

Strictly According to the Revised Syllabus of
VTU - 2006 Course

Microprocessor

[06EC62] Semester - VI (EC)

Atul P. Godse

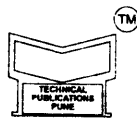
M. S. Software Systems (BITS Pilani)
B.E. Industrial Electronics
Formerly Lecturer in Department of Electronics Engg.
Vishwakarma Institute of Technology
Pune

Mrs. Deepali A. Godse

B.E. Industrial Electronics, M. E. (Computer)
Assistant Professor in Bharati Vidyapeeth's
Women's College of Engineering
Pune

Price Rs. 380/-

Visit us at : www.vtubooks.com



Technical Publications Pune™

Syllabus (Microprocessor)

PART - A

Unit - 1 (Chapters - 1, 2)

The 8086/8088 Processors : Historical background, The microprocessor based personal computer system, 8086 CPU architecture, Machine language instructions, Instruction execution timing, The 8088.

Unit - 2 (Chapter - 3)

Instruction Set of 8086/8088 : Assembler instruction format, Data transfer and arithmetic, Branch type, Loop, NOP and HALT, Flag manipulation, Logical and shift and rotate instructions. Illustration of these instructions with example programs, Directives and operators.

Unit - 3 (Chapters - 3, 4, 5)

Byte and String Manipulation : String instructions, REP Prefix, Table translation, Number format conversions, Procedures, Macros, Programming using keyboard and video display.

Unit - 4 (Chapter - 6)

8086 Interrupts : 8086 Interrupts and interrupt responses, Hardware interrupt applications, Software interrupt applications, Interrupt examples.

PART - B

Unit - 5 (Chapters - 8, 9, 10)

8086 Interfacing : Interfacing microprocessor to keyboard (keyboard types, keyboard circuit connections and interfacing, Software keyboard interfacing, Keyboard interfacing with hardware), Interfacing to alphanumeric displays (interfacing LED displays to microcomputer), Interfacing a microcomputer to a stepper motor.

Unit - 6 (Chapter - 11)

8086/8088 Based Multiprocessing Systems : Coprocessor configurations, The 8087 numeric data processor : Data types, Processor architecture, Instruction set and examples.

Unit - 7 (Chapters - 7, 12)

System Bus Structure : Basic 8086/8088 configurations : Minimum mode, Maximum mode, Bus interface : Peripheral component interconnect (PCI) bus, The parallel printer interface (LTP), The universal serial bus (USB).

Unit - 8 (Chapters - 13)

80386, 80486 and Pentium Processors : Introduction to the 80386 microprocessor, Special 80386 registers, Introduction to the 80486 microprocessor, Introduction to the Pentium microprocessor.

Table of Contents :

Chapter - 1	Introduction to Microprocessor Based Computer System	(1 - 1) to (1 - 44)
Chapter - 2	8086 / 8088 CPU	(2 - 1) to (2 - 16)
Chapter - 3	Instruction Set of 8086/8088 and ALP	(3 - 1) to (3 - 110)
Chapter - 4	BIOS and DOS Interrupts	(4 - 1) to (4 - 40)
Chapter - 5	Assembly Language Programs	(5 - 1) to (5 - 74)
Chapter - 6	8086 Interrupts	(6 - 1) to (6 - 28)
Chapter - 7	8086/8088 Configurations	(7 - 1) to (7 - 28)
Chapter - 8	Memory and I/O Interfacing	(8 - 1) to (8 - 30)
Chapter - 9	Programmable Peripheral Interface 8255	(9 - 1) to (9 - 40)
Chapter - 10	Keyboard and Display Interfacing	(10 - 1) to (10 - 48)
Chapter - 11	8086/8088 Based Multiprocessing Systems	(11 - 1) to (11 - 32)
Chapter - 12	Bus Interface	(12 - 1) to (12 - 42)
Chapter - 13	The 80386, 80486 and Pentium Processor	(13 - 1) to (13 - 118)
Appendix - A	Instruction Formats for 8086	(A - 1) to (A - 6)
Appendix - B	Instruction Set Summary	(B - 1) to (B - 8)
Appendix - C	Debugger	(C - 1) to (C - 8)
Microprocessors Lab 8086 Software Experiments		(L - 1) to (L - 52)
Chapterwise University Questions with Answer		(P - 1) to (P - 46)

Features of Book

- * Use of clear, plain and lucid language making the understanding very easy.
- * Approach of the book resembles class room teaching.
- * Book provides detailed insight into the subject.
- * Large number of illustrative diagrams to support the concepts.
- * Large number of programming examples.

Best of Technical Publications

As per Revised Syllabus of VTU-2006 Course
Semester-VI (EC)

 Digital Communication	Chitode
 Microprocessor	Godse
 Antennas and Propagation	Bakshi
 Information Theory and Coding	Chitode

Microprocessor

Atul P. Godse

M. S. Software Systems (BITS Pilani)
B.E. Industrial Electronics
Formerly Lecturer in Department of Electronics Engg.
Vishwakarma Institute of Technology
Pune

Mrs. Deepali A. Godse

B.E. Industrial Electronics, M. E. (Computer)
Assistant Professor in Bharati Vidyapeeth's
Women's College of Engineering
Pune

Visit us at : www.vtubooks.com

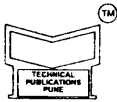


Technical Publications Pune[™]

S I T Library
Valachil, Mangalore



Accn No: 014688



Srinivas Institute of Technology

Accn No: 14688

Accn No: 14688

Microprocessor

ISBN 9788184315936

All rights reserved with Technical Publications. No part of this book should be reproduced in any form, Electronic, Mechanical, Photocopy or any information storage and retrieval system without prior permission in writing, from Technical Publications, Pune.

Published by :

Technical Publications Pune[®]

#1, Amit Residency, 412, Shaniwar Peth, Pune - 411 030, M.S., India.

Printer :

Alert DTPrinters
Sr.no. 10/3, Sinhadag Road,
Pune - 411 041

Preface

The importance of **Microprocessor** is well known in various engineering fields. Overwhelming response to our books on various subjects inspired us to write this book. The book is structured to cover the key aspects of the subject **Microprocessor**.

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All the chapters in the book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of the subject.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

We wish to express our profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by our whole family. We wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

Authors

A. P. Godse
D. A. Godse

Dedicated to God

Table of Contents

Chapter - 1 Introduction to Microprocessor Based Computer System (1 - 1) to (1 - 44)

1.1 Historical Background	1 - 1
1.1.1 The Mechanical Age	1 - 1
1.1.2 The Electrical Age	1 - 2
1.1.3 The Electronic Age	1 - 3
1.1.4 Programming Advancements	1 - 4
1.1.5 The Microprocessor Age	1 - 5
1.2 Microprocessor Basics	1 - 8
1.2.1 Simple Model of Microprocessor	1 - 11
1.2.1.1 Counter.	1 - 12
1.2.1.2 Decoder A	1 - 12
1.2.1.3 Register Array	1 - 12
1.2.1.4 Common Bus	1 - 13
1.2.1.5 Register I	1 - 13
1.2.1.6 Decoder B	1 - 13
1.2.1.7 Control Unit	1 - 13
1.2.1.8 Switch Control Circuit	1 - 14
1.2.1.9 ALU	1 - 15
1.2.1.10 Operation	1 - 15
1.2.2 Terminologies used in Microprocessor	1 - 22
1.2.3 Different Phases in the Execution Process	1 - 24
1.2.3.1 Fetch	1 - 24
1.2.3.2 Decode	1 - 24
1.2.3.3 Execute	1 - 24
1.2.4 Microprocessor and Programmer's Model	1 - 25

1.3 Microprocessor Based Personal Computer System	1 - 32
1.3.1 Memory System	1 - 33
1.3.1.1 Conventional (Base) Memory (Transient Program Area)	1 - 34
1.3.1.2 Upper Memory Area (UMA)	1 - 35
1.3.1.3 Extended Memory	1 - 36
1.3.1.4 High Memory Area (HMA)	1 - 37
1.3.1.5 Expanded Memory	1 - 38
1.3.2 I/O System	1 - 39
1.3.3 Microprocessor	1 - 39
1.3.4 Operating System	1 - 42
Review Questions	1 - 42

Chapter-2 8086 / 8088 CPU (2 - 1) to (2 - 16)

2.1 Features of 8086	2 - 1
2.2 Architecture of 8086	2 - 2
2.2.1 Bus Interface Unit [BIU]	2 - 2
2.2.2 Execution Unit [EU]	2 - 4
2.3 Register Organization	2 - 5
2.3.1 General Purpose Registers	2 - 5
2.3.2 Segment Registers	2 - 5
2.3.3 Pointers and Index Registers	2 - 7
2.3.4 Flag Register	2 - 7
2.4 Bus Operation	2 - 9
2.5 Memory Segmentation	2 - 9
2.6 The Processor 8088	2 - 14
Review Questions	2 - 16

Chapter - 3 Instruction Set of 8086/8088 and ALP (3 - 1) to (3 - 110)

3.1 Introduction	3 - 1
3.2 Addressing Modes	3 - 1
3.2.1 Data Addressing Modes	3 - 1
3.2.2 Program Memory Addressing Modes	3 - 9

3.2.3 Stack Memory Addressing Modes	3 - 11
3.3 Instruction Set of 8086/8088	3 - 14
3.4 Data Movement Instructions	3 - 15
3.4.1 MOV Instruction	3 - 15
3.4.2 PUSH/POP Instructions	3 - 16
3.4.3 Load Effective Address	3 - 18
3.4.4 String Data Transfer Instructions	3 - 19
3.4.5 Miscellaneous Data Transfer Instructions	3 - 21
3.5 Arithmetic and Logic Instructions	3 - 23
3.5.1 Addition	3 - 23
3.5.2 Subtraction	3 - 25
3.5.3 Comparison	3 - 26
3.5.4 Multiplication	3 - 27
3.5.5 Division	3 - 28
3.5.6 BCD and ASCII Arithmetic	3 - 28
3.5.6.1 BCD Arithmetic	3 - 29
3.5.6.2 ASCII Arithmetic	3 - 30
3.5.7 Basic Logic Instructions	3 - 32
3.5.8 Shift and Rotate	3 - 36
3.5.8.1 Shift	3 - 36
3.5.8.2 Rotate	3 - 39
3.6 String Instructions	3 - 42
3.7 Program Control Transfer Instructions	3 - 44
3.7.1 CALL and RET Instructions	3 - 44
3.7.2 JMP Instruction	3 - 46
3.7.3 Cond - Conditional Jump	3 - 48
3.8 Iteration Control Instructions	3 - 49
3.9 Processor Control Instructions	3 - 49
3.10 External Hardware Synchronization Instructions	3 - 50
3.11 Interrupt Instructions	3 - 51
3.12 Sign Extension Instructions	3 - 52

3.13 Assembler Directives	3 - 52
3.13.1 Summary of Assembler Directives	3 - 58
3.13.2 Variables, Suffix and Operators	3 - 59
3.13.3 Accessing a Procedure and Data from another Assembly Module	3 - 60
3.14 Assembly Language Programming	3 - 60
3.14.1 Assembly Language Programs	3 - 62
3.14.2 Assembly Language Programming Tips	3 - 64
3.14.3 Programming with an Assembler	3 - 66
3.14.3.1 Assembling Process	3 - 67
3.14.3.2 Linking Process	3 - 68
3.14.3.3 Debugging Process	3 - 68
3.15 Assembly Language Example Programs	3 - 70
3.16 Timings and Delays	3 - 73
3.16.1 Timer Delay using NOP Instruction	3 - 73
3.16.2 Timer Delay using Counters	3 - 73
3.16.3 Timer Delay using Nested Loops	3 - 75
3.17 Data Conversions	3 - 76
3.17.1 Routines to Convert Binary to ASCII	3 - 77
3.17.1.1 By AAM Instruction (For Number Less than 100)	3 - 77
3.17.1.2 By Series of Decimal Division	3 - 80
3.17.2 Routine to Convert ASCII to Binary	3 - 83
3.17.3 Routine to Read Hexadecimal Data	3 - 86
3.17.4 Routine to Display Hexadecimal Data	3 - 91
3.17.5 Lookup Tables for Data Conversions	3 - 94
3.18 Procedures	3 - 97
3.18.1 Reentrant Procedure	3 - 99
3.18.2 Recursive Procedure	3 - 99
3.19 Macro	3 - 100
3.20 Instruction Formats	3 - 101
Review Questions	3 - 108

4.1 ROM-BIOS (Basic Input/Output System).....	4 - 1
4.2 Disk Operating System (DOS).....	4 - 2
4.2.1 Intervals of DOS	4 - 4
4.2.2 Loading of DOS.....	4 - 5
4.3 Executable Files.....	4 - 7
4.3.1 Introduction to .COM Programs	4 - 8
4.3.2 Introduction to .EXE Program	4 - 9
4.3.3 Comparison between .EXE and .COM Programs	4 - 11
4.3.4 Programmer's Template	4 - 11
4.3.5 Conversion of .ASM to .EXE and .EXE to .COM.....	4 - 12
4.3.6 EXEC Function	4 - 12
4.3.7 Ending Program Execution	4 - 13
4.4 PSF (Structure Details).....	4 - 14
4.5 DOS and BIOS Calls.....	4 - 16
4.5.1 Character Input Functions	4 - 16
4.5.2 Character Display Functions	4 - 19
4.5.3 File Control Block Functions	4 - 20
4.5.4 Handle Functions	4 - 26
4.5.5 Memory Management Functions.....	4 - 32
4.5.6 Display Functions Provided by ROM BIOS.....	4 - 34
4.5.7 Printer Functions	4 - 38
Review Questions	4 - 39

6.1 Introduction	6 - 1
6.2 Interrupt Cycle of 8086/88.....	6 - 2
6.2.1 External Signal (Hardware Interrupt)	6 - 2
6.2.2 Special Instruction	6 - 2
6.2.3 Condition Produced by Instruction	6 - 2

6.3 8086 Interrupt Types.....	6 - 4
6.3.1 Divide by Zero Interrupt (Type 0)	6 - 4
6.3.2 Single Step Interrupt (Type 1)	6 - 4
6.3.3 Non Maskable Interrupt (Type 2)	6 - 5
6.3.4 Breakpoint Interrupt (Type 3)	6 - 5
6.3.5 Overflow Interrupt (Type 4)	6 - 5
6.3.6 Software Interrupts	6 - 6
6.3.7 Maskable Interrupt (INTR)	6 - 7
6.4 Interrupt Priorities.....	6 - 8
6.5 Expanding Interrupt Structure using PIC 8259	6 - 8
6.5.1 Features of 8259	6 - 9
6.5.2 Block Diagram of 8259A	6 - 9
6.5.3 Interrupt Sequence	6 - 11
6.5.4 Priority Modes and Other Features	6 - 12
6.5.5 Programming the 8259A	6 - 15
6.5.6 8259A Interfacing	6 - 22
6.6 Interrupt Example.....	6 - 26
Review Questions	6 - 28

Chapter - 7 8086/8088 Configurations (7 - 1) to (7 - 28)

7.1 Introduction	7 - 1
7.2 Signal Description of 8086	7 - 1
7.2.1 Signals with Common Functions in both Modes	7 - 2
7.2.2 Signal Definitions (24 to 31) for Minimum Mode	7 - 4
7.2.3 Signal Definitions (24 to 31) for Maximum Mode	7 - 4
7.3 Physical Memory Organisation	7 - 5
7.4 I/O Addressing Capability	7 - 7
7.5 General 8086 System Bus Structure and Operation.....	7 - 8
7.6 Minimum Mode 8086 System and Timings.....	7 - 10
7.6.1 Minimum Mode Configuration	7 - 10
7.6.2 Minimum Mode 8086 System	7 - 15
7.6.3 Bus Timings for Minimum Mode	7 - 16

7.6.3.1 Timings for Read and Write Operations	7 - 16
7.6.3.2 HOLD Response Sequence	7 - 18
7.7 Maximum Mode 8086 System and Timings	7 - 18
7.7.1 Maximum Mode Configuration	7 - 18
7.7.2 Maximum Mode 8086 System.	7 - 20
7.7.3 Bus Timings for Maximum Mode.	7 - 22
7.7.3.1 Timings for Read and Write Operations	7 - 22
7.7.3.2 Timings for $\overline{RQ}/\overline{GT}$ Signals	7 - 23
7.8 Minimum Mode and Maximum Mode 8088 Systems	7 - 24
7.9 Typical 8088 System Timing Diagram	7 - 26
Review Questions	7 - 27

Chapter - 8 Memory and I/O Interfacing (8 - 1) to (8 - 30)

8.1 Terminology and Operations	8 - 1
8.2 Memory Structure and its Requirements	8 - 2
8.3 Basic Concepts in Memory Interfacing	8 - 3
8.3.1 Address Decoding Techniques	8 - 4
8.3.2 Applications	8 - 6
8.3.3 Dynamic RAM Interfacing	8 - 13
8.3.3.1 DRAM Controller	8 - 14
8.4 Interfacing I/O Ports	8 - 17
8.4.1 I/O Interfacing Techniques	8 - 18
8.4.1.1 I/O Mapped I/O.	8 - 18
8.4.1.2 Memory Mapped I/O.	8 - 18
8.4.1.3 I/O Device Selection.	8 - 18
8.4.1.4 Interfacing 8-bit Input Port	8 - 19
8.4.1.5 Interfacing 16-bit Input Port	8 - 20
8.4.1.6 Interfacing 8-bit Output Device.	8 - 22
8.4.1.7 Interfacing 16-bit Output Device	8 - 23
8.4.1.8 I/O Interfacing with 16-bit Port Address	8 - 26
8.4.1.9 I/O Interfacing with Memory Mapped I/O	8 - 27
8.4.2 Comparison between Memory Mapped I/O and I/O Mapped I/O	8 - 28
Review Questions	8 - 28

9.1 Features of 8255A9 - 1

9.2 Pin Diagram9 - 2

9.3 Block Diagram.....9 - 4

 9.3.1 Data Bus Buffer. 9 - 4

 9.3.2 Control Logic. 9 - 5

 9.3.3 Group A and Group B Controls. 9 - 5

9.4 Operation Modes9 - 5

 9.4.1 Bit Set-Reset (BSR) Mode 9 - 5

 9.4.2 I/O Modes 9 - 5

9.5 Control Word Formats.....9 - 7

 9.5.1 For Bit Set/Reset Mode 9 - 7

 9.5.2 For I/O Mode 9 - 8

9.6 8255 Programming and Operation.....9 - 11

 9.6.1 Programming in Mode 0 9 - 11

 9.6.2 Programming in Mode 1 (Input / Output with Handshake) 9 - 13

9.7 Programming in Mode 2 (Strobes Bi-directional Bus I/O).....9 - 18

9.8 Interfacing 8255 to 8086 in I/O Mapped I/O Mode.....9 - 21

9.9 Interfacing 8255 to 8086 in Memory Mapped I/O9 - 22

9.10 Interfacing 8255 to 8088 in I/O Mapped I/O Mode.....9 - 23

9.11 Interfacing 8255 to 8088 in Memory Mapped I/O.....9 - 24

9.12 Centronics Parallel Printer Interface9 - 25

9.13 Stepper Motor Interfacing9 - 31

9.14 Control of High Power Devices using 82559 - 35

 9.14.1 Integrated Circuit Buffers. 9 - 35

 9.14.2 Transistor Buffers 9 - 35

 9.14.3 Isolation Circuits 9 - 37

 9.14.3.1 Electromagnetic Relays 9 - 38

 9.14.3.2 Solid State Relays 9 - 39

Review Questions9 - 39

10.1 Interfacing of Switches	10 - 1
10.1.1 Key Debounce using Hardware	10 - 2
10.1.2 Key Debouncing using Software	10 - 2
10.2 Simple Keyboard Interface	10 - 3
10.3 Matrix Keyboard Interface	10 - 4
10.4 Display Interfacing	10 - 9
10.4.1 LED Displays	10 - 10
10.4.2 Interfacing LED Displays	10 - 12
10.5 Keyboard and Display Interface using 8279	10 - 17
10.5.1 Features of 8279	10 - 17
10.5.2 Pin Description	10 - 18
10.5.2.1 CPU Interface Pins	10 - 19
10.5.2.2 Keyboard Data	10 - 20
10.5.2.3 Display Data	10 - 20
10.5.3 Block Diagram	10 - 20
10.5.3.1 CPU Interface and Control Section	10 - 21
10.5.3.2 Scan Section (Scan Counter)	10 - 22
10.5.3.3 Keyboard Section	10 - 23
10.5.3.4 Display Section	10 - 24
10.5.4 Operating Modes	10 - 24
10.5.4.1 Input Modes	10 - 24
10.5.4.2 Display Modes	10 - 27
10.5.5 8279 Commands	10 - 30
10.5.5.1 Keyboard/Display Mode Set Command (000)	10 - 30
10.5.5.2 Program Clock Command (001)	10 - 32
10.5.5.3 Read FIFO/Sensor RAM Command (010)	10 - 32
10.5.5.4 Read Display RAM Command (011)	10 - 33
10.5.5.5 Write Display RAM Command (100)	10 - 34
10.5.5.6 Display Write Inhibit/Blanking Command (101)	10 - 34
10.5.5.7 Clear Command (110)	10 - 35
10.5.5.8 End Interrupt/Error Mode Set Command (111)	10 - 36

10.5.6 Interfacing 8279 in I/O Mapped I/O	10 - 37
10.5.7 Interfacing 8279 in Memory Mapped I/O	10 - 38
10.5.8 Applications	10 - 39
Review Questions	10 - 48

Chapter - 11 8086/8088 Based Multiprocessing Systems (11 - 1) to (11 - 32)

11.1 Introduction	11 - 1
11.2 Closely Coupled System using 8086	11 - 2
11.3 Loosely Coupled System using 8086.....	11 - 4
11.4 The 8087 Numeric Data Processor.....	11 - 5
11.4.1 Features of 8087	11 - 5
11.4.2 Pin Diagram of 8087	11 - 5
11.4.3 Circuit Connection for 8087	11 - 7
11.4.4 Interaction between 8086 and 8087.....	11 - 8
11.4.5 The 8087 Architecture	11 - 9
11.4.5.1 Instruction Queue	11 - 10
11.4.5.2 Data Registers	11 - 10
11.4.5.3 Status Registers	11 - 10
11.4.5.4 Control Register	11 - 11
11.4.6 Data Formats and Conversions.....	11 - 12
11.4.7 Stacks in 8087	11 - 18
11.4.8 Instructions of 8087	11 - 19
11.4.8.1 Data Transfer Instructions	11 - 19
11.4.8.2 Arithmetic Instructions	11 - 21
11.9.8.3 Compare Instructions	11 - 24
11.4.8.4 Transcendental (Trigonometric and Exponential) Instructions	11 - 26
11.4.8.5 Instructions which Load Constants	11 - 26
11.4.8.6 Processor Control Instructions	11 - 27
11.4.9 Programming using 8087 Coprocessor	11 - 28
Review Questions	11 - 31

12.1 Introduction	12 - 1
12.2 The Peripheral Component Interconnect (PCI) Bus	12 - 1
12.2.1 Features	12 - 2
12.2.2 PCI Configurations	12 - 3
12.2.3 PCI Bus Signals	12 - 4
12.2.4 PCI Bus Commands	12 - 7
12.2.5 Data Transfer	12 - 8
12.2.6 PCI Arbitration	12 - 9
12.2.7 Configuration Space	12 - 11
12.3 Parallel Printer Interface(LPT)	12 - 18
12.3.1 Accepting 16-bit Input using Parallel Port	12 - 20
12.3.2 Interfacing Stepper Motor through Parallel Port	12 - 23
12.3.3 Bidirectional Operation of Parallel Port	12 - 24
12.3.4 Accepting 16-bit Input using Bidirectional Parallel Port	12 - 25
12.3.5 Interfacing 8-bit ADC using Parallel Port	12 - 26
12.4 Universal Serial Bus (USB).....	12 - 28
12.4.1 USB Features	12 - 28
12.4.2 USB System	12 - 30
12.4.3 Cables	12 - 30
12.4.4 USB Connector	12 - 31
12.4.5 USB Data	12 - 32
12.4.6 USB Commands	12 - 35
12.4.7 USB Host	12 - 36
12.4.8 USB Device	12 - 37
12.4.9 USB Descriptor	12 - 38
12.4.10 Device Controller	12 - 39
12.4.11 Functions	12 - 40
12.4.12 Enumeration	12 - 40
12.4.13 USB Hub	12 - 40
Review Questions	12 - 41

13.1 Introduction to 80386 Microprocessor	13 - 1
13.1.1 Features of 80386	13 - 2
13.1.2 Architecture of 80386DX	13 - 2
13.1.3 Special 80386 Registers	13 - 12
13.1.4 Summary of Various Registers in 80386	13 - 17
13.1.5 Data Types	13 - 18
13.1.6 Addressing Modes of 80386DX	13 - 20
13.1.7 New 80386 Instructions	13 - 25
13.1.8 Instruction Enhancements	13 - 28
13.2 Pin Description of 80386DX Microprocessor	13 - 29
13.2.1 Memory/IO Interface Signals	13 - 29
13.2.2 Interrupt Interface Signals	13 - 33
13.2.3 DMA Interface Signals	13 - 34
13.2.4 Coprocessor Interface Signals	13 - 34
13.3 Bus Interface	13 - 34
13.3.1 System Clock	13 - 35
13.3.2 Bus States	13 - 36
13.3.3 Dynamic Bus Sizing	13 - 36
13.3.4 Nonpipelined Bus Cycles	13 - 37
13.3.5 Pipelined Bus Cycle	13 - 37
13.3.6 Idle State in Pipelined Bus Cycle	13 - 38
13.3.7 READ and WRITE Bus Cycles	13 - 39
13.3.8 Nonpipelined Write Cycle	13 - 40
13.3.9 Pipelined Read/Write Cycle	13 - 42
13.3.10 Interrupt Acknowledge Cycle	13 - 43
13.3.11 HALT/Shutdown Cycle	13 - 44
13.3.12 Control Input BS16	13 - 45
13.3.13 Bus Lock	13 - 45
13.3.14 HOLD/HLDA	13 - 47

13.4 Memory Interface	13 - 47
13.4.1 Basic Memory Interface.	13 - 48
13.4.2 Memory Organization	13 - 50
13.4.3 Interfacing with Different Memory Types.....	13 - 53
13.4.4 Interleaved memory	13 - 60
13.5 I/O Interface	13 - 61
13.5.1 I/O Mapped I/O	13 - 62
13.5.2 Memory Mapped I/O	13 - 63
13.5.3 I/O Interface.....	13 - 64
13.6 Introduction to 80486 Processor	13 - 67
13.6.1 Features	13 - 67
13.6.2 Architecture of 80486	13 - 68
13.6.3 Programmable Model	13 - 70
13.6.4 Pin Description of 80486	13 - 70
13.6.5 New Instructions in 80486	13 - 74
13.6.6 80486 Memory System	13 - 74
13.6.7 Memory Management of 80486	13 - 77
13.7 Introduction to Pentium Processor	13 - 77
13.7.1 Pentium Architecture and Functional Description	13 - 79
13.7.2 Pin Description	13 - 82
13.7.3 The Memory System	13 - 91
13.7.4 Input/Output System	13 - 92
13.7.5 System Timings.....	13 - 92
13.7.6 New Pentium Instructions	13 - 95
13.7.7 Pentium RISC Features	13 - 98
13.7.8 Pentium Super-Scalar Architecture.....	13 - 102
13.7.9 Pipelining.....	13 - 103
13.7.10 Instruction Pairing Rules	13 - 104
13.7.11 Branch Prediction	13 - 106
13.7.12 The Instruction and Data Caches	13 - 108

13.7.12.1 Cache Memory	13 - 108
13.7.12.2 Two Level Cache System.	13 - 109
13.7.12.3 Pentium Cache Organisation	13 - 110
13.7.13 Floating Point Unit.	13 - 113
Review Questions	13 - 115

Appendix - A	Instruction Formats for 8086	(A - 1) to (A - 6)
---------------------	-------------------------------------	---------------------------

Appendix - B	Instruction Set Summary	(B - 1) to (B - 8)
---------------------	--------------------------------	---------------------------

Appendix - C	Debugger	(C - 1) to (C - 8)
---------------------	-----------------	---------------------------

List of Experiments

Programs Involving Data Transfer Instructions L - 1

1. **Program Statement** : Write an 8086 assembly language program to transfer a block of 256 bytes of data from offset 1000H in DS to offset 2000H in DS. L - 1
2. **Program Statement** : Write an 8086 assembly language program to transfer a block of 256 word of data from offset 000H in DS to offset 2000H in DS in the reverse order. So that byte at offset 1000H should be copied at offset 20FFH in DS L - 2
3. **Program Statement** : Write an 8086 assembly language program to interchange a block of 128 bytes of data from offset 1000H in DS and data from offset 2000H in DS. . . . L - 2
4. **Program Statement** : Write an 8086 assembly language program to transfer a block of 256 bytes of data from offset 1000H in DS to offset 1080H in DS. L - 2
5. **Program Statement** : Write 8086 ALP to perform non-overlapped and overlapped block transfer L - 3

Programs Involving Arithmetic and Logical Operations L - 3

1. **Program Statement** : Addition of two bytes L - 3
2. **Program Statement** : Average of two numbers. L - 3
3. **Program Statement** : Performs addition or subtraction L - 3
4. **Program Statement** : Addition of two 32-bit numbers L - 3
5. **Program Statement** : Program to evaluate the expression $W = X * (X + Y - Z)$. . . L - 4
6. **Program Statement** : Code conversion binary to BCD L - 5
7. **Program Statement** : Code conversion BCD to binary L - 5
8. **Program Statement** : Converting ASCII to binary L - 5
9. **Program Statement** : Converting binary to ASCII L - 5
10. **Program Statement** : Program to find LCM of two 16-bit unsigned numbers . . L - 5

- 11. **Program Statement** : Program to find HCF of two numbers. L - 5
- 12. **Program Statement** : Program to find LCM of two given numbers. L - 5
- 13. **Program Statement** : Program to calculate factorial of a number L - 5
- 14. **Program Statement** : Multiplication of two 8-bit numbers L - 5

Programs Involving Bit Manipulation Instructions like Checking **L - 5**

- 1. **Program Statement** : Separate even and odd numbers in the array : L - 5
- 2. **Program Statement** : Separate positive or negative numbers in the array. L - 5
- 3. **Program Statement** : Find logical 1's and 0's in a given data L - 5
- 4. **Program Statement** : Program to find whether given code is 2 out of 5 code or not. L - 6
- 5. **Program Statement** : Program to find whether given word is nibblewise palindrome or not. L - 6
- 6. **Program Statement** : Program to find whether given word is bitwise palindrome or not. L - 7
- 7. **Program Statement** : Write an ALP that will pack four 12-bit quantities in four consecutive words into three consecutive words. L - 8

Programs Involving Branch/Loop Instructions **L - 9**

- 1. **Program Statement** : Sum of array L - 9
- 2. **Program Statement** : Find maximum number in the array L - 9
- 3. **Program Statement** : Search a number in the array L - 9
- 4. **Program Statement** : Program to sort array by bubble sort method L - 9
- 5. **Program Statement** : Write a program to sort given 16-bit unsigned numbers in the ascending order by insertion sort method. L - 11
- 6. **Program Statement** : Sorting of array in the descending order by selection sort method. L - 12
- 7. **Program Statement** : Program to count the even and odd numbers from given list of numbers. L - 13
- 8. **Program Statement** : Write an 8086 assembly language program (ALP) to add array of N number stored in the memory L - 14

9. Program Statement : Write 8086 ALP to find and count negative numbers from the array of signed numbers stored in memory L - 15

10. Program Statement : Sort given array in the ascending or descending order as per the instruction given by the user L - 15

Programs on String Manipulations L - 15

1. Program Statement : Read and validate password L - 15

2. Program Statement : Reverse the words in string L - 15

3. Program Statement : Search numbers, alphabets, special characters in a given string L - 15

4. Program Statement : Program to find whether a string is palindrome or not. L - 15

5. Program Statement : Program to display given string in lower case L - 15

6. Program Statement : Program to search a given byte in the string. L - 15

7. Program Statement : Program to perform binary search. L - 15

8. Program Statement : Write 8086 ALP for the following operations on the string entered by the user. L - 16

9. Program Statement : Write 8086 ALP to perform string manipulation. The strings to be accepted from the user is to be stored in code segment Module_1 and write FAR PROCEDURES in code segment Module_2 for following operations on the string. L - 16

Programs to use DOS Interrupt INT21H Function Calls L - 17

1. Program Statement : Reading a character from keyboard L - 17

2. Program Statement : Buffered keyboard input L - 17

3. Program Statement : Display of character L - 17

4. Program Statement : Display of string L - 17

5. Program Statement : Program to find file size L - 17

6. Program Statement : Simulation of type command of DOS. L - 20

7. Program Statement : Simulation of copy command of DOS. L - 22

8. Program Statement : Program to convert HEX to BCD. L - 25

9. Program Statement : Program to convert BCD to HEX.	L - 27
10. Program Statement : Program to read system date	L - 32
11. Program Statement : Program to set system date.	L - 34
12. Program Statement : Program to read system time	L - 36
13. Program Statement : Program to set system time	L - 37

Interfacing Experiments **L - 39**

1. Program Statement : Interface 4 × 4 matrix keyboard to 8086 microprocessor using 8255	L - 39
2. Program Statement : Interface an 8-digit 7 segment LED display using 8255 to the 8086 microprocessor system and write an 8086 assembly language routine to display message on the display.	L - 39
3. Program Statement : Interface stepper motor to the 8086 microprocessor system and write an 8086 assembly language program to control the stepper motor	L - 40
4. Program Statement - Real Time Clock : Generate a real time clock by generating a periodic interrupt request signal on the NMI input of 8086.	L - 40
5. Program Statement : Design a pre-settable alarm system using 8253/54 timer. Use thumbwheel switches to accept 4 digit value in seconds. Alarm should last for 5 seconds. Do not use interrupt.	L - 40
6. Program Statement : DC Motor Speed and Direction Control	L - 44
7. Program Statement : Interfacing Printer.	L - 48
8. Program Statement : Serial data transfer from PC to PC through COM port.	L - 48
Program a) : PC to PC communication to send/receiver data between two PCs serial link is used. COM2 of one PC is connected to the COM2 of another PC.	L - 48
Program b) : Program to transfer one file from one PC to another PC	L - 49

Introduction to Microprocessor Based Computer System

1.1 Historical Background

1.1.1 The Mechanical Age

- Babylonians invented the **abacus** - the first mechanical calculator. The abacus used strings of beads to perform calculations.
- The abacus, which was used extensively and is still in use today, was improved in 1642 by mathematician Blaise Pascal. He invented a calculator that was constructed of gears and wheels.
- Each gear in this mechanical calculator contained 10 teeth that, when moved one complete revolution, advanced a second gear by one place. This is the basic principle used in all mechanical calculators and the automobile's odometer mechanism.
- The first practical geared mechanical machine used to compute information dates was designed in early 1800.
- In the 19th century Charles Babbage designed the first computing machinery to perform multistep operations automatically, that is, without a human intervening in every step.
- Babbage's first computing machine, which he called the **Difference Engine**, was intended to compute and print mathematical tables automatically. The difference engine performed only one arithmetic operation : addition.
- Babbage later designed an improved version (difference engine no. 2), which was to handle seventh-order polynomials and have 31 decimal digit of accuracy.
- Babbage then designed a much more powerful computing machine that he called the **Analytical Engine**. This machine is considered to be the first general purpose programmable computer ever designed.

- The Fig. 1.1 shows the basic structure of Babbage's analytical engine. The main components of analytical engine are a memory called the store and an ALU called the mill; the later was designed to perform the four basic arithmetic operations.

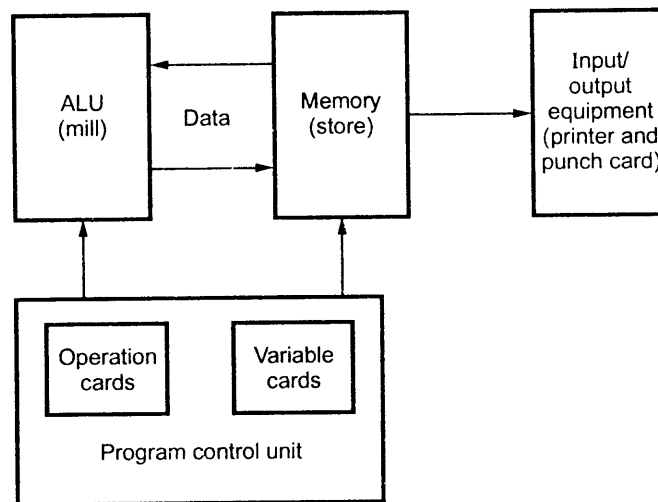


Fig. 1.1 Structure of Babbage's analytical engine

- Babbage proposed to use punch cards for input. He obtained the idea of using punch cards from Joseph Jacquard, a Frenchman who used punch cards as input to weaving machine he invented in 1801, which is today called **Jacquard's loom**.
- After many years of work, Babbage realized that the machinists of his day were unable to create the mechanical parts needed to complete his work.

1.1.2 The Electrical Age

- A later innovation was the use of electric motors (conceived by Michael Faraday) to drive the mechanical components, thus making calculators "electromechanical" and greatly increasing their speed.
- Blaise Pascal then developed electromechanical calculator. It was in well use until the early 1970, when the small hand-held electronic calculator, first introduced by Bomar, appeared.
- In 1860, Herman Hollerith developed the punch card tabulating machine. It was used to process the data collected in the 1880 united states census. The machine designed by Hollerith was driven by one of the new electric motors that counted, sorted and collected information stored on punch cards.
- In honour of Herman Hollerith, the 12-bit code used on a punch card is called **Hollerith code**.

The mechanical and electromechanical machines suffered from two serious drawbacks :

- Their computing speed was limited by the inertia of their moving parts, and
- The transmission of digital information by mechanical means was quite unreliable.

The further research in this field invented the electronic computer, in which the "moving parts" are electrons and they can be transmitted and processed reliably at speeds approaching that of light (300,000 km/s).

1.1.3 The Electronic Age

- Mechanical machines driven by electric motors continued to dominate the information processing world until the construction of the first electronic calculating machine in 1941 by a German inventor named Konrad Zuse.
- The first electronic computer, ENIAC (Electronic Numerical Integrator And Computer), was designed and constructed under the direction of Eckert and Mauchly at the Moore School of Engineering (University of Pennsylvania). It was made up of more than 18000 vacuum tubes and 1500 relays. ENIAC's primary function was to compute ballistic trajectories. It was able to perform nearly 5000 additions or subtractions per second.
- The ENIAC was a decimal rather than a binary machine. All numbers in an ENIAC were represented in decimal form and arithmetic was performed in the decimal system. Its data memory consists of 20 "accumulators", each capable of storing a ten digit decimal number. Each digit was represented by a ring of 10 vacuum tubes and only one vacuum tube was in the ON state to represent one of the ten digits. The major drawback of the ENIAC was that it was wired in for specific computations. For modifications and replacements of programs manually setting of switches and plugging and unplugging of cables was necessary. It was a very time consuming process. Despite these shortcomings, ENIAC was used for about ten years.
- The idea of having computer wired for general computations with program stored in memory was introduced by John Von Neumann when he was working as a consultant at the Moore school. He and originators of ENIAC designed the first *stored program computer* named EDVAC (Electronic Discrete Variable Computer). The stored program concept in EDVAC facilitated the users to enter and alter various programs and do variety of computations.
- The EDVAC project was further developed by Von Neumann with his collaborators at the Institute for Advanced Studies (IAS) in Princeton. They came up with a new machine referred to as IAS or Von Neumann machine. It has now become the usual frame of reference for many modern computers.

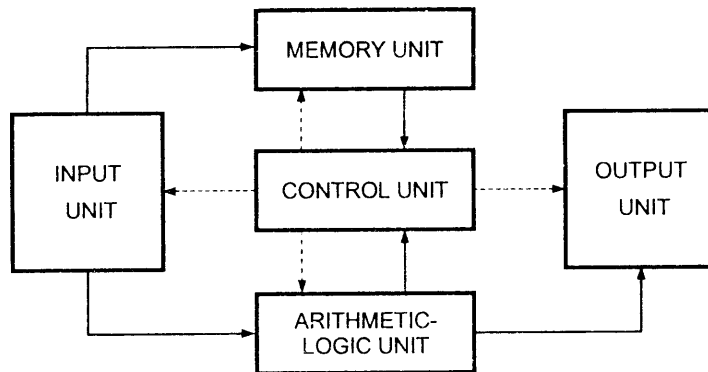


Fig. 1.2 A Von Neumann machine

Fig. 1.4 shows the general structure of a Von Neumann machine. It consists of five basic units whose functions can be summarized as follows :

- The input unit transmits data and instructions from the outside world to machine. It is operated by control unit.
- The memory unit stores both, data and instructions.
- The arithmetic-logic unit performs arithmetic and logical operations.
- The control unit fetches and interprets the instructions in memory and causes them to be executed.
- The output unit transmits final results and messages to the outside world.

1.1.4 Programming Advancements

- Mathematician John Von Neumann was the first person to develop a system that accepted instructions and stored them in memory.
- Instructions are nothing but a binary codes stored in the computer memory. A group of instructions arranged in a specific sequence are called **program**, and language used for program is called **machine language**.
- In the early 1950s, the computer systems such as the UNIVAC became available with assembly language programs.
- Assembly language is a language of mnemonics codes such as ADD for addition, SUB for subtraction and so on. In other words, we can say that mnemonic codes are the short hand english notations used to represent the instructions.
- The assembler (the program which converts assembly language program to machine language program) allowed programmers to use mnemonic codes for programming.

- In 1957, Grace Hopper developed the first high level programming language called FLOW MATIC. In the same year, IBM developed FORTRAN (FORMula TRANslator) for its computer systems.
- The FORTRAN language allowed programmers to develop programs with formulas to solve mathematical problems.
- A year after introduction of FORTRAN, another language called ALGOL (ALGORithmic Language) was introduced.
- The first truly successful and wide spread programming language for business applications was COBOL (COMputer Business Oriented Language).
- Another business language known as RPG (Report Program Generator) allows programming by specifying the form of the input, output and calculations.
- In early programming days more programming languages such as BASIC, C/C++, PASCAL and ADA was introduced.
- Recently, basic and C/C++ are available in their visual versions allowing programmers to write programs in the WINDOWS environment.
- The C/C++ language is found suitable for control programs. It also allows to include assembly language code to perform machine control functions more efficiently.
- The ADA language is mostly used by the department of defence.

1.1.5 The Microprocessor Age

- The world's first microprocessor, the Intel 4004, was a 4-bit microprocessor. (A bit is a binary digit with a value zero or one and 4-bit microprocessor means the microprocessor can process 4-bit word in one cycle. It has 12-bit address lines to access 4096 4-bit wide memory locations. The 4004 microprocessor has only 45 instructions.
- The Intel released the 4040, as updated version of 4004 with enhancement in speed, and without any improvement in word length and memory size.
- In 1971 announced the 8008, 8-bit and faster version of 4004. This version came up with expanded memory size upto 16 kbytes and additional instructions to make total of 48 instructions. (A byte is 8-bit binary number and a K is 1024).
- In 1974 Intel came out with 8080 was a considerable improvement over its predecessors. A six month later Motorola corporation announced its 8-bit processor MC6800. Then Zilog and so on.

The Table 1.1 shows the early 8-bit microprocessors.

Microprocessor number	Manufacturer
6502	MOS technology
8080	Intel
F-8	Fairchild
IMP-8	National semiconductor
PPS-8	Rockwell international
Z-8	Zilog

Table 1.1

- In 1977, Intel introduced updated version of 8080-8085.

The Table 1.2 shows the improvement of 8080 over 8008 and 8085 over 8080.

Parameter	Processor		
	8008	8080	8085
Speed	Requires 20 μ s for execution one instruction.	Requires 2 μ s for execution one instruction (10 times faster)	Requires 1.3 μ s for execution of one instruction.
Memory size	16 kbytes	64 kbytes (4 times more)	64 kbytes
TTL compatibility	Not directly compatible.	Compatible.	Compatible.
Interfacing	More costly and complex.	Easier and less expensive.	More easier and less expensive as it contains internal clock generator and internal system controller.

Table 1.2

- The interesting thing is Z-80 machine language code is compatible with 8085.
- The next generation was 8086 processor, a 16-bit processor, with advanced architecture and instruction set. At the same time Intel introduced processor 8088. The 8088 is an 8-bit version of the 8086 which has fewer data lines but retains all of the processing features of the 8086. The programs that run on 8088 will also run, without modification on the 8086. The 8086/88 pair were the first members of iAPX 86, family of microprocessors. This pair has 20 address lines to address upto 1 Mbyte of memory (1 Mbyte = 1024 kbyte) and also supported with 4 or 6 byte instruction queue to implement pipelining feature. (Feature of fetching the next instruction while the current instruction is executing is called pipelining). This pair belongs to CISC (Complex Instruction Set Computers) because of the number and complexity of instructions.

In 1983 the next version was announced, the 80186/88 very similar to 8086/8088 pair. The 80186/88 included many useful peripheral I/O functions as an integral part of the microprocessor. The improved instruction set of 80186/88 supports these peripheral I/O functions. Although the 80186 provided increased functionality, it maintained compatibility with the 8086, ensuring that it could execute 8086 programs.

After 80186/88, Intel has announced 80286, which is 16-bit processor like 8086. The 80286 was the first family member designed specifically for use as a CPU in a multi-user microcomputer. It contains many advanced modes of operations not supported by 8086. The 80286 boosted a new mode of operation-protected mode. Due to this the entire concept of memory segmentation was changed. The virtual memory management circuitry were included in the 80286, which allow an 80286 to operate in either real address mode or protected virtual address mode.

In 1986, the next advanced processor, the 80386DX, was introduced. As expected, 80386DX is faster than any of its predecessors, with a minimum operating frequency of 16 MHz. It is an 32-bit processor with 32-bit register set, address bus and data bus.

Internal cache memory	Chip	Introduction	Data bus	Address bus	Number of instructions executed per second
-	4004	1971	4	8	50000
-	8008	1972	8	8	50000
-	8080	1974	8	16	500000
-	8085	1977	8	16	769230
-	8086/88	1978	16/8	20	2.5 million
-	80186/188	1982	16/8	20	
-	80286	1983	16	24	4.0 million
-	80386DX	1986	32	32	25 million
	80386SX	1988	16	24	25 million
8 K	80486DX (With coprocessor)	1989	32	32	50 million
8 K	80486SX (Without coprocessor)	1989	32	32	50 million

Table 1.3 80X86 family tree

During 1988, an "economy version" of the 80386, called the 80386SX was introduced by Intel. This processor had the same outside connections as the 80286, but inside it was a 386-processor supporting the 386's expanded instruction set and various operating modes. The Table 1.3 shows the 80X86 family tree.

Early in 1989, Intel introduced the 80486DX, the more highly integrated microprocessor with built-in coprocessor. Meanwhile, Intel has also developed step-down version 80486SX (without coprocessor and lower clock speed).

The Pentium, introduced in 1993, was similar to the 80386 and 80486 microprocessors. It contained larger internal cache and data bus width is extended to 64-bit.

The Table 1.4 shows the comparison between various pentium processors.

Processor	Released year	Data bus width	Memory size	L1 cache data-code	L2 cache	Bus transfer speed
Pentium 60 MHz 66 MHz 120 MHz 133 MHz 233 MHz	1993	64	4 Gbyte	8 K - 8 K	–	60 - 66 MHz
Pentium Pro 150-166 MHz	1995	64	64 Gbyte	8 K - 8 K	256 K	60 - 66 MHz
Pentium II 350 MHz 400 MHz 450 MHz	1997	64	64 Gbyte	16 K - 16 K	512 K	100 MHz
Pentium II Xeon	1998	64	64 Gbyte	16 K - 16 K	512 K or 1 M	100 MHz
Pentium III 1 GHz Slot 1 version	1998	64	64 Gbyte	16 K - 16 K	512 K	100 MHz
Pentium III 1 GHz Flip chip version	1998	64	64 Gbyte	16 K - 16 K	256 K	100 MHz
Pentium III 1 GHz Celeron	1998	64	64 Gbyte	16 K - 16 K	256 K	66 MHz
Pentium IV 1.3 GHz 1.4 GHz 1.5 GHz	2000	64	64 Gbyte	16 K - 16 K	256 K	100 MHz

Table 1.4 Comparison between pentium processors

- Pentium IV uses the RAMBUS memory technology in place of SDRAM technology used in other pentium processors.

1.2 Microprocessor Basics

This section is written primarily to introduce to the students the microprocessor based system and the role of microprocessor in it. It also explains various components required to build a microprocessor system and the working of generalized microprocessor unit.

What is a Microprocessor Based System? Don't worry, we shall get fair idea about this on completion of this chapter.

As a part of introduction to the microprocessor based system, I would like to share philosophy behind this microprocessor based systems. Let us consider a small organization. It consists of managing body, manager and his subordinates, incoming unit and dispatch unit. (To maintain the simplicity and just to explain analogy I have considered small organization as shown in the Fig. 1.3)

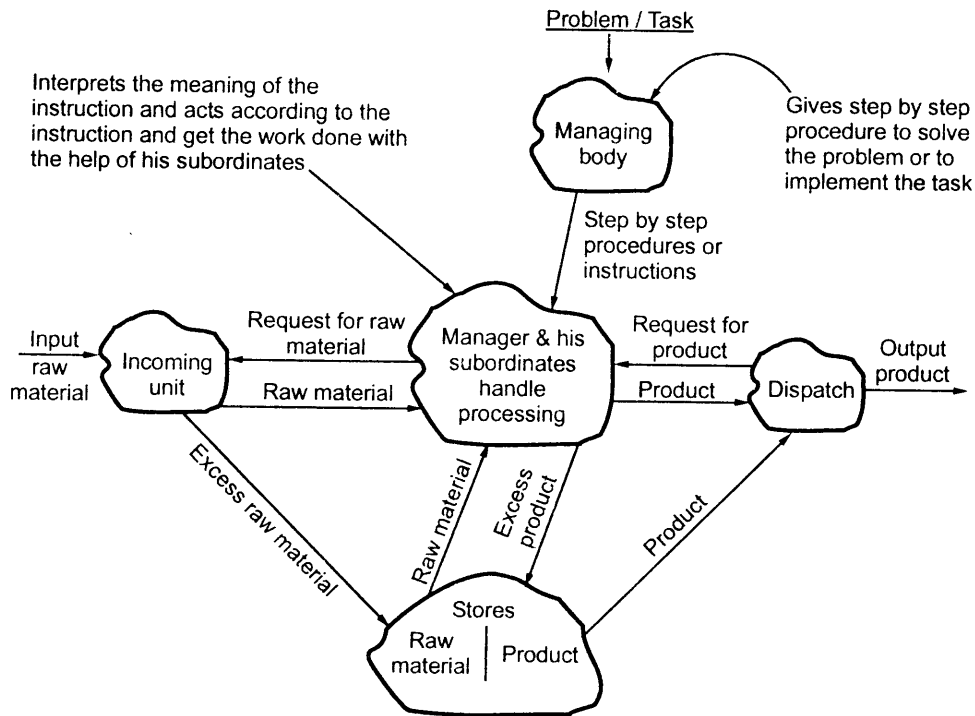


Fig. 1.3 Working of small organization

As shown in the Fig. 1.3, managing body gives step by step procedure to solve the problem or to implement the task. Manager interprets the step by step procedures and instructions given by the managing body and executes those with the help of his subordinates. The manager and his subordinates who belong to processing unit can request for raw material to stores. If raw material is not available in stores, processing unit can request for raw material to incoming unit. After processing the raw material, processing unit can either dispatch product or keep it in stores.

Looking at this environment somebody came up with ideas.

- Can we design such an organization without any involvement human beings?

- Can we replace human being with some electronic devices which can **mimic** the operations executed by human beings?

In this way the idea of microprocessor based system was born and design process was started. Over the period of design process, designers first came up with an integrated circuit which has an ability to mimic the processes executed by the processing unit. They replaced raw material by data and instructions by code with 1s and 0s. The integrated circuit accepts data as an input and process it according to instruction code to generate an information. As an example, we can say, marks of students in various subjects is a data and result prepared on the basis of these marks is an information.

To get the data as an input and to give information to the users, the designer team came up with the input and output devices which is an another integrated circuit. Therefore, with the use of input device processing unit can get the input and after processing the input, processor will give the information to the user with the help of output device. The integrated circuit was also designed which has an ability to store the data, instruction code and information. The processing unit reads instruction code from this circuit and act accordingly. Some meaningful name was given to each integrated circuit as given below.

For Processing Unit Integrated Circuit :

As it is a heart of organization and has an ability to process input as per the instructions of managing body, it was named '**Central Processing Unit**' (CPU) or simply '**Processor**'.

For Incoming Unit Integrated Circuit :

As it gives input data to the processor it was named **Input device**.

For Dispatch Unit Integrated Circuit :

As it gives information to the user, it was named **Output device**.

For Storage Unit Integrated Circuit :

As it stores data instruction code and information, it was called **Memory**.

The system with central processing unit, input device, output device and memory can be visualized as shown in the Fig. 1.4 (See Fig. 1.4 on the next page). The system designed here is very small in physical size as compared to the organization with a human being. The processor is an integrated circuit in a very small size and hence more appropriate name for it was suggested as a '**Microprocessor**'. The system as a whole was then referred to as '**Microprocessor Based System**'.

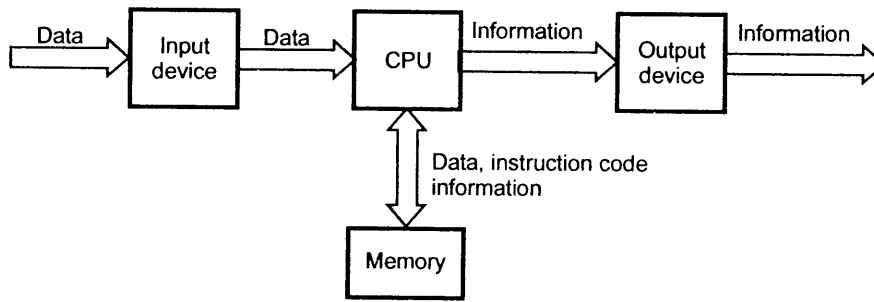


Fig. 1.4 Simple microprocessor based system

1.2.1 Simple Model of Microprocessor

In the last section we have seen the basic blocks of the microprocessor based system. Let us discuss each block with some details. This section explains the working of microprocessor block with help of simple model.

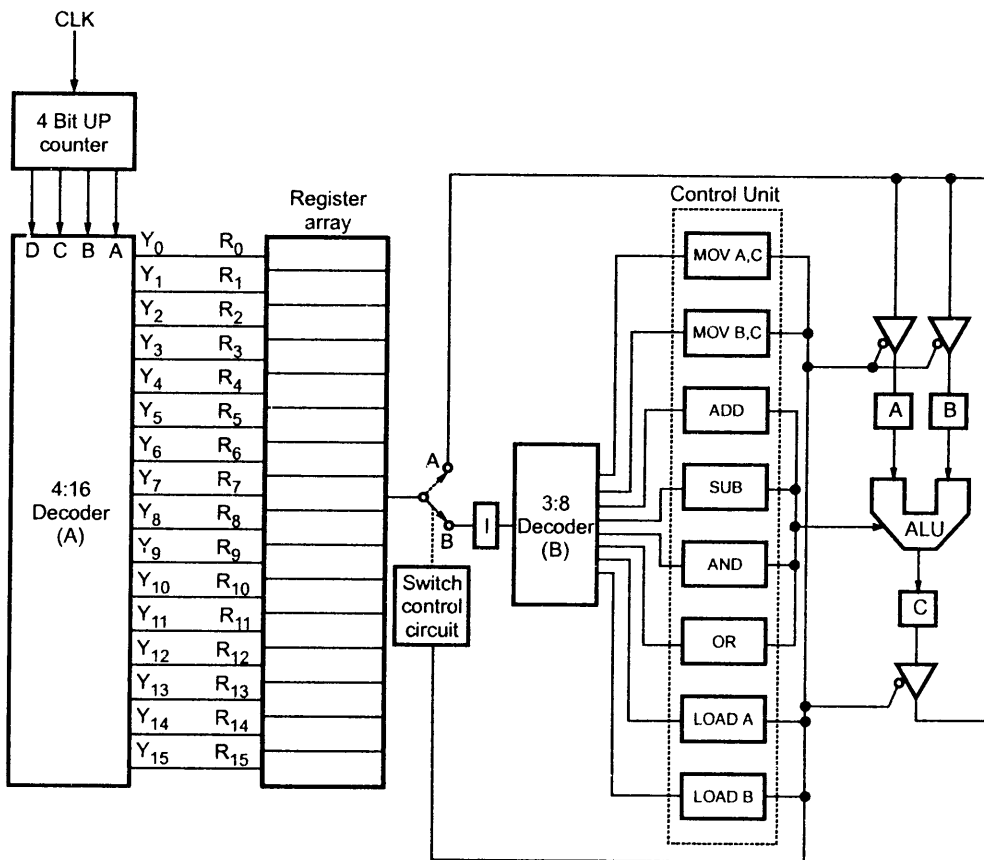


Fig. 1.5 Sequential digital circuit

Simple model is a sequential digital circuit as shown in the Fig. 1.5. It consists of 4 bit up-counter, decoders, registers, switch control circuit, control unit and arithmetic logic unit (ALU).

1.2.1.1 Counter

It is a 4-bit up-counter. It counts from 0 to 15 and it always starts from zero. The counter is driven by the clock signal and the output of counter is given to the decoder A.

1.2.1.2 Decoder A

It is a 4:16 decoder. The four-bit output of counter is used as an input for the decoder A. It activates its output depending on the status of inputs. The output of the decoder A is used to select one of the registers within the register array.

1.2.1.3 Register Array

Register array consists of sixteen 3-bit registers. Each register can store 3-bit word. The outputs of all registers are connected to the common bus through the Output Enable (OE) switch as shown in Fig. 1.6.

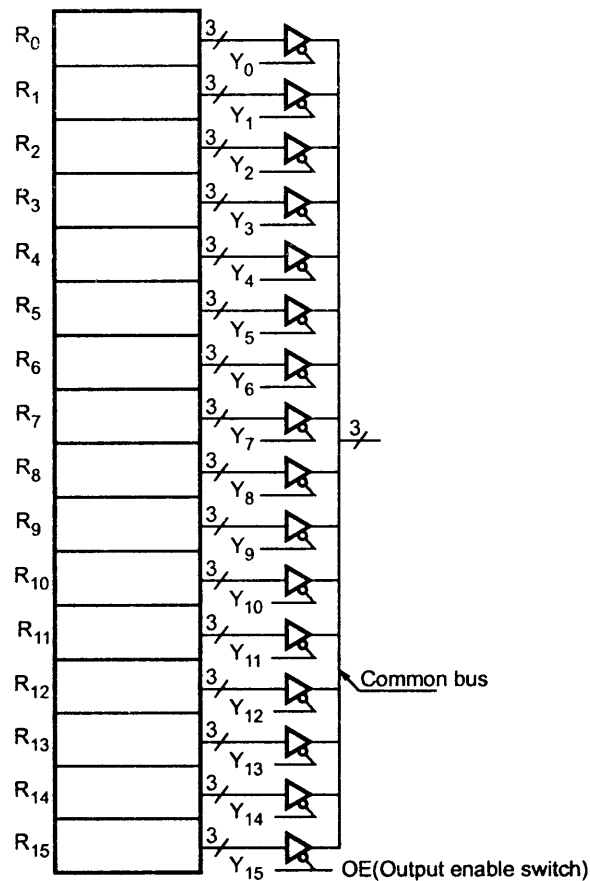


Fig. 1.6 Register array with tri-state switches

The output of the decoder A is used to activate output enable signals for sixteen registers (R_0 - R_{15}). At a time only one output of decoder is activated according to the input signal, so data from only one register is available on the common bus.

1.2.1.4 Common Bus

The three-bit common bus is connected to register A, register B, and to the register I through the switch. If the switch is at position A then common bus is connected to the A or B register; otherwise it is connected to the register I. The switch position is controlled by the switch control circuit.

1.2.1.5 Register I

It is used to store data from the common bus.

1.2.1.6 Decoder B

It is a 3:8 decoder. The data from the Register I is used as an input for the decoder. If switch is in position B, this data is decoded and used to select appropriate control circuit from the control unit as shown in Table 1.5.

D_2	D_1	D_0	Selected control circuit
0	0	0	MOV A, C
0	0	1	MOV B, C
0	1	0	ADD
0	1	1	SUB
1	0	0	AND
1	0	1	OR
1	1	0	LOAD A
1	1	1	LOAD B

Table 1.5

1.2.1.7 Control Unit

It consists of eight control circuits : MOV A, C, MOV B, C, ADD, SUB, AND, OR, LOAD A and LOAD B. These control circuits are used to generate signals which select the operation of ALU and activate input enable signal of register A and register B. At a time output from only one control circuit is activated as selected by the decoder output.

MOV A, C

This control circuit generates control signals which activate input enable signal for register A and output enable signal for register C. So the data from register C is copied into register A.

MOV B, C

This control circuit generates control signals which activate input enable signal for register B and output enable signal for register C. So the data from register C is copied into register B.

ADD

This control circuit generates control signals required for addition operation which is to be performed by the ALU i.e. $A + B \rightarrow C$.

SUB

This control circuit generates control signals required for subtraction operation which is to be performed by the ALU. i.e. $A - B \rightarrow C$.

AND

This control circuit generates control signals required for logical AND operation which is to be performed by the ALU.

OR

This control circuit generates control signals required for logical OR operation which is to be performed by the ALU.

LOAD A

This control circuit generates control signals which activate input enable signal for register A. It also indicates switch control circuit to change switch position to A.

LOAD B

This control circuit generates control signals which activate input enable signal for register B. It also indicates switch control circuit to change switch position to A.

1.2.1.8 Switch Control Circuit

Switch control circuit is responsible for switch position. It gets the input from LOAD A and LOAD B control circuits. If any of the above inputs is activated, it changes switch position to A, otherwise it holds switch at position B.

1.2.1.9 ALU

ALU takes the input from register A and register B. This input is processed according to the operation selected by the control unit. The process result is stored in the register C.

1.2.1.10 Operation

As mentioned earlier, counter starts from zero. When counter output is zero, decoder selects the first register (R_0) from the register array and the data from the selected register (R_0) is available on the common bus. Initially switch is at position B, so data from the common bus is decoded. According to data, control unit selects the operation and ALU performs the operation.

The important thing in this process is the data from selected register decides the operation. The data which decides the operation is called "Operation code" or "Opcode". Each operation has its own opcode. The Table 1.6 shows the opcodes for different operations.

For operations MOV A, C and MOV B, C the data is transferred from register C to register A and register B respectively.

In operations ADD, SUB, AND and OR data which is to be processed is taken from A and B registers. But in case of LOAD A and LOAD B operations data from common bus is directly loaded into the A and B register respectively. For this operation switch must be in position A.

Operation	Opcode		
	D ₂	D ₁	D ₀
MOV A, C	0	0	0
MOV B, C	0	0	1
ADD	0	1	0
SUB	0	1	1
AND	1	0	0
OR	1	0	1
LOAD A	1	1	0
LOAD B	1	1	1

Table 1.6

After each operation counter is incremented by one, so that decoder selects the next register. Again data from selected register is available on the common bus and the process is repeated. The following example will help you to understand the process sequence.

Example : Add two numbers.
Steps : Load 1st number in A.
 Load 2nd number in B.
 ADD two numbers.

First step is to load the number in register A. To do this, it is necessary to have opcode of LOAD A operation in the register 0 (R₀) and the number in the register 1 (R₁). Similarly in the second step it is necessary to have opcode of Load B operation in register 2 (R₂) and the number in register 3 (R₃). Finally, it is necessary to have opcode of ADD operation in register 4 (R₄).

Fig. 1.7 shows the register array contents if number 1 = 100 and number 2 = 010.

R ₀	1 1 0
R ₁	1 0 0
R ₂	1 1 1
R ₃	0 1 0
R ₄	0 1 0
R ₅	—
R ₆	—

Fig. 1.7

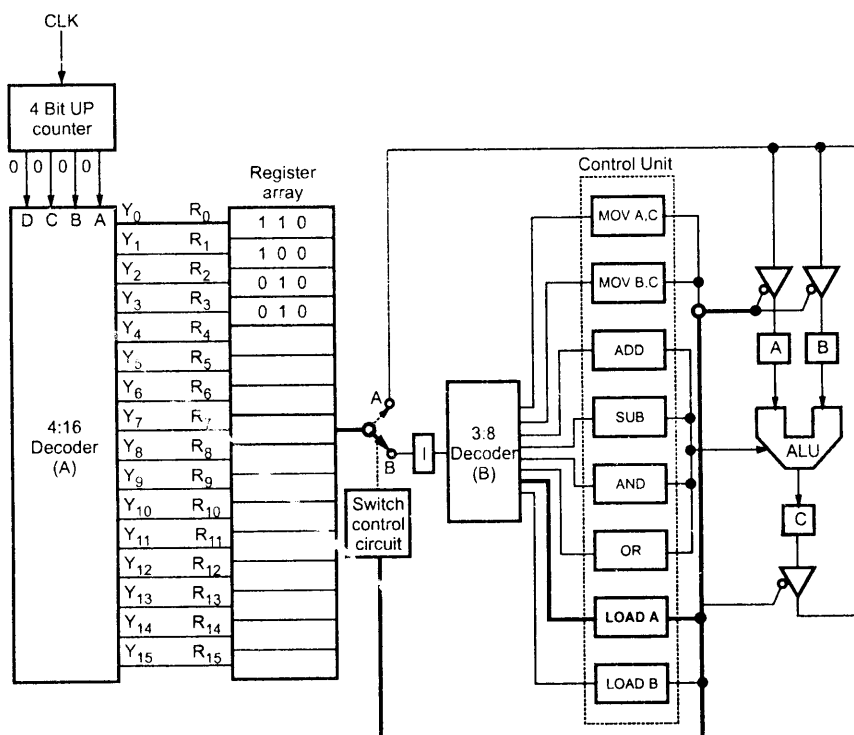


Fig. 1.8

Operation :

STEP 1 : Counter output (0000)

Initially the counter is in the reset condition so counter output is zero (0000). The decoder A will select the register 0 (R_0), since the input for decoder is zero. Once the register 0 (R_0) is selected, data from register 0 i.e. 1 1 0 is available on the common bus. The decoder B will use this data to select the operation. As data is 1 1 0, decoder B will select LOAD A operation. LOAD A operation will enable input for register A and change the switch position from B to A. Refer Fig. 1.8 on previous page.

STEP 2 : Counter output (0001)

The decoder A will select the register 1 (R_1), since the input for decoder is one. Once the register 1 is selected, data from register 1 i.e. 100 is available on the common bus. This data is directly transferred to the register A because switch is positioned at A and input for register A is already enabled. After data transfer, switch position is changed from position A to position B. Refer Fig. 1.9.

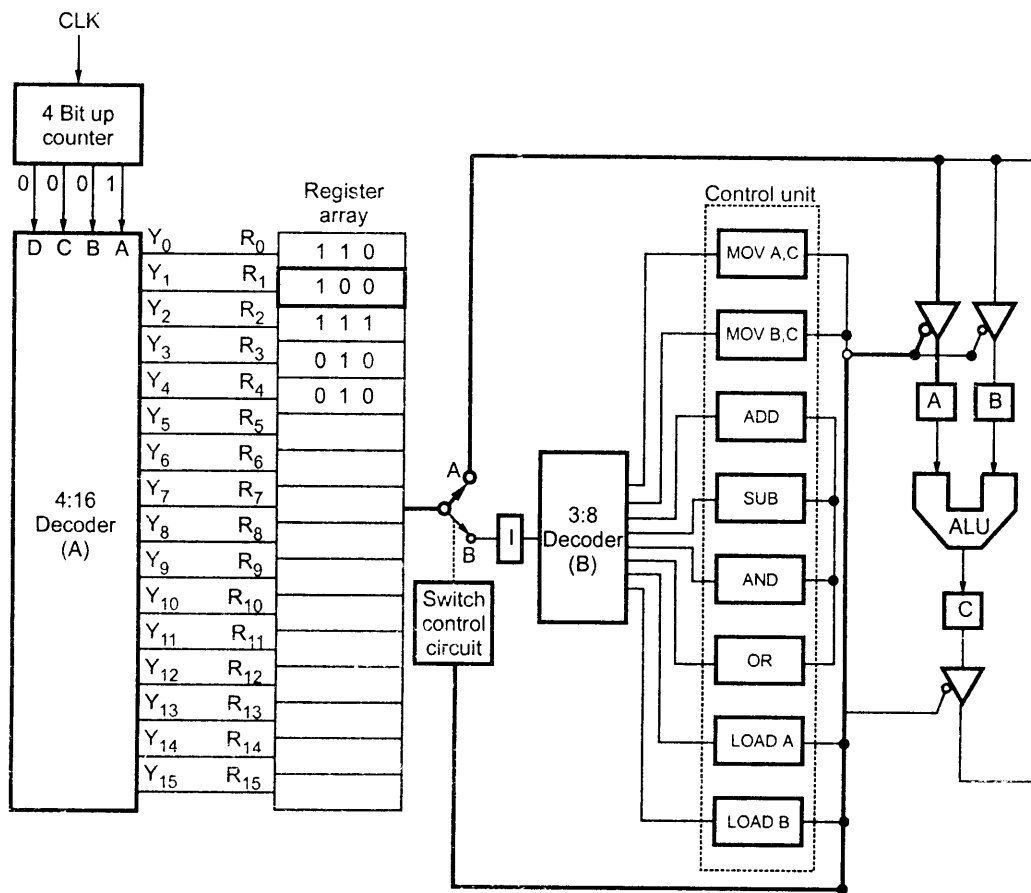


Fig. 1.9

STEP 3 : Counter output (0010)

The decoder A will select the register 2 (R_2). Once the register 2 (R_2) is selected, data from register 2 i.e. 1 1 1 is available on the common bus. The decoder B will use this data to select the operation. As data is 1 1 1, decoder B will select LOAD B operation. LOAD B operation will enable input for register B and change the switch position from B to A.

Refer Fig. 1.10.

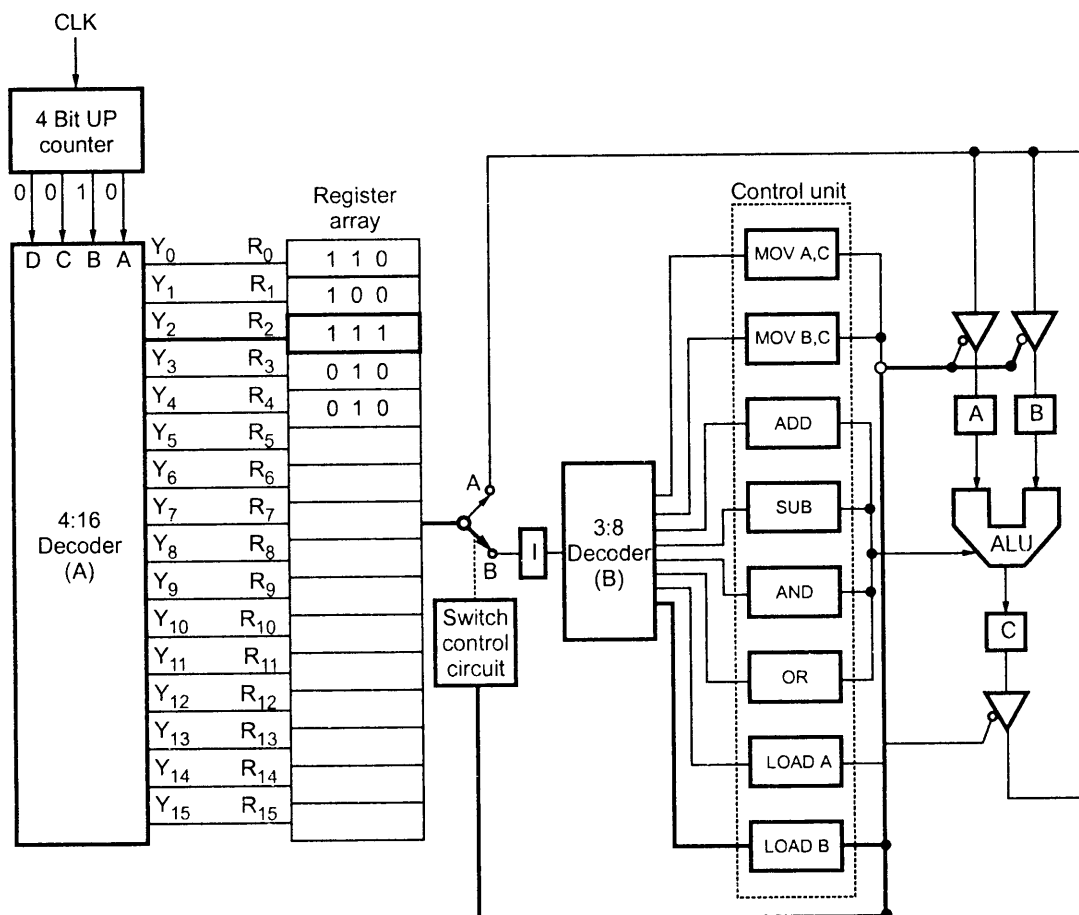


Fig. 1.10

STEP 4 : Counter output (0011)

The decoder A will select the register 3 (R_3). Once the register 3 is selected, data from register 3 i.e. 0 1 0 is available on the common bus. This data is directly transferred to the register B because switch is positioned at A and input for register B is already enabled. After data transfer, switch position is changed from position A to position B. Refer Fig. 1.10

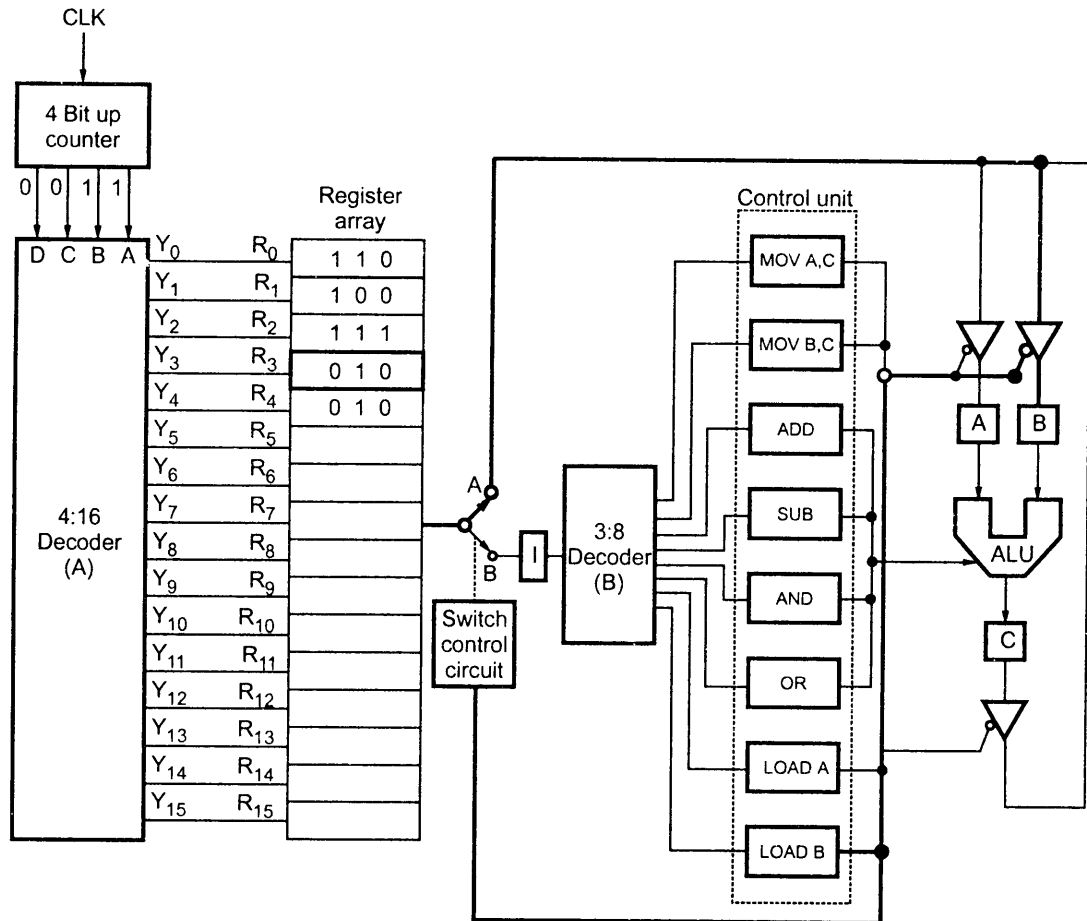


Fig. 1.11

STEP 5 : Counter output (0100)

The decoder A will select the register 4 (R_4). Once the register 4 is selected, data from register 4 i.e. 0 1 0 is available on the common bus. As data is 0 1 0, decoder B will select ADD control circuit. ADD operation will generate signals to select add operation which is to be performed by ALU. Then ALU will add the contents of register A and register B and it will store the result in the register C i.e. 1 1 0. Refer Fig. 1.12 on next page

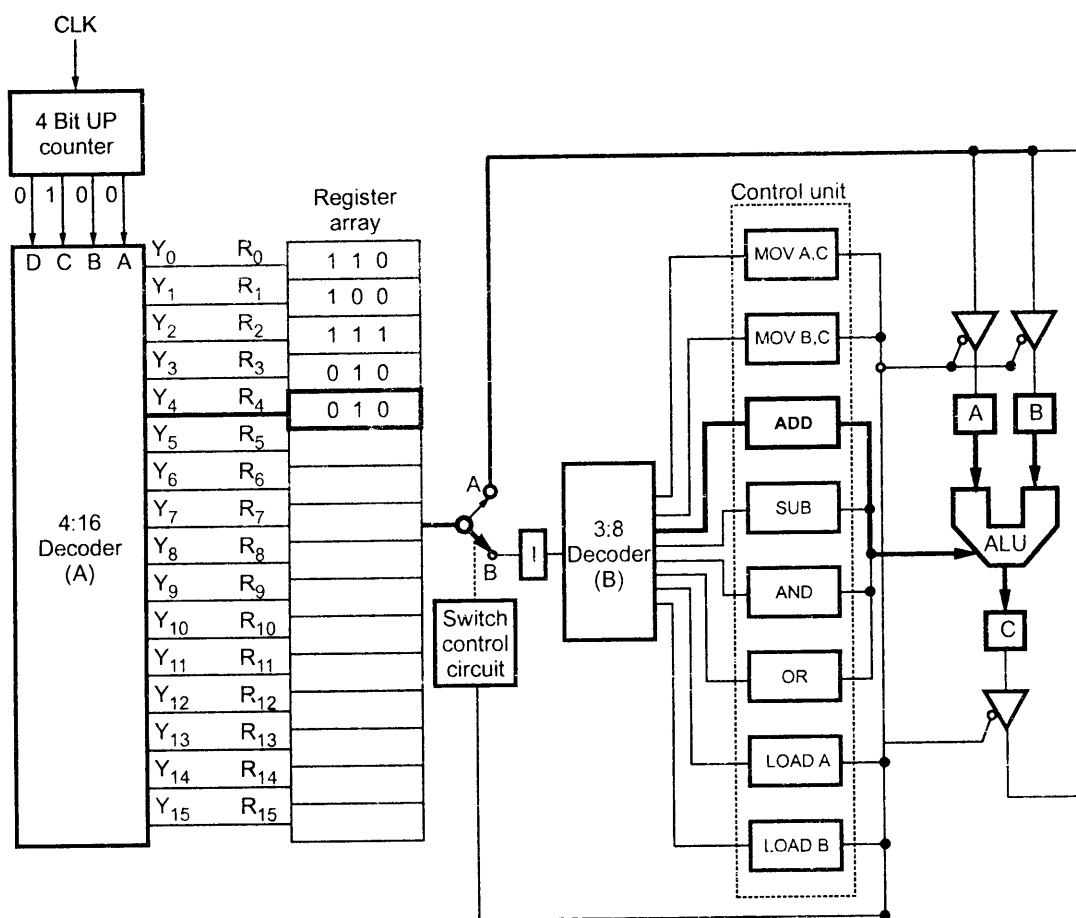


Fig. 1.12

The above example shows how two numbers can be added using the sequential circuit. Using same circuit we can subtract, multiply, divide, logically AND or logically OR the two numbers. This can be achieved by writing appropriate opcodes with necessary data in proper sequence into the register array.

Another important thing you must be noticed that LOAD A and LOAD B operations require two steps. In step one, register input is enabled and switch position is changed. In the second step actual data is moved from the next register. This data is not the operation code/opcode, but it is called **operand**. It is important to note that, to complete one operation, opcode or sometimes opcode and operand are required. This complete operation is referred as **instruction**. Thus instruction is a combination of opcode and operand.

Instruction sequences for different operations.

a) Subtract 0 1 0 (2) from 1 1 1 (7)

Register array			Operation
1	1	0	Load A register.
1	1	1	With 7 (1 1 1).
1	1	1	Load B register.
0	1	0	With 2 (0 1 0).
0	1	1	Subtract contents of B register from the contents of A register and store the result in register C.

b) Logically AND 0 1 1 (3) with 1 0 1 (5)

Register array			Operation
1	1	0	Load A register.
0	1	1	With 3 (0 1 1).
1	1	1	Load B register.
1	0	1	With 5 (1 0 1).
1	0	0	Logically AND the contents of register A and register B and store the result in register C.

c) Solve 1 0 1(5) + 0 0 1 (1) – 0 1 1 (3)

Register array			Operation
1	1	0	Load register A.
1	0	1	With 5 (1 0 1).
1	1	1	Load register B.
0	0	1	With 1 (0 0 1).
0	1	0	Add contents of register A with the contents of register B and store the result in register C.
0	0	0	Copy the contents of register C to register A.
1	1	1	Load register B.
0	1	1	With 3 (0 1 1).
0	1	1	Subtract contents of register B from the contents of register A and store the result in register C.

d) Solve : 100 (4) \wedge 0 1 1 (3) \vee 0 0 1 (1)

Register array			Operation
1	1	0	Load register A.
1	0	0	With 4 (100).
1	1	1	Load register B.
0	1	1	With 3 (011).
1	0	0	Logically AND the contents of register A and register B and store the result in register C.
0	0	0	Copy the contents of register C to register A.
1	1	1	Load register B.
0	1	1	With 3 (0 1 1).
1	0	1	Logically OR the contents of register A and register B and store the result in register C.

From the above description and examples you must have noticed that in simple model, hardware components are fixed, but by changing the sequence of opcodes and operands (instructions) it is possible to process different operations. Controlling the operation in a fixed hardware by changing sequence of instructions is the basic principle used in the microprocessor. The simple model used in this chapter is a very small version of the microprocessor. In the next section we will discuss the various terminologies used in the microprocessor.

1.2.2 Terminologies used in Microprocessor

Fig. 1.13 shows the small model with specific labels and special blocks. Block 1 represents a microprocessor whereas block 2 represents memory.

Microprocessor : It consists of counter, switch control circuit, decoder B, control unit, ALU and registers.

Memory : It is a combination of register array and the decoder circuit.

Program : The sequence of instructions written for specific operation is called **program**.

Program counter : The counter used in a small model is used to locate instructions in a proper sequence (program). Thus it is called a program counter.

Address and address bus : The output of the program counter is given to the memory. The internal decoder (decoder A) in the memory, decodes this output of the program counter to locate a specific register in the memory. This means that output of the program counter addresses a specific register in the memory. Thus the output of program counter is called address and the group of wires that carries this address is called **address bus**. Address bus shown in Fig. 1.12 is 4-bit wide. So it can address $16 (2^4)$ registers in the register array. The number of address lines decide the how many registers from the register array (memory locations) can be accessed. For example if there are 16 address lines then it is possible to access 2^{16} (65536) memory locations.

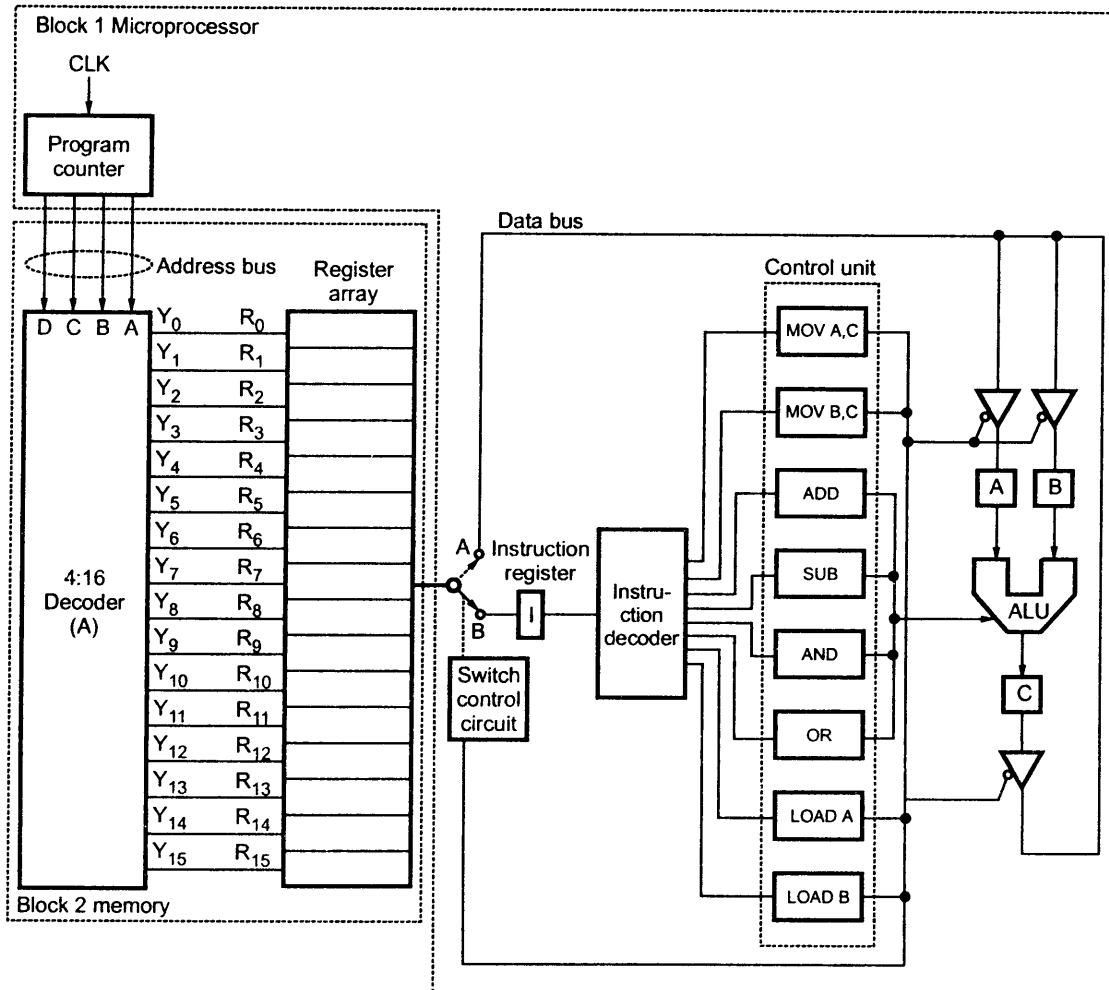


Fig. 1.13

Data and data bus : The data within the registers is used in different forms such as opcodes and operands. This data is moved from register to register, from memory to register through group of wires called **data bus**. The data bus within the microprocessor is called **internal data bus** whereas data bus outside the microprocessor is called **external data bus**. Data bus shown in the Fig. 1.12 is 3-bit wide. The size of the data bus decides the size of operands. It also decides the maximum number of operations that microprocessor can perform. As data bus in the small model is 3-bit wide it can perform (2^3) eight different operations. The standard size of data bus is in multiple of 8 i.e. 8, 16, 24, 32 etc. The microprocessor with 8-bit data bus can execute maximum 256 (2^8) instructions.

Word length : The group of bits that microprocessor can recognize and process is called word, and microprocessors are classified according to their word length. For example, a microprocessor with an 8-bit word is known as 8-bit microprocessor and a microprocessor with an 16-bit word is known as 16-bit microprocessor.

Instruction register : The register in which instruction opcode is stored temporarily is called instruction register.

Instruction decoder : The decoder (decoder B in small model) which takes the instruction opcode from the instruction register and decodes it to generate appropriate control signals corresponding to the instruction is called instruction decoder.

1.2.3 Different Phases in the Execution Process

The three blocks in the Fig. 1.14 shows the three different phases involved in the execution process. These are : 1) Fetch 2) Decode 3) Execute

Refer Fig. 1.14 on next page.

1.2.3.1 Fetch

In fetch phase, microprocessor places the contents of the program counter on the address bus and gets instruction code, opcode from the addressed memory location. The microprocessor then saves opcode in the instruction register.

1.2.3.2 Decode

In this phase, the opcode from the instruction register is decoded with the help of instruction decoder to generate appropriate control signals to execute the instruction.

1.2.3.3 Execute

In this phase, microprocessor generates appropriate control signals and executes the instruction.

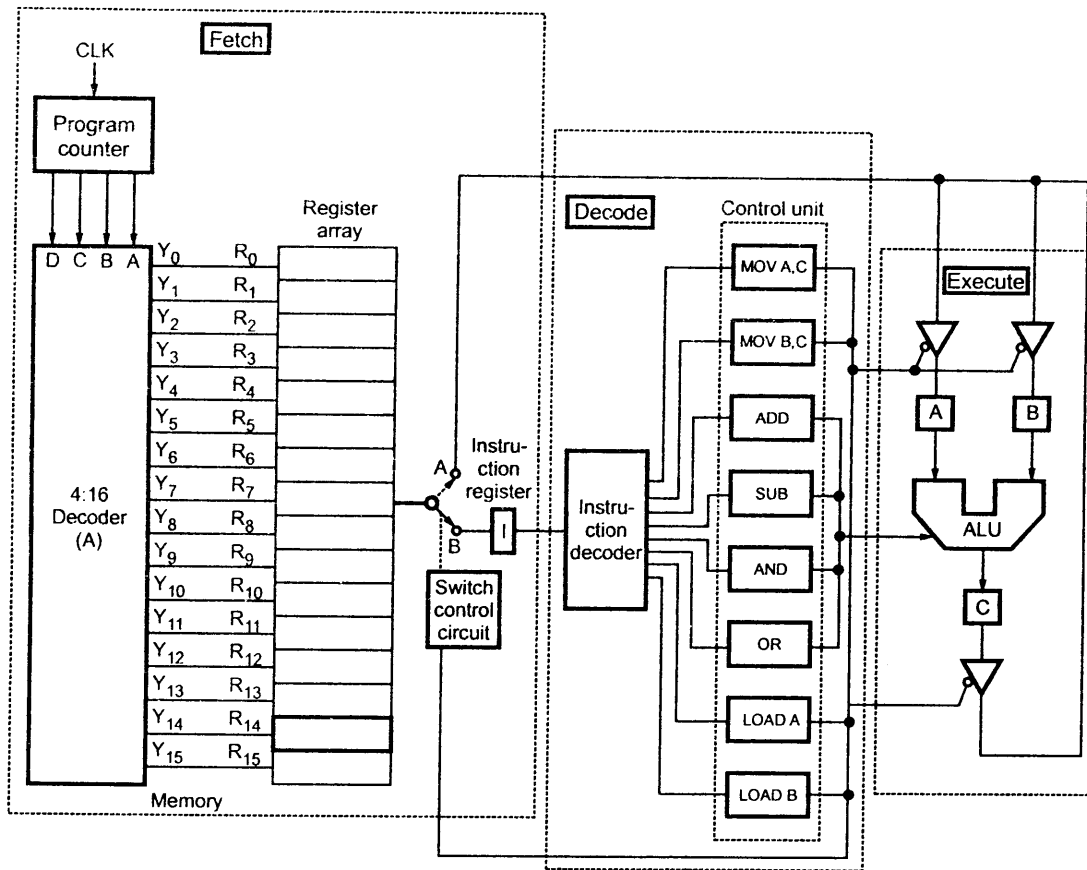


Fig. 1.14 Fetch, decode and execute

1.2.4 Microprocessor and Programmer's Model

In the last section we have seen simple model of microprocessor system. We have seen the important components of the microprocessor system like microprocessor, memory unit and clock. In microprocessor block we have seen program counter, instruction decoder, instruction register, different registers, control logic and ALU. In this section, we are going to see more practical microprocessor block diagram and programmer's model. (Refer Fig. 1.15 and Fig. 1.16). The microprocessor's block diagram and the microprocessor programming model show you how a specific microprocessor is constructed. The block diagram shows microprocessor's functions for data processing and data handling. It also shows how each of these logic functions are connected together. The programming model assists you in the programming process. The difference is that the programming model shows only those parts of the microprocessor which the programmer can change. So we can say that block diagram makes it easier to understand the architecture of a

microprocessor and the programming model makes it easier to understand the working of microprocessor in a programming environment.

The block diagram shown in Fig. 1.15 includes three major logic devices.

- ALU
- Several registers
- Control unit

The internal data bus is used to transmit data between these logic devices.

ALU :

One of the microprocessor's major logic devices is the arithmetic logic unit (ALU). It contains the microprocessor's data processing logic. It has two inputs and an output. The internal data bus of microprocessor is connected to the two inputs of ALU through the temporary register and the accumulator.

The ALU's single output is connected to the internal data bus. It allows to send the output of ALU over the bus to any device connected to the bus. In most of the microprocessors register A gives data for the ALU and after performing the operation, the resulting data word is sent to the register A and stored there. This special register, where the result is accumulated is commonly known as **accumulator**.

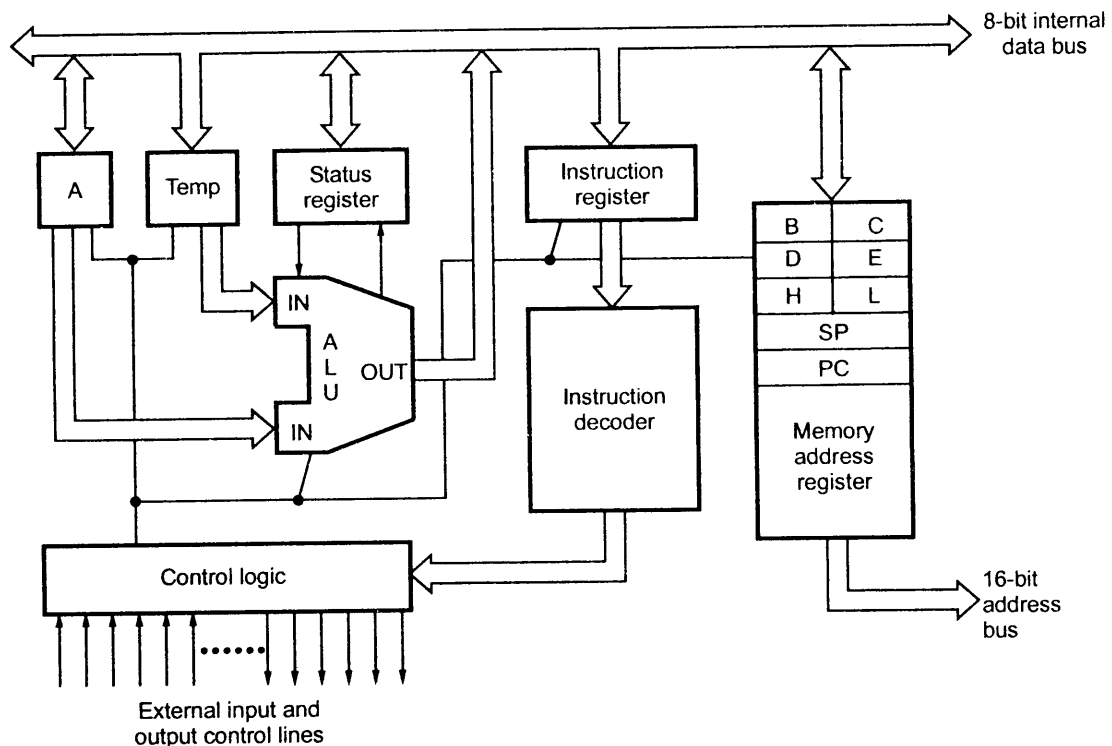


Fig. 1.15 Block diagram of microprocessor

The ALU works on either one or two data words, depending on the kind of operation. The ALU uses input ports as necessary. For example, addition operation uses both ALU inputs while complementing data operation uses only one input. To complement the data word, all the bits of the word's that are logic 1 are set to logic 0, and all the bits of the word at logic 0 are set to logic 1.

The ALU of the most of the microprocessors can perform following functions.

- Add
- Subtract
- AND
- OR
- Exclusive OR
- Complement
- Shift right
- Shift left
- Increment
- Decrement

Registers :

Registers are a prominent part of the block diagram and the programming model of any microprocessor. The basic registers found in most of the microprocessors are the accumulator, the program counter, the stack pointer, the status register, the general purpose registers, the memory address register, the instruction register and the temporary data registers.

The Accumulator :

The accumulator is the major working register of microprocessor. Most of the time it is used to hold the data for manipulation. Whenever the operation processes two words, whether arithmetically or logically, the accumulator contains one of the words. The other

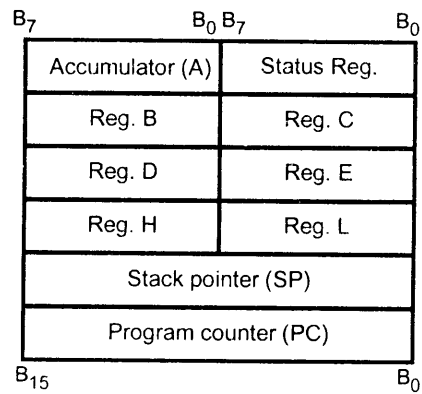


Fig. 1.16 Programmer's model

word may be present in another register or in a memory location. Most of the times the result of an arithmetic or logical operation is placed in the accumulator. In such cases, after execution of instruction original contents of accumulator are lost because they are overwritten.

The accumulator is also used for data transfer between an I/O port and a memory location, or between one memory location and another.

The Program Counter :

The **Program Counter** is one of the most important registers in the microprocessor. As mentioned earlier, a program is a series of instructions stored in the memory. These instructions tell the microprocessor exactly how to solve a problem. It is important that these instructions must be executed in a proper order to get the correct result. This sequence of instruction execution is monitored by the program counter. It keeps track of which instruction is being used and what the next instruction will be.

The program counter gives the address of memory location from where the next instruction is to be fetched. Due to this the length of the program counter decides the maximum program length in bytes. For example, microprocessor that has 16-bit program counter, can address 2^{16} bytes (64 K) of memory.

Before the microprocessor can start executing a program, the program counter has to be loaded with valid memory address. This memory location must contain the opcode of first instruction in the program. In most of the microprocessors this location is fixed. For example, memory address (0000H) for 16-bit program counter. The fixed address is loaded into the program counter by resetting the microprocessor.

As said earlier, the instructions must be executed in a proper order to get the correct result. This does not mean that every instruction must follow the last instruction in the memory. But it must follow the logical sequence of the instructions. In some situations, it is better to execute part of a program that is not in sequence (don't confuse with logical sequence) with the main program. For example, there may be a part of a program that must be repeated many times during the execution of the entire program. Rather than writing repeated part of the program again and again, the programmer can write that part only once. This part is written separately. The part of the program which is written separately is called **subroutine**. The Fig. 1.17 shows how the main and subroutine programs are executed.

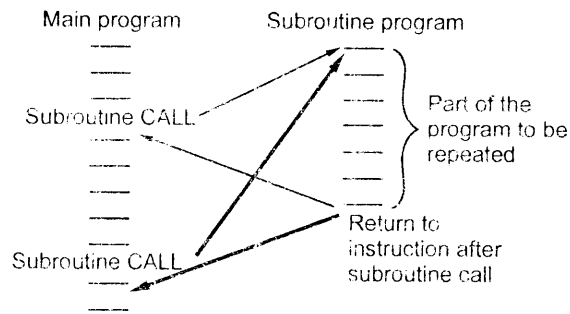


Fig. 1.17 Execution of subroutine programs

The program counter does the major role in subroutine execution as it can be loaded with required memory address. With the help of instruction it is possible to load any memory address in the program counter. When subroutine is to be executed, the program counter is loaded with the memory address of the first instruction in the subroutine. After execution of the subroutine, the program counter is loaded with the memory address of the next instruction from where the program control was transferred to the subroutine program.

The status register : The status register is used to store the results of certain condition when certain operations are performed during execution of the program. The **status register** is also referred to as **flag register**. ALU operations and certain register operations may set or reset one or more bits in the status register. Status bits lead to a new set of microprocessor instructions. These instructions permit the execution of a program to change flow on the basis of the condition of bits in the status register. So the condition bits in the status register can be used to take logical decisions within the program. Some of the common status register bits are :

1) **Carry/Borrow :** The carry bit is set when the summation of two 8-bit numbers is greater than 1111 1111 (FFH). A borrow is generated when a large number is subtracted from a smaller number.

2) **Zero :** The zero bit is set when the contents of register are zero after any operation. This happens not only when you decrement the register, but also when any arithmetic or logical operation causes the contents of register to be zero.

3) **Negative or sign :** In 2's complement arithmetic, the most significant bit is a sign bit. If this bit is logic 1, the number is negative number, otherwise a positive number. The negative bit or sign bit is set when any arithmetic or logical operation gives a negative result.

4) **Auxiliary carry** : The auxiliary carry bit of status register is set when an addition in the first 4-bits causes a carry into the fifth bit. This is often referred as half carry or intermediate carry. This is used in the **BCD arithmetic**.

5) **Overflow flag** : In 2's complement arithmetic, most significant bit is used to represent sign and remaining bits are used to represent magnitude of a number (See Fig. 1.17). This flag is set if the result of a signed operation is too large to fit in the number of bits available (7-bits for 8-bit number) to represent it.

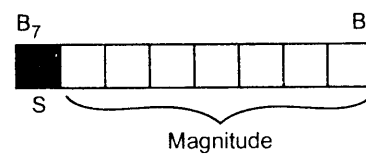


Fig. 1.18 2's Complement 8-bit number

For example, if you add the 8-bit signed number 01110110 (+118 decimal) and the 8-bit signed number 00110110 (+ 54 decimal), the result will be 10101100 (+ 172 decimal), which is the correct binary result, but in this case it is too large to fit in the 7-bits allowed for the magnitude in an 8-bit signed number. The overflow flag will be set after this operation to indicate that the result of the addition has overflowed into the sign bit.

6) **Parity** : When the result of an operation leave the indicated register with an even number of 1s, parity bit is set.

The stack pointer :

This is an important register which programmer uses frequently. In the earlier sections we have seen how subroutines are executed by changing the program counter contents. But one question you may have in your mind is that how the program counter is loaded with the address of the next instruction (return address) from where the program control was transferred to the subroutine. This return address is kept in a special memory area called **the stack**. Before transferring the program control to the subroutine the return address is pushed onto the stack. After the execution of subroutine the return address is popped off from the stack and loaded into the program counter.

The memory address of the stack area is given by a special register called the **stack pointer**. Like the program counter, the stack pointer automatically points to the next available location in the memory. In most microprocessors, the stack pointer decrements (points to the next lower memory address) when data is pushed on the stack. This allows the programmer to build the stack down in memory as shown in the Fig. 1.18. Usually stack operations are 2 byte operations. This means that the stack pointer decrements by two memory address locations each time when 2 byte data is pushed on the stack. When the data is popped off from the stack, the stack pointer is incremented by two memory address locations.

It is important to note that as you go on storing (pushing) data on the stack, the stack pointer always points the last data placed on the stack and when you try to remove (pop) data you always get the last data placed on the stack. This kind of stack operation is called **LIFO (Last In First Out)** operation.

General Purpose Registers :

In addition to the six basic registers, most microprocessors have other registers called **general purpose registers**. The general purpose registers are used as simple storage area, mainly these are used to store intermediate results of the operation. Getting the operand from the general purpose registers is more faster than from memory so it is better to have sufficient number of general purpose register in the microprocessor. The microprocessor used in this chapter has six general purpose registers (Refer Fig. 1.14) called the B, C, D, E, H, and L registers. These registers individually can operate as 8-bit registers. Together the BC, DE and HL registers can operate as 16-bit register pairs.

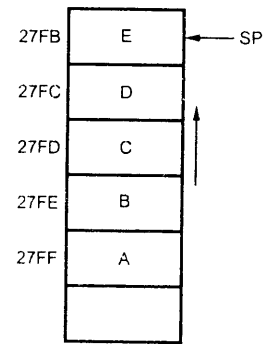


Fig. 1.19 Stack operation

Memory Address Register :

The memory address register gives the address of memory location that the processor wants to use. That is, memory address register holds 16-bit binary number. The output of the memory address register drives the 16-bit address bus. This output is used to select a memory location.

The Instruction Register :

The instruction register holds the operation code (opcode) of the instruction the microprocessor is currently executing. The instruction register is loaded during the opcode fetch cycle. The contents of the instruction register is used to drive the part of the control logic known as the **instruction decoder**.

Temporary Data Register :

The need for the temporary data registers arises because the ALU has no storage of its own. The ALU has two inputs. One input is supplied by accumulator and other from temporary data register. The programmer cannot access this temporary data register and, therefore it is not a part of programming model.

Control Logic :

The control logic is an important block in the microprocessor. The control logic is responsible for working of all other parts of the microprocessor together. It maintains the synchronization in operation of different parts in the microprocessor. The synchronization is achieved with the help of one of the control logic's major external inputs, microprocessor's clock. The clock is a signal which is the basis of all the timings inside the microprocessor.

Usually microprocessor's control logic is **microprogrammed**. This means that the architecture of the control logic itself is much like the architecture of a very special purpose microprocessor.

The control logic receives the signal from instruction decoder which decodes the instruction stored in the instruction register. The control logic then generates the control signals necessary to carry out this instruction. The control logic does a few other special functions. It looks after the microprocessor power-up sequence. It also processes interrupts. An interrupt is like a request to the microprocessor from other external devices such as the memory and I/O. The interrupt asks the microprocessor to execute a special program.

Internal Data Bus :

The internal data bus connects the different parts of microprocessor together and it enables the communication between these parts. The data transfer through this internal data bus is controlled by control logic.

Microprocessor's internal data bus usually connected to an external data bus. Due to this microprocessor can communicate with external memory or I/O devices. Usually the internal data bus is connected to the external data bus by logic called a bi-directional bus (transceiver).

1.3 Microprocessor Based Personal Computer System

Recent years, there are many positive changes in the computer system. The size of the computer system: getting reduced whereas its memory capacity, processing speed and power is tremendously increased. This improvement was possible because of microprocessor. The computer system built with microprocessor is known as microprocessor based computer system. The computer systems which specially used for

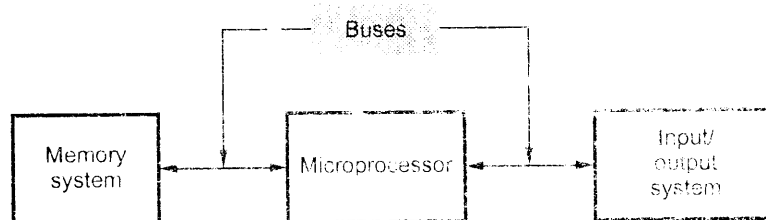


Fig. 1.20 The structure of microprocessor based personal computer system

desktop and personal use are therefore known as microprocessor based personal computer systems. Let us see the structure of such system. The Fig. 1.20 shows the basic structure of microprocessor based personal computer system.

Microprocessor based personal computer includes three blocks : Memory system, Microprocessor and input/output system. These blocks are interconnected by buses. Buses are used to carry the data between them.

1.3.1 Memory System

Memory system of microprocessor based personal computer is divided into three parts as shown in Fig. 1.21.

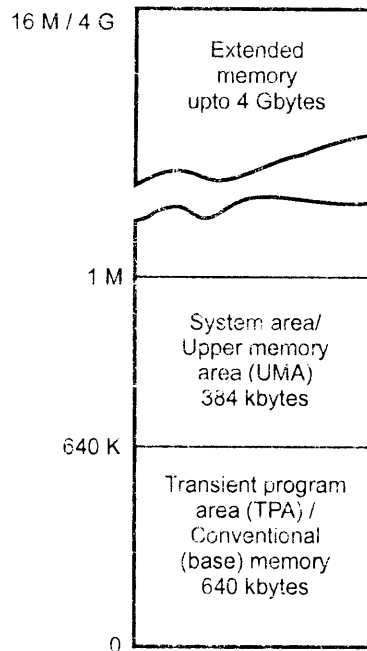


Fig. 1.21 Typical map of memory system of microprocessor based personal computer system

The memory system in the computer is organized by the operating system. The operating system decides the logical layout of the memory system.

The operating system manages the small and large pieces of different kinds of memory, some accessible by software application programs, and some not. These are as follows :

- Conventional (Base) Memory
 - Upper Memory Area (UMA)
 - Extended Memory Area (XMS)
 - High Memory Area (HMA)
 - Expanded Memory (Absolute)
- } Direct Addressable Memory
- } Not Direct Addressable Memory

1.3.1.1 Conventional (Base) Memory (Transient Program Area)

The original PC/XT type was designed to use 1 Mbyte of memory workspace consists of random access memory. This 1 Mbyte RAM is divided in various sections, some of which has a special uses. DOS can read and write to the entire megabyte but can manage the loading of programs only in the portion of RAM. On IBM PCs and compatibles, the memory occupied by DOS and other programs starts at address 00000H and may reach as high as address 09FFFH ; the 640 kbyte area of RAM. This area of RAM is referred to as **conventional memory** or transient program area (TPA). TPA stores application and system programs. The Fig. 1.22 shows the detail map of TPA memory. As shown in Fig. 1.22, out of 640 kbytes of TPA memory, 35 kbytes memory space is used for system programs, data and drivers. The remaining part of memory space is available for application programs.

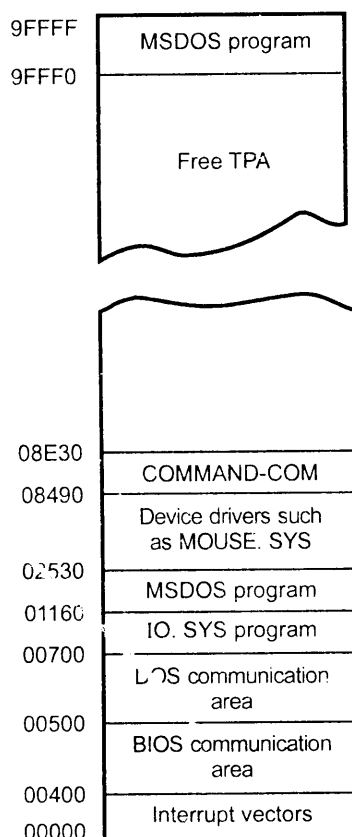


Fig. 1.22 TPA memory map

- The interrupt vectors access various features of DOS, BIOS and applications.
- The system BIOS and DOS communication areas contains transient data used by programs to access I/O devices and internal features of the computer system.
- IO.SYS is a program that loads into the TPA from the disk whenever an MSDOS or PC DOS is started. It contains the programs that allows DOS to use keyboard, video display, printer and other I/O devices.
- MSDOS programs are stored at two locations that is shown in Fig. 1.22 one is at the top at 9FFF0H address and one is at 01160H locations. DOS program controls the operation of computer system.
- Device drivers controls the installation of I/O devices. These are the files with .SYS and .EXE extensions.
- Whenever DOS commands are given through keyboard COMMAND.COM program processes that commands.
- Free TPA contains application programs like spread sheet programs, CAD programs, word processor etc.

1.3.1.2 Upper Memory Area (UMA)

The term **Upper Memory Area** also called **System Area**, describes the reserved 384 kbytes at the top of the first megabyte of the system memory on a PC/XT and AT

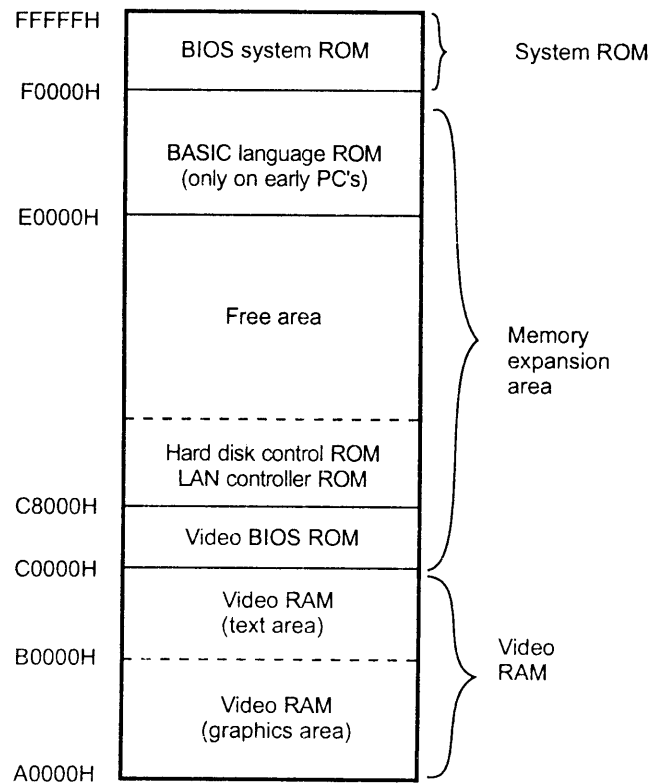


Fig. 1.23 System area map

systems. This memory has addresses from A0000H through FFFFFFFH. This memory is divided into three major portions, as shown in the Fig. 1.23.

- Video RAM
- Memory Expansion Area
- System ROM

Video RAM

The first 128 kbytes after conventional memory is called **Video RAM**. It is reserved for use by video adapters. The memory space allocated for this video RAM is from A0000H through BFFFFFFH. When text or graphics is displayed on the screen its image is stored in this section of memory. Memory locations from A0000H through AFFFFFFH are used to store graphical data and the memory locations from B0000H to BFFFFFFH are used to store text data.

This memory space is allocated for various video adapters as shown in the Table 1.7.

Display adapter	Portion reserved in video RAM
Monochrome Display Adapter (MDA)	B0000 - B0FFF (4 K)
Graphics Adapter (CGA)	B8000 - (16 K)
Enhanced Graphics Adapter (EGA)	A0000 - (128 K)
Video Graphics Adapter (VGA)	A0000 - (128 K)
Super Video Graphics Adapter (SVGA)	A0000 - (128 K)

Table 1.7 Video graphics adapters and their portion reserved in the video RAM

Memory Expansion Area

The next 192 kbytes are reserved as memory expansion area, that resides on read-only memory (ROM). The memory space allocated for this area is from C0000H through F0000H. This memory space is mainly reserved for BIOS adapters such as EGA BIOS, hard disk controller BIOS, IBM PC Network NETBIOS and 32 K IBM cluster adapter. The memory area from EC000H through EFFFFFFH is reserved for spare ROM sockets on AT.

System ROM

The last 64 K of memory is reserved for motherboard BIOS, (The basic Input/Output System). It resides on read only memory (ROM). The memory space allocated for this area is from F0000H through. Along with the BIOS it consists of routines for POST (Power-ON Self Test) and bootstrap loader, which handles computer system at bootup until the operating system takes over.

1.3.1.3 Extended Memory

The memory map on a system based on the 80286 or higher processor can be extended beyond the 1 Mbytes boundary. On a 80286 or 80386SX system, the extended memory limit is 16 Mbytes; on a 80386 DX, 80486, Pentium MMX or Pentium pro system, the

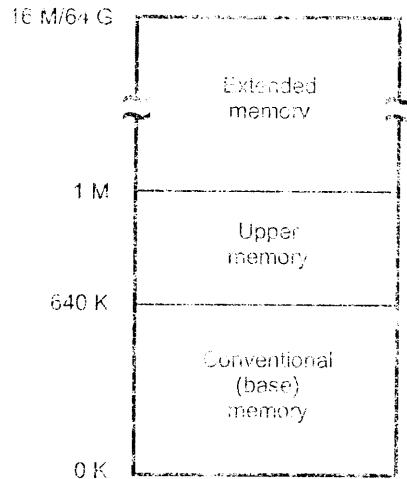


Fig. 1.24 Conventional and extended memory map

programs at once in virtual real mode is called multitasking.

The extended memory specification (XMS) was developed in 1987 by Microsoft, Intel, AST Corp., and Lotus Development to specify how programs would use extended memory. The XMS specification functions on systems based on the 80286 or higher and allows real mode programs (those designed to run in DOS) to use extended memory. Extended memory can be made to conform to the XMS specification by installing a device driver in the CONFIG.SYS file. The most common XMS driver is HIMEM.SYS, which is included with windows 3.X and later versions of DOS, starting with 4.0 and up.

1.3.1.4 High Memory Area (HMA)

The **High Memory Area (HMA)** is an area of memory 16 bytes short of 64K in size, starting at the beginning of the first megabyte of extended memory. It can be used to load device drivers and memory - resident programs to free up conventional memory for use by real mode programs. Only one device driver or memory - resident program can be loaded into HMA at one time, no matter what its size.

The HMA area is extremely important to those who use DOS 5 or higher versions because these DOS versions can move their own kernel (about 45 kbytes of program instructions) into this area. This, in effect, frees 45 K of conventional memory. The DOS Kernel can be moved into extended memory by first loading an XMS driver (such as HIMEM.SYS) and adding the line `DOS = HIGH` to system's CONFIG.SYS file.

extended memory limit is 4 G (4,096 Mbytes). Systems based on the new Pentium II processor have a limit of 64 Gbytes (65,536 Mbytes). Fig. 1.24 shows the map for extended memory.

On 80286, only programs designed to run in protected mode can take advantage of extended memory. The 80386 and higher processor offer another mode, called virtual real mode, which enables extended memory to be, in effect, chopped into 1 Mbyte pieces (each its own real mode session). The virtual real mode then allows several of these sessions to be running simultaneously in protected areas of memory. Running several

1.3.1.5 Expanded Memory

Unlike conventional or extended memory, expanded memory is not directly addressable by the processor. Instead, it can only be accessed through a 64 K and small 16 K pages established in the upper memory area (UMA). These small pages are also referred to as page frames. The exact location of the page frame is user configurable, so it need not conflict with other hardware options.

The EMS (Expanded Memory Specification) provides a uniform means for applications to access expanded memory, as shown in the Fig. 1.25. The supporting software, which is called the **Expanded Memory Manager (EMM)**, provides a hardware - independent interface between application software and the expanded memory board (s). The EMM is supplied in the form of an installable device driver that we link into the DOS system by adding a line to the CONFIG.SYS file on the system boot disk.

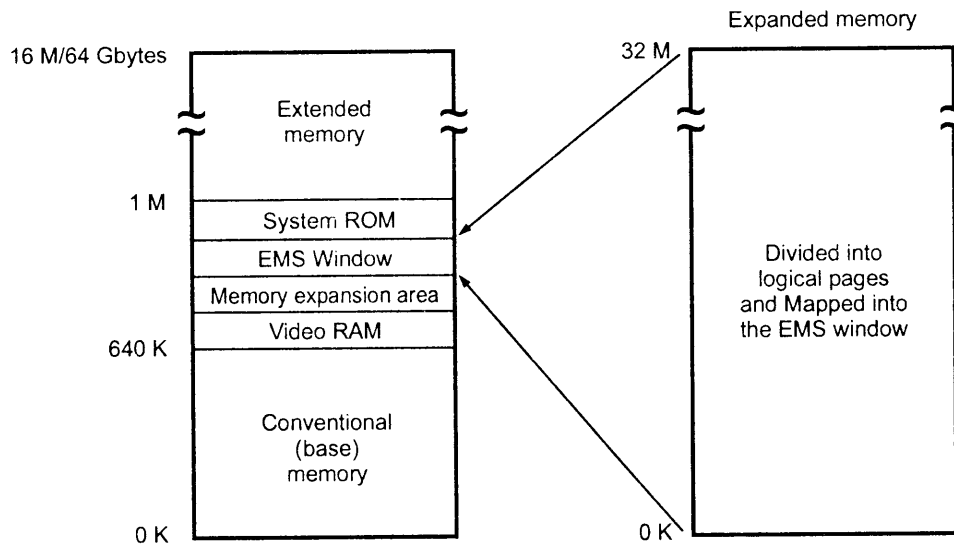


Fig. 1.25 Conventional, extended, and expanded memory map

Internally, the expanded memory manager consists of two major portions, which may be referred to as the **driver** and the **manager**. The driver mimics the actions of installable device driver and manager provides several services such as :

- Verification of functionality of hardware and software modules.
- Allocation of expanded - memory pages.
- Mapping of logical pages into the physical page frame.
- Deallocation of expanded memory pages.
- Support for multitasking operating system.

Note : Expanded memory should not be confused with extended memory. Extended memory is the term used by IBM to refer to the memory at physical addresses above 1 Mbyte that can be accessed by an 80286 and higher processors in protected mode. On the other hand expanded memory is not directly accessible by the processor. The EMS (Expanded Memory Specification) provides a uniform means for applications to access expanded memory.

1.3.2 I/O System

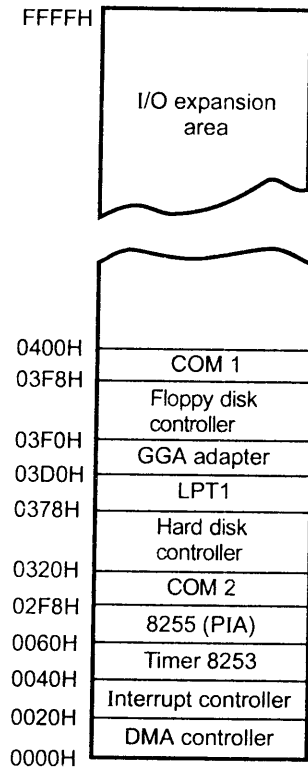


Fig. 1.26 I/O map

- I/O devices allow the microprocessor to communicate with outside world.
- I/O system allows the computer to access upto 64 K different 8-bit I/O devices.
- The Fig. 1.26 shows map of I/O space that ranges the address from 0000H to FFFFH.
- I/O space upto 0400H is reserved and above that is available for I/O expansion.
- Addresses between 0000H and 00FFH are used by the components on the main board.
- I/O space from 0100H to 03FFH is used for plug in cards.
- System operations related to I/O devices are not addressed directly. BIOS ROM does this job.
- Most I/O accessing is done through DOS or BIOS to maintain compatibility between computers.

1.3.3 Microprocessor

Microprocessor is the heart of microprocessor based computer system. It has an ability to perform simple arithmetic, logical and decision making operations on the data. For this purpose microprocessor has to read the data from memory or input/output devices. The data to be processed is temporarily stored into register array of microprocessor, register array contains a program counter register, which is used to hold the address of the next

instruction or data to be fetched from memory. Instruction decoder decodes the fetched instructions and generate the control signal required to execute the instructions. Arithmetic logic unit (ALU) performs the arithmetic operations such as addition, subtraction and logical operations such as AND, OR and exclusive OR. Result of these operations is stored into register that is also called accumulator. According to the result of operation, bits of the flag register either set or reset. This flag register plays an important role in the case of decision making process. Control unit controls all the data transfer between microprocessor and memory and I/O devices.

Internal structure of microprocessor is shown in Fig. 1.27.

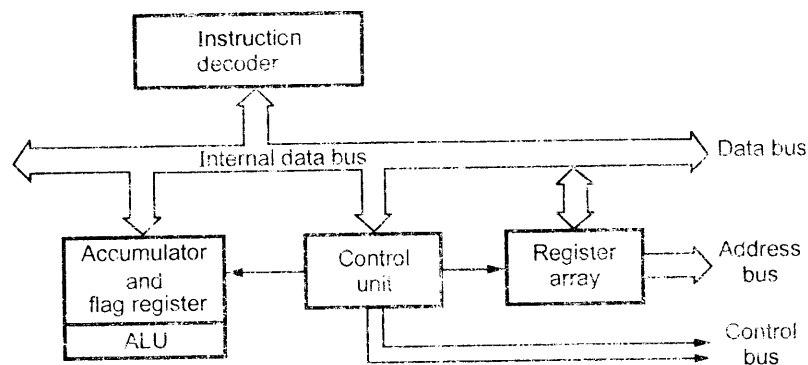


Fig. 1.27 Internal structure of microprocessor

Microprocessor is said to be powerful depending on its capacity to execute instructions per second and the data storing capacity. Data bits to be processed by the microprocessor depends on the length of the register of that microprocessor. 8086, 8085, 80286 can process only 8 or 16-bit data not 32-bit. 80386 and 80486 processors can directly manipulate 32-bits. The pentium processors can process 64-bit data. The microprocessors having built-in numeric coprocessor such as 80486 perform complex arithmetics.

Buses :

Bus is a set of conductors that interconnects the microprocessor to its memory and I/O devices. As shown in Fig. 1.28 buses are separated into three groups.

1) **Data bus :** The data bus consist of 8, 16, 32 or more parallel signal lines. These signal lines are also called **data bus width**. Data and address bus width depends on the type of microprocessor. Data bus width and address bus width for different microprocessors are given in Table 1.8.