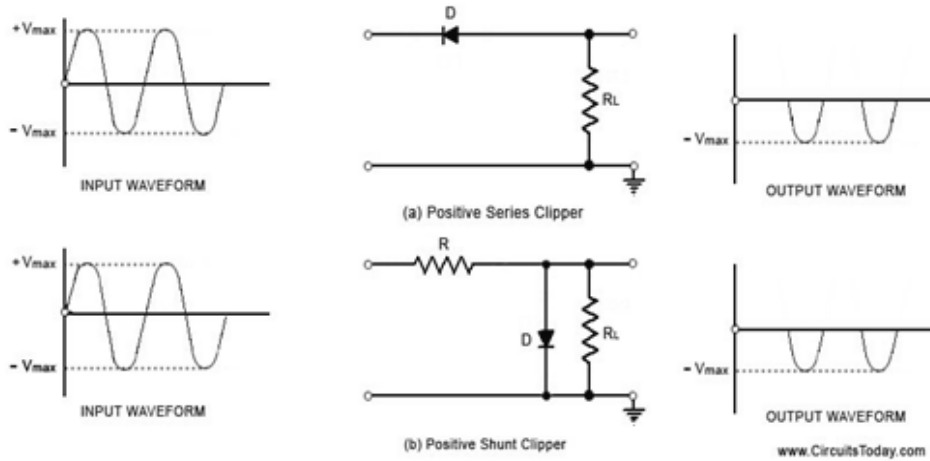
The function of a clipping circuit is to cut off part of an input waveform. When the diode is forward biased, it acts as a closed switch, and when it is reverse biased, it acts as an open switch. Different levels of clipping can be obtained by varying the amount of voltage of the battery and also interchanging the positions of the diode and resistor. Depending on the features of the diode, the positive or negative region of the input signal is “clipped” off and accordingly the diode clippers may be positive or negative clippers. There are two general categories of clippers: series and parallel (or shunt). The series configuration is defined as one where diode is in series with the load, while the shunt clipper has the diode in a branch parallel to the load.

Clipping The function of a clipping circuit is to cut off part of an input waveform. When the diode is forward biased, it acts as a closed switch, and when it is reverse biased, it acts as an open switch. Different levels of clipping can be obtained by varying the amount of voltage of the battery and also interchanging the positions of the diode and resistor. Depending on the features of the diode, the positive or negative region of the input signal is “clipped” off and accordingly the diode clippers may be positive or negative clippers. There are two general categories of clippers: series and parallel (or shunt). The series configuration is defined as one where diode is in series with the load, while the shunt clipper has the diode in a branch parallel to the load.

#### 1. Positive Clipper and Negative Clipper

##### *Positive Diode Clipper*

In a positive clipper, the positive half cycles of the input voltage will be removed. The circuit arrangements for a positive clipper are illustrated in the figure given below.



**Figure 21: Positive Series clipper and Positive Shunt Clipper**

As shown in the figure above, the diode is kept in series with the load. During the positive half cycle of the input waveform, the diode 'D' is reverse biased, which maintains the output voltage at 0 Volts. Thus causes the positive half cycle to be clipped off. During the negative half cycle of the input, the diode is forward biased and so the negative half cycle appears across the output.

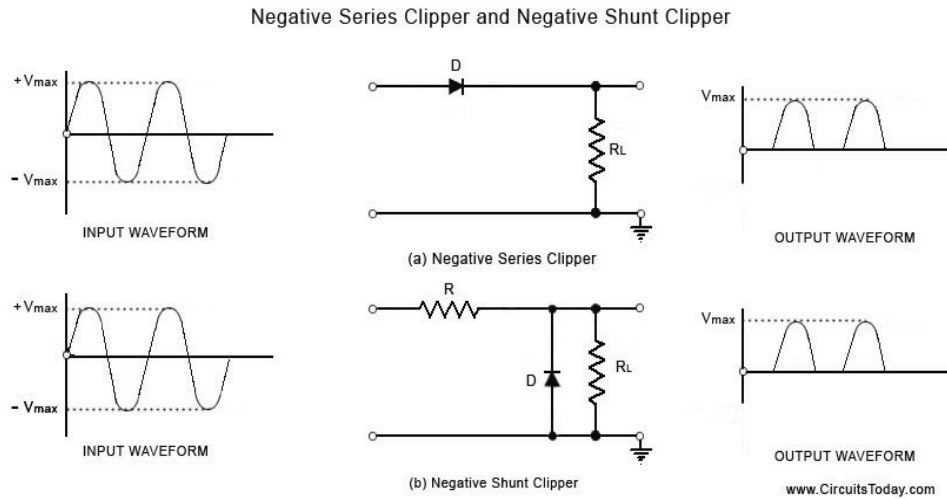
In Figure (b), the diode is kept in parallel with the load. This is the diagram of a positive shunt clipper circuit. During the positive half cycle, the diode 'D' is forward biased and the diode acts as a closed switch. This causes the diode to conduct heavily. This causes the voltage drop across the diode or across the load resistance  $R_L$  to be zero. Thus output voltage during the positive half cycles is zero, as shown in the output waveform. During the negative half cycles of the input signal voltage, the diode D is reverse biased and behaves as an open switch. Consequently the entire input voltage appears across the diode or across the load resistance  $R_L$  if R is much smaller than  $R_L$ .

Actually the circuit behaves as a voltage divider with an output voltage of  $[R_L / R + R_L] V_{\max} = -V_{\max}$  when  $R_L \gg R$

### ***Negative Diode Clipper***

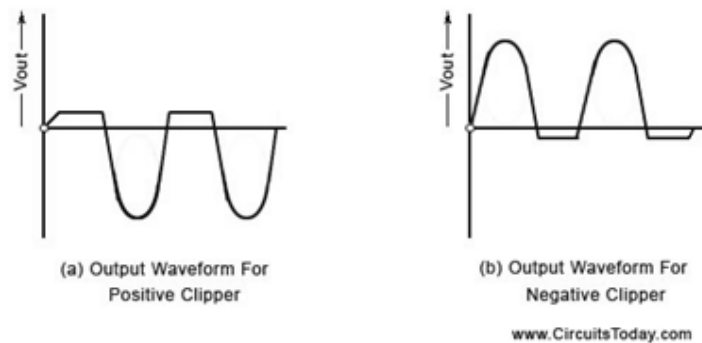
The negative clipping circuit is almost same as the positive clipping circuit, with only one difference. If the diode in figures (a) and (b) is reconnected with reversed polarity, the circuits will become for a negative series clipper and negative shunt clipper respectively. The negative series

and negative shunt clippers are shown in figures (a) and (b) as given below.



**Figure 22: Negative Series Clipper and Negative Shunt Clipper**

In all the above discussions, the diode is considered to be ideal one. In a practical diode, the breakdown voltage will exist (0.7 V for silicon and 0.3 V for Germanium). When this is taken into account, the output waveforms for positive and negative clippers will be of the shape shown in the figure below.



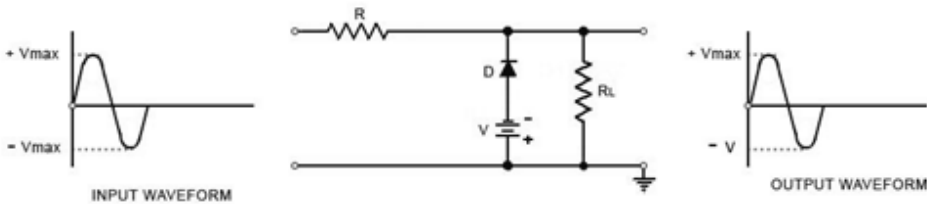
**Figure 23: Biased Negative Clipper**

## 2. Biased Positive Clipper and Biased Negative Clipper

A biased clipper comes in handy when a small portion of positive or negative half cycles of the signal voltage is to be removed. When a small portion of the negative half cycle is to be removed, it is called a biased

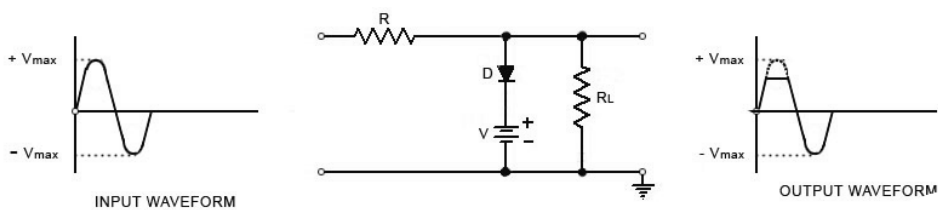


negative clipper. The circuit diagram and waveform is shown in the figure below.



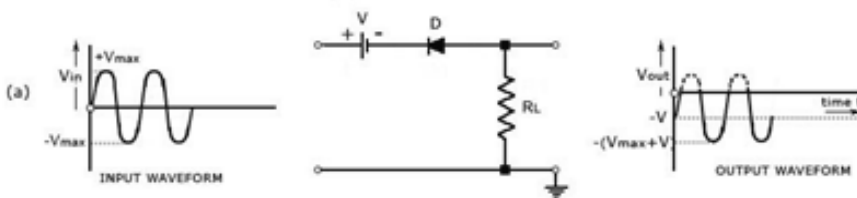
**Figure 24: Biased Negative Clipper**

In a biased clipper, when the input signal voltage is positive, the diode 'D' is reverse-biased. This causes it to act as an open-switch. Thus the entire positive half cycle appears across the load, as illustrated by output waveform [figure (a)]. When the input signal voltage is negative but does not exceed battery the voltage 'V', the diode 'D' remains reverse-biased and most of the input voltage appears across the output. When during the negative half cycle of input signal, the signal voltage becomes more than the battery voltage V, the diode D is forward biased and so conducts heavily. The output voltage is equal to '- V' and stays at '- V' as long as the magnitude of the input signal voltage is greater than the magnitude of the battery voltage, 'V'. Thus a biased negative clipper removes input voltage when the input signal voltage becomes greater than the battery voltage. Clipping can be changed by reversing the battery and diode connections, as illustrated in figure (b).



**Figure 25: Biased Positive Clipper**

Some of other biased clipper circuits are given below in the Figure 14. While drawing the wave-shape of the output basic principle discussed above are followed. The diode has been considered as an ideal one.



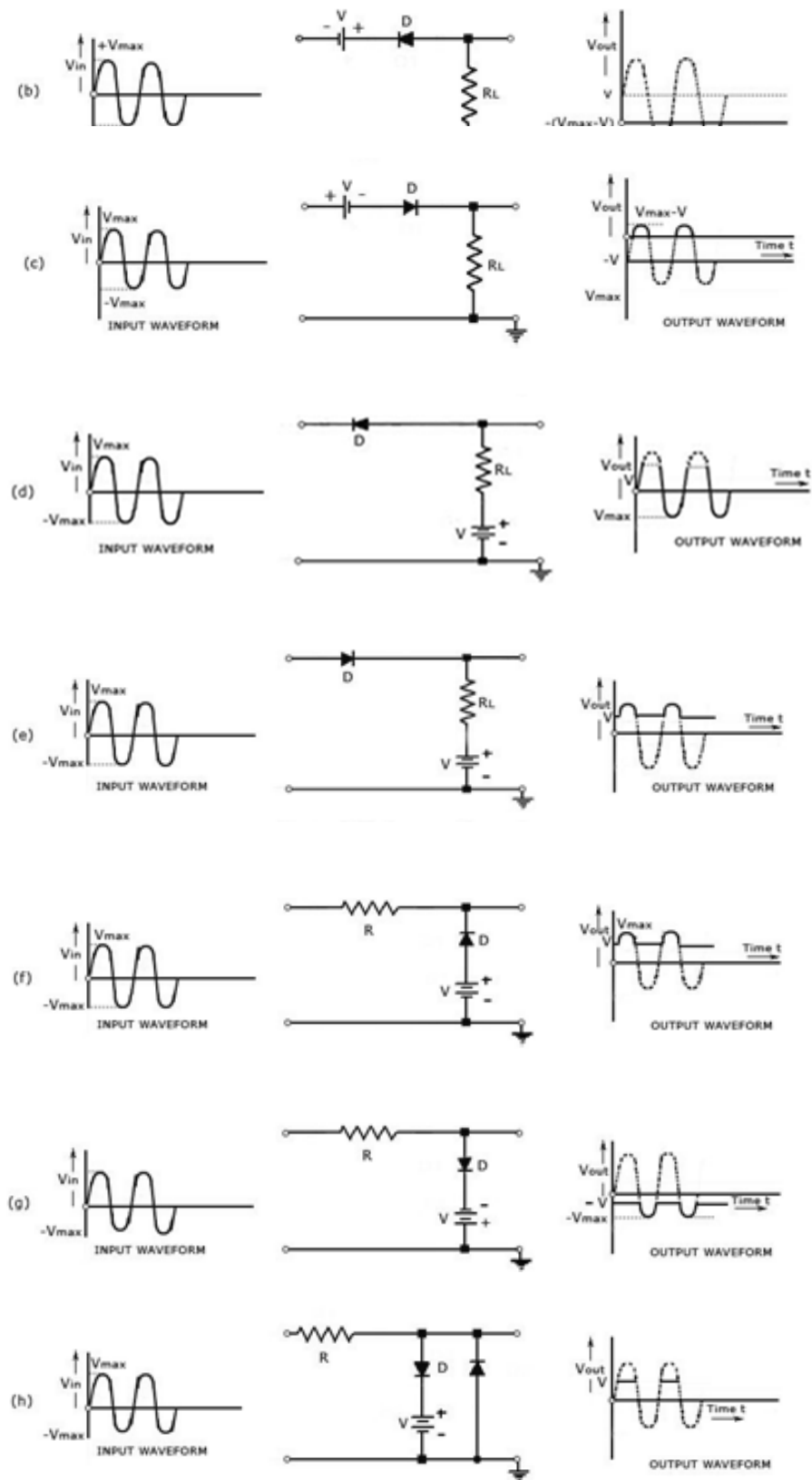
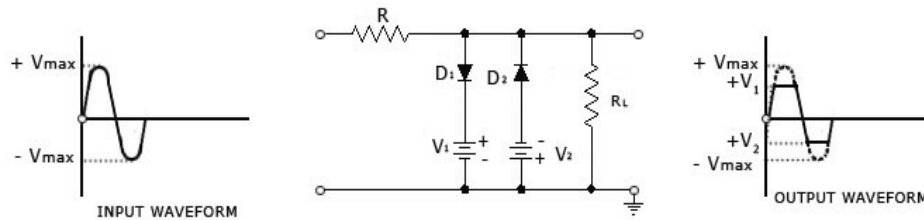


Figure 146: Different Clipper Circuits

### 3. Combination Clipper

When a portion of both positive and negative of each half cycle of the input voltage is to be clipped (or removed), combination clipper is employed. The circuit for such a clipper is given in the



**Figure 157: Combinational Clipper**

The action of the circuit is summarized below. For positive input voltage signal when input voltage exceeds battery voltage ' $+V_1$ ' diode ' $D_1$ ' conducts heavily while diode ' $D_2$ ' is reverse biased and so voltage ' $+V_1$ ' appears across the output. This output voltage ' $+V_1$ ' stays as long as the input signal voltage exceeds ' $+V_1$ '. On the other hand for the negative input voltage signal, the diode ' $D_1$ ' remains reverse biased and diode ' $D_2$ ' conducts heavily only when input voltage exceeds battery voltage ' $V_2$ ' in magnitude. Thus during the negative half cycle the output stays at ' $-V_2$ ' so long as the input signal voltage is greater than ' $-V_2$ '.

#### Drawbacks of Series and Shunt Diode Clippers

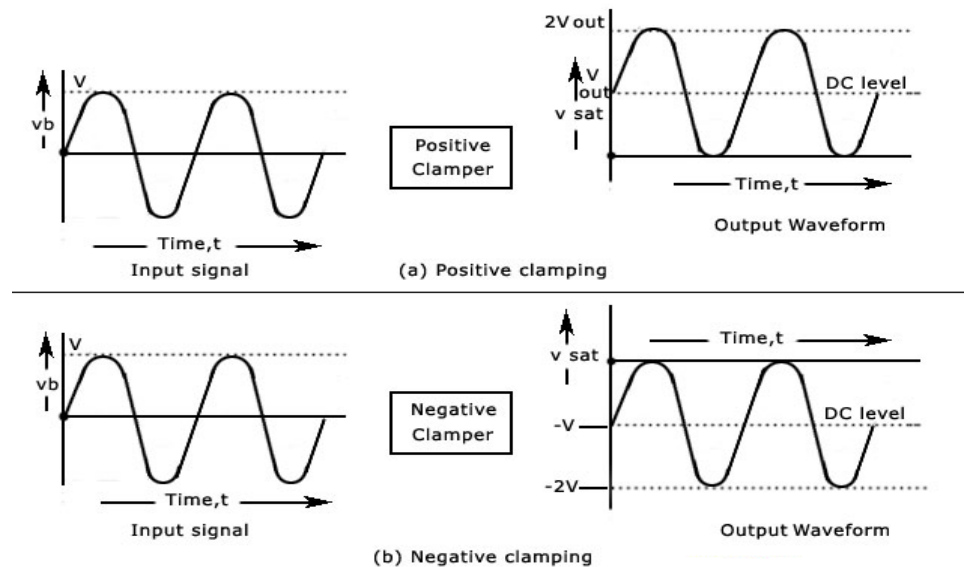
- In series clippers, when the diode is in 'OFF' position, there will be no transmission of input signal to output. But in case of high frequency signals transmission occurs through diode capacitance which is undesirable. This is the drawback of using diode as a series element in such clippers.
- In shunt clippers, when diode is in the 'off condition, transmission of input signal should take place to output. But in case of high frequency input signals, diode capacitance affects the circuit operation adversely and the signal gets attenuated (that is, it passes through diode capacitance to ground).

#### 2.3.3.2 Clamping

A clamping circuit is used to place either the positive or negative peak of a signal at a desired level. The dc component is simply added or subtracted to/from the input signal. The clamper is also referred to as an IC restorer and ac signal level shifter.

In some cases, like a TV receiver, when the signal passes through the capacitive coupling network, it loses its dc component. This is when the clamper circuit is used so as to re-establish the the dc component into the signal input. Though the dc component that is lost in transmission is not the same as that introduced through a clamping circuit, the necessity to establish the extremity of the positive or negative signal excursion at some reference level is important.

A clamp circuit adds the positive or negative dc component to the input signal so as to push it either on the positive side, as illustrated in figure (a) or on the negative side, as illustrated in figure (b). The circuit will be called a positive clamper , when the signal is pushed upward by the circuit. When the signal moves upward, as shown in figure (a), the negative peak of the signal coincides with the zero level. The circuit will be called a negative clamper, when the signal is pushed downward by the circuit. When the signal is pushed on the negative side, as shown in figure (b), the positive peak of the input signal coincides with the zero level.



**Figure 168: Positive and Negative Clamping Circuits**

For a clamping circuit at least three components — a diode, a capacitor and a resistor are required. Sometimes an independent dc supply is also required to cause an additional shift. The important points regarding clamping circuits are:

- i. The shape of the waveform will be the same, but its level is shifted either upward or downward,
- ii. There will be no change in the peak-to-peak or rms value of the wave-form due to the clamping circuit. Thus, the input waveform and output waveform will have the same peak-to-peak value that is,  $2V_{max}$ . This is shown in above figure . It must also be noted

that same readings will be obtained in the ac voltmeter for the input voltage and the clamped output voltage.

- iii. There will be a change in the peak and average values of the waveform. In the figure shown above, the input waveform has a peak value of  $V_{max}$  and average value over a complete cycle is zero. The clamped output varies from  $2 V_{max}$  and 0 (or 0 and  $-2V_{max}$ ). Thus the peak value of the clamped output is  $2V_{max}$  and average value is  $V_{max}$ .
- iv. The values of the resistor  $R$  and capacitor  $C$  affect the waveform.
- v. The values for the resistor  $R$  and capacitor  $C$  should be determined from the time constant equation of the circuit,  $t = RC$ . The values must be large enough to make sure that the voltage across the capacitor  $C$  does not change significantly during the time interval the diode is non-conducting. In a good clamper circuit, the circuit time constant  $t = RC$  should be at least ten times the time period of the input signal voltage.

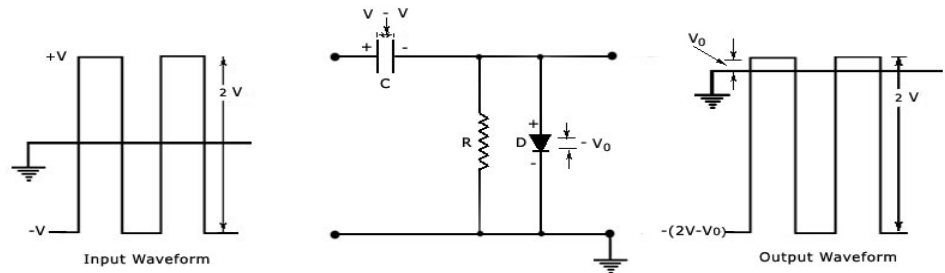
It is advantageous to first consider the condition under which the diode becomes forward biased.

Clamping circuits are often used in television receivers as dc restorers. The signal that is sent to the TV receiver may lose the dc components after being passed through capacitively coupled amplifiers. Thus the signal loses its black and white reference levels and the blanking level. Before passing these signals to the picture tube, these reference levels have to be restored. This is done by using clamper circuits. They also find applications in storage counters, analog frequency meter, capacitance meter, divider and stair-case waveform generator.

Consider a negative clamping circuit, a circuit that shifts the original signal in a vertical downward direction, as shown in the figure below. The diode  $D$  will be forward biased and the capacitor  $C$  is charged with the polarity shown, when an input signal is applied. During the positive half cycle of input, the output voltage will be equal to the barrier potential of the diode,  $V_0$  and the capacitor is charged to  $(V - V_0)$ . During the negative half cycle, the diode becomes reverse-biased and acts as an open-circuit. Thus, there will be no effect on the capacitor voltage. The resistance  $R$ , being of very high value, cannot discharge  $C$  a lot during the negative portion of the input wave-form. Thus during negative input, the output voltage will be the sum of the input voltage and the capacitor voltage and is equal to  $-V - (V - V_0)$  or  $-(2V - V_0)$ . The value of the peak-to-peak output will be the

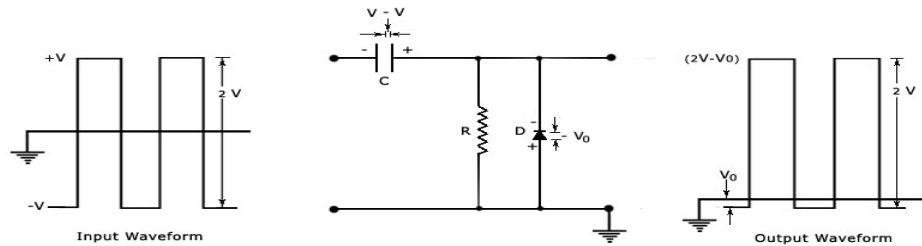
difference of the negative and positive peak voltage levels is equal to  $V_0 - [-(2V - V_0)]$  or  $2V$ .

The figure shown below can be modified into a positive clamping circuit by reconnecting the diode with reversed polarity. The positive clamping circuit moves the original signal in a vertical upward direction. A positive clamping circuit is shown in the figure below. It contains a diode  $D$  and a capacitor  $C$  as are contained in a negative clamper. The only difference in the circuit is that the polarity of the diode is reversed. The remaining explanation regarding the working of the circuit is the same as it is explained for the negative clamper. To remember which way the dc level of a signal moves, look at figure shown below. Notice that the diode arrows point downward, the same direction as the dc shift.



**Figure 179: Negative Clamper**

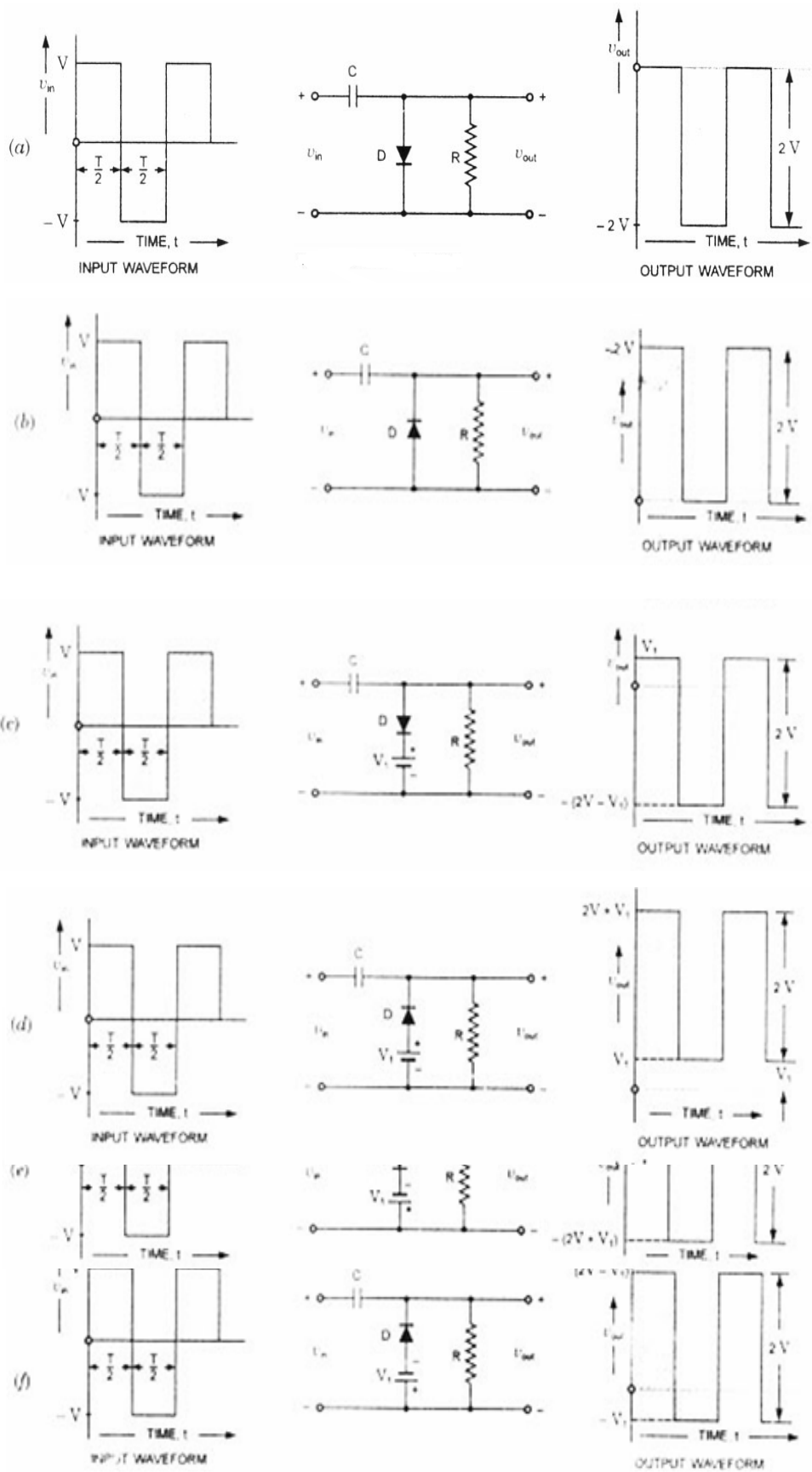
Similarly in the figure shown below, the diode arrow points upward, again the same direction as the dc shifts. It means that, when the diode points upward. We have a positive dc clamper and when the diode points downward, the circuit is a negative dc clamper.

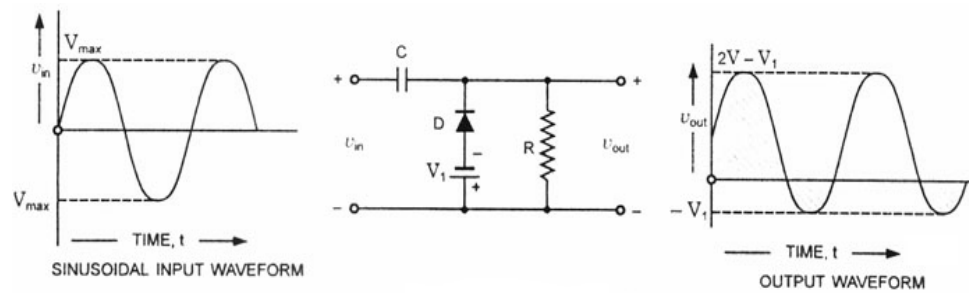


**Figure 18: Positive Clamper**

A number of clamping circuits with their effect on the input signal are shown in the figure given below. All the figures shown below have the input and output signals in square waves, the same procedure can be used for sinusoidal inputs. In fact, one approach to the analysis of clamping networks with sinusoidal inputs is to replace the sinusoidal wave signal by a square wave of the same peak values. The resulting output will then form an envelope for the sinusoidal response, as illustrated in figure (g)

for a network appearing in figure (f). The diodes have been assumed to be ideal and  $5 RC \gg T/2$  in drawing the output wave-forms.





**Figure 19: Different Coupling Circuits**

### 2.4.3 Zener Diodes

A Zener diode is a type of diode that permits current to flow in the forward direction like a normal diode, but also in the reverse direction if the voltage is larger than the rated breakdown voltage or "Zener voltage". The device was named for Clarence Zener, who discovered this electrical property.



**Figure 20: Zener Diode**

A conventional solid-state diode will not let significant current flow if reverse-biased below its reverse breakdown voltage. By exceeding the reverse bias breakdown voltage, a conventional diode is subject to high current flow due to avalanche breakdown. Unless this current is limited by external circuitry, the diode will be permanently damaged. In case of large forward bias (current flow in the direction of the arrow), the diode exhibits a voltage drop due to internal resistance. The amount of the voltage drop depends on the design of the diode.

A Zener diode exhibits almost the same properties, except the device is especially designed so as to have a greatly reduced breakdown voltage, the so-called Zener voltage. A Zener diode contains a heavily doped p-n junction allowing electrons to tunnel from the valence band of the p-type material to the conduction band of the n-type material. A reverse-biased Zener diode will exhibit a controlled breakdown and let the current flow to keep the voltage across the Zener diode at the Zener voltage. For example, a diode with a Zener breakdown voltage of 3.2 V will exhibit a voltage drop of 3.2 V if reverse biased. However, the current is not unlimited, so the Zener diode is typically used to generate a reference voltage for an amplifier stage, or as a voltage stabilizer for low-current applications.



The breakdown voltage can be controlled quite accurately in the doping process. Tolerances to within 0.05% are available though the most widely used tolerances are 5% and 10%.

Another mechanism that produces a similar effect is the avalanche effect as in the avalanche diode. The two types of diode are in fact constructed the same way and both effects are present in diodes of this type. In silicon diodes up to about 5.6 volts, the zener effect is the predominant effect and shows a marked negative temperature coefficient. Above 5.6 volts, the avalanche effect becomes predominant and exhibits a positive temperature coefficient.

### 2.4.3.1 V-I Characteristics of Zener Diode

The forward characteristic of a Zener diode is similar to that of a P N Junction diode. The reverse characteristic of Zener diode is obtained as follows.

- The reverse current that is present at the origin and the knee of the curve is due to the reverse leakage current due to the minority carriers. This current is specified by stating its value at 80% of the Zener voltage  $V_z$ .
- As the reverse voltage is gradually increased, the breakdown occurs at the knee and the current increases rapidly. To control this current a suitable external resistance has to be used. The maximum permissible value of the current is denoted by  $I_{zmax}$ . The minimum usable current is  $I_{zmin}$

The voltage across the terminals of the diode for a current  $I_z$  which is the approximate midpoint of the linear range of the reverse characteristics is called the zener voltage  $V_z$ . At the knee point, the breakdown voltage remains constant between  $I_{zmax}$  and  $I_{zmin}$ . This ability of a diode is called regulating ability and is an important feature of a Zener Diode

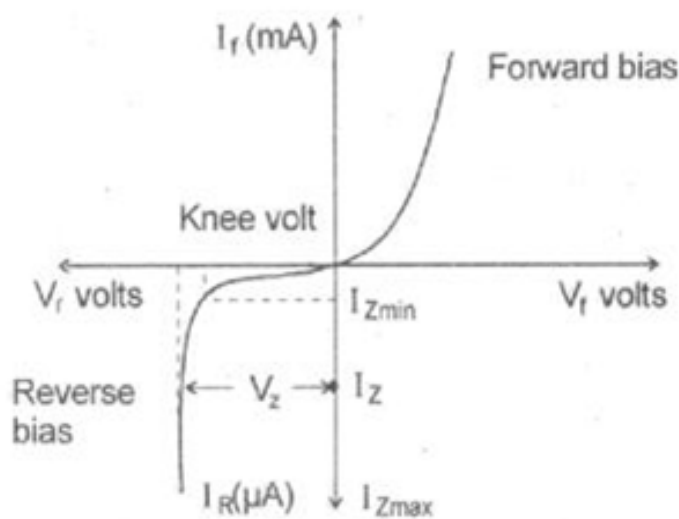


Figure 33: V-I Characteristics of Zener Diode

### **2.4.3.2 Application of Zener Diode**

It can be used:

- a. As voltage regulators
- b. As peak clippers
- c. For reshaping waveforms
- d. For meter protection against damage from accidental application of excessive voltage.

### **2.4.4 Optical Diodes**

#### **2.4.4.1 Light Emitting Diode**

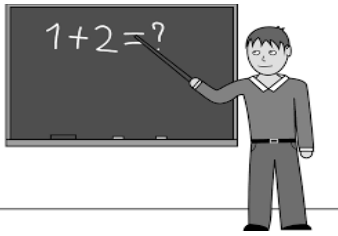
A light-emitting diode (LED) is a semiconductor device that emits incoherent monochromatic light when electrically biased in the forward direction. The colour depends on the semiconducting material used, and can be near-ultraviolet, visible or infrared. The wavelength of the light emitted, and therefore its colour, depends on the bandgap energy of the materials forming the p-n junction. A normal diode, typically made of silicon or germanium, emits invisible far-infrared light. Currently used materials for LEDs are gallium arsenide (GaAs) for the infrared, gallium arsenide phosphate (GaAsP) for yellow and red light, and gallium phosphate (GaP) for red and green light. The intensity of the light depends on the current that passes through the LED.

#### **2.4.4.2 Photodiode**

A photodiode is a diode working in reverse polarization and having a window where the light can enter and hit directly the p-n junction. As in the case of the LED, the energy level of the impurities has been chosen in order to allow electrons to jump from valence to conduction band. In the absence of light the leakage current is negligible, but when light is present, the leakage current increases to measurable values.

#### **2.4.4.3 Optocoupler**

An optocoupler is a device that combines a LED and a photodiode in such a way that the, light emitted by the LED hits the photodiode. LED and photodiode can be connected to different circuits, so this device allows to send a signal between two isolated networks coupled just through light emission and reception.



## Check Your Progress

1. A \_\_\_\_\_ diode contains a heavily doped p-n junction allowing electrons to tunnel from the valence band of the p-type material to the conduction band of the n-type material.
2. Optical diode is also known as optical\_\_\_\_\_ .

---

## 2.5 SUMMARY

---

1. Diode allows current to flow only in one direction and blocks the current that flows in the opposite direction.
2. The two terminals of the diode are called as anode and cathode.
3. An ideal switch when open does not conduct current in either directions and in closed state conducts in both directions.
4. By ideal characteristics, the **diodes** is designed to meet these features theoretically but are not achieved practically.
5. The **diode** operates when a voltage signal is applied across its terminals.
6. The 'ON' state of a diode is achieved by 'Forward biasing' which means that positive or higher potential is applied to the anode and negative or lower potential is applied at the cathode of the diode.
7. A rectifier is a dispositive that ideally transforms the AC input voltage into a DC voltage (voltage is always positive or zero).
8. Half-wave rectifiers only have one side of a waveform, while the full-wave rectifiers use both sides of a waveform.
9. The function of a clipping circuit is to cut off part of an input waveform.
10. A clamping circuit is used to place either the positive or negative peak of a signal at a desired level.
11. A Zener diode is a type of diode that permits current to flow in the forward direction like a normal diode, but also in the reverse direction if the voltage is larger than the rated breakdown voltage or "Zener voltage".
12. An optical diode is an optical component which allows the transmission of light in only one direction.

---

## **2.6 ANSWERS TO CHECK YOUR PROGRESS**

---

1. Diode
2. Switch
3. Biasing
4. Rectifier diodes
5. Shunt
6. Series
7. Combination
8. Zener
9. Isolater

---

## **2.7 TERMINAL QUESTIONS**

---

1. What is a transistor?
2. What is a diode? Explain the working of a diode.
3. Explain the working of a diode.
4. What are the various types of diodes?
5. What is the difference between Clipping and Clamping? Explain.
6. What is a reactifier diode? Explain the functioning of a half wave reactifier.
7. What is Zener Diode? Explain the VI characteristics of the Zener diode.
8. Explain the difference between PNP and NPN transistor.
9. What are optical diodes? Explain the various types of optical diodes.

# UNIT-3

---

## TRANSISTORS

---

### Structure

#### 3.0 Learning Objectives

#### 3.1 Introduction

#### 3.2 Transistors

##### 3.2.1 NPN Transistor

##### 3.2.2 PNP Transistor

#### 3.3 Bipolar Junction Transistors (BJT)

##### 3.3.1 NPN Bipolar Junction Transistor

##### 3.3.2 PNP Bipolar Junction Transistor

##### 3.3.3 Regions of Operation

##### 3.3.4 Configurations

###### 3.3.4.1 The Common Base Configuration

###### 3.3.4.2 The Common Emitter Configuration

###### 3.3.4.3 The Common Collector Configuration

#### 3.4 Field Effect Transistor (FET)

##### 3.4.1 Junction FET (JFET)

##### 3.4.2 Metal-oxide- semiconductor FET (MOSFET)

#### 3.5 analog and digital electronics

#### 3.6 Transistor as a switch

#### 3.7 Summary

#### 3.8 Answers to Check Your progress

#### 3.9 Terminal Questions

---

### 3.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Define a transistor.
- Explain the working principal of PNP and NPN transistors.
- Differentiate between an analog signal and an electronic signal.
- Explain MOSFET
- Explain the functioning of a transistor as a switch.

---

## **3.1 INTRODUCTION**

---

Transistors are active components and are found everywhere in electronic circuits. They are used as amplifiers and switching devices. As amplifiers, they are used in high and low frequency stages, oscillators, modulators, detectors and in any circuit needing to perform a function. In digital circuits they are used as switches.

The most common type of transistor is called bipolar and these are divided into NPN and PNP types. Their construction material is most commonly silicon (their marking has the letter B) or germanium (their marking has the letter A). Original transistors were made from germanium, but they were very temperature-sensitive. Silicon transistors are much more temperature-tolerant and much cheaper to manufacture.

In this unit, we will discuss about transistors in details. We will also discuss various types of transistors and its applications.

---

## **3.2 TRANSISTORS**

---

A transistor is an electronic control device, where an electrical signal input can control another electrical signal. The very name derives from the fact that this control action was seen as an input current flowing into one resistor (base-emitter) causing a larger current in another resistor (collector-emitter). The early name for this device was the transfer-resistor and hence transistor. Transistors can be regarded as a type of switch, as can many electronic components. They are used in a variety of circuits and you will find it in almost all the electronic departments. They are central to electronics and there are two main types; NPN and PNP. Most circuits tend to use NPN. There are hundreds of transistors which work at different voltages but all of them fall into these two categories.



**Figure 34: Transistor**

A transistor is a solid state device made by joining three positive-type and negative-type semiconductors together. In general, all transistors have three pins: base, collector, and emitter.

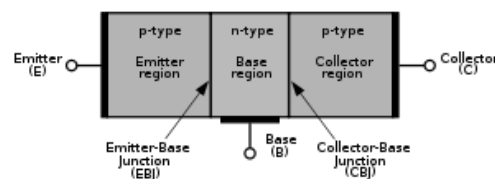
- The *BASE* - which is the lead responsible for activating the transistor.
- The *COLLECTOR* - which is the positive lead.
- The *EMITTER* - which is the negative lead.

A lightly doped region called base is sandwiched between two regions called the emitter and collector respectively. The collector handles large quantities of current, hence its dopant concentration is the highest. The emitter's dopant concentration is slightly lesser, but its area is larger to provide for more current than the collector. The collector region should be heavily doped because electron-hole pairs recombine in that region, while the emitter is not such a region. We can have two varieties in this kind of transistor.

The diagram Figure 36 shows the symbol of an *NPN* transistor. They are not always set out as shown in the diagrams to the left and right, although the `_tab` on the type shown to the left is usually next to the `_emitter`.

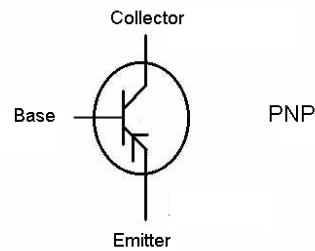
### 3.2.1 NPN Transistor

An NPN transistor is made by joining one positive-type semiconductor in between two negative-type semiconductors.



**Figure 35: Transistor**

Here a lightly doped p-type semiconductor (semiconductor with more holes than electrons) is sandwiched between two well-doped n-type regions. It is like two pn-junctions facing away. An IEEE symbol for the NPN transistor is shown here.



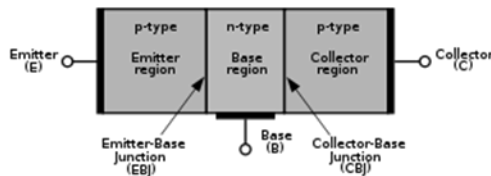
**Figure 36: NPN Symbol**

If the base is at a higher voltage than the emitter, current flows from collector to emitter. Small amount of current also flows from base to emitter. Voltage at base controls amount of current flow through transistor (collector to emitter). The arrow between the base and emitter is in the same direction as current flowing between the base-emitter junction. Power dissipated in the transistor is:  $P=V_{CE}I_C$

Where,  $V_{CE}$  is the voltage between the collector and the emitter and  $I_C$  is the collector current.

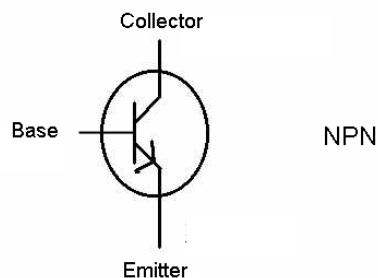
### 3.2.2 PNP Transistor

A PNP transistor is made by sandwiching a negative-type semiconductor in between two positive-type semiconductors.



**Figure 37: PNP Transistor**

If the base is at a lower voltage than the emitter, current flows from emitter to collector. Small amount of current also flows from emitter to base. Voltage at base controls amount of current flow through transistor (emitter to collector).



**Figure 38: PNP Transistor**



**Note:** To find the direction of the current, follow the direction of the arrow.

---

### **3.3 BIPOLAR JUNCTION TRANSISTORS (BJT)**

---

The transistor is the main building block —element|| of electronics. It is a semiconductor device and it comes in two general types: the Bipolar Junction Transistor (BJT) and the Field Effect Transistor (FET). A Bipolar Junction Transistor is a semiconductor device consisting of two P-N Junctions connecting three terminals called the Base, Emitter and Collector terminals. The arrangement of the three terminals affects the current and the amplification of the transistor. The behavior of Bipolar junction transistors is also very different for each circuit configuration. The three different circuit configurations produce different circuit characteristics with regards to input impedance, output impedance and gain. These characteristics affect whether the transistor exhibits voltage gain, current gain or power gain. One of the primary operations of a bipolar junction transistor is to amplify the signal of the current. Bipolar junction Transistors are able to regulate the current so that the current magnitude is proportional to the biased voltage applied at the base terminal of the transistor. The application of Bipolar Junction Transistors can be found in devices that utilize analog circuits such as computers, mobile phones and radio transmitters.

Bipolar Junction Transistors have three semiconductor regions. The three regions are the emitter region (E), base region (B), and the collector region (c) and these regions are differently doped depending on the type of bipolar transistor it is. The two types of bipolar transistors are the PNP Transistor, whose three regions are p type, n type, and p type respectfully, and NPN Transistor, whose regions are n type, p type, and n type respectfully. Both types of transistors have one P-N junction between the collector region and base region and another P-N junction between the base region and emitter region. The base region is always the structure's center connection with the emitter and collector regions connected on either side. Both types of transistors also have the same principle of operation, with the single difference being in the polarity of power and biasing for each type.

Bipolar Junction Transistors ability to amplify a signal, through the regulation of current, allows for the transfer of an input signal from one circuit to another, regardless of the different level of resistance in each circuit. The amount of current flowing through the transistor is proportional to the magnitude of the biasing voltage applied to the base terminal. This allows the transistor to act like a current-controlled switch. Depending on whether the bipolar transistor is PNP or NPN, the controlled current will flow from the collector to the emitter or from the emitter to

the collector while the smaller controlling current will flow from base to emitter or from emitter to base respectively.

The transistor contains a maximum allowed current that is able to restrict the amount of current as it passes from terminal to terminal. Depending on the order of the terminals in the transistor, the transistor will act as either a conductor or an insulator when in the presence of a controlled current. This ability to change between these two states, insulator or conductor, enables the transistor to act like a switch or as an amplifier of small amplitude signals applied to the base depending on the structure and order of the three semiconductor regions.

Bipolar Junction Transistors contain three doped extrinsic semiconductor regions each connected to a circuit. The transistor is not symmetrical due to the different doping ratios of the emitter, collector and base regions. The base region consists of a lightly doped materials that exhibits high resistivity. The base is located between the heavily doped emitter region and the lightly doped collector region. The collector engulfs the emitter region which eliminates the ability for electrons injected into the base region to escape the base region without being collected. The emitter region is heavily doped to increase the current gain of the transistor.

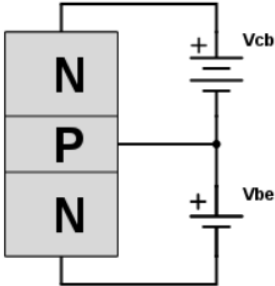
For high current gain, a high ratio of carriers injected by the emitter to those injected by the base is needed. Increasing the emitter injection efficiency results in the majority of the carriers injected into the emitter-base junction coming from the emitter region. The high doping ratio of the emitter and collector regions, also means the collector-base junction is reverse biased. The collector-base junction can therefore have a high magnitude reverse bias voltage applied before the junction breaks down. For the transistor as a whole, the fundamental difference between the NPN Transistor and the PNP Transistor is current directions and voltage polarities of the transistor junctions. Making sure these two are always opposite each other ensures the transistors are properly biased.

### **3.3.1 NPN Bipolar Junction Transistor**

A NPN Bipolar Junction Transistor has a P-doped semiconductor base in between an N-doped emitter and N-doped collector region. NPN bipolar transistors are the highest used bipolar transistors due to the ease of electron mobility over electron hole mobility.

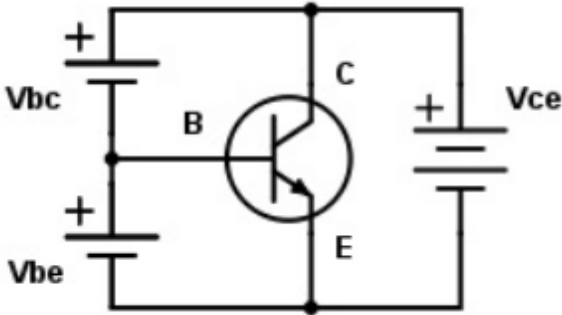
For this type of transistor, large magnitude collector and emitter currents get produced through the amplification of a small current which enters through the base. This small current only gets amplified when the transistor becomes active. In this active state, a positive potential difference is found between both the base region to the collector region and the emitter region to the base region which results in current that gets

carried by electrons, between the collector and emitter regions. The construction and terminal voltages for a NPN Transistor are shown in Figure 39 below.



**Figure 39: NPN Transistor schematic**

For a bipolar NPN transistor to conduct the Collector is always more positive with respect to both the Base and the Emitter. The voltage between the Base and Emitter ( $V_{BE}$ ), is positive at the Base and negative at the Emitter. The Base terminal is always positive with respect to the Emitter. Another way to display a NPN Transistor is shown in Figure 40 below.



**Figure 40: NPN Bipolar Transistor circuit**

**3.3.2 PNP Bipolar Junction Transistor**

A PNP Bipolar Junction Transistor has an N-doped semiconductor base in between a P-doped emitter and P-doped collector region. The PNP Transistor 54

has very similar characteristics to the NPN Transistor, with the difference being the biasing of the current and voltage directions are reversed. For PNP Transistors, current enters into the transistor through the emitter terminal. A small current leaving the base is amplified in the collector output. The emitter-base region is forward biased so electric field and carriers will be generated. The voltage sources are connected to a PNP transistor are as shown in Figure 41 and Figure 42 below.

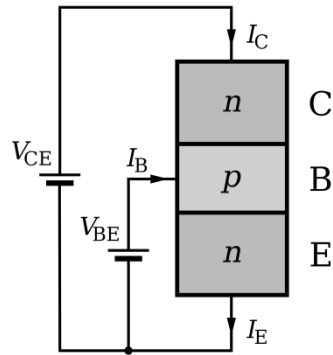


Figure 41: PNP transistor schematic 26

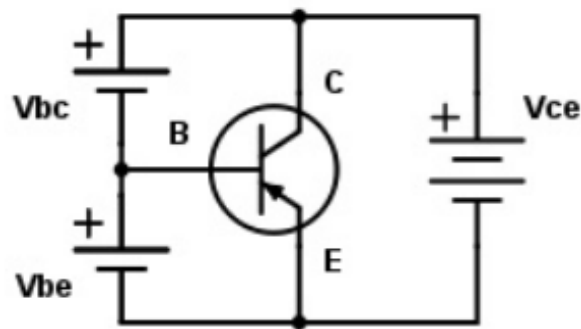


Figure 42: PNP Transistor circuit

The voltage between the Base and Emitter ( $V_{BE}$ ), is now negative at the Base and positive at the Emitter. The Base terminal is always biased negative with respect to the Emitter while. The Emitter is positive with respect to the Collector ( $V_{CE}$ ). The reverse biased collector base part has generated holes. Due to the electric field, carriers or electrons get pulled by the holes. For a PNP transistor to conduct, the Emitter is always more positive with respect to both the Base and the Collector.

### 3.3.3 Regions of Operation

Bipolar transistors have four distinct regions of operation. These regions are defined by the biases placed on the junction of the Bipolar Junction Transistor.

- **Cutoff:** The Cutoff region is when the transistor is inactive due to minimal current being passed through the transistor, which makes the transistor appear as an open circuit. Both  $V_{BE}$  and  $V_{BC}$  are reverse biased so all depletion region edges exhibit small minority carrier densities. This region has biasing conditions opposite of saturation.
- **Forward-active:** The Forward-active region occurs when the transistor is in its active state which allows the transistor to amplify the voltage variations present on the base. With the base-emitter junction is forward biased and the base-collector junction is

reversed biased, the transistor can amplify voltage because the collector to emitter voltage is greater than the base to emitter voltage and is also in between the cutoff and saturation states. The output current is proportional to the base current and can be extracted at the collector.

- **Reverse-active:** The Reverse-active region occurs when the transistor is in its active state but the maximum current gain in the reverse active mode is much smaller than the forward active mode. The biasing conditions are reversed so that the base collector junction is forward biased and the base emitter junctions is reverse biased, which switches the roles of the collector and emitter regions. The base contains a much lower reverse bias voltage than in the forward-active region.
- **Saturation:** The saturation region allows the transistor to conduct current from the emitter to the collector. With both the base collector junction and the base emitter junction forward-biased, the base current is so strong it exceeds the magnitude at which it can increase the collector current flow. As a result, the circuit between the collector and emitter terminals appears to have short circuited due to the over saturation of current.

### 3.3.4 Configurations

There are three methods of connection for a Bipolar Junction Transistor within an electronic circuit. The Common Base configuration, Common Emitter configuration and Common Collector configuration all respond differently to the circuit's input signal, thus varying the characteristics of each configuration.

#### 3.3.4.1 The Common Base Configuration

The common base configuration has a strong high frequency response which is good for single stage amplifier circuits. However, is not very common due to its low current gain characteristics and low input impedance. The input signal gets applied between the base and emitter terminals while the output signal is

taken from between the base and collector terminal. For this to occur, the base terminal has to be grounded so the reference voltage is a fixed amount. The Common Base Configuration is shown below.

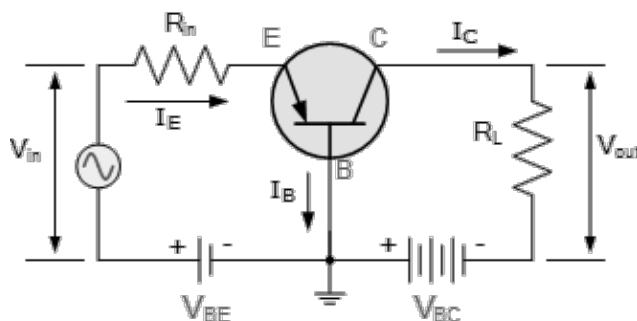
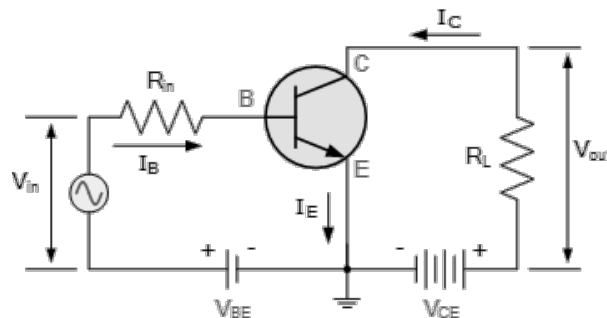


Figure 213: The Common Base Transistor Circuit

This type of amplifier configuration is a non-inverting voltage amplifier circuit. The configuration has a resistance gain due to the ratio between the load resistance ( $R_{load}$ ) in series with the collector and the  $R_{in}$  resistor. The input current flowing into the emitter is the sum of both the base current and collector current respectively therefore, the collector current output is less than the emitter current input resulting in a current gain. Its input characteristics represent that of a forward biased diode

### 3.3.4.2 The Common Emitter Configuration

The common emitter amplifier configuration produces the highest current and power gain of all the three bipolar transistor configurations which is why this type of configuration is the most commonly used circuit for transistor based amplifiers. The input signal applied between the base and emitter is small due to the forward biasing of the PN junction and the output taken from between the collector and emitter is large due to the reverse biased PN junction. This is mainly because the input impedance is small as it is connected to a forward biased PN-junction, while the output impedance is large as it is taken from a reverse biased PN-junction. However, its voltage gain is much lower. The Common Emitter configuration is shown below.

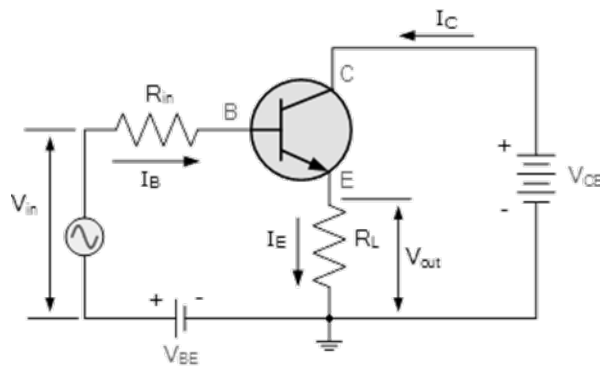


**Figure 44 : The Common Emitter Amplifier Circuit**

The common emitter configuration is an inverting amplifier circuit. Therefore the output signal is out-of-phase with the input voltage signal.

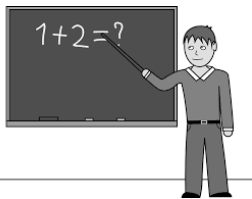
### 3.3.4.3 The Common Collector Configuration

The common collector configuration is very useful for impedance matching applications because of the very large ratio of input impedance to output impedance. The configuration has the input signal directly connected to the base. With the emitter region in series with the load resistor, the current flowing through the load resistance is the same value as the emitter current. This is why the output is taken from the emitter load and the current gain of the configuration is approximately equal to the  $\beta$  value of the transistor.



**Figure 45: The Common Collector Transistor Circuit**

This type of bipolar transistor configuration is a non-inverting circuit in that the signal voltages of  $V_{in}$  and  $V_{out}$  are in-phase. The load resistance receives both the base and collector currents which results in a large current gain as well as providing good current amplification with very little voltage



## Check Your Progress

1. A \_\_\_\_\_ d negative-type semiconductors together.
2. If the base is at a lower voltage than the emitter, current flows from emitter to\_\_\_\_\_ .
3. A \_\_\_\_\_ is a semiconductor device consisting of two P-N Junctions connecting three terminals called the Base, Emitter and Collector terminals.
4. Bipolar Junction Transistors have \_\_\_\_\_ semiconductor regions.
5. Bipolar transistors have \_\_\_\_\_ distinct regions of operation.
6. There are \_\_\_\_\_ methods of connection for a Bipolar Junction Transistor within an electronic circuit.

---

## 3.4 Field Effect Transistor (FET)

---

The field-effect transistor (FET), sometimes called a unipolar transistor, uses either electrons (N-channel FET) or holes (P-channel FET) for conduction. The four terminals of the FET are named source, gate, drain, and body (substrate). On most FETs the body is connected to the source inside the package and this will be assumed for the following description.

A voltage applied between the gate and source (body) controls the current flowing between the drain and source. As the gate/source voltage ( $V_{gs}$ ) is increased the drain/source current ( $I_{ds}$ ) increases roughly parabolically ( $I_{ds} \propto V_{gs}^2$ ). In FETs the drain/source current flows

through a conducting channel near the gate. This channel connects the drain region to the source region. The channel conductivity is varied by the electric field generated by the voltage applied between the gate/source terminals. In this way the current flowing between the drain and source is controlled.

It is a type of transistor commonly used for weak-signal amplification. The device can amplify analog or digital signals. In the FET, current flows along a semiconductor path called the channel. At one end of the channel, there is an electrode called the source and at the other end of the channel, there is an electrode called the drain. A small change in gate voltage can cause a large variation in the current from the source to the drain.

Field-effect transistors exist in two major classifications

- (i) *Junction FET (JFET)*
- (ii) *Metal-oxide- semiconductor FET (MOSFET)*.

### 3.4.1 Junction FET (JFET)

Metal–semiconductor FETs (MESFETs) are JFETs in which the reverse biased PN junction is replaced by a metal–semiconductor Schottky-junction. These, and the HEMTs (high electron mobility transistors, or HFETs), in which a two-dimensional electron gas with very high carrier mobility is used for charge transport, are especially suitable for use at very high frequencies (microwave frequencies; several GHz).

JFETs are exclusively voltage-controlled in that they do not need a biasing current. Electric charge flows through a semiconducting channel between source and drain terminals. By applying a reverse bias voltage to a gate terminal, the channel is "pinched", so that the electric current is impeded or switched off completely. A JFET is usually on when there is no potential difference between its gate and source terminals. If a potential difference of the proper polarity is applied between its gate and source terminals, the JFET will be more resistive to current flow, which means less current would flow in the channel between the source and drain terminals. Thus, JFETs are sometimes referred to as depletion-mode devices.

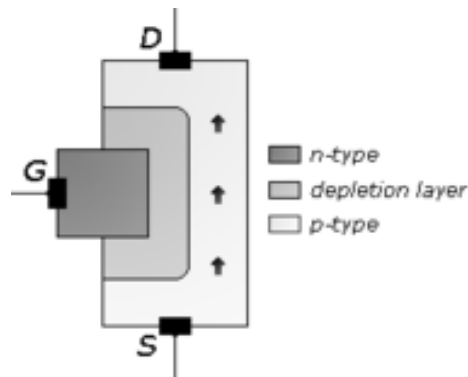
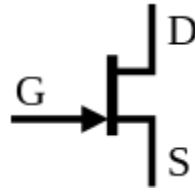


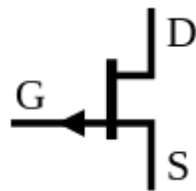
Figure 46 N typed channel JFET



JFETs can have an n-type or p-type channel. In the n-type, if the voltage applied to the gate is less than that applied to the source, the current will be reduced (similarly in the p-type, if the voltage applied to the gate is greater than that applied to the source). A JFET has a large input impedance (sometimes on the order of 10<sup>10</sup> ohms), which means that it has a negligible effect on external components or circuits connected to its gate.



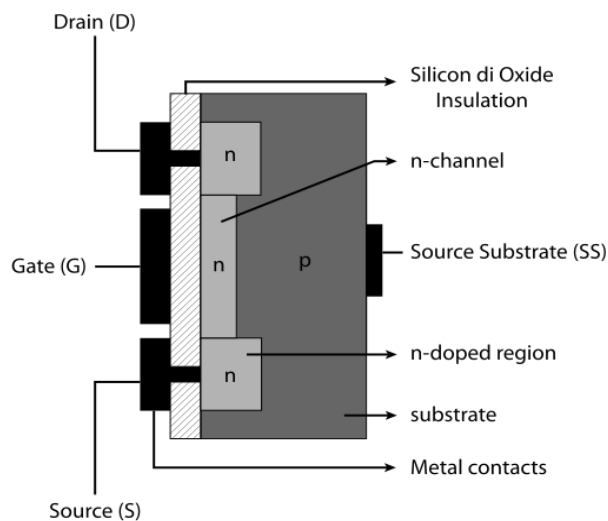
**Figure 47: p typed channel JFET**



**Figure 48 N typed channel JFET**

### 3.4.2 Metal-oxide- semiconductor FET (MOSFET)

The IGFET is more commonly known as metal–oxide–semiconductor FET MOSFET, from their original construction as a layer of metal (the gate), a layer of oxide (the insulation), and a layer of semiconductor. Unlike IGFETs, the JFET gate forms a PN diode with the channel which lies between the source and drain. Functionally, this makes the N-channel JFET the solid state equivalent of the vacuum tube triode which, similarly, forms a diode between its grid and cathode. Also, both devices operate in the depletion mode, they both have a high input impedance, and they both conduct current under the control of an input voltage.



**Figure 49: The structure of an N-channel MOSFET and its different parts**

MOSFETs have two regions, called the source and drain which are heavily doped. These are embedded in a substrate, which is doped the other way. The gap between the source and drain regions, which spans the substrate, is where the current will eventually flow. A layer of insulating oxide is placed over this gap (the channel), and on top of that, a gate contact, usually made of polysilicon.

The way a MOSFET works is to modify a thin layer of this gap, or channel, using an electric field that propagates through the insulation. This modification could be either increasing the current carrying capacities of the channel, or reducing it. Thus, we have two kinds of devices - enhancement mode MOSFETs and depletion mode MOSFETs. Depending on what kind of silicon is in the channel, a MOSFET can be p-channel or n-channel. A p-channel MOSFET has p-type silicon in the channel when 'on.'

---

## 3.5 ANALOG AND DIGITAL ELECTRONICS

---

The field of electronics can be divided into two branches: *analog electronics* and *digital electronics*.

- a. **Analog electronics:** It is the branch that deals with such things as *resistors* and *capacitors*. In analog electronics, voltages and currents can take on practically any value (to within the physical limitations of the equipment or electrical components of the circuit being used). Thus, in an analog circuit, information tends to be conveyed by the magnitude of the voltage or current signal.
- b. **Digital electronics:** It is that branch of electronics that deals with components like *logic gates*, counters, flip-flops, or, more generically, the *digital* integrated circuit chip (often simply called *circuit chips*). In digital electronics, voltages can take on only one of two possible values (to within limited ranges). This means that information in a digital circuit cannot be conveyed by the magnitude of the voltage signal, but rather must be conveyed using the concepts of *binary logic*.

Analog and digital signals are used to transmit information, usually through electric signals. In both these technologies, the information, such as any audio or video, is transformed into electric signals. The difference between analog and digital technologies is that in analog technology, information is translated into electric pulses of varying amplitude. In digital technology, translation of information is into binary format (zero or one) where each bit is representative of two distinct amplitudes.

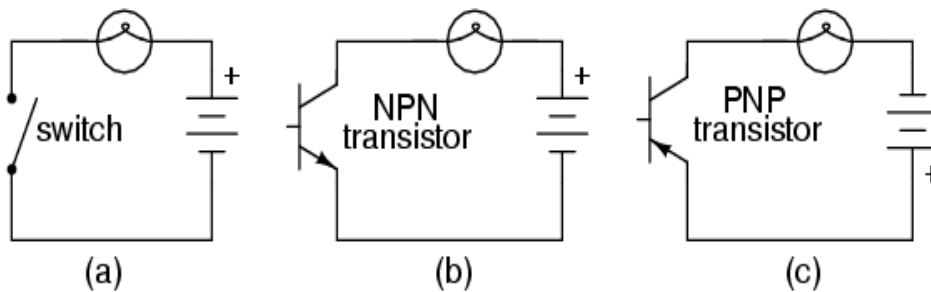
---

## 3.6 TRANSISTOR AS A SWITCH

---

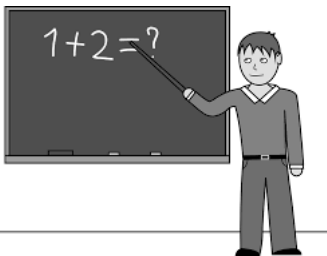
Transistors are commonly used as electronic switches, both for high-power applications such as switched mode power supply and for low-power applications such as logic gates. Because a transistor's collector

current is proportionally limited by its base current, it can be used as a sort of current-controlled switch. A relatively small flow of electrons sent through the base of the transistor has the ability to exert control over a much larger flow of electrons through the collector. Suppose we had a bulb and we just want to implement a switch to switch it on and off. This type of circuit is extremely simple to implement and is shown in Figure 50(a). For the sake of illustration, let's insert a transistor in place of the switch to show how it can control the flow of electrons through the lamp. Remember that the controlled current through a transistor must go between collector and emitter. Since it is the current through the lamp that we want to control, we must position the collector and emitter of our transistor where the two contacts of the switch were. We must also make sure that the lamp's current will move against the direction of the emitter arrow symbol to ensure that the transistor's junction bias will be correct as in Figure below Figure 50(b)



**Figure 50: (a) mechanical switch, (b) NPN transistor switch, (c) PNP transistor switch**

A PNP transistor could also have been chosen for the job. Its application is shown in Figure above **Error! Reference source not found.** (c). The choice between NPN and PNP is really arbitrary. All that matters is that the proper current directions are maintained for the sake of correct junction biasing (electron flow going against the transistor symbol's arrow).



## Check Your Progress

1. The \_\_\_\_\_ type of amplifier configuration is a non-inverting voltage amplifier circuit..
2. \_\_\_\_\_ are exclusively voltage-controlled in that they do not need a biasing current.
3. The IGFET is more commonly known as \_\_\_\_\_.

---

## 3.7 SUMMARY

---

1. A transistor is an electronic control device, where an electrical signal input can control another electrical signal.
2. An NPN transistor is made by joining one positive-type semiconductor in between two negative-type semiconductors.
3. A PNP transistor is made by sandwiching a negative-type semiconductor in between two positive-type semiconductors.
4. A Bipolar Junction Transistor is a semiconductor device consisting of two P-N Junctions connecting three terminals called the Base, Emitter and Collector terminals.
5. In common base configuration, the input signal gets applied between the base and emitter terminals while the output signal is taken from between the base and collector terminal.
6. The common emitter amplifier configuration produces the highest current and power gain of all the three bipolar transistor configurations which is why this type of configuration is the most commonly used circuit for transistor based amplifiers.
7. The common collector configuration is very useful for impedance matching applications because of the very large ratio of input impedance to output impedance.
8. The field-effect transistor (FET), sometimes called a unipolar transistor, uses either electrons (N-channel FET) or holes (P-channel FET) for conduction.
9. Metal–semiconductor FETs (MESFETs) are JFETs in which the reverse biased PN junction is replaced by a metal–semiconductor Schottky-junction.
10. Analog and digital signals are used to transmit information, usually through electric signals.
11. In both these technologies, the information, such as any audio or video, is transformed into electric signals.
12. The difference between analog and digital technologies is that in analog technology, information is translated into electric pulses of varying amplitude.
13. In digital technology, translation of information is into binary format (zero or one) where each bit is representative of two distinct amplitudes.

---

## **3.8 ANSWERS TO CHECK YOUR PROGRESS**

---

1. Transistor
2. Collector
3. Bipolar Junction Transistor
4. Three
5. Four
6. Three
7. Common base configuration
8. JFETs
9. MOSFET



# UNIT-4

---

## INTEGRATED CIRCUIT

---

### Structure

#### 4.0 Learning Objectives

#### 4.1 Introduction

#### 4.2 Integrated Circuit

#### 4.3 Multivibrators

##### 4.3.1 Astable Multivibrator

##### 4.3.2 Monostable Multivibrator

##### 4.3.3 Bistable Multivibrator

#### 4.4 Counters

##### 4.4.1 Asynchronous counters(or ripple counters)

##### 4.4.2 Synchronous Counters

##### 4.4.2.1 Synchronous Decade Counter

#### 4.5 Summary

#### 4.6 Answers to check your progress

#### 4.7 Terminal Questions

---

### 4.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Define an Integrated Circuit(IC).
- Classify ICs.
- Define a Multivibrator.
- Know the various types of Multivibrator.
- Define Counter.
- Differentiate between synchronous and asynchronous counts.
- Explain the working of Synchronous Decade Counter.

---

## 4.1 INTRODUCTION

---

Integrated Circuits<sup>29</sup> play a very important part in electronics. Most are specially made for a specific task and contain up to thousands of transistors, diodes and resistors. Special purposes IC's such as audio-amplifiers, FM radios, logic blocks, regulators and even a whole micro computers in the form of a micro controller can be fitted inside a tiny package.

Multivibrator is an electronic circuit used to implement a variety of simple two-state systems such as oscillators, timers, and flip-flops.

A counter is a device that generates some patterned binary value depending on a clock or some other pulsed input.

In this unit we will discuss about Integrated Circuits, Multivibrators and counters. We will also discuss their types and applications in details.

---

## 4.2 INTEGRATED CIRCUIT

---

An integrated circuit<sup>30</sup> is a thin slice of silicon or sometimes another material that has been specially processed so that a tiny electric circuit is etched on its surface. The circuit can have many millions of microscopic individual elements, including transistors, resistors, capacitors, and conductors, all electrically connected in a certain way to perform some useful function.

The first integrated circuits were based on the idea that the same process used to make clusters of transistors on silicon wafers might be used to make a functional circuit, such as an amplifier circuit or a computer logic circuit. Slices of the semiconductor materials silicon and germanium were already being printed with patterns, the exposed surfaces etched with chemicals, and then the pattern removed, leaving dozens of individual transistors, ready to be sliced up and packed individually. But wires, a few resistors and capacitors might later connect those same transistors to make a circuit. Why not do the whole thing at one time on that slice of silicon?

The idea occurred to a number of inventors at the same time, but the first to accomplish it were Jack Kilby of Texas Instruments and Robert Noyce of Fairchild Semiconductor Incorporated. The idea caught on like wildfire because the integrated circuit had many of the advantages that had made the transistor attractive earlier. These advantages included small size, high

reliability, low cost, and small power consumption. However, these circuits were difficult to make because if one component of the chip was faulty, the whole chip was ruined. As engineers got better and better at squeezing more and more transistors and other components onto a single chip, the problems of actually making these chips increased. When the



transistors were shrunk down to microscopic size, even the smallest bit of dust could ruin the chip. That's why today, chips are made in special "clean rooms" where workers wear the "bunny suits" that we often see on TV.

Compared to the original integrated circuit, which was a simple device with just a few components, the number of components on today's integrated circuits is amazing. In the 1960s, an engineer named Gordon Moore predicted that the number of elements on a chip would double every year (later revised to every two years) into the foreseeable future. "Moore's Law" has held true so far. By the beginning of the twenty-first century, the Intel Pentium chip had over 100 million transistors on it, with the total number of components including resistors, capacitors, and conductors being even larger.



**Figure 51: Integrated Circuit31**

Integrated circuit finds its application to almost every electronic devices including microprocessors, audio and video equipment, and automobiles. Based

on the number of electronic components fabricated on a single chip, and Integrated circuits are broadly classified into following categories:

- **SSI (small-scale integration):** Up to 100 electronic components per chip
- **MSI (medium-scale integration):** From 100 to 3,000 electronic components per chip

- **LSI (large-scale integration):** From 3,000 to 100,000 electronic components per chip
- **VLSI (very large-scale integration):** From 100,000 to 1,000,000 electronic components per chip
- **ULSI (ultra large-scale integration):** More than 1 million electronic components per chip

Classification of IC's is summarized in the table below:

**Table 5: Generations of ICs**

Name	Period	Numbers of transistors on each chip (approximately)
SSI (Small-Scale Integration)	early 1960s	one chip contains only a few transistors
MSI (Medium-Scale Integration)	late 1960s	hundreds of transistors on each chip
LSI (Large-Scale Integration)	mid 1970s	tens of thousands of transistors per chip
VLSI (Very Large-Scale Integration)	late 20th century	about hundreds of thousands of transistors ~ several billion transistors
ULSI (Ultra-Large Scale Integration)	21st century	more than 1 million transistors

---

## 4.3 MULTIVIBRATORS

---

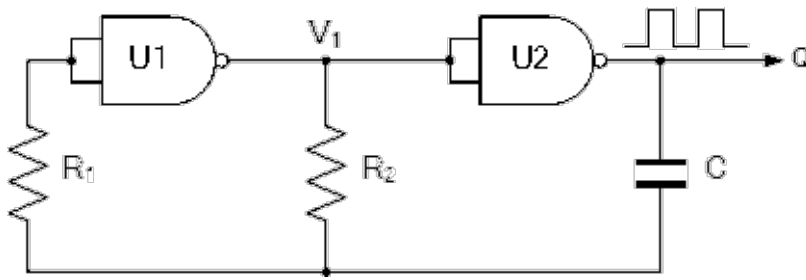
It is an electronic circuit that switches rapidly between two or more states by means of positive feedback. Its operation is similar to a pendulum of a wall.

clock which continually oscillates between two states. Based on the type of clock pulse output generated by a multivibrator, it can be classified into three types : –

- Astable multivibrator
- Monostable multivibrator
- Bistable multivibrator

### 4.3.1 Astable Multivibrator

It is a circuit which is not stable in any of the states, which produces a train of square wave pulses at output at a fixed frequency. This makes it an ideal candidate for timing and clock pulse applications.



**Figure 52: Monostable Multivibrator implemented using NAND gate**

Suppose that initially the output from the NAND gate U2 is HIGH at logic level "1", then the input must therefore be LOW at logic level "0" (NAND gate principles) as will be the output from the first NAND gate U1. Capacitor, C is connected between the output of the second NAND gate U2 and its input via the timing resistor, R2. The capacitor now charges up at a rate determined by the time constant of R2 and C.

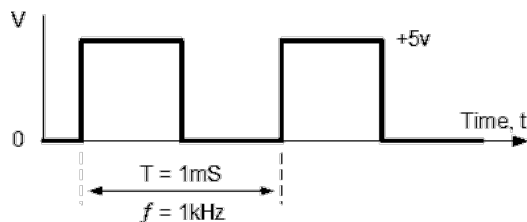
As the capacitor, C charges up, the junction between the resistor R2 and the capacitor, C, which is also connected to the input of the NAND gate U1 via the stabilizing resistor, R2 decreases until the lower threshold value of U1 is reached at which point U1 changes state and the output of U1 now becomes HIGH. This causes NAND gate U2 to also change state as its input has now changed from logic "0" to logic "1" resulting in the output of NAND gate U2 becoming LOW, logic level "0".

Capacitor C is now reverse biased and discharges itself through the input of NAND gate U1. Capacitor, C charges up again in the opposite direction determined by the time constant of both R2 and C as before until it reaches the upper threshold value of NAND gate U1. This causes U1 to change state and the cycle repeats itself over again. Then, the time constant for a NAND gate Astable Multivibrator is given as  $T = 2.2RC$  in seconds with the output frequency given as  $f = 1/T$ . 70

For example: if resistor  $R2 = 10k\Omega$  and the capacitor  $C = 45nF$ , then the oscillation frequency will be given as:

$$f = 1/T = 1/(2.2RC) = 1/(2.2 \times 10 \times 10^3 \times 45 \times 10^{-9}) = 1kHz$$

then the output frequency is calculated as being 1kHz, which equates to a time constant of 1mS so the output waveform would look like:

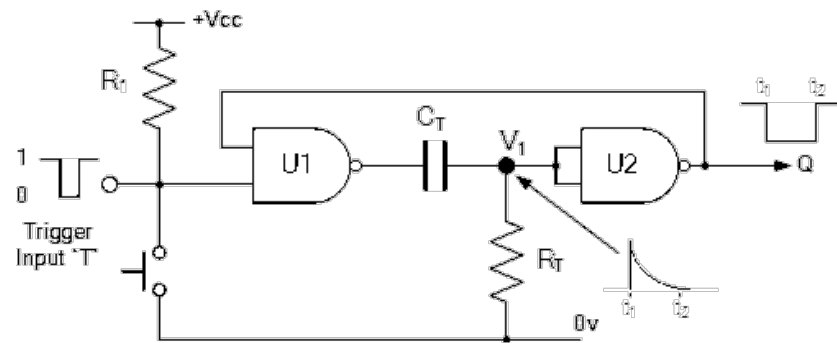


**Figure 53: Output waveform of Astable multivibrator**

### 4.3.2 Monostable Multivibrator

As the name suggest, in monostable multivibrator, one of the states is stable, but the other state is unstable. A trigger causes the circuit to enter the unstable state. After entering the unstable state, the circuit will return to the stable state after a set time. Such a circuit is useful for creating a timing period of fixed duration in response to some external event. This circuit is also known as a one shot. Monostable multivibrators generate a single output pulse, either "high" or "low", when a suitable external trigger signal or pulse T is applied. This trigger pulse signal initiates a timing cycle which causes the output of the monostable to change state at the start of the timing cycle, (  $t_1$  ) and remain in this second state until the end of the timing period, (  $t_1$  ) which is determined by the time constant of the timing capacitor,  $C_T$  and the resistor,  $R_T$ .

The monostable multivibrator now stays in this second timing state until the end of the RC time constant and automatically resets or returns itself back to its original (stable) state. Then, a monostable circuit has only one stable state.



**Figure 54: Monostable Multivibrator implemented using NAND gate**

Suppose that initially the trigger input T is held HIGH at logic level "1" by the resistor R1 so that the output from the first NAND gate U1 is LOW at logic level "0", (NAND gate principals). The timing resistor  $R_T$  is connected to a voltage level equal to logic level "0", which will cause the capacitor,  $C_T$  to be discharged. The output of U1 is LOW, timing capacitor  $C_T$  is completely discharged therefore junction  $V_1$  is also equal to "0" resulting in the output from the second NAND gate U2, which is connected as an inverting NOT gate will therefore be HIGH.

The output from the second NAND gate, ( U2 ) is fed back to one input of U1 to provide the necessary positive feedback. Since the junction  $V_1$  and the output of U1 are both at logic "0" no current flows in the capacitor  $C_T$ . This results in the circuit being **Stable** and it will remain in this state until the trigger input changes.

If a negative pulse is now applied either externally or by the action of the push-button to the trigger input of the NAND gate U1, the output of U1

will go HIGH to logic "1" (NAND gate principles). Since the voltage across the capacitor cannot change instantaneously (capacitor charging principals) this will cause the junction at V1 and also the input to U2 to also go HIGH, which in-turn will make the output of the NAND gate U2 change LOW to logic "0" The circuit will now remain in this second state even if the trigger input pulse T is removed. This is known as the **Meta-stable** state.

The voltage across the capacitor will now increase as the capacitor CT starts to charge up from the output of U1 at a time constant determined by the resistor/capacitor combination. This charging process continues until the charging current is unable to hold the input of U2 and therefore junction V1 HIGH. When this happens, the output of U2 switches HIGH again, logic "1", which in turn causes the output of U1 to go LOW and the capacitor discharges into the output of U1 under the influence of resistor RT. The circuit has now switched back to its original stable state.

Thus for each negative going trigger pulse, the monostable multivibrator circuit produces a LOW going output pulse. The length of the output time period is determined by the capacitor/resistor combination (RC Network) and is given as the **Time Constant**  $T = 0.69RC$  of the circuit in seconds. Since the input impedance of the NAND gates is very high, large timing periods can be achieved.

### 4.3.3 Bistable Multivibrator

It is a multivibrator in which the circuit is stable in either state. The circuit can be flipped from one state to the other by an external event or trigger. It is basically a SR flip-flop with the addition of an inverter or NOT gate to provide the necessary switching function. As with flip-flops, both states of a bistable 72

multivibrator are stable, and the circuit will remain in either state indefinitely. This type of multivibrator circuit passes from one state to the other "only" when a suitable external trigger pulse T is applied and to go through a full "SET-RESET" cycle two triggering pulses are required. This type of circuit is also known as a "Bistable Latch", "Toggle Latch" or simply "T-latch".

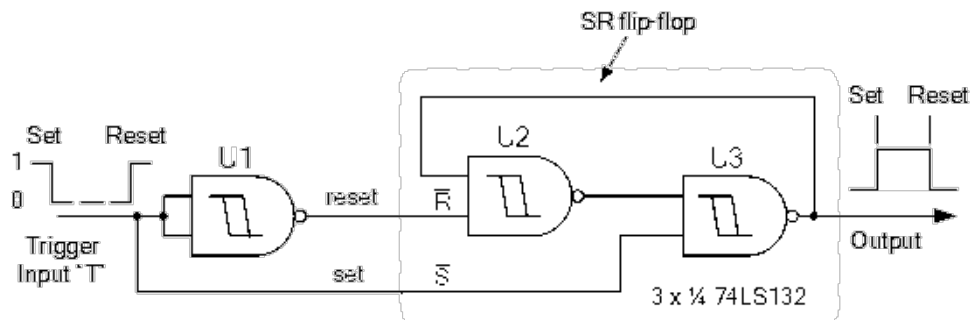
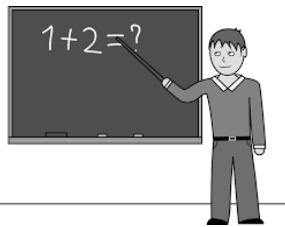


Figure 55: A Bistable Multivibrator implemented with NAND Gate

The simplest way to make a **Bistable Latch** is to connect together a pair of Schmitt NAND gates to form a SR latch as shown above. The two NAND gates, U2 and U3 form the bistable which is triggered by the input NAND gate, U1. When the input pulse goes "LOW" the bistable latches into its "SET" state, with its output at logic level "1", until the input goes "HIGH" causing the bistable to latch into its "RESET" state, with its output at logic level "0". The output of a bistable multivibrator will stay in this "RESET" state until another input pulse is applied and the whole sequence will start again. Then a **Bistable Latch** or "Toggle Latch" is a two-state device in which both states are either positive or negative, (logic "1" or logic "0") are stable.



## Check Your Progress

1. ULSI stands for \_\_\_\_\_.
2. The operation of a \_\_\_\_\_ is similar to a pendulum of a wall clock which continually oscillates between two states.
3. Based on the type of clock pulse output generated by a multivibrator, it can be classified into \_\_\_\_\_ types.
4. \_\_\_\_\_ multivibrator is an ideal circuit for timing and clock pulse applications.
5. In \_\_\_\_\_ multivibrator, one of the states is stable, but the other state is unstable.

---

## 4.4 COUNTERS

---

Counter is a digital sequential circuit which follows a predetermined sequence of state. Counters can be classified into two categories:

- Asynchronous counters(or ripple counters)
- Synchronous counters

Before proceeding to the details of the counters, we must understand the difference between level triggered and edge triggered circuits. A circuit is said to be level trigger if the input signal is sampled when the clock signal is either HIGH or LOW. Example: Latch.

A circuit is said to be edge triggered if the input signal is sampled at the RISING EDGE or FALLING EDGE of the clock signal. Example: Flipflop.

### 4.4.1 Asynchronous counters(or ripple counters)

This counter is called asynchronous because not all flip-flops are driven by the same clock. A simple ripple counter is shown below in the fig. 30 which is implemented using JK Flip-flop. Both the inputs of the JK flip flops are tied to logic —HIGH|| of 1 state, since it will toggle state on the clock edge. For simplicity, the falling edge is used and assuming all flip flops start in reset state.

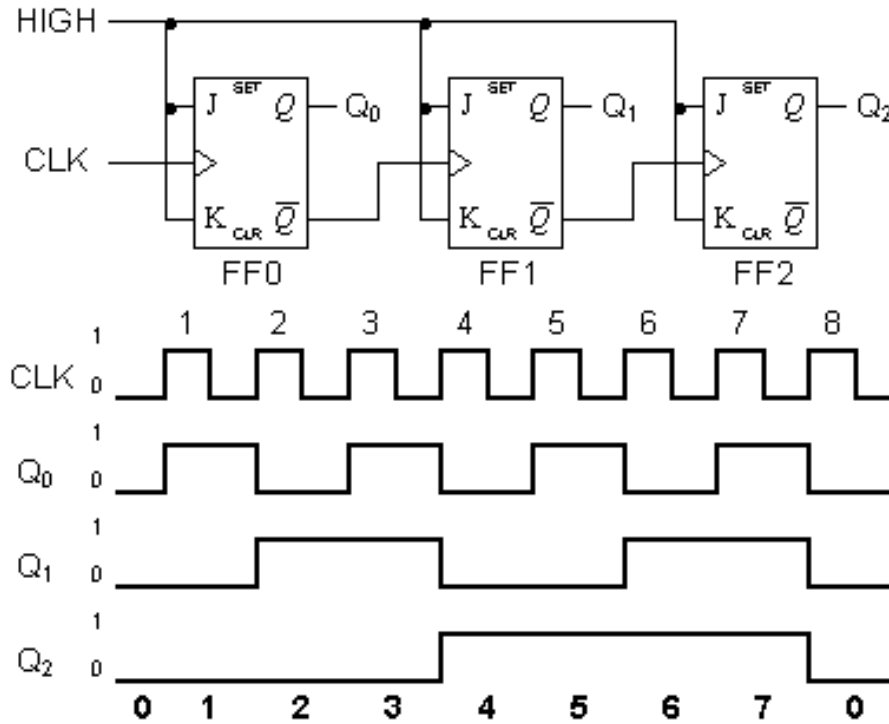


Figure 56: Asynchronous Counter

Both the inputs of FF0, the first Flip Flop in the sequence are hooked to logic 1 and the clock input is attached to the clock source. When it transitions from low to high to low again (this last transition generating a falling edge), in other words, when the input pulses, the FF changes to Set. Since the First FF's output has not made a falling edge transition, the second FF remains Reset. When another pulse appears at the input, the first FF changes to Reset again, creating a falling edge at its output, which triggers the second FF to transition to the Set state.

Another pulse, the first FF changes to Set; No falling edge at its output, the second FF keeps its state. Yet another pulse (Now four if you have been keeping the count), the first FF goes back to Reset, producing a falling edge; the second FF also goes back to Reset, producing a falling edge at its output that will trigger a third FF and making it Set. As you can see, when all the transitions have occurred, the counter ends up with the count of input pulses it has received, representing them in a binary number.

### 4.4.2 Synchronous Counters

In case of synchronous counters, the clock inputs of all flip-flops receive the common clock pulses. Thus, all the flip-flops change state simultaneously (in parallel). The circuit below in fig. 31 is a 3-bit synchronous counter. The J and K inputs of FF0 are connected to HIGH. FF1 has its J and K inputs connected to the output of FF0, and the J and K inputs of FF2 are connected to the output of an AND gate that is fed by the outputs of FF0 and FF1.

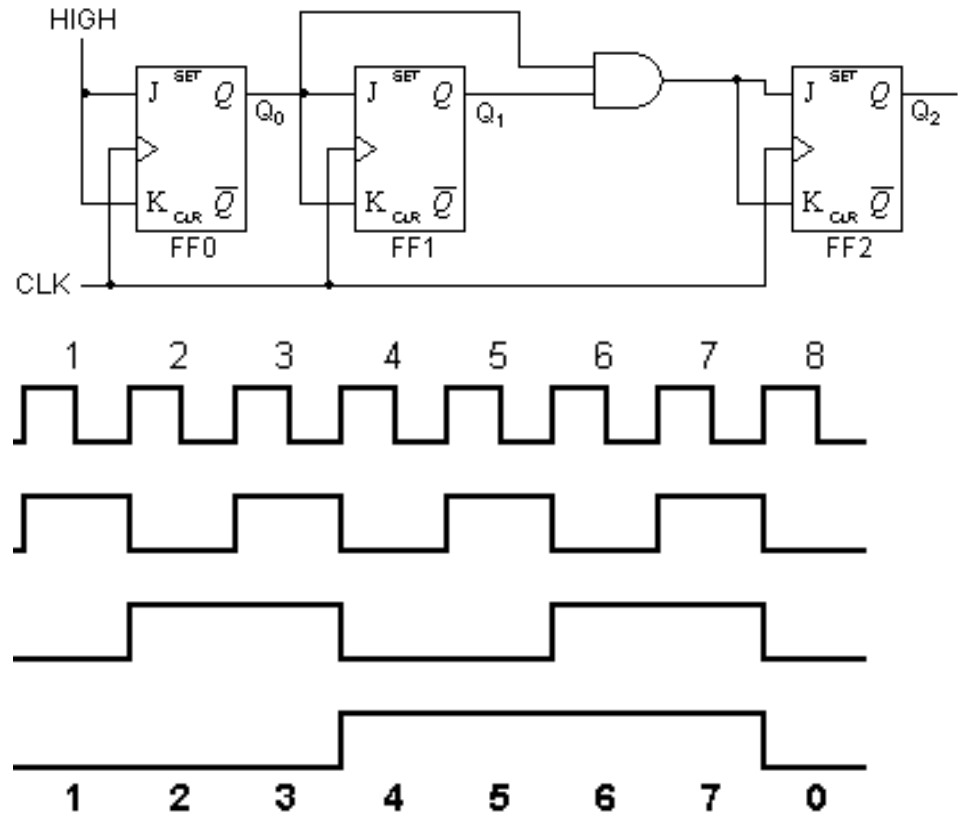


Figure 57: 3-bit Synchronous counter

#### 4.4.2.1 Synchronous Decade Counter

Synchronous decade counter counts from 0 to 9 and then recycles to 0 again. This is done by forcing the 1010 state back to the 0000 state. This so called truncated sequence can be constructed by the following circuit.

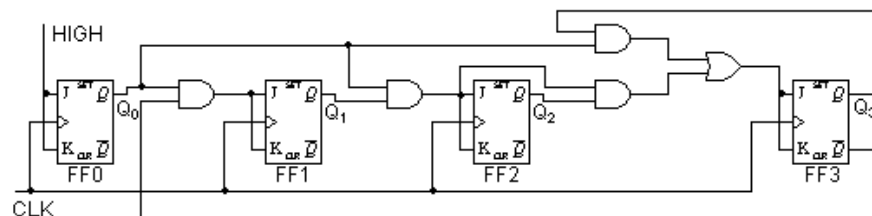


Figure 58: Synchronous Decade Counter

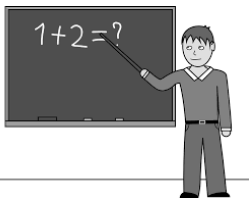


From Table 6, we observe that:

The output of the first flip flop FF0, i.e. Q0, toggles on each clock pulse. The output of FF1, i.e. Q1 changes on the next clock pulse each time Q0=1 and Q3=0. And the output of FF2, Q2 changes on the next clock pulse each time Q0=Q1=1. The output of FF3, i.e. Q3 changes on the next clock pulse each time Q0=1, Q1=1 and Q2=1 (count 7), or when Q0=1 and Q3=1 (count 9).

**Table 1: Truth Table of Synchronous Decade Counter**

Clock Pulse	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1



### Check Your Progress

1. In case of \_\_\_\_\_ counters, the clock inputs of all flip-flops receive the common clock pulses.
2. An example of a level triggered circuit is \_\_\_\_\_.
3. An example of an edge triggered circuit is \_\_\_\_\_.
4. The first integrated circuit was developed in the year \_\_\_\_\_.
5. Synchronous \_\_\_\_\_ counter counts from 0 to 9 and then recycles to 0 again.

---

## 4.5 SUMMARY

---

1. The idea occurred to a number of inventors at the same time, but the first to accomplish it were Jack Kilby of Texas Instruments and Robert Noyce of Fairchild Semiconductor Incorporated.

2. An integrated circuit is a thin slice of silicon or sometimes another material that has been specially processed so that a tiny electric circuit is etched on its surface.
3. The first integrated circuits were based on the idea that the same process used to make clusters of transistors on silicon wafers might be used to make a functional circuit, such as an amplifier circuit or a computer logic circuit.
4. The idea occurred to a number of inventors at the same time, but the first to accomplish it were Jack Kilby of Texas Instruments and Robert Noyce of Fairchild Semiconductor Incorporated.
5. Multivibrator is an electronic circuit that switches rapidly between two or more states by means of positive feedback.
6. Astable multivibrator is a circuit with is not stable in any of the state, which produces a train of square wave pulses at output at a fixed frequency.
7. Monostable multivibrators generate a single output pulse, either "high" or "low", when a suitable external trigger signal or pulse T is applied.
8. Counter is a digital sequential circuit which follows a predetermined sequence of state.

---

## **4.6 ANSWERS TO CHECK YOUR PROGRESS**

---

1. Ultra large-scale integration
2. Multivibrator
3. Three
4. Astable
5. Monostable
6. Synchronous
7. Latch
8. Flipflip
9. 1950
10. Decade

---

## **4.7 TERMINAL QUESTIONS**

---

- 1.** What is an Integrated Circuit(IC)? Classify ICs based on the generations.
- 2.** What is the difference between a level and edge triggered circuits?
- 3.** What are counters? Explain the working of a synchronous decade counter.
- 4.** What is the difference between synchronous and asynchronous counter?
- 5.** Explain the functioning of a ripple counter.





॥ सरस्वती नः सुभगा मयस्कृत ॥

Uttar Pradesh Rajarshi Tandon  
Open University

# Bachelor of Computer Application

## BCA-1.10 Computer Organization

### BLOCK

# 2

### Basic Building Block

UNIT 5	95-112
<b>Basic Building Blocks of a Computer</b>	
UNIT 6	113-126
<b>Basic Computer Organization and Design</b>	
UNIT 7	127-142
<b>Instruction Cycle</b>	
UNIT 8	143-156
<b>Design of Basic Computer</b>	
UNIT 9	157-162
<b>Central Processing Unit</b>	
UNIT 10	163-170
<b>Stack Organization</b>	
UNIT 11	171-176
<b>Instruction Formats</b>	
UNIT 12	177-182
<b>Addressing Modes</b>	

---

## Course Design Committee

---

**Dr. Ashutosh Gupta** **Chairman**  
Director (In-charge)  
School of Computer and Information Science,  
UPRTOU Prayagraj

**Prof. R. S. Yadav** **Member**  
Department of Computer Science and Engineering  
MNNIT Prayagraj

**Ms Marisha** **Member**  
Assistant Professor (Computer Science),  
School of Science, UPRTOU Prayagraj

**Mr. Manoj Kumar Balwant** **Member**  
Assistant Professor (Computer Science)  
School of Science, UPRTOU Prayagraj

---

## Course Preparation Committee

---

**Dr. Jitendra Pande** **Author**  
Associate Professor  
School of Computer Sciences & Information Technology  
Haldwani, Uttarakhand 263139

**Dr. Abhay Sexena** **Editor**  
Professor and Head, Department of Computer Science  
Dev Sanskriti Vishwavidyalya, Hardwar, Uttrakhand

**Dr. Ashutosh Gupta**  
Director (In-Charge)  
School of Computer & Information Sciences,  
UPRTOU Prayagraj

**Mr. Manoj Kumar Balwant** **Coordinator**  
Assistant Professor (computer science),  
School of Sciences, UPRTOU Prayagraj

---

© UPRTOU, Prayagraj. 2019

**ISBN :**

*All Rights are reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from the*

*Uttar Pradesh Rajarshi Tondon Open University, Prayagraj.*

Printed and Published by Dr. Arun Kumar Gupta Registrar, Uttar Pradesh Rajarshi Tandon Open University, 2019.

**Printed By:** Chandrakala Universal Pvt. Ltd. 42/7 Jawahar Lal Neharu Road, Prayagraj.

# UNIT-5

---

## BASIC BUILDING BLOCKS OF A COMPUTER

---

### Structure

- 5.0 Learning Objectives
- 5.1 Introduction
- 5.2 Basic Building Blocks of a Computer
- 5.3 Input Unit
- 5.4 OUTPUT Devices
- 5.5 Central Processing Unit
  - 5.5.1 Arithmetic Logic Unit
  - 5.5.2 Control Unit
  - 5.5.3 Register Set
- 5.6 Memory
  - 5.6.1 Random Access Memory (RAM)
  - 5.6.2 Read Only Memory (ROM)
  - 5.6.3 Units of Storage
- 5.7 Secondary Storage Devices
- 5.8 Summary
- 5.9 Answers to Check Your Progress
- 5.10 Terminal Questions

---

### 5.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Understand the basic building blocks of a a Computer.
- Know the functional units of a computer.
- Explain the functioning of a CPU.
- Understand the functioning of Hardwired Control Unit.
- Understand the functioning of Programmed Control Unit.
- Understanding the functioning of various input and output devices.

- Differentiate primary and secondary memories.

---

## 5.1 INTRODUCTION

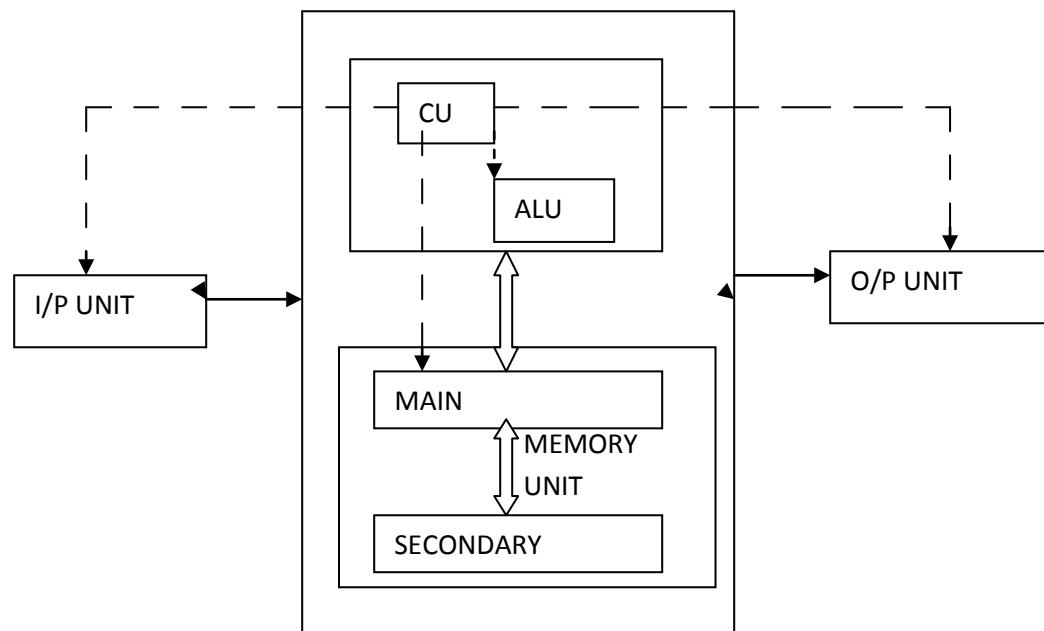
---

Computer Technology is now part of our everyday life, and almost every task we encounter involves the use of computer technology. This unit presents a brief discussion on computer and their components and other basic concepts you need to familiarize yourself with before discussing the details of the internal architecture of the computer.

---

## 5.2. BASIC BUILDING BLOCKS OF A COMPUTER

---



**Figure 59: Basic Building Block of a Computer**

- Dotted lines indicate the control signals issued by Control unit.
- Represent data or instructions.

A computer is an electronic device that takes input such as numbers, text, sound, image, animations, video, etc., processes it, and converts it into meaningful information that could be understood, presenting the changed input (processed input) as output. All numbers, text, sound, images, animations, and video used as input are called data, and all numbers, text, sound, images, animations, and video returned as output are called information. Input is the raw data entered into the computer by using input devices. It is an electronic machine/device which can input data, process them according to the instruction given and then give out the meaningful information.



- The data consists of numbers, text, sound, images, animations, and video.
- The process converts numbers, text, sound, images, animations, and video (data) into usable data, which is called information.
- The information consists of numbers, text, sound, images, animations, and video that has been converted by the process.
- The data is inserted using an input device.
- The central processing unit (CPU) converts data to information.
- The information is put on an output device.

A storage device is an apparatus for storing data and information. A basic computer consists of 4 components: an input device , a CPU, output devices, and memory. All the units of a digital computer are connected through a conducting path called Bus.

The task of the digital computing unit is to accept the data to be processed via its input unit. The CPU of the computer process the data based on the issued to the computer by the user through program. After processing, the result may be stored in the memory of the computer or may be displayed to the user via output unit.

The four functions are carried out by basic **functional units** namely:

1. Input Unit.
2. Output Unit.
3. Central Processing Unit.
4. Memory Unit.

---

## 5.3 INPUT UNIT

---

We use input devices to provide information to a computer, such as typing a letter or giving instructions to a computer to perform a task. Some examples of input devices are described in the following list.

1. **Mouse** : A device that you use to interact with items displayed on the computer screen. A standard mouse has a left and a right button.



**Figure 60: Mouse**

2. **Trackball** : This is an alternative to the traditional mouse and is favoured by graphic designers. It gives a much finer control over the movement of items on the screen. Other screen pointing devices are pointing stick, touch pad, joystick, light pen, digitizing table.



**Figure 61: Trackball**

3. **Keyboard** : A set of keys that resembles a typewriter keyboard. You use the keyboard to type text, such as letters or numbers into the computer.



**Figure 62: Keyboard**

4. **Scanner**: A device that is similar to a photocopier machine. You can use this device to transfer an exact copy of a photograph or document into a computer. A scanner reads the page and translates it into a digital format, which a computer can read. For example, you can scan photographs of your family using a scanner.



**Figure 63: Scanner**

5. **Barcode Readers :** When used in a business barcodes provide a lot of information. Made up of columns of thick and thin lines, at the bottom of which a string of numbers is printed.



**Figure 64: Barcode reader**

6. **Multimedia devices :** This is the combination of sound and images with text and graphics. To capture sound and image data, special input devices are required.

a. **Microphone:** Voice input, for instance, can be recorded via a microphone. A device that you can use to talk to people in different parts of the world. You can record sound into the computer by using a microphone. You can also use a microphone to record your speech and let the computer convert it into text.



**Figure 65: Microphone**

b. **Webcam:** A device that is similar to a video camera. It allows you to capture and send the live pictures to the other user. For example, a webcam allows your friends and family to see you when communicating with them.



**Figure 66: Webcam**

c. **Digital cameras** : record photographs in the form of digital data that can be stored on a computer. These are often used to record photographs on identity cards.



**Figure 67: Digital camera**

---

## 5.4 OUTPUT DEVICES

---

Output devices in the computer system are the equipment whereby the result of a computer operation can be viewed, heard or printed. You use output devices to get feedback from a computer after it performs a task.

1. **Monitor** : A device that is similar to a television. It is used to display information, such as text and graphics, on the computer.



**Figure 69: Monitor**

2. **Printer**: A device that you use to transfer text and images from a computer to a paper or to another medium, such as a transparency film. You can use a printer to create a paper copy of whatever you see on your monitor.
  - a. **Impact printers** : Dot matrix printers are an example of impact printers. They form characters from patterns of dots. They are inexpensive, but the output can be difficult to read.



**Figure 69: Impact printer**

**b. Non impact printers :** Inkjet printers work by shooting a jet of ink in the shape of the character required, they provide good, low-cost colour printing.



**Figure 70: Non-impact printer**

**c. Laser printer :** a laser beam is directed at an electro-statically charged surface, creating a template of the page to be printed. This template is then used to transfer the ink to the page. Toner sticks to the light images and is transferred to paper.



**Figure 71: Laser Printer**

3. Plotter: A plotter is an output device similar to a printer, but normally allows you to print larger images. It is used for printing house plans and maps.



**Figure 72: Plotter**

4. **Multimedia Output Device:** The most common multimedia output is sound, including music. The audio output device on a computer is a speaker. Headphones can also be used to receive audio output.



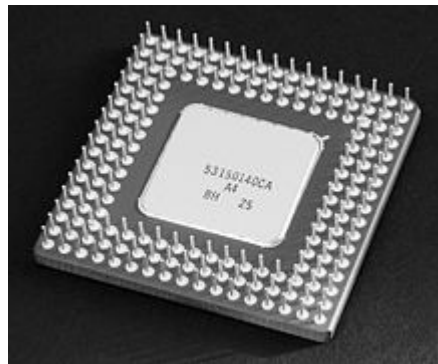
**Figure 73: Multimedia Output Device**

---

## **5.5 CENTRAL PROCESSING UNIT**

---

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. Traditionally, the term —CPU|| refers to a processor, more specifically to its processing unit and control unit (CU), distinguishing these core elements of a computer from external components such as main memory and I/O circuitry.



**Figure 74: Bottom side of an Intel 80486DX233**

The form, design and implementation of CPUs have changed over the course of their history, but their fundamental operation remains almost unchanged. Principal components of a CPU include the arithmetic logic unit (ALU) that performs arithmetic and logic operations, processor registers that supply operands to the ALU and store the results of ALU operations, and a control unit that fetches instructions from memory and —executes|| them by directing the coordinated operations of the ALU, registers and other components.



**Figure 75: An Intel 80486DX2 CPU, as seen from above**

Central Processing Unit is the brain of the computer. Based on the input provided to the CPU via one of its input device, the CPU process the data and converts it into meaningful information. It is the place where all the computing takes place. The CPU mainly consists of three parts:

- Arithmetic Logic Unit(ALU)
- Control Unit (CU)
- Register Set (Memory)

### **5.5.1 Arithmetic Logic Unit**

It is the part of a computer that performs all arithmetic computations, such as addition and multiplication, and all comparison operations. Typically, the ALU has direct input and output access to the processor controller, main memory (random access memory or RAM in a personal computer), and input/output devices. The data is transferred between the ALU and the Input/Output devices & memory through an electronic conducting path called bus. The input consists of an instruction word (sometimes called a machine instruction word) that contains an operation code (sometimes called an "op code"), one or more operands, and sometimes a format code. The operation code tells the ALU what operation to perform and the operands are used in the operation. (For example, two operands might be added together or compared logically.) The format may be combined with the op code and tells, for example, whether this is a fixed-point or a floating-point instruction. The output consists of a result that is placed in a storage register and settings that indicate whether the operation was performed successfully. (If it isn't, some sort of status will be stored in a permanent place that is sometimes called the machine status word.)

In general, the ALU includes storage places for input operands, operands that are being added, the accumulated result (stored in an accumulator), and shifted results. The flow of bits and the operations performed on them in the subunits of the ALU is controlled by gated circuits. The gates in these circuits are controlled by a sequential logic unit that uses a

particular algorithm or sequence for each operation code. In the arithmetic unit, multiplication and division are done by a series of adding or subtracting and shifting operations. There are several ways to represent negative numbers. In the logic unit, one of 16 possible logic operations can be performed - such as comparing two operands and identifying where bits don't match.

### 5.5.2 Control Unit

A control unit is a part of CPU which directs operations within the computer's processor by directing and controlling the input and output of a computer system. The processor then controls how the rest of the computer operates (giving directions to the other parts and systems). A control unit works by gathering input through a series of commands it receives from instructions in a running program and then outputs those commands into control signals that the computer and other hardware attached to the computer carry out.

Control Unit perform various functions in a computer such as: it controls the movement of data between various units of a computer. It is responsible for the deciding the sequence in which the various instructions are to be executed. It is also responsible for handles multiple tasks, such as fetching, decoding, execution handling and storing results.

CUs are designed in two ways:

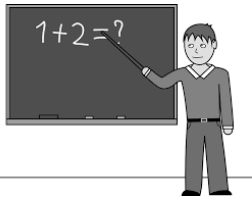
- *Hardwired control:* CU is made up of sequential and combinational circuits to generate the control signals The CU is made up of flip-flops, logic gates, digital circuits and encoder and decoder circuits that are wired in a specific and fixed way. When instruction set changes are required, wiring and circuit changes must be made. This is preferred in a reduced instruction set computing (RISC) architecture, which only has a small number of instructions.
- *Microprogram control:* Microprograms are stored in a special control memory and are based on flowcharts. The operation of all the hardware of the computer is control unit. It monitors and controls the input devices, output devices, memory and the ALU of the computer. In case of microprogrammed implementation of a control unit, it some addition/modification is required at the later stage of implementation, one need not to redesign the whole circuit, as in the case of hardwired unit, but control memory is updated with new microprogramme.

### 5.5.3 Register Set

The operand (data on which the operation is to be performed) and the operation is supplied to the CPU via. memory. Now the CPU requires some location where the data on which the operation is to be performed can be



manipulated. For this purpose, a very fast memory element, called registers, are used. These registers along with CU and ALU are the part of CPU. These registers hold the data and directly attached to the electronic circuitry which is required to perform various operations.



## Check Your Progress

1. A \_\_\_\_\_ device is an apparatus for storing data and information.
2. Dot matrix printers are an example of \_\_\_\_\_ printers.
3. A \_\_\_\_\_ is an output device similar to a printer, but normally allows you to print larger images.
4. A \_\_\_\_\_ 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.
5. \_\_\_\_\_ is the part of a computer that performs all arithmetic computations, such as addition and multiplication, and all comparison operations.
6. The data is transferred between the ALU and the Input/Output devices & memory through an electronic conducting path called \_\_\_\_\_ .
7. Op-code stands for \_\_\_\_\_ .
8. A \_\_\_\_\_ unit is a part of CPU which directs operations within the computer's processor by directing and controlling the input and output of a computer system.

---

## 5.6 MEMORY

---

All computers need to store and retrieve data for processing. The CPU is constantly using memory from the time that it is switched on until the time you shut it down. There are two types of storage devices as illustrated in the flow chart below.

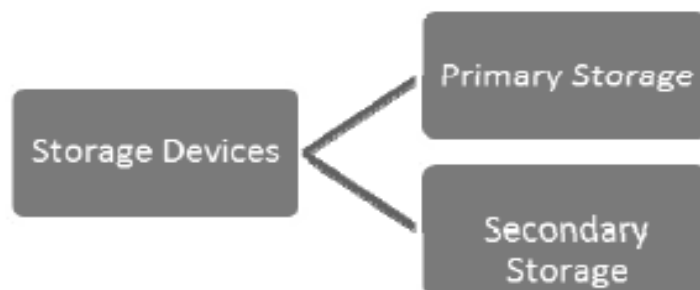


Figure 77: Types of storage devices

**Primary Storage** is also called main memory or immediate access store (IMAS). This is necessary since the processing unit can only act on data and instructions that are held in primary storage. Primary storage consists of two types of memory chips:

- Random Access Memory (RAM)
- Read Only Memory (ROM)

### 5.6.1 Random Access Memory (RAM)

**Random Access Memory (RAM)** is the main working memory. RAM is only filled after computer has been turned on and is given something to do. It holds data and instructions temporarily while processing takes place. RAM is volatile – this means that if the power is turned off or the computer reboots (start up again) all the information held in RAM will be lost. RAM is measured in MB (megabytes) and most entry level computers will have 1024 MB RAM but you also find some computers having up to 3 GB RAM. RAM chips are expensive and the price of a computer is determined by the amount of RAM space in the chip.



**Figure 78: RAM**

### 5.6.2 Read Only Memory (ROM)

**Read Only Memory (ROM)** holds data and instructions necessary for starting up the computer when it is switched on. These instructions are hard-wired at the time of manufacture. ROM is permanent and cannot be deleted but can only be accessed or read, hence the name Read Only Memory. Data stored in ROM is non-volatile – meaning that memory will not be lost when power is turned off.



**Figure 79: ROM**

### 5.6.3 Units of Storage

The memory of all digital computers is two-state (bi-stable) devices. Computers operate using a **binary number system** – and therefore use *binary digits (bits)*. Bits have only two values by 0 and 1. A **bit** is the smallest unit of storage in a computer. The amount of data and instructions that can be stored in the memory of a computer or secondary storage is measured in bytes.

A **byte** is made up of a combination of eight (8) **bits** and has the storage power to represent one character (a character is a letter or symbol or punctuation mark or blank space).

**Table 7: Unit of Storage**

Units of Storage	
<b>1 byte</b>	8 bits
<b>1 kilobyte (K)</b>	1024 bytes
<b>1 megabyte (MB)</b>	1000 kilobytes (approx. 1 million bytes)
<b>1 gigabyte (GB)</b>	1000 megabytes (approx. 1 billion bytes)
<b>1 terabyte (TB)</b>	1000 gigabytes (approx. 1 trillion bytes)

---

## 5.7 SECONDARY STORAGE DEVICES

---

PCs use a simple method of designating disk drives to store data. These drives are assigned letters of the alphabet.

<b>A Drive</b>	Floppy drive. Still found in older computers
<b>C Drive</b>	Internal hard drive (hard disk drive) situated inside the system case.
<b>D Drive</b>	Usually the CD-ROM/DVD-ROM drive although can also be used for another virtual or physical hardware if a second one is deployed.
<b>E Drive or Higher</b>	Usually use for any other disks, such as CD-writer, USB flash drive, external hard drive, etc.

Data and information stored on a permanent basis for later use. Secondary storage is cheaper to purchase and access. Hard disks, Zip drives, Optical disks (CD's and DVD's) are all examples of secondary storage.

1. **Internal Hard Disks** are rigid inflexible disks made of highly polished metal. Data is stored magnetically. They can contain a single disk or two or disks stacked on a single spindle. They come in a variety of sizes but all have a very high storage capacity compared to floppy disks. An average computer has a hard disk of about 80 -250GB. It provides direct access to information.



**Figure 79: Internal hard disc**

2. **External hard Disks/Drive** - same features as the internal hard disks, but are external to the system unit and therefore can be carried around.



**Figure 80: External hard Disc**

3. **USB flash drive** consists of a flash memory data storage device integrated with a USB (Universal Serial Bus) interface. USB flash drives are typically removable and rewritable. They come in a variety of sizes to include 128MB, 256MB, 512MB, 1G, 2G, 8G etc.



**Figure 81: USB flash drive**

4. **Memory Card** - Use mainly with digital cameras, cellular phones and music players (MP3, MP4 and iPods). They offer high-re-record ability and fast and power-free storage. Data can be access by linking the card to a computer using a USB cable or a memory card reader.



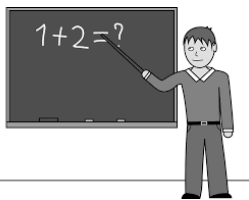
**Figure 82: Memory card**

5. **Optical Disks** are disks that are read by laser beams of lights. The three main types are CD-R, CD-RW and DVD.



**Figure 83: Optical disc**

- CD-R** or CD-ROM (Compact Disk – Read Only Memory) are so called because you can only red the information on the CD-ROM. They are particularly useful for storing multimedia (texts, graphics, sound and videos), application software packages.
- CD-R** or Compact Disk Recordable allows you to write information onto the disk only once using a CD recordable burner.
- CD-RW** or Compact Disk Rewriteable, allows you to write and erase information from the disk many times. They are used to store large volumes of information such as texts, graphics, sound and video.
- DVD** disks or Digital Versatile Disks are specifically created to store movies. A typical DVD disk can hold between 4.7GB and 17GB of information.



## Check Your Progress

- \_\_\_\_\_ is the main working memory.
- \_\_\_\_\_ holds data and instructions necessary for starting up the computer when it is switched on.
- A \_\_\_\_\_ is the smallest unit of storage in a computer.

---

## 5.8.1 SUMMARY

---

1. Since the 1940s when computer technology was used to support the creation of firing tables for the artillery and to the introduction of the World Wide Web network of computers in the 1980s computer technologies have become a large part of our everyday life. The use of computers is accelerating. They are now in our cars, our phones, our refrigerators. Almost every type of electronic device has a computer chip in it. Each chip relays on commands. Commands must be input using different devices. The next topic will examine some of these input and output devices.
2. Computer hardware consists of input, output, process and storage devices.
3. You use input devices such as keyboard, mouse, scanner and multimedia devices to provide information to a computer.
4. Output devices are use to get feedback from a computer after it performs a task.
5. Examples include monitor, printer and multimedia devices.
6. CPU takes raw data and turns it into information. The CPU is made up of Control Unit, Arithmetic Logic Unit (ALU) and the main memory.
7. Storage devices are divided in primary and secondary storage devices.
8. Primary/main memory is subdivided into ROM and RAM.
9. Random Access Memory (RAM) is the main memory and allows you to temporarily store commands and data.
10. 10. Read Only Memory (ROM) retains its contents even after the computer is turned off.
11. 11. A bit is the smallest unit of storage in a computer.
12. 12. The amount of data and instructions that can be stored in the memory of a computer or secondary storage is measured in bytes. A byte is made up of a combination of eight bits and has the storage power to represent one written character.
13. 13. Hard disks, CD-R, CD-RW and DVD are secondary storage devices.

---

## 5.9 ANSWERS TO CHECK YOUR PROGRESS

---

1. Storage
2. Impact

3. Plotter
4. Central Processing Unit(CPU)
5. ALU
6. Bus
7. Operation Code
8. Control
9. Random Access Memory (RAM)
10. Read Only Memory (ROM)
11. Bit

---

## **5.10 TERMINAL QUESTIONS**

---

1. What is the role of Control Unit of a CPU?
2. What are the two possible configurations of a Control Unit?
3. Draw and explain the basic building block of a computer.
4. What is volatile memory? Give examples.
5. Give examples of some screen pointing devices.
6. What is an input device? Give some examples.
7. What is an output device? Give some examples.
8. What is a bus?
9. What is the difference between RAM and ROM?





# UNIT-6

---

## BASIC COMPUTER ORGANIZATION AND DESIGN

---

### Structure

- 6.0 Learning Objectives
- 6.1 Introduction
- 6.2 The Basic Computer
  - 6.2.1 Addressing Modes
  - 6.2.3 Processor Register
- 6.3 Basic Computer Instructions
- 6.4 Common Bus System
- 6.5 Control Unit
- 6.6 Summary
- 6.7 Answers to Check Your Progress
- 6.8 Terminal Questions

---

### 6.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Define a basic computer
- Explain an Instruction Format
- Explain various types of addressing modes.
- Differentiate between direct and indirect addressing mode.
- Identify the processor registers.
- Understand the basic computer instruction.
- Know the functioning of common bus system.
- Understand the functioning of control unit.

---

## 6.1 INTRODUCTION

---

In the previous unit, we have discussed a basic structure of a computer. Now let's discuss the organization and the architecture in more details. The most important part of the computer is processor. A processor contains:

1. Set of registers
2. Arithmetic and Logic units for performing both arithmetic and logical operations
3. A mechanism to pipeline several consecutive instructions to speed execution

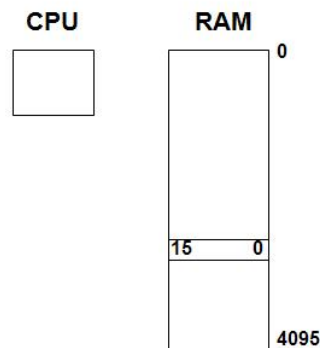
The internal organization of a processor is very complex. For the purpose of understanding of the internal working of a processor, we have considered a very basic computer with limited capabilities. In this unit, we will discuss the architecture of a basic computer. We will also discuss the Instruction Format, addressing modes, common bus system and hardwired implementation of control unit in details.

---

## 6.2 THE BASIC COMPUTER

---

The Basic Computer has two components, a processor and memory. In our basic computer, the memory has 4096 words in it i.e. there are 4096 different unique locations in the memory. Therefore, to differentiate between these 4096 locations, we require at least,  $2^{12} = 4096$ , 12 bits. Each unique location in the memory can store 16 bits i.e. the word size is 16 bits.



**Figure 84: Memory Architecture of a simple memory**

A processor is capable of performing various operations in the data for which the processor is designed for. But to perform certain operations, a programmer needs to specify the following:

- The operation to be performed.
- The operand i.e. the data item on which the operation is performed.

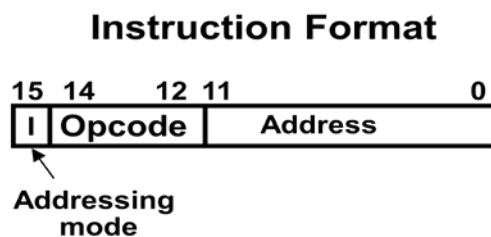
- The sequence in which the operations are to be performed.

An *instruction* is used to specify which operation is to be performed by the computer. It is a group of bits that tell the computer to *perform a specific operation* (a sequence of micro-operation).

The sequence of instructions is known as *program*. It specifies the sequence in which the operation is to be performed. The instructions of a program, along with any needed data are stored in memory. The CPU reads the next instruction from memory. It is placed in an *Instruction Register (IR)*. Control circuitry in control unit then translates the instruction into the sequence of microoperations necessary to implement it.

Each unique location in the memory contains a 16-bit instruction. A computer instruction is often divided into two parts:

- An *opcode* (Operation Code) that specifies the operation for that instruction
- An *address* that specifies the registers and/or locations in memory to use for that operation



**Figure 85: An Instruction Format**

As discussed before, the memory of the Basic Computer contains 4096 (= 2<sup>12</sup>) words, we need 12 bits [0 to 11] to specify which memory address this instruction will use. 3 bits [12-14] are used to specify different operations to be performed. Bit 15 of the instruction specifies the *addressing mode* (0: direct addressing, 1: indirect addressing). Based on the various combinations of 3 bits [12-14], 2<sup>3</sup>=8 different operations can be specified in both the modes.

### 6.2.1 Addressing Modes

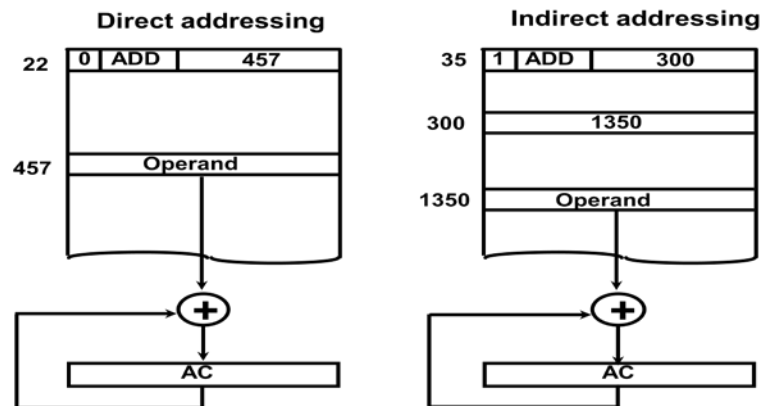
The address field of an instruction can represent either

a. **Direct address:** the address part of the instruction code is used to specify the address of the operand in the main memory. For example, the address part of the instruction code specifies location number 457. Now the location number 457 contains the operand.

**Indirect address:** the address part of the instruction code specifies the address of that location in the memory, which contains the address of the

operand in main memory. For example, instruction code specifies address 300 in the main memory. Now, in case of indirect addressing, the location 300 does not contain the operand, but it again contains the address of that location, which contains the operand. i.e. 1350

The bits of the instruction code are used to translate address of the operand in the main memory by calculating the effective address. Effective Address (EA), is the address, that can be directly used without modification to access an operand for a computation-type instruction, or as the target address for a branch-type instruction.



**Figure 86: Direct and Indirect Addressing Modes**

### 6.2.3 Processor Register

There are several general purpose registers present in the processor referred to as processor registers. The number of registers varies from architecture to architecture. These registers are used by the processors to perform various operations on the data that is fetched from the memory to processor. The contents of these registers could be directly accessed by the processor. These registers are directly connected with the necessary circuitry, like incrementer, adder, subtractor, shifter etc. The significance of a general purpose register is that it can be referred to in instructions. Table 8 below summarized the registers present in our basic computer model.

**Table 2 : Processor Registers**

DR	16	Data Register	Holds Memory Operand
AR	12	Address Register	Holds Address for Memory
AC	16	Accumulator	Processor Register
IR	16	Instruction Register	Holds Instruction Code
PC	12	Program Counter	Holds Address of Instruction
TR	16	Temporary Register	Holds Temporary Data
INPR	8	Input Register	Holds Input Character
OUTR	8	Output Register	Holds Output Character

**Data Register(DR) :** Data register holds the data in which the operation is to be performed. When an operand is found, using either direct or indirect addressing, it is placed in the *Data Register (DR)*. The processor then uses this value as data for its operation. Its capacity is equal to the length of data, in this example it is 16 bits.

- **Program Counter(PC):** The processor has a register, the *Program Counter (PC)* that holds the memory address of the next instruction to be executed. Since the memory in the Basic Computer only has 4096 locations, the PC only needs 12 bits.
- **Address Register(AR):** In a direct or indirect addressing, the processor needs to keep track of what locations in memory it is addressing: The *Address Register (AR)* is used for this purpose. The AR is a 12 bit register in the Basic Computer.
- **Accumulator(AC):** The Basic Computer has a single *general purpose register* – the *Accumulator (AC)*. Since it is used to store data, it is a 16-bit register. A general purpose register can be referred to in instructions directly. For eg. *load AC* is an instruction using which the contents of AC into a specified memory location. We can clearly observe that here Accumulator(AC) is referred in the instruction itself.
- **Temporary Register (TR):** It is a scratch register to store intermediate results or other temporary data; in the Basic Computer this is the *Temporary Register (TR)*. As it is again used to store data, its length it 16 bits.
- **Input Register (INPR):** it holds an 8 bit character gotten from an input device
- **Output Register (OUTR):** it holds an 8 bit character to be sends to an output device.

The input and output devices are attached to serial port i.e. they transmit data serially whereas the processor process whole of the data at a time. Therefore, we need an additional input or output register, which are attached to input and output devices respectively. The data is transmitted to the input register by an input device serially. Once the register is full, the data is transmitted from input register to the processor in parallel fashion. Similarly, the data is transmitted from the processor to the output register in parallel fashion and from output register to output device serially. Since the input and output devices uses extended ASCII format, which is 8-bits, therefore the width of input and output registers is 8 bits.

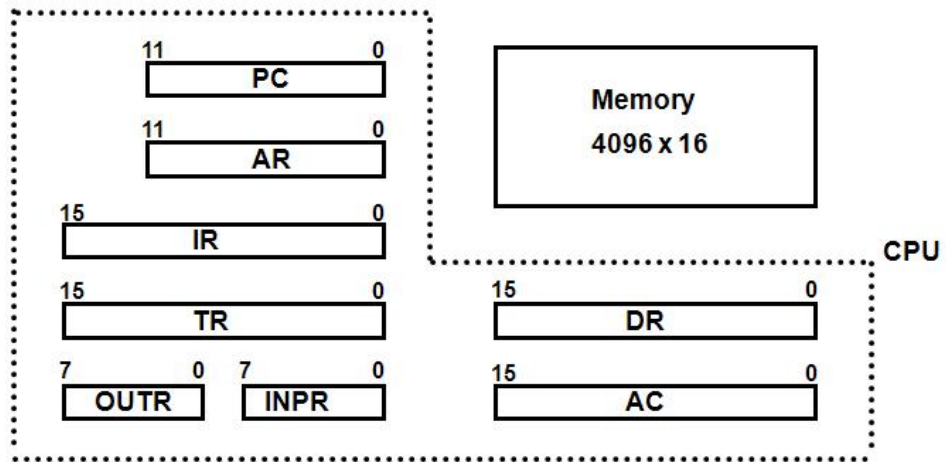
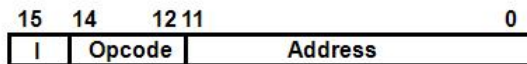


Figure 87: Processor Registers

## 6.3 BASIC COMPUTER INSTRUCTIONS

The basic instruction code has three fields, as shown in Figure 89.

### Memory-Reference Instructions (OP-code = 000 ~ 110)



### Register-Reference Instructions (OP-code = 111, I = 0)



### Input-Output Instructions (OP-code = 111, I = 1)

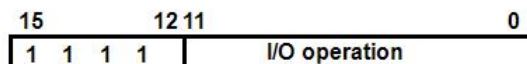


Figure 88: Basic Computer Instruction Format

The most significant bit, i.e. bit 15, is marked as *I*, which is a mode selection bit. If it is set, it is indirect addressing mode else direct addressing mode. This bit has a direct effect on the address part (bit 0-11) of the instruction code and it is used in conjunction with other three opcode bits (bit 12-14) to specify one of the various addressing modes. In direct addressing mode, the address part of the instruction code gives the address of the operand in the main memory. In indirect mode, the address part of the instruction code specifies that address of the main memory, which contains the address of the operand in the main memory. There are some situations, when the operand is stored in one of the CPU registers or the operand is specified by the Input/Output devices itself. In these cases, no memory reference is required. Hence, the address part of the instruction

code is no longer required to specify the address of the operand in the main memory. Therefore, the other 12 bits(0-11) can be used to specify test or other conditions. The above situation can be summarized as follows:

- When  $I=0$  and bit 12-14 are all 1's(0111),i.e. the hexadecimal equivalent of the digit 7, it is register mode. In register mode, the other 12 bits of the address part of the instruction code are used to specify one of the various register mode instruction(please refer to table 6). The 16 bits of the instruction code is equivalent to four hexadecimal bits, with the first most significant bit of the hexadecimal equal to four most significant bits(12-15) of the instruction code. In register mode, the most significant hexadecimal bit is always 7. Rest of the 12 bits are used to specify different register related operation. For eg. 7800, which is equivalent to 0111 1000 0000 0000 in binary, specifies Clear AC( clear accumulator) operation.
- When  $I=1$  and bit 12-14 are also all 1's(1111), i.e. the hexadecimal equivalent of digit F, it specifies Input/Output mode. In this case also the other 12 bits are used to specify one of the various Input/Output operations. For eg. F800 is an Input/Output operation which is used to transfer input characters to AC.
- When  $I=0$  and bit(12-14) have any combinations except all 1's. i.e. hexadecimal equivalent values of 0-6, it is a direct addressing mode. In this case the other 12 bits are used to specify the address of the operand in the main memory. For eg. 1xxx( i.e. binary equivalent 0001 xxxx xxxx xxxx xxxx( x's denotes don't care conditions) specifies ADD which performs addition operation on the operand and the content of AC.
- When  $I=1$  and bit(12-14) have any combinations except all 1's, i.e. hexadecimal equivalent values of 8-E, it is a indirect addressing mode. In this case the other 12 bits are used to specify the address of that location in the memory, which contains the address of the operand in main memory. If you see Table 9 you will observe that all the operation of direct and indirect memory mode are same, the only difference is number of main memory access to fetch operand to the CPU registers. For eg. 1xxx and 9xxx, both are used to add memory word to AC, but in case of 1xxx it is direct mode, where only one memory reference is required to fetch the operand. In case of 9xxx, two memory references are required.

A computer should have a set of instructions so that the user can construct machine language programs to evaluate any function that is known to be computable. The instruction set of a computer is said to be complete if it contains the following category of instructions to manipulate the data.

- Functional Instructions
  - Arithmetic, logic, and shift instructions
  - ADD, CMA, INC, CIR, CIL, AND, CLA

➤ Transfer Instructions

- Data transfers between the main memory and the processor registers
- LDA, STA

<b>Symbol</b>	<b>Hex Code</b>		<b>Description</b>
	<b>I = 0</b>	<b>I = 1</b>	
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instr. if AC is positive
SNA	7008		Skip next instr. if AC is negative
SZA	7004		Skip next instr. if AC is zero
SZE	7002		Skip next instr. if E is zero
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

➤ Control Instructions

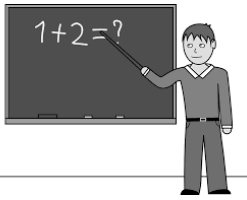
- Program sequencing and control
- BUN, BSA, ISZ

➤ Input/Output Instructions

- Input and output
- INP, OUT

The basic instruction set disused above and summarized in table 2 consists of all the instructions required by a basic computer. Hence, we can claim that the instruction set is complete.





## Check Your Progress

1. The sequence of instructions is known as \_\_\_\_\_.
2. The CPU reads the next instruction from memory. It is placed in an \_\_\_\_\_.
3. When an operand is found, using either direct or indirect addressing, it is placed in the \_\_\_\_\_.

## 6.4 COMMON BUS SYSTEM

The registers in the Basic Computer are connected using a bus. This gives a savings in circuitry over complete connections between registers.

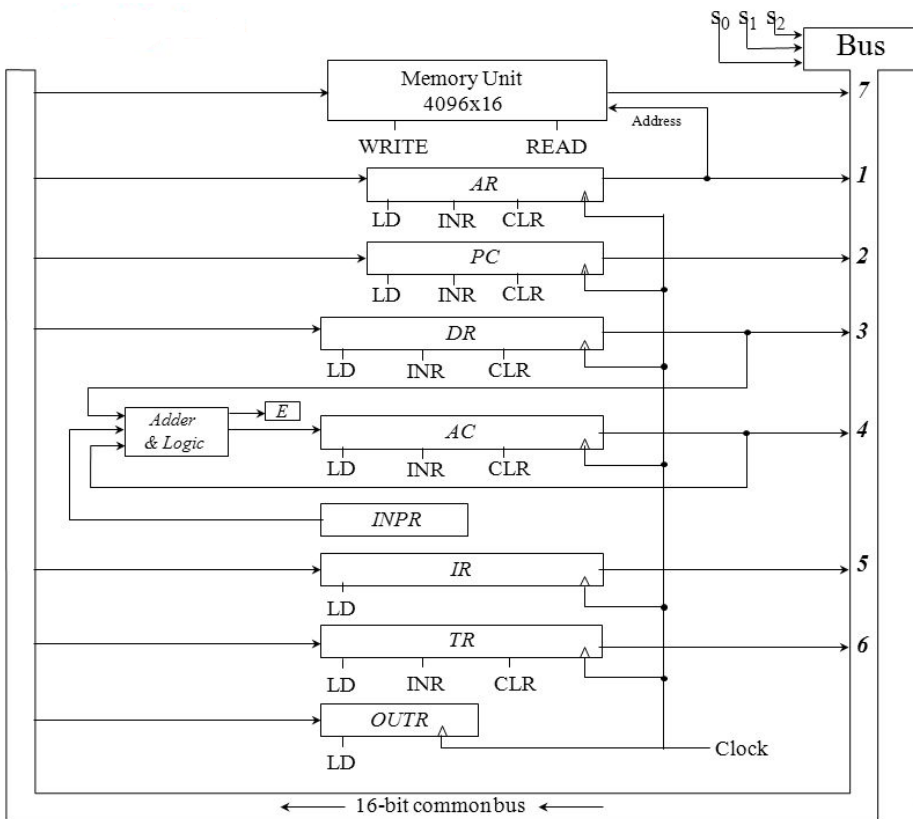


Figure 89: 16-bit common bus

The common bus is used by the registers to communicate between the registers and main memory. There is a control circuitry, located at the right-top corner in the Figure 90, used to control the operation of the bus. It is basically a decoder 105

with three control lines. These three control lines, S<sub>2</sub>, S<sub>1</sub>, and S<sub>0</sub> control which register the bus selects as its input. This is shown in the Table 10 below.

S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	Register
0	0	0	x
0	0	1	AR
0	1	0	PC
0	1	1	DR
1	0	0	AC
1	0	1	IR
1	1	0	TR
1	1	1	Memory

The inputs of all the register, except *Accumulator(AC)* and input register, is directly connected to the common bus. In case of accumulator, its input is connected to adder and logic circuit. This adder and logic circuit receives input from *Input Register(INPR)*, *Data Register(DR)* and a feedback for *Accumulator(AR)* itself. During adding operation, sometime a carry bit is generated. A one bit flip-flop *E*, is the carry bit. If this bit is set, this shows that a carry is generated during addition operation. The input of the *Input Register(INPR)* is directly connected through the input devices. Similarly, the output of the *Output Register(OUTR)* is connected to the output devices. *Load(LD)*, *Clear(CLR)* and *Increment(INC)* control signals are attached to the registers. Whenever a *Load(LD)* signal is enabled, the content of the common bus are loaded to the register whose Load signal is enabled. *Clear(CLR)* signal is used to clear the content of a register. Whenever the Clear signal of a register is enabled, all the bits of that register become 0. *Increment(INR)* signal is used to increment the content of the register by 1. All the control signals S<sub>2</sub>,S<sub>1</sub>and S<sub>0</sub>;LOAD, INR, CLR, READ, WRITE, etc are generated and controlled by the control unit of the processor. Either one of the registers will have its load signal activated, or the memory will have its read signal activated. The 12-bit registers, AR and PC, have 0's loaded onto the bus in the high order 4 bit positions. When the 8-bit register OUTR is loaded from the bus, the data comes from the low order 8 bits on the bus.

The operation of the common bus can be explained with the help of an example. Suppose the processor need to transfer data from *Data Register(DR)* to *Temporary Register(TR)*. The selection lines S<sub>2</sub>,S<sub>1</sub> and S<sub>0</sub> will have the value 0,1, and respectively(refer to the Table 10 above where DR have the value 011). The register selected by the selection circuit will have the hold of the common bus. The contents of the *Data Register(DR)* is transferred to the common bus. The load signal of the destination register, i.e. *Temporary Register(TR)* is high. Hence the data from the common bus is transferred to the *Temporary Register(TR)*.

Similarly, if the data is to be transferred from main memory to Data Register(DR), the control circuit select memory through the selection lines S2,S1 and S0. Memory is selected when the value of S2,S1 and S0 is 1,1 and 1 respectively. Memory have different locations and one have to specify the address of the location from where the data is to be transferred to Data Register(DR). This is address is given by Address Register(AR). For this reason the output of the Address Register(AR) is directly connected to the Memory. There are two control signals READ and WRITE which controls the read and write operation of the memory. To write into the memory, at the location specified by the Address Register(AR), WRITE signal is enabled and for reading the content of the Memory, from the address specified by the Address Register(AR), READ signal is enabled. Since, this is a memory read operation, so in our example, READ signal is enabled. The content that location of the memory, whose addresses is specified by the Address Register(AR), is transferred to the common bus. Now the load signal of the Data Register(DR) is enabled which transfers the content of the common bus to Data Register(DR).

---

## 6.5 CONTROL UNIT

---

Control unit (CU) of a processor translates from machine instructions to the control signals for the micro-operations that implement them. Control units are implemented in one of two ways:

- *Hardwired* Control: Control Unit is made up of sequential and combinational circuits to generate the control signals.
- *Microprogrammed* Control: A control memory on the processor contains microprograms that activate the necessary control signals.

Now let us discuss the functioning of a Control Unit. Let us consider a hardwired implementation of a control unit shown in fig. 40 below. It consists of a 4-bit sequence counter, which can in binary from 0 to 15. The output of the counter are decoded into 16 timing signals from T0 to T15. The sequence counter have three inputs viz. INR, CLR and Clock. Increment for increment, CLR is for clear and Clock in input to sequence counter from clock source. The CLR signals sets to high if sequence of all the states from T0 to T15 are generated. When CLR is enabled, Clock pulse T0 is generated. Every instruction takes varying time to execute. In case the instruction is executed in less than 16 clock pulses, the CLR is enabled once the instruction is executed. The Instruction Register(IR) stores the instruction which is fetched form the memory and is ready to be executed. The 12 least significant bits(0-11) of IR contains either Operand

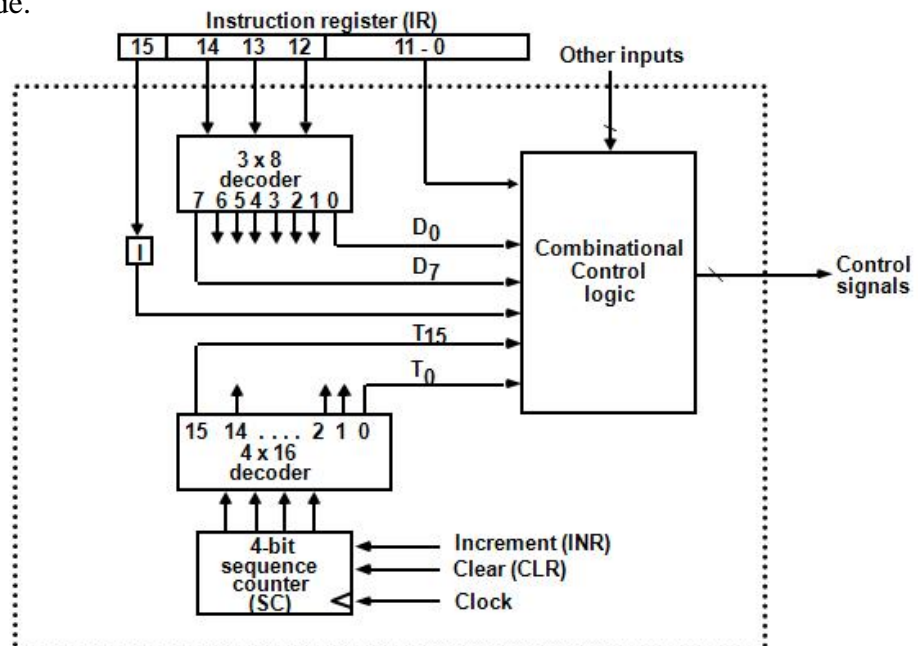
itself or the address of the operand. 3 bits(12-14) of the IR, known as op-code bits, are fed to a input of a 3 X 8 decoder. The decoder generated signal D0 to D7 based on the combination of the bits. The details are listed in Table 11.

**Table 3: Operation Code table**

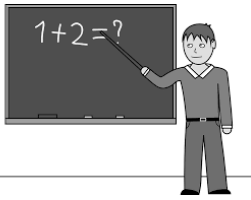
Bit-14	Bit-13	Bit-12	Decoder Output
0	0	0	D <sub>0</sub>
0	0	1	D <sub>1</sub>
0	1	0	D <sub>2</sub>
0	1	1	D <sub>3</sub>
1	0	0	D <sub>4</sub>
1	0	1	D <sub>5</sub>
1	1	0	D <sub>6</sub>
1	1	1	D <sub>7</sub>

D0 to D7 , the output of the 3 X 8 decoder , decided the operation which is to be performed. Suppose D4 is allocated for addition. The output of D4 is fed to the combinational control unit. This unit generates all the necessary control signals to initiate the addition operation by full adder.

The 15th Bit of the IR which is marked as *I*, is the mode selection bit. *I* decides whether the operand, which is to be fetched form main memory is in direct mode or indirect mode. If *I* is 1 then indirect mode, else direct mode.



**Figure 90 : Hardwired Control Unit**



## Check Your Progress

1. The registers in the Basic Computer are connected using a\_\_\_\_\_.
2. The inputs of all the register, except \_\_\_\_\_ and input register, is directly connected to the common bus.
3. \_\_\_\_\_Control Unit is made up of sequential and combinational circuits to generate the control signals.
4. In a \_\_\_\_\_ control unit, a control memory on the processor contains microprograms that activate the necessary control signals.

---

## 6.6 SUMMARY

---

1. There are several general purpose registers present in the processor referred to as processor registers.
2. These registers are used by the processors to perform various operations on the data that is fetched from the memory to processor.
3. Data register holds the data in which the operation is to be performed.
4. The processor has a register, the *Program Counter (PC)* that holds the memory address of the next instruction to be executed.
5. In a direct or indirect addressing, the processor needs to keep track of what locations in memory it is addressing.
6. A Basic Computer has a single *general purpose register* – the *Accumulator (AC)*.
7. The common bus is used by the registers to communicate between the registers and main memory.
8. Control unit (CU) of a processor translates from machine instructions to the control signals for the micro-operations that implement them.

---

## 6.7 ANSWERS TO CHECK YOUR PROGRESS

---

1. Program
2. Instruction Register (IR)
3. Data Register (DR)

4. Bus
5. Accumulator(AC)
6. Hardwired
7. Microprogrammed

---

## **6.8 TERMINAL QUESTIONS**

---

1. What is the difference between a direct and an indirect addressing mode?
2. What is a control unit?
3. What is a difference between a hardwired and a microprogrammed control unit.
4. Explain the working of a hardwired control unit with a help of a diagram.
5. Explain how the mode selection bit I is used to differentiate between various addressing modes.
6. Explain the working of 16-bit common bus with a help of a diagram.
7. What are the functions of the various CPU registers. Explain in detail.
8. What is an instruction format?

# UNIT-7

---

## INSTRUCTION CYCLE

---

### Structure

- 7.0 Learning Objectives
- 7.1 Introduction
- 7.2 Instruction Cycles and Subcycles
  - 7.2.1 Instruction Fetch from Memory
  - 7.2.2 Instruction Decode
  - 7.2.3 Instruction Execution
- 7.3 Register Reference Instructions
- 7.4 Memory Reference Instructions
- 7.5 Input-Output Reference Instructions
- 7.6 Summary
- 7.7 Answers to Check Your Progress
- 7.8 Terminal Questions

---

### 7.0 LEARNING OBJECTIVES

---

After reading this chapter, you will be able to:

- Define an Instruction cycle.
- Explain different steps involved in an instruction cycle.
- Understand the execution of register reference instructions.
- Understand the execution of memory reference instructions.
- Understand the execution of Input/Output reference instructions.

---

### 7.1 INTRODUCTION

---

Before looking at *how* a computer does what it does, let us look at *what* it can do. The definition of a computer outlines its capabilities; a computer is an electronic device that can store, retrieve, and process data. Therefore, all of the

instructions that we give to the computer relate to storing, retrieving, and processing data.

The underlying principle of the von Neumann machine is that data and instructions are stored in memory and treated alike. This means that instructions and data are both addressable. Instructions are stored in contiguous memory locations; data to be manipulated are stored together in a different part of memory.

In this unit, we will discuss the phases involved in the execution of an instruction by the CPU. We will also discuss how a register reference, memory reference and input/output reference instructions are executed by CPU.

---

## 7.2 INSTRUCTION CYCLES AND SUBCYCLES

---

A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. Each instruction cycle in turn is subdivided into a sequence of sub cycle. Each instruction cycle consists of

1. Instruction Fetch from Memory
2. Instruction Decode
3. Read Effective Address(if indirect addressing mode)
4. Instruction Execution
5. Go to step 1) : Next Instruction[PC + 1]

After execution, the control goes back to step number 1 to again fetch the new instruction. This cycle goes on till the end of the program and stops as soon as a HALT instruction is encountered. Now let us discuss each of the above steps in detail.

### 7.2.1 Instruction Fetch from Memory

In register transfer language, the instruction fetch can be represented as follows:

$$T_0 : AR \leftarrow PC$$

$$T_1 : IR \leftarrow M[AR], PC \leftarrow PC + 1$$

This could be explained as follows:

At  $T_0 = 1$

The content of PC are placed onto the bus by making the bus selection inputs  $S_2S_1S_0=010$  and these contents of the common bus are transferred to AR by enabling the LD input of AR. This is denoted by the following register transfer statement:



When  $T_1 = 1$

The READ signal line of the main memory is enabled to facilitate the memory read operation. This is followed by placing the content of the desired location of the main memory to the common bus by making  $S_2S_1S_0 = 111$ . The choice of the location of memory is specified by the AR. This is followed by transfer of the content of the common bus to the Instruction Register(IR) by enabling the LD input of IR. Now the instruction fetch is complete. At the end the Program Counter(PC) is incremented by enabling the INR input of PC. This can be denoted by the following register transfer statement:

$$T_1 : IR \leftarrow M[AR], PC \leftarrow PC + 1$$

### 7.2.2 Instruction Decode

In this sub-phase, the instruction which is fetched from the main memory and now residing in the Instruction Register(IR) is decoded.

At  $T_2=1$

The opcode bits are decoded to enable one of the eight outputs i.e.  $D_0 - D_7$ . Simultaneously, the address of the operand, i.e. bit(0-11) of the instruction register are transferred to Address Register. To check whether it is a direct addressing mode or an indirect addressing mode, value of bit(15) of Instruction Register(IR) is transferred to  $I$ . This can be denoted by the following register transfer statement:

$$T_2 : D_0, \dots, D_7 \leftarrow \text{Decode } IR(12-14), AR \leftarrow IR(0-11), I \leftarrow IR(15)$$

### 7.2.3 Instruction Execution

In this sub-phase, the value of  $I$  is tested to decide between register, input/output, direct memory or indirect memory mode.

At  $T_3=1$

To make this decision, the value of  $D_7$  is tested. The bit  $D_7$  is set when bit(12- 14) are all 1's. If  $D_7$  is 1, then it means that it is either a register mode or input/output mode. To resolve this, the status of  $I$  is checked. If  $I=0$ , then it is register mode i.e. when  $D_7=1$  ; Register( $I=0$ ) then register mode. In this case no memory reference is required. Hence the instruction is executed. This can be denoted by the following register transfer statement:

$$D_7 I' T_3 \text{ (Execute)}$$

If  $I=1$ , then it is input/output mode i.e. when  $D_7=1$  ; Register( $I=1$ ) then input/output mode. i.e., when  $D_7=1$  and Register( $I=1$ ) then input/output mode.

$T_5=1$

$$AR \leftarrow M[AR]$$

T<sub>6</sub>=1

$$DR \leftarrow M[AR]$$

In this case also, no memory reference is required. Hence, the instruction is executed. This can be denoted by the following register transfer statement:

D7IT3 (Execute)

At T<sub>4</sub>=1,

In case, D<sub>7</sub> is 0, it means that it is a memory reference instruction. Now to select between direct and indirect memory instruction, again the status of I is checked.

If I=0, then it is direct memory instruction. In this case, the operand is fetched from memory and transferred to Data Register for processing.

$$DR \leftarrow M[AR]$$

After this, the instruction is executed. Only one access to the main memory is required in case of direct memory mode.

If I=1, then this is an indirect memory instruction. In the first step, the address is fetched i.e.

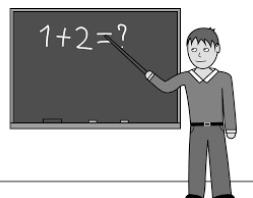
T<sub>5</sub>=1

$$AR \leftarrow M[AR]$$

T<sub>6</sub>=1

$$DR \leftarrow M[AR]$$

After this, the instruction is executed. Two accesses to the main memory is required in case of indirect memory mode. The above process could be summarized with the help of a flowchart as shown in Figure 93.



## Check Your Progress

1. A \_\_\_\_\_ residing in the memory unit of the computer consists of a sequence of instructions.
2. The \_\_\_\_\_ increments one by one to point to the next instruction to be executed,

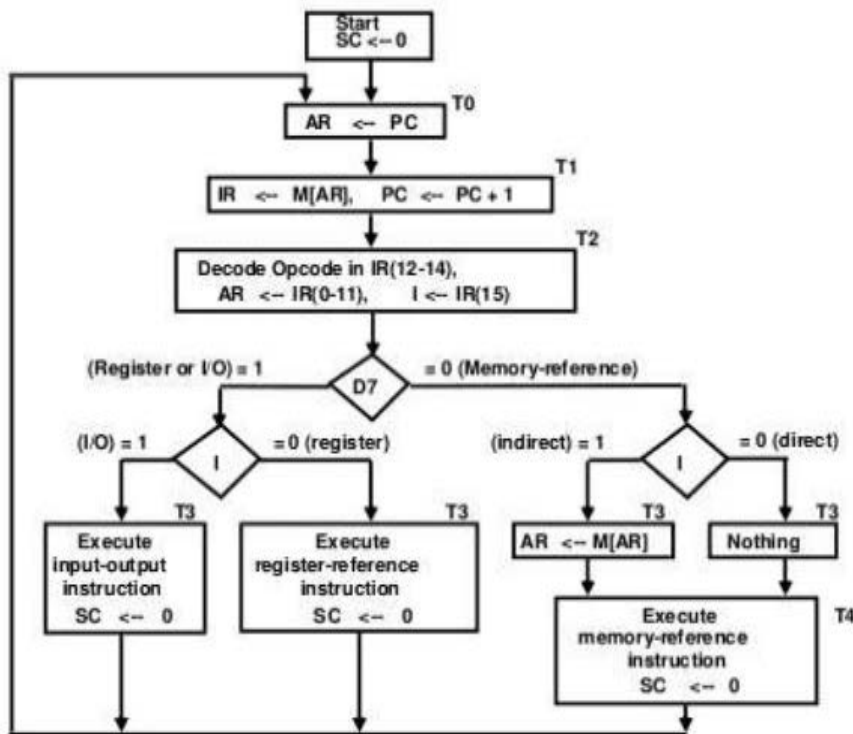


Figure 91: Instruction Cycle

## 7.3 REGISTER REFERENCE INSTRUCTIONS

As discussed in the above section, we can recognize register reference instructions when:  $D7=1$  and the mode selection bit  $I=0$

**Register-Reference Instructions (OP-code = 111, I = 0)**



Figure 92: Register Reference Instruction

In register reference instructions, memory reference is not required since the operand is present in the register. Hence, the 12 bits of the address part of the instruction code can be used to specify one of the various register operations. Various register reference instructions are shown in the Table 12 below.

$$r = D7 I \quad \square T3 \Rightarrow \text{Register Reference Instruction}$$

For convenience,  $D7 I \quad \square T3$  is represented as  $r$ . It represents a register reference instruction occurs when  $D7$  is 1,  $I$  is 0 during clock transition  $T3$ .  $B$  denotes  $i$ th bit of address part of the instruction code. For instance,

B0 represents 0000 0000 0001

*B1 represents 0000 0000 0010*

.

.

.

*B11 represents 1000 0000 0000 1115*

The first element in the table 9 is rB11, which can be represented in binary as rB11 =>D7 I T3B11 => T CLA

CLA is clear accumulator, which clears all the bits of the accumulator to 0. Likewise there is different register reference instructions operation explained in Table 12 below.

**Table 12 : Register Reference Instructions**

Symbol	Operational Decoder	Symbolic Instruction
	r:	SC←0
<b>CLA</b>	rB <sub>11</sub> :	AC←0
<b>CLE</b>	rB <sub>10</sub> :	E←0
<b>CMA</b>	rB <sub>9</sub> :	AC←AC'
<b>CME</b>	rB <sub>8</sub> :	E←E
<b>CIR</b>	rB <sub>7</sub> :	AC←shr AC, AC(15) ←E, E←AC(0)
<b>CIL</b>	rB <sub>6</sub> :	AC←shr AC, AC(0) ←E, E←AC(15)
<b>INC</b>	rB <sub>5</sub> :	AC←AC+1
<b>SPA</b>	rB <sub>4</sub> :	If(AC(15)=0) then (PC←PC+1)
<b>SNA</b>	rB <sub>3</sub> :	If(AC(15)=1) then (PC←PC+1)
<b>SZA</b>	rB <sub>2</sub> :	If(AC = 0) then (PC←PC+1)
<b>SZE</b>	rB <sub>1</sub> :	If( E = 0) then (PC←PC+1)
<b>HLT</b>	rB <sub>0</sub> :	S←0(S is a start-stop flip-flop)

---

## 7.4 MEMORY REFERENCE INSTRUCTIONS

---

Memory Reference Instruction can be recognized by either of D0 to D6=1. If I=0 then direct memory reference and if I=1 then indirect memory instruction. Various memory reference instructions can be selected by varying the values of opcode bits, which ultimately set one of seven

outputs from D0 to D6 equal to 1. This could be summarized by Table 13 below:

**Table 13 : Memory Reference Instructions**

Symbol	Operational Decoder	Symbolic Instruction
AND	D <sub>0</sub>	$AC \leftarrow AC \wedge M[AR]$
ADD	D <sub>1</sub>	$AC \leftarrow AC + M[AR]$ . $E \leftarrow C_{out}$
LDA	D <sub>2</sub>	$AC \leftarrow M[AR]$
STA	D <sub>3</sub>	$M[AR] \leftarrow AC$
BUN	D <sub>4</sub>	$PC \leftarrow AR$
BSA	D <sub>5</sub>	$M[AR] \leftarrow PC$ , $PC \leftarrow AR + 1$
ISZ	D <sub>6</sub>	$M[AR] \leftarrow M[AR] + 1$ , if $M[AR] + 1 = 0$ then $PC \leftarrow PC + 1$

The effective address of the instruction is in AR and was placed there during timing signal T2 when I = 0, or during timing signal T3 when I = 1. The execution of memory reference instruction starts with T4.

Now let us discuss the various memory reference instructions one by one.

(i) AND to AC

D<sub>0</sub>T<sub>4</sub>:  $DR \leftarrow M[AR]$  Read operand

D<sub>0</sub>T<sub>5</sub>:  $AC \leftarrow AC \wedge DR$ ,  $SC \leftarrow 0$  AND with AC

During T4, when D0 is set, the content of main memory, located at the address specified by Address Register(AR) are transferred to Data Register(DR). Now at T5, the content of DR and ANDED with content of AC and the final result is stored in AC. Once the operation is complete, SC is cleared so as to generate T0 again, which will fetch next instruction to CPU for processing.

(ii) ADD to AC

D<sub>1</sub>T<sub>4</sub>:  $DR \leftarrow M[AR]$  Read operand

D<sub>1</sub>T<sub>5</sub>:  $AC \leftarrow AC + DR$ ,  $E \leftarrow C_{out}$ ,  $SC \leftarrow 0$  Add to AC and store carry in E

During T4, when D1 is set, the content of main memory, located at the address specified by Address Register(AR) are transferred to Data Register(DR). Now at T5, the content of DR and added with content of AC and the final result is stored in AC. If a carry is generated, E is set. Once the operation is complete, SC is cleared so as to generate T0 again, which will fetch next instruction to CPU for processing.

(iii) LDA: Load to AC

D<sub>2</sub>T<sub>4</sub>:  $DR \leftarrow M[AR]$

D<sub>2</sub>T<sub>5</sub>:  $AC \leftarrow DR$ ,  $SC \leftarrow 0$

In the previous operations, the operand is added to the content of the accumulator. One may think how the accumulator is loaded with data. The above instruction LDA is used to load the accumulator. During T4, when D2 is set, the content of main memory, located at the address specified by Address Register(AR) are transferred to Data Register(DR). Now at T5, the content of DR are transferred to Accumulator(AC) and the SC is cleared.

(iv) STA: Store AC

$D_3T_4: M[AR] \leftarrow AC, SC \leftarrow 0$

When a new operand is to be loaded in the accumulator and AC already contains some data, which can be further required. This data is stored in the memory for future use. During T4, when D3 is set, the content of AC are transferred to main memory at the address specified by Address Register(AR). Once the operation is complete, SC is cleared so as to generate T0 again, which will fetch next instruction to CPU for processing.

(v) BUN: Branch Unconditionally

$D_4T_4: PC \leftarrow AR, SC \leftarrow 0$

During execution of an instruction, if a branching instruction is encountered, the program sequence branch to the address specified by the address register. Since it is an unconditional branching, therefore no condition is to be checked before branching. During T4, when D4 is set, the content of AR are transferred to PC and SC is cleared so as to generate T0 again, which will fetch next instruction to CPU for processing from the address that is specified by PC i.e. the branching address that was loaded to PC from AR.

(vi) BSA: Branch and Save Return Address

There are often situations in programming when a subroutine is called. The process of calling a subroutine is as follows: As soon as subroutine is called, the address of the starting location of the subroutine is written into the PC. But after the subroutine is called and executed, the control needs to be transferred to the location where the subroutine was called. Therefore the address of the location where a subroutine was called need to be remembered. To facilitate this, the first location of the subroutine is always kept empty so that the returning address can be saved at that location. This can be explained with the help of a register transfer statement:

There are often situations in programming when a subroutine is called. The process of calling a subroutine is as follows: As soon as subroutine is called, the address of the starting location of the subroutine is written into the PC. But after the subroutine is called and executed, the control needs to be transferred to the location where the subroutine was called. Therefore the address of the location where a subroutine was called need

to be remembered. To facilitate this, the first location of the subroutine is always kept empty so that the returning address can be saved at that location. This can be explained with the help of a register transfer statement:

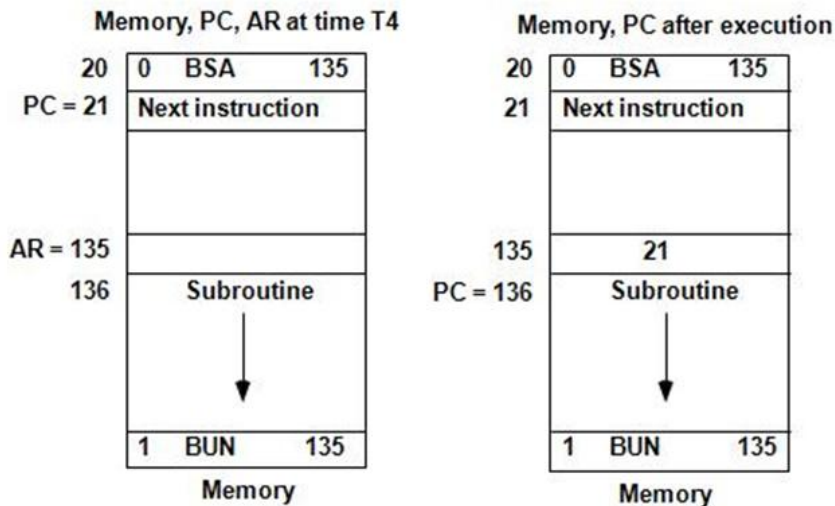
$$D_5T_4: M[AR] \leftarrow PC, PC \leftarrow AR + 1$$

At T5, when D5 is active, the content of the Program Counter(PC), which holds the returning address, are saved in the memory location specified by AR( it is the starting address of the subroutine which is always kept empty). Simultaneously, The PC is updated with the starting address of the subroutine( remember, the starting address of the subroutine is specified by AR. Since, the first location is empty and used for storing the starting address, therefore content of AR are incremented before updating the PC).

This can be explained with the help of an example. The CPU is currently processing the instruction location at 20. Therefore, the current value of PC will be 21. While decoding the instruction, it was found that it was a Branch and Save instruction. The branching address is given by the address part of the Instruction Code i.e. 135. The AR currently stores this address. The starting location of the subroutine at 135 is empty. This is used to store the returning address, i.e. 21, which is stored in PC. The current content of the PC are written into the first location, i.e. 135 of the subroutine. It is followed by updating the PC with incremented content of AR i.e. 135+1=136. This could be explained as follows:

$$D_5T_4: M[135] \leftarrow 21, PC \leftarrow 135 + 1$$

Now the instruction located at location 136 i.e. subroutine starting address is loaded into the memory. The subroutine is executed. The last line of the subroutine contains an instruction which unconditionally branch to the starting location of the subroutine, i.e. 135. Please note that the mode selection bit I is 1. Therefore, it is an indirect addressing mode. Hence, the control returns to location number 21.



**Figure 96: Branch and Save Address**

It is not possible to perform the operation of the BSA instruction in one clock cycle when we use the bus system of the basic computer. To use the memory and the bus properly, the BSA instruction must be executed with a sequence of two microoperations. This is expressed in RTL as follows:

$$D_5T_4: M[AR] \leftarrow PC, AR \leftarrow AR + 1$$

$$D_5T_5: PC \leftarrow AR, SC \leftarrow 0$$

Or

$$D_5T_4: M[135] \leftarrow 21(PC), 136(AR) \leftarrow 135 + 1$$

$$D_5T_5: 136(PC) \leftarrow 136(AR), SC \leftarrow 0$$

Timing signal T4 initiates a memory write operation, places the content of PC onto the bus, and enables the INR input of AR. The memory write operation is completed and AR is incremented by the time the next clock transition occurs. The bus is used at T5 to transfer the content of AR to PC.

(vii) ISZ: Increment and Skip-if-Zero

There are some situations when before branching some condition is to be checked. If the condition is true then proceed with branching else fetch the next instruction from the sequence. Once such instructions is ISZ, increment and skip if zero. At T4, when D6 is set, the data is fetched from the memory from the location specified by the AR. The data from memory is transferred to DR. At T5, the content of DR is incremented. Now at T6, if after incrementing the content of DR, the incremented content of DR becomes zero, then skip the next instruction.

$$D_6T_4: DR \leftarrow M[AR]$$

$$D_6T_5: DR \leftarrow DR + 1$$

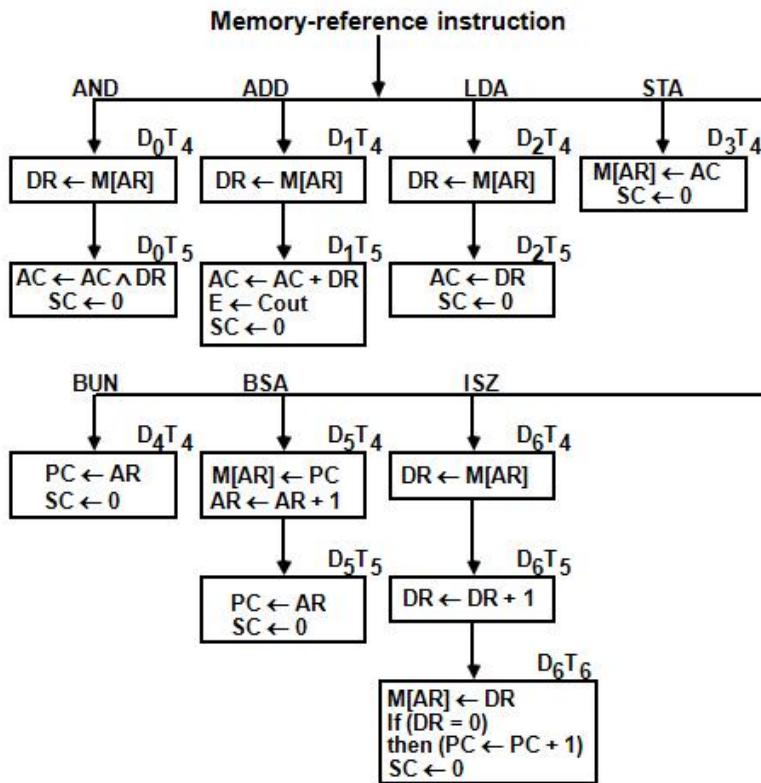
$$D_6T_6: M[AR] \leftarrow DR, \text{ if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$$

We can skip one instruction by incrementing the PC twice. Once the PC is incremented during T1, now if we again increment it at T6, it is equivalent to skipping one instruction. Also, the content of DR are saved to main memory and the SC is cleared.

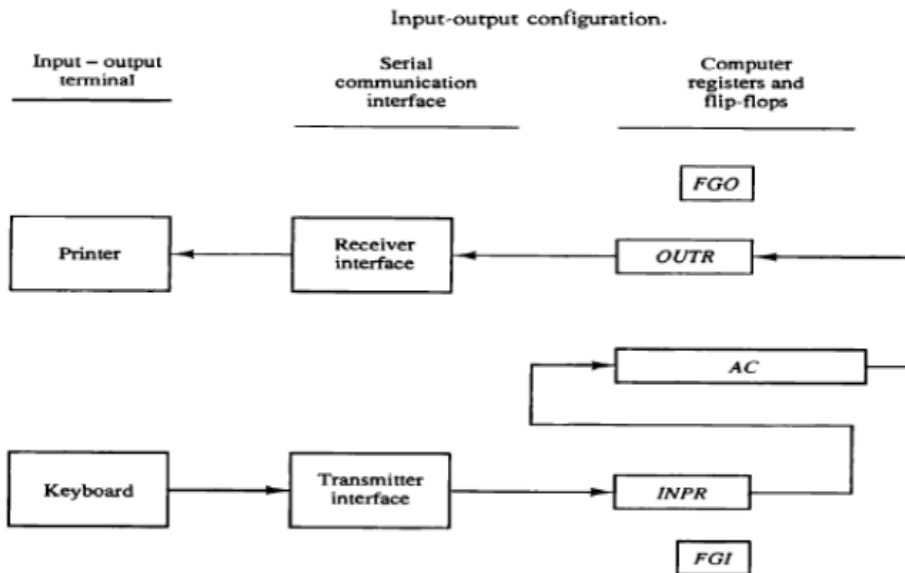
The operation of Memory Reference Instruction could be summarized with the help of a flowchart given at Figure 97.

Can you think of a computer which cannot communicate with its environment? Most of us will have the same answer, NO. A computer does not have any practical application if it cannot interact with its environment via input and output devices. The input devices like keyboard, scanner, etc. are used to feed input/data to the CPU or memory and the output devices like monitor, printer are used to display the processed information. The basic input-output configuration of a computer is explained with the help of Figure 98





**Figure 98: Input-Output Configuration**



**Figure 99: Input-Output Configuration**

The input/output devices receives and transmits data at much slower rate as compared to CPU. To compensate this speed mismatch, an input and output interfaces are required, which receives and transmit the data at different rates. Input devices are attached to input interface, which is connected to INPR register of CPU. INPR is able to receive the data from

input device serially and when the whole of the data item is received, it can transmit it to the CPU parallelly. Whenever we press an key in the keyboard, it transmit the 8-bit alphanumeric code to INPR serially. INPR is attached to a 1-bit status flag FGI. This FGI bit represents the status of the INPR, whether it is ready to accept data or not. We can imagine a situation when the INPR already holds previous data, which is not yet transferred to CPU. If another key in the keyboard is stiked, the alphanumeric bits of the new data can currrupt the old data. To overcome this situation, FGI status flag is used. FGI is set when INPR already holds the data. In case, a new key is pressed, its alphanumeric input are not transmitted to INPR until and unless flag FGI is 0, which is a signal that the INPR is empty and ready to store new data. Whenever the INPR transmits the data to AC, it clears FGI.

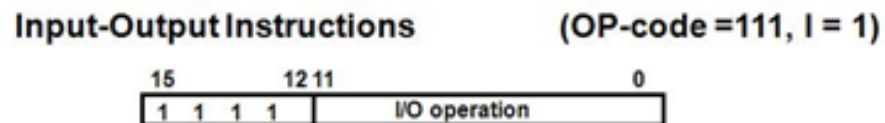
Similarly, output devices are attahed to output interface, which is connected to OOUTR of CPU. OOUTR have the capability to receive the data from CPU parallelly and transmit it to the output devices serially. Status flag FGO in set state is used to represent the OOUTR is ready to receive data from AC. Once the data from data is transmitted from AC to OOUTR, FGO is cleared to 0. Now once the data from OOUTR is transferred to output device, again FGO is set.

---

## 7.5 INPUT-OUTPUT REFERENCE INSTRUCTIONS

---

As discussed in the previous unit, we can recognize register reference instructions when:  $D7=1$  and the mode selection bit  $I=1$ .



**Figure 100: Input-Output Reference Instruction**

In input-output reference instructions, memory reference is not required since the operand is present in the register. Hence, the 12 bits of the address part of the instruction code can be used to specify one of the various input-output operations. Various input-output reference instructions are shown in the table 11 below.

$$p = D_7 I T_3 \Rightarrow \text{Input-Output Reference Instruction}$$

For convenience,  $D_7 I T_3$  is represented as  $p$ . It represent that input-output reference instruction occurs when  $D_7$  is 1,  $I$  is 1 during clock transition  $T_3$ .  $B$  donotes ith bit of address part of the instruction code. For instance,

B6 represents 0000 0010 0000

B7 represents 0000 0100 0000

.

.

.

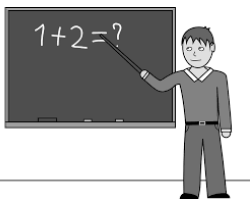
B11 represents 1000 0000 0000

The first element in the table 11 is  $pB_{11}$ , which can be represented in binary as

$pB_{11} \Rightarrow D_7 I T_3 B_{11} \Rightarrow T_3: D_7 I B_{11} \Rightarrow 1111 1000 0000 0000 \Rightarrow INP$

INP transfers the content of INPR to AC. After transferring the content, FGI is cleared. Likewise there is different register reference instructions operation explained in Table 14 below.

<i><math>D_7 I T_3 = p</math> (common to all input-output instructions)</i>			
<i><math>IR(i) = B_i</math> (Bit in IR(6-11) that specifies the instruction)</i>			
	<b>p:</b>	<b>SC ← 0</b>	<b>Clear SC</b>
<b>INP</b>	<b><math>pB_{11}</math>:</b>	<b>AC(0-7) ← INPR, FGI ← 0</b>	<b>Input Character</b>
<b>OUT</b>	<b><math>pB_{10}</math>:</b>	<b>OUTR ← AC(0-7), FGO ← 0</b>	<b>Output Character</b>
<b>SKI</b>	<b><math>pB_9</math>:</b>	<b>If(FGI=1) then (PC ← PC+1)</b>	<b>Skip on Input flag</b>
<b>SKO</b>	<b><math>pB_8</math>:</b>	<b>If(FGO=1) then (PC ← PC+1)</b>	<b>Skip on Output flag</b>
<b>ION</b>	<b><math>pB_7</math>:</b>	<b>IEN ← 1</b>	<b>Interrupt enable on</b>
<b>IOF</b>	<b><math>pB_6</math>:</b>	<b>IEN ← 0</b>	<b>Interrupt enable off</b>



## Check Your Progress

1. In \_\_\_\_\_ reference instructions, memory reference is not required since the operand is present in the register.
2. The input/output devices receives and transmits data at \_\_\_\_\_ rate as compared to CPU.
3. To compensate the speed mismatch, an input and output \_\_\_\_\_ are required, which receives and transmit the data at different rates.
4. In \_\_\_\_\_ reference instructions, memory reference is not required since the operand is present in the register.

---

## 7.6 SUMMARY

---

Instructions are processed under the direction of the control unit in a step-by-step manner. There are four fundamental steps in the instruction cycle:

### 1. Fetch the instruction

- The next instruction is fetched from the memory address that is currently stored in the Program Counter (PC), and stored in the Instruction register (IR). At the end of the fetch operation, the PC points to the next instruction that will be read at the next cycle.

### 2. Decode the instruction

- The decoder interprets the instruction. During this cycle the instruction inside the IR (instruction register) gets decoded.

### 3. Execute

- The Control Unit of CPU passes the decoded information as a sequence of control signals to the relevant function units of the CPU to perform the actions required by the instruction such as reading values from registers, passing them to the ALU to perform mathematical or logic functions on them, and writing the result back to a register. If the ALU is involved, it sends a condition signal back to the CU.

### 4. Store result

- The result generated by the operation is stored in the main memory, or sent to an output device. Based on the condition of any feedback from the ALU, Program Counter may be updated to a different address from which the next instruction will be fetched.

---

## 7.7 ANSWERS TO CHECK YOUR PROGRESS

---

1. Program
2. Program Counter(PC)
3. Register
4. Slower
5. Interfaces
6. input-output

---

## 7.8 TERMINAL QUESTIONS

---

1. What is the difference between a direct and an indirect address instruction? How many references to memory are required for each type of instruction to bring an operand into a processor register?
2. What is instruction cycle? What are the sub-phases of an instruction cycle.
3. Explain ISZ memory reference instruction in detail.
4. Explain BSA memory reference instruction in detail.
5. What is an interface? Why is it required?



# UNIT-8

---

## DESIGN OF BASIC COMPUTER

---

### Structure

- 8.0 Learning Objectives
- 8.1 Introduction
- 8.2 Basic Computer
  - 8.2.1 Control Logic Gates
  - 8.2.2 Control of Registers and Memory
  - 8.2.3 Control of Single Flip-flops
  - 8.2.4 Control of Common Bus
- 8.3 Design of Accumulator Logic
  - 8.3.1 Control of AC Register
  - 8.3.2 Adder and Logic Circuit
- 8.4 Summary
- 8.5 Answers to Check Your Progress
- 8.6 Terminal Questions

---

### 8.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Understand the design of a basic computer.
- Understand the functioning of the hardwired control unit.
- Define control logic gates.
- Explain the control of register and memory.
- Explain the functioning of a common bus.
- Know the design of accumulator logic.

---

### 8.1 INTRODUCTION

---

The control unit (CU)<sup>34</sup> is a component of a computer's central processing unit (CPU) that directs the operation of the processor. It tells the computer's memory, arithmetic/logic unit and input and output devices on how to respond to a program's instructions. It directs the operation of

the other units by providing timing and control signals. Most computer resources are managed by the CU. It directs the flow of data between the CPU and the other devices. John von Neumann included the control unit as part of the von Neumann architecture. In modern computer designs, the control unit is typically an internal part of the CPU with its overall role and operation unchanged since its introduction. The Control Unit (CU) is digital circuitry contained within the processor that coordinates the sequence of data movements into, out of, and between a processor's many sub-units. The result of these routed data movements through various digital circuits (sub-units) within the processor produces the manipulated data expected by a software instruction (loaded earlier, likely from memory). It controls (conducts) data flow inside the processor and additionally provides several external control signals to the rest of the computer to further direct data and instructions to/from processor external destination's (i.e. memory). The Control Unit have two variants: hardwired control unit and microprogrammed control unit. In this unit, we will discuss the implementations of hardwired control unit in details.

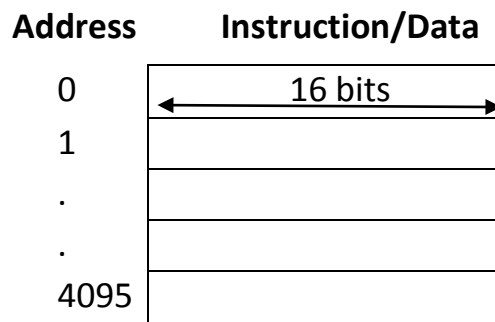
---

## 8.2 BASIC COMPUTER

---

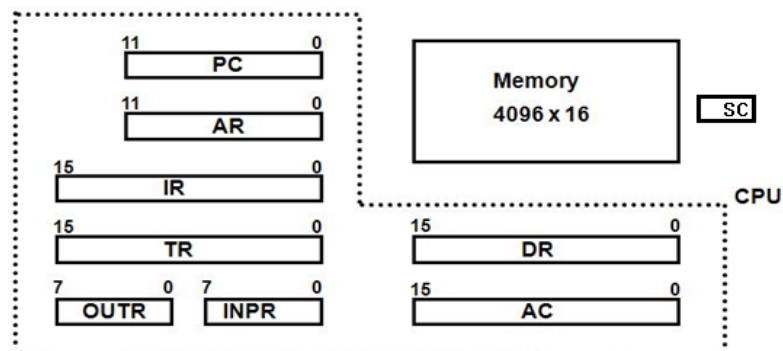
Now let us design a very basic computer, which does not solve any practical purpose but it is designed for ease of our understanding of the working of a basic computer. Its consists of the following hardware components:

1. A memory unit with 4096 words of 16 bits each



**Figure 101 : 4096 X 16 Memory**

2. Nine registers: AR, PC, DR, AC, IR, TR, OUTR, INPR, and SC



**Figure 102 : 4096 X 16 Memory**



3. Seven flip-flops: I, S, E, R, IEN, FGI, and FGO
4. Two decoders: a 3 x 8 operation decoder and a 4 x 16 timing decoder

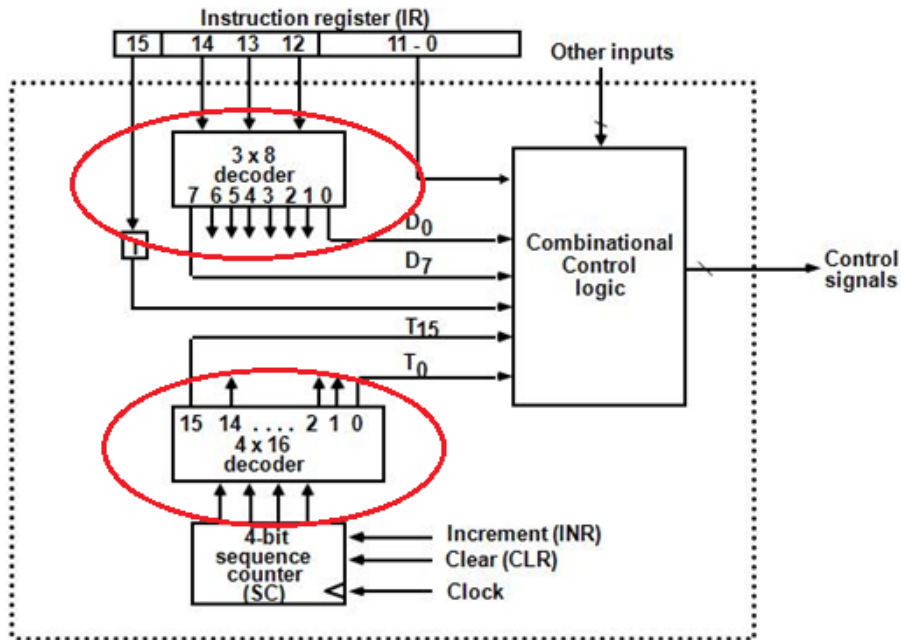


Figure 103: Hardwired implementation of CPU denoting inputs to Control Logic

5. A 16-bit common bus.

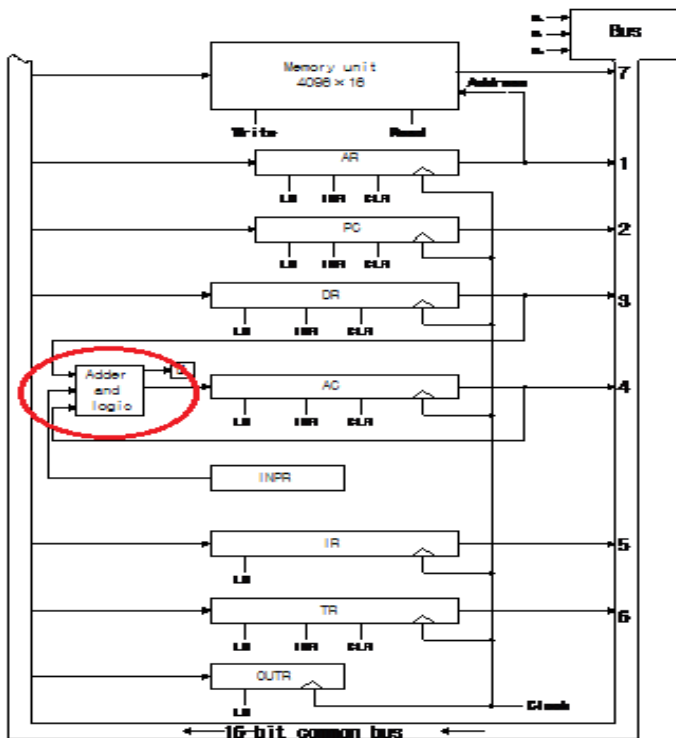
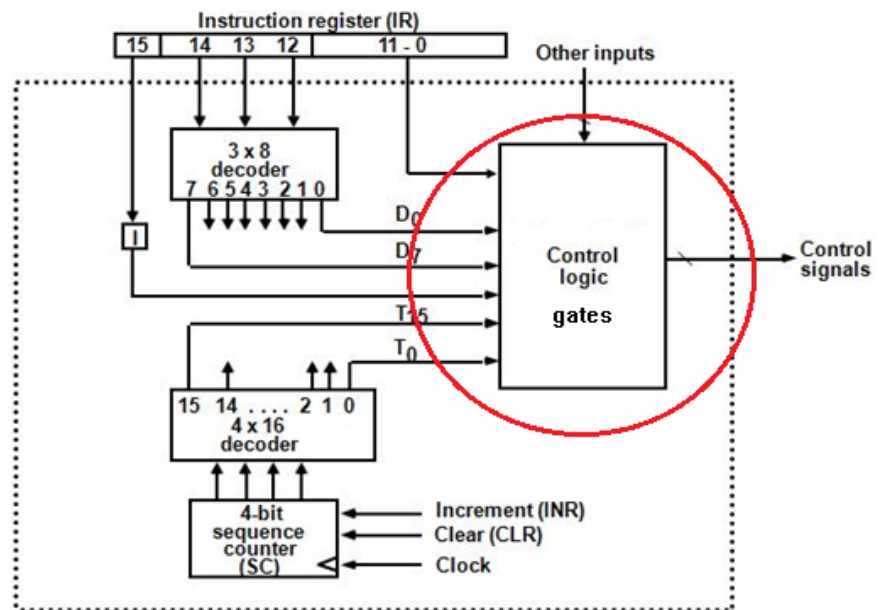


Figure 104: 16-Bit Common Bus Representing Adder and Logic Input to AC\

6. Control logic gates
7. Adder and logic circuit connected to the input of AC

### 8.2.1 Control Logic Gates



**Figure 105: Control Logic Gates**

The Control Logic Gates The control logic gates, as shown below in the hardwired implementation of the control unit, receives input from:

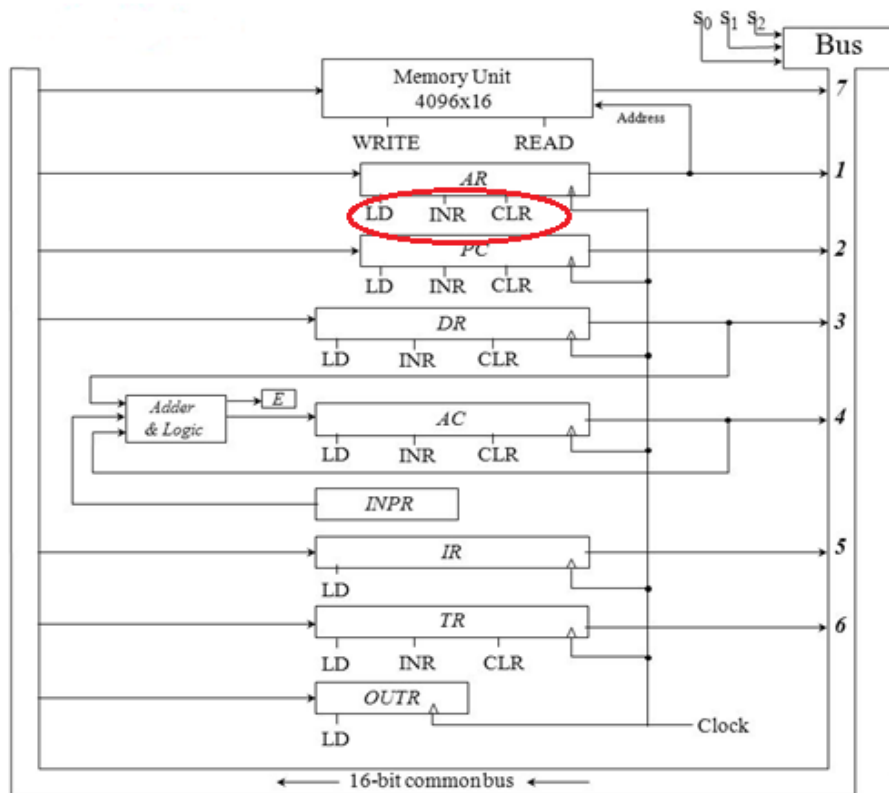
- a. 3 X 8 decoder
- b. 4 X 16 decoder
- c. 1 bit of instruction code
- d. Address part, bit(0-11), of the instruction code
- e. AC bits(0-15) to check whether AC=0 and to detect the sign bit in AC(15)
- f. DR bits(0-15) to check whether DR=0
- g. Values of seven flip-flops

The outputs of the control logic circuit are:

- a. Signals to control the inputs of the nine registers
- b. Signals to control the read and write inputs of memory
- c. Signals to set, clear, or complement the flip-flops
- d. Signals for S2, S1 and S0 to select a register for the bus
- e. Signals to control the AC adder and logic circuit

## 8.2.2 Control of Registers and Memory

We can clearly see below the 16-bit common bus architecture. We can observe that all the CPU registers are attached to this 16-bit common bus. These registers are controlled through the control inputs LD (load), INR (increment), and CLR (clear).



**Figure 106: 16-Bit common bus showing LD, INR and CLR Inputs to Registers 130**

Now we will study the gate structure associated with the control inputs of AR. For this, we need to explore Table 12 to find all the statements that change the content of AR:

$R'T0: AR \leftarrow PC$

$R'T2: AR \leftarrow IR(0-11)$

$D7'IT3: AR \leftarrow M[AR]$

$RT0: AR \leftarrow 0$

$D5T4: AR \leftarrow AR + 1$

**Table 4 : Control Functions and Microoperations for the Basic Computer**

Fetch	R'T <sub>0</sub> : R'T <sub>1</sub> :	AR←PC IR ←M[AR], PC ←PC+1
Decode	R'T <sub>2</sub> :	D <sub>0</sub> .....D <sub>7</sub> ← Decode IR(12-14) AR←IR(0-11) , I←IR(15)
Indirect	D <sub>7</sub> IT <sub>3</sub> :	AR←M[AR]
<b>Interrupt:</b>		
	T <sub>0</sub> T <sub>1</sub> T <sub>2</sub> (IEN)(FGI+FGO)	R←1
		RT <sub>0</sub> : AR←0, TR←PC
		RT <sub>1</sub> : M[AR] ←TR, PC←0
		RT <sub>2</sub> : PC←PC+1, IEN←0, R←0, SC←0
<b>Memory-reference:</b>		
AND	D <sub>0</sub> T <sub>4</sub> D <sub>0</sub> T <sub>5</sub>	DR←M[AR] AC←AC^DR, SC←0
ADD	D <sub>1</sub> T <sub>4</sub> D <sub>1</sub> T <sub>5</sub>	DR←M[AR] AC←AC+DR. E←C <sub>out</sub> ,SC←0
LDA	D <sub>2</sub> T <sub>4</sub> D <sub>2</sub> T <sub>5</sub>	DR←M[AR] AC←DR, SC←0
STA	D <sub>3</sub> T <sub>4</sub>	M[AR] ←AC, SC←0
BUN	D <sub>4</sub> T <sub>4</sub>	PC←AR, SC←0
BSA	D <sub>5</sub> T <sub>4</sub> D <sub>5</sub> T <sub>5</sub>	M[AR] ←PC, AR←AR+1 PC←AR, SC←0
ISZ	D <sub>6</sub> T <sub>4</sub> D <sub>6</sub> T <sub>5</sub> D <sub>6</sub> T <sub>6</sub>	DR←M[AR] DR←DR+1 M[AR] ←DR, if(DR=0) then (PC←PC+1), SC←0
<b>Register- reference:</b>		
D <sub>7</sub> I'T <sub>3</sub> =r(common to all register-reference instructions)		
IR(i)=B <sub>i</sub> (i=0,1,2,.....,11)		
	r:	SC←0
CLA	rB <sub>11</sub> :	AC←0
CLE	rB <sub>10</sub> :	E←0
CMA	rB <sub>9</sub> :	AC←AC'
CME	rB <sub>8</sub> :	E←E
CIR	rB <sub>7</sub> :	AC←shr AC, AC(15) ←E, E←AC(0)
CIL	rB <sub>6</sub> :	AC←shr AC, AC(0) ←E, E←AC(15)
INC	rB <sub>5</sub> :	AC←AC+1
SPA	rB <sub>4</sub> :	If(AC(15)=0) then (PC←PC+1)
SNA	rB <sub>3</sub> :	If(AC(15)=1) then (PC←PC+1)
SZA	rB <sub>2</sub> :	If(AC = 0) then (PC←PC+1)
SZE	rB <sub>1</sub> :	If( E = 0) then (PC←PC+1)
HLT	rB <sub>0</sub> :	S←0
<b>Input-output:</b>		
D <sub>7</sub> IT <sub>3</sub> =p(common to all register-reference instructions)		
IR(i)=B <sub>i</sub> (i=6,7,8,9,10,11)		
	P:SC←0	
INP	pB <sub>11</sub> :	AC(0-7) ←INPR, FGI←0
OUT	pB <sub>10</sub> :	OUTR←AC(0-7), FGO←0

SKI	pB <sub>9</sub> :	If(FGI=1) then (PC←PC+1)
SKO	pB <sub>8</sub> :	If(FGO=1) then (PC←PC+1)
ION	pB <sub>7</sub> :	IEN←1
IOF	pB <sub>6</sub> :	IEN←0

In the above RTL statements, the first RTL statement transfers the content of PC to AR for instruction fetch at T<sub>0</sub>. The second statement, transfers bits(0-11) to AR at T<sub>2</sub>. Now at T<sub>3</sub>, the address fetch takes place from main memory to AR by enabling LD signal of AR. In the forth statement, AR is cleared by enabling CLR. Finally, the fifth statement increments AR by enabling INR control signal.

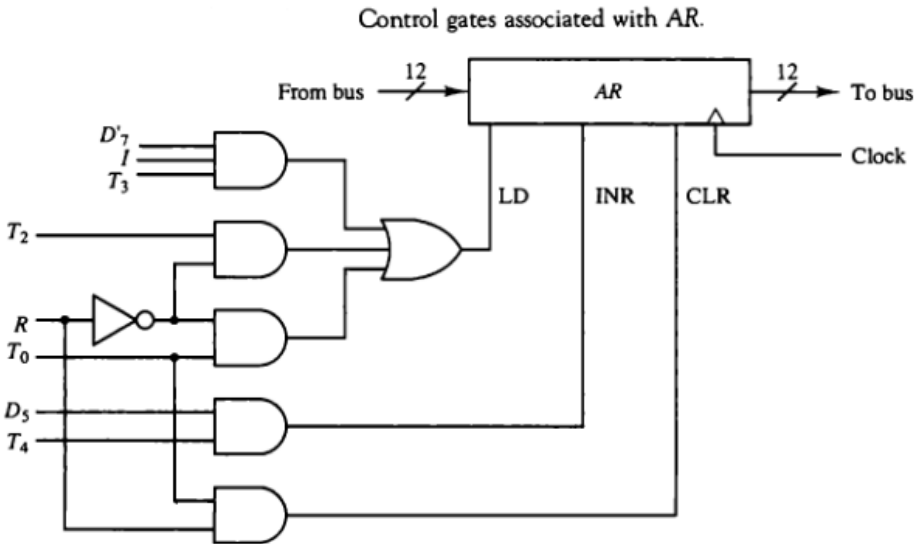
The control functions can be combined into three Boolean expressions as follows:

$$LD(AR) = R'T_0 + R'T_2 + D_7IT_3$$

$$CLR(AR) = RT_0$$

$$INR(AR) = D_5T_4$$

The above could be implemented using circuit shown below in Figure 107



**Figure 107 : Control Gates associated with AR**

### 8.2.3 Control of Single Flip-flops

The control gates for the seven flip-flops can be determined in a similar manner. For Example- IEN may change as a result of the two instructions ION and IOF.

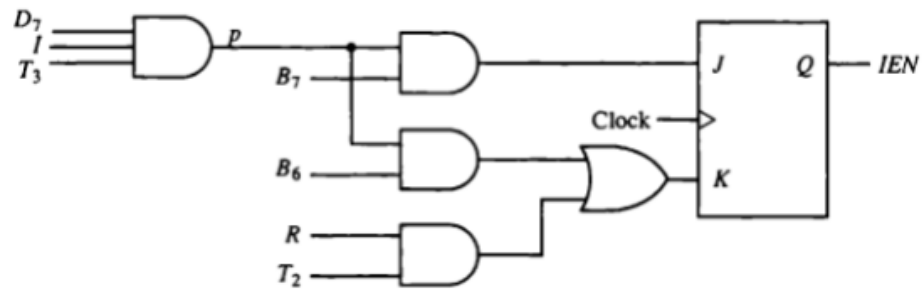
$$pB_7: IEN \leftarrow 1$$

$$pB_6: IEN \leftarrow 0$$

where p = D<sub>7</sub>I<sub>3</sub> and B<sub>7</sub> and B<sub>6</sub> are bits 7 and 6 of IR, respectively. Moreover, at the end of

the interrupt cycle IEN is cleared to 0.

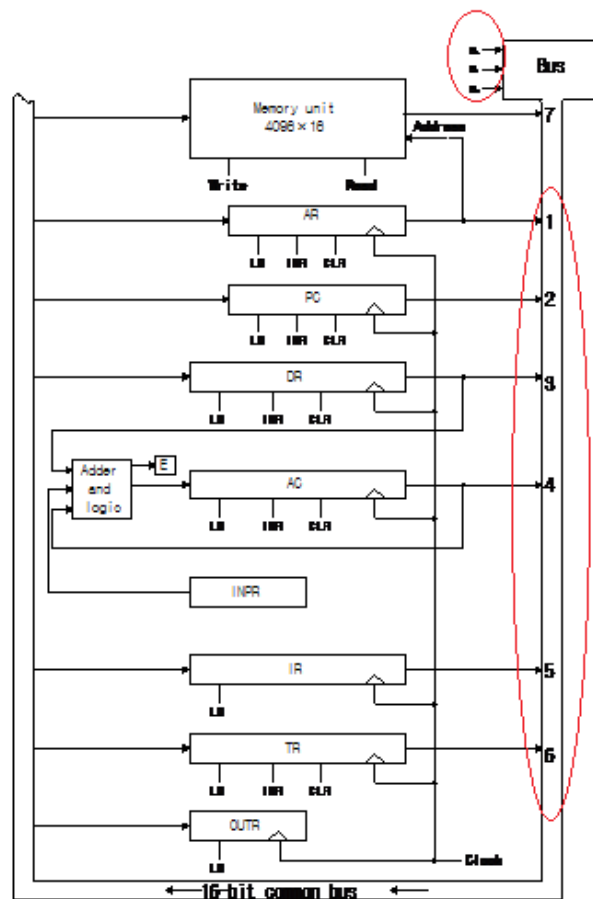
$RT2: IEN \leftarrow 0$



**Figure 108: Control Inputs for IEN**

### 8.2.4 Control of Common Bus

Let us now discuss the control of a 16-bit common bus. It contains three selection inputs S0, S1 and S2 through which it can select one of the seven register (AR, DR, AC, INPR, OUTR, TR, IR) and memory.



**Figure 109: 16-Bit Common Bus denoting Registers and Decimal Equivalent of Registers**

The decimal value of each of the register is shown in the figure. The combination of S0,S1 and S2 decides which register will be selected at any point of time. We have assigned a binary number for S0,S1 and S2 that select each register. Please refer table 13 for details, which represent the truth table of a binary encoder. Each binary number is associated with a Boolean variable x1 through x7, corresponding to the gate structure that must be active in order to select register or memory for the bus. The Boolean functions for the encoder are

$$S0 = x1 + x3 + x5 + x7$$

$$S1 = x2 + x3 + x6 + x7$$

$$S2 = x4 + x5 + x6 + x7$$

**Table 5 : Encoder for Bus Selection Circuit**

Inputs							Outputs			Register select for bus
X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	X <sub>6</sub>	X <sub>7</sub>	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	
0	0	0	0	0	0	0	0	0	0	None
1	0	0	0	0	0	0	0	0	1	AR
0	1	0	0	0	0	0	0	1	0	PC
0	0	1	0	0	0	0	0	1	1	DR
0	0	0	1	0	0	0	1	0	0	AC
0	0	0	0	1	0	0	1	0	1	IR
0	0	0	0	0	1	0	1	1	0	TR
0	0	0	0	0	0	1	1	1	1	Memory

Now let us determine the logic for encoder input. But to do this, we have to find out the control functions that place the content of the corresponding register onto the 16-bit common bus. Let us find out all the logic from table 14 that makes x1=1. These statements are:

$$D4T4: PC \leftarrow AR$$

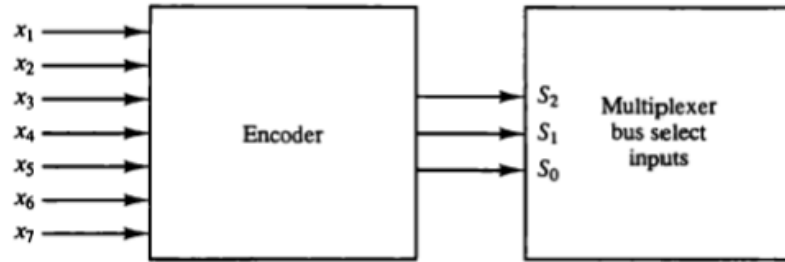
$$D5T5: PC \leftarrow AR$$

Therefore, the Boolean function for x1 is:

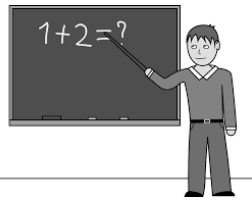
$$x1 = D4T4 + D5T5$$

Now let us derive the Boolean function for memory read operation. We know from our knowledge from previous units that, memory is selected when S0,S1 and S2 are 111 and x7=1. Also the logic gate that generates x7 must also be applied to read control of the memory. Therefore the Boolean function for memory read is:

$$x_7 = R'T_1 + D_7'IT_3 + (D_0 + D_1 + D_2 + D_6)T_4$$



**Figure 110 : Encoder for Bus Selection Inputs**



## Check Your Progress

1. The \_\_\_\_\_ is digital circuitry contained within the processor that coordinates the sequence of data movements into, out of, and between a processor's many sub-units.
2. In \_\_\_\_\_ control unit, the control unit uses fixed logic circuits to interpret instructions and generate control signals from them.
3. Hardwired control units are \_\_\_\_\_ than microprogrammed designs.
4. Hardwired control units are implemented through use of \_\_\_\_\_, featuring a finite number of gates that can generate specific results based on the instructions that were used to invoke those responses.

## 8.3 DESIGN OF ACCUMULATOR LOGIC

Now let us design the control circuit for Accumulator. First of all, we have to find out the all the register transfer statements and extract all the statements that change the content of AC, from.

**Table 6 : Register Transfer Statements that changes the content of AC**

$D_0T_5:$	$AC \leftarrow AC \wedge DR$	AND with <i>DR</i>
$D_1T_5:$	$AC \leftarrow AC + DR$	Add with <i>DR</i>
$D_2T_5:$	$AC \leftarrow DR$	Transfer from <i>DR</i>
$pB_{11}:$	$AC(0-7) \leftarrow INPR$	Transfer from <i>INPR</i>
$rB_9:$	$AC \leftarrow \overline{AC}$	Complement
$rB_7:$	$AC \leftarrow shr\ AC, \quad AC(15) \leftarrow E$	Shift right
$rB_6:$	$AC \leftarrow shl\ AC, \quad AC(0) \leftarrow E$	Shift left
$rB_{11}:$	$AC \leftarrow 0$	Clear
$rB_5:$	$AC \leftarrow AC + 1$	Increment



From this list we can derive the control logic gates and the adder and logic circuit.

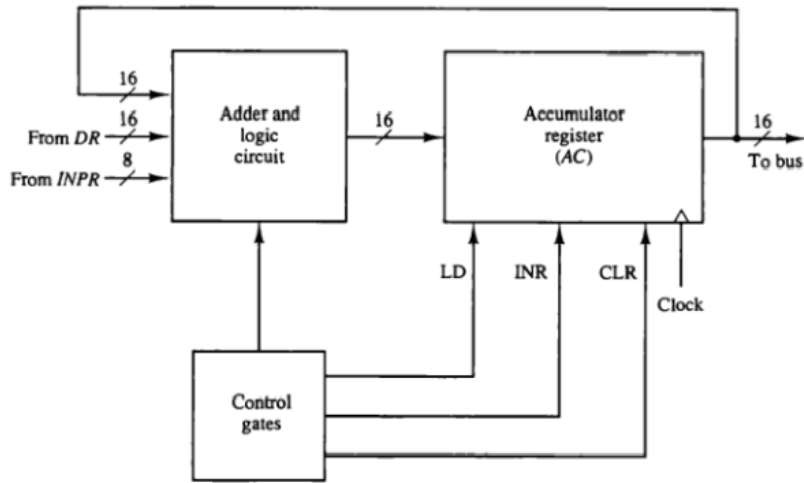


Figure 111 : Circuits associated with AC

### 8.3.1 Control of AC Register

The control function in the table 14 are used to derive the gate configuration that control the control signals viz. LD, INR and CLR inputs of AC. The circuit is shown in Figure 112 below:

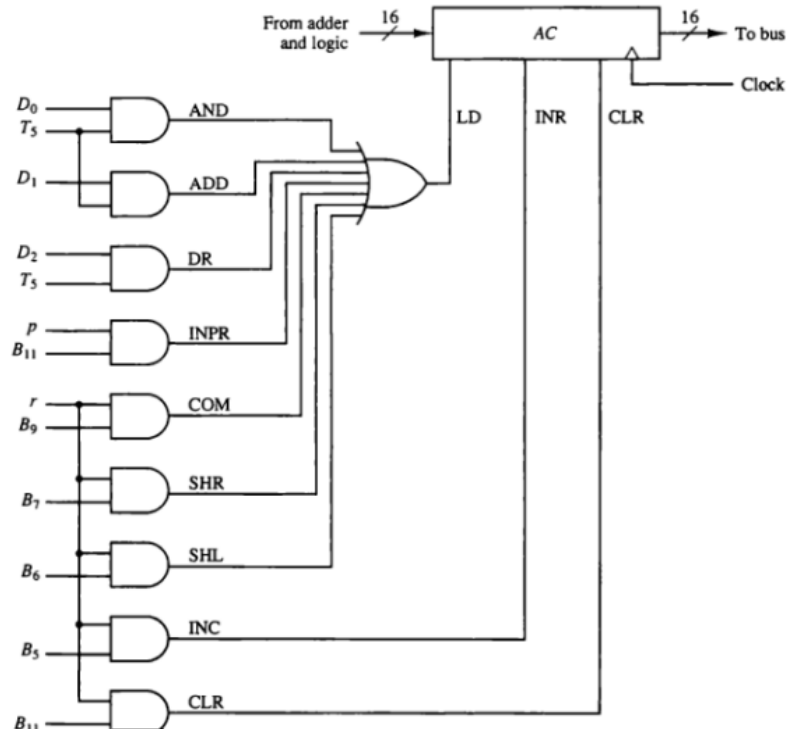


Figure 112: Gate Structure for Controlling the LD, INR and CLR of AC

### 8.3.2 Adder and Logic Circuit

To implement a 16-bit adder, 16 such unit, as shown in Figure 113 below, are required. Each such unit is required to add corresponding the  $i$ th bit of DR and AC. The one stage of the adder and logic circuit consists of AND gates, one OR gate and a fulladder(FA).

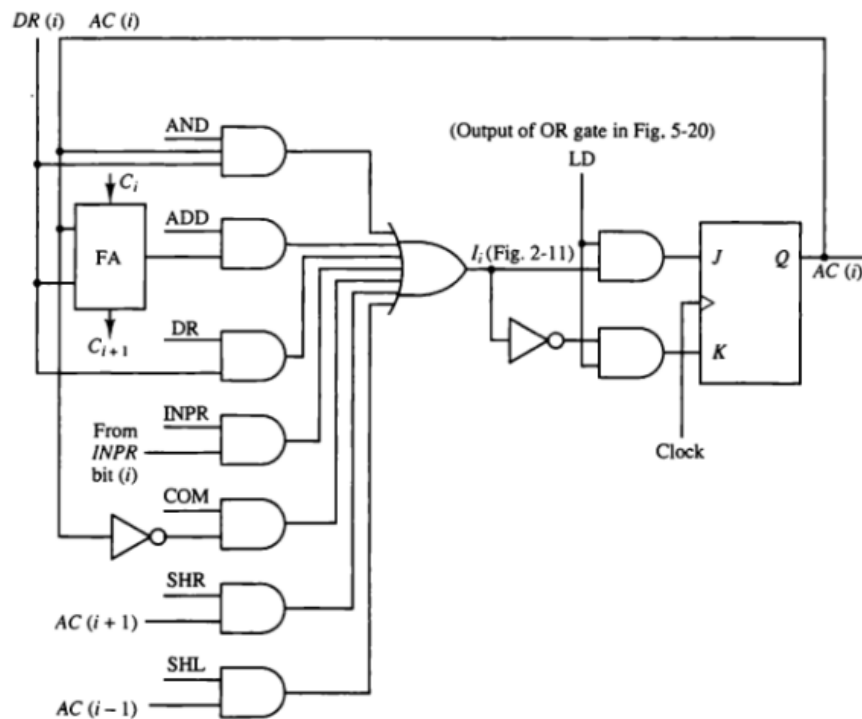


Figure 113: One Stage of Adder and Logic Circuit

---

## 8.4 SUMMARY

---

Hardwired control units are implemented through use of combinational logic units, featuring a finite number of gates that can generate specific results based on the instructions that were used to invoke those responses. Hardwired control units are generally faster than microprogrammed designs.

Their design uses a fixed architecture—it requires changes in the wiring if the instruction set is modified or changed. This architecture is preferred in reduced instruction set computers (RISC) as they use a simpler instruction set.

A controller that uses this approach can operate at high speed; however, it has little flexibility, and the complexity of the instruction set it can implement is limited.

The hardwired approach has become less popular as computers have evolved. Previously, control units for CPUs used ad-hoc logic, and they were difficult to design.

---

## **8.5 ANSWERS TO CHECK YOUR PROGRESS**

---

1. Control Unit(CU)
2. Hardwired
3. Master
4. Combinational logic units

---

## **8.6 TERMINAL QUESTIONS**

---

1. Draw the control gates associated with the program counter PC in the basic computer.
2. Draw the control gates for the write input of the memory in the basic computer.
3. Draw the control gates for the read input of the memory in the basic computer.
4. Explain with the help of a diagram one Stage of Adder and Logic Circuit.



# UNIT-9

---

## CENTRAL PROCESSING UNIT

---

### Structure

- 9.0 Learning Objectives
- 9.1 Introduction
- 9.2 General Register Organization
- 9.3 Summary
- 9.4 Answers to Check Your Progress
- 9.5 Terminal Questions

---

### 9.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Understand the functioning of a Central Processing Unit.
- Know the General Register Organization.
- Understand the operation of a multiplexer in generation of control signals.
- Understand the operation of a decoder in the selection of a register.

---

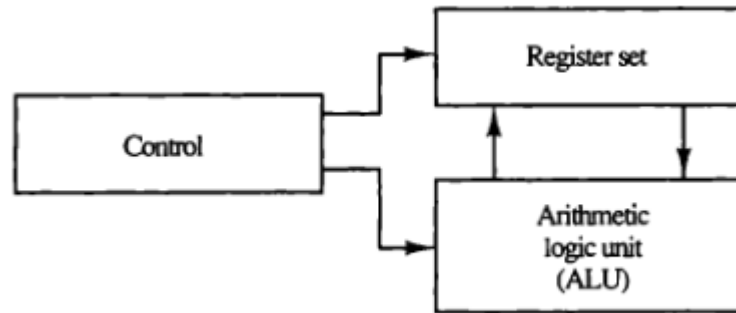
### 9.1 INTRODUCTION

---

Central Processing Unit (CPU)[1] , also known as the chip or processor, is the engine of the computer. It contains a control unit (CU) and arithmetic logic unit (ALU).

CU controls the order and sequence in which the program stored in memory will be executed. It also governs the movement of data in and out of the CPU via the data bus, which is connected to the computer's memory. The data bus moves data round in the CPU. CU also uses memory address bus to get memory addresses. The address bus is connected to the computer's memory as well, in the address section. ALU goes through logical operations (for example comparisons and arithmetic operations like addition).

Primary Memory (also known as Immediate Access Storage, IAS) includes the memory bus, data bus, cache, RAM and ROM. The primary memory stores data that the CU and ALU get. Some definitions of CPU include the primary 140



**Figure 114 : Major Components of CPU**

Now let us discuss the organization and architecture of the CPU. We will discuss how the register communicates among each other and the ALU through bus. Also we will briefly discuss the operation of memory stack.

---

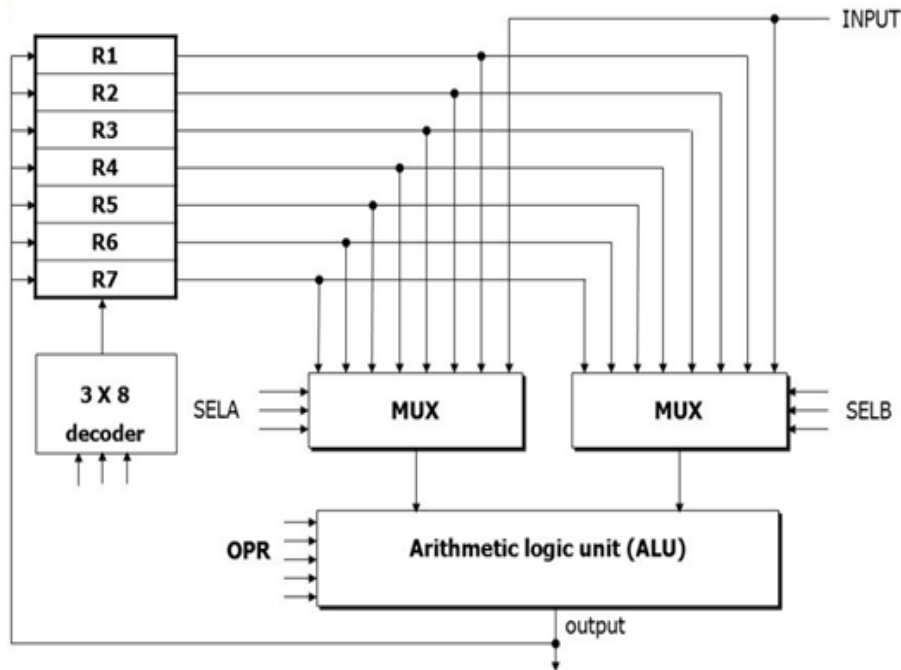
## **9.2 GENERAL REGISTER ORGANIZATION**

---

To facilitate the fast operation of a computer, CPU access to memory is minimized, as CPU access is expensive in terms of time. For this purpose, CPU registers are used, which are located within the CPU, known as register-set. This register-set is used to store the intermediately results of the operation, pointers, return addresses, etc and thus minimize the memory access which leads to faster operation. These registers are connected to the ALU via bus so as to facilitate the transfer of data during arithmetic, logic and shift operations. The detailed description of the operation can be explained using a diagram shown in Figure 115.

Figure 115 consists of two 8 X 1 multiplexers and one 3 X 8 decoder. There are seven general purpose registers labeled from R1 to R7. The output of each register is connected to the input of both the multiplexers to form two buses A and B. These multiplexers are used to provide input data to ALU for manipulation. The one of the seven register or the direct input line is selected as a source of operand with the help of selection lines SELA and SELB. OPR is 141

used to instruct which arithmetic/logic operation is to be performed by the ALU. Once the operation is complete, the SELD is used to select the destination register to store the intermediately result. The three bits of SELD can be used to select one of the destination register by enabling one of the seven load control of the registers. The data flow among the register within the CPU can be explained with the help of an example. Suppose we have to perform the following operation:



(a) Block Diagram

3	3	3	5
SELA	SELB	SELD	OPR

(b) Control Word

**Figure 115: Register Set with Common ALU**

$$R1 \leftarrow R2 + R3$$

To perform the above operation we need some control that must provide the binary selection variables for:

1. UX A selector (SELA): to place the content of R2 into bus A.
2. UX B selector (SELB): to place the content of R3 into bus B.
3. LU operation selector (OPR): to provide the arithmetic addition  $A + B$ .
4. decoder destination selector (SELD): to transfer the content of the output bus into R1.

The control unit is responsible for generating these 14 binary control selection variables (3 for SELA, 3 for SELB, 3 for SELD and 5 for OPR). The combination of these 14 binary control selection variables is known as *control word* (refer Figure 115(b)). These 14 binary control variables are divided into our fields. Based on the value of SELA and SELB (refer Table 18), R1, R2 and R3 are selected.

**Table 7 : Encoding of Register Select Field**

Binary Code	SELA	SELB	SELD
000	Input	Input	None
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

The OPR field is selected from Table 19. The ALU provides arithmetic and logic operations. In addition, the CPU must provide shift operations. The shifter may be placed in the input of the ALU to provide a preshift capability, or at the output of the ALU to provide postshifting capability.

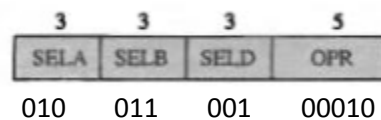
**Table 19: Encoding of ALU Operations.**

OPR Select	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A+B	ADD
00101	Subtract A-B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift right A	SHRA
11000	Shift left A	SHLA

To perform the following operation:

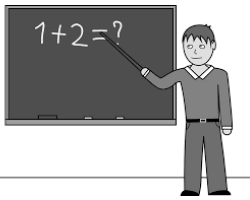
$$R1 \leftarrow R2 + R3$$

The value of binary control variables on control word must be:



This will select the contents of register R2 and R3 through the multiplexer buses A and B using SELA and SELB lines and transfer it to ALU for processing. The OPR field will select addition operation to add the content of R2 and R3. After processing the output is placed in the bus connected to the input of all the registers. Finally, SELD will enable the load control variable of register R1 to transfer the content of bus to destination register R1.





## Check Your Progress

1. A multiplexer or mux is a combinational circuits that selects several analog or digital input signals and forwards the selected input into a single output line.
2. A \_\_\_\_\_ is one of a small set of data holding places that are part of the computer processor.
3. \_\_\_\_\_ is a combinational logic circuit that converts binary information from the n coded inputs to a maximum of  $2^n$  unique outputs.
4. RTL stands for \_\_\_\_\_.

---

### 9.3 SUMMARY

---

1. During execution of a program, data values, results, return addresses, and partial results are stored in the memory location.
2. Accessing the memory is the most time consuming operation in a computer.
3. The CPU can access a register more quickly than a memory location and the register-to-register operation executes faster than compared to the memory-to-memory or register-to-memory operation.
4. Due to the faster execution in register, the intermediate data is stored in the register so that it is more convenient and more efficient for CPU to access these data and perform operation.
5. When a large number of registers are included in the CPU, it is most efficient to connect them through a common bus system.
6. The registers communicate with each other not only for direct data transfers, but also while performing various micro-operations.
7. The control unit that operates the CPU bus system directs the information flow through the registers and ALU by selecting the various components in the systems.
8. The data from the two source registers propagate through the gates in the multiplexer and the ALU, to the output bus, and into the into of the destination registers, all during the clock cycle intervals.

---

## 9.4 ANSWERS TO CHECK YOUR PROGRESS

---

1. Multiplexer
2. Processor Register/CPU register
3. Decoder
4. Register Transfer language

---

## 9.5 TERMINAL QUESTIONS

---

1. Why register set is required in the CPU?
2. What is the role of control unit in CPU?
3. Explain with the help of a diagram how register transfer takes place within the CPU.
4. Explain the working of the following operation:

**$R1 \leftarrow R2 - R3$**

Show the control word for the above operation.

# UNIT-10

---

## STACK ORGANIZATION

---

### Structure

- 10.0 Learning Objectives
- 10.1 Introduction
- 10.2 Memory Stack
- 10.3 Evaluation of Arithmetic Expressions
- 10.4 Summary
- 10.5 Answers to Check Your Progress
- 10.6 Terminal Questions

---

### 10.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Understand stack organization.
- Explain LIFO organization.
- Understand Push and Pull operation on stack.
- Evaluate arithmetic operations using stack.

---

### 10.1 INTRODUCTION

---

Now we will discuss a very useful data structure, stack organization. It follows *Last In, First Out(LIFO)* structure. The operation of a stack can be compared to a stack of trays. The last tray placed on top of the stack is the first to be taken off.

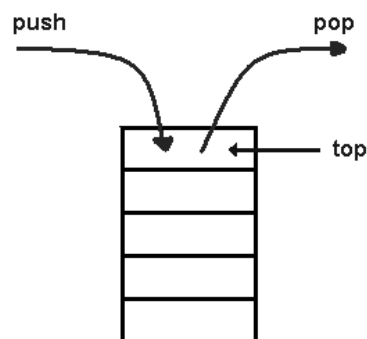
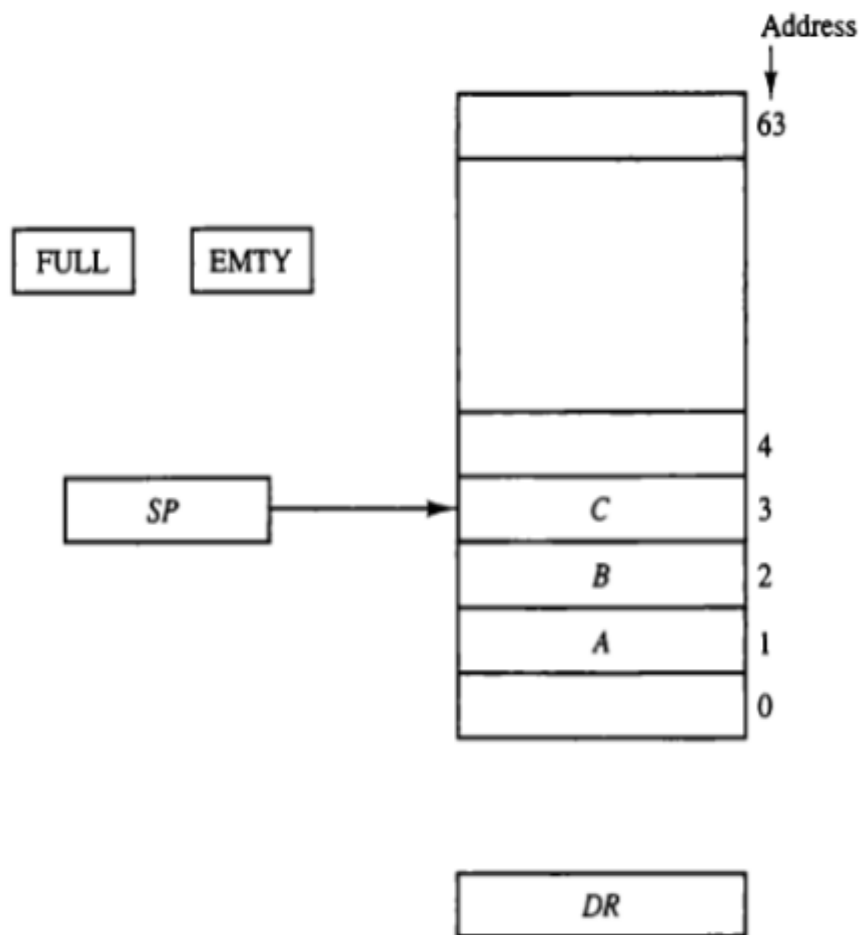


Figure 116 : Stack Organization 146

Stack structure could be assumed to be similar to a room with a single entry/exit point. One can either enter or exit at a time. We cannot perform both the operations simultaneously. Whenever someone enters the room, it is known as *push* operation and whenever somebody leaves the room, it is known as *pop* operation. A register, known as *stack pointer(SP)* keep track of the top-most element of the stack always points at the top item in the stack. Whenever, an element is added to a stack the stack pointer is incremented. Similarly, if any item is removed for the stack pointer is decremented.

A stack can be placed in a portion of a large memory or it can be organized as a collection of a finite number of memory words or registers. Following figure shows the organization of a 64-word register stack.



**Figure 117: Block Diagram of 64-word stack**

The operation of the stack can be explained as follows: Initially, when the stack is empty, Stack Pointer(SP) is cleared to 0, EMTY is set to 1, and FULL is cleared to 0, so that SP points to the word at address 0 and the stack is marked empty and not full. If the stack is not full (if FULL =0), a new item is inserted with a push operation. The push operation is implemented with the following sequence of micro-operations: 147

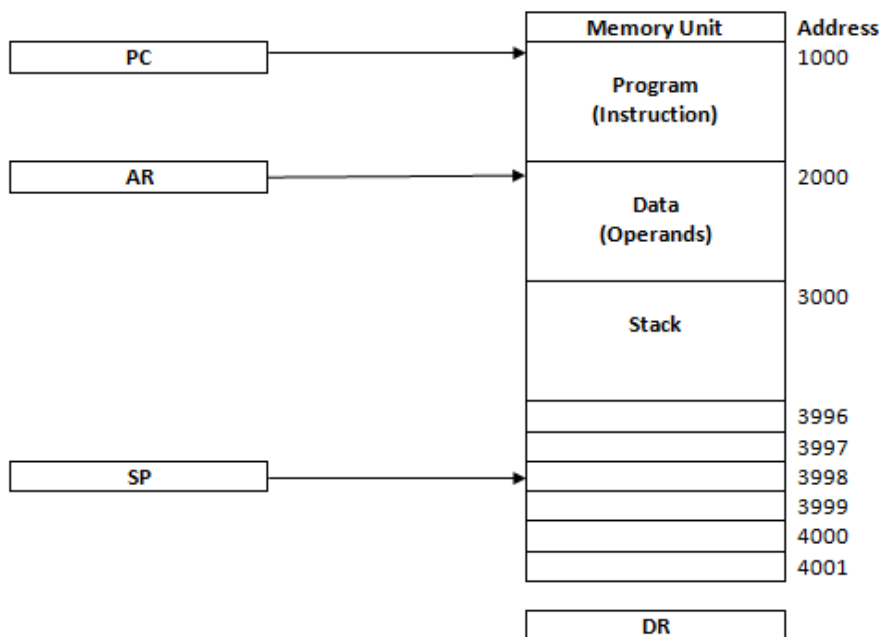
$SP \leftarrow SP + 1$  Increment stack pointer  
 $M[SP] \leftarrow DR$  Write item on top of the stack  
 If  $(SP = 0)$  then  $(FULL \leftarrow 1)$  Check if stack is full  
 $EMPTY \leftarrow 0$  Mark the stack not empty

Whenever a new item is deleted from the stack if the stack is not empty (if pop  $EMPTY = 0$ ). The pop operation consists of the following sequence of micro- operations:

$DR \leftarrow M[SP]$  Read item from the top of stack  
 $SP \leftarrow SP - 1$  Decrement stack pointer  
 If  $(SP = 0)$  then  $(EMPTY \leftarrow 1)$  Check if stack is empty  
 $FULL \leftarrow 0$  Mark the stack not full

## 10.2 MEMORY STACK

Following Figure 118 shows a portion of computer memory partitioned into three segments: program, data, and stack. The program counter PC points at the address of the next instruction in the program. The address register AR points at an array of data. The address register SP points at an array of data.



**Figure 118: Computer Memory with Program, Data and Stack Segment**

We can see in the above Figure 118, the address range of the stack is from 4001(initial address) to 3000(last address). The stack grows with decreasing address. We assume that the items in the stack communicate with a data register DR. A new item is inserted with the push operation as follows:

$$SP \leftarrow SP - 1$$

$$M[SP] \leftarrow DR$$

The stack pointer is decremented so that it points at the address of the next word. A memory write operation inserts the word from DR into the top of the stack. A new item is deleted with a pop operation as follows:

$$DR \leftarrow M[SP]$$

$$SP \leftarrow SP + 1$$

---

### 10.3 EVALUATION OF ARITHMETIC EXPRESSIONS

---

The stack organization is very useful and is used by the CPU for evaluating arithmetic expression. Reverse Polish notation, combined with a stack arrangement of registers, is the most efficient way known for evaluating arithmetic expressions. This procedure is employed in some electronic calculators and also in some computers.

The rule for converting the infix notation( the notation used in day to day life where the operator is placed between the operands, for eg. 3+4), is as follows:

Suppose we are given an expression in infix notation:

$$A * B$$

The equivalent postfix(reverse polish notation) for the expression is as follows:

$$AB*$$

The operator is placed after the operands.

The stack is particularly useful for handling long, complex problems involving chain

calculations. It is based on the fact that any arithmetic expression can be expressed in

parentheses-free Polish notation.

1. Scan the expression from left to right.
2. Place any operand that is encountered, in the stack and update SP.
3. Whenever an operator is encountered, perform the operation with the two topmost operands of the stack by POPping the items for the stack.
4. The result of the operation is PUSHed onto the stack and the SP is updated.

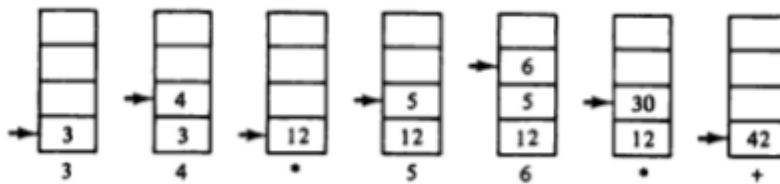
This could be explained more clearly using an example. Consider the arithmetic expression:

$$(3*4) + (5*6)$$

In reverse Polish notation, it is expressed as

$$34*56*+$$

Now using the following routing, we can evaluate the arithmetic expression using stack as follows:

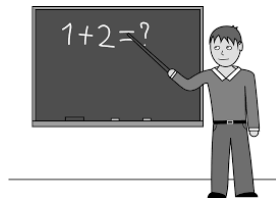


**Figure 119: Stack Operation to Evaluate  $(3*4) + (5*6)$**

1. Scan the expression from left to right.
2. Place any operand that is encountered, in the stack and update SP (this will place 3 and 4 in the stack and the SP will point to 4, i.e. the top most element of the stack).
3. Whenever an operator is encountered, perform the operation with the two topmost operands of the stack by POPping the items from the stack( an operator \* is encountered. POP 4 and 3 from the stack and \* operation is performed on 3 and 4).
4. The result of the operation is PUSHed onto the stack and the SP is updated. (After the operation is performed, the result i.e. 12 is PUSHed onto the stack and SP is updated).

1. Scan the expression from left to right.
2. Place any operand that is encountered, in the stack and update SP (this will place 3 and 4 in the stack and the SP will point to 4, i.e. the top most element of the stack).
3. Whenever an operator is encountered, perform the operation with the two topmost operands of the stack by POPping the items from the stack( an operator \* is encountered. POP 4 and 3 from the stack and \* operation is performed on 3 and 4).
4. The result of the operation is PUSHed onto the stack and the SP is updated. (After the operation is performed, the result i.e. 12 is PUSHed onto the stack and SP is updated).

This continues till the last element. And at the end the stack contains the result of the arithmetic operation.



## Check Your Progress

1. SP stands for \_\_\_\_\_.
2. The one bit register \_\_\_\_\_ is set to 1 when the stack is full.
3. One-bit register \_\_\_\_\_ is set to 1 when the stack is empty.

---

## 10.4 SUMMARY

---

1. A useful feature that is included in the CPU of most computers is a stack or last-in first out (LIFO) list.
2. A stack is a storage device that stores information in such a manner that the item stored last is the first item retrieved.
3. The operation a stack can be compared to a stack of trays.
4. The stack in Digital Computer is essentially a memory unit with an address register that can count only (after an initial value is loaded into it.)
5. The register that holds the address for the stack is called a Stack Pointer (SP) because its values always points at the top item in the stack.
6. A stack can be placed in a portion of a large memory as it can be organized as a collection of a finite number of memory words as register.
7. DR is the data register that holds the binary data to be written into on read out of the stack.

---

## 10.5 ANSWERS TO CHECK YOUR PROGRESS

---

1. Stack pointer
2. FULL
3. EMTY



---

## 10.6 TERMINAL QUESTIONS

---

1. What is stack?
2. What is reverse polish notation?
3. How reverse polish notations are useful in evaluating arithmetic expressions using stack organization.
4. Convert the following numerical arithmetic expression into reverse polish notation and show the stack operation for evaluating the numerical result.

$(3+4)[(10(2+6)+8)]$



# UNIT-11

---

## INSTRUCTION FORMATS

---

### Structure

- 11.0 Learning Objectives
- 11.1 INTRODUCTION
- 11.2 Three Address Instructions
- 11.3 Two Address Instructions
- 11.4 One Address Instructions
- 11.5 Zero Address Instructions
- 11.6 RISC Instructions
- 11.7 Summary
- 11.8 Answers to Check Your progress
- 11.9 Terminal Questions

---

### 11.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Understand the Three register instructions.
- Understand the Two register instructions.
- Understand the One register instructions.
- Understand the Zero register instructions.
- Understand the stack instructions.

---

### 11.1 INTRODUCTION

---

The instruction code format of a computer varies from platform to platform. A single computer can also have variety of instruction formats. The instructions are given to the computer via program. Once the program is executes, the instructions is decoded and the CPU generates all the necessary control signals to initiate the operation mentioned in the instruction.

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. 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 Operands residing in processor registers are specified with a **register address**. Computers may have instructions of several different lengths containing varying number of addresses. The number of address fields in the instruction format of a computer depends on the internal organization of its registers. Most computers fall into one of three types of CPU organizations:

1. Single accumulator organization.
2. General register organization.
3. Stack organization.

---

## 11.2 THREE ADDRESS INSTRUCTIONS

---

This address instruction has the following parts

- Operation code
- Address of two operands called Address 1 and Address 2
- Address of the memory location where the result of the operation is to be stored i.e. Address of the destination.

Operation Code	Destination	Source 1	Source 2
----------------	-------------	----------	----------

← Address 1 → ← Address 2 → ← Address 3 →

**Figure 120 : Three Address Instruction Format**

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 ← M[A] + M[B]*

*ADD R2, C, D R2 ← M[C] + M[D]*

*MUL X, R1, R2 M[X] ← R1 \* R2* 153

We will assume that the operands are in memory addresses A, B, C, and D, and the result must be stored in memory at address X. 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. The advantage of the three-address format is that it results in short programs when evaluating arithmetic expressions. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

An example of a commercial computer that uses three-address instructions is the Cyber 170. The instruction formats in the Cyber computer are restricted to either three register address fields or two register address fields and one memory address field.

---

## 11.3 TWO ADDRESS INSTRUCTIONS

---

Two-address instructions are the most common in commercial computers.

Operation	Destination	Source
-----------	-------------	--------

← Address 1 → ← Address 2 →

**Figure 121: Two Address Instruction Format**

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*  $M[X] \leftarrow R1$

---

## 11.4 ONE ADDRESS INSTRUCTIONS

---

One-address instructions use an implied accumulator (AC) register for all data manipulation.

Op-Code	Address 1
---------	-----------

**Figure 1: One Address Instruction Format**

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

$$\mathbf{X = (A + B) * (C + D)}$$

is as follows:

LOAD	A	AC ← M[A]
ADD	B	AC ← AC + M[B]
STORE	T	M[T] ← AC
LOAD	C	AC ← M[C]
ADD	D	AC ← AC + M[D]
MUL	T	AC ← AC + M[T]
STORE	X	M[X] ← AC

---

## 11.5 ZERO ADDRESS INSTRUCTIONS

---

A stack-organized computer does not use an address held for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address held to specify the operand that communicates with the stack. The following program shows how  $\mathbf{X = (A + B) * (C + D)}$  will be written for a stack-organized computer. (TOS stands for top of stack.)

<b>PUSH</b>	<b>A</b>	<b>TOS ← A</b>
<b>PUSH</b>	<b>B</b>	<b>TOS ← B</b>
<b>ADD</b>		<b>TOS ← (A+B)</b>
<b>PUSH</b>	<b>C</b>	<b>TOS ← C</b>
<b>PUSH</b>	<b>D</b>	<b>TOS ← D</b>
<b>ADD</b>		<b>TOS ← (C+D)</b>
<b>MUL</b>		<b>TOS ← (C+D)*(A+B)</b>
<b>POP</b>	<b>X</b>	<b>M[X] ← TOS</b>

---

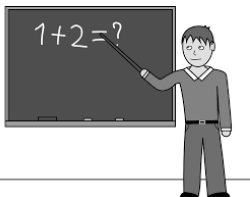
## 11.6 RISC INSTRUCTIONS

---

The instruction set of a typical RISC processor is restricted to the use of load and store instructions when communicating between memory and CPU. All other instructions are executed within the registers of the CPU without referring to memory. The following is a program to evaluate

$$X = (A + B) * (C + D)$$

<b>LOAD</b>	<b>R1,A</b>	<b>R1←M[A]</b>
<b>LOAD</b>	<b>R2,B</b>	<b>R2←M[B]</b>
<b>LOAD</b>	<b>R3,C</b>	<b>R3←M[C]</b>
<b>LOAD</b>	<b>R4,D</b>	<b>R4←M[D]</b>
<b>ADD</b>	<b>R1,R1,R2</b>	<b>R1←R1+R2</b>
<b>ADD</b>	<b>R3,R3,R4</b>	<b>R3←R3+R4</b>
<b>MUL</b>	<b>R1,R1,R3</b>	<b>R1←R1*R3</b>
<b>STORE</b>	<b>X,R1</b>	<b>M[X] ←R1</b>



### Check Your Progress

1. The Operands residing in processor registers are specified with a \_\_\_\_\_ address.
2. TOS stands for \_\_\_\_\_ .
3. In case of, Zero-address instruction method the operands are stored in \_\_\_\_\_..

---

## 11.7 SUMMARY

---

1. Computers may have instructions of several different lengths containing varying number of addresses.
2. The number of address field in the instruction format of a computer depends on the internal organization of its registers.

3. Most computers fall into one of three types of CPU organization: Single Accumulator Organization, General Register Organization and Stack Organization.
4. Computer with three addresses instruction format can use each address field to specify either processor register or memory operand.
5. The advantage of the three address formats is that it results in short program when evaluating arithmetic expression.
6. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

---

## **11.8 ANSWERS TO CHECK YOUR PROGRESS**

---

1. Register
2. Top Of Stack
3. Push down stack

---

## **11.9 TERMINAL QUESTIONS**

---

1. What is single accumulator organization.
2. What is Stack Organization
3. What is general register organization.
4. Give an example of three address instruction format and evaluate an expression

$$X = (A + B) * (C + D)$$

using three address instruction.



# UNIT-12

---

## ADDRESSING MODES

---

### Structure

#### 12.0 Learning Objectives

#### 12.1 Introduction

#### 12.2 Addressing Modes

##### 12.2.1 Implied Mode

##### 12.2.2 Immediate Mode

##### 12.2.3 Register Mode

##### 12.2.4 Register Indirect Mode

##### 12.2.5 Auto-increment or Auto-decrement Mode

##### 12.2.6 Direct Address Mode

##### 12.2.7 Indirect Address Mode

##### 12.2.8 Relative Address Mode

##### 12.2.9 Indexed Addressing Mode

##### 12.2.10 Base Register Addressing Mode

#### 12.3 Summary

#### 12.4 Answers to Check Your Progress

#### 12.5 Terminal Questions

---

### 12.0 Learning Objectives

---

After reading this unit, you will be able to:

- Explain various addressing modes.
- Define immediate addressing mode
- Differentiate between direct and indirect addressing mode.
- Explain Register mode.

---

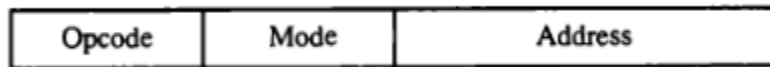
### 12.1 Introduction

---

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. Computers use addressing mode techniques for 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.



**Figure 23: Instruction Code with Mode Field**

The control unit of a computer is designed to go through an instruction cycle that is divided into three major phases:

1. Fetch the instruction from memory.
2. Decode the instruction.
3. Execute the instruction.

There is one register in the computer called the program counter or PC that keeps track of the instructions in the program stored in memory. PC holds the address of the instruction to be executed next and is incremented each time an instruction is fetched from memory.

The **mode field** is used to locate the operands needed for the operation.

---

## 12.2 ADDRESSING MODES

---

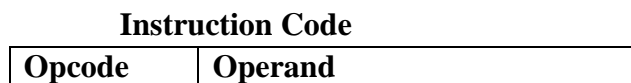
There are various addressing modes which are discussed in length in the following section.

### 12.2.1 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. In fact, all register reference instructions that use an accumulator are implied-mode instructions.

### 12.2.2 Immediate Mode

In this mode the operand is specified in the instruction itself i.e. the address part of the instruction code contains the operand itself.



**Figure 34: Immediate mode**

### 12.2.3 Register Mode

In this mode the operands are in registers that reside within the CPU. The particular register is selected from a register field in the instruction. A  $k$ -bit field can specify any one of  $2^k$  registers.

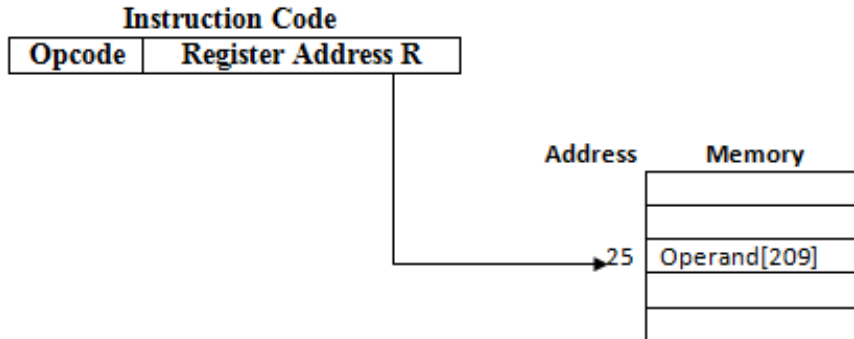


Figure 125: Register Mode

### 12.2.4 Register Indirect Mode

In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory.

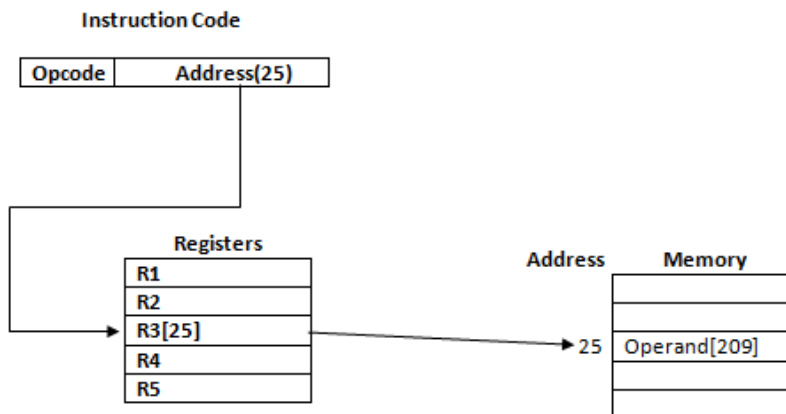


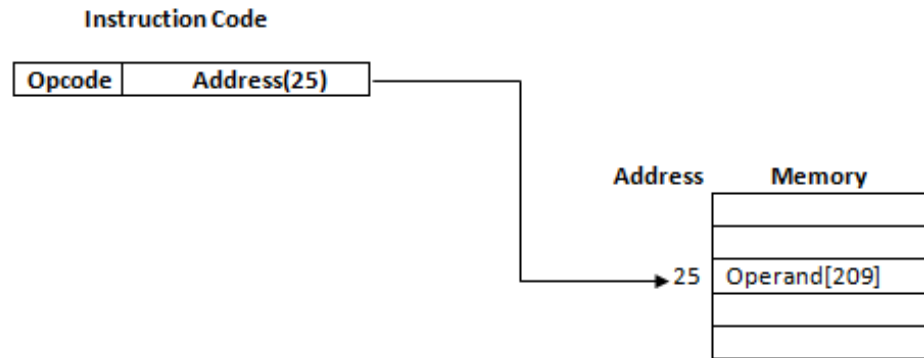
Figure 126: Register Indirect mode

### 12.2.5 Auto-increment or Auto-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. The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode.

### 12.2.6 Direct Address Mode

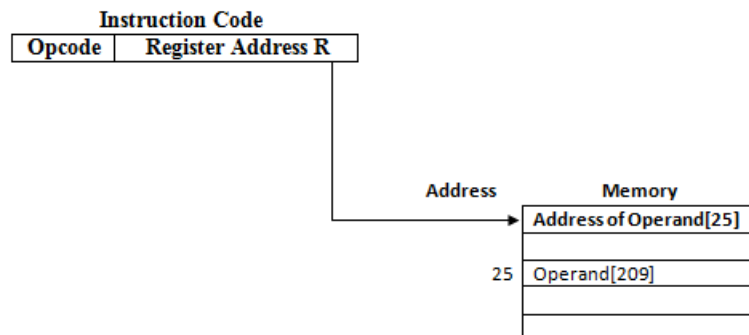
In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction.



**Figure 127: Direct addressing mode**

### 12.2.7 Indirect Address Mode

In this mode the address field of the instruction gives the address where the effective address is stored in memory. A few addressing modes require that the address field of the instruction be added to the content of a specific register in the CPU. The effective address in these modes is obtained from the following computation: effective address = address part of instruction + content of CPU register



**Figure 4: Indirect Address Mode**

### 12.2.8 Relative Address Mode

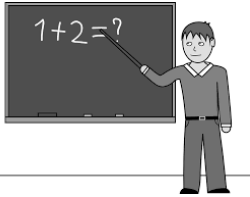
In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.

### 12.2.9 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.

## 12.2.10 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. This is similar to the indexed addressing mode except that the register is now called a base register instead of an index register.



### Check Your Progress

1. . An \_\_\_\_\_ mode instruction has an operand field rather than the address field.
2. In \_\_\_\_\_ mode, the instruction has the address of the Register where the operand is stored.
3. In \_\_\_\_\_ mode, the register contains the address of operand rather than the operand itself.
4. \_\_\_\_\_ mode is a version of Displacement addressing mode.
5. \_\_\_\_\_ mode is most suitable to change the normal sequence of execution of instructions .

---

## 12.3 SUMMARY

---

1. Addressing modes are an aspect of the instruction set architecture in most central processing unit (CPU) designs.
2. The various addressing modes that are defined in a given instruction set architecture define how machine language instructions in that architecture identify the operand(s) of each instruction.
3. An addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within a machine instruction or elsewhere.
4. In immediate mode, the operand is specified in the instruction itself.
5. In register mode, the operand is stored in the register and this register is present in CPU.
6. In register indirect mode, the instruction specifies the register whose contents give us the address of operand which is in memory.
7. In autoincrement/autodecrement mode, the register is incremented or decremented after or before its value is used.

8. In direct addressing mode, effective address of operand is present in instruction itself.
9. In indirect addressing mode,, the address field of instruction gives the address where the effective address is stored in memory.
10. In relative addressing mode, the contents of PC(Program Counter) is added to address part of instruction to obtain the effective address.

---

## **12.4 ANSWERS TO CHECK YOUR PROGRESS**

---

1. Immediate
2. Register
3. Register indirect
4. Relative addressing
5. Relative addressing

---

## **12.5 TERMINAL QUESTIONS**

---

1. What are the three phases of instruction cycle?
2. What is a difference between register mode and auto-increment/auto-decrement mode?
3. What is an effective address? How it is computer?
4. Compare index address mode with base register addressing mode.
5. Explain the various address modes using an example.



॥ सरस्वती नः सुभगा मयस्कृत ॥

Uttar Pradesh Rajarshi Tandon  
Open University

# Bachelor of Computer Application

## BCA-1.10 Computer Organization

### BLOCK

# 3

## Memory & I/O

UNIT 13	185-204
Memory	
UNIT 14	205-218
Peripheral Devices	
UNIT 15	219-242
Introduction To 8085 Microprocessor	

---

## Course Design Committee

---

**Dr. Ashutosh Gupta** **Chairman**  
Director (In-charge)  
School of Computer and Information Science,  
UPRTOU Prayagraj

**Prof. R. S. Yadav** **Member**  
Department of Computer Science and Engineering  
MNNIT Prayagraj

**Ms Marisha** **Member**  
Assistant Professor (Computer Science),  
School of Science, UPRTOU Prayagraj

**Mr. Manoj Kumar Balwant** **Member**  
Assistant Professor (Computer Science)  
School of Science, UPRTOU Prayagraj

---

## Course Preparation Committee

---

**Dr. Jitendra Pande** **Author**  
Associate Professor  
School of Computer Sciences & Information Technology  
Haldwani, Uttarakhand 263139

**Dr. Abhay Sexena** **Editor**  
Professor and Head, Department of Computer Science  
Dev Sanskriti Vishwavidyalya, Hardwar, Uttrakhand

**Dr. Ashutosh Gupta**  
Director (In-Charge)  
School of Computer & Information Sciences,  
UPRTOU Prayagraj

**Mr. Manoj Kumar Balwant** **Coordinator**  
Assistant Professor (computer science),  
School of Sciences, UPRTOU Prayagraj

---

© UPRTOU, Prayagraj. 2019

ISBN :

*All Rights are reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from the*

*Uttar Pradesh Rajarshi Tondon Open University, Prayagraj.*

Printed and Published by Dr. Arun Kumar Gupta Registrar, Uttar Pradesh Rajarshi Tandon Open University, 2019.

**Printed By:** Chandrakala Universal Pvt. Ltd. 42/7 Jawahar Lal Neharu Road, Prayagraj.



# UNIT-13

---

## MEMORY

---

### Structure

- 13.0 Learning Objectives
- 13.1 Introduction
- 13.2 Main Memory
  - 13.2.1 RAM (Random access memory)
  - 13.2.2 ROM (Read Only Memory)
  - 13.2.3 Difference between RAM and ROM
  - 13.2.4 Memory Address Table
- 13.3 Cache Memory
  - 13.3.1 Associative Mapping
  - 13.3.2 Direct Mapping
  - 13.3.3 Set-Associative Mapping
  - 13.3.4 Writing into Cache
  - 13.3.5 Page Replacement Algorithm
- 13.4 Virtual memory
  - 13.4.1 Address Space and Memory Space
  - 13.4.2 Address mapping Using Pages
  - 13.4.3 Associative Memory Page Table
- 13.5 Summary
- 13.6 Answers to Check Your Progress
- 13.7 Terminal Questions

---

### 13.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Explain memory hierarchy.
- Define Main Memory.
- Differentiate between RAM and ROM.

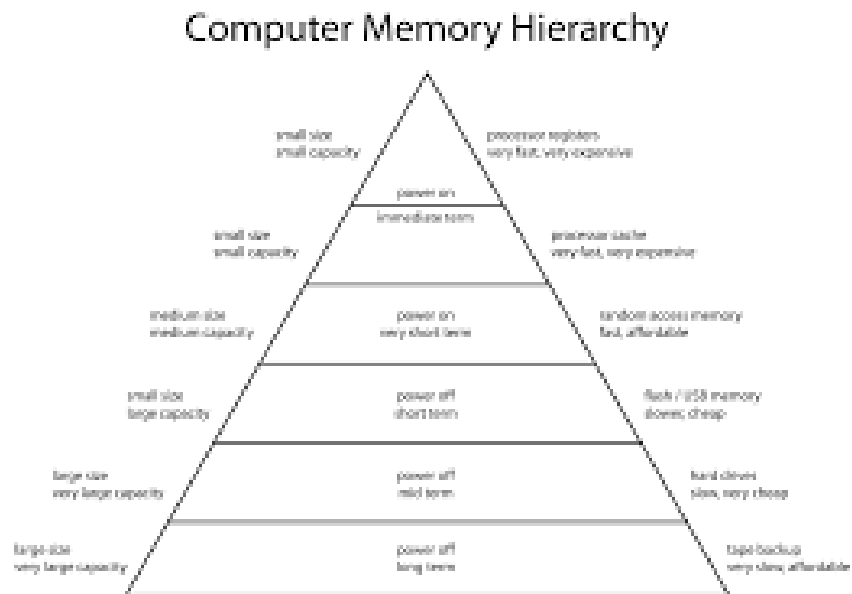
- Define Cache memory.
- Different mapping techniques in cache memory.
- Define Virtual memory.
- Different mapping techniques in virtual memory.

---

## 13.1 INTRODUCTION

---

A computer system have various type of memories like CPU registers, cache memory, main memory, secondary memory, etc. All the memories have same function i.e. to store the data, but they differ in their speed and cost. These memories are arranges in hierarchies. The access speed of the expensive memory is fast. By using a hierarchy of memories, each with different access speeds and storage capacities, a computer system can exhibit performance above what would be possible without a combination of the various types. The base types that normally constitute the hierarchical memory system include registers, cache, main memory, and secondary memory.



**Figure 130: Memory Hierarchy**

In the top of this hierarchy, there are CPU registers. Their access speed is comparable to the speed of CPU. They are used by CPU for storing temporary data during calculations. At second level, a very high-speed memory, called a *cache*, is attached to the computer. It is used to store data temporarily that is very frequently used from memory locations may. It is connected to *main memory*, which is typically a medium-speed memory. This memory is complemented by a very large secondary memory, composed of a hard disk and various removable media. By using such a hierarchical scheme, one can 165

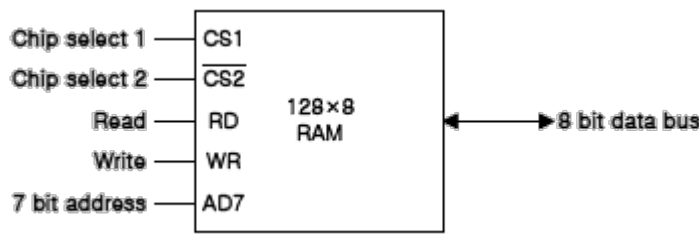
improve the effective access speed of the memory, using only a small number of fast (and expensive) chips. This allows designers to create a computer with acceptable performance at a reasonable cost.

## 13.2 MAIN MEMORY

The main memory constitutes of RAM and ROM, is the central storage unit in a computer system. It is a temporary storage medium and the data is lost in case the power is lost( except in the case of ROM). RAM and ROM are available in a variety in size. If the memory needed for the computer is large than the capacity of one chip it is necessary to combine a number of chips to obtain the required memory size. It is called as main memory because it can be accessed directly by the CPU. Now let us discuss the different variants of main memory.

### 13.2.1 RAM (Random access memory)

It is read write memory. It is just like a page of a notebook, where you can write something to or read something from. All the programs are brought into RAM just before execution. The block diagram of a RAM can be represented as:



**Figure 131: Block diagram of a RAM**

The functional table of the RAM is shown below:

**\ Table 20 : Functional Table of RAM**

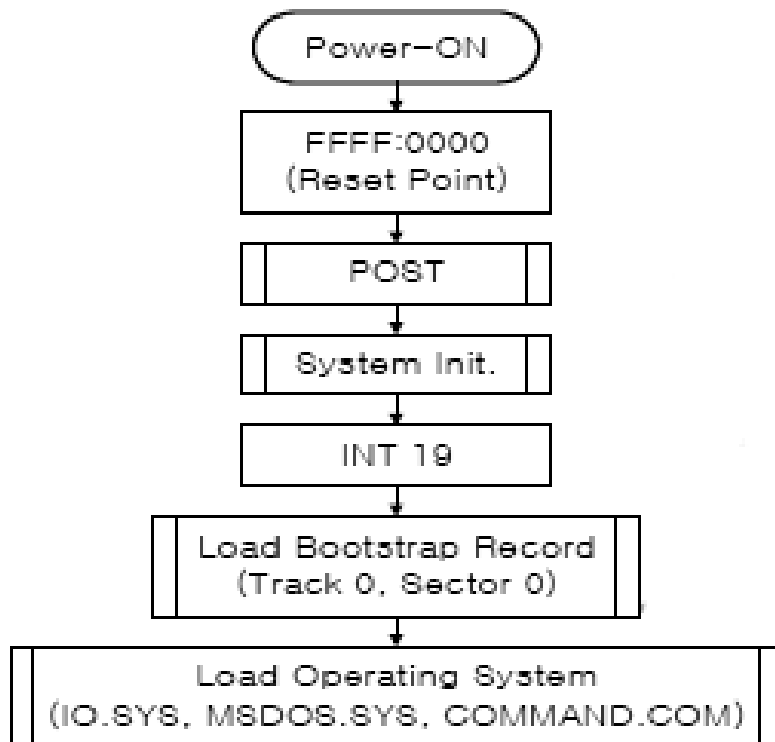
CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	x	x	Inhibit	High-impedance
0	1	x	x	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	x	Read	Output data from RAM
1	1	x	x	Inhibit	High-impedance

There are many variants of RAM. Some of them are:

- *DRAM (Dynamic RAM):* This is the most common type of computer memory. It is called dynamic because it must be refreshed, or re-energized, hundreds of times each second in order to retain the data in its words. Each bit in a word in DRAM is designed around a tiny capacitor that can store an electrical charge. A charged capacitor indicates a 1-bit. However, the capacitor loses its charge rapidly, which is why DRAM must be refreshed.
- *SRAM (Static RAM):* This type of memory is about five times faster, twice as expensive, and twice as big as a DRAM. SRAM does not need to be refreshed like a DRAM. Each bit in SRAM is a flip-flop; a circuit that has two stable states. Once placed in a certain state, it stays in that state. Flip-flops are faster to read and write than the capacitors of the DRAM, but they consume more power.

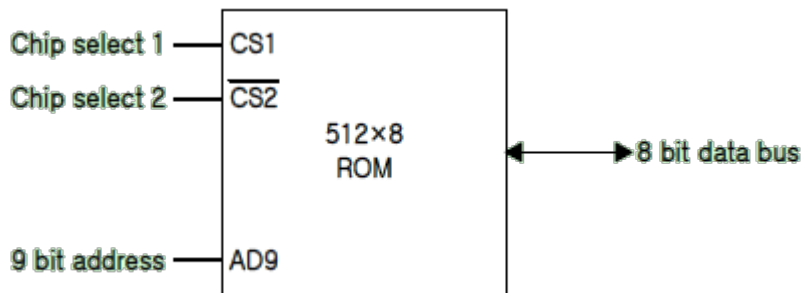
### 13.2.2 ROM (Read Only Memory)

It is Nonvolatile memory i.e. the content remains in the memory even if the power is switched off. It stores mainly the monitor programs and BIOS (Basic Input Output System) Programs.



**Figure 132 : Bootstrap loader flow chart**

The information stored in ROM can only be read but cannot be modified. The contents of ROM can be programmed under special conditions. It is manufacturer programmed memory. The block diagram of a ROM can be represented as:



**Figure 133: Block Diagram of a ROM**

There are many variants of ROM available in the market. Some of them are:

- *PROM - Programmable Read Only Memory*: this is write-once, read-many type of memory, which could be programmed after fabrication. The data is written to the memory electrically using some special circuitry.
- *EPROM – Erasable Programmable Read Only Memory*: this type of ROM has multiple read and writes facility. The old data present in the ROM can be erased by exposing the ROM chip to UV radiation, after which new data can be written into the ROM.
- *EEPROM – Electrically Erasable Programmable Read Only Memory*: This is the most expensive variant of the ROM and the per-bit storage cost of this variant is more as it is less dense than EPROM. It performs Byte-level writing, in which any part(s) of the memory can be written at any time.

### 13.2.3 Difference between RAM and ROM

RAM	ROM
Read- Write Memory	Read Only Memory
Volatile memory i.e. the contents of the RAM are lost when power is turned off	Non-volatile memory i.e. the contents of the ROM are not lost when power is turned off
Temporary storage medium	Permanent storage medium
The data can be read and written	The data can only be read, but the data cannot be written
The programs are brought into RAM just before execution	BIOS and monitor programs are stored

### 13.2.4 Memory Address Table

While designing a computer system, the designer of the system must foresee in advance, the total memory required by the system and the type of memory that 168

will be required for that application. For example, for permanent programs, ROM is an ideal choice. In practical, both the types of memory is used in designing a system. Based on the type of application, the designer of the system must calculate the amount of memory required for the particular application and assign it to either RAM or ROM. The interconnection between memory and processor is then established from knowledge of the size of memory needed and the type of RAM and ROM chips available. The addressing of memory can be established by means of a table that specifies the memory address assigned to each chip. The table, called a memory address map, is a pictorial representation of assigned address space for each chip in the system.

Now let us demonstrate the above situation using an example. Suppose a computer uses RAM chips of 128x8 and ROM chips of 512 x 8. The memory address map with the chips needed for constructing system memory with 512x8 RAM and 512x8 ROM can be represented as in Table 21. We need 4 RAM and 1 ROM memory address map is:

**Table 21 : Memory Address Table**

Component	Address in hexadecimal	Address Bus									
		10	9	8	7	6	5	4	3	2	1
RAM1	0000-007F	0	0	0	x	x	x	x	x	x	x
RAM2	0080-00FF	0	0	1	x	x	x	x	x	x	x
RAM3	0100-017F	0	1	0	x	x	x	x	x	x	x
RAM4	0180-01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200-03 FF	1	x	x	x	x	x	x	x	x	x

The above memory connection to the CPU can be designed as shown in Figure 134.

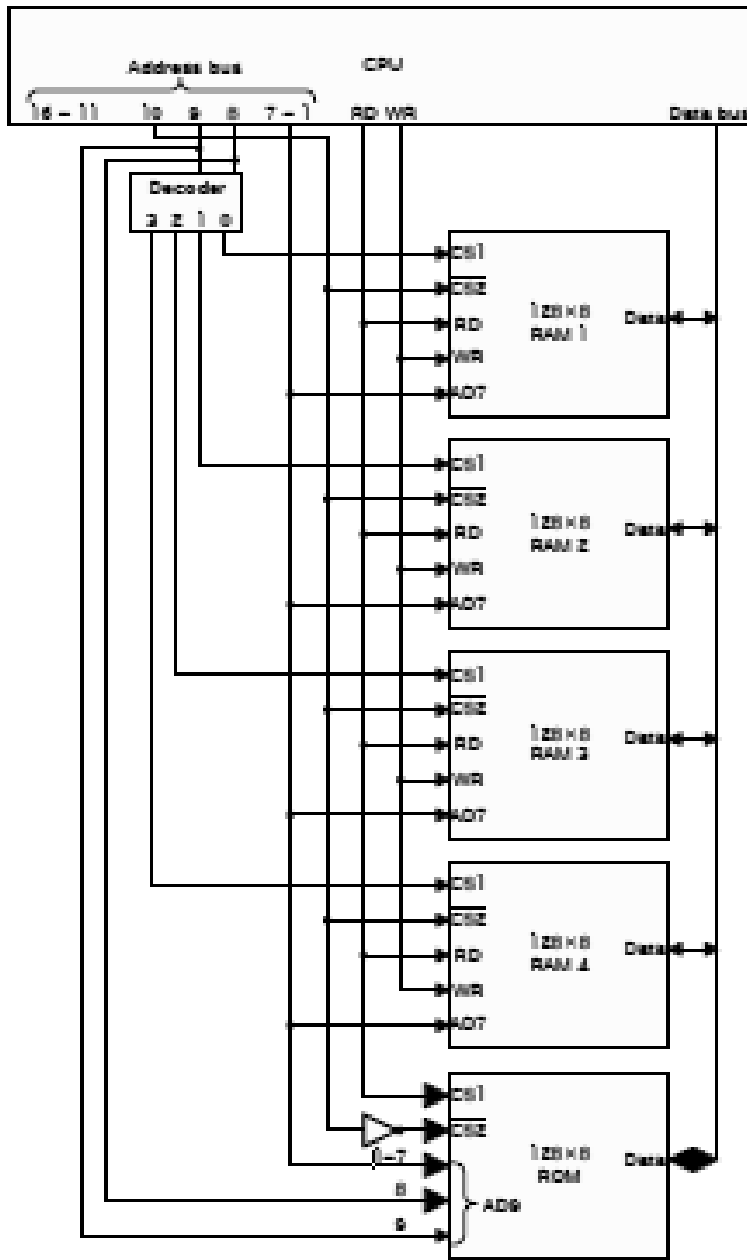


Figure 134 : Memory Connections to the CPU

### 13.3 CACHE MEMORY

The CPU processing speed is very fast as compared to the time taken by a memory-access. The speed of the CPU to perform an operation is limited by the memory-access time. If, by any mechanism, the memory access can be limited, the speed of the CPU to perform an operation can be increased considerably. Moreover, it has been found after analyzing a large number of programs that the references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of locality of

reference. If the content of these —local|| areas of the memory are brought closer to CPU i.e. placed in a memory whose access time is comparable to CPU speed, the CPU performance can increase to many folds. For this purpose, a fast small memory is referred to as a cache memory is employed. The cache memory access time is less than the access time of main memory by a factor of 5 to 10. The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.

As the cache is the amongst the topmost element of the memory hierarchy, its cost per bit storage is also very high. Due to economics, it is only a small fraction of the size of main memory. Therefore, the data from the main memory must be exchanged frequently to keep the frequently and most accessed data in the cache. The transformation of data from main memory to cache memory is referred to as a *mapping process*. Three types of mapping procedures are of practical interest when considering the organization of cache memory:

- Associative mapping.
- Direct mapping.
- Set-associative mapping.

### 13.3. 1 Associative Mapping

This is one of the optimal cache organization which employes associative memory for searching, and thus making the mapping process very fast. The associated mapping can be explained using Figure 135. The associative memory stores both the address and content (data) of the memory word. This permits any location in cache to store any word from main memory. The diagram shows three words presently stored in the cache. The address value of 15 bits is shown as a five-digit octal number and its corresponding 12-bit word is shown as a four-digit octal number. A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address.

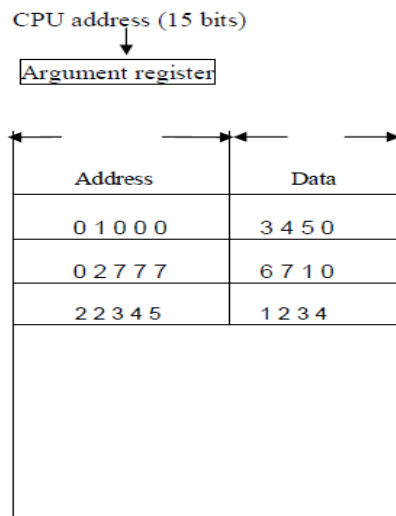


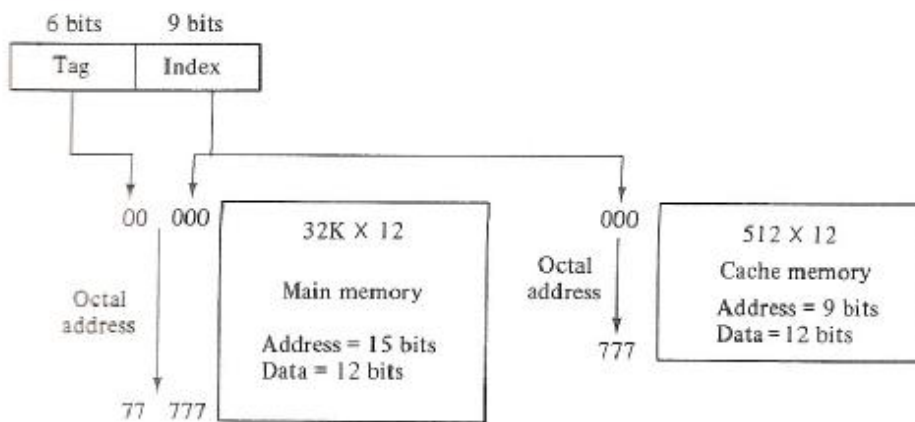
Figure 135: Associative Mapping



### 13.3.2 Direct Mapping

Direct mapping can be implemented using random access memory as shown below in Figure 136. Consider a main memory of size 32K X 12 and cache of size 512 X 12. 15 bits address is required to access each location uniquely. The 15-bit address part is divided into two parts. The 9 least significant bits (29=512) are required to address each location of cache of size 512 X 12 uniquely. Remaining 15-9= 6 bits are used for tag field.

The Figure 136 shows that main memory needs an address that includes both the tag and the index bits.



**Figure 136: Direct Mapping**

The direct mapping cache organization uses the n-bits address to access the main memory and the k-bits index to access the cache. The internal organization of the words in the cache memory is as shown in Figure 137. Each word in cache consists of the data word and its associated tag. When a new word is first brought into the cache, the tag bits are stored alongside the data bits. When the CPU generates a memory request, the index field is used for the address to access the cache. The tag field of the CPU address is compared with the tag in the word read from the cache. If the two tags match, there is a hit and the desired data word is in the cache. If there is no match, there is a miss and the required word is read from main memory. It is then stored in the cache together with the new tag, replacing the previous value.

The direct mapping technique has one disadvantage. The hit ratio can drop considerably if two or more words whose addresses have the same index but different tags are accessed repeatedly. However, this possibility is minimized by the fact that such words are relatively far apart in the address range (multiples of 512 locations in this example.).

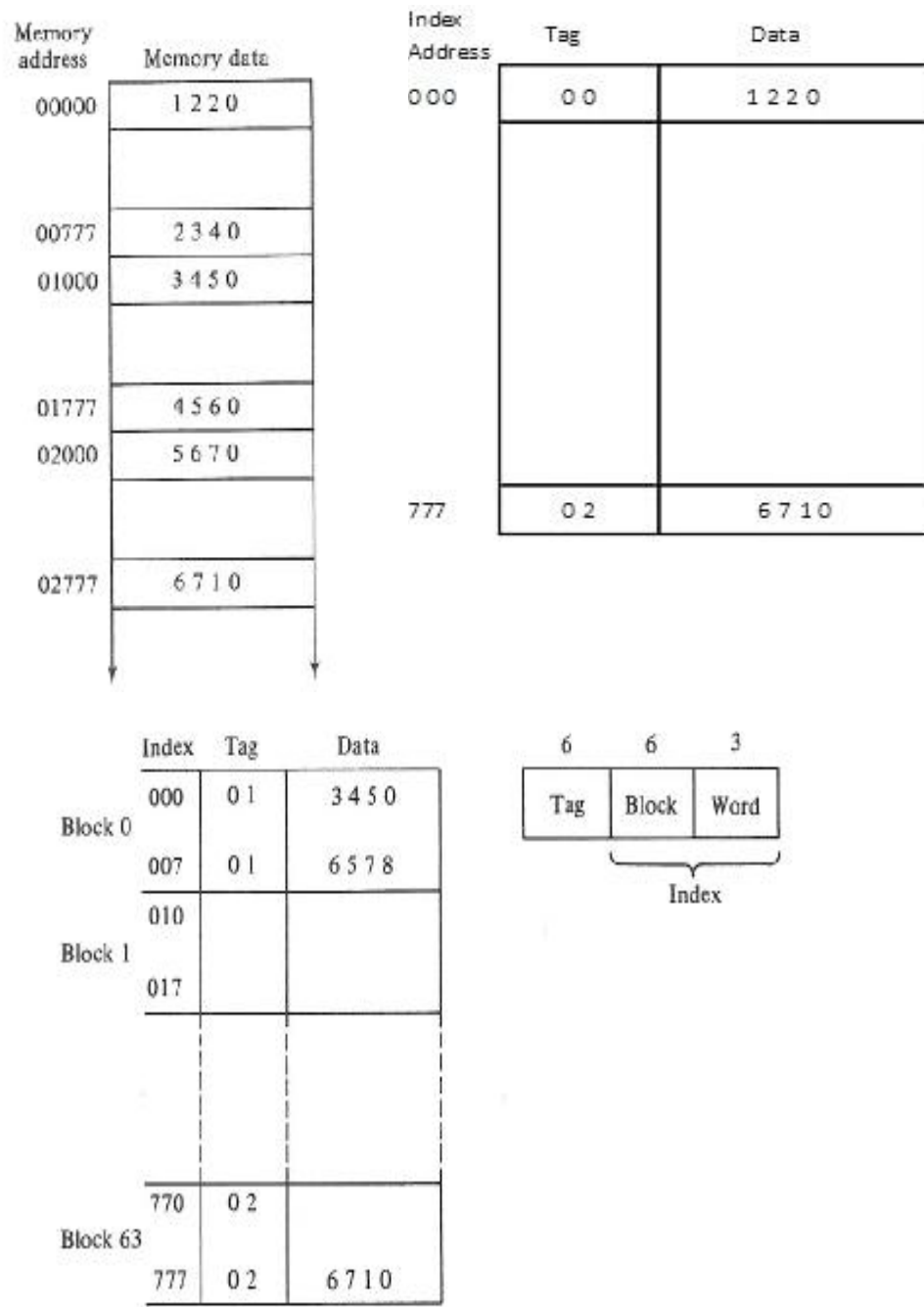


Figure 137: Direct Mapping Example

### 13.3.3 Set-Associative Mapping

To overcome the limitation of direct mapping i.e. each word of cache cannot store two or more words of memory under the same index address, a new mapping technique named as set-associative mapping is introduced. In this mapping technique, each word of cache can store two or more words of memory under the same index address. This can be explained using Figure 138 .

Index	Tag	Data	Tag	Data
000	0 1	3 4 5 0	0 2	5 6 7 0
777	0 2	6 7 1 0	0 0	2 3 4 0

**Figure 138: Set-Associative Mapping**

Each data word is stored together with its tag and the number of tag-data items in one word of cache is said to form a set. Each index address refers to two data words and their associated tags. For a main memory of size 32K X 12 and cache memory of size 512 X 12, each tag requires six bits and each data word has 12 bits, so the word length is  $2(6 + 12) = 36$  bits. An index address of nine bits can accommodate 512 words. Thus the size of cache memory is 512 X 36. It can accommodate 1024 words of main memory since each word of cache contains two data words. In general, a set-associative cache of set size  $k$  will accommodate  $k$  words of main memory in each word of cache.

### 13.3.4 Writing into Cache

The very idea of using cache memory is to limit memory-access so that the operation can be performed at a much faster speed. If there is a data read request, this problem of frequent data access could be easily resolved as the data is residing in the cache. But if there is a data write request, we cannot deny memory access. The main memory is updated with every memory write operation; with cache memory being updated in parallel if it contains the word at the specified address. This is called the **write-through** method. This method has the advantage that main memory always contains the same data as the cache. This characteristic is important in systems with direct memory access transfers. It ensures that the data residing in main memory are valid at all times so that an I/O device communicating through DMA would receive the most recent updated data.

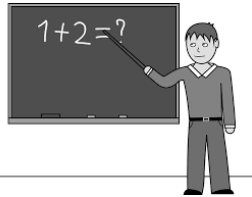
However, there are some techniques by which the frequency at which the memory is accessed can be controlled through a procedure called the **write-back** method. In this method, only the cache location is updated during a write operation. The location is then marked by a flag so that later when the word is removed from the cache it is copied into main memory. The reason for the write-back method is that during the time a word resides in the cache, it may be updated several times; however, as long as the word remains in the cache, it 174

does not matter whether the copy in main memory is out of date, since requests from the word are filled from the cache.

### 13.3.5 Page Replacement Algorithm

When a miss occurs in a Cache memory and the Cache is full, it is necessary to replace one word with a new word from main memory. The most common replacement algorithms are:

- Random Replacement: Select the item randomly.
- FIFO (First-In First-Out): Select the item has been in the Cache the longest.
- LRU (Least Recently Used): Select the item that has been least recent used by the CPU.



### Check Your Progress

1. The last on the hierarchy scale of memory devices is \_\_\_\_\_.
2. The effectiveness of the cache memory is based on the property of \_\_\_\_\_.
3. The correspondence between the main memory blocks and those in the cache is given by \_\_\_\_\_.
4. The algorithm to remove and place new contents into the cache is called \_\_\_\_\_.
5. The bit used to signify that the cache location is updated is \_\_\_\_\_.
6. In \_\_\_\_\_ protocol the information is directly written into main memory.
7. The method of mapping the consecutive memory blocks to consecutive cache blocks is called \_\_\_\_\_.
8. While using the direct mapping technique, in a 16 bit system the higher order 5 bits is used for \_\_\_\_\_.
9. The technique of searching for a block by going through all the tags is \_\_\_\_\_ search.
10. A control bit called \_\_\_\_\_ bit has to be provided to each block in set-associative.
11. The bit used to indicate whether the block was recently used or not is \_\_\_\_\_ bit.

---

## 13.4 VIRTUAL MEMORY

---

Initially the program resides in the secondary storage device like hard disk. Whenever the program is executed, a copy of that program is brought into main memory. When a program is large enough to fit into the main memory in one go, the operating system need to move program and data constantly between main memory and secondary storage. This gives an illusion to the programmer that has a very large main memory at his disposal. This concept is that automatically swaps programmed data blocks between main memory and secondary storage device are called virtual memory.

This concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. Each address that is referenced by the CPU goes through an address mapping from the so called virtual address to a physical address in main memory. A virtual memory system provides a mechanism for translating program-generated addresses into correct main memory locations. This is done dynamically, while programs are being executed in the CPU. The translation or mapping is handled automatically by the hardware by means of a mapping table.

### 13.4.1 Address Space and Memory Space

Let us start our discussion by defining address space and memory space.

- **Virtual Address:** An address used by a programmer is called virtual address.
- **Address Space:** Set of virtual address is known as address space. It is the set of addresses generated by programs as they reference instructions and data.
- **Physical Address:** Address of main memory is known as Physical Address.
- **Physical Space:** Set of main memory addresses is know as Physical Space. It is the memory space consists of the actual main memory locations directly addressable for processing.

The virtual address is used by programmers who have the illusion that he have a very large main memory, equal to the size of secondary memory, at his disposal. Therefore, the computers with virtual memory have address space greater than memory space. This can be further clerified using an example. If we have an auxiliary memory of size 1024K and main memory of size 32K.

Number of bits required to specify a virtual address unquily will be

$$1024K = 1024 \times 1024 = 2^{10} \times 2^{10} = 2^{20}$$

Therefore, 20 bits are required to specify any location in virtual space uniquely.

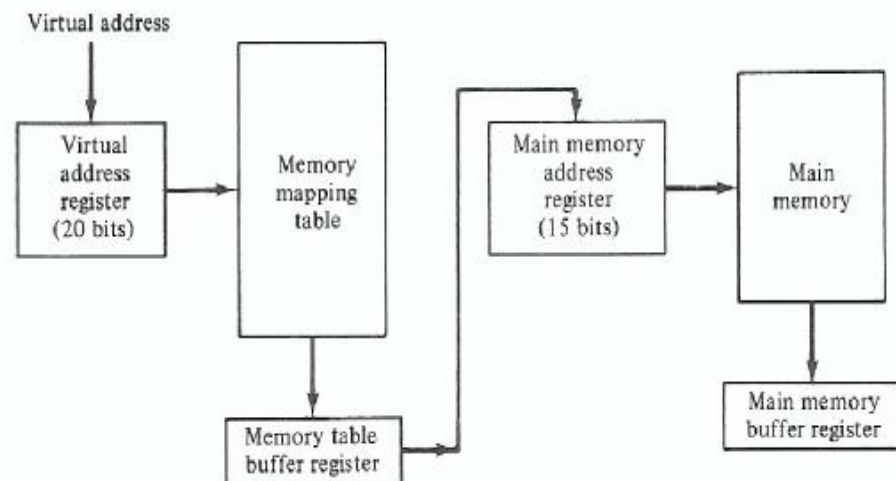
Similarly, number of bits required to specify a physical address will be

$$32K=32 \times 1024 = 25 \times 2^{10} = 2^{15}$$

Therefore, 15 bits are required to specify any location in physical space uniquely. But the instruction code is designed with address field of 20 bits and the main memory address is of 15 bits only. Hence, a mapping logic is required to map these 20 bits generated by the programmer to 15 bits main memory address. A table, known as mapping table, is used to map the virtual addresses to physical address. This mapping needs to be stored in the computer for translation to take place. There are three alternatives to store this table, with each alternative having its own merits and demerits.

- (a) A separate memory is used to store the mapping table. This will require an additional memory as well as an additional access to this memory for address translation. This increases the cost as well as it decreases the performance as two accesses to memory are required.
- (b) The mapping table is stored in the main memory. This does not increase the cost but, decreases the performance because this alternative also requires two accesses to the main memory.
- (c) Associative memory is used for storing the mapping table. In this case, the speed of the operation increases considerably as the associative memory facilitates the search operation. However, associative memory is one of the costliest memory; therefore, the cost also increases.

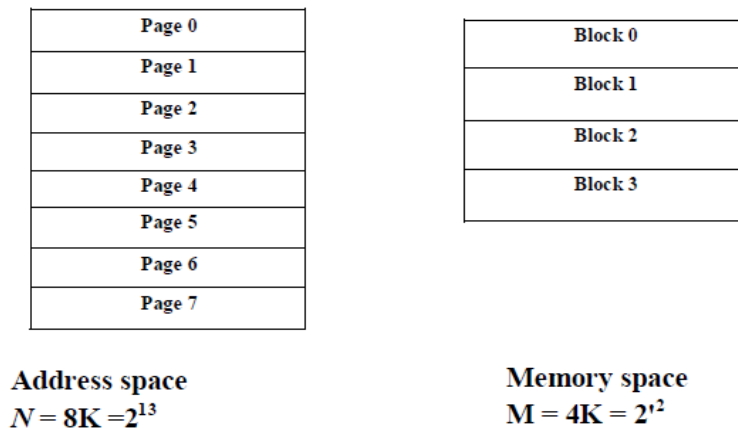
The Figure 139 below explains the address translation process in the virtual memory system when an additional memory as explained in (a) is employed to store mapping table.



**Figure 139: Virtual memory address translation**

### 13.4.2 Address mapping Using Pages

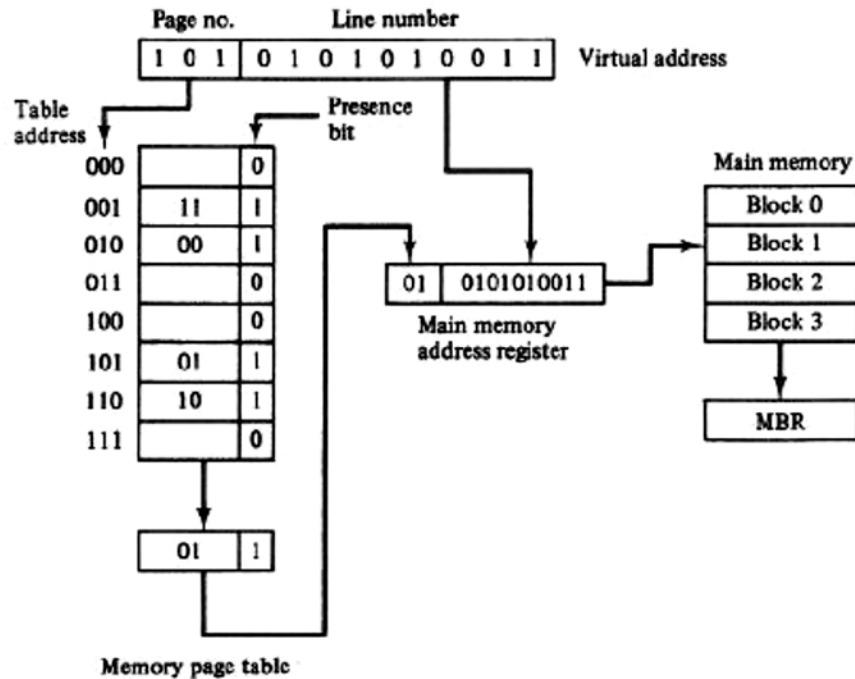
To facilitate the storing and reference the data, the memory is further divided into small groups. In the case of address space, these equal-sized group of memory is known as pages. And in the case of memory, these equal size group of memory is known as block. The address translation process is greatly facilitated if the size of the block and the page is same. For example: Consider a computer with an address space of 8K and a memory space of 4K. If we divide the address space and memory space into blocks and pages with size 1K i.e., the block size and page size is same(1K). This is shown in the Figure 140 below:



**Figure 140: Address and Memory Space**

Since the address space is 8K, therefore we have 8 pages of 1K each. Similarly, memory space is of 4K therefore, we have 4 blocks of 1K each. Since the page and block size is the same, with this arrangement, at any point of time, the memory space can accommodate 4 out of 8 pages in any of its 4 blocks. Now we will observe how the equal size of block and page facilitates the mapping process. The page and block size is 1K i.e. there are 1024 address locations ranging from 0 to 1023. Therefore, if we transfer any page from address space to memory space and we want to refer that page in the memory space, we need to map only the corresponding block number where the page is stored.

The 20-bit virtual address is divided into two parts. The first part specifies the page number and the second part denotes the word within the page (0 to 1023). If a page is transferred to memory space in the block, we need to map the page with the associated block. The word need not to be mapped since the block and the page size are equal. 4th word of the 2nd page, if transferred to the 3rd block in the memory space, it will be the same 4th word in the 3rd block. Mapping is required only for the 2nd page to 4th block i.e., the only mapping required is from a page number to a block number. This is explained in Figure 141 below:



**Figure 141: Memory table in paged system**

In the above figure, an auxiliary memory of 8K and main memory of 4K is considered. The 13-bit virtual address generated by the programmer, is divided into two parts, page number and line number. Page number consists of 3 bits which are used to denote one of the 8 pages.

The page table consists of eight locations ranging from 000 to 111. Since the memory space is of 4K therefore, only 4 pages can be accommodated by the main memory at any point of time. Therefore, the address of the block numbers corresponding to the pages that are transferred to those blocks are written in adjacent to the page numbers. The table shows that pages 1, 2, 5, and 6 are now available in main memory in blocks 3, 0, 1, and 2, respectively. A presence bit in each location indicates whether the page has been transferred from auxiliary memory into main memory. Using this scheme, the virtual address can be used to generate the main memory address.

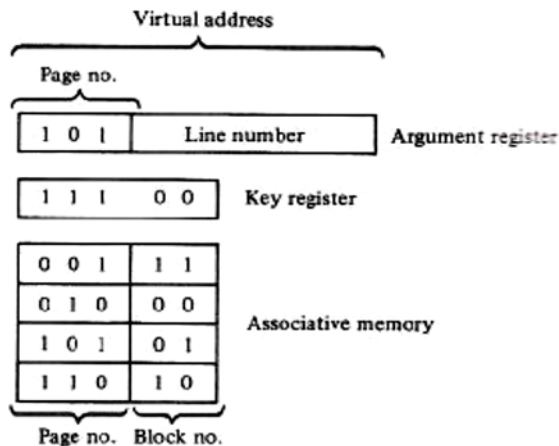
### 13.4.3 Associative Memory Page Table

The above implementation is useful when the size of address space and the memory space is small. As the size increases, the random-access memory page table is inefficient with respect to storage utilization. In the example cited above, when the address space and the memory space is 8K and 4 K respectively, we have clearly seen that eight words of memory are needed, one for each page, but at least four words will always be marked empty because main memory cannot accommodate more than four blocks. But if we consider an address space of 1024K words and memory space of 32K words with page and block size equal to 1K, the number of pages is 1024 and the number of 179



blocks 32. The capacity of the memory-page table must be 1024 words and only 32 locations may have a presence bit equal to 1. At any given time, at least 992 locations will be empty and not -in use, which is clearly a wastage of memory.

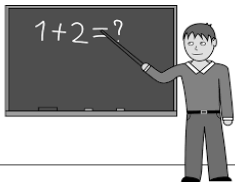
The above problem could be overcome, if the associative memory is used to store the page table as shown in Figure 142.



**Figure 142: Page table stored in associative memory**

In this case, the size of the memory is chosen according to the block size of the memory space, in contrast to the example in section 13.2.2, where the page table was selected according to the number of the pages in the address space. This will reduce the size of the page table considerably.

In associative memory, each block number is stored along with the associated page in the address space. Since the associated memory facilitates the content search, the page field in each word is compared with the page number in the virtual address. If a match occurs, the word is read from memory and its corresponding block number is extracted.



## Check Your Progress

1. The techniques which move the program blocks to or from the physical memory is called as \_\_\_\_\_.
2. The binary address issued to data or instructions are called as \_\_\_\_\_ address.
3. \_\_\_\_\_ is used to implement virtual memory organisation.
4. \_\_\_\_\_ translates logical address into physical address.
5. The virtual memory basically stores the next segment of data to be executed on the \_\_\_\_\_.
6. The associatively mapped virtual memory makes use of \_\_\_\_\_.

---

## 13.5 SUMMARY

---

1. A computer system have various type of memories like CPU registers, cache memory, main memory, secondary memory, etc.
2. All the momories have same function i.e. to store the data, but they differ in their speed and cost.
3. The base types that normally constitute the hierarchical memory system include registers, cache, main memory, and secondary memory.
4. The main memory constitutes of RAM and ROM, is the central storage unit in a computer system.
5. The CPU processing speed is very fast as compared to the time taken by a memory-access.
6. It has been found after analyzing a large number of programs that the references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of *locality of reference*.

---

## 13.6 ANSWERS TO CHECK YOUR PROGRESS

---

1. Secondary memory
2. Locality of reference
3. Mapping function
4. Replacement algorithm
5. Dirty bit
6. Write through
7. Direct
8. Tag
9. Associative
10. Valid
11. Dirty
12. Virtual memory organization
13. Logical
14. Memory Management Unit(MMU)
15. Memory Management Unit(MMU)
16. Secondary storage

---

## 13.7 TERMINAL QUESTIONS

---

1. What is principle of locality of reference.
2. What is memory hierarchy? Why hierarchy concept is used in memory?
3. What is an address space and the memory space?
4. Explain hit ratio and miss ratio.
5. Explain associative mapping in cache organization.
6. Explain direct mapping in cache organization.
7. Explain set-associative mapping in cache organization.
8. How the cache is initialized?
9. Explain the difference between the write-through and write-back method in cache.
10. What is virtual memory?
11. Explain virtual address mapping process using a diagram.
12. An address space is specified by 24 bits and the corresponding memory space by 16 bits.
  - (a) How many words are there in the address space?
  - (b) How many words are there in the memory space?
  - (c) If a page consists of 2K words, how many pages and block are there in the system?



# UNIT-14

---

## PERIPHERAL DEVICES

---

### Structure

- 14.0 Learning Objectives
- 14.1 Introduction
- 14.2 Peripheral Devices
  - 14.2.1 Keyboard
  - 14.2.2 Magnetic Tape
  - 14.2.3 Display System
- 14.3 Input/Output Interface
- 14.4 Asynchronous Data Transfer
- 14.5 I/O cards in personal computers
  - 14.5.1 Graphic card
  - 14.5.2 Sound Card
  - 14.5.3 Network Interface Card (NIC)
- 14.6 Summary
- 14.7 Answers to Check Your Progress
- 14.8 Terminal Questions

---

### 14.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

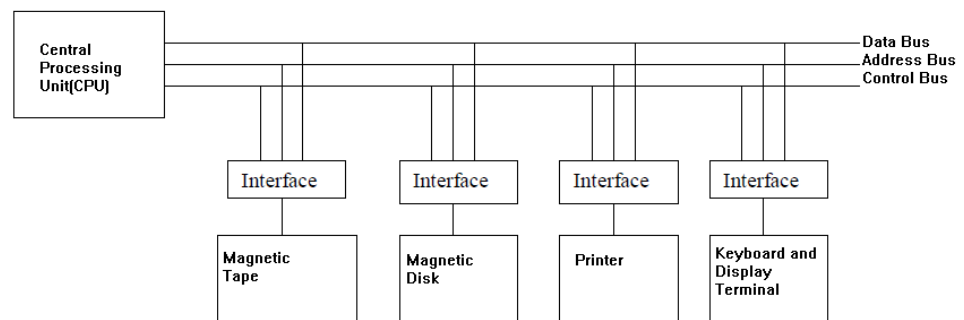
- Define peripheral devices.
- Understand the necessity of an interface.
- Define synchronous data transfer.
- Define asynchronous data transfer.
- Identify various peripheral devices.

---

## 14.1 INTRODUCTION

---

A CPU is referred to as the brain of the computer. For any processing on data, the data need to be brought inside the CPU. The data is transferred inside the CPU via input devices and the processed information is displayed via output devices. These input/output devices like, printer, keyboard, mouse, monitor, etc. are collectively known as Peripheral devices. At times the information is stored in the secondary storage devices for future reference. Clearly, the CPU needs to communicate with memory and peripheral devices. Therefore, the CPU requires some path, known as bus, through which it can communicate to these devices. Figure 143 shows how CPU is connected to these devices.



**Figure 143 : Processor Interconnection with Peripheral Devices**

In the figure above, the peripheral devices are connected to CPU via interface. There are many differences in the implementation of CPU and the peripherals devices, some of the major differences are listed below:

1. CPU is an electronic device whereas the peripheral devices are electromechanical (like printer, which have mechanical motors for page movement) and electromagnetic devices (like secondary memory). Therefore, the technology behind these two entities and the manner of operation is different.
2. CPU transmits and processes the data at very fast rate whereas the data transfer rate of peripheral devices is comparatively very slow. Moreover, CPU transmits the data in parallel whereas peripheral devices usually transmit data serially.
3. Data codes and formats in peripherals differ from the word format in the CPU and memory.

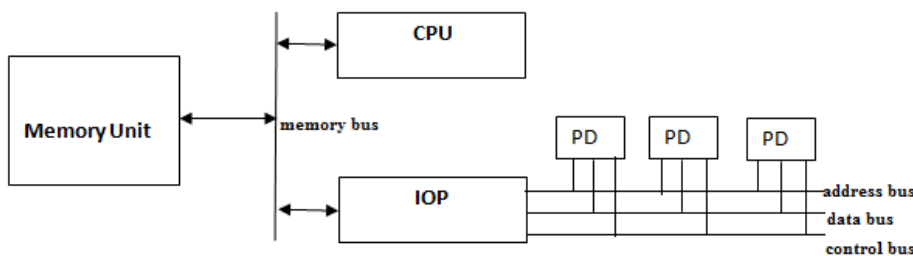
To overcome these mismatches, a special hardware, known as interface is connected between CPU and these peripherals device to take care of all the above issues and to supervise and synchronize all input and output transfers. These components are called **interface**.

CPU needs to communicate with memory and other peripherals devices. There are many peripheral devices attached to a computer, so to locate a specific device, an address need to be specified. Similarly, to read/write data from/to memory, an address need to be specified. Also, CPU need to specify the control information i.e., what operation is to be performed. Like in case of memory, CPU needs to specify whether it is a read operation or a write operation. After specifying the address and the control information, the data need to be transferred between CPU and peripheral devices/memory. For this purpose, bus is required. Bus is a conducting path that connects the CPU with other devices. Based on the purpose for which the bus is used, it is categorised into three categories.

- a. *Address Bus:* An address bus is used by the CPU to transmit the address of the peripherals/memory location. Since, it is used by CPU only, this bus is unidirectional.
- b. *Control Bus:* It is used by the CPU to control and regulate the various devices attached to it. This bus is bidirectional.
- c. *Data Bus:* It is used by CPU and the devices attached to the CPU to transfer data. This Bus is bidirectional.

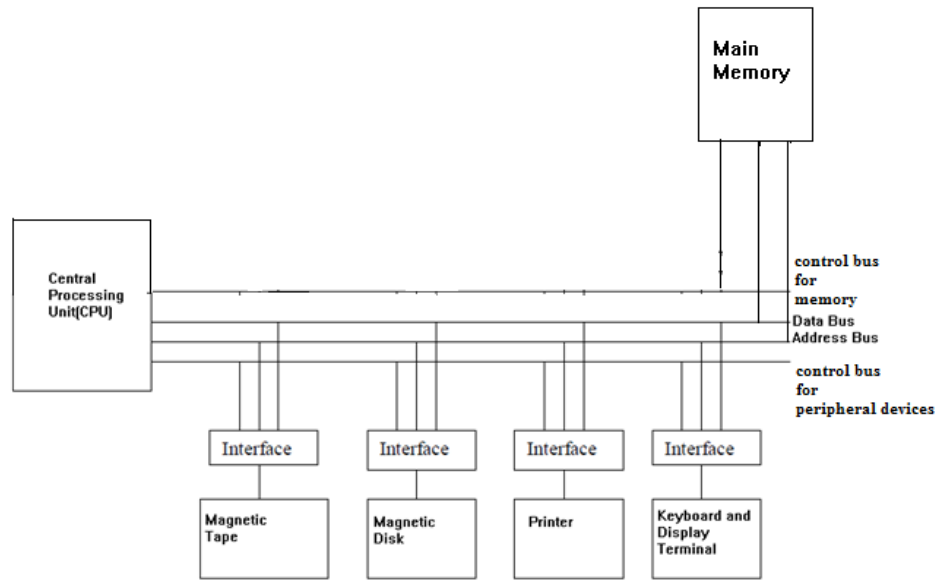
Apart from peripheral devices, the CPU is also connected to main memory and needs to frequently communicate with it. So bus is also required to connect the CPU with memory also. There are three possible architectures to connect the CPU with memory and peripheral devices.

1. Different set of address, data and control buses are used to connect memory and other Peripheral Devices (PD), as shown in Figure 144. In this case a separate Input/Output processor (IOP) is required to control the peripheral devices. The memory communicates with CPU and IOP using memory bus whereas the IOP communicates with its separate address data and control bus.



**Figure 144: Interconnection of Peripheral Devices and Memory using different set of Buses**

2. The second alternative is to connect CPU to memory and peripheral is using different set of control buses but data and address buses are common.



**Figure 145: Interconnection of CPU and Peripheral Devices using Common Address & Data Bus and different Control Buses**

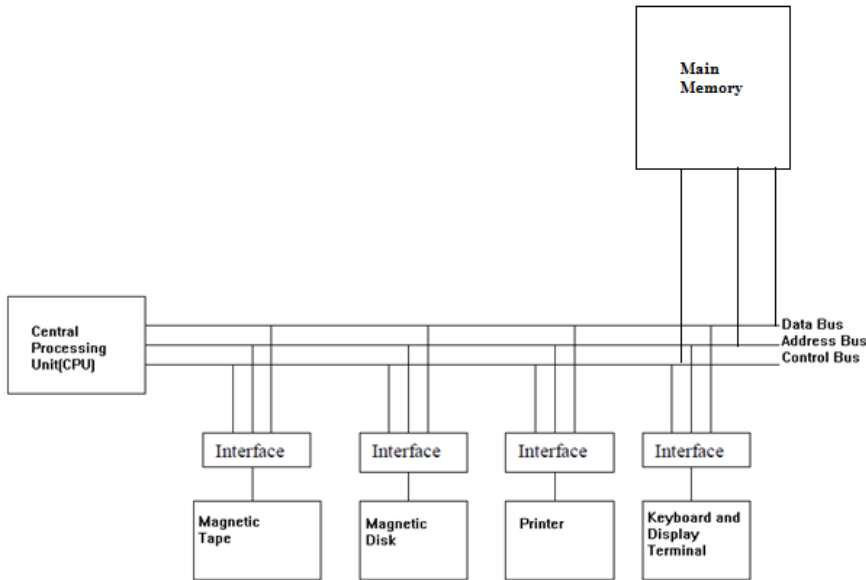
The same is shown in Figure 145. When CPU needs to communicate with memory or I/O device, it places the data in the data bus and specifies the address using address bus. The distinction between I/O device and memory is made via control lines. If CPU wants to interact with memory, it will use the control lines dedicated for memory. This will isolate all the I/O devices and the memory will clearly recognize that the address is directed to memory, not I/O devices. Similarly, if CPU wants to interact with I/O devices, it will use the control lines dedicated for I/O devices. Whenever the CPU communicates with memory, the I/O device remains isolated and when CPU communicates with I/O devices, memory becomes isolated. Therefore this configuration is also known as Isolated I/O method.

3. The third alternative is to use all the busses i.e., data, addresses and control bus to connect the CPU with peripheral and memory. The same is shown below in Figure 146.

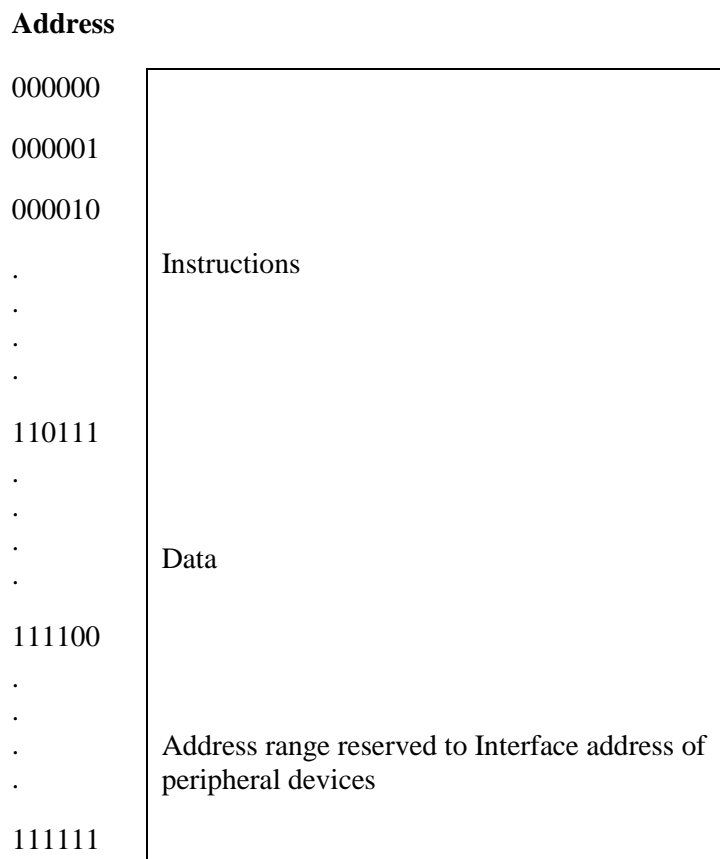
In this case, some of the address range of the main memory is reserved to peripheral devices. This configuration is known as memory-mapped I/O. In this case, the CPU does not make any distinction between memory and peripheral devices and use the same set of read and write signals for both memory-read/ peripheral-read and memory write/peripheral-write operations as the interface of the peripheral devices are treated similar to memory address( refer Figure 146). Thus, this reduces the number of instructions in the instruction set of the computer as the same instructions can be used for memory as well as 186



peripheral devices. The drawback of this scheme is, the memory cannot be fully utilized as some of the address from the address range of memory is reserved for peripheral address.



**Figure 146: Interconnection of CPU and Peripheral Devices using Common Address, Data Bus Control Bus**



**Figure 147: Address Range in Memory-Mapped I/O Configuration**

---

## 14.2 PERIPHERAL DEVICES

---

The CPU of a computer interacts with the outer world using Input/Output devices, which are attached to the computer and are commonly known as peripheral devices. Some of the most commonly used peripherals devices are keyboard, printer, magnetic tapes, magnetic disks, display unit, etc. Now let us discuss some of these I/O and peripheral devices in detail.

### 14.2.1 Keyboard

The CPU needs data for processing. Depending on the type of data to be processed, there are several ways in which the data can be transmitted to CPU. If the data is an image which is to be processed, then the preferred input devices would be camera, scanner, etc. If an audio is to be processed, then microphone would be an idle choice. However, in most of the cases the input data is text and a keyboard is used to input the alphanumeric information into computer. Figure 148 below shows a normal keyboard of a computer.



**Figure 148: A keyboard**

There are many variant of a computer available in the market nowadays and most of them have between 80-110 keys which consists of set of alphabet keys, numeric keys, function keys, control keys, arrow keys and operator keys. The keys of the typewriter are laid out on the same pattern as typewriter. This pattern is known as QWERTY, which is the first six alphabets of the top row from left to right. The logic behind this arrangement was to avoid colliding and jamming the metal arms of the typewriter while typing. There are several other variants of the keyboards like ADCBS, XpeRT, QWERTZ, AZERTY etc.

### 14.2.2 Magnetic Tape

After the data is processed by the CPU, the information may be stored for future reuse. For this purpose, a computer has a variety of internal and external memory elements. Some of the popular options for external

storage are magnetic disks and magnetic tapes. The per bit storage cost of magnetic disk is higher than the magnetic tape. But Magnetic disks have less access time( faster) as compared to magnetic tapes. Generally these are used for system backup.

### **14.2.3 Display System**

When the information is generated by the CPU, we need to confirm whether the information generated is correct and relevant. For this purpose, variety of display devices are used. For example, monitor, printer, plotter, etc.

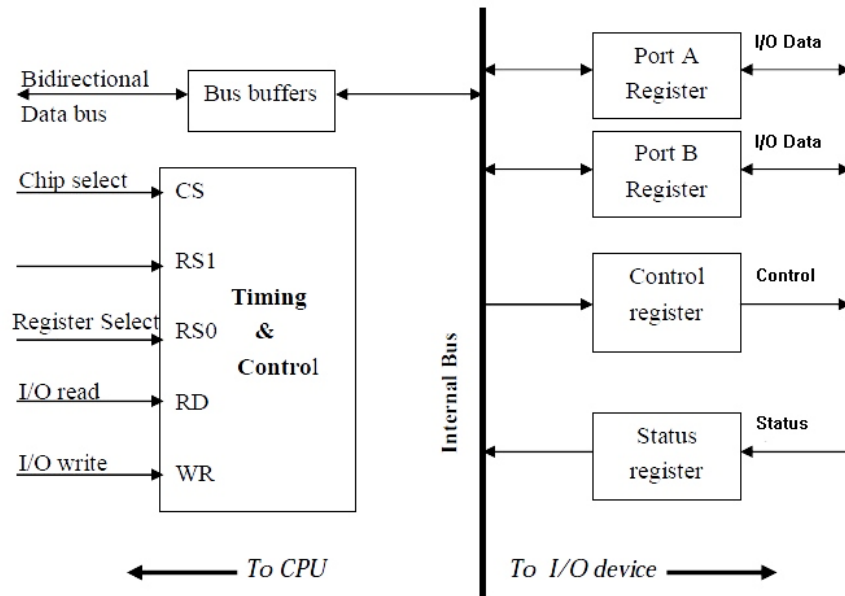
As discussed earlier, these peripheral devices are implemented using different technologies, varies in speed and operation. Therefore, an interface, which acts as a mediator is required between the CPU and the peripheral device to communicate effectively. Now let us discuss the operation and architecture of an interface unit in detail.

---

## **14.3 INPUT/OUTPUT INTERFACE**

---

The importance of interface unit is already pointed out in the introduction section. Now let us discuss an example of an interface device to bring more clarity on the working of an interface unit. Figure 14.9 shown below explains the block diagram of interface unit. The right side of the Interface unit is connected to peripheral devices. It consists of four registers namely, Port A register, Port B register, Control register and Status register. These registers can communicate with the peripheral device attached to it. Port A and Port B are bi-directional registers used for sending and receiving data. Control register is used by the CPU to send the control information to the peripheral device. Status register is used by the CPU to check the status of the device or the data transfer that is currently taking place. The left side of the I/O interface unit consists of timing and control unit which receives the signals from CPU. The timing and control unit consists of a chip select(CS) signal, which is used by the CPU to select the interface of a particular device. This chip select logic is generally connected to the address bus of the CPU via decoder. Whenever the CPU generate the address of the device, it place the address of that device in the address bus. The decoder attached to the Chip select logic continuously monitors the address line and as soon as the address in the address bus of the CPU matches the address of a device the decoder attached to the chip select logic enables CS signal and the interface is selected and available to the CPU for further orders.



**Figure 149: Block diagram of an I/O interface**

The CPU communicates to the device via interface registers. If CPU wants to send the control information, it will use control register. Similarly, if the CPU wants to check the status of the device, it will check the bits of status register. In case, the data is to be transmitted/received, CPU uses port A and port B for this purpose. Now the question arises, how the CPU select a particular registers among the four registers. The Answer to this question is, CPU uses two register select lines, RS0 and RS1 to select between the four register. The Table 22 show how the register select lines are used to select a particular register.

CS	RS1	RS0	Selected Register
0	X	X	None: as the chip is not selected
1	0	0	Port A
1	0	1	Port B
1	1	0	Control Register
1	1	1	Status Register

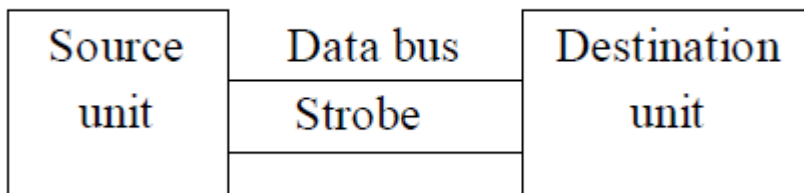
When the chip select logic is 0, the data bus is in high-impedance state. When the address bus contains the address of a particular device, the decoder attached to the chip select logic of the device enables the CS logic and one of the register can be selected using the RS1 and RS0 line. There are two additional lines RD and WR to differentiate between the read and write operation.

---

## 14.4 ASYNCHRONOUS DATA TRANSFER

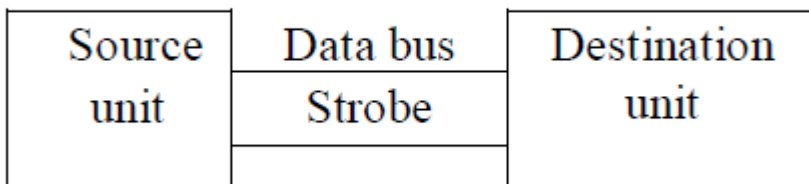
---

A computer is a digital system which is composed of variety of sub-systems like CPU, memory, I/O units. The operations of these units are synchronized by a clock pulses, which are generated by a common pulse generator. If the CPU and the interface unit shares a common clock then these two unit are said to be *synchronous* to each other. Whereas if the two units have its own private clock, the two units are said to be *asynchronous* to each other. Whenever two asynchronous units communicate with each other, some control signals are needed to be sent along with the data signals. These control signals consists of the time information at which the data is being transmitted. One of the mechanism to achieve this is by sending a strobe pulse from the source, which consists of the information for the destination unit regarding the time of data transfer.



**Figure 150 : Source-initiated strobe for data transfer**

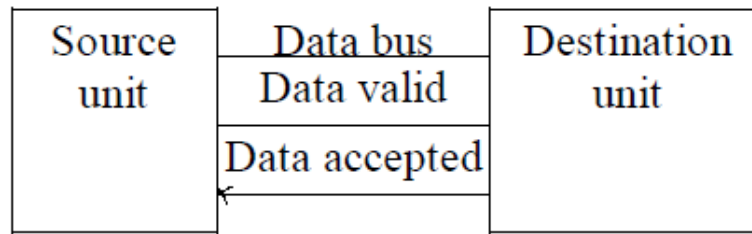
The strobe may be activated by either source or the destination unit. The Strobe signal is disabled indicates that the data bus does not contain valid data. New valid data will be available only after the strobe is enabled again.



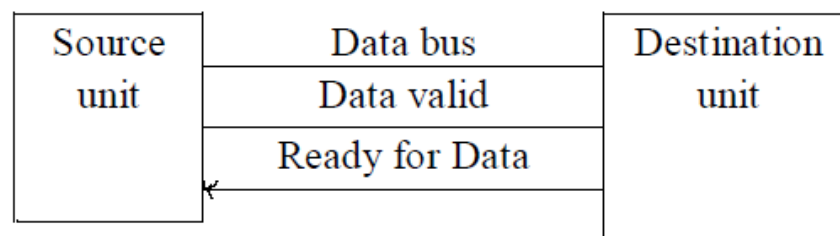
**Figure 151: Destination-initiated strobe for data transfer**

But this technique have a serious drawback. The source does not have any mechanism to know whether the data sent to the destination is received by the destination unit, and if yes, when? To overcome this problem, another technique, known as *handshaking* is used. In this method, whenever a source need to transmit data, it sends a strobe signal to the destination unit which is a signal for the destination unit that the data bus now contains valid data. After this, the data is placed in the data bus by the source. Once

the data is received by the destination, the destination returns the acknowledgement signal which is a signal to the source that the destination unit has received data successfully.



**Figure 152: Source-initiated transfer using handshaking**



**Figure 153: Destination-initiated transfer using handshaking**

---

## 14.5 I/O CARDS IN PERSONAL COMPUTERS

---

Personal Computer is built on open architecture i.e. it can be expanded in future, if need arises. The expansion can be done using expansion slots, which are the sockets present in the motherboard of the computer wherein expansion cards, also known as interface cards, plug-in boards, or adapter cards can be plugged-in. Graphic card, sound card, network interface card etc are some of the popular I/O cards that are popularly known. These expansion cards are connected to the buses of the motherboard so that the device attached to the expansion card can communicate with the CPU. As we already know that the bus that connects the CPU with the main memory is known as system bus. The bus that connects the CPU with the expansion slot is known as expansion bus. Some of the well-known expansion buses are ISA (Industry Standard Architecture), PCI(Peripheral Component Interconnect), and AGP(Accelerated Graphic Port). Now we will discuss some of the popular I/O cards used in PC.

### 14.5.1 Graphic card

Graphic card is already chipped in the motherboard of the personal computer. It is a card through with the monitor of the computer is connected. It is also known as video card or video adapter. Its function is to convert the signals received from the CPU into video signals or image so that it can be displayed in 193

an output device like monitor. It comes in the variety of configuration like 8 MB, 16 MB or 32 MB.

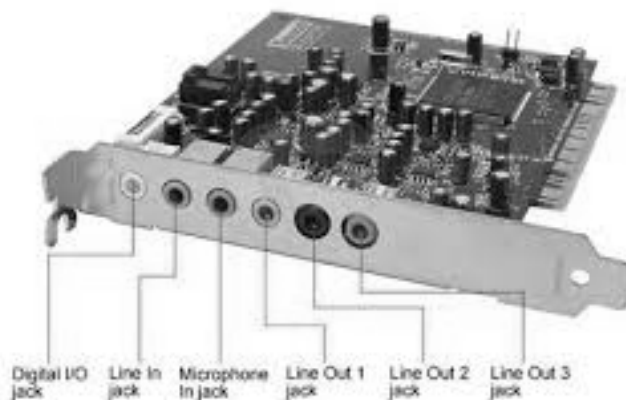


**Figure 154: Graphic card**

As shown in Figure 154, like CPU, the graphic cards have its own processor, BIOS, and RAM and are designed for performing complex mathematical and geometric calculation necessary for graphics rendering.

### 14.5.2 Sound Card

Like Graphic cards, the sound cards are also preinstalled in the motherboard of PC nowadays and it is used to transmit the digital sound through speakers and headphones.



**Figure 155: Sound card 194**

The basic function of a sound card is to convert digital data into analog information and analog information to digital data. This can be accomplished either by using separate A to D (Analog to Digital ) and D to A(Digital to Analog) convertor or a single CODEC(coder/decoder) chip which can perform both the functions. Music is produced by sound card using a process called wavetable synthesis. It is a method of creating music based on wave table, which is a collection of digitalized sound

samples taken from recording of actual instruments. A sound card converts analog information it has received through one of its inputs into digital data by taking precise measurements of an analog sound wave at a rate of thousands of times per second. These sound samples are mixed and stored on the sound card. When the user wants to create sound/music from computer, the sound card reproduces analog sound waves from those digital measurements.

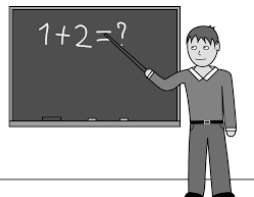
### 14.5.3 Network Interface Card (NIC)

This card is also pre-installed now-a-days on the motherboard of a computer and its basic purpose is to provide a physical and logical link for a PC to a network



**Figure 156: NIC card**

Networks transmit data in a serial data format (1 bit at a time), and the data bus of the PC moves data in a parallel format (8 bits at a time). The NIC acts as an interface between the PC and the network and its primary task is to convert the signal from serial to parallel format or from parallel to serial format, depending on its direction. The NIC also formats the data as required by the network architecture.



### Check Your Progress

1. The input/output devices like, printer, keyboard, mouse, monitor, etc. are collectively known as \_\_\_\_\_ devices.
2. If the CPU and the interface unit shares a common clock then these two unit are said to be \_\_\_\_\_ to each other.
3. ISA stands for \_\_\_\_\_.
4. PCI stands for \_\_\_\_\_.
5. AGP stands for \_\_\_\_\_.
6. CODEC stands for \_\_\_\_\_.
7. NIC stands for \_\_\_\_\_.



---

## 14.6 SUMMARY

---

1. A CPU is referred to as the brain of the computer. For any processing on data, the data need to be brought inside the CPU.
2. The data is transferred inside the CPU via input devices and the processed information is displayed via output devices.
3. CPU needs to communicate with memory and other peripherals devices. There are many peripheral devices attached to a computer, so to locate a specific device, an address need to be specified.
4. Apart from peripheral devices, the CPU is also connected to main memory and needs to frequently communicate with it.
5. In this case, some of the address range of the main memory is reserved to peripheral devices. This configuration is known as memory-mapped I/O.
6. The CPU of a computer interacts with the outer world using Input/ Output devices, which are attached to the computer and are commonly known as peripheral devices.
7. Some of the most commonly used peripherals devices are keyboard, printer, magnetic tapes, magnetic disks, display unit, etc.
8. A computer is a digital system which is composed of variety of sub-systems like CPU, memory, I/O units. The operations of these units are synchronized by a clock pulses, which are generated by a common pulse generator.

---

## 14.7 ANSWERS TO CHECK YOUR PROGRESS

---

1. Peripheral
2. Synchronous
3. Industry Standard Architecture
4. Peripheral Component Interconnect
5. Accelerated Graphic Port
6. coder/decoder
7. Network Interface Card

---

## 14.8 TERMINAL QUESTIONS

---

1. What are the three possible architecture to connect the CPU with memory and peripheral devices? Explain in details using diagram.
2. What is wavetable synthesis?
3. What is isolated I/O method?

4. What is memory mapped method?
5. How isolated I/O method is different from memory mapped method.
6. What is the purpose of using interface unit?
7. Explain the working of I/O interface using a diagram.
8. Discuss some of the popularly used I/O cards in personal computers.
9. What is asynchronous data transfer?
10. What is Handshaking?

# UNIT-15

---

## INTRODUCTION TO 8085 MICROPROCESSOR

---

### Structure

- 15.0 Learning Objectives
- 15.1 Introduction
- 15.2 Architecture of Microprocessor
  - 15.2.1 8085 Microprocessor
  - 15.2.2 The Internal Architecture of 8085
- 15.3 Instruction Set of 8085 Microprocessor
  - 15.3.1 Data Transfer Group
  - 15.3.2 Arithmetic Group
  - 15.3.3 Logical Group
  - 15.3.4 Branch Control Group
  - 15.3.5 I/O and Machine Control Group
- 15.4 SUMMARY
- 15.5 Answers to Check Your Progress
- 15.6 Terminal Questions

---

### 15.0 LEARNING OBJECTIVES

---

After reading this unit, you will be able to:

- Define a Microprocessor.
- Explain the pin diagram of a 8085 microprocessor.
- Understand the instruction set of 8085 microprocessor.

---

### 15.1 INTRODUCTION

---

A microprocessor( $\mu$ p) is an electronic device that is used by the computer to its processing. The term CPU and microprocessor are often used interchangeably, but there is a difference. The CPU of a computer consists of Arithmetic Logic Unit (ALU) and Control Unit(CU). In the

earlier days, when the chip fabrication technology was in its initial phase, it was not possible to integrate the ALU and CU into the same chip. The ALU and CU were connected to each other and the combined unit was known as CPU. With the advent of LSI, VLSI and VVLSI technologies, it was possible to fabricate both ALU and CU into the same chip. When the ALU and CU were fabricating into the same chip, it is known as a microprocessor. So a microprocessor, similar to a CPU, performs the basic arithmetical, logical, and input/output operations of the system. Unlike, PC which is designed to be a general purpose machine, a microprocessor can be designed for both general purpose and specific purpose. Almost all of your automatic home appliances like an automatic washing machine, dishwasher, digital set-top box, music system, air conditioners, etc. Even it finds place in your car, motorcycle and scooter. This small magical device has affected our day-to-day life to such an extent that today we cannot think of our life without it. Be it our public transportation infrastructure, where it find place in traffic lights, railways, transport and fright management. Medical sciences' sophisticated equipments and automatic patient monitoring systems are designed using microprocessors. Most of the communication and handheld devices like smartphones, GPS systems, etc have a microprocessor inside it. Now in the next section, we will take a closer look at the working and functionality of a microprocessor.

---

## **15.2 ARCHITECTURE OF MICROPROCESSOR**

---

A microprocessor is a programmable device which can perform different sets of operations on the data it receives as an input depending on the sequence of instructions supplied in the given program.

The processing power of a microprocessor depends on its instruction set. Like a CPU, a microprocessor can perform only those operations for which it is designed. The collection of all such instructions for which the operations are defined is called an instruction set. Larger the number of instructions in the instruction set, better is the execution power of a microprocessor. A microprocessor primarily consists of three units:

- The Arithmetic/Logic Unit (ALU)
- The Control Unit.
- Internal Register Set

Depending on the width of the data bus of a microprocessor, it can be categorized into 8-bit, 16-bit, 32-bit or 64-bit microprocessor. The width of the data bus signifies the number of data bits a microprocessor can process simultaneously. Our aim here is to introduce with the basics of microprocessor and its internal working, therefore we will discuss a simple 8-bit microprocessor. There are variety of microprocessors available in the market manufactured by different companies like from Motorola 6800, ZiLOG Z80 and Intel 8085. We will discuss the most popular one i.e., Intel 8085 in detail.

## 15.2.1 8085 Microprocessor

Intel launched its first 8-bit microprocessor in 1972 and names it Intel 8008. Soon after, it improved version, Intel 8080 was launched. Finally, Intel launched 8085 in 1977, which was much powerful than its two earlier versions. Intel 8085 is a 8-bit general purpose microprocessor capable of addressing 64K memory. It has 40 pins and runs on +5 V power supply. It operates with 3 MHz clock. In 8085, the 8-bit data bus was multiplexed with the lower part of 16-bit address bus so that the number of pins are limited to 40. It has a 16-bit address bus, hence it is capable of addressing  $2^{16} = 64$  KB memory.

It consists of:

- Control unit: control microprocessor operations.
- ALU: performs data processing function.
- Registers: provide storage internal to CPU.
- Interrupts
- Internal data bus

The pin diagram of Intel 8085 is shown in fig. 97. The pins on the chip can be grouped into 6 groups:

- Address Bus.
- Data Bus.
- Control and Status Signals.
- Power supply and frequency.
- Externally Initiated Signals.
- Serial I/O ports.

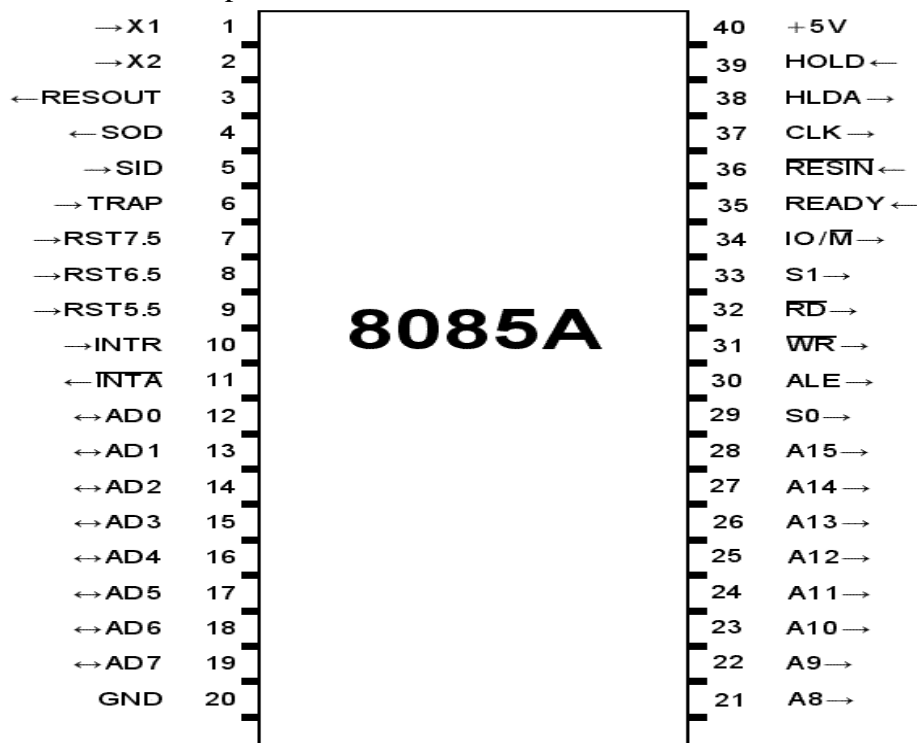
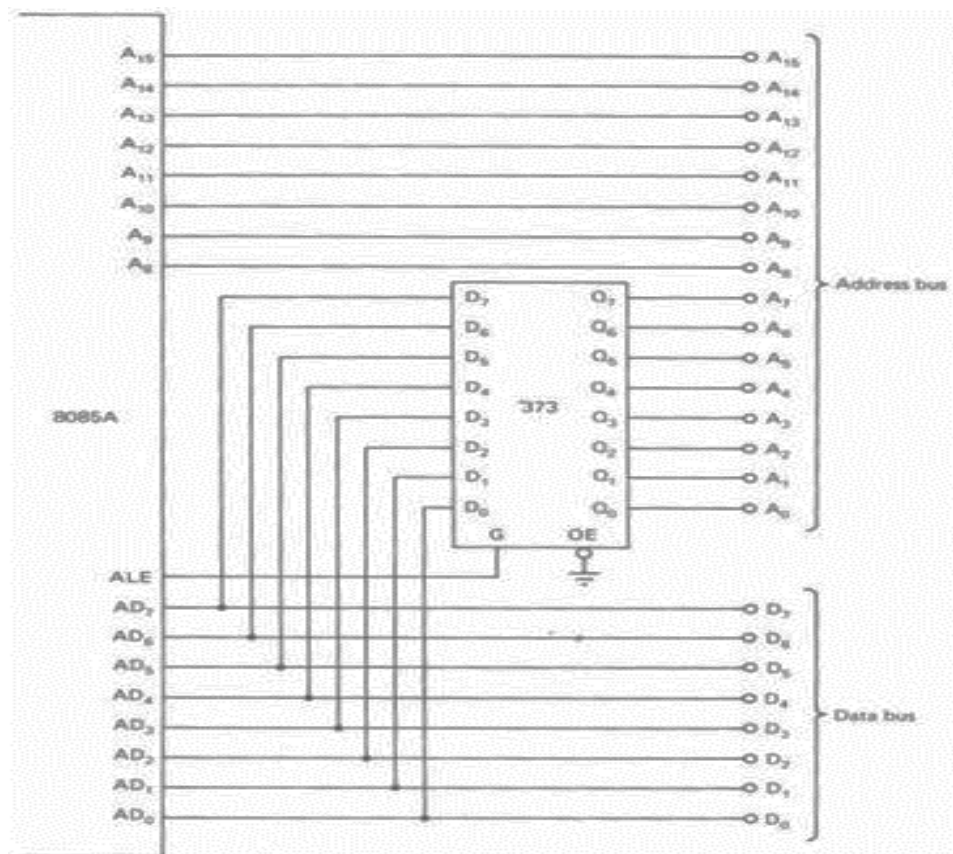


Figure 157 : Pin Diagram of Intel 808536

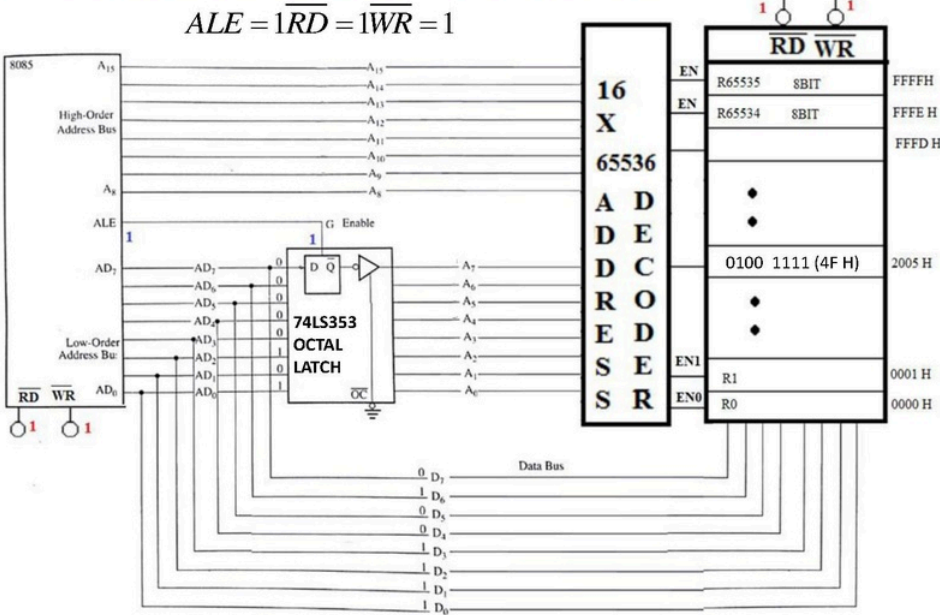
1. *Address and Data Signals:* The address bus has 8 signal lines A8 – A15 which are unidirectional. The other 8 address bits are multiplexed (time shared) with the 8 data bits. The bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time. During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits. In order to separate the address from the data, we use a latch to save the value before the function of the bits changes.
2. *Control and Status Signals:* There are 4 main control and status signals. These are:
  - a. ALE(pin 30): Address Latch Enable. This signal is a pulse that become 1 when the AD0 – AD7 lines have an address on them. It becomes 0 after that. This signal can be used to enable a latch to save the address bits from the AD lines.
  - b. RD(pin 32): Read (Active low).  
WR(pin 31): Write(Active low).
  - c. IO/M(pin 34): This signal specifies whether the operation is a memory operation (IO/M=0) or an I/O operation (IO/M=1).
  - d. S1(pin 33) and S0(pin 29) : Status signals to specify the kind of operation being performed. Usually not used in small systems.



## DEMULTIPLEXING THE BUS AD7- AD0

AD7 - AD0 to be used as ADDRESS BUS ALONG WITH A15-A8

Ignoring Chip Select signal



**Figure 158: Multiplexed Data and Address Bus in Intel 808537**

3. *Power Supply and Frequency:* There are 3 important pins in the frequency control group.
  - a. X1(pin 1) and X2(pin 2) are the inputs from the crystal or clock generating circuit. The frequency is internally divided by 2. So, to run the microprocessor at 3 MHz, a clock running at 6 MHz should be connected to the X1 and X2 pins.
  - b. CLK (OUT)(Pin 37): An output clock pin to drive the clock of the rest of the system.
4. *Interrupts and Externally initiated signals:*
  - a. INTR(pin 10): Input Request. This signal is used as a general-purpose interrupt.
  - b. INTA(Active Low)(pin 11): Interrupt Acknowledge. This signal is used to acknowledge an interrupt.

RST 7.5(pin7:} Restart Interrupts. These are vectored interrupts that transfer 8 the control of the program to specified memory locations.

RST 6.5(pin8)  
RST 5.5(pin 9):

- d. TRAP(pin 6): This is a non-maskable interrupt and has a highest priority.
  - e. HOLD(pin 39): Whenever a peripheral device, such as DMA want the hold of the address and the data bus, this signal is initiated by the peripheral device.
  - f. HLDA(pin 38): Hold Acknowledge. This signal acknowledges the HOLD request.
  - g. READY(pin 35): This signal is used to delay the microprocessor Read or Write cycles until a slow peripheral device is ready to send or receive data. When this signal is low, the microprocessor waits for a integral number of clock cycles until it goes high.
- There are two kinds of RESET signals in 8085:
- h. RESET IN(pin 36): an active low input signal, Program Counter (PC) will be set to 0 and thus MPU will reset.
  - i. RESET OUT(pin 3): an output reset signal to indicate that the  $\mu$ p was reset (i.e. RESET IN=0). It also used to reset external devices.

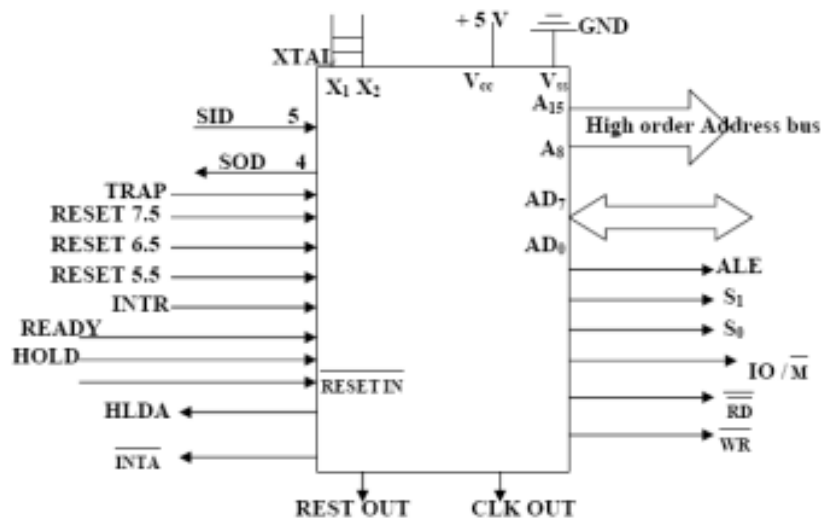
5. *Serial I/O Ports:*

- a. SID (Input)(pin5): Serial input data line The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
- b. SOD (output)(pin 4): Serial output data line. The output SOD is set or reset as specified by the SIM instruction.

There are two more pins left:

- a. Vcc(pin 40): +5 volt supply.
- b. Vss(pin 20): Ground Reference.

The signal groups of Intel 8085 microprocessor is shown in Figure 159.

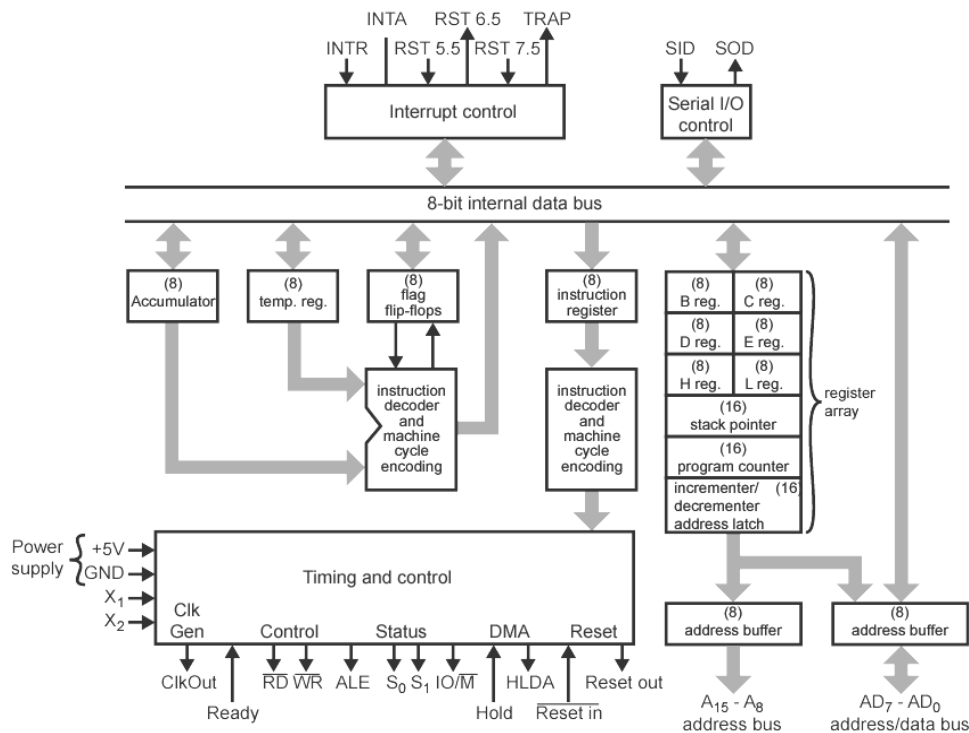


**Figure 159: Signal Groups of Intel 8085**



## 15.2.2 The Internal Architecture of 8085

Now let discuss the internal architecture of the 8085 microprocessor in detail. Figure 160 explains the internal architecture of the 8085.



**Figure 160: Internal Architecture of 8085 Microprocessor**

The ALU consists of arithmetic and logic circuits to perform arithmetic and logic operations. It also consists of register set to perform these operations on the data. The details of these registers are discussed below.

The 8085 microprocessor contains seven 8-bit register which are directly accessible to the programmer. These registers are named as A, B, C, D, E, H, and L.

A	Status Register
B	C
D	E
H	L
Program Counter	
Stack Pointer	

**Figure 161: Register Set of 8085 Microprocessor**

A is the 8-bit accumulator where all the operation on data takes place. The other six registers can be used as 8-bit register or a 16-bit register pair to manipulate 16-bit data. The register pair is as follows: BC, DE, and HL. In case a operation on a 16-bit data is to be performed, the HL pair can be used as a 16-bit accumulator. Apart from that, it also consists of two 16-bit registers. PC, program counter, controls the sequencing of the execution of instructions and is used to store the address of the next instruction to be executed. And SP, stack pointer, is used to point the address of the top-most element of the stack. The register set is shown in Figure 161.

It register set also contains 8-bit status register. Each bit of this status register contains a flag, which is a 1-bit flip-flop. These status flags are affected by the arithmetic and logic operations before or after the operation. There are six status flags in the status register and these are S (sign flag), Z (zero flag), AC (auxiliary carry flag), P (parity flag) & CY (carry flag).

<b>D7</b>	<b>D6</b>	<b>D5</b>	<b>D4</b>	<b>D3</b>	<b>D2</b>	<b>D1</b>	<b>D0</b>
S	Z		AC		P		CY

**Figure 162: Status Flags**

The use of these flags is explained below:

- a. S(sign flag): The sign flag is set if bit D7 of the accumulator is set after an arithmetic or logic operation.
- b. Z(zero flag): Set if the result of the ALU operation is 0. Otherwise is reset. This flag is affected by operations on the accumulator as well as other registers. (DCR B).
- c. AC(Auxiliary Carry): This flag is set when a carry is generated from bit D3 and passed to D4 . This flag is used only internally for BCD operations.
- d. P(Parity flag): After an ALU operation, if the result has an even # of 1s, the p-flag is set. Otherwise it is cleared. So, the flag can be used to indicate even parity.
- e. CY(carry flag): This flag is set when a carry is generated from bit D7 after an unsigned operation.
- f. OV(Overflow flag): This flag is set when an overflow occurs after a signed operation.

Whenever an Instruction from the memory is fetched, the instruction is placed inside a 8-bit register, known as Instruction Register(IR). A decoder is attached to the Instruction Register which enables the CPU to decode the instruction and take appropriate action. Suppose, after decoding the instruction, it was found that it was an addition instruction, the CPU will generate necessary control signals to initialize the adder and fetch the operand either from the memory or the input device.

The 8085 microprocessor has 8-bit data bus and 16-bit address bus. The address bus has 8 signal lines A8 –A15 which are unidirectional. The other lower order 8 address bits are multiplexed (time-shared) with the 8 data bits. So, the bits AD0 –AD7 are bi-directional and serve as A0–A7 and D0 –D7 at the same time. During the execution of the instruction, these lines carry the address bits during the early part, and then during the late parts of the execution, they carry the 8 data bits. In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

Now let us discuss how the demultiplexing of AD7-AD0 is done to serve the dual purpose i.e. the same line are used as address lines and data lines. The high order bits of the address remain on the bus for three clock periods. However, the low order bits remain for only one clock period and they would be lost if they are not saved externally. Therefore, an external latch is used to save the value of AD7–AD0 when the lines are carrying the address bits. Address Latch Enable(ALE) signal is used to enable the latch. Whenever AD7- AD0 is to be used for the data bus, the ALE goes low. Direct Memory Access(DMA) technique is used when a fast speed I/O device want to transmit the data to memory at high speed and the speed of CPU limits the speed of transfer. In this case, the CPU is bypassed and the control of the data and address buses is given to the transmitting device. Once the transfer is complete, the control of Data and Address bus is relinquished to the CPU. To facilitate DMA transfer, HOLD and HLDA signals are used. Whenever an I/O device request for DMA transfer, it enables the HOLD line. As soon as the HOLD line is enabled, the microprocessor data and the address bus of the CPU are placed in the high impedance state and the control of the buses is transferred to I/O device. After this, the HLDA signal is initialized by the CPU which is a signal for the I/O device to start the transfer of data. Once the data transfer is complete, the HOLD signal is disabled and the control of buses is returned to CPU.

An Interrupt is a mechanism by which an I/O or an instruction can suspend the normal execution of processor and get itself serviced. 8085 microprocessor has few maskable and non-maskable Interrupts.

There are four hardware interrupts in 8085:

- TRAP
- RST 7.5
- RST6.5
- RST5.5

Interrupts are generally used to stop the normal execution sequence of the instructions by the CPU and address a higher priority task first. This usually happens when the peripheral device want to transmit data to either to memory or CPU. The device which seeks CPU attention sends an interrupt signal INTR to CPU. The CPU holds the operation which it was

performing, save the intermediate data and the registers value so that it could resume the task later. The CPU sends the interrupt acknowledgment INTA to the device, which is a signal that the CPU is now ready to serve the request of the device which initiated the interrupt and the vectored address, the address of the Interrupt Service Routine to handle the interrupts is stored in the Program Counter(PC). After this the CPU executes the *Interrupt Service Routine(ISR)*, which is a small program/routine to service the corresponding interrupting source. The interrupts are of two categories, maskable and non-maskable.

- a. *Maskable interrupts*: these are the class of interrupts which a CPU can ignore if it is servicing an important task. TRAP is an example of a non-maskable interrupt.
- b. *Non-maskable interrupts*: these are the class of interrupts which the CPU cannot afford to ignore and has to be serviced immediately, come what may. Typically this category of interrupts is used for critical condition. RST 7.5, RST 6.5 and RST 5.5 are the examples of maskable interrupt.

The priority order of the interrupts is as follows:

TRAP > RST 7.5 > RST 6.5 > RST5.5 > INTR

---

## 15.3 INSTRUCTION SET OF 8085 MICROPROCESSOR

---

A computer can perform all the operations for which it is designed i.e. perform operations that are defined in its instruction set. To perform a specific task, a program, which is a sequence of instructions, is written and through this sequence of instructions we instruct the computer to perform what operation is to be performed, on what data and in which sequence. The programmer can write a program in assembly language using these instructions. These instructions have been classified into the following groups:

- Data Transfer Group
- Arithmetic Group
- Logical Group
- Branch Control Group
- I/O and Machine Control Group

### 15.3.1 Data Transfer Group

This category of instructions are used to transfer the content of source register to another destination register and after the transfer of the data, the content of the source remains unaltered. It is similar to copy command, where the selected contents from the source are transferred to destination

without the affecting of content at the source. The example of this category instructions are MOV, MVI, LDA, LXI, STA, etc.

Example #1: MOV R1, R2

This instruction moves/copies the content of the source register R2 to Destination register R1.

MOV [Destination Register], [Source Register]

Suppose before the execution of the above instruction, the content of register R1 and R2 are 20 and 30 respectively.

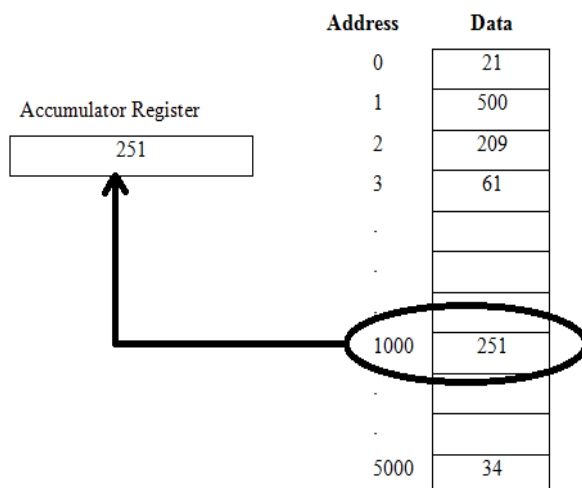


After the execution of the statement MOV R1, R2, the content of R1= 30 and R2=30 i.e. the content of the source register remains same even after the execution of the instruction. Only the content of destination register is altered.



Example #2: LDA 1000

This instruction load the content of the memory location, as specified in the instruction to the accumulator register and the after the execution of the instruction, the content of the memory location remains intact.



LDA [ Address of Memory Location]

In the above example, after the LDA 1000 instruction is executed, the content of location 1000 is transferred to Accumulator register and after the transfer; the content at memory location remains unchanged.

The list of instructions which falls in this category is:

- MOV r1, r2 (Move Data; Move the content of the one register to another).  
 $[r1] \leftarrow [r2]$
- MOV r, m (Move the content of memory register).  
 $r \leftarrow [M]$
- MOV M, r. (Move the content of register to memory).  
 $M \leftarrow [r]$
- MVI r, data. (Move immediate data to register).  
 $[r] \leftarrow \text{data.}$
- MVI M, data. (Move immediate data to memory).  
 $M \leftarrow \text{data.}$
- LXI rp, data 16. (Load register pair immediate).  
 $[rp] \leftarrow \text{data 16 bits, } [rh] \leftarrow \text{8 LSBs of data.}$
- LDA addr. (Load Accumulator direct).  
 $[A] \leftarrow [\text{addr}].$
- STA addr. (Store accumulator direct).  
 $[\text{addr}] \leftarrow [A].$
- LHLD addr. (Load H-L pair direct).  
 $[L] \leftarrow [\text{addr}], [H] \leftarrow [\text{addr}+1].$
- SHLD addr. (Store H-L pair direct).  
 $[\text{addr}] \leftarrow [L], [\text{addr}+1] \leftarrow [H].$
- LDAX rp. (LOAD accumulator indirect)  
 $[A] \leftarrow [[rp]]$
- STAX rp. (Store accumulator indirect)  
 $[[rp]] \leftarrow [A].$
- XCHG. (Exchange the contents of H-L with D-E pair)  
 $[H-L] \leftrightarrow [D-E].$

### 15.3.2 Arithmetic Group

All the arithmetic operations like addition, subtraction; increment or decrement comes under this category.

Example #1:

DAA (Decimal adjust accumulator)- The instruction DAA is used in the program after ADD, ADI, ACI, ADC, etc instructions. After the execution of ADD, ADC, etc instructions the result is in hexadecimal and it is placed in the accumulator. The DAA instruction operates on this result and gives the final result in the decimal system. It uses carry and auxiliary carry for decimal adjustment. 6 is added to 4 LSBs of the content of the accumulator if their value lies in between A and F or the AC flag is set to 1. Similarly, 6 is also added to 4 MSBs of the content of the accumulator if their value lies in between A and F or the CS flag is set to 1. All status flags are affected. When DAA is used data should be in decimal numbers.

The example of this category instructions are:

- ADD r. (Add register to accumulator)  
 $[A] \leftarrow [A] + [r].$
- ADD M. (Add memory to accumulator)  
 $[A] \leftarrow [A] + [[H-L]].$
- ADC r. (Add register with carry to accumulator).  
 $[A] \leftarrow [A] + [r] + [CS].$
- ADC M. (Add memory with carry to accumulator)  
 $[A] \leftarrow [A] + [[H-L]] [CS].$
- ADI data (Add immediate data to accumulator)  
 $[A] \leftarrow [A] + \text{data}.$
- ACI data (Add with carry immediate data to accumulator)  
 $[A] \leftarrow [A] + \text{data} + [CS].$
- DAD rp. (Add register pair to H-L pair)  
 $[H-L] \leftarrow [H-L] + [rp].$
- SUB r. (Subtract register from accumulator)  
 $[A] \leftarrow [A] - [r].$
- SUB M. (Subtract memory from accumulator)  
 $[A] \leftarrow [A] - [[H-L]].$
- SBB r. (Subtract register from accumulator with borrow)  
 $[A] \leftarrow [A] - [r] - [CS].$
- SBB M. (Subtract memory from accumulator with borrow)  
 $[A] \leftarrow [A] - [[H-L]] - [CS].$

- SUI data. (Subtract immediate data from accumulator)  
 $[A] \leftarrow [A] - \text{data}.$
- SBI data. (Subtract immediate data from accumulator with borrow).  
 $[A] \leftarrow [A] - \text{data} - [CS].$
- INR r (Increment register content)  
 $[r] \leftarrow [r] + 1.$
- INR M. (Increment memory content)  
 $[[H-L]] \leftarrow [[H-L]] + 1.$
- DCR r. (Decrement register content).  
 $[r] \leftarrow [r] - 1.$
- DCR M. (Decrement memory content)  
 $[[H-L]] \leftarrow [[H-L]] - 1.$
- INX rp. (Increment register pair)  
 $[rp] \leftarrow [rp] + 1.$
- DCX rp (Decrement register pair)  
 $[rp] \leftarrow [rp] - 1.$

### 15.3.3 Logical Group

All the instructions related to logical operations like AND, OR, compare, etc. in data are grouped under logic group instructions.

The example of this category instructions are:

- ANA r. (AND register with accumulator)  
 $[A] \leftarrow [A] \wedge [r].$
- ANA M. (AND memory with accumulator).  
 $[A] \leftarrow [A] \wedge [[H-L]].$
- ANI data. (AND immediate data with accumulator)  
 $[A] \leftarrow [A] \wedge \text{data}.$
- ORA r. (OR register with accumulator)  
 $[A] \leftarrow [A] \vee [r].$
- ORA M. (OR memory with accumulator)  
 $[A] \leftarrow [A] \vee [[H-L]]$
- ORI data. (OR immediate data with accumulator)  
 $[A] \leftarrow [A] \vee \text{data}.$



- XRA r. (EXCLUSIVE – OR register with accumulator)  
 $[A] \leftarrow [A] \vee [r]$
- XRA M. (EXCLUSIVE-OR memory with accumulator)  
 $[A] \leftarrow [A] \vee [[H-L]]$
- XRI data. (EXCLUSIVE-OR immediate data with accumulator)  
 $[A] \leftarrow [A]$
- CMA. (Complement the accumulator)  
 $[A] \leftarrow [A]$
- CMC. (Complement the carry status)  
 $[CS] \leftarrow [CS]$
- STC. (Set carry status)  
 $[CS] \leftarrow 1.$
- CMP r. (Compare register with accumulator)  
 $[A] - [r]$
- CMP M. (Compare memory with accumulator)  
 $[A] - [[H-L]]$
- CPI data. (Compare immediate data with accumulator)  
 $[A] - \text{data}.$

The 2nd byte of the instruction is data, and it is subtracted from the content of the accumulator. The status flags are set according to the result of subtraction. But the result is discarded. The content of the accumulator remains unchanged.

- RLC (Rotate accumulator left)  
 $[A_{n+1}] \leftarrow [A_n], [A_0] \leftarrow [A_7], [CS] \leftarrow [A_7].$

The content of the accumulator is rotated left by one bit. The seventh bit of the accumulator is moved to carry bit as well as to the zero bit of the accumulator. Only CS flag is affected.

- RRC. (Rotate accumulator right)  
 $[A_7] \leftarrow [A_0], [CS] \leftarrow [A_0], [A_n] \leftarrow [A_{n+1}].$

The content of the accumulator is rotated right by one bit. The zero bit of the accumulator is moved to the seventh bit as well as to carry bit. Only CS flag is affected.

- RAL. (Rotate accumulator left through carry)  
 $[A_{n+1}] \leftarrow [A_n], [CS] \leftarrow [A7], [A0] \leftarrow [CS]$ .
- RAR. (Rotate accumulator right through carry)  
 $[A_n] \leftarrow [A_{n+1}], [CS] \leftarrow [A0], [A7] \leftarrow [CS]$

### 15.3.4 Branch Control Group

Normally the order of execution of the instructions of the program is sequential. Often we encounter a situation in real world programming where we want the control of the program jump to some location as specified by the instruction. For which, a condition is tested. If the condition holds true, the instruction at the location specified in the instruction is executed, else the instruction in the next sequential location is executed. This is known as a conditional jump. At times, we encounter a situation where we want to execute an instruction which is not located at the next sequential location at any cost and we don't want any condition to hold true for that. This is an unconditional jump. And such situation arises when the CPU is executing a program and an urgent operating system related subroutine needs to be executed, or an interrupt occurs. Branch control group contains such instructions which are used for conditional and unconditional jump, subroutine call and return, and restart purposes.

This group includes the instructions. Examples are:

- JMP addr (label)- (Unconditional jump: jump to the instruction specified by the address).  
 $[PC] \leftarrow \text{Label}$ .
- Conditional Jump addr (label)- After the execution of the conditional jump instruction the program jumps to the instruction specified by the address (label) if the specified condition is fulfilled. The program proceeds further in the normal sequence if the specified condition is not fulfilled. If the condition is true and program jumps to the specified label, the execution of a conditional jump takes 3 machine cycles: 10 states. If condition is not true, only 2 machine cycles; 7 states are required for the execution of the instruction.
- JZ addr (label)- (Jump if the result is zero)
- JNZ addr (label)- (Jump if the result is not zero)
- JC addr (label)- (Jump if there is a carry)
- JNC addr (label)- (Jump if there is no carry)
- JP addr (label)- (Jump if the result is plus)
- JM addr (label)- (Jump if the result is minus)
- JPE addr (label)- (Jump if even parity)

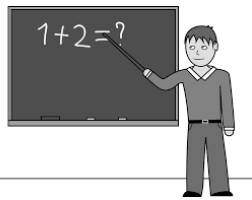
- JPO addr (label)- (Jump if odd parity)
- CALL addr (label)- (Unconditional CALL: call the subroutine identified by the operand)
- CALL instruction is used to call a subroutine- Before the control is transferred to the subroutine, the address of the next instruction of the main program is saved in the stack. The content of the stack pointer is decremented by two to indicate the new stack top. Then the program jumps to subroutine starting at address specified by the label.
- RET (Return from subroutine)
- RST n (Restart)- Restart is a one-word CALL instruction. The content of the program counter is saved in the stack. The program jumps to the instruction starting at restart location.

### 15.3.5 I/O and Machine Control Group

This group includes the instructions for input/output ports, stack and machine control.

The example of this category instructions are:

- IN port-address. (Input to accumulator from I/O port)
- $[A] \leftarrow [\text{Port}]$
- OUT port-address (Output from accumulator to I/O port)
- $[\text{Port}] \leftarrow [A]$
- PUSH rp (Push the content of register pair to stack)
- PUSH PSW (PUSH Processor Status Word)
- POP rp (Pop the content of register pair, which was saved, from the stack)
- POP PSW (Pop Processor Status Word)
- HLT (Halt)
- XTHL (Exchange stack-top with H-L)
- SPHL (Move the contents of H-L pair to stack pointer)
- EI (Enable Interrupts)
- DI (Disable Interrupts)
- SIM (Set Interrupt Masks)
- RIM (Read Interrupt Masks)
- NOP (No Operation)



## Check Your Progress

1. The ALU and CU were connected to each other and the combined unit was known as \_\_\_\_\_.
2. Depending on the \_\_\_\_\_ of the data bus of a microprocessor, it can be categorized into 8-bit, 16-bit, 32-bit or 64-bit microprocessor.
3. 8085 operates with \_\_\_\_\_ MHz clock.
4. ALE stands for \_\_\_\_\_ .
5. Whenever an Instruction from the memory is fetched, the instruction is placed inside a 8-bit register, known as \_\_\_\_\_.
6. The 8085 microprocessor has 8-bit data bus and \_\_\_\_\_ -bit address bus.

---

## 15.4 SUMMARY

---

1. A microprocessor( $\mu\text{p}$ ) is an electronic device that is used by the computer to its processing.
2. A microprocessor is a programmable device which can perform different sets of operations on the data it receives as an input depending on the sequence of instructions supplied in the given program.
3. The processing power of a microprocessor depends on its instruction set.
4. The width of the data bus signifies the number of data bits a microprocessor can process simultaneously.
5. Intel launched its first 8-bit microprocessor in 1972 and names it Intel 8008.
6. Intel 8985 is a 8-bit general purpose microprocessor capable of addressing 64K memory.
7. 8085 has 40 pins and runs on +5 V power supply.
8. In 8085, the 8-bit data bus was multiplexed with the lower part of 16-bit address bus so that the number of pins are limited to 40.
9. It has a 16-bit address bus, hence it is capable of addressing  $2^{16}=64$  KB memory.
10. The address bus has 8 signal lines A8 – A15 which are unidirectional.

11. There are 4 main control and status signals.
12. An Interrupt is a mechanism by which an I/O or an instruction can suspend the normal execution of processor and get itself serviced.

---

## **15.5 ANSWERS TO CHECK YOUR PROGRESS**

---

1. CPU
2. Width
3. 3
4. Address Latch Enable
5. Instruction Register(IR)
6. 16

---

## **15.6 Terminal Questions**

---

1. What is a microprocessor? What is a difference between a microprocessor and CPU?
2. Draw and explain the pin-diagram of 8085 microprocessor.
3. Explain the classification of the instructions of 8085 microprocessor.
4. What is conditional and unconditional jump.
5. What is an interrupt? What is the order of priority of these interrupts?
6. Explain the MOV and LDA instruction with the help of diagram.
7. Explain JMP instruction.
8. What is an interrupt?
9. What is the different between a maskable and non-maskable interrupt? What is the priority of various interrupts?

---

## REFERENCES

---

- (s.j.). Onttrek Dec. 15, 2015 uit <http://www.cs.umd.edu>
- (s.j.). Onttrek Dec. 15, 2015 uit [http://www.technicalsymposium.com/MICROPROCESSOR\\_Instruction\\_Set\\_of\\_Intel\\_8085.doc](http://www.technicalsymposium.com/MICROPROCESSOR_Instruction_Set_of_Intel_8085.doc)
- (s.j.). Onttrek Dec. 15, 2015 uit <https://en.wikipedia.org/wiki/Astable>
- (s.j.). Onttrek Dec. 15, 2015 uit [http://mcquestion.blogspot.in/2012/08/computer-system-architecture\\_6.html](http://mcquestion.blogspot.in/2012/08/computer-system-architecture_6.html)
- Anand, A. (2012, Nov.). *Memory Organization*. Onttrek Dec. 15, 2015 uit <http://www.slideshare.net/rashcommuz/memory-organization-16934580>
- Arivazhagan, S. S. (2012). *Digital Circuits and Design*. S.Chand & Company .
- AspenCore. (2016). *Bipolar Transistor*. Onttrek Oct. 29, 2017 uit [http://www.electronics-tutorials.ws/transistor/tran\\_1.html](http://www.electronics-tutorials.ws/transistor/tran_1.html)
- Astable Multivibrator*. (s.j.). Onttrek Dec. 15, 2015 uit [http://evalidate.in/lab1/pages/IC555/AstableMultivibrator/AstableMultivibrator\\_I.html](http://evalidate.in/lab1/pages/IC555/AstableMultivibrator/AstableMultivibrator_I.html)
- Basheer, N. (2011). *ZENER DIODE* . Onttrek Oct. 29, 2017 uit <https://mediatoget.blogspot.in/2011/10/zener-diode.html> available under Creative Commons Attribution 3.0 Unported License.
- Belurkar, V. (2012, Dec. 07). *Lithium-ion Battery*. Onttrek Dec. 15, 2015 uit <http://simpleelectronic-project.blogspot.com/>
- Burke, T. (2006). *Resistor Codes*. Onttrek Oct. 29, 2017 uit <https://commons.wikimedia.org/wiki/File:Resistor-Codes.svg> available under the Creative Commons Attribution-Share Alike 2.5 Generic, 2.0 Generic and 1.0 Generic license.
- Community, E. e. (2014). *Diode Characteristics*. Onttrek Oct. 29, 2017 uit <http://engineering.electrical-equipment.org/electrical-distribution/diode-characteristics.html> available under Creative Commons By Attribution License.
- Dutt, S. (2012). *An introduction to Microprocess Architecture using Intel 8085 as a classical processor*. Onttrek Dec. 15, 2015 uit Slideshare: <http://www.slideshare.net/sdutt36/8085-14257924>
- Elcap. (2012). *Ceramic disc capacitor*. Onttrek Oct. 29, 2017 uit [https://commons.wikimedia.org/wiki/File:Ceramic\\_disc\\_capacitor.png](https://commons.wikimedia.org/wiki/File:Ceramic_disc_capacitor.png) available under Creative Commons CC0 1.0 Universal Public Domain Dedication.
- ETHW. (2017). *Integrated Circuits*. Onttrek Oct. 30, 2017 uit [http://ethw.org/Integrated\\_Circuits?gclid=EAIaIQobChMIjaXJ36KX1wI](http://ethw.org/Integrated_Circuits?gclid=EAIaIQobChMIjaXJ36KX1wI)

V0hFoCh0IPwUAEAAyAAEgLj3PD\_BwE available under Creative Commons Attribution-ShareAlike License.

*Find a Tech Definition.* (s.j.). Onttrek Dec. 15, 2015 uit <http://whatis.techtarget.com/>

Gao, Y. (s.j.). *Review of Flip Flops.* Onttrek Dec. 15, 2015 uit <http://www.readbag.com/maxwell-ict-griffith-au-yg-teaching-dns-dns-module3-p1>

*Halfwave rectifier.* (2005). Onttrek Oct. 29, 2017 uit <https://simple.wikipedia.org/wiki/File:Halfwave.rectifier.en.png> available under Public Domain License.

Hamacher. (2011). *Computer Organization.* Tata McGraw Hill.

*Instruction Seti of Intel 8085.* (s.j.). Onttrek Dec. 15, 2015 uit <http://www.daenotes.com/electronics/digital-electronics/instruction-set-intel-8085>

Learning, L. (2012). *The Central Processing Unit.* Onttrek Oct. 30, 2017 uit Introduction to Computer Applications and Concepts: <https://courses.lumenlearning.com/zeliite115/chapter/reading-the-central-processing-unit/> available under Creativr Commons Attribution-ShareAlike License.

LibreTexts. (2016). *Bipolar Junction Transistor.* Onttrek Oct. 29, 2017 uit [https://eng.libretexts.org/Core/Materials\\_Science/Materials\\_and\\_Devices/Bipolar\\_Junction\\_Transistor](https://eng.libretexts.org/Core/Materials_Science/Materials_and_Devices/Bipolar_Junction_Transistor) available under Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License.

*LOGIC CIRCUIT AND SWITCHING THEORY.* (2010, Feb. 05). Onttrek Dec. 15, 2015 uit Sequential Logic Basics: <http://logicckt.blogspot.in/2010/02/week-5.html>

Mano, M. M. (2008). *Computer System Architecture.* Pearson.

Mano, M. M. (2008). *Digital Logic and Computer Design.* Pearson.

*Multivibrator.* (2014, Jan. 11). Onttrek Dec. 15, 2015 uit Slideshare: <http://www.slideshare.net/nakulrtm/multivibrators-including-monostable-astable-and-bistable> Omegatron. (2006). *Diode symbol.* Onttrek Oct. 29, uitWikibooks:

[https://commons.wikimedia.org/wiki/File:Diode\\_symbol.svg](https://commons.wikimedia.org/wiki/File:Diode_symbol.svg) available under Creative Commons Attribution-Share Alike 3.0 Unported, 2.5 Generic, 2.0 Generic and 1.0 Generic license.

Omegatron. (2015). *Zener Diode Figure.* Onttrek Oct. 29, 2017 uit [https://commons.wikimedia.org/wiki/File:Zener\\_diode\\_symbol.svg](https://commons.wikimedia.org/wiki/File:Zener_diode_symbol.svg) available under Creative Commons Attribution-Share Alike 3.0 Unported, 2.5 Generic, 2.0 Generic and 1.0 Generic license.

Peripitus. (2007). *Toroidal inductor.* Onttrek Oct. 29, 2017 uit [https://commons.wikimedia.org/wiki/File:Toroidal\\_inductor.jpg](https://commons.wikimedia.org/wiki/File:Toroidal_inductor.jpg) available

under Creative Commons Attribution-Share Alike 4.0 International, 3.0 Unported, 2.5 Generic, 2.0 Generic and 1.0 Generic license.

Powley, R. (2011). *Introduction to Computers*. Onttrek Oct. 30, 2017 uit [http://doer.col.org/bitstream/123456789/8192/1/2011\\_VUSSC\\_Intro-Computers.pdf](http://doer.col.org/bitstream/123456789/8192/1/2011_VUSSC_Intro-Computers.pdf) available under CC-BY-SA license.

Storr, W. (2015, Dec.). *Basic Electronics Tutorials*. Onttrek Dec. 10, 2015 uit <http://www.electronics-tutorials.ws/>

Storr, W. (2015, Dec. 13). *Multivibrators*. Onttrek Dec. 15, 2015 uit Electronics Tutorials: [http://www.electronics-tutorials.ws/sequential/seq\\_3.html](http://www.electronics-tutorials.ws/sequential/seq_3.html)

TES. (2017). *Explain the machine instruction cycle*. Onttrek Oct. 31, 2017 uit <https://compsci2014.wikispaces.com/2.1.4+Explain+the+machine+instruction+cycle> available under Creative Commons Attribution License.

*Types of capacitor*. (2014). Onttrek Oct. 29, 2017 uit [https://commons.wikimedia.org/wiki/File:Types\\_of\\_capacitor.svg](https://commons.wikimedia.org/wiki/File:Types_of_capacitor.svg) available under Creative Commons CC0 1.0 Universal Public Domain Dedication.

*What is a computer?* (2017). Onttrek Oct. 30, 2017 uit [https://en.wikiversity.org/wiki/What\\_is\\_a\\_computer%3F](https://en.wikiversity.org/wiki/What_is_a_computer%3F) available under Creative Commons Attribution-ShareAlike License.

Wikibooks. (2017, Aug. 16). *Electronics Handbook/Components/Diodes/Photo*. Onttrek Oct. 28, 2017 uit [https://en.wikibooks.org/wiki/Electronics\\_Handbook/Components/Diodes/Photo](https://en.wikibooks.org/wiki/Electronics_Handbook/Components/Diodes/Photo) available under the Creative Commons Attribution-ShareAlike License.

Wikibooks. (2017). *Transistor Basics*. Onttrek Oct. 29, 2017 uit [https://en.wikibooks.org/wiki/Digital\\_Circuits/Transistor\\_Basics](https://en.wikibooks.org/wiki/Digital_Circuits/Transistor_Basics) available under Creative Commons Attribution-ShareAlike License. Wikipedia. (2017). *Diode*. Onttrek Oct. 29, 2017 uit [http://sciencewise.info/resource/Diode/Diode\\_by\\_Wikipedia](http://sciencewise.info/resource/Diode/Diode_by_Wikipedia) available under Creative Commons Attribution-ShareAlike License.

Wikipedia. (2017). *Optical isolator*. Onttrek Oct. 29, 2017 uit [https://en.wikipedia.org/wiki/Optical\\_isolator](https://en.wikipedia.org/wiki/Optical_isolator) available under Creative Commons Attribution-ShareAlike License.

Wikispaces. (s.j.). *Computer Architecture*. Onttrek Oct. 31, 2017 uit <https://isscs.wikispaces.com/3.2+-+Computer+Architecture> available under Creative Commons Attribution license.

Wikiversity. (2017). *Field-Effect Transistors*. Onttrek Oct. 29, 2017 uit [https://en.wikiversity.org/wiki/Fundamental\\_Physics/Electronics/Field-Effect\\_Transistors](https://en.wikiversity.org/wiki/Fundamental_Physics/Electronics/Field-Effect_Transistors) available under Creative Commons Attribution-ShareAlike License.



Yewale, J. (2011). *Diode Clamping Circuit*. Onttrek Oct. 29, 2017 uit <http://todayscircuits.blogspot.com/2011/06/diode-clamping-circuits.html> available under Creative Commons Attribution-ShareAlike 2.5 India License.

Yewale, J. (2011). *Diode Clippers – A study of various Clipping Circuits*. Onttrek Oct. 29, 2017 uit <http://todayscircuits.blogspot.com/2011/06/diode-clippers-overview-of-clipping.html> available under Creative Commons Attribution-ShareAlike 2.5 India License.

*Zener Diode*. (2009). Onttrek Oct. 29, 2017 uit [https://en.wikibooks.org/wiki/Semiconductors/Zener\\_Diode](https://en.wikibooks.org/wiki/Semiconductors/Zener_Diode) available under Creative Commons Attribution-ShareAlike License.

(n.d.). Retrieved Dec. 15, 2015, from <http://www.cs.umd.edu>

(n.d.). Retrieved Dec. 15, 2015, from [http://www.technicalsymposium.com/MICROPROCESSOR\\_Instruction\\_Set\\_of\\_Intel\\_8085.doc](http://www.technicalsymposium.com/MICROPROCESSOR_Instruction_Set_of_Intel_8085.doc)

(n.d.). Retrieved Dec. 15, 2015, from <https://en.wikipedia.org/wiki/Astable>

(n.d.). Retrieved Dec. 15, 2015, from [http://mcqquestion.blogspot.in/2012/08/computer-system-architecture\\_6.html](http://mcqquestion.blogspot.in/2012/08/computer-system-architecture_6.html)

Anand, A. (2012, Nov.). *Memory Organization*. Retrieved Dec. 15, 2015, from <http://www.slideshare.net/rashcommuz/memory-organization-16934580>

Arivazhagan, S. S. (2012). *Digital Circuits and Design*. S.Chand & Company .

*Astable Multivibrator*. (n.d.). Retrieved Dec. 15, 2015, from [http://evaldate.in/lab1/pages/IC555/AstableMultivibrator/AstableMultivibrator\\_I.html](http://evaldate.in/lab1/pages/IC555/AstableMultivibrator/AstableMultivibrator_I.html)

Belurkar, V. (2012, Dec. 07). *Lithium-ion Battery*. Retrieved Dec. 15, 2015, from <http://simpleelectronic-project.blogspot.com/>

Dutt, S. (2012). *An introduction to Microprocess Architecture using Intel 8085 as a classical processor*. Retrieved Dec. 15, 2015, from Slideshare: <http://www.slideshare.net/sdutt36/8085-14257924>

*Find a Tech Definition*. (n.d.). Retrieved Dec. 15, 2015, from <http://whatis.techtarget.com/>

Gao, Y. (n.d.). *Review of Flip Flops*. Retrieved Dec. 15, 2015, from <http://www.readbag.com/maxwell-ict-griffith-au-yg-teaching-dns-dns-module3-p1>

Hamacher. (2011). *Computer Organization*. Tata McGraw Hill.

*Instruction Seti of Intel 8085.* (n.d.). Retrieved Dec. 15, 2015, from <http://www.daenotes.com/electronics/digital-electronics/instruction-set-intel-8085>

*LOGIC CIRCUIT AND SWITCHING THEORY.* (2010, Feb. 05). Retrieved Dec. 15, 2015, from Sequential Logic Basics: <http://logicckt.blogspot.in/2010/02/week-5.html>

Mano, M. M. (2008). *Computer System Architecture*. Pearson.

Mano, M. M. (2008). *Digital Logic and Computer Design*. Pearson.

*Multivibrator.* (2014, Jan. 11). Retrieved Dec. 15, 2015, from Slideshare: <http://www.slideshare.net/nakulrtm/multivibrators-including-monostable-astable-and-bistable>

Storr, W. (2015, Dec.). *Basic Electronics Tutorials*. Retrieved Dec. 10, 2015, from <http://www.electronics-tutorials.ws/>

Storr, W. (2015, Dec. 13). *Multivibrators*. Retrieved Dec. 15, 2015, from Electronics Tutorials: [http://www.electronics-tutorials.ws/sequential/seq\\_3.html](http://www.electronics-tutorials.ws/sequential/seq_3.html)

Wikibooks. (2017, Aug. 16). *Electronics Handbook/Components/Diodes/Photo*. Retrieved Oct. 28, 2017, from [https://en.wikibooks.org/wiki/Electronics\\_Handbook/Components/Diodes/Photo](https://en.wikibooks.org/wiki/Electronics_Handbook/Components/Diodes/Photo) available under the Creative Commons Attribution-ShareAlike License.