DIPLOMA OF ASSOCIATE ENGINEER

3RD YEAR

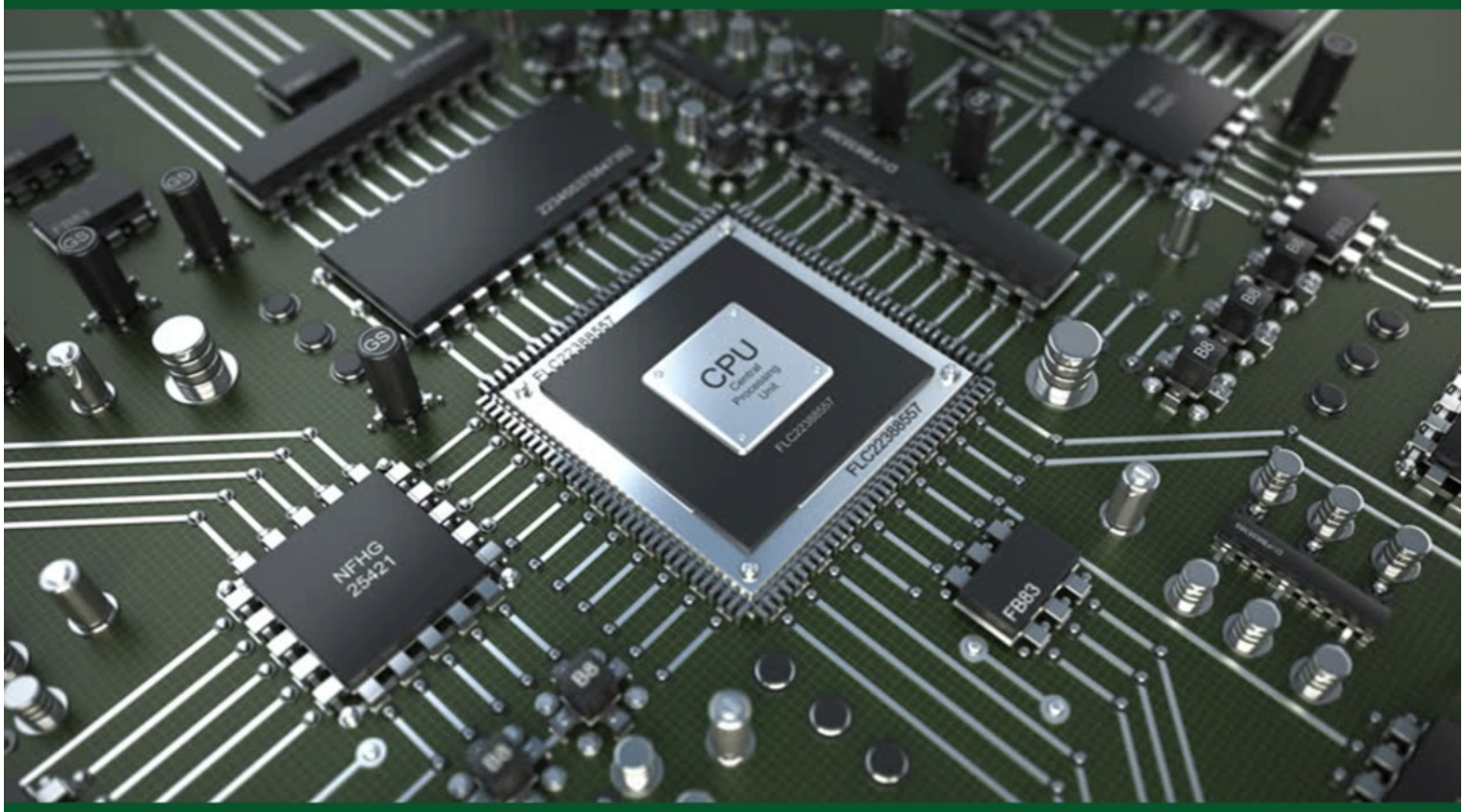
ELECTRONICS TECHNOLOGY

APPROVED BY TEVTA PUNJAB

A TEXT BOOK OF

# COMPUTER ARCHITECTURE

ELTR-314




Developed By
Academics Wing
Technical Education & Vocational Training Authority Punjab

# COMPUTER ARCHITECTURE

## ELTR-314

## For DAE Electronics 3rd YEAR

**TECHNICAL EDUCATION & VOCATIONAL TRAINING AUTHORITY PUNJAB**

# PREFACE

The text book has been written to cover the syllabus of Computer Architecture, 3rd year D.A.E (Electronics).

The aim of bringing out this book is to enable the students to have sound knowledge of the subject. Every aspect has been discussed to present the subject matter in the most concise, compact lucid & simple manner to help the subject without any difficulty. Frequent use of illustrative figures has been made for clarity. Multiple Choice Questions ,Short Questions & have also been included at the end of each chapter which will serve as a quick learning tool for students.

The author would like to thank the reviewers whose valuable recommendations have made the book more readable and understandable. Constructive criticisms and suggestions for the improvements in future are welcome.

**AUTHORS**

# MANUAL DEVELOPMENT COMMITTEE

**Engr. Shahbaz Hussain**

Chief Instructor Govt. Women Polytechnic Institute Lytton Road Lahore (CONVENER)

**Engr. Tauqeer Muhammad**

Instructor Electronics Govt. College of Technology, Bahawalpur

(MEMBER)

**Mst. Saba Wahab**

Jr. Instructor Govt. Women Polytechnic Institute Lytton Road Lahore

(Member)

**ELTR-314: COMPUTER ARCHITECTURE**

| T | P | C |
|---|---|---|
| 3 | 3 | 4 |

**Total contact hours:**

Theory: 96 Hours

Practical: 96 Hours

**Pre-requisite:** Electrical Circuits
Digital Electronics

**AIM.** This subject has been designed so as to enable the student:

1. Understand digital computation.
2. Understand microprocessor architecture, programming and interfacing.
3. Understand the microcomputer hardware.

**COURSE CONTENTS.**

1. Solve the digital problems related to computer.
2. Program an Intel 8086 microprocessor using assembly and machine language.
3. Debug and program.
4. Differentiate a 16 bit and 32 microprocessor.
5. Identify the function of standard interfaces.
6. Explain the memory organization.
7. Desc1ibe the stored program concept.
8. Identify the units of digital computer.
9. Describe computer peripheral devices.
10. Describe microcontroller's role.

**1. BASIC MICROPROCESSOR ARCHITECTURE. (16 Hours)**

- 1.1 Introduction of Microprocessor
- 1.2 Basic Architecture of a 16 bit Microprocessor (8086/88)
  - 1.1.1 Block Diagram
  - 1.2.2 Function of each Block
  - 1.2.3 Buses
    - 1.2.3.l Data Bus
    - l.2.3.2 Address Bus
    - l.2.3.3 Control Bus
  - 1.2.4 Bus Timing
  - 1.2.5 Pin Configuration of 8086/88
  - 1.2.6 Clocking and Power requirements.
  - 1.2.7 Bus Buffering and Latching
- 1.3 Processor Features
  - 1.3.1 Pipelining
  - 1.3.2 Multiprocessing.
  - 1.3.3 Parallel Processing.
- 1.4 Programming Models
  - 1.4.1 General Purpose Registers
  - 1.4.2 Segment register
  - 1.4.3 Pointer register
  - 1.4.4 Flag Register
- 1.5 Memory
  - 1.5.1 ROM
  - 1.5.2 Cache Memory
  - 1.5.3 RAM and its Types
    - 1.5.3.1 Static
    - 1.5.3.2 Dynamic
    - 1.5.3.3 Flash
- 1.6 Components of Processor

1.6.1 Control Unit
1.6.2 ALU

**2. ASSEMBLY LANGUAGE. (20 Hours)**

2.1 Instruction set
2.2 Programming language
2.3 Difference between Assembler, Interpreter and Compiler
2.4 Comparison between assembly language and machine language.
2.5 Data Addressing Modes.
2.6 Data Movement Instructions.
2.7 Arithmetic and Logic Instructions.
2.8 Program Control Instructions.

**3. MEMORY AND BASIC I/O INTERFACE (16 Hours)**

3.1 Memory Devices.
2.2 Address Decoding.
2.3 8088 and 80188 (8 bit) Memory Interface.
2.4 8086 and 80186 (16 bit) Memory Interface.
2.5 32 bit and 64 bit Memory Interface.
3.6 Dynamic RAM.
3.7 Introduction to I/O Interface.
3.8 I/O Port Address Decoding.
3.9 8254 Programmable Interval Timer.
3.10 Programmable Communication Interface
3.11 ADC and DAC Converters
3.12 Basic USB Interface
3.13 COM PUTER PERIPHERALS.
3.13.1 Introduction to computer peripherals.
3.13.2 key Board, VDU
3.13.3 Hard disk.
3.13.4 Pointer.
3.13.5 Mouse.
3.13.6 Plotter.
3.13.7 Digitizer and Scanner.

**4. INTERRUPTS. (04 Hours)**

4.1 Basic Interrupt Processing
4.2 Types of Interrupts.
4.3 Hardware Interrupt.
4.4 Software Interrupt.
4.5 Programmable Interrupt Controller (8259A).

**5. DIRECT MEMORY ACCESS. (04 Hours)**

5.1 Basic DMA Operation.
5.2 DMA Controller (8237).

**6. BUS INTERFACE. (05 Hours)**

6.1 The ISA Bus.
6.2 The Extended ISA (EISA) and VESA Local Buses.
6.3 The Peripheral Component Interconnect (PCI) Bus.
6.4 The Universal Serial Bus (USB).
6.5 Accelerated Graphics Port (AGP).

**7. THE PENTIUM MICROPROCESORS. (05 Hours)**

7.1 Summary of growth from 808186 to 80486.
7.2 Introduction to the Pentium Microprocessors.
7.3 Special Pentium Registers.
7.4 Pentium Memory Management.

7.5 New Pentium Instructions

**8.** THE MICROCONTROLLER. **(06 Hours)**

8.1 Single-chip Microprocessor.
8.2 Introduction to microcontrollers.
8.3 8051 internal RAM and registers.
8.4 8051 interrupts systems.
8.5 8051 instruction set.
8.6 Microcontrollers on the 8051 family.

**TEXT /REFERENCE BOOKS:**

1. Barry B. Brey ··The Intel Microprocessors (8086/8088, 80186. 80286. 80386. 80486)".

**ELTR-314: COMPUTER ARCHITECTURE**

**INSTRUCTIONAL OBJECTIVES.**

**1. UNDERSTANDING BASIC MICROPROCESSOR ARCHITECTURE.**

1.1 Understand the term of Microprocessor
1.2 Understand Basic Architecture of a 16 bit Microprocessor (8086/88)
1.2.1 Draw Block Diagram
1.2.2 Discuss Function of each Block
1.2.3 Discuss Buses
1.2.3.1 Describe Data Bus
1.2.3.2 Describe Address Bus
l.2.3.3 Describe Control Bus
1.2.4 Discuss Bus Timing
1.2.5 Discuss Pin Configuration of 8086/88
1.2.6 Discuss Clocking and Power requirements.
1.2.5 Describe Bus Buffering and Latching
1. 3 Understand Processor features
1.3.1 Describe Pipelining
1.3.2 Describe Multiprocessing.
1.3.3 Describe Parallel Processing.
1.4 Understand Programming Models
1.4. l Describe General Purpose Registers
1.4.2 Describe Segment register
1.4.3 Describe Pointer register
1.4.4 Describe Flag Register
1.5 Understand Memory
1.5.1 Discuss ROM
1.5.2 Describe Cache Memory
1.5.3 Describe RAM and its types
1.5.3.1 Describe Static
1.5.3.2 Describe Dynamic
1.5.3.3 Describe Flash
1.6 Understand Components of Processor
1.6.1 Discuss Control Unit
1.6.2 Discuss ALU

**2. UNDERSTAND THE APPLICATIONS OF ASSEMBLY LANGUAUE.**

2.1 Discuss the Instruction set.
2.2 Discuss Programming languages.
2.3 List three programming languages for a microcomputer.
2.4 Differentiate between Assembler, Interpreter and Compiler
2.5 Discuss fields of assembly language statement.
2.6 Compare assembly language with machine language.
2.7 Describe Data Addressing Modes.

2.8 Apply Data Movement Instructions.
2.9 Apply Arithmetic and Logic Instructions.
2.10 Apply Program Control Instructions.

**3. UNDERSTAND THE USE OF MEMORY AND BASIC I/O INTERFACE.**

3.1 Discuss Memory Devices.
3.2 Describe Address Decoding.
3.3 Discuss 8088 and 80188 (8 bit) Memory Interface.
3.4 Discuss 8086, 80186 (16 bit) Memory Interface.
3.5 Discuss 32 bit and 64 bit Memory Interface.
3.6 Discuss Dynamic construction and working of RAM
3.7 Describe Introduction to I/O Interface.
3.8 Describe I/O Port Address Decoding.
3.9 Discuss the construction and operation of 8254 Programmable Interval Timer.
3.10 Discuss the construction and operation of Programmable Communications Interface
3.11 Describe the operation of ADC and DAC Converters
3.12 Describe the operation of USB Interface
3.13 Describe the role of peripherals in computer system.
3.14 Discuss construction and working of key Board, VDU
3.15 Discuss construction and working of Hard disk.
3.16 Discuss construction and working of Pointer.
3.17 Discuss construction and working of Mouse.
3.18 Discuss construction and working of Plotter.
3.19 Discuss construction and working of Digitizer and Scanner.

**4. UNDERSTAND INTERRUPTS.**

4.1 Discuss basic Interrupt Processing
4.2 Discuss Types of Interrupts.
4 .3 Describe Hardware Interrupts.
4.4 Describe Software Interrupts.
4.5 Discuss operation of Programmable Interrupt Controller (8295A).

**5. UNDERSTAND THE CONCEPT OF DIRECT MEMORY ACCESS.**

5.1 Discuss basic DMA Operation.
5.2 Discuss the construction and operation of DMA Controller (8237).
5.3 Describe Shared Bus Operation.

**6. UNDERSTAND THE BUS INTERFACE.**

6.1 Describe the operation of ISA Bus.
6.2 Discuss the Extended ISA (EISA) and VESA Local Buses.
6.3 Describe the role of Peripheral Component Interconnect (PCI) Bus.
6.4 Describe the role of the Universal Serial Bus (USB)
6.5 Describe the role of Accelerated Graphics Port (AGP).

**7. THE PENTIUM MICROPROCESORS.**

7.1 Discuss summary of growth from 808186 to 80486.
7.2 Describe introduction to the Pentium Microprocessors.
7.3 Describe Special Pentium Registers.
7.4 Discuss Pentium Memory Management.
7.5 Describe New Pentium Instructions.

**8. THE MICROCONTROLLER**

8.1 Define Single-chip Microprocessor.
8.2 Express microcontrollers.
8.3 Describe the role of 8051 internal RAM and registers.
8.4 Describe the role of 8051 interrupts systems.
8.5 Discuss 8051 instruction set.
8.6 Discuss Microcontrollers on the 8051 family.

# CONTENTS

**PUNJAB BOARD OF TECHNICAL EDUCATION 21-A,**
**Kashmir Block Allama Iqbal Town, Lahore** www.pbte.edu.pk- Tel #
37800279, 37800088-89 (Ext. 139) R&D Cell Tel # 37800025

Ref . No. PBTE/R&D/2013/505 Dated:- 23-02-2013

To

All Concerned Principals, Govt.
College of Technology /
Affiliated Polytechnic Institutions with PBTE
Lahore.

**Subject**: **Bifurcation/Division of the marks of the subject Computer Architecture (ELTR-314) DAE Electronics 3rd Year Revised Course**

The bifurcation / division of the subject "Computer Architecture (ELTR-314) of DAE Electronics Technology (Revised 3r d year) has been made and will be available at PBTE website i.e. www.pbte.edu.pk (News & Updates section) for the guidance of teachers and students for the DAE Examination 2013 and onward.

| Sr. # | Subject Name | Paper "A" Course Contents /Topics Marks = 75 | | Paper "B" Course Contents / Topics Marks = 75 | |
|---|---|---|---|---|---|
| 1 | Computer Architecture (ELTR 314) | 1. | Basic Microprocessor Architecture | 2. | Assembly Language |
| | | 3. | Memory and Basic I/O Interface. | 5. | Direct Memory Access |
| | | 4. | Interrupts | 6. | Bus Interface |
| | | | | 7. | The Pentium Microprocessors |
| | | | | 8. | The Microcontrolle |

# CHAPTER 01
# Basic Microprocessor Architecture

## 1.1 Understand the term Microprocessor

"Microprocessor is a VLSI device which performs arithmetic and logic operations under various instructions." First microprocessor was 4004 (4-bit) which was launched by Intel Corporation in 1971. Now a days Core i3, Core i5 and Core i7 are modern microprocessors.

### Block Diagram of Microprocessor

There are three parts of a microprocessor as shown in fig. 1.1

1. Control Unit
2. Arithmetic Logic Unit
3. Register Unit (Memory Unit)

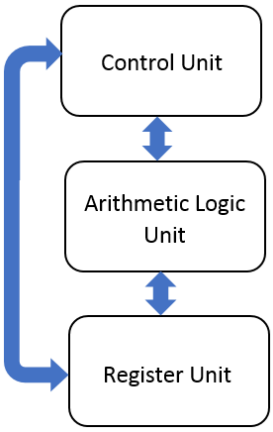



Fig. 1.1

### Control Unit:

This unit generates timing and control signals to control data flow between microprocessor and peripheral devices. These signals are according to instructions.

### Register Unit:

This unit contains various registers which stores data temporary. It also stores temporary the results of instructions. This unit is also called memory unit.

### Arithmetic Logic Unit:

ALU performs processing of arithmetic (addition, subtraction, multiplication, division) and logic (NOT, OR, AND, XOR) operations. The results of this unit are either stored in a register or sent to a peripheral device.

These all units are connected to each other through group of wires called buses.

### Word Length and Data Path Size of a Microprocessor

The maximum number of bits processed by a processor is called word length of that processor. The number of data lines of a microprocessor is

called data path size of that processor. Data path size depends on data bus of microprocessor.

## Steps of Microprocessor to Execute an Instruction

There are following four steps of a microprocessor to execute an instruction.

(i) Fetching (ii) Read Operand (iii) Execution (iv) Store result

## HISTORY OF MICROPROCESSOR

**The 4004 Microprocessor:**

The world's first microprocessor, the Intel 4004, was a 4-bit microprocessor introduced in 1971. It addressed a mere 4,096 4-bit wide memory locations. The Intel 4004 instruction set contained only 45 instructions.

**The 8008 Microprocessor:**

In the same year, Intel Corporation released the 8008-an extended 8-bit version of the Intel 4004 microprocessor. The 8008 addressed an expanded memory size (16K bytes) and contained 48 instructions.

**The 8080 Microprocessor:**

In 1973, Intel introduced the 8080 microprocessor-the first of the modern 8-bit microprocessors. Not only could the 8080 address more memory and execute additional instructions, but it executed them 10 times faster than the 8008. An instruction that took 20 µs on an 8008-based system required only 2.0 µs.

**The 8085 Microprocessor:**

In 1977, Intel Corporation introduced an updated version of the 8080--the 8085.This was to be the last 8-bit general-purpose microprocessor developed by Intel. 8085 required only 1.3 µs per instruction. The main advantages of the 8085 were its internal clock generator, internal system controller, and higher clock frequency.

**The 8086/8088 Microprocessors:**

In 1978, Intel released the 8086 microprocessor, a year or so later, it released the 8088. Both devices were 16-bit microprocessors, which executed instructions in as little as 400 ns (2.5 MIPS). In addition, the 8086 and 8088 addressed 1M bytes of memory. One other feature found in the 8086/8088 was a small 4 or 6-byte instruction cache or queue that prefetches a few instructions before they are executed.

**The 80286 Microprocessor:**

The 80286 microprocessor (also a 16-bit architecture microprocessor) was almost identical to the 8086 and 8088 except it addressed a 16M byte memory system. The instruction set of the 80286 was also almost identical to the 8086 and 8088 except for a few instructions. The clock speed of the 80286 was increased, so it executed some instructions in as little as 250 ns (4.0 MIPS).

**The 80386 Microprocessor:**

The 80386 was Intel's first practical 32-bit microprocessor that contained a 32-bit data bus and a 32-bit memory address. Through these 32-bit buses, the 80386 addressed up to 4G bytes of memory. The instruction set of the 80386 microprocessor was upward compatible with the earlier 8086, 8088, and 80286 microprocessors.

**The 80486 Microprocessor:**

In 1989, Intel released the 80486 microprocessor, which incorporated an 80386-like microprocessor, an 80387-like numeric coprocessor, and an 8K byte cache memory system into one integrated package. The internal structure of the 80486 was modified from the 80386 so about half of its instructions executed in 25ns (50 MIPS).

**The Pentium Microprocessor:**

The Pentium, introduced in 1993, was similar to the 80386 and 80486 microprocessors. This microprocessor was originally labeled the P5 or 80586. The two introductory versions of the Pentium operated with a clocking frequency of 60 MHz and 66 MHz and a speed of 110 MIPS. Another difference was that the cache size was increased to 16K bytes. The Pentium contains an 8K byte instruction cache and an 8K byte data cache. The memory system contained up to 4G bytes, with the data bus width increased from the 32-bits found in the 80386 and 80486 to full 64-bits.

# 1.2 Basic Architecture of a 16 Bit Microprocessor (8086/8088)

## Block Diagram of 8086/8088 Microprocessor:

The block diagram of 8086/8088 microprocessors is shown in fig. 1.2. Both microprocessors have same structure except the data bus and instruction queue register. 8086 has 16 bits wide data bus, whereas 8088 has 8 bits data bus. The instruction queue of 8086 is 6 bytes and 8088 has 4 bytes

instruction queue. There are two main functional parts of 8086/8088 microprocessor architecture; Bus Interface Unit and other is Execution Unit.

## 1.2.2 Function of each Block

The figure shows that it has two functional blocks i.e.

i. Bus Interface Unit
ii. Execution Unit

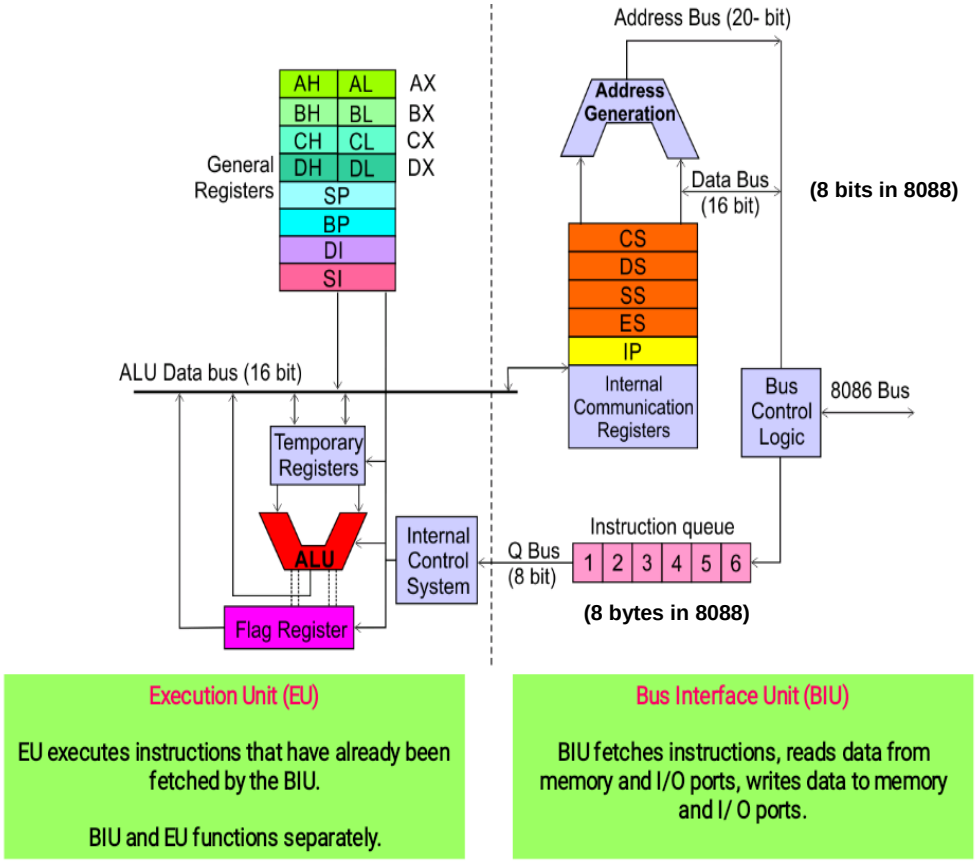



Fig. 1.2

### Bus Interface Unit

The purpose of bus interface unit is: -

i- To keep Instruction Queue filled with instructions.
ii- To generate and accept the system control signals.
iii- To provide the system with address or I/O port number.
iv- To act a window between EU (Execution Unit) and Memory for data.

Following are the parts of bus interface unit: -

i- Instruction Pointer
ii- Segment Registers
iii- Address Summing Unit

iv- Instruction Queue / Prefetch Queue

**Instruction Pointer**

It is a 16-bit pointer which points the next instruction to be executed. It contains a 16-bit offset address of instruction that is to be executed next.

**Segment Registers**

There are four segment registers in 8086/88 microprocessor. The purpose of segment registers is to store the base address of 64KB memory segment.

**Address Summing Unit**

To form a 20-bit address of the next instruction, the 16-bit address of the IP is added by the address summing block to the address contained in the CS, which has been shifted four bits to the left.

**Instruction Queue / Prefetch Queue**

When EU is executing or decoding an instruction, buses are not use in EU. In this period, BIU uses these buses to fetch instructions from memory. These pre-fetched instructions are stored in a FIFO register. This register is called pre-fetch queue. EU receives instructions from Instruction Queue instead of memory. The size of Instruction Queue register is 4 bytes in 8088 microprocessor and 6 bytes in 8086 microprocessor. During the execution of an instruction, fetching of next instruction of BIU is called “Pipelining”.

During the execution of an instruction, BIU fetches next instruction. Two operations are occurred at a time. This process is called overlapped fetch/execute cycle.

## EXECUTION UNIT

The purpose of execution unit is: -

i- To carry out the instructions from prefetch queue.
ii- To decode the instructions.
iii- To execute the instructions.

Following are the parts of execution unit: -

i- General purpose registers
ii- Pointer and Index Registers
iii- Flag Register
iv- Instruction Decoder
v- Arithmetic Logic Unit
vi- Control Unit

# 1.2.3 Buses

A bus is a group of wires which connects the parts of the CPU to one another. It is also a connection between devices connected to a computer. For example, a bus carries data between a CPU and the system memory or I/O devices. There are three types of buses. Address, Data & Control bus.

**Address Bus**

The group of electrical wires through which a microprocessor sent addresses signals to select different ports or memory is called address bus. It is a unidirectional bus.

**Data Bus**

The group of electrical wires through which a microprocessor sent or receives data from one section to another section is called data bus. It is a bidirectional bus.

**Control Bus**

The group of electrical wires through which a microprocessor generates timing and control signals to control data flow between microprocessor and peripheral devices. These signals are according to instructions.

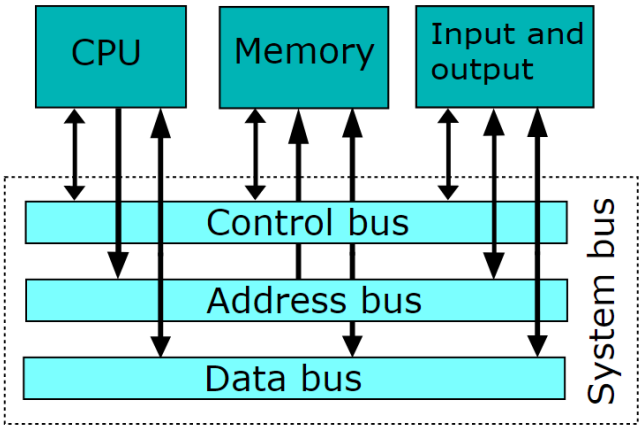



Fig. 1.3

## 1.2.4 Bus Timings

The 8086/8088 microprocessors use the memory and I/O in periods called bus cycles. Each bus cycle equals four system-clocking periods (T states). There are two types of bus cycles.

### Memory Read Cycle

**During T1 State of Clock Signal:**

1. ALE is asserted HIGH which activates the latches and the address is put on the buses.
2. DT/$\overline{R}$ is asserted LOW for selecting the data direction.
3. M/$\overline{IO}$ is asserted HIGH to indicate memory operation.
4. $\overline{BHE}$ is made LOW for selecting a memory bank.

**During T2 State of Clock Signal:**

1. Address lines are in active in this state and reserved for carrying the data.
2. $\overline{RD}$ goes LOW at the end of T2 for memory read operation.
3. $\overline{DEN}$ goes LOW for data reception.
4. 8086 samples the READY signal and if it is HIGH, T3 & T4 are executed; otherwise TW is inserted after T3.

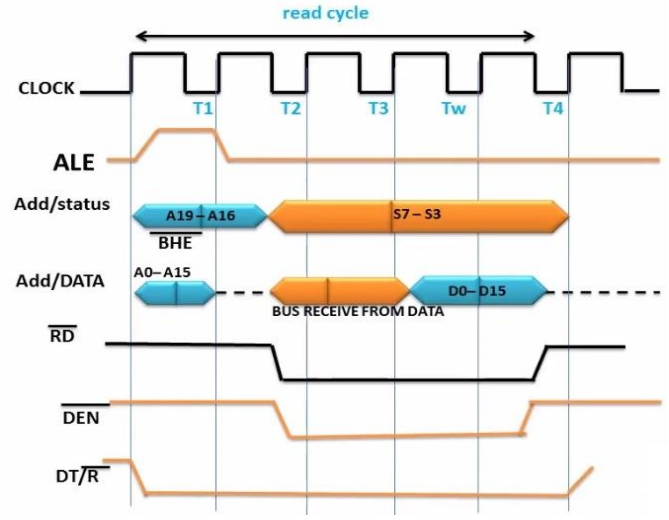



Fig. 1.4

**During T3 State of Clock Signal:**

Data is put on the data bus.

1. Status signals will remain on the upper address line A19-A16

**During T4 State of Clock Signal:**

1. $\overline{RD}$ goes HIGH to indicate the end of read operation.
2. $\overline{DEN}$ goes HIGH to disable the transceiver.

**Summary of Read Cycle**

When a read cycle is to be performed, during T1 microprocessor puts an address on address bus, and then bus is put in high impedance state during T2 state. Data to be read must be out on bus during T3 and T4. During T3 bus is made “reserved for data in” and finally data is read during T4.

## Memory Write Cycle

**During T1 State of Clock Signal:**

1. ALE is asserted HIGH which activates the latches and the address is put on the buses.
2. DT/$\overline{R}$ is asserted HIGH for selecting the data direction.
3. M/$\overline{IO}$ is asserted HIGH to indicate memory operation.
4. $\overline{BHE}$ is made LOW for selecting a memory bank.

**During T2 State of Clock Signal:**

1. Address lines are in active in this state and reserved for carrying the data.
2. $\overline{WR}$ goes LOW at the mid of T2 for memory write operation.
3. $\overline{DEN}$ goes LOW for data reception.
4. 8086 samples the READY signal and if it is HIGH, T3 & T4 are executed; otherwise TW is inserted after T3.

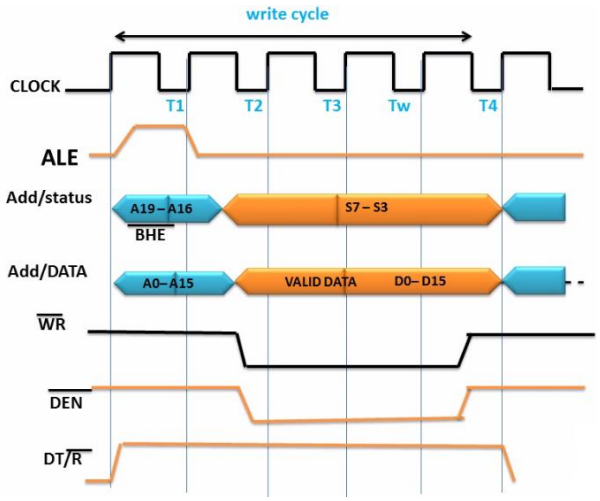



Fig. 1.5

**During T3 State of Clock Signal:**

1. Data is written to the memory through the data bus.
2. Status signals will remain on the upper address line A19-A16

**During T4 State of Clock Signal:**

$\overline{WR}$ goes HIGH to indicate the end of write operation.

1. $\overline{DEN}$ goes HIGH to disable the transceiver.

**Summary of Write Cycle**

In case of write memory cycle, during T1 state microprocessor puts an address on address bus. Data is put on data bus by CPU during T2 state and maintained during T3 and T4 states that are written out to memory or I/O devices.

### I/O Read Write Cycles

The I/O read write cycles also work similar to memory read write cycles except that the signal $M/\overline{IO}$ goes LOW to indicate the I/O related operation on the bus. Other signals remain same as in memory read write cycles.

## 1.2.5 Pin Configuration of 8086/88 Microprocessor

Both 8086 and 8088 microprocessors ICs are packaged in 40-pin dual in-line packages (DIPs). The 8086 has pin connections AD0- AD15, and the 8088 has pin connections AD0- AD7. Data bus width is therefore the only major difference between these microprocessors.

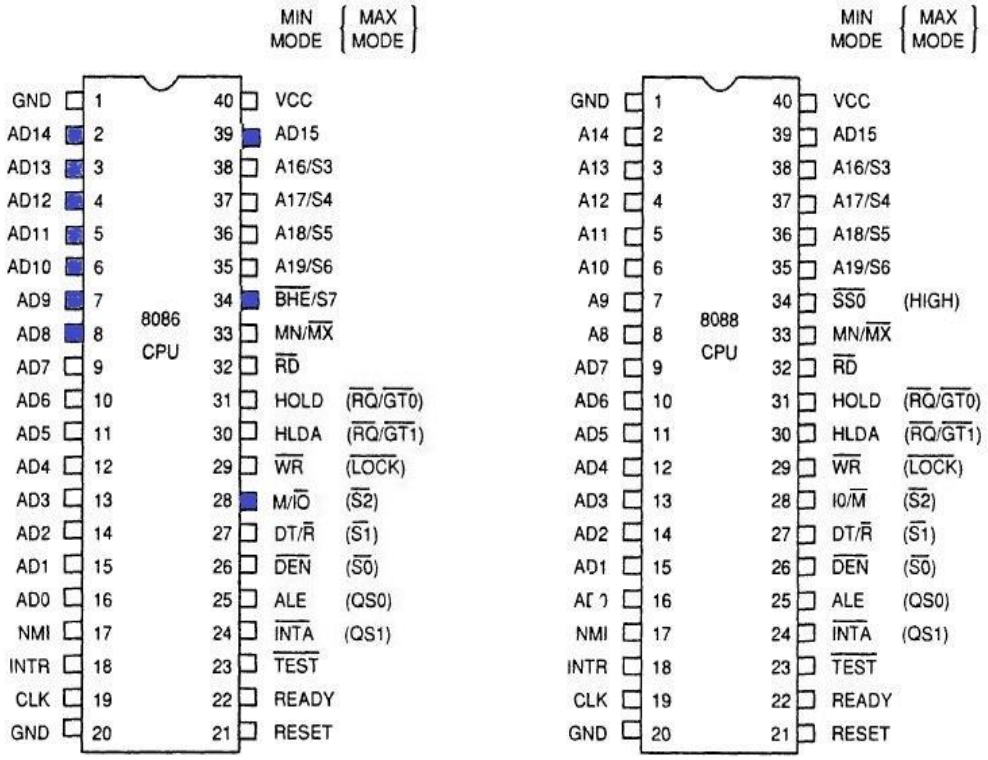



Fig.1.6

There is, however, a minor difference in one of the control signals. The 8086 has an $M/\overline{IO}$ pin, and the 8088 has an $IO/\overline{M}$ pin. The only other hardware difference appears on Pin 34 of both chips: on the 8088, it is an SSO pin, while on the 8086, it is a $\overline{BHE}/S7$ pin.

There is a detail of pin configuration of 8086 microprocessor as under.

**$V_{CC}$ and GND**

It uses 5V DC supply at $V_{CC}$ pin 40, and uses ground at $V_{SS}$ pin 1 and 20 for its operation.

**Clock signal**

It provides timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.

**Address/Data bus**

AD0-AD15. These are 16 address/data bus. $AD_0$-$AD_7$ carries low order byte data and $AD_8$-$AD_{15}$ carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it carries 16-bit data.

**Address/Status bus**

$A_{16}$-$A_{19}$ / $S_3$-$S_6$. These are the 4 address/status buses. During the first clock cycle, it carries 4-bit address and later it carries status signals.

**$\overline{BHE}$/S7**

BHE stands for Bus High Enable. It is used to indicate the transfer of data using data bus $D_8$-$D_{15}$. This signal is low during the first clock cycle, thereafter it is active.

**$\overline{RD}$**

It is used to read signal for Read operation.

**Ready**

It is an acknowledgement signal from I/O devices that data is transferred. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.

**RESET**

It is used to restart the execution. It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.

**INTR**

It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not.

**NMI**

It stands for non-maskable interrupt. It is an edge triggered input, which causes an interrupt request to the microprocessor.

**$\overline{\text{TEST}}$**

This signal is like wait state. When this signal is high, then the processor has to wait for IDLE state, else the execution continues.

**MN/$\overline{\text{MX}}$**

It stands for Minimum/Maximum. It indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-versa.

**$\overline{\text{INTA}}$**

It is an interrupt acknowledgement signal. When the microprocessor receives this signal, it acknowledges the interrupt.

**ALE**

It stands for address latch enable. This signal indicates the availability of a valid address on the address/data lines.

**DEN**

It stands for data enable. It is used to enable transceiver 8286. The transceiver is a device used to separate data from the address/data bus.

**DT/$\overline{\text{R}}$**

It stands for Data transmit/receive signal. It decides the direction of data flow through the transceiver. When it is high, data is transmitted out and vice versa.

**M/$\overline{\text{IO}}$**

This signal is used to distinguish between memory and I/O operations. When it is high, it indicates the memory operation and low indicates I/O operation.

**$\overline{\text{WR}}$**

It stands for write signal. It is used to write the data into the memory or the output device depending on the status of $\text{M}/\overline{\text{IO}}$ signal.

**HLDA**

It stands for Hold Acknowledgement signal. This signal acknowledges the HOLD signal.

**HOLD**

This signal indicates to the processor that external devices are requesting to access the address/data buses.

**$QS_1$ and $QS_0$**

These are queue status signals. These signals provide the status of instruction queue.

**$S_0$, $S_1$, $S_2$**

These are the status signals that provide the status of operation, which is used by the Bus Controller 8288 to generate memory & I/O control signals.

**LOCK**

When this signal is active, it indicates to the other processors not to ask the CPU to leave the system bus.

**$RQ/GT_1$ and $RQ/GT_0$**

These are the Request/Grant signals used by the other processors requesting the CPU to release the system bus. When the signal is received by CPU, then it sends acknowledgment. $RQ/GT_0$ has a higher priority than $RQ/GT_1$.

## 1.2.6 Clocking and Power Requirement of 8086/88 Microprocessor

**Clocking Requirement**

A special clock generator 8284A provides clock timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.

The clock signal must have a duty cycle of 33% (high for one-third of the clocking period and low for two-thirds) to provide proper internal timing for the 8086/8088.

**Power Requirement**

Both the 8086 and 8088 microprocessors require +5.0 V with a supply voltage tolerance of ±10 percent. The 8086 draws a maximum supply current of 360 mA, and the 8088 draws a maximum of 340 mA. Both microprocessors operate in ambient temperatures of between 32°F and about 180°F. The 80C88 and 80C86 are CMOS versions that require only 10 mA of power supply current and function in temperature extremes of-40°F through +225°F.

## 1.2.7 Bus Buffering and Latching

All computer systems have three buses: an address bus, a data bus, and a control bus. These buses must be present in order to interface to memory and I/O. The address/data bus of 8086/88 is multiplexed to reduce no of pins. Memory and I/O require that address remains valid and stable

throughout a read and write cycle. So, 8086/88 system requires separate address, data, and control buses.

Latches are used to de-multiplex the address/data and address/status lines and commonly have output buffers for driving external loads. Buffers are used to drive external loads, and to isolate component when disabled.

**Demultiplexing the 8086:**

In 8086 A19/S6–A16/S3, AD15–AD0 and $\overline{\text{BHE}}$/S7 are multiplexed. Fig. 1.7 illustrates a demultiplexed 8086 with all three buses.

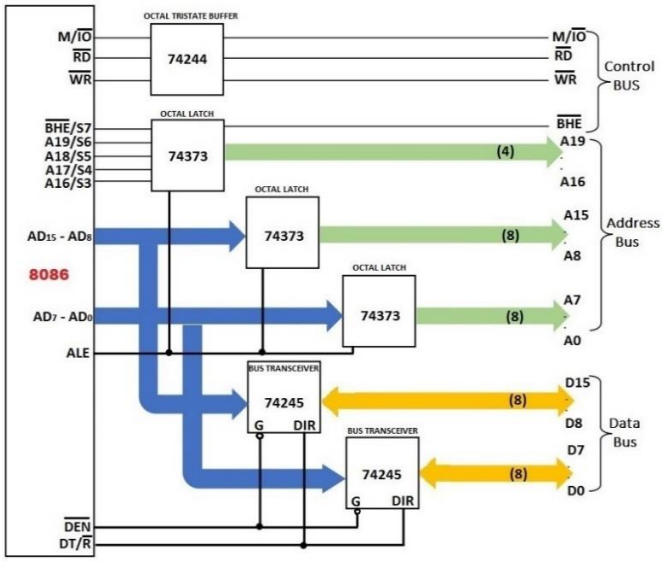



Fig. 1.7

Three 74LS373 latches have been used to demultiplex the address/data bus and address/status pins. Latches pass inputs to outputs whenever ALE become HIGH. The memory and I/O system see the 8086 as a device with a 20-bit address bus (A19– A0), a 16-bit data bus (D15–D0), and a three-line control bus (M/$\overline{\text{IO}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$).

**Demultiplexing the 8088:**

In the 8088, only AD7–AD0 and A19/S6–A16/S3 are multiplexed. Fig. 1.8 illustrates a demultiplexed 8088 with all three buses. Two 74LS373 latches have been used to demultiplex the address/data bus and address/status pins.

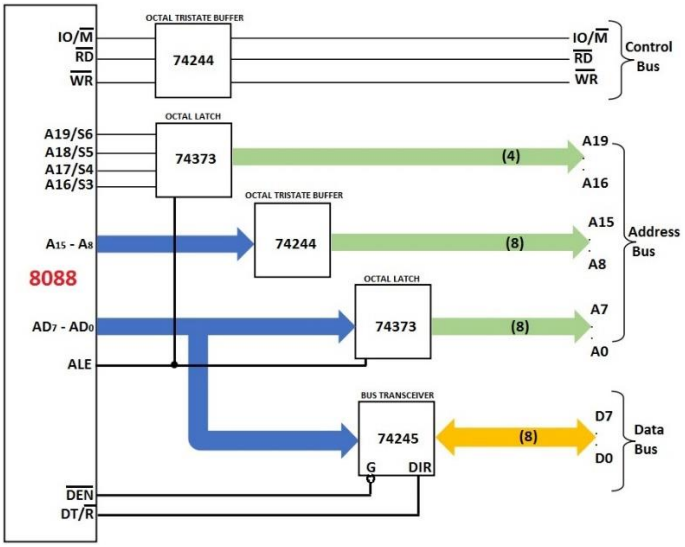



Fig. 1.8

**Buffering**

If more than 10 units loads are attached to any bus pin, the entire 8086 or 8088 system must be buffered using 74244. Demultiplexed pins are already buffered by 74LS373 latch.

# 1.3 Microprocessor Features

## 1.3.1 Pipelining

8086/88 contains two internal units, the bus interface unit and the execution unit. The BIU is responsible for fetching and instruction or operand from the memory, and the EU is responsible for execution the instructions.

A separate BIU and EU allow the fetch and execute cycles to overlap. Initially, BIU fetches the instruction from memory and stores it in a FIFO register called instruction queue. EU draws this instruction from instruction queue and begins execution.

While the EU is executing this instruction, the BIU proceeds to fetch a new instruction. During the execution of an instruction, fetching of next instruction of BIU is called pipelining. The BIU may fill the queue with several new instructions before the EU is ready to draw its next instruction. The advantage of this technique is that the EU can execute instructions continually instead of having to wait for the BIU to fetch the new instruction.

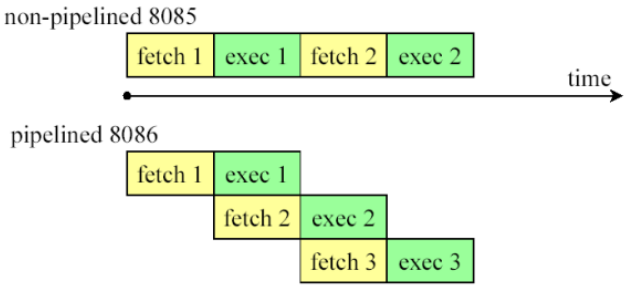



Fig. 1.9

### *1.3.2* Multitasking

Multitasking is the running of multiple programs in one computer at the same time. It is controlled by the operating system, which loads programs into the computer for processing and oversees their execution until they are finished. For example, when you open your Web browser and then open Word at the same time, you are causing the operating system to do multitasking.

### *1.3.3 Parallel Processing*

Parallel processing is a method in computing of running two or more processors to handle separate parts of an overall task. This technique helps to reduce the amount of time to run a program. Any system that has more than one processor can perform parallel processing, as well as multi-core processors which are commonly found on computers today.

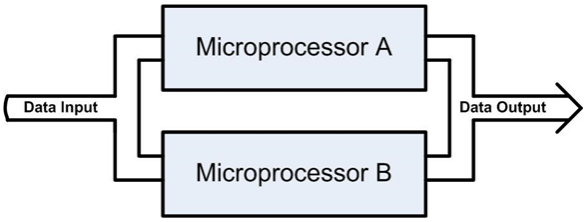



Fig. 1.9

Typically, a computer scientist divides a task into multiple parts with a software tool and assigns each part to a processor, then each processor will solve its part, and the data is reassembled by a software tool to read the solution or execute the task.

## 1.4 Understand the Programming Models

The programming model of 8080/8088 microprocessor is shown in fig. 1.10:

### 1.4.1 General Purpose Registers

There are four 16-bit general purpose registers in 8086/88 microprocessor. These registers are used in any manner that the programmer wishes. Some instructions used these registers for specific tasks. For this reason, each is also given a name.

**AX (Accumulator):** Preferred register to use in arithmetic, logic and data transfer instructions.

**BX (Base):** It serves as an address register.

**CX (Count)**: It is used as a loop counter and in shift and rotate operations.

**DX (Data):** It is used in multiplication and division. Also used in I/O operations.

General registers can be used as 8-bit registers namely AH, AL, BH, BL, CH, CL, DH and DL. The H represents the high- order byte and the L represents the low-order byte.

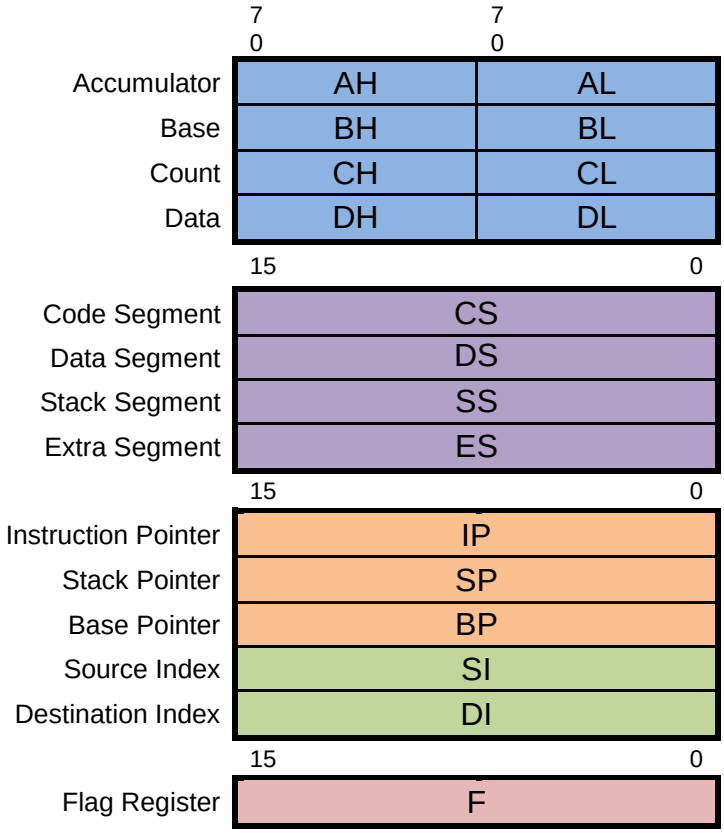



Fig. 1.10

## 1.4.2 Segment Registers

Segmentation is the process in which the main memory of the computer is logically divided into different segments and each segment has its own base address. It is basically used to enhance the speed of execution of the computer system, so that the processor is able to fetch and execute the data from the memory easily and fast. There are four portions in 8086/88 1MB memory. Each portion is called memory segment and consists of 64K bytes.

(i) Code Segment (ii) Data Segment (iii) Extra Segment and (iv) Stack Segment.

There are four segment registers in 8086/88 microprocessor.

**Code Segment Register:** It contains most left 16-bits of starting address of code segment. The contents of IP are added in code segment register × 10H to generate 20-bit address.

**Data Segment Register:** It contains most left 16-bits of starting address of data segment. The contents of BX, SI or DI are added in data segment register × 10H to generate 20-bit address.

**Extra Segment Register:** It contains most left 16-bits of starting address of extra segment. The contents of SI or DI are added in extra segment register × 10H to generate 20-bit address.

**Stack Segment Register:** It contains most left 16-bits of starting address of stack segment. The contents of SP or BP are added in stack segment register × 10H to generate 20-bit address.

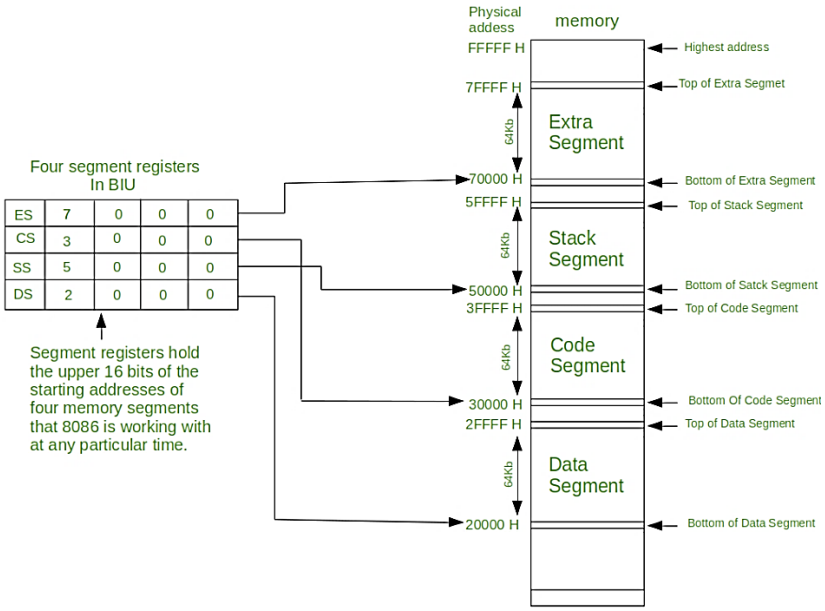



Fig. 1.11

There are four portions in 8086/88 1MB memory. Each portion is called memory segment and consists of 64K bytes.

i. Code Segment
ii. Data Segment
iii. Extra Segment and
iv. Stack Segment.

There are four segment registers in 8086/88 microprocessor.

i. CS
ii. DS
iii. ES
iv. SS

Each segment register contains most left 16-bits of starting address of a related memory segment. For example, if starting address of Code Segment is 10000H, then 1000H is stored in CS register.

To generate 20-bit address, the contents of Pointer/Index/Base register is added in segment register × 10H

For Example:

IP = A0B0H CS = 1000H

20-bit address ➡ CS × 10H = 10000H ➡ CS × 10H + IP = 1A0B0H

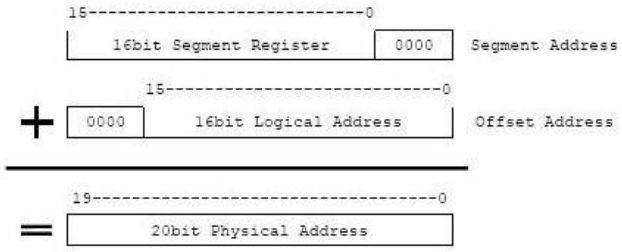



Fig. 1.12

The contents of Pointer/Index/Base register are called offset address and the contents of Segment registers are called base address.

20-bit address generated by address summing block is called effective address or physical address.

Effective address of CS = IP + CS × 10H

Effective address of DS = [BX, SI or DI] + DS × 10H

Effective address of SS = [SP or BP] + SS × 10H

Effective address of ES = [DI or SI] + ES × 10H

## 1.4.3 Pointer & Index Registers

Pointer and Index registers are used to point or index to the memory. All are 16 bits wide; L/H bytes are not accessible.

**Instruction Pointer (IP):** It points the next instruction to be executed.

**Stack Pointer (SP):** It is used to address data in a LIFO stack memory.

**Base Pointer (BP):** It is general pointer often used to point an array of data in the stack memory.

**Source Index (SI):** It is used to address source data during string instruction execution.

**Destination Index (DI):** It is used to address destination data during string instruction execution.

## 1.4.4. Flag Register

Flag register indicates the condition about the operation of Arithmetic Logic Unit. It is also called Status register. It is a 16-bit register which contains 9 flags. The right most 8 bits contains 5 flags (C, P, A, Z, and S) and left most 8 bits contains 4 flags (T, I, D, and O).

| | | | | O | D | I | T | S | Z | | A | | P | | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**Carry Flag**

Carry Flag indicates an arithmetic carry or borrow has been generated out of the most significant ALU bit position.

C = 1 (If a carry or borrow is generated in result)

C = 0 (If a carry or borrow is not generated in result)

**Parity Flag**

Parity flag indicates if the number of set bits is odd or even in the binary representation of the result of the last operation.

P = 1 (If number of set bits are even in result)

P = 0 (If number of set bits are odd in result)

**Auxiliary Carry Flag**

Auxiliary Carry indicates when an arithmetic carry or borrow has been generated out of the 4 least significant bits.

A = 1 (If there is a carry or borrow between low nibble and high nibble of the low order 8-bit of a 16-bit number)
A = 0 (If there is no carry or borrow between low nibble and high nibble of the low order 8-bit of a 16-bit number)

**Zero Flag**

Zero flag indicates if the result of arithmetic or logic operation is zero or non-zero.
Z = 0 (If result is non-zero)
Z = 1 (If result is zero)

**Sign Flag**

Sign Flag indicates whether the result of the last mathematical operation is positive or negative.
S = 0 (If result is positive)
S = 1 (If result is negative)

**Overflow Flag**

Overflow Flag indicates that the signed two's-complement result would not fit in the ALU width.
O = 0 (If result is in the capacity of processor)
O = 1 (If result is not in the capacity of processor)

**Trap Flag**

Trap flag permits operation of a processor in single-step mode.
T = 0 (Processor is in normal mode)
T = 1 (Processor is in single step mode)

**Interrupt Flag**

Interrupt Flag determines whether or not the CPU will handle maskable hardware interrupts.
I = 0 (Maskable hardware interrupts will be ignored.)
I = 1 (Maskable hardware interrupts will be handled.)

**Direction Flag**

Direction Flag controls the direction of string processing.
I = 0 (SI and DI are in auto-incrementing mode in string operations.)
I = 1 (SI and DI are in auto-decrementing mode in string operations.)

Trap, Interrupt and Direction flags are called control flags because these flags can be controlled by the programmer.

# 1.5 Memory

## 1.5.1 Read Only Memory (ROM)

A ROM contains permanently or semi-permanently stored data, which can be read from the memory but either cannot be changed at all or cannot be changed without specialized equipment. ROMs retain stored data when the power is off and are therefore nonvolatile memories.

Semiconductor ROMs are categorized as shown in fig. 1.13.

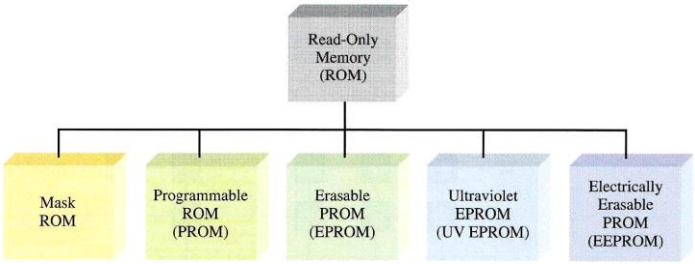



Fig. 1.13

**MASK ROM**

The mask ROM is the type in which the data are permanently stored in the memory during the manufacturing process. Once the memory is programmed, it cannot be changed. Fig. 1.14 shows MOS ROM cells. The presence of a connection from a row line to the gate of a transistor represents a 1 and no gate connection to a row line represents a 0.

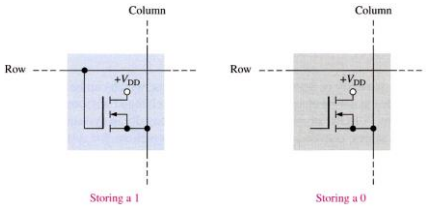



Fig. 1.14

**PROM**

The PROM, or programmable ROM, is the type in which the data are electrically stored by the user with the aid of specialized equipment. In the programming process, a sufficient current is injected through the fusible link to burn it open to create a stored 0. The link is left intact for a stored 1. Both the mask ROM and the PROM can be of either MOS or bipolar technology.

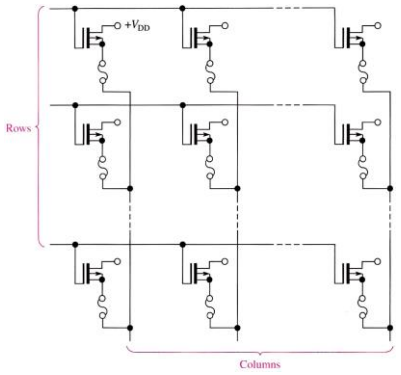



Fig. 1.15

**EPROM**

An EPROM is an erasable PROM. It can be reprogrammed. The data bits in this type of array are represented by the presence or absence of a stored gate charge. Erasure of a data bit is a process that removes the gate charge.

Two basic types of erasable PROMs are the ultraviolet erasable PROM (UV EPROM) and the electrically erasable PROM (EEPROM).

**UV PROM**

UV EPROM device is recognized by the transparent quartz lid on the package, as shown in figure. Erasure is done by exposure of the chip to high-intensity ultraviolet radiation through the quartz window on top of the package. The positive charge stored on the gate is neutralized after several minutes to an hour of exposure time.

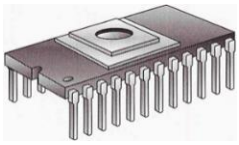


Fig. 1.16

**EEPROM**

An electrically erasable PROM can be both erased and programmed with electrical pulses. The EEPROM can be rapidly programmed and erased in-circuit for reprogramming.

**Applications of ROM**

A ROM stores data that are used repeatedly in system applications, such as tables, conversions, or programmed instructions for system initialization and operation.

## 1.5.2 Random Access Memory (RAM)

RAMs are read/write memories in which data can be written into or read from any selected address in any sequence. There are two types of RAM.

i- Static Random Access Memory

ii- Dynamic Random Access Memory

## Static Random Access Memory (SRAM)

Static RAMs generally use latches as storage elements. As long as dc power is applied to a static memory cell, it can retain a 1 or 0 state indefinitely. If power is removed, the stored data bit is lost.

**SRAM Cell**

Fig. 1.17 shows a basic SRAM latch memory cell. The cell is selected by putting HIGH on row and column lines. A data bit (1 or 0) is written into the cell by placing it on the Data in line. A data bit is read by taking it off the Data out line.

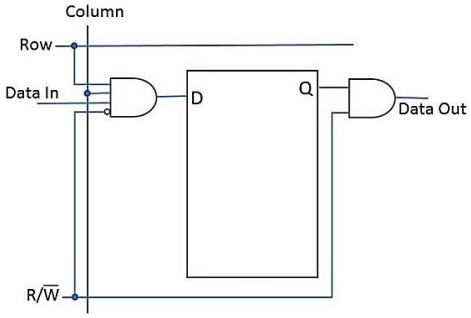



Fig. 1.17

**Write operation in SRAM Cell**

A HIGH on the row and column lines is applied to select the cell. To write a data unit, a bit (1 or 0) is applied to data input line. A LOW on the $R/\overline{W}$ line (WRITE mode) enables the input AND gate and disables the output AND gate, which causes each data bit to be stored in a selected cell.

**Read operation in SRAM Cell**

A HIGH on the row and column lines is applied to select the cell. To read a data unit, the Read line is taken to its active state by applying HIGH on $R/\overline{W}$. This disables the input AND gate and enables the output AND gate. This causes the data bit stored in the selected cell to appear on the output.

## Dynamic Random Access Memory (DRAM)

Dynamic memory cells store a data bit in a small capacitor. The advantage of this type of cell is allowing very large memory arrays to be

constructed on a chip at a lower cost per bit. The disadvantage is that the capacitor loses the stored data bit unless its charge is refreshed periodically. To refresh requires additional memory circuitry and complicates the operation of the DRAM.

**DRAM Cell**

Fig. 1.18 shows a typical DRAM cell consisting of a single MOS transistor (MOSFET) and a capacitor. In this type of cell, the transistor acts as a switch.

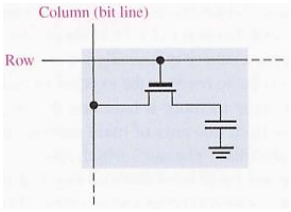



Fig. 1.18

Input and output buffers are used for write and read operations. These buffers are controlled by $R/\overline{W}$ signal. A refreshing buffer is also used for refreshing the data. These buffers are not shown in the cell for simplicity.

**Write operation in DRAM Cell**

A LOW on the $R/\overline{W}$ line (WRITE mode) enables the tristate input buffer and disables the output buffer. A HIGH on the row line is applied to close the transistor. When a voltage is applied on the bit line, this will transfer to capacitor and store in it. When the row line is taken back LOW, the transistor turns off and disconnects the capacitor from the bit line.

**Read operation in DRAM Cell**

To read from the cell, the $R/\overline{W}$ (Read/Write) line is HIGH, enabling the output buffer and disabling the input buffer. When the row line is taken HIGH, the transistor turns on and connects the capacitor to the bit line and thus to the output buffer (sense amplifier), so the data bit appears on the data-output line.

**Refreshing Process**

For refreshing the memory cell, the $R/\overline{W}$ line is HIGH, the row line is HIGH, and the refresh line is HIGH. The transistor turns on, connecting the capacitor to the bit line. The output buffer is enabled, and the stored data bit is applied to the input of the refresh buffer, which is enabled by the HIGH on

the refresh input. This produces a voltage on the bit line corresponding to the stored bit, thus refreshing the capacitor.

### 1.5.3 Cache Memory

Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It is used to hold those parts of data and program which are most frequently used by CPU. Cache memory is used to reduce the average time to access data from the main memory.

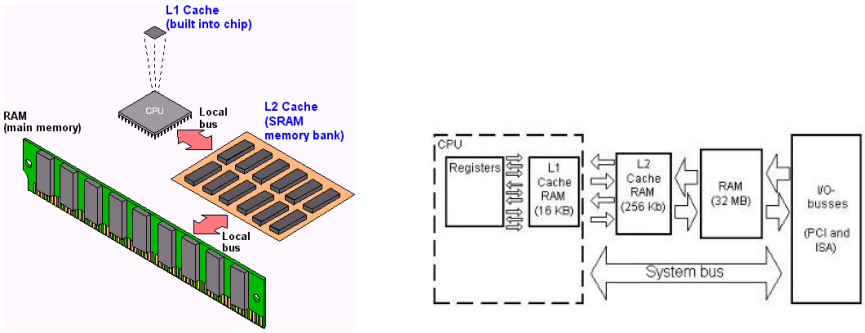



Fig. 1.19

There are different types of cache called L1, L2 and L3 cache.

L1 cache is the smallest and fastest cache which is directly built into the processor and it is used to store CPU's recently accessed information.

L2 cache is larger than L1 but smaller than L3. It is slower than L1 but faster than L3. It is located outside and separated from the CPU chip core.

L3 cache is the largest and slowest cache which is built on the motherboard within CPU module itself and is used by CPU.

### Flash Memory

Flash memory is a non-volatile memory chip used for storage and for transferring data between a personal computer (PC) and digital devices. It has the ability to be electronically reprogrammed and erased. It is often found in USB flash drives, MP3 players, SD memory cards, digital cameras and solid-state drives.

Flash memory, erases data in units called blocks and rewrites data at the byte level. Flash is technically a variant of EEPROM, but the industry reserves the term EEPROM for byte-level erasable memory and applies the term flash memory to larger block-level erasable memory.

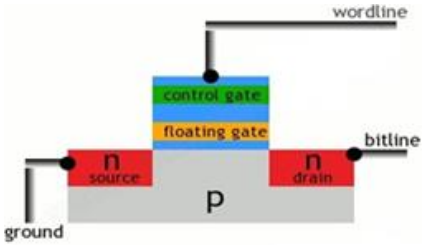


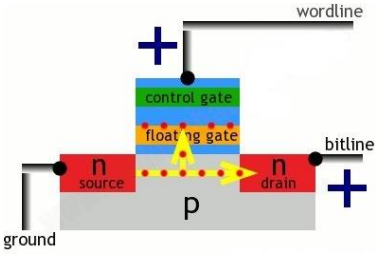

Fig. 1.20

**STRUCTURE**

A basic flash memory cell consists of a storage transistor like MOSFETs only they have two gates on top instead of one as shown in fig. 1.20. It's an n-p-n sandwich with two gates, one called a control gate and one called a floating gate. The two gates are separated by oxide layers through which current cannot normally pass.

**WORKING**

Flash memory incorporates the use of floating-gate transistors to store data. The floating gate stores the electrical charge and controls the flow of the electrical current. Electrons are added to or removed from the floating gate to change the storage transistor's threshold voltage. Changing the voltage affects whether a cell is programmed as a zero or a one.




Both the source and the drain regions are rich in electrons, but electrons cannot flow from source to drain because of the p-type material between them. But if we apply a positive voltage to the transistor's two contacts, called the bit line and the word line, electrons get pulled in a rush from source to drain. A few also move through the oxide layer by a process called tunneling and get stuck on the floating gate. The presence of electrons on the floating gate stores a one. The electrons will stay there, even when the positive voltages are removed.

The electrons can be flushed out by putting a negative voltage on the wordline—which repels the electrons back the way they came, clearing the floating gate and making the transistor store a zero again.

**Flash Memory Types**

A flash memory chip is composed of NOR or NAND gates. NOR was created by Intel in 1988. NOR has a faster read speed than NAND and can

read and edit more precisely, but it comes at a higher price point per byte. NOR flash memory mostly used for executing code. The shelf life of NOR flash is 10,000 to 1,000,000 write/erase cycles.

NAND was developed by Toshiba a year after NOR was produced. It has slower read speed, has a lower cost per bit, requires less chip area per cell and has added resilience. The shelf life of a NAND gate is approximately 100,000 write/erase cycles.

# 1.6 Understand Components of Processor

## Control Unit:

This unit generates timing and control signals to control data flow between microprocessor and peripheral devices. These signals are according to instructions.

## Register Unit:

This unit contains various registers which stores data temporary. It also stores temporary the results of instructions. This unit is also called memory unit.

## Arithmetic Logic Unit:

ALU performs processing of arithmetic (addition, subtraction, multiplication, division) and logic (NOT, OR, AND, XOR) operations. The results of this unit are either stored in a register or sent to a peripheral device. These all units are connected to each other through group of wires called buses.

## EXERCISE

### *SECTION-I (MULTIPLE CHOICE QUESTIONS)*

1. Which processor is a 4-bit processor?
   (a) 4004 (b) 8008 (c) 8080 (d) 8085
2. The size of instruction queue in 8086-microprocessor is;
   (a) 3 bytes (b) 4 bytes (c) 6 bytes (d) 8 bytes
3. If sign flag is set, then result of ALU operation is;
   (a) Negative (b) Positive (c) Zero (d) None of these
4. Stack Pointer stores the offset address of;
   (a) Stack Segment (b) Data Segment
   (c) Code Segment (d) Extra Segment
5. If CS = 0200H and IP = 1234H, then physical address will be;
   (a) 02000H (b) 01234H (c) 03234H (d) 02343H

6. Which is not a part of execution unit?
   (a) General registers (b) ALU (c) Flag register (d) Instruction Pointer
7. BIU stands for;
   (a) Bus Interrupt Unit (b) Bus Interface Unit
   (c) Buffer Interface Unit (d) None of these
8. Flag which controls the single step mode of 8086-microprocessor is;
   (a) Interrupt Flag (b) Trap Flag (c) Direction Flag (d) Sign Flag
9. Which of the following is a correct physical address formation technique?
   (a) CS:DS (b) AX:BX (c) IP:AX (d) CS:IP
10. 8086 microprocessor can address memory;
    (a) 1 MB (b) 2 MB (c) 4 MB (d) 8 MB
11. Offset address of Code segment is stored in;
    (a) IP (b) BP (c) SP (d) SI
12. Flag register of 8086 microprocessor consists of;
    (a) 5 Flags (b) 7 Flags (c) 9 Flags (d) 11 Flags
13. $IO/\overline{M}$ signal is a part of;
    (a) Data Bus (b) Address Bus (c) Control Bus (d) None of these
14. Flash memory is a type of;
    (a) EEPROM (b) UVPROM (c) SRAM (d) DRAM
15. Trap Flag is a type of flag;
    (a) Base (b) Offset (c) Conditional (d) Control

**ANSWER KEY**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. | a | 2. | c | 3. | a | 4. | a | 5. | c |
| 6. | d | 7. | b | 8. | b | 9. | d | 10. | a |
| 11. | a | 12. | c | 13. | c | 14. | a | 15. | d |

## *SECTION-II (SHORT QUESTIONS)*

1. Define dedicated microprocessor.
2. Define general purpose microprocessor.
3. Define word length of a microprocessor.
4. What is the function of arithmetic logic unit?
5. What is the function of Bus Interface Unit?
6. What is the function of Execution Unit?
7. Name the parts of Bus Interface Unit.
8. Name the parts of Execution Unit.
9. Write the function of accumulator register.

10. Name pointer registers of 8088 processor.
11. Name index registers of 8088 processor.
12. What is the purpose of pointer registers?
13. Define Flag register.
14. What is the purpose of zero flag?
15. What is the purpose of sign flag?
16. Define conditional Flags.
17. Define control Flags.
18. What is the purpose of segment registers?
19. Name the segment registers of 8086 microprocessor.
20. Define Instruction Queue register.
21. What is meant by base address?
22. What is meant by offset address?
23. What is the function of address summing block?
24. Write the purpose of Instruction Pointer.
25. Define the term “Pipelining”.
26. Define multitasking.
27. Define parallel processing.
28. Write any two differences between 8086 and 8088 microprocessors?
29. What is the function of ALE (Address Latch Enable) Signal?
30. Define cache memory.

## ***SECTION-III (LONG QUESTIONS)***

1. Draw the block diagram of 8086/8088 microprocessor & explain each block.
2. Explain the bus timing diagrams of 8086/8088 microprocessors.
3. Draw Flag Register of 8088 Microprocessor and describe function of each flag.
4. What is meant by bus buffering & latching? Explain the bus buffering and latching of 8086 microprocessor.
5. Enlist types of RAM and explain each briefly.

# CHAPTER 02
# Assembly Language

## 2.1 INSTRUCTION & INSTRUCTION SET

### INSTRUCTION

An instruction is a binary pattern designed inside the microprocessor to perform a specific function. In other words, it is actually a command to the microprocessor to perform a given task on specified data.

### INSTRUCTION FORMAT

Each instruction has two parts: one is the task to be performed called the operation code (opcode) and the other is the data to be operated on called the operand (data).

### INSTRUCTION SET

The entire group of instructions that a microprocessor supports is called Instruction Set. The instruction set determines what functions the microprocessor can perform. 8085 has instruction set of 246 instructions. There are 117 basic instructions in 8086/88 microprocessors.

### CLASSIFICATION OF INSTRUCTION GROUPS

The 8086 microprocessor supports 6 types of instructions −

- Data Transfer Instructions
- Arithmetic Instructions
- Bit Manipulation Instructions
- String Instructions
- Program Transfer Instructions
- Processor Control Instructions

#### DATA TRANSFER INSTRUCTIONS

These instructions are used to transfer the data from the source operand to the destination operand. Following are the list of instructions under this group.

**Instruction to transfer a word**

- **MOV** − Copy the byte or word from the provided source to the provided destination.
- **PUSH** − Put a word at the top of the stack.
- **POP** − Get a word from the top of the stack to the provided location.
- **PUSHA** − Put all the registers into the stack.
- **POPA** − Get words from the stack to all registers.
- **XCHG** − Exchange the data from two locations.
- **XLAT** − Translate a byte in AL using a table in the memory.

**Instructions for input and output port transfer**

- **IN** − Read a byte or word from the provided port to the accumulator.
- **OUT** − Send out a byte or word from the accumulator to the provided port.

**Instructions to transfer the address**

- **LEA** − Load the address of operand into the provided register.
- **LDS** − Load DS register and other provided register from the memory
- **LES** − Load ES register and other provided register from the memory.

**Instructions to transfer flag registers**

- **LAHF** − Load AH with the low byte of the flag register.
- **SAHF** − Store AH register to low byte of the flag register.
- **PUSHF** − Copy the flag register at the top of the stack.
- **POPF** − Copy a word at the top of the stack to the flag register.

**ARITHMETIC INSTRUCTIONS**

These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc. Following is the list of instructions under this group –

**Instructions to perform addition**

- **ADD** − Add the provided byte to byte/word to word.
- **ADC** − Add with carry.
- **INC** − Increment the provided byte/word by 1.
- **AAA** − Adjust ASCII after addition.
- **DAA** − Adjust the decimal after the addition/subtraction operation.

**Instructions to perform subtraction**

- **SUB** − Subtract the byte from byte/word from word.
- **SBB** − Perform subtraction with borrow.
- **DEC** − Decrement the provided byte/word by 1.

- **NPG** − Negate each bit of the provided byte/word and add 1/2's complement.
- **CMP** − Compare 2 provided byte/word.
- **AAS** − Adjust ASCII codes after subtraction.
- **DAS** − Adjust decimal after subtraction.

**Instruction to perform multiplication**

- **MUL** − Multiply unsigned byte by byte/word by word.
- **IMUL** − Multiply signed byte by byte/word by word.
- **AAM** − Adjust ASCII codes after multiplication.

**Instructions to perform division**

- **DIV** − Divide the unsigned word by byte or unsigned double word by word.
- **IDIV** − Divide the signed word by byte or signed double word by word.
- **AAD** − Adjust ASCII codes after division.
- **CBW** − Fill the upper byte of the word with the copies of sign bit of the lower byte.
- **CWD** − Fill the upper word of the double word with the sign bit of the lower word.

**BIT MANIPULATION INSTRUCTIONS**

These instructions are used to perform operations where data bits are involved, i.e. operations like logical, shift, etc. Following is the list of instructions under this group –

**Instructions to perform logical operation**

- **NOT** − Invert each bit of a byte or word.
- **AND** − ANDing each bit in a byte/word with the corresponding bit in another byte/word.
- **OR** − ORing each bit in a byte/word with the corresponding bit in another byte/word.
- **XOR** − Perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
- **TEST** − Add operands to update flags, without affecting operands.

**Instructions to perform shift operations**

- **SHL/SAL** − Shift bits of a byte/word towards left and put zero(S) in LSBs.
- **SHR** − Shift bits of a byte/word towards the right and put zero(S) in MSBs.

- **SAR** − Shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

**Instructions to perform rotate operations**

- **ROL** − Rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
- **ROR** − Rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
- **RCR** − Rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.
- **RCL** − Rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

**STRING INSTRUCTIONS**

String is a group of bytes/words and their memory is always allocated in a sequential order. Following is the list of instructions under this group −

- **REP** − Repeat the given instruction till CX ≠ 0.
- **REPE/REPZ** − Repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **REPNE/REPNZ** − Repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **MOVS/MOVSB/MOVSW** − Move the byte/word from one string to another.
- **COMS/COMPSB/COMPSW** − Compare two string bytes/words.
- **INS/INSB/INSW** − Input string/byte/word from the I/O port to the provided memory location.
- **OUTS/OUTSB/OUTSW** − Output string/byte/word from the provided memory location to the I/O port.
- **SCAS/SCASB/SCASW** − Scan a string and compare its byte with a byte in AL or string word with a word in AX.
- **LODS/LODSB/LODSW** − Store the string byte into AL or string word into AX.

**PROGRAM TRANSFER INSTRUCTIONS**

These instructions are used to transfer/branch the instructions during an execution. It includes the following instructions −

**Instructions to transfer the instruction without any condition −**

- **CALL** − Call a procedure and save their return address to the stack.

- **RET** − Return from the procedure to the main program.
- **JMP** − Jump to the provided address to proceed to the next instruction.

**Instructions to transfer the instruction with some conditions −**

- **JA/JNBE** − Jump if above/not below/equal instruction satisfies.
- **JAE/JNB** − Jump if above/not below instruction satisfies.
- **JBE/JNA** − Jump if below/equal/ not above instruction satisfies.
- **JC** − Jump if carry flag CF = 1
- **JE/JZ** − Jump if equal/zero flag ZF = 1
- **JG/JNLE** − Jump if greater/not less than/equal instruction satisfies.
- **JGE/JNL** − Jump if greater than/equal/not less than instruction satisfies.
- **JL/JNGE** − Jump if less than/not greater than/equal instruction satisfies.
- **JLE/JNG** − Jump if less than/equal/if not greater than instruction satisfies.
- **JNC** − Jump if no carry flag (CF = 0)
- **JNE/JNZ** − Jump if not equal/zero flag ZF = 0
- **JNO** − Jump if no overflow flag OF = 0
- **JNP/JPO** − Jump if not parity/parity odd PF = 0
- **JNS** − Jump if not sign SF = 0
- **JO** − Jump if overflow flag OF = 1
- **JP/JPE** − Jump if parity/parity even PF = 1
- **JS** − Jump if sign flag SF = 1

**Instructions to execute the instructions for number of times.**

- **LOOP** − Loop a group of instructions until the condition satisfies, i.e., CX = 0
- **LOOPE/LOOPZ** − Loop a group of instructions till it satisfies ZF = 1 & CX = 0
- **LOOPNE/LOOPNZ** − Loop a group of instructions till it satisfies ZF = 0 & CX = 0
- **JCXZ** − Jump to the provided address if CX = 0

**Instructions to call the interrupts**

- **INT** − Interrupt the program during execution and calling service specified.
- **INTO** − Interrupt the program during execution if OF = 1
- **IRET** − Return from interrupt service to the main program

**PROCESSOR CONTROL INSTRUCTIONS**

These instructions are used to control the processor action by setting/resetting the flag values. Following are the instructions under this group −

- **STC** − Set carry flag CF to 1
- **CLC** − Clear/reset carry flag CF to 0
- **CMC** − Put complement at the state of carry flag CF.
- **STD** − Set the direction flag DF to 1
- **CLD** − Clear/reset the direction flag DF to 0
- **STI** − Set the interrupt enable flag to 1, i.e., enable INTR input.
- **CLI** − Clear the interrupt enable flag to 0, i.e., disable INTR input.

# 2.2 PROGRAMMING LANGUAGES

The set of instructions in a specific order executed by the CPU to solve a particular problem is called program or software. A programming language defines a format for writing commands, instructions, and other syntax in a specific order to create a software program. Programming languages are a way of communicating with the computer.

There are many different computer languages available for writing programs. Each has its strengths and weaknesses and must be assessed based upon need. A language that is particularly well suited for one application may not work for another.

There are two types of computer languages:

- Low Level Languages
- High Level Languages

## LOW LEVEL LANGUAGES

Computer languages which are machine dependent are called low-level languages. Low level languages provide the programmer with a high degree of control, but they require a detailed knowledge of the hardware to be used. They are really only required for advanced programming needs. There are two main types of low-level languages.

- Machine language
- Assembly language

## MACHINE LANGUAGE

Machine language instructions typically use some binary codes to represent operations, such as addition, and some bits to represent operands, or perhaps the location of the next instruction. A computer language which is in the binary form is called machine language.

Machine language is difficult to read and write, since it does not resemble human language, and its codes vary from computer to computer. Programs written in this way would tend to be error prone and would be very difficult to debug. For this reason, programs are generally written in a language which is easier for humans to understand and can also be translated into machine code for the processor to understand.

## ASSEMBLY LANGUAGE

Assembly language is very close to machine language. The commands are represented in assembly language by short names called mnemonics. For example, LDA means Load Accumulator with a particular data value. Because each type of processor has a different set of operations, different processors use different assembly languages. Assembly language programming is complex but it provides a much higher degree of control than high level languages. Programs written in assembly language code are translated into machine code by an assembler.

## HIGH LEVEL LANGUAGES

High level languages are close to human languages and far from the machine language. These are machine independent languages which are also known as "third generation" languages. These languages consist of English words, basic mathematical Symbols and a few punctuation characters. These languages allow simple statements to be expressed concisely. Each high-level language has its own language translator. BASIC, FORTRAN, PASCAL, COBOL, C, C++, Visual BASIC and JAVA are the examples are high level languages.

## COMPARISON BETWEEN ASSEMBLY AND MACHINE LANGUAGES

| Assembly Language | Machine Language |
|---|---|
| Assembly language refers to a low-level programming language. | Machine language ranks as the lowest level programming language. |
| Machine language is only understood by the computer. | Assembly language is only understood by human beings. |

| | |
|---|---|
| In machine language data only represented with the help of binary or hexadecimal format. | In assembly language data can be represented with the help of mnemonics such as MOV, ADD, SUB, etc. |
| Machine language is very difficult to understand by the human being. | Assembly language is easy to understand by the human being. |
| Modifications and error fixing cannot be done in machine language. | Modifications and error fixing can be done in assembly language. |
| The Machine language can't be memorized. | The assembly language can be memorized. |
| Execution speed of machine language is fast. | Execution speed of assembly language is slow. |
| There is no need of translator. The machine understandable form is the machine language. | There is need of a translator to convert mnemonics into machine understandable form. |
| Machine language is hardware dependent. | Assembly language is the machine dependent and it is not portable. |

# 2.3 PROGRAMMING LANGUAGE TRANSLATORS

The computer understands instructions written in the machine language. Language translators are software programming tools that convert a high level or assembly programs into machine code. The software checks the program for errors optimizes the code and generates machine language for that program. The program which is to be translated is called as source code whereas the translated program is known as machine code or object program which is in binary form.

All the software available for this purpose can be categorized into three main categories.

- Assembler
- Compiler
- Interpreter

## ASSEMBLER

An assembler is a program that translates an assembly language program into machine code. Assembly program is called source code and machine code is called object code.

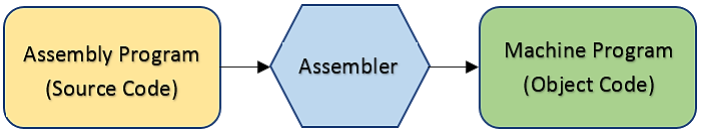



Fig. 2.1

### COMPILER

A compiler is a program that translates a source program written in some high-level programming language into machine language (or machine code). A compiler first reads the whole program before executing it.

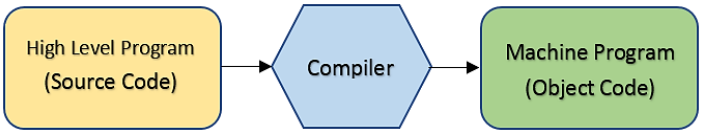



Fig. 2.2

### INTERPRETER

An interpreter is a program that transforms a high-level programming code into code that can be understood by the machine (machine code). The interpreter reads each statement of code and then converts or executes it directly.

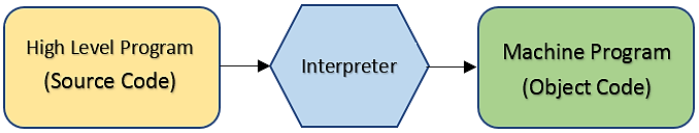



Fig. 2.3

## 2.4 FIELDS OF ASSEMBLY LANGUAGE STATEMENTS

Assembly language statements are entered one statement per line. Each statement follows the following format –

[label:] Opcode [Operand] [; comment]

The fields in the square brackets are optional. A basic instruction has two parts, the first one is the name of the instruction (or the mnemonic), which is to be executed, and the second are the operands or the parameters of the command.

### LABEL FIELD:

It consists of symbols and used to represent the memory address. A label can be placed at the beginning of a statement. During assembly, the label is

assigned the current value of instruction pointer and serves as an instruction operand. Label is followed by a colon (:).

### OPERATION CODE:

The part of program statement, that specifies the operation to be performed is called operation code. It occurs after the label field.

### OPERAND:

The part of program statement, that specifies the data on which the operation is to be performed is called operand.

### COMMENTS:

The last field of an Assembler source statement is the comment field. This field is optional and personal remarks of programmer. It does not execute during program running. A semi-colon (;) is used before comment field.

## 2.5 ADDRESSING MODES

The different ways in which a processor can access data are called addressing modes. 8086 microprocessor accesses the data in different ways such as from different registers, from memory locations or from I/O ports. 8086/88 have following addressing modes.

i. Register addressing mode
ii. Immediate addressing mode
iii. Direct addressing mode
iv. Indirect addressing mode
v. Base plus Index addressing mode
vi. Base relative addressing mode
vii. Base relative plus index addressing mode

### REGISTER ADDRESSING MODE

In register addressing mode, a byte or word is transferred from the source register to the destination register.

Fig. 2.4 illustrates the MOV instruction and defines the direction of data flow. The source is to the right and the destination is to the left. The MOV AX, BX instruction transfers the word contents of the source register (BX) into the destination register (AX).

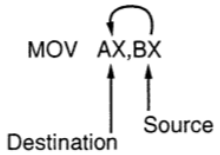


Fig. 2.4

The source never changes, but the destination almost always changes.

The 8-bit or 16-bit registers are certainly valid operands for this instruction. But both operands should be in the same size. Never mix 8- and 16-bit register types.

**Prohibited Instructions**

It is important for instructions to use registers that are the same size. Never mix an 8-bit register with a 16-bit register.

A segment-to-segment register MOV instruction is virtually the only type of register MOV instruction not allowed.

Code segment register may not be changed by a MOV instruction, because the address of the next instruction is found in both IP and CS. If only CS were changed, the address of the next instruction would be unpredictable.

| Instruction | Operation |
|---|---|
| MOV ES, DS | Not allowed (segment-to-segment) |
| MOV BL, DX | Not allowed (mixed sizes) |
| MOV CS, AX | Not allowed (the CS may not be the destination register) |

## IMMEDIATE ADDRESSING MODE

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode. Immediate addressing transfers the source immediate byte or word of data into the destination register or memory location.

Fig. 2.5 shows the operation of a MOV AX,3456H instruction. This instruction copies the 3456H from the instruction, located in the memory immediately following the hexadecimal opcode, into register AX. The source data overwrite the destination data.

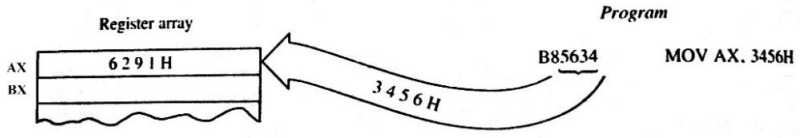



Fig. 2.5

Table shows many different variations of MOV instructions that apply immediate data.

| *Assembly Language* | *Size* | *Operation* |
|---|---|---|
| MOV BL,44 | 8-bits | Copies a 44 decimal (2CH) into BL |
| MOV AX,44H | 16-bits | Copies a 0044H into AX |
| MOV SI,0 | 16-bits | Copies a 0000H into SI |
| MOV CH,100 | 8-bits | Copies a 100 decimal (64H) into CH |
| MOV AL,'A' | 8-bits | Copies an ASCII A into AL |
| MOV AX,'AB' | 16-bits | Copies an ASCII BA* into AX |
| MOV CL,11001110B | 8-bits | Copies a 11001110 binary into CL |

## DIRECT ADDRESSING MODE

The addressing mode in which the operand's offset is given in the instruction as 16-bit displacement element. In this addressing mode the 16-bit effective address of the data is the part of the instruction.

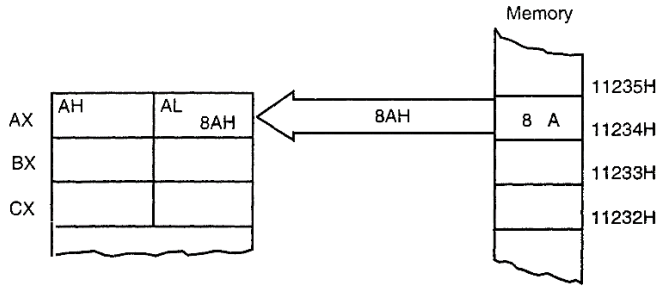
Direct addressing moves a byte or word between a memory location and a register. The instruction set does not support a memory-to-memory transfer.

There are two basic forms of direct data addressing:

- Direct addressing, which applies to a MOV between a memory location and AL and AX.
- Displacement addressing, which applies to almost any instruction in the instruction set.

Direct addressing, with a MOV instruction, transfers data between a memory location, located within the data segment, and the AL (8-bit) or AX (16-bit), register. A MOV instruction using this type of addressing is usually a 3-byte long instruction.

MOV AL, [1234H] instruction is the example of direct addressing. This instruction loads AL from data segment memory location (1234H). Fig. 2.6 shows how this instruction transfers a copy of the byte-sized contents of memory location 11234H into AL. The effective address is formed by adding 1234H (the offset address) to 10000H (the data segment address) in a system operating in the real mode.




The operation of the MOV AL,[1234H] instruction when DS = 1000H

Fig. 2.6

In the direct addressing instructions, only the AX and AL registers may be moved to and from the memory.

MOV CL, [1234H] instruction is the example of displacement addressing. Both basically perform the same operation except for the destination register (CL verses AL). The MOV AL, [1234H] instruction is three bytes long and the MOV CL, [1234H] instruction is four bytes long.

## INDIRECT ADDRESSING MODE

In register indirect addressing mode, the operand's offset is placed in any one of the registers BX, BP, SI, DI as specified in the instruction. The effective address of the data is in the base register or an index register that is specified by the instruction.

Register indirect addressing transfers a byte or word between a register and a memory location addressed by an index or base register. The index and base registers are BP, BX, DI, and SI.

Example: the MOV AX, [BX] instruction copies the word-sized data from the data segment offset address indexed by BX into register AX.

Register indirect addressing allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI, and SI. For example, if register BX contains a 2000H and the MOV AX, [BX] instruction executes, the word contents of data segment offset address 2000H is copied into register AX.

If the microprocessor is operated in the real mode and DS = 0100H, this instruction addresses a word stored at memory bytes 03000H and 03001H and transfers it into register AX as shown in fig. 2.7. The contents of 03000H are moved into AL and the contents of 03001H are moved into AH.

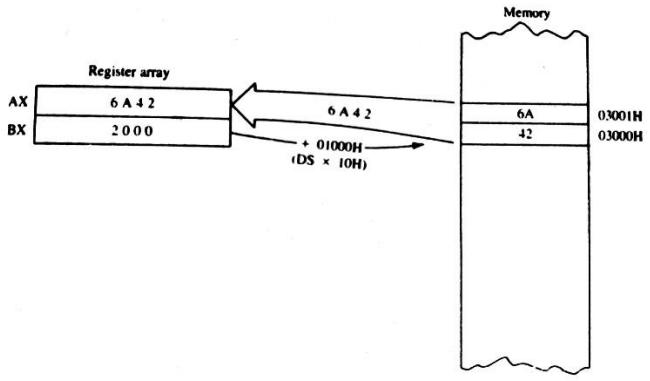



Fig. 2.7

The data segment is used by default with addressing mode that uses BX, DI, or SI to address memory. If register BP addresses memory, the stack segment is used by default.

## BASE PLUS INDEX ADDRESSING MODE

Base-plus-index addressing is similar to indirect addressing because it indirectly addresses memory data. This type of addressing uses one base register (BP or BX), and one index register (DI or SI) to indirectly address memory. The base register often holds the beginning location of a memory array, while the index register holds the relative position of an element in the array. Remember that whenever BP addresses memory data, both the stack segment register and BP generate the effective address.

Base plus index addressing transfers a byte or word between a register and the memory location addressed by a base register (BP or BX) plus an index register (DI or SI).

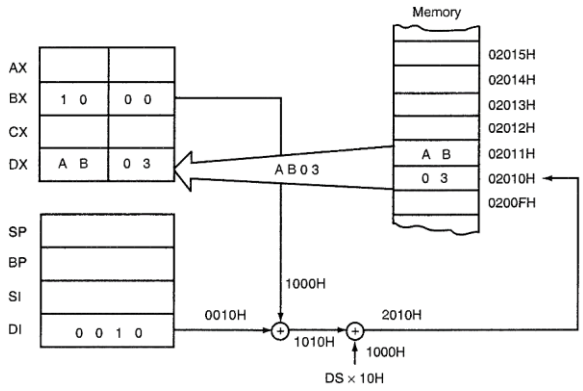



Fig. 2.8

Example: the MOV [BX+DI], CL instruction copies the byte- sized contents of register CL into the data segment memory location addressed by BX plus DI.

In this example, BX = 1000H, DI= 00I0H, and DS = 0100H, which translate into memory address 02010H. This instruction transfers a copy of the word from location 02010H into the DX register as shown in fig. 2.8.

A major use of the base plus index addressing mode is to address elements in a memory array. To accomplish this, load the BX register (base) with the beginning address of the array and the DI register (index) with the element number to be accessed. Fig. 2.9 shows the use of BX and DI to access an element in an array of data.

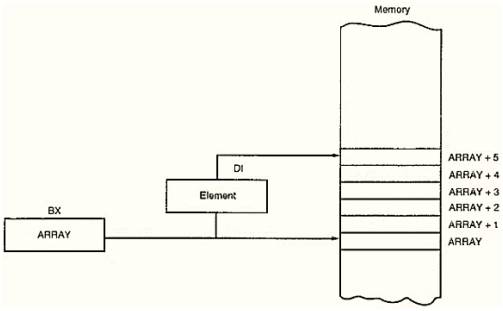



Fig. 2.9

## BASE RELATIVE ADDRESSING MODE

Register relative addressing is similar to base-plus-index addressing and displacement addressing. In register relative addressing, the data in a segment of memory are addressed by adding the displacement to the contents of a base or an index register (BP, BX, DI, or SI).

This addressing mode moves a byte or word between a register and the memory location addressed by an index or base register plus a displacement.

Example: MOV AX, [BX+4] or MOV AX, ARRAY[BX]. The first instruction loads AX from the data segment address formed by BX plus 4. The second instruction loads AX from the data segment memory location in ARRAY plus the contents of BX.

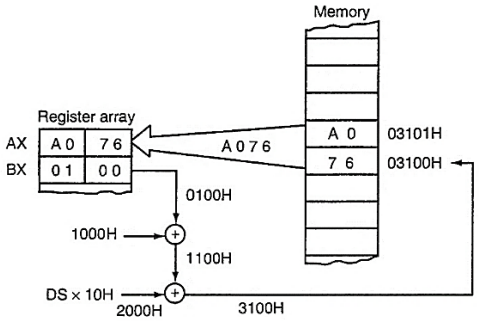



Fig. 2.10

Fig. 2.10 shows the operation of the MOV AX, [BX+l000H] instruction. In this example, BX = 0100H and DS = 0200H, so the address generated is the sum of DS × 10H, BX, and the displacement of 1000H is 03100H. Remember that BX, DI, or SI address the data segment and BP addresses the stack segment.

## BASE RELATIVE PLUS INDEX ADDRESSING MODE

The base relative-plus-index addressing mode is similar to the base-plus-index addressing mode, but adds a displacement besides using a base

register and an index register to form the memory address. This type of addressing mode often addresses a two-dimensional array of memory data.

Base relative-plus-index addressing is the least-used addressing mode. Moves a byte or word between a register and the memory location addressed by an index and base register plus a displacement.

Example: MOV AX, [BX+SI+100H]. The instruction loads AX from the data segment address formed by BX plus SI plus 100H.

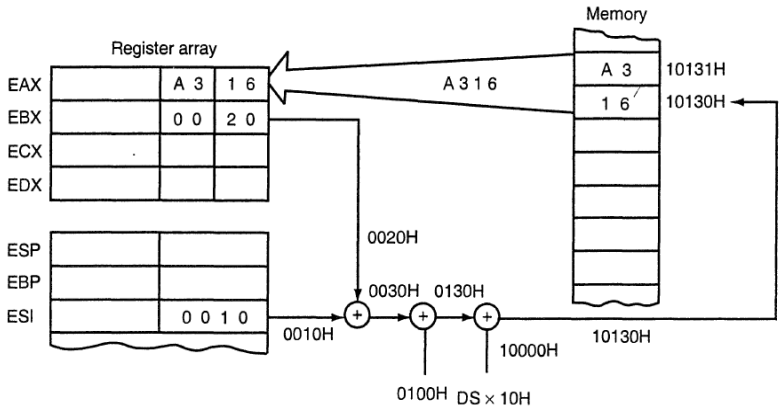



Fig. 2.11

Fig. 2.11 shows how data are referenced if the instruction executed by the microprocessor is a MOV AX, [BX+SI+100H]. The displacement of 100H adds to BX and SI to form the offset address within the data segment. Registers BX = 0020H, SI = 0010H, and DS = 1000H, so the effective address for this instruction is 10130H, the sum of these registers plus a displacement of 100H. This addressing mode is too complex for frequent use in a program.

# 2.6 DATA MOVEMENT INSTRUCTIONS

## PUSH AND POP INSTRUCTIONS

### PUSH

PUSH instruction always transfers two bytes of data to the stack in 8086/88 microprocessors. The source of data may be any internal 16-bit register, immediate data, any segment register or, any two bytes of memory data.

Whenever data are pushed onto the stack, the first (most-significant) data byte moves to the stack segment memory location addressed by SP-1. The second (least significant) data byte moves into the stack segment memory location addressed by SP-2 as shown in fig. 2.12.

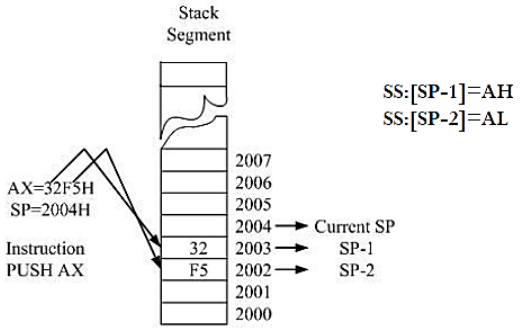



Fig. 2.12

**POP**

POP instruction performs the inverse operation of a PUSH instruction. The POP instruction removes data from stack and places it into the target 16-bit register, segment register or, 16-bit memory location.

The least significant byte of data is removed from SP and most significant byte is removed from stack segment memory location addressed by SP+1 as shown in fig. 2.13.

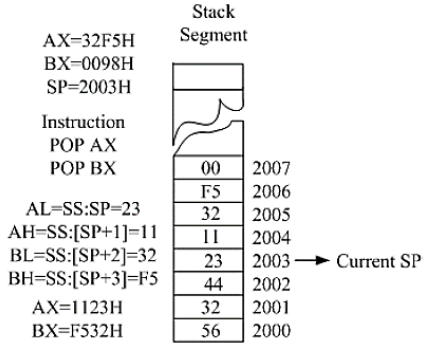



Fig. 2.13

**Example:**

An assembly program is given below:

MOV AX, 7645H
MOV BX, 4477H
MOV CX, 8899H
MOV DX, BX
PUSH DX
PUSH AX
PUSH BX
PUSHF
POP CX
PUSH 1000H

POP DX

POP AX

Assume that, SS = 2000H, SP = 2009H and F = FFCDH then, Find out

(a) The physical address.

(b) The value of SS and SP after the end of program.

(c) The value of AX, BX, CX, DX and Flag register after the end of the program.

(d) Draw the memory map in detail.

**Solution:**

(a) Physical address = SS × 10H + SP = 2000 × 10H + 2009H = 22009H

(b) Final value of SS = 2000H and SP = Current SP – No. of PUSH × 2 + No. of POP × 2

= 2009H – 5 × 2 + 3 × 2 = 2009H – 10 + 6 = 2009H – 4 = 2005H

(c) AX = 4477H; BX = 4477H; CX = FFCDH; DX = 1000H; F = FFCDH

(d)

| Instructions | Operation |
|---|---|
| MOV AX, 7645H | AX=7645H |
| MOV BX, 4477H | BX=4477H |
| MOV CX, 8899H | CX=8899H |
| MOV DX, BX | DX=4477H |
| PUSH DX | SP=2007H |
| PUSH AX | SP=2005H |
| PUSH BX | SP=2003H |
| PUSHF | SP=2001 |
| POP CX | CX=FFCDH<br>SP=2003H |
| PUSH 1000H | SP=2001H |
| POP DX | DX= 1000H |
| POP AX | AX=4477H |

(c) AX= 4477H
CX=FFCDH
DX=1000H
BX= 4477H
F=FFCDH

= 2005H

(d) Memory Map
Stack
Segment

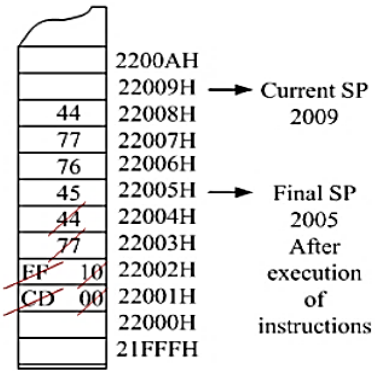



Fig. 2.14

## IN AND OUT INSTRUCTIONS

### IN INSTRUCTION

IN instruction transfer data from an input device to the microprocessor's accumulator register (AL, AX or EAX).

### OUT INSTRUCTION

OUT instruction transfer data from the microprocessor's accumulator register (AL, AX or EAX) to an output device.

IN and OUT instructions are basically two types.

- Fixed port instruction

- Variable port instruction

**Fixed Port Instruction**

If the port address is directly given in the instruction, then it is called fixed port instruction. For example:

IN AL, 19H ; 8 bits are transferred to AL register from input port 19H.

OUT 19H, AX ; 16 bits are transferred to output port 0019H from AX register.

**Variable Port Instruction:**

If the port address is stored in DX register, then it is called variable port instruction. For example:

IN AX, DX ; 16 bits are transferred to AX register from input port DX.

OUT DX, AX ; 16 bits are transferred to output port DX from AX register.

Only 16-bits (A0 to A15) are decoded. Address connections above A15 are undefined for I/O instructions.

# 2.7 ARITHMETIC INSTRUCTIONS

## THE ADD AND SUB INSTRUCTIONS

The ADD and SUB instructions are used for performing simple addition/subtraction of binary data in byte, word and double word size.

Syntax

The ADD and SUB instructions have the following syntax −

ADD/SUB destination, source

The ADD/SUB instruction can take place between −

- Register to register
- Memory to register
- Register to memory
- Register to constant data
- Memory to constant data

However, like other instructions, memory-to-memory operations are not possible using ADD/SUB instructions.

An ADD or SUB operation sets or clears the overflow and carry flags.

**Example:**

MOV AL, 3FH ; a byte is moved to AL

MOV BH, 23H ; immediate data must be in a register

SUB AL, BH ; AL = AL - BH

## THE MUL/IMUL INSTRUCTION

There are two instructions for multiplying binary data. The MUL (Multiply) instruction handles unsigned data and the IMUL (Integer Multiply) handles signed data. Both instructions affect the Carry and Overflow flag.

**Syntax**

The syntax for the MUL/IMUL instructions is as follows −

MUL/IMUL multiplier

Multiplicand in both cases will be in an accumulator, depending upon the size of the multiplicand and the multiplier and the generated product is also stored in two registers depending upon the size of the operands. Following section explains MUL instructions with different cases –

**When two bytes are multiplied -**

The multiplicand is in the AL register, and the multiplier is a byte in the memory or in another register. The product is in AX. High-order 8 bits of the product is stored in AH and the low-order 8 bits are stored in AL.

**When two one-word values are multiplied -**

The multiplicand should be in the AX register, and the multiplier is a word in memory or another register. For example, for an instruction like MUL DX, you must store the multiplier in DX and the multiplicand in AX.

The resultant product is a double word, which will need two registers. The high-order (leftmost) portion gets stored in DX and the lower-order (rightmost) portion gets stored in AX.

**Example:**

```
MOV AL, 25H          ; a byte is moved to AL
MOV BL, 65H          ; immediate data must be in a register
MUL BL               ; AL = 25H × 65H
MOV RESULT, AX       ; the result is saved
```

## THE DIV/IDIV INSTRUCTIONS

The division operation generates two elements - a quotient and a remainder. In case of multiplication, overflow does not occur because double-length registers are used to keep the product. However, in case of division, overflow may occur. The processor generates an interrupt if overflow occurs.

The DIV (Divide) instruction is used for unsigned data and the IDIV (Integer Divide) is used for signed data.

**Syntax**

The format for the DIV/IDIV instruction −

DIV/IDIV divisor

The dividend is in an accumulator. Both the instructions can work with 8-bit, 16-bit or 32-bit operands. The operation affects all six status flags. Following section explains three cases of division with different operand size –

**When the divisor is 1 byte -**

The dividend is assumed to be in the AX register (16 bits). After division, the quotient goes to the AL register and the remainder goes to the AH register.

**When the divisor is 1 word -**

The dividend is assumed to be 32 bits long and in the DX:AX registers. The high-order 16 bits are in DX and the low-order 16 bits are in AX. After division, the 16-bit quotient goes to the AX register and the 16-bit remainder goes to the DX register.

**Example:**

```
MOV AL, 96H             ; AL holds numerator
SUB AH, AH              ; AH must be cleared
MOV BH, 65H             ; move denominator to register
DIV BH                  ; divide AX by BH
MOV QUOTIENT, AL        ; quotient = AL
MOV REMAINDER, AH       ; remainder = AH
```

# 2.8 LOGIC INSTRUCTIONS

## AND INSTRUCTION

This instruction ANDs each bit in a source byte or word with the same numbered bit in a destination byte or word. The syntax of this instruction is as,

**AND destination, source**

The result is put in the specified destination. The content of the specified source is not changed. The source can be an immediate number, the content of a register, or the content of a memory location. The destination can be a register or a memory location. The source and the destination cannot both be memory locations.

## OR INSTRUCTION

This instruction ORs each bit in a source byte or word with the same numbered bit in a destination byte or word. The syntax of this instruction is as,

**OR destination, source**

The result is put in the specified destination. The content of the specified source is not changed. The source can be an immediate number, the content of a register, or the content of a memory location. The destination can be a register or a memory location. The source and destination cannot both be memory locations.

## XOR INSTRUCTION

This instruction Exclusive-ORs each bit in a source byte or word with the same numbered bit in a destination byte or word. The syntax of this instruction is as,

**XOR destination, source**

The result is put in the specified destination. The content of the specified source is not changed. The source can be an immediate number, the content of a register, or the content of a memory location. The destination can be a register or a memory location. The source and destination cannot both be memory locations.

## NOT INSTRUCTION

The NOT instruction inverts each bit (forms the 1's complement) of a byte or word in the specified destination. The syntax of this instruction is as,

**NOT destination**

The destination can be a register or a memory location. This instruction does not affect any flag.

NOT BX ; complement content or BX register

## SHIFT AND ROTATE INSTRUCTIONS

The computer system works on machine language, which consists of binary numbers. In the 8086 Microprocessor, there is sometimes the need to perform some necessary shift and rotate operations on data. For this purpose, following Shift and Rotate instructions are present in the 8086 Microprocessor.

1. SHR (Shift Right)
2. SHL (Shift Left)
3. ROL (Rotate Left)
4. ROR (Rotate Right)

**SHR INSTRUCTION**

The SHR instruction is an abbreviation for 'Shift Right'. This instruction simply shifts all bits in the register to the right side one by one. The MSB is replaced by a ZERO and LSB is stored in the Carry Flag (CF) as shown in fig. 2.15.

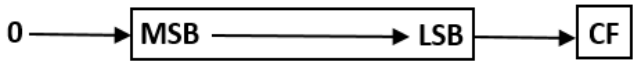



Fig. 2.15

If we want to shift more than one bit, then that value is first stored in CL register. The instruction syntax is:

SHR destination, bits to be shifted

**SHL INSTRUCTION**

The SHR instruction is an abbreviation for 'Shift Left'. This instruction simply shifts all bits in the register to the left side one by one. The LSB is replaced by a ZERO and MSB is stored in the Carry Flag (CF) as shown in fig. 2.16.

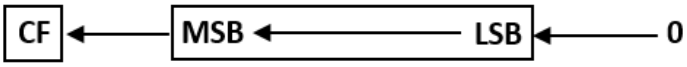



Fig. 2.16

**ROR INSTRUCTION**

The ROR instruction stands for 'Rotate Right'. This instruction rotates the mentioned bits in the register to the right side one by one such that rightmost bit that is being rotated is again stored as the MSB in the register, and it is also stored in the Carry Flag (CF).

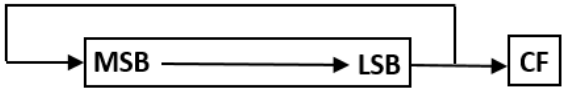



Fig. 2.17

Syntax: ROR Register, Bits to be shifted

Example: ROR AH, 4

**ROL INSTRUCTION**

The ROL instruction is an abbreviation for 'Rotate Left'. This instruction rotates the mentioned bits in the register to the left side one by one such that leftmost bit that is being rotated is again stored as the rightmost bit in the register, and it is also stored in the Carry Flag (CF).

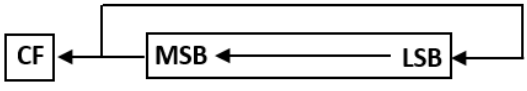



Fig. 2.18

Syntax: ROL Register, Bits to be shifted

Example: ROL AH, 4

# 2.9 PROGRAMME CONTROL INSTRUCTIONS

Program control instructions modify or change the flow of a program. It is the instruction that alters the sequence of the program's execution, which means it changes the value of the program counter, due to which the execution of the program changes. There are four types of program control instructions for 8086/88 microprocessor.

(i) Jump (ii) Call (iii) Return (iv) Interrupt

## JUMP INSTRUCTIONS

Jump Instructions are used for changing the flow of execution of instructions in the processor. We can jump to any instruction in between the program, then this can be achieved by these instructions. There are two types of Jump instructions:

- Unconditional Jump Instructions
- Conditional Jump Instructions

### UNCONDITIONAL JUMP INSTRUCTIONS

These instructions are used to jump on a particular location unconditionally, i.e. there is no need to satisfy any condition for the jump to take place. There are three types of procedures used for unconditional jump. They are:

**NEAR –** This procedure targets within the same code segment. (Intra-segment)

**SHORT -** This procedure also targets within the same code segment, but the offset is 1 byte long. (Intra-segment).

**FAR -** In this procedure, the target is outside the segment and the size of the pointer is double word. (Inter-segment).

**Syntax:**

JMP procedure_namememory_location

**Example:** JMP short target

### CONDITIONAL JUMPS

In these types of instructions, the processor must check for the particular condition. If it is true, then only the jump takes place else the normal flow in the execution of the statements is maintained.

The ALU operations set flags in the status word (Flag register). The conditional jump statement tests the flag and jump if the flag is set. There are many conditional jump instructions just like JC, JNC, JZ, JNZ and so on.

## CALL INSTRUCTION

The CALL instruction is used whenever we need to make a call to some procedure or a subprogram. A procedure is group of instructions that usually performs one task. It is a reusable section of a software program which is stored in memory once but can be used as often as necessary. Whenever a CALL is made, the following process takes place inside the microprocessor:

- The address of the next instruction that exists in the caller program is stored in the stack.
- The instruction queue is emptied for accommodating the instructions of the procedure.
- Then, the contents of the instruction pointer (IP) are changed with the address of the first instruction of the procedure.
- The subsequent instructions of the procedure are stored in the instruction queue for execution.

The Syntax for the CALL instruction is as follows:

CALL subprogram_name

CALL instruction can be of two types.

**Near CALL:** A procedure is known as NEAR procedure if is written in the same code segment which is calling that procedure. Only Instruction Pointer contents will be changed in NEAR procedure.

**FAR CALL:** A procedure is known as FAR procedure if it is written in the different code segment than the calling segment. In this case both Instruction Pointer and the Code Segment register content will be changed.

## THE RET INSTRUCTION

The RET instruction stands for return. This instruction is used at the end of the procedures or the subprograms. This instruction transfers the execution to the caller program. Whenever the RET instruction is called, the following process takes place inside the microprocessor:

- The address of the next instruction in the mainline program which was previously stored inside the stack is now again fetched and is placed inside the instruction pointer (IP).

- The instruction queue will now again be filled with the subsequent instructions of the mainline program.

The Syntax for the RET instruction is as follows:

RET

The following diagram illustrates how the control of the instruction execution is transferred within the code from one program to another whenever a procedure is called and whenever it returns the execution.

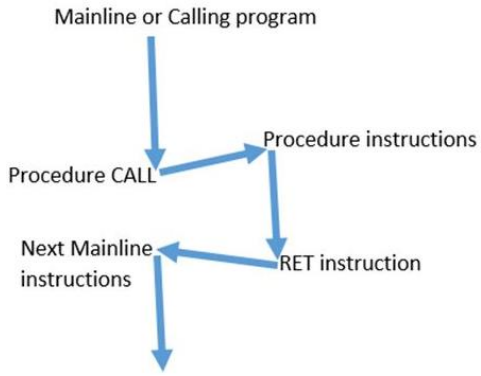



Fig. 2.19

## EXERCISE

### *SECTION-I (MULTIPLE CHOICE QUESTIONS)*

1. The computer language in which 0's and 1's are used is called;
   (a) Assembly (b) C ++ (c) Machine (d) None
2. The process of saving data in stack is done by the instruction;
   (a) CALL (b) PUSH (c) POP (d) HLT
3. Instruction MOV AX, 10H uses the addressing mode;
   (a) Register (b) Immediate (c) Direct (d) Indirect
4. The symbol used in assembly program after label field is
   (a) Colon (b) Comma (c) Semi Colon (d) Asterisk
5. A microprocessor can directly understand the language;
   (a) Machine (b) High Level (c) Assembly (d) All
6. Which of the following is an arithmetic instruction?
   (a) ADD (b) JMP (c) XCHG (d) MOV

7. Which one of the following is an invalid instruction?
   (a) MOV AX, BX (b) MOV CS, DS
   (c) MOV AX, 1234H (d) MOV AX, (1234H)
8. The instruction used to enter the subroutine into the main program?
   (a) PUSH (b) POP (c) IN (d) CALL
9. The part of the instruction that tells the processor what to do is called;
   (a) Comment (b) Label (c) Opcode (d) Operand
10. The data on which the operation is to be performed is called
   (a) Comment (b) Label (c) Opcode (d) Operand
11. If CX = 000EH, then result of INC CX instruction will be
   (a) 000FH (b) 000DH (c) 000AH (d) 000BH
12. In JNZ instruction, jump is occurred when zero flag is
   (a) High (b) Set (c) Reset (d) Undefined
13. The instruction in an assembly program is called
   (a) Line (b) Comment (c) Label (d) Mnemonic
14. The result of MOV AL, 65 is to store……. in AL register
   (a) $(0100\ 0010)_2$ (b) 65 H (c) $(0100\ 0001)_2$ (d) 42 H
15. The result of instruction AND AX, 0 is always
   (a) FFFFH (b) 000FH (c) FF00H (d) 0000H

**ANSWER KEY**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. | c | 2. | b | 3. | b | 4. | a | 5. | a |
| 6. | a | 7. | b | 8. | d | 9. | c | 10. | d |
| 11. | a | 12. | c | 13. | d | 14. | c | 15. | d |

## *SECTION-II (SHORT QUESTIONS)*

1. Define instruction set.
2. Define data transfer instructions.
3. Define arithmetic instructions.
4. What are logical instructions?
5. Define bit manipulation instructions.
6. What is meant by program transfer instructions?
7. Define processor control instructions.
8. Give two examples of data transfer instructions.
9. Give any two examples of program transfer instructions.
10. Give two examples of processor control instructions.
11. Give two examples of string instructions.
12. What is the wrong with MOV BL, CX instruction?

13. What is the function of PUSH & POP instructions?
14. Describe the operation of instruction ADD AX, BX.
15. How two numbers are multiplied in 8086 assembly language?
16. Describe the operation of instruction AND CX, DX.
17. Describe the operation of instruction OR AX, BX.
18. What is difference between SHIFT and ROTATE instructions?
19. What is difference between JMP and CALL instructions?
20. Write down the difference between near and far jump instructions.
21. Define assembly language.
22. Define machine language.
23. Define high level language.
24. Define low level language.
25. What are Mnemonics?
26. Differentiate between Opcode and Operand with example.
27. Define Label field in assembly language.
28. Define Comments field in assembly language.
29. Differentiate between interpreter and compiler.
30. What is the purpose of Assembler?
31. Define addressing mode.
32. Define register addressing mode.
33. Define direct addressing mode.
34. Define immediate addressing mode.
35. What is benefit of index addressing?

## ***SECTION-III (LONG QUESTIONS)***

1. Explain PUSH and POP instructions with examples.
2. Explain the three programming languages for microcomputer.
3. Discuss fields of assemble language statement.
4. Describe register addressing and direct addressing with example.
5. Describe register relative addressing and Immediate addressing with example.
6. Describe register indirect addressing and base plus index addressing with example.

# CHAPTER 03
# Memory & Basic I/O Interface

## 3.1 MEMORY DEVICES

Every microprocessor-based system has a memory system. There are two basic types of memory.

- ROM: It stores system software and permanent system data.
- RAM: It stores temporary data and instructions.

Four commonly used memories are ROM, Flushable EEPROM, Static RAM, and Dynamic RAM.

**PIN CONNECTIONS**

Address Pins: All memory devices have address inputs. They select a memory location within the memory device. The number of address pins is related to the number of memory locations. Address inputs are labeled from $A_0$ to $A_N$. N is the total number of address pins minus 1. For example, the 2K memory has 11 address lines which are labelled as $A_0$ - $A_{10}$.

Data Pins: All memory devices have a set of data outputs or input/outputs. They are used to enter the data for storage or extract the data for reading. The data pins are typically bi-directional in read-write memories. The number of data pins is related to the size of the memory location. For example, memory device of 1K X 8 indicate a byte addressable memory with 10 address pins.

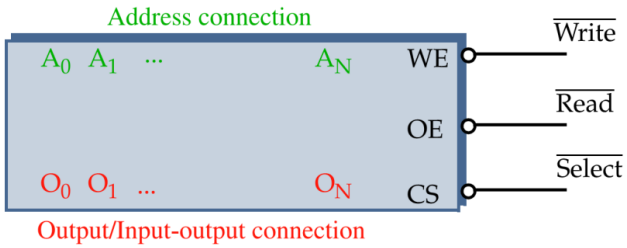



Fig. 3.1

Selection Pins: Each memory device has at least one chip select ($CS$) or chip enable ($CE$) pin that enables the memory device. This enables read and write operations. If more than one are present, then all must be 0 in order to perform a read or write. If they are active (logic 0), the memory device

performs a read or write operation. If they are inactive (logic 1), the memory is disabled and do not do any operation.

Control Pins: Each memory device has at least one control pin. For ROMs, an Output Enable ($OE$) or Gate ($G$) is present. The $OE$ pin enables and disables a set of tristate buffers. For RAMs, a read-write (R/$W$) or write enable ($WE$) and read enable ($OE$) are present. For dual control pin devices, it must be hold true that both are not 0 at the same time.

## 3.2 ADDRESS DECODING

Address decoding is the process of generating chip select (CS) signals from the address bus for each device in the system.

Different portions of memory are used for different purposes: RAM, ROM, I/O devices. Even if all the memory was of one type, we still have to implement it using multiple ICs. This means that for a given valid address, one and only one memory-mapped component must be accessed.

The address bus lines are split into two sections. The N most significant bits are used to generate the CS signals for the different devices. The M least significant signals are passed to the devices as addresses to the different memory cells or internal registers as shown in fig. 3.2.

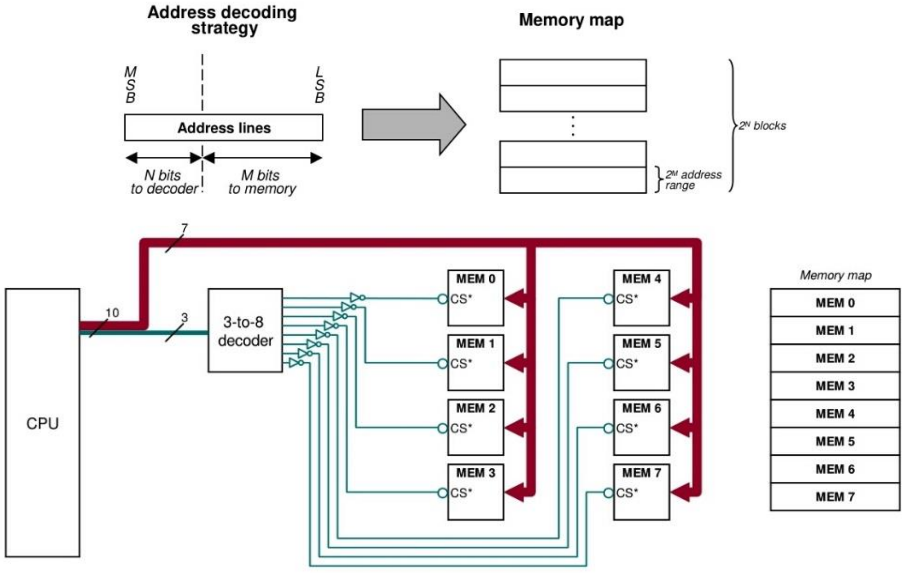



Fig. 3.2

**Address Decoding Methods**

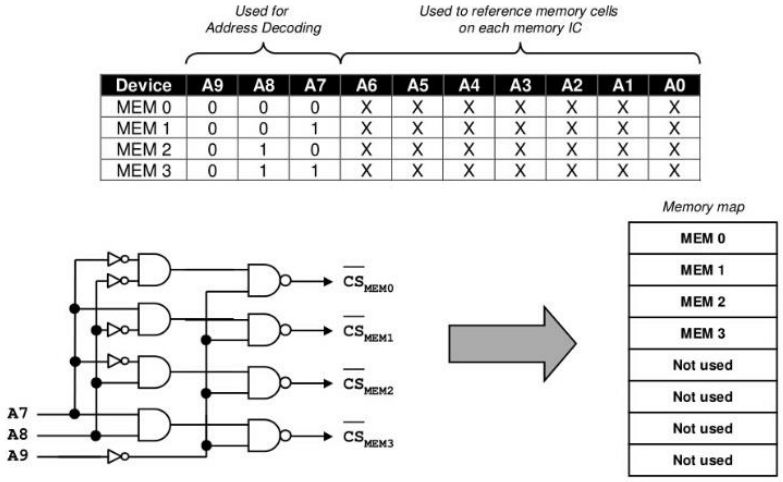
There are two basic address decoding strategies.

- Full address decoding
- Partial address decoding

**Full Address Decoding**

All the address lines are used to specify a memory location. Each physical memory location is identified by a unique address.

Example: Microprocessor with 10 address lines. We wish to address 512 bytes of memory. We still must use 128-byte memory chips.



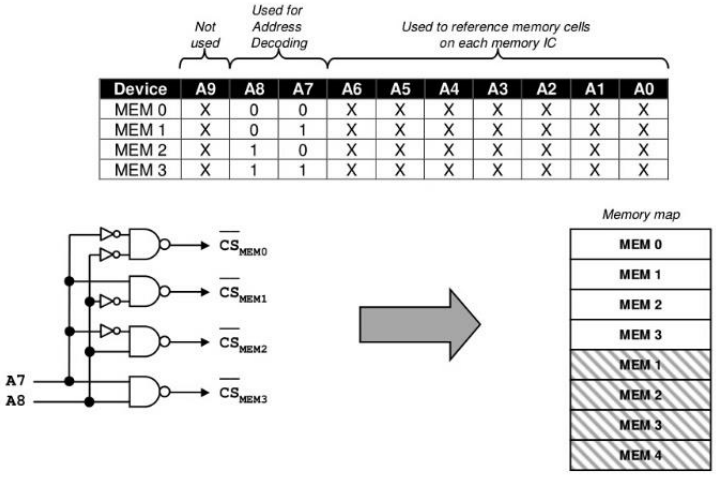

| Device | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|
| MEM 0 | 0 | 0 | 0 | X | X | X | X | X | X | X |
| MEM 1 | 0 | 0 | 1 | X | X | X | X | X | X | X |
| MEM 2 | 0 | 1 | 0 | X | X | X | X | X | X | X |
| MEM 3 | 0 | 1 | 1 | X | X | X | X | X | X | X |

Fig. 3.3

**Partial Address Decoding**

Since not all the address space is implemented, only a subset of the address lines is needed to point to the physical memory locations. Each physical memory location is identified by several possible addresses (using all combinations of the address lines that were not used).

Example: Microprocessor with 10 address lines. We wish to address 512 bytes of memory. We still must use 128-byte memory chips.




| Device | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|
| MEM 0 | X | 0 | 0 | X | X | X | X | X | X | X |
| MEM 1 | X | 0 | 1 | X | X | X | X | X | X | X |
| MEM 2 | X | 1 | 0 | X | X | X | X | X | X | X |
| MEM 3 | X | 1 | 1 | X | X | X | X | X | X | X |

Fig. 3.3

## 3.3 8088 and 80188 (8-BIT) MEMORY INTERFACE

8088/80188 both microprocessors have an 8-bit data bus, which makes them ideal to connect to common 8 bits memory devices available. For the 8088/80188 to function correctly with memory, however, the system must decode the address to select a memory component.

A controller is required to interface 8088/80188 microprocessors with memory. This receives 20 bits address from microprocessor and generates $\overline{RD}$, $\overline{WR}$ and $IO/\overline{M}$ to control memory system.

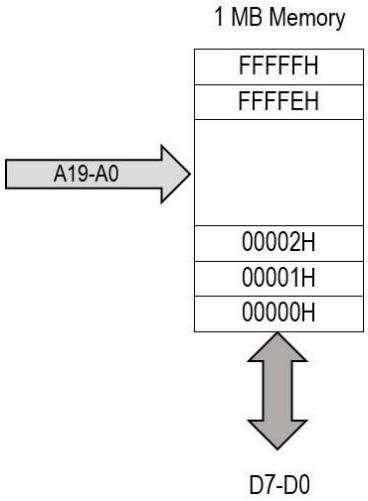



Fig. 3.4

The address range of 1MB memory is from 00000H to FFFFFH. 20 bits address bus selects required memory location and 8 bits data on that location transfers on data bus.

## 3.4 8086/80186 (16-BIT) MEMORY INTERFACE

16-bit memory interface is used in 8086, 80186, 80286, and 80386SX microprocessors. The 8086, 80186, 80286, and 80386SX microprocessors differ from the 8088/80188 in three ways:

- The data bus is 16 bits wide instead of 8 bits wide as on the 8088.
- The $IO/\overline{M}$ pin of the 8088 is replaced with a $M/\overline{IO}$ pin.
- There is a new control signal called bus high enable $\overline{BHE}$.

The address bit A0 or $\overline{BLE}$ is also used differently.

A few other differences exist between the 8086/80186 and the 80286/80386SX. The 80286/80386SX contains a 24-bit address bus (A23–A0) instead of the 20-bit address bus (A19–A0) of the 8086/80186. The 8086/80186 contain an $M/\overline{IO}$ signal, while the 80286 system and 80386SX

microprocessor contain control signals $\overline{MRDC}$ and $\overline{MWTC}$. instead of $\overline{RD}$ and $\overline{WR}$.

The 8086 and 80186 must be able to write data to any 16-bit location—or any 8-bit location. This means that the 16-bit data bus must be divided into two separate sections (or banks) that are 8 bits wide so that the microprocessor can write to either half (8-bit) or both halves (16-bit). For this purpose, 1MB memory is divided into two banks which are called even (low) bank and odd (high) bank.

Fig. 3.5 (a) illustrates the two banks of the memory. One bank (low bank) holds all the even-numbered memory locations, and the other bank (high bank) holds all the odd-numbered memory locations. Fig. 3.5 (b) shows the generation of separate 8086 write strobes for the memory.

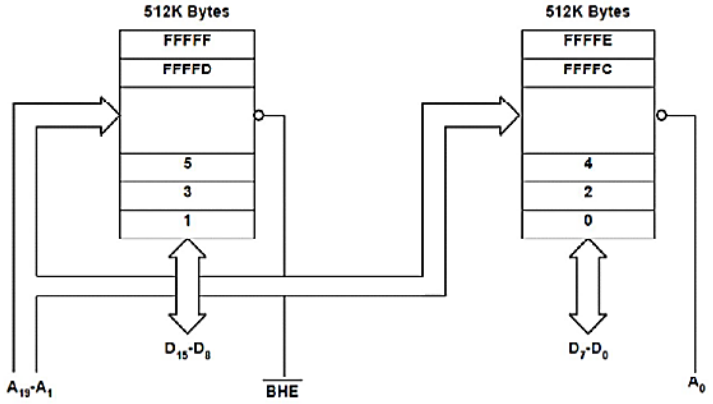



Fig. 3.5 (a)

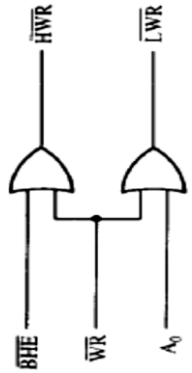



Fig. 3.5 (b)

To write 8-bit data on a memory location, required bank is selected. $\overline{BHE}$ is used to select higher bank and A0 is used to select lower bank of memory. To write 16-bit data on a memory location, both banks are selected. Table 3.1 shows the logic levels on these two pins for the bank or banks selected.

| $\overline{BHE}$ | $\overline{BLE}$ ($A_0$) | Function |
|---|---|---|
| 0 | 0 | Both banks enabled for a 16-bit transfer |
| 0 | 1 | High bank enabled for an 8-bit transfer |
| 1 | 0 | Low bank enabled for an 8-bit transfer |
| 1 | 1 | No banks enabled |

Table 3.1

# 3.5 32 and 64-BIT MEMORY INTERFACE

## i. 32-Bit Memory Interface

32-bit memory interface is used in 80386DX, and 80486 microprocessors. These processors have 32-bit data bus and therefore 4 banks of memory are required. The 80386DX/80486 contains a 32-bit address bus (A31–A0) which can address 4GB memory.

32-bit, 16-bit and 8-bit transfers are accomplished by different combinations of the bank selection signals, BE3, BE2, BE1, BE0.

To write 8-bit data on a memory location, any bank can be selected.

To write 16-bit data on a memory location, two banks are selected which can be Bank0 and Bank1 or Bank2 and Bank3.

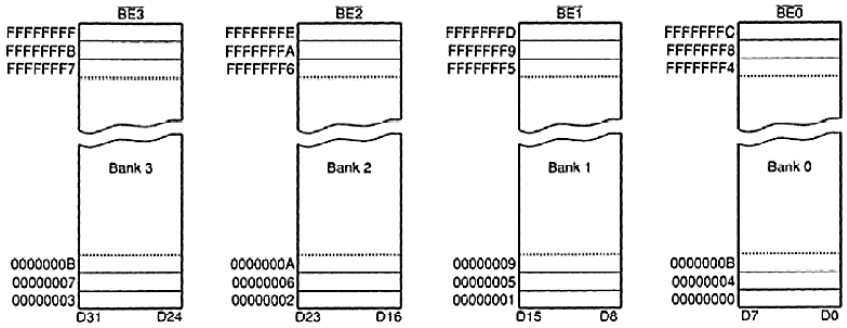



Fig. 3.6

To write 32-bit data on a memory location, all banks are selected simultaneously using decoder shown in fig. 3.7.

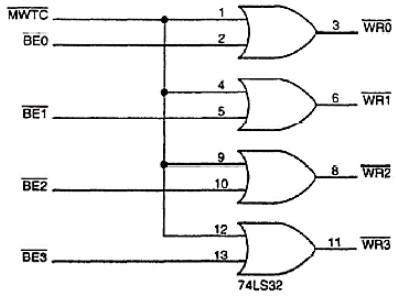



Fig. 3.7

## ii. 64-BIT Memory Interface

64-bit memory interface is used in Pentium, Pentium Pro, Pentium-II, and Pentium-III microprocessors. These processors have 64-bit data bus and therefore 8 banks of memory are required. These processors contain a 32-bit address bus (A31–A0) which can address 4GB memory.

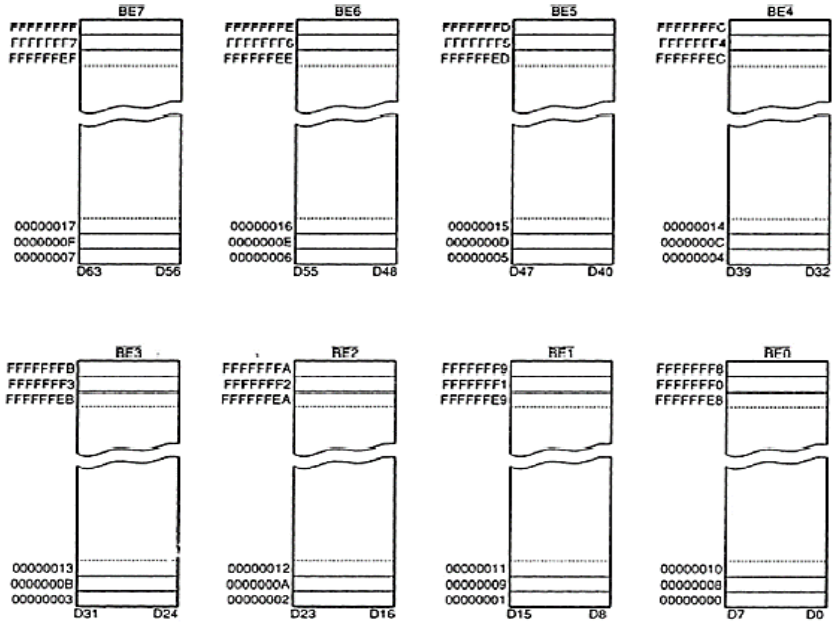



Fig. 3.8

64 bit, 32-bit, 16-bit and 8-bit transfers are accomplished by different combinations of the bank selection signals, BE7, BE6, BE5, BE4, BE3, BE2, BE1, BE0.

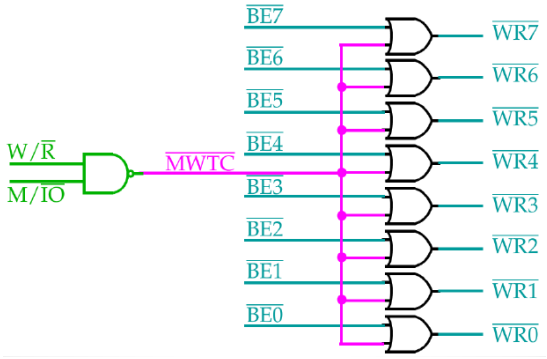



Fig. 3.9

- To write 8-bit data on a memory location, any bank can be selected.
- To write 16-bit data on a memory location, two banks are selected which can be Bank0 and Bank1, Bank2 and Bank3, Bank4 and Bank5 or Bank6 and Bank7.
- To write 32-bit data on a memory location, four banks are selected which can be Bank0, Bank1, Bank2 and Bank3 or Bank4, Bank5, Bank6 and Bank7.
- To write 64-bit data on a memory location, all banks are selected simultaneously using following decoder. The write strobes are

obtained by combining the bank enable signals ($BEx$) with the $\overline{MWTC}$ signal. $\overline{MWTC}$ is generated by combining the $M/\overline{IO}$ and $\overline{WR}$ signals.

# 3.6 DYNAMIC RANDOM ACCESS MEMORY (DRAM)

Dynamic memory cells store a data bit in a small capacitor. The advantage of this type of cell is allowing very large memory arrays to be constructed on a chip at a lower cost per bit. The disadvantage is that the capacitor loses the stored data bit unless its charge is refreshed periodically. To refresh requires additional memory circuitry and complicates the operation of the DRAM.

**DRAM Cell**

Fig. 3.11 shows a typical DRAM cell consisting of a single MOS transistor (MOSFET) and a capacitor.

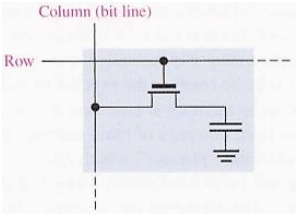



Fig. 3.11

In this type of cell, the transistor acts as a switch. Input and output buffers are used for write and read operations. These buffers are controlled by $R/\overline{W}$ signal. A refreshing buffer is also used for refreshing the data. These buffers are not shown in the cell for simplicity.

**Write operation in DRAM Cell**

A LOW on the $R/\overline{W}$ line (WRITE mode) enables the tristate input buffer and disables the output buffer. A HIGH on the row line is applied to close the transistor. When a voltage is applied on the bit line, this will transfer to capacitor and store in it. When the row line is taken back LOW, the transistor turns off and disconnects the capacitor from the bit line.

**Read operation in DRAM Cell**

To read from the cell, the $R/\overline{W}$ (Read/Write) line is HIGH, enabling the output buffer and disabling the input buffer. When the row line is taken HIGH, the transistor turns on and connects the capacitor to the bit line and thus to the output buffer (sense amplifier), so the data bit appears on the data-output line.

**REFRESHING PROCESS**

For refreshing the memory cell, the $R/\overline{W}$ line is HIGH, the row line is HIGH, and the refresh line is HIGH. The transistor turns on, connecting the capacitor to the bit line. The output buffer is enabled, and the stored data bit is applied to the input of the refresh buffer, which is enabled by the HIGH on the refresh input. This produces a voltage on the bit line corresponding to the stored bit, thus refreshing the capacitor.

# 3.7 INTRODUCTION TO I/O INTERFACE

## Input Interface

Input interface is a circuit arrangement which is used to enter data to microprocessor. In this circuit arrangement, tristate buffers are used.

The basic input device (to the microprocessor) is a set of tri-state buffers as shown in figure. The data received by toggle switches is applied to the input of tristate buffers (74ALS244 IC). The output of buffers is connected to the data bus of microprocessor.

When processor executes IN instruction then,

- I/O port address is decoded and $\overline{SEL}$=0 is generated to select desired port.
- This generated signal is applied to controlling signals $1\bar{G}$ and $2\bar{G}$ of tristate buffer 74ALS244.
- Due to this signal, tristate buffer IC is enabled and the data available at the input terminals “A” is transferred to its output terminals “Y”. This data is delivered to processor via data bus.
- After reading data, a high signal is applied to the controlling signals $1\bar{G}$ and $2\bar{G}$ of tristate buffer 74ALS244.
- Due to high signal, buffer IC is disabled and acts an open switch and isolates inputs from data bus until it reselects again.

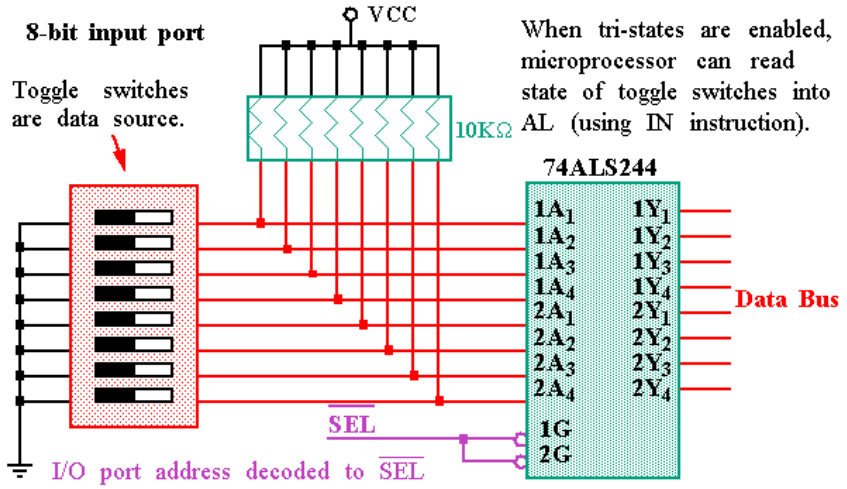



Fig. 3.12

**OUTPUT INTERFACE**

Output interface is a circuit arrangement which is used to receive data from microprocessor. In this circuit arrangement, data latches are used.

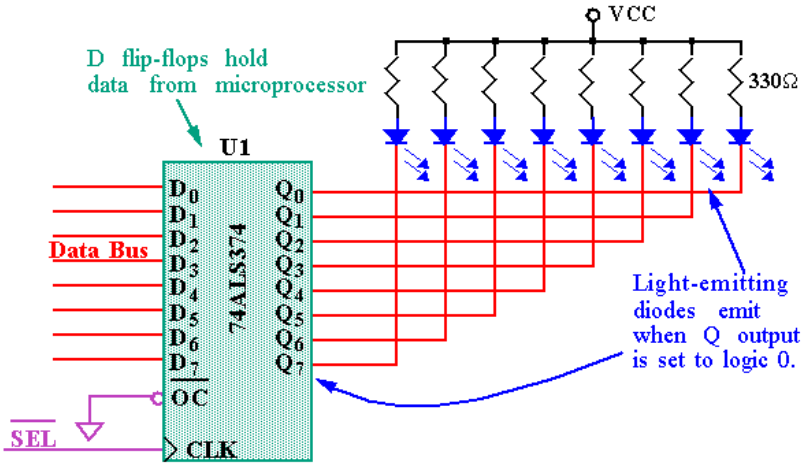



Fig. 3.13

The basic output device (from the microprocessor) is a set of latches as shown in figure. The data received from data bus is applied to the input of tristate latches (74ALS374 IC). The output of latches is connected to the LED's.

When processor executes OUT instruction then,

- I/O port address is decoded and $\overline{SEL}$=0 is generated to select desired port.
- This generated signal is applied to the CLK terminal of latches 74ALS374.

- After receiving CLK signal, the data available at the input terminals "$D_0$-$D_7$" is transferred to its output terminals "$Q_0$-$Q_7$".
- This data is delivered to LED's and LED's become ON or OFF according to processor data.

When instruction OUT is executed, then data remains on the data bus for less than 1 micro second. Thus, to receive this processor data at output, latches are used so that it can be stored until new data is received.

# 3.9 8254 PROGRAMMABLE INTERVAL TIMER

A clock signal is used in every microprocessor to synchronize all peripherals with computer. Usually, the frequency of this clock signal is very high. Computer has some functions in which low frequency is required. 8254 is a programmable interval timer designed to solve the timing control problems in a microprocessor.

It has three independent 16-bit down counters. These counters can be programmed for either binary or BCD count. It can handle inputs from DC to 10 MHz. It operates in +5V regulated power supply and has 24 pin signals.

## BLOCK DIAGRAM OF INTEL 8254

The fig. 3.14 shows the functional block diagram of 8254 Timer: It has 3 counters each with two inputs (Clock and Gate) and one output. Gate is used to enable or disable counting. When any value of count is loaded and value of gate is set (1), after every step value of count is decremented by 1 until it becomes zero.

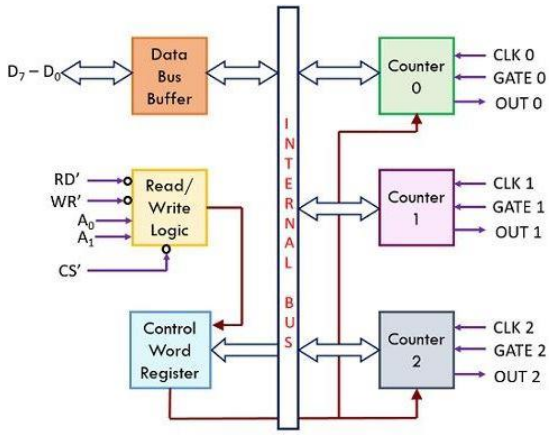



Fig. 3.14

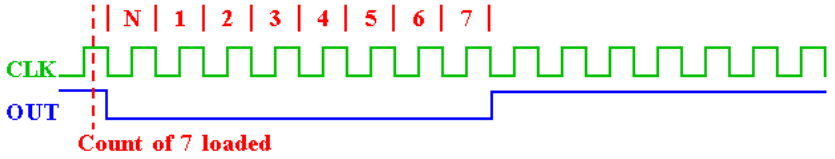
Each counter is programmed by writing a control word, followed by the initial count. The control word allows the programmer to select the counter, mode

of operation, type of operation (read/write) and selection of either a binary or BCD count.

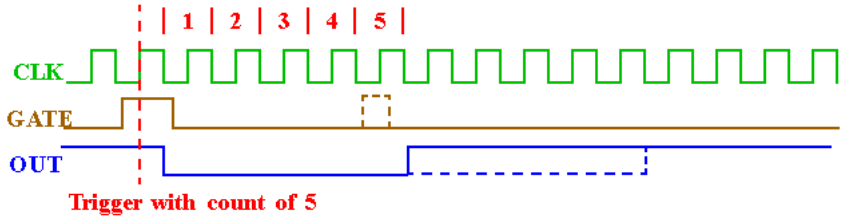
Each of the 8254 counters can be operated in six modes which are as under.

**Operating Modes of 8254:**

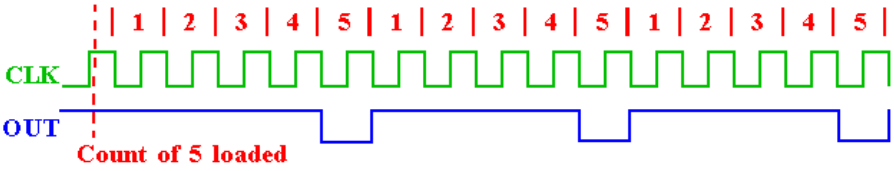
1. **Mode 0 (Interrupt on Terminal Count) –** Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially low, and will remain low until the counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written into the counter.



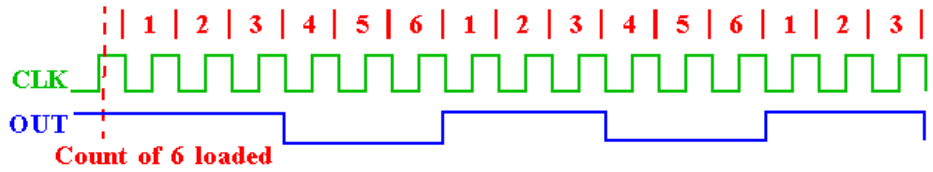

2. **Mode 1 (Programmable One Shot) –** OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the counter reaches zero.



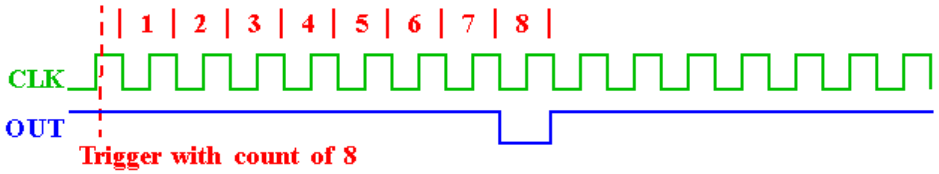

3. **Mode 2 (Rate Generator) –** Initially value of OUT is low. When counting is enabled, it becomes high and this process repeats periodically. This mode works as a frequency divider.




4. **Mode 3 (Square Wave Generator) –** This mode is used to generate square waveform. If the count is even, output is high for one half of the count and low for one half of the count. If the count is odd, output is high for one clocking period longer than it is low.




5. **Mode 4 (Software Triggered Strobe) –** In this mode counting is enabled by using GATE = 1 and disabled by GATE = 0. Initially value of OUT is high and becomes low when value of count is at last stage. Count is reloaded again for subsequent clock pulse.




6. **Mode 5 (Hardware Triggered Strobe) –** OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one clock pulse and then go high again. After writing the Control Word and initial count, the counter will not be loaded until the clock pulse after a trigger.

## 3.10 PROGRAMMABLE COMMUNICATION INTERFACE

A USART (Universal Synchronous/Asynchronous Receiver/Transmitter) is also called a programmable communications interface (PCI). When information is to be sent by 8086 over long distances, it is economical to send it on a single line. So, 8086 has to convert parallel data to serial data. Similarly, if 8086 receives serial data over long distances, the 8086 has to internally convert this into parallel data before processing it. Thus, lot of microprocessor time is required for such conversions.

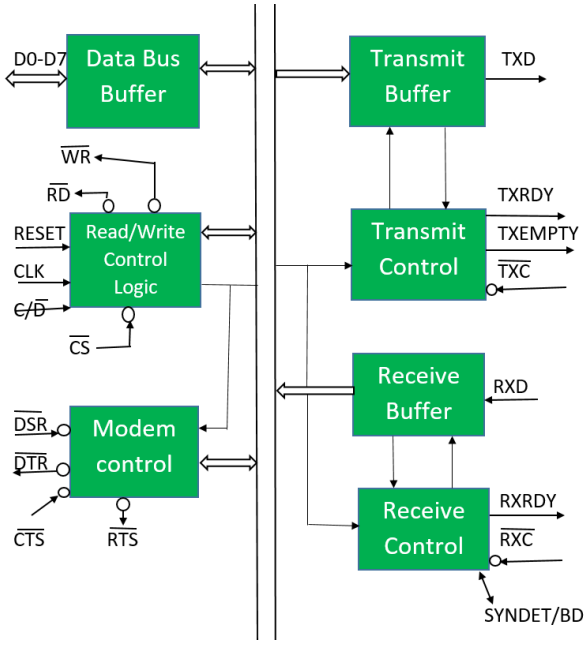



Fig. 3.15

The device used to perform the job of conversion from serial to parallel and parallel to serial is called USART or programmable communications interface (PCI).

The Intel 8251A is the industry standard USART, designed for data communications with Intel microprocessor families such as 8080, 85, 86 and 88. The 8251A converts the parallel data received from the processor on the D7-D0 data pins into serial data, and transmits it on TxD (transmit data) output pin of 8251A. Similarly, it converts the serial data received on RxD (receive data) input into parallel data, and the processor reads it using the data pins D7-D0.

**Features**

- Compatible with extended range of Intel microprocessors.
- It provides both synchronous and asynchronous data transmission.
- Synchronous and Asynchronous 5-8 bit characters.
- It has full duplex, double buffered transmitter and receiver.
- Detects the errors-parity, overrun and framing errors.
- All inputs and outputs are TTL compatible.
- Available in 28-pin DIP package.

**Transmitter Section**

The transmitter section accepts parallel data from microprocessor and converts them into serial data. The transmitter section is double

buffered; i.e., it has a buffer register to hold an 8-bit parallel data and another register called output register to convert the parallel data into serial bits. When output register is empty, the data is transferred from buffer to output register. Now the processor can again load another data in buffer register.

**Receiver Section**

The receiver section accepts serial data and converts them into parallel data. The receiver section is double buffered, i.e., it has an input register to receive serial data and convert to parallel, and a buffer register to hold the parallel data. When the RxD line goes low, the control logic assumes it as a START bit, waits for half a bit time and samples the line again. If the line is still low, then the input register accepts the following bits, forms a character and loads it into the buffer register. The microprocessor reads the parallel data from the buffer register.

## 3.11 ANALOG TO DIGITAL & DIGITAL TO ANALOG CONVERTER

### ANALOG TO DIGITAL CONVERTER

An electronic integrated **c**ircuit which transforms a signal from analog (continuous) to digital (discrete) form is called A-to-D converter. Digital data can be processed by computers for various purposes. There are following types of ADC.

- Simultaneous or Flash ADC
- Ramp type ADC
- Counter type ADC
- Dual slope ADC
- Successive approximation ADC

**Resolution and Step Size**

An 8-bit ADC has a resolution of 8 bits. There are 255 quantization levels. The step size can be calculated as follows:

$$Step\ Size = \frac{V_{CC}}{2^N - 1}$$

Where the $V_{CC}$ is the reference voltage of ADC with n-bit resolution. Below is table in which resolution versus step size for ADC is provided ($V_{CC}$ = 5V).

| n-bit | Number of steps | Step Size (mV) |
|---|---|---|
| 8 | 2^8 = 256 | 5/255 = 19.61 |
| 10 | 2^10 = 1024 | 5/1023 = 4.89 |
| 12 | 2^12 = 4096 | 5/4095 = 1.22 |
| 16 | 2^16 = 65536 | 5/65535 = 0.076 |

**Table 3.2**

**ADC0804**

Fig. 3.16 shows the pin-out of the ADC0804 converter. The ADC080X requires up to 100 µs to convert an analog input voltage into a digital output code.

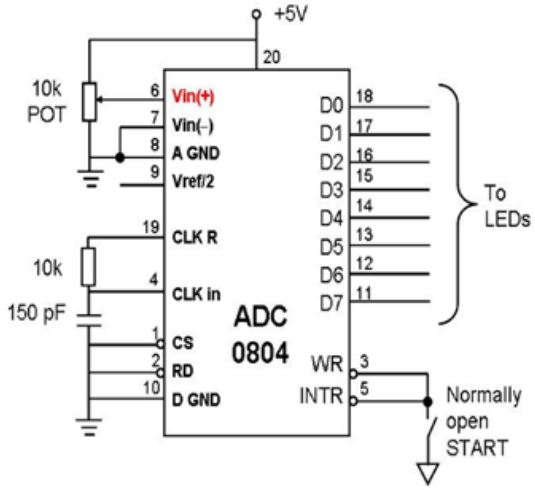



Fig. 3.16

The ADC0804 requires a clock source for operation. The clock can be an external clock applied to the CLK IN pin or it can be generated with an RC circuit. The range of clock frequencies is between 100 KHz to 1460 KHz. If the clock is generated with an RC circuit, we use the CLK IN and CLK R pins connected to an RC circuit. Frequency is calculated by the following equation:

$$f = 1/(1.1RC)$$

There are two analog inputs to the ADC0804: $V_{IN(+)}$ and $V_{IN(-)}$. Often the $V_{IN(-)}$ pin is connected to ground and the $V_{IN(+)}$ pin is used as the analog input to be converted to digital.

Input voltage pin used for the reference voltage is pin 9. If this pin is open, the analog input voltage for the ADC is ranged from 0 to 5 volts. This is optional input pin. It is used only when the input signal range is small.

| Vref/2 (V) | Vin (V) | Step Size (mV) |
|---|---|---|
| Not connected | 0 to 5 | 5/255 = 19.60 |
| 2.0 | 0 to 4 | 4/255 = 15.69 |
| 1.5 | 0 to 3 | 3/255 = 11.76 |
| 1.28 | 0 to 2.56 | 2.56/255 = 10.04 |
| 1.0 | 0 to 2 | 2/255 = 7.84 |
| 0.5 | 0 to 1 | 1/255 = 3.92 |

To operate the converter, the $\overline{WR}$ pin is pulsed with $\overline{CS}$ grounded to start the conversion process. the ADC0804 starts converting the analog input value of Vin to an 8-bit digital number. When the data conversion is complete, the $\overline{INTR}$ pin is forced low by the ADC0804. After $\overline{INTR}$ goes low, we make $\overline{CS} = 0$ and send a high-to-low pulse to the $\overline{RD}$ pin to get the data out at the D0-D7 data pins of the ADC.

## DIGITAL TO ANALOG CONVERTER

An electronic integrated circuit which transforms a signal from digital (discrete) to analog (continuous) form is called D-to-A converter. A DAC consists of a resistor network followed by a summing amplifier as shown in fig. 3.17.

There are following types of DAC.

i. Binary weighted resistors DAC
ii. R-2R ladder DAC

**Resolution and Step Size**

An 8-bit DAC has a resolution of 8 bits. There are 255 quantization levels. The step size can be calculated as follows:

$$Step\ Size = \frac{V_{CC}}{2^N - 1}$$

Where the $V_{CC}$ is the reference voltage of DAC with n-bit resolution.

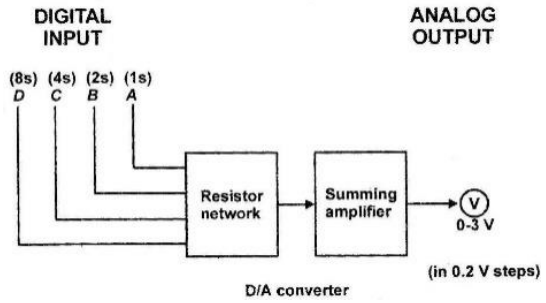



Fig. 3.17

**DAC0830**

Fig. 3.18 shows the pin-out of the DAC0830 converter. This device is an 8-bit converter. It requires up to 1 μs to convert a digital input code to an analog output voltage.

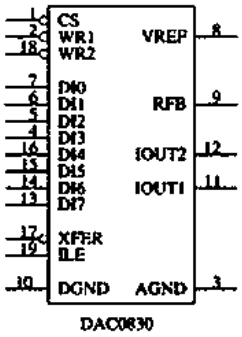



Fig. 3.18

This device has a set of eight data bus connections for the application of the digital input code, and a pair of analog outputs labeled $I_{OUT1}$ and $I_{OUT2}$ that are designed as inputs to an external operational amplifier.

Because this is an 8-bit converter, its output step voltage is +0.0196 V, if the reference voltage is -5.0 V. If an input of $(10010010)_2$ is applied to the device, the output voltage will be the step voltage times $(10010010)_2$, or, in this case, +2.862 V.

The converter has a reference input pin ($V_{REF}$) that establishes the full-scale output voltage.

# 3.13 COMPUTER PERIPHERALS

## 3.13.1 INTRODUCTION TO COMPUTER PERIPHERALS

Peripheral devices are those devices that are attached either internally or externally to a computer. These devices are commonly used to transfer data. Peripherals are commonly divided into three kinds: input devices, output devices, and storage devices.

An input device converts incoming data and instructions into a pattern of electrical signals in binary code that are understandable to a digital computer. Input devices include keyboard, mouse, trackball, joystick, scanner, and special pen with pressure-sensitive pad; microphones, webcams, and digital cameras.

An output device reverses the process, translating the digitized signals into a form understandable to the user. Output equipment includes video display terminals, printers, loudspeakers, and headphones.

Storage devices includes CD-ROM and DVD drives, flash memory drives, and external disk drives.

### 3.13.2 KEYBOARD

A computer keyboard is an input device used to enter characters and functions into the computer system by pressing buttons, or keys. A keyboard typically contains keys for individual letters, numbers and special characters, as well as keys for specific functions. A keyboard is connected to a computer system using a cable or a wireless connection.

A keyboard consists of a key matrix and a keyboard controller. The key matrix is a grid of circuits underneath the keys in the form of columns and rows. Keyboard Controller is an IC chip that is used to interface keyboard with the computer or system. It is the main controlling device of keyboard. It reads signals coming from the keyboard matrix and interpret them according to the main program written in its memory. Most popular Keyboard controllers are Intel 8042 and Intel 8048. A keyboard controller consists of a RAM, ROM, Processor and some other Input/ Output registers.

**Working principle of keyboard**

In all keyboards except capacitive models, each circuit is broken at a point below each key. When a specific key is pressed, a specific row and column got short circuited. The corresponding signals for those specific rows and columns are sent to keyboard controller. It processes that incoming signal and generates equivalent ASCII code and send to the main CPU. Thus, CPU gets the idea that which key is pressed and then display that specific key.

**Types of Keyboard**

Depending upon the working principle, there are two main types of keys, namely, capacitive and hard-contact. In both the key types, the circuit signals the processor to read and/or identify the character that has been pressed.

**Hard Contact (Mechanical) Key Board**

A hard contact key is attached with a metallic plate that helps in connecting the circuit board. When the hard contact key is pressed, it pushes a metallic plate, which in turn touches the metallic portion of the circuit plate. This overall process of completing a circuit result in a circuit flow, allowing the transfer of the message to the central processing unit (CPU), which is further transmitted to the software.

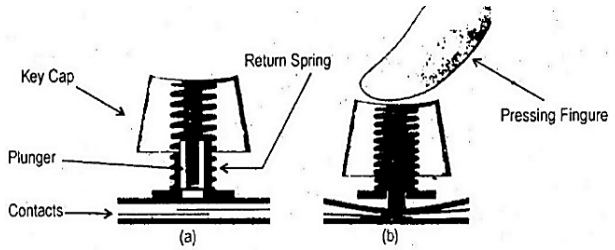



Fig. 3.19

**Merits:** These are simple and cheap. Any faulty key switch is replaceable.

**Demerits:** These are large in size. After some time, carbon layer is composed on contacts. During switch operation some power is consumed, so it cannot be used in laptops. The mechanical action of the switch causes some vibration, called bounce, which the processor filters out.

**Capacitive Key Board**

In this type of keyboard, an insulating material is placed between two metal plates. One plate is fixed on printed circuit board where other plate is moveable which is coated on a foam pad of a plunger as shown in fig 3.20.

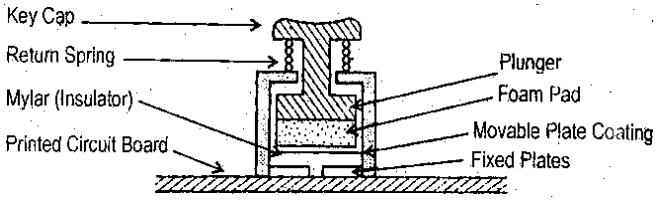



Fig. 3.20

When key is pressed, moveable plate is also pressed. The distance between plates decreases, as a result, capacitance increases. Microcontroller detects this pressed key during scanning process and generates equivalent ASCII code.

**Merits:** These switches are simple, durable and low cost. There is no mechanical contact, so there is no risk of getting damaged due to dust or oxidation.

**Demerits:** These switches are not repairable. A circuitry is needed to sense the change in capacitance.

**Membrane Key Board**

This is the type of mechanical keyboard. The key switch consists of three layers of plastic or rubber as shown in fig. 3.21.

The upper and bottom layers have row and column contacts respectively. Middle layer has a hole under the key.

When a key is pressed, both upper and bottom layers make a connection through the hole.

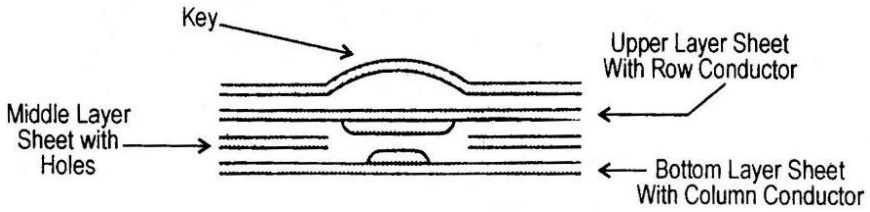



Fig. 3.21

## 3.13.2 VIDEO DISPLAY UNIT

A video display unit (VDU) is an output computer peripheral device. The VDU displays the information send by computer in the form of text or graphics (pictures) using cathode ray tube or liquid crystal display.

A VDU or Monitor basically consists of two sections.

- Display section
- Control section.

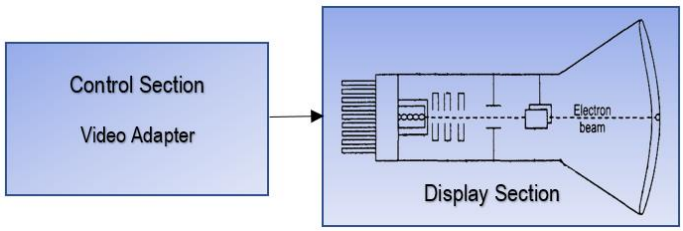



Fig. 3.22

Display section includes picture tube, high voltage circuit and scanning system. Control section is a circuit on mother board of a computer which tells the monitor how to draw a character or graphic on display. This is also called video card or video adapter. There are five important types of video adapter based on resolution and color support of display. The resolution tells how many pixels the monitor can display across and down the screen. The higher the resolution, the better the definition and the more expensive the monitor.

- **MDA (Monochrome Display Adapter):** It was designed by IBM for black and white monitors. It could only display text. The adapter designed by Hercules company (HGC) could display both text and graphics. The resolution of both adapters was 720 × 350.
- **CGA (Color Graphic Adapter):** It was also design by IBM for color monitors. It could display text in better quality but graphics resolution was 640 × 200 in four colors.

- **VGA (Video Graphics Adapter):** It was designed by IBM and used with PS/2 port. This card had 256KB onboard video memory. It was able to display sixteen colors with 640 × 480 resolution.
- **SVGA (Super VGA):** SVGA was a standard followed by some manufacturers. It can support 256 colors with 800×600 resolution, or 16 colors with 1024×768 resolution or 65536 colors with 640×480 resolution.
- **XGA (Extended Graphics Array):** This standard was designed by IBM which can support 256 colors with 1024×768 resolution or 65536 colors with 640×480 resolution.

Video image basically consists of large amount of data so it is necessary to store data before display on temporary basis. A graphics card shares a part of system memory for its functioning which is not suitable for 3D softwares and HD videos. If a dedicated graphics card is in use, it has its own memory thus freeing up memory from the computer.

### 3.13.3 MAGNETIC HARD DISKS

Computers use hard disks as the internal mass storage media. Hard disks consist of two or more platters made of aluminum alloy or a mixture of glass and ceramic covered with a magnetic coating. These platters are stacked to each other on a common shaft that turns the assembly at several thousand rpm. A hard disk drive is airtight sealed to keep the disks dust-free.

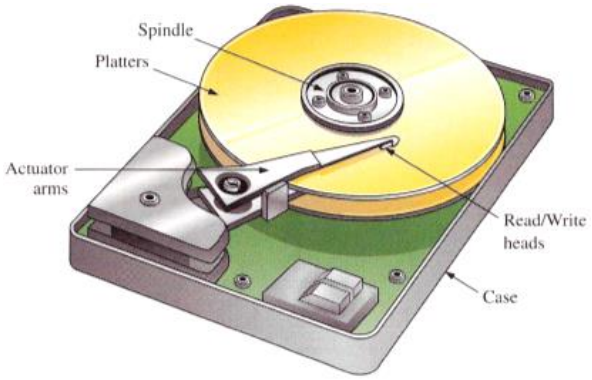



Fig. 3.23

There is a read/write head for both sides of each disk since data are recorded on both sides of the disk surface.

**Basic Read /Write Head Principles**

The read and write heads are usually combined in a single unit. The magnetic field produced by the write head according to the direction of a current pulse in the winding magnetizes a small spot on the disk surface in the direction of

the magnetic field. A magnetized spot of one polarity represents a binary 1, and one of the opposite polarities represents a binary 0.

When the magnetic surface passes a read head, the magnetized spots produce magnetic fields in the read head, which induce voltage pulses in the winding.

The polarity of these pulses indicates whether the stored bit is a 1 or a 0.

**Hard Disk Format**

Data is stored on the surface of a platter in sectors and tracks. The arrangement of tracks and sectors on a disk is known as the format. Tracks are concentric circles, and sectors are pie-shaped wedges on a track. A typical track is shown in yellow; a typical sector is shown in blue. A sector contains a fixed number of bytes -- for example, 256 or 512. Each track and sector has a physical address that is used by the operating system.

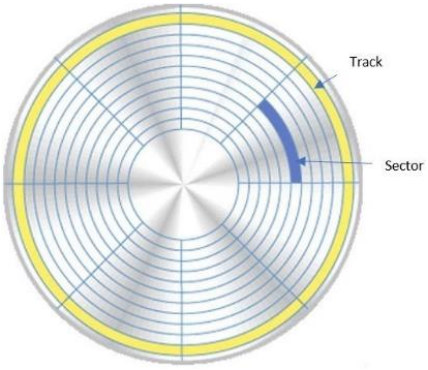


Fig. 3.24

## 3.13.4 POINTER

A pointer is an input device which is used to control the movement of the pointer or cursor on the screen. It can also be used for sending command signals to the computer, selecting items on the screen and drawing graphs.

The most important pointing devices are mouse, trackball, joy stick, touch pad, touch screen and light pen.

## 3.13.5 COMPUTER MOUSE

A mouse is a small hardware input device used by hand. It controls the movement of the cursor on the computer screen and allows users to move and select folders, text, files, and icons on a computer.

Computer mouse needs to put on a hard-flat surface to use. When the users move the mouse, the cursor moves in the same direction on the display screen. The name mouse is derived from its elliptical shape that looks a bit like a mouse.

**PARTS OF A COMPUTER MOUSE**

There are different parts of a computer mouse which are given below with their functions:

**Buttons:** Almost every mouse has two buttons, left and right. When a user clicks a button of the mouse, it communicates with the computer to perform an activity on the screen. A computer system understands the left or right-click based on the configuration of your mouse driver.

**Ball, laser, or LED:** A mouse, if it is a mechanical mouse, uses a ball and rollers, and an optical mouse uses a laser or LED. These parts allow the mouse to track the movement on an x-axis and y-axis directions and move the mouse cursor on the screen.

**Circuit board:** A circuit board is located inside of mouse chasses, which is used to transmit all mouse signal information, clicks, and other information. This board accepts input in the form of electronic signals when a user gives any instruction by clicking the mouse buttons, scrolling, etc.

**Mouse wheel:** Nowadays, computer mouses also include a wheel that is used to scroll the document page up and down direction.

**Cable/Wireless Receiver:** The corded mouse has a cable with a plug that connects to the computer. If the mouse is wireless, it requires a USB receiver to get the wireless signal and input it into the computer.

**Microprocessor:** It is a processor that is embedded on the circuit board of the mouse. All components of the mouse are not able to work without a microprocessor, as it is the brain of the mouse.

**TYPES OF MOUSE**

There are two main kinds of mice and they do this job in two different ways, either using a rolling rubber ball (in a ball-type mouse) or by bouncing a light off your desk (in an optical mouse).

**Mechanical Mouse:** It is a type of computer mouse, also called a ball mouse. It consists of a rubber or metal ball on its underside. The ball is touched with the two wheels that contain holes on its surface and it is covered with infrared pairs from two sides which contains sensors.

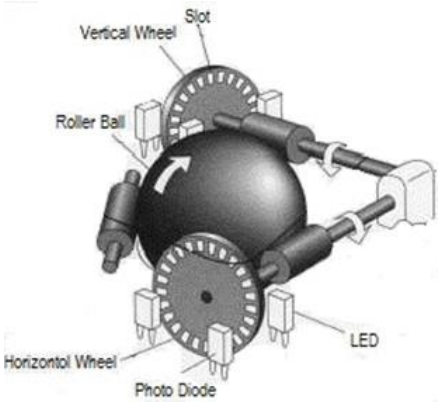


Fig. 3.26

As user moves the mouse, the ball moves the rollers that turn one or both of the wheels. If mouse moves the straight up or down, only the y-axis wheel turns; if you move to the right or left, only the x-axis wheel turns. And if you move the mouse at an angle, the ball turns both wheels at once.

Whenever a wheel moves the LED light goes through the holes and the sensor detects the movement of the mouse and through the IC it converts the signal into digital form and sends it to the CPU.

There are three PCB switches inside the mouse used for the left, right, and for the scrolling.

**Optical Mouse:** To detect the movement, an optical mouse uses a light source (LED or Laser), a photodetector (typically a CMOS sensor) and a digital signal processor (DSP). First, the light source produces light that shines onto the surface. The light is reflected back and picked up by the CMOS sensor forming an image of the local surface. Thousands of images are taken every second by the CMOS sensor and these images are sent to the DSP for analysis. The DSP compares these images to determine whether the mouse has moved, in what direction and at what speed. This information is then sent to the computer (through wire or wireless receiver), which updates the position of cursor on the screen accordingly.

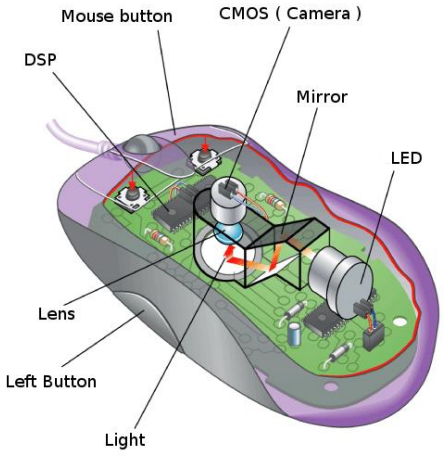



Fig. 3.27

## 3.13.6 PLOTTER

A plotter is a computer hardware device similar to a printer that is used to print vector graphics. Plotters use a pen, pencil, marker, or another writing instrument to draw multiple, continuous lines on paper instead of a toner. Multicolor plotters use different colored pens to draw different colors. Color

plots are made by using four pens cyan, magenta, yellow and black. Plotters differ from printers in that plotters use continuous lines to create images while printers use a collection of dots. Plotters are much slower than printers because of the mechanical motion necessary to draw detailed graphics using continuous lines.

Architects and product designers use plotters for technical drawings and computer-aided design purposes since plotters have the ability to create large images on oversized sheets of paper. Additionally, many garment and sign manufacturers use cutting plotters in which the plotter's pen is replaced with a sharp razorblade.

**Working of Plotters**

There are two main types of plotter for printing: Flatbed Plotter and Drum Plotter.

A flatbed plotter is a mechanical drafting device used with many CAD programs for designers. Paper remains stationary on a flat surface while a pen moves horizontally and vertically. This plotter can use many different pen colors to create graphics. The size of the picture is limited by the size of the flatbed plotter's surface.

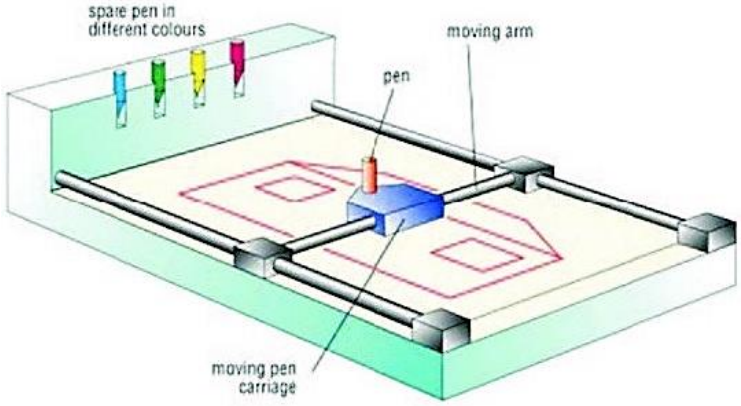



Fig. 3.28

The Drum Plotter moves the pen up and down, and the drum rotates the paper left and right. This enables drum plotters to have a smaller footprint than the final paper size. In addition, plotters can use more than one pen, allowing different colors to be drawn.

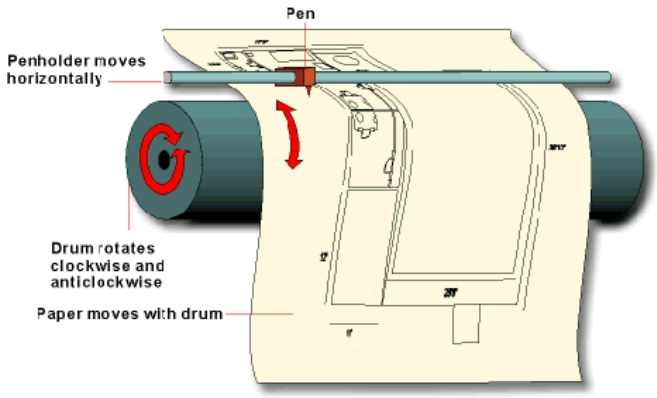



Fig. 3.29

**Applications of Plotters**

Computer Aided Design, architectural blueprint, building plan, line art, chart, electric circuit layout, banners and billboards, geographic layout and textile printing.

## 3.13.7 DIGITIZER & SCANNER

**DIGITIZER**

A digitizer is an input device used for converting pictures, maps and drawing into digital form for storage in computer.

Digitizer is a computer input device that has a flat surface and usually comes with a stylus. The size of the tablet, a square block, varies from 10 inches to 5 sq. ft. depending upon application. It enables the user to draw images and graphics using the stylus as we draw on paper with a pencil. The stylus senses a position through a transducer (pressure-sensitive switch) so that the movement of the stylus over the tablet causes a corresponding line on the CRT screen. The images or graphics drawn on the digitizer appear on the computer monitor or display screen. The software converts the touch inputs into lines and can also convert handwritten text to typewritten words.

Fig. 3.30

**SCANNER**

A scanner is an input device that reads and converts documents such as photos and pages of text into a digital signal that can be viewed and edited on a computer. Scanner uses light as an input source to convert an image into an electronic form that can be stored on the computer. Scanner accepts the source paper document, scans the document and translates it into an image to be stored on the computer. The quality of scan increases with the increase in resolution. There are three main types of a scanner. Flatbed, Sheet-fed and Handheld scanner.

**Working of flatbed scanner**

A scanner consists of a flat transparent glass bed under which the CCD sensors, lamp, lenses, filters and mirrors are fixed. The document has to be placed on the glass bed.

The scanner head consists of the mirrors, lens, CCD sensors and also the filter. A stepper motor under the scanner moves the scanner head from one end to the other in a constant path. The movement is controlled by a belt. When head has reached the other end the scanning of the document has been completed.

As the scan head moves under the glass bed, the light from the lamp hits the document and is reflected by a series of mirrors. In the end, the light passes through the scanner lens and reaches the CCD sensors. The CCD sensors convert the light to analog voltage according to its intensity. The analog voltage is changed to digital values by an ADC and sent to the logic board and transmitted back to the computer.

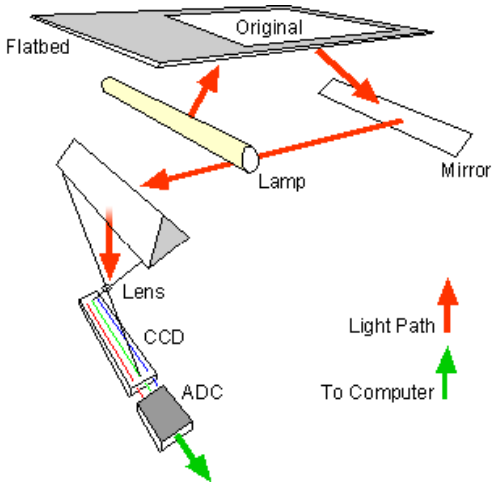



Fig. 3.31

Colour scanners have three light sources, one for each primary color—red, green, blue.

**Sheet-fed scanners** are similar to flatbed scanners except the document is moved and the scan head is immobile. A sheet-fed scanner looks a lot like a small portable printer.

**Handheld scanners** use the same basic technology as a flatbed scanner, but rely on the user to move them instead of a motorized belt. This type of scanner typically does not provide good image quality. However, it can be useful for quickly capturing text.

# EXERCISE

## *SECTION-I (MULTIPLE CHOICE QUESTIONS)*

1. 8086 microprocessor uses ……. memory interface.
   (a) 32 bit (b) 8 bit (c) 16 bit (d) 64 bit
2. $A_0$ signal is used to enable ……. bank of memory in 8086 interface.
   (a) Lower (b) Higher (c) Both (d) None of these
3. The number of horizontal and vertical pixels on a display screen is called
   (a) Accuracy (b) Precision (c) Resolution (d) None of these
4. In 8086 microprocessor, the high bank of memory is enabled by signal.
   (a) $A_0$ (b) $\overline{RD}$ (c) $\overline{BHE}$ (d) HOLD
5. Number of function keys in a computer keyboard
   (a) 10 (b) 12 (c) 14 (d) 16
6. A DRAM needs special external circuits to retain data;
   (a) Decoding (b) Storing (c) Refreshing (d) Addressing
7. Simultaneous converter is a type of converter;
   (a) Serial to Parallel (b) Parallel to Serial
   (c) Analog to Digital (d) Digital to Analog
8. An 8 bit DAC can generate ……. different levels.
   (a) 128 (b) 256 (c) 512 (d) 1024
9. A 64 K bit memory can be organized as
   (a) 64K × 1 (b) 8K × 8 (c) 16K × 4 (d) All of these
10. The data in ……. memory can be erased electrically.
    (a) Flash (b) Floppy (c) CD (d) Hard Disk
11. 8251 is an IC of;
    (a) UART (b) USART (c) ADC (d) DAC
12. 32-bit memory interface is used for;
    (a) 80186 (b) 80188 (c) 80286 (d) 80486

**ANSWER KEY**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | c | 2. | a | 3. | c | 4. | c |
| 5. | b | 6. | c | 7. | c | 8. | b |
| 9. | d | 10. | a | 11. | b | 12. | d |

## *SECTION-II (SHORT QUESTIONS)*

1. Define partial address decoding.

2. Define absolute address decoding.
3. Name two microprocessors, which use 8-bit memory interface.
4. What is the function of $\overline{BHE}$ signal in 16-bit memory interface?
5. Define the term “handshaking”.
6. Define memory mapped I/O decoding.
7. Define isolated I/O decoding.
8. Define programmable interval timer.
9. What is meant by UART?
10. Differentiate between half and full duplex transmission.
11. What is the purpose of programmable communication interface?
12. What are the disadvantages of mechanical switch key board?
13. What are the merits of capacitive keyboards?
14. Write the formula for “output voltage step” of 8-bit DAC.
15. Differentiate between sector and track of a hard disk.
16. Define pointer. Name any two pointers.
17. Define Digitizer.
18. Describe the function of a scanner.

## ***SECTION-III (LONG QUESTIONS)***

1. Explain Input Interface with help of diagram.
2. Draw block diagram of DAC and explain its working with table.
3. Discus construction and working of a key board.
4. Discuss construction and working of video display unit.
5. Explain different modes of 8254 Programmable Interval Timer.
6. Explain linear addressing decoding with help of diagram.

# CHAPTER 04
# INTERRUPTS

## INTERRUPT

An interrupt is a signal from a device attached to a computer or from a program within the computer that creates a temporary halt during program execution and allows peripheral devices to access the microprocessor.

Whenever an interrupt occurs the processor completes the execution of the current instruction and starts the execution of an Interrupt Service Routine (ISR) or Interrupt Handler. ISR is a short program that tells the processor what to do when the interrupt occurs. After the execution of ISR, control returns back to the main routine where it was interrupted.

## 4.1 BASIC INTERRUPT PROCESSING / STRUCTURE

The operation of an interrupt sequence on the microprocessor is called interrupt structure or interrupt processing. These steps are as follows.

- First device issues interrupt to CPU.
- The CPU finishes execution of current instruction.
- The CPU pushes the Flag register values on to the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.

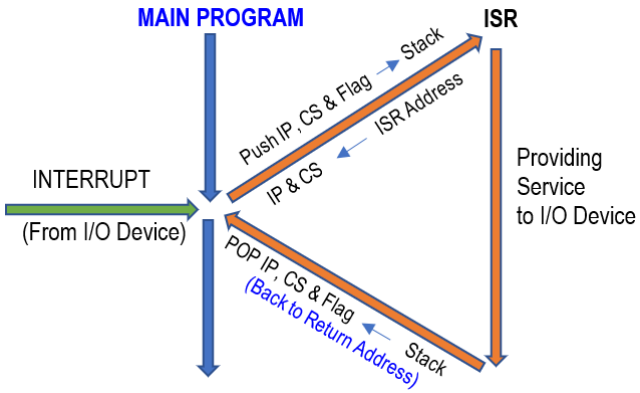


Fig. 4.1

- Interrupt flag and trap flag are reset to 0. This disables the INTR pin and the trap or single-step feature.
- IP is loaded from the contents of word location (Interrupt type). CS is loaded from the contents of next word location, so that the next instruction executes at the interrupt service procedure addressed by the interrupt vector.
- While returning from the interrupt-service routine by the Interrupt Return (IRET) instruction, the IP, CS and Flag registers are popped from the Stack and return to their state prior to the interrupt.

## 4.2 TYPES OF INTERRUPTS

The fig. 4.2 shows the types of interrupts we have in an 8086 microprocessor.

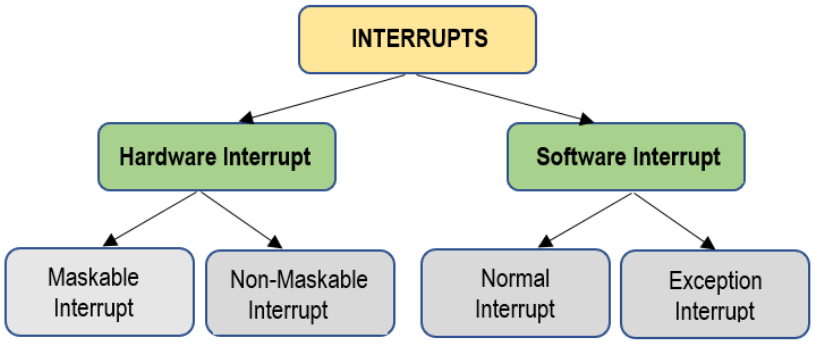



Fig. 4.2

## 4.3 HARDWARE INTERRUPTS

Hardware interrupts are those interrupts which are caused by any peripheral device by sending a signal through a specified pin to the microprocessor.
For example − In a keyboard if we press a key to do some action this pressing of the keyboard generates a signal that is given to the processor to do action.
There are two hardware interrupts in 8086 microprocessor. They are:
**(a) NMI (Non Maskable Interrupt) –** It is a single pin non maskable hardware interrupt which cannot be disabled. It is the highest priority interrupt in 8086 microprocessor.
**(b) INTR (Interrupt Request) –** It provides a single interrupt request and is activated by I/O port. This interrupt can be masked or delayed. It is a level triggered interrupt. It can receive any interrupt type, so the value of IP and CS will change on the interrupt type received.
Hardware interrupts are classified into two types which are as follows −

**Maskable Interrupt –** The hardware interrupts that can be delayed when a highest priority interrupt has occurred to the processor.

**Non Maskable Interrupt –** The hardware that cannot be delayed and immediately be serviced by the processor.

## 4.4 SOFTWARE INTERRUPTS

The interrupt signal that is caused by any internal system of the computer system and software programs is known as software interrupt.

Software interrupt is divided into two types. They are as follows −

**Normal Interrupts –** The interrupts that are caused by the software instructions are called software instructions.

**Exception –** Unplanned interrupts which are produced during the execution of some program are called exceptions. For example − while executing a program if we got a value that is divided by zero is called an exception.

There are 256 software interrupts in 8086 microprocessor. The instructions are of the format INT type where type ranges from 00 to FF. Some important software interrupts are:

(A) TYPE 0 corresponds to division by zero (0).

(B) TYPE 1 is used for single step execution for debugging of program.

(C) TYPE 2 represents NMI and is used in power failure conditions.

(D) TYPE 3 represents a break-point interrupt.

(E) TYPE 4 is the overflow interrupt.

The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interrupts from 32 to Type 255 are available for hardware and software interrupts.

**INT 3-Break Point Interrupt Instruction**

It is a 1-byte instruction. These instructions are inserted into the program so that when the processor reaches there, then it stops the normal execution of program and follows the break-point procedure.

**INTO - Interrupt On Overflow Instruction**

It is a 1-byte instruction and their mnemonic **INTO**. As the name suggests it is a conditional interrupt instruction, i.e., it is active only when the overflow flag is set to 1 and branches to the interrupt handler whose interrupt type number is 4. If the overflow flag is reset then, the execution continues to the next instruction.

## 4.5 PROGRAMMABLE INTERRUPT CONTROLLER (PIC)

A programmable interrupt controller (PIC) is an integrated circuit that helps a microprocessor to handle interrupt requests coming from multiple different sources which may occur simultaneously.

There are 5 hardware interrupts and 2 hardware interrupts in 8085 and 8086 respectively. For applications where processor interrupts are insufficient, we have to use external device called a Programmable Interrupt Controller (PIC). By connecting such a device, it is possible to increase the interrupt handling capacity of the microprocessor. 8259 combines the multi-interrupt input sources into a single interrupt output. Interfacing of single PIC provides 8 interrupts inputs from IR0-IR7.

For example, Interfacing of 8085 and 8259 increases the interrupt handling capability of 8085 microprocessor from 5 to 8 interrupt levels.

**Features of 8259 PIC microprocessor –**

- Intel 8259 is designed for Intel 8085 and Intel 8086 microprocessor.
- It can be programmed either in level triggered or in edge triggered interrupt level.
- It can handle eight priority interrupts. This is equivalent to providing eight interrupt pins on the processor in place of INTR pin.
- We can increase interrupt handling capability up to 64 interrupt level by cascading nine 8259 PIC's.
- We can mask individual bits of interrupt request register.
- Clock cycle is not required.

The fig. 4.3 shows the architectural representation of 8259 programmable interrupt controller:

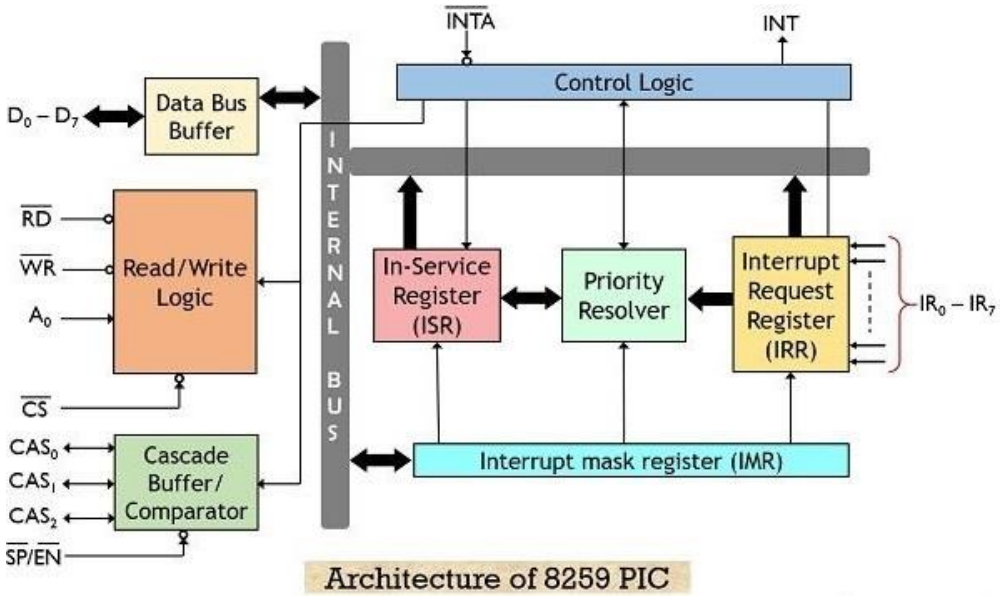



Fig. 4.3

It has an 8-bit of data bus. As we have already discussed that 8259 never services the interrupt, it simply forwards the interrupts to the microprocessor.

Thus, the above architecture has different units that combinedly functions to increase the interrupts handled by the processor. 8259 can be connected to 8085/8086 as shown in fig. 4.4.

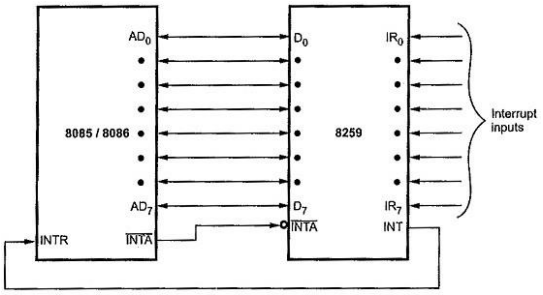



Fig. 4.4

# EXERCISE

## *SECTION-I (MULTIPLE CHOICE QUESTIONS)*

1. A signal sent to the processor that disturbs the current process is called;
   (a) Acknowledge (b) Test (c) Interrupt (d) Ready
2. An interrupt that can be disable by the programmer is called;
   (a) Maskable (b) Non maskable (c) Both a & b (d) None of these
3. An interrupt type which has lowest priority is;
   (a) Divide Error (b) Single Step (c) NMI (d) INTR
4. 8259A programmable interrupt controller can control hardware interrupts.
   (a) 8 (b) 10 (c) 12 (d) 14
5. In 8086 microprocessor the following has the highest priority among all type interrupts.
   (a) Type 255 (b) DIV 0 (c) NMI (d) Overflow
6. Which interrupt represents division by zero situation.
   (a) Type 0 (b) Type 1 (c) Type 2 (d) Type 3
7. INTR is a …… interrupt.
   (a) Maskable (b) Non-Maskable (c) Software (d) None of these
8. Which IC is programmable interrupt controller?
   (a) 8259 (b) 8031 (c) 8086 (d) 8051
9. An interrupt signal which is applied to pin of microprocessor is called;
   (a) Software Interrupt (b) Hardware Interrupt
   (c) Both a & b (d) None of these
10. INT 00 is a type of interrupt;
    (a) Software (b) Hardware (c) NMI (d) All of these
11. An interrupt signal which is applied in the form of an instruction is called;
    (a) Software Interrupt (b) Hardware Interrupt
    (c) Both a & b (d) None of these
12. NMI is a/an interrupt;
    (a) Level triggered (b) Edge triggered (c) Both a & b (d) None

**ANSWER KEY**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | c | 2. | a | 3. | d | 4. | a |
| 5. | c | 6. | a | 7. | b | 8. | a |
| 9. | b | 10. | a | 11. | a | 12. | b |

## *SECTION-II (SHORT QUESTIONS)*

1. Define partial address decoding. Define the term interrupt.
2. Define absolute address decoding. What is an interrupt vector?
3. Define hardware interrupt.
4. Define maskable Interrupt.
5. Define non-maskable Interrupt.
6. Write the function of signals INTR and $\overline{\text{INTA}}$.
7. Define software interrupt.
8. List five interrupt instructions for microprocessor.
9. What is the function of INTO instruction?
10. What is the function of INT3 instruction?
11. Define error interrupt.
12. What is meant by programmable interrupt controller?
13. What is meant by Interrupt Structure?
14. Enlist the various hardware interrupts of 8086.
15. How many software interrupts are available for 8086?

## *SECTION-III (LONG QUESTIONS)*

1. Define interrupt and describe software and hardware interrupts in detail.
2. Describe all steps of a microprocessor when it receives an interrupt signal.
3. Explain the operation of 8259 programmable interrupt controller.

# CHAPTER 05
# DIRECT MEMORY ACCESS

## 5.1 DIRECT MEMORY ACCESS

Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations.

We have two other methods of data transfer, programmed I/O and Interrupt driven I/O.

In programmed I/O, the processor keeps on scanning whether any device is ready for data transfer. If an I/O device is ready, the processor fully dedicates itself in transferring the data between I/O and memory and can't get involved in any other activity during data transfer. This is the major drawback of programmed I/O.

In Interrupt driven I/O, whenever the device is ready for data transfer, then it raises an interrupt to processor. Processor completes executing its ongoing instruction and saves its current state. It then switches to data transfer which causes a delay. So, it is also not an effective way of data transfer.

The above two modes of data transfer are not useful for transferring a large block of data. The DMA controller completes this task at a faster rate and is also effective for transfer of large data block.

## 5.2 DMA CONTROLLER OPERATION

Whenever there exists a need for transfer of data between peripherals and main memory, then first the data is given to the processor by the input device and then the processor further transfers the data to the memory. At the time of data transfer operation, the processor cannot be able to execute any other operation.

So, to enhance the performance of the processor, an external device is used that can manage data transfer operation between peripherals and memory with least CPU utilization. This method of data transfer is known as direct

memory access and the external device used for this purpose is known as DMA controller.

In the idle cycle of the system, initially when the system gets on, then the processor has control over the system buses, as the switch are connected with the X position.

Whenever an I/O device needs to transfer the data to or from the memory, it sends a request (DRQ) to DMA controller. On receiving data transfer request, the controller sends HOLD request to the CPU and waits for HLDA which is hold acknowledge by the CPU.

CPU receives the Hold request (HRQ) from DMA controller and leaves the control over all the buses (i.e., data, control and address bus) and sends the Hold acknowledgement (HLDA) to DMA controller.

Hence now the processor gets free from any data transfer operation until an interrupt is generated by the DMA controller about the completion of the transfer. The fig. 5.1 represents a system having a DMA controller:

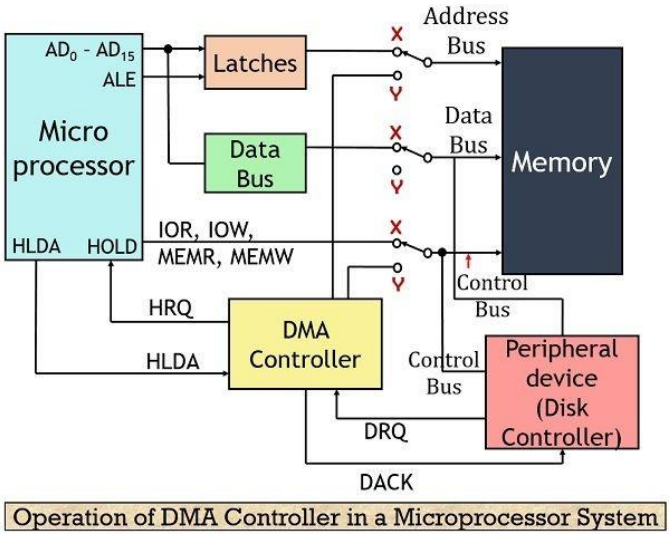



Fig. 5.1

On receiving HLDA signal, the control over the buses is given to the DMA controller as the switch position now changes from X to Y.

After receiving the Hold acknowledgement (HLDA), DMA controller acknowledges I/O device (DACK) that the data transfer can be performed and DMA controller takes the charge of the system bus and transfers the data to or from memory.

When the data transfer is accomplished, the DMA controller generates an interrupt by varying switch position from Y to again X. This indicates the microprocessor about the completion of the data transfer operation and the processor can take control over the bus again and start processing where it has left.

**Advantages:**

- Transferring the data without the involvement of the processor will speed up the read-write task.
- DMA reduces the clock cycle requires to read or write a block of data.
- Implementing DMA also reduces the overhead of the processor.

**Disadvantages:**

- As it is a hardware unit, it would cost to implement a DMA controller in the system.
- Cache coherence problem can occur while using DMA controller.

## 5.3 SHARED BUS SYSTEM

In the time-shared common bus, there are numerous processors linked by a common direction to the memory unit in a common-bus multiprocessor system. The fig. 5.2 shows the organization of time-shared common buses for three processors.

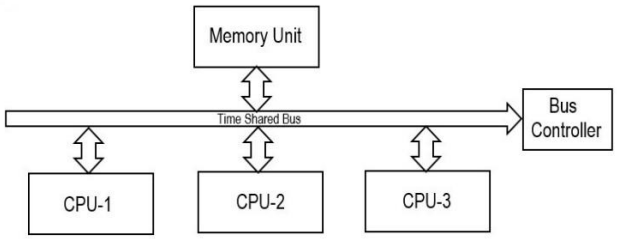



Fig. 5.2

There is only one processor that can interact with the memory of another processor. The processor that is in control of the bus at the time implements transfer operations. Any processor that needs to start a transfer should first check the availability condition of the bus.

As all processors share a common bus, the system may display some transfer conflicts. The incorporation of a bus controller that creates priorities among the requesting units helps in resolving the transfer conflicts.

There is a restriction of one transfer at a time for a single common-bus system. This means that other processors are busy with internal operations or remain useless waiting for the bus when one processor is interacting with the memory.

The system processors are kept busy by the execution of multiple separate buses, to allow multiple bus transfers simultaneously. The fig. 5.3 shows a more economical execution of a dual bus structure for multiprocessors.

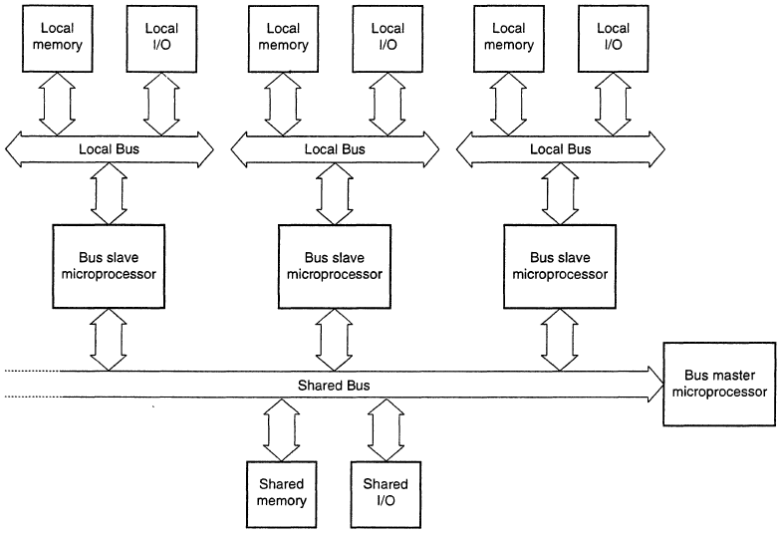



Fig. 5.3

In a multiprocessing and multitasking environment, each microprocessor accesses two buses: the local bus and the remote or shared bus as shown in fig. 5.3.

The **local bus** is the bus that is immediately available to the microprocessor. The local bus contains the local memory and I/O. The local memory and local I/O are accessed by the microprocessor that is directly connected to them without any special protocol or access rules.

The **remote (shared) bus** contains memory and I/O that are accessed by any microprocessor in the system. It is connected to all microprocessors in the system. The shared bus is used to exchange data between microprocessors in the system. A shared bus may contain memory and I/O devices that are accessed by all microprocessors in the system. Access to the shared bus is controlled by some form or arbiter that allows only a single microprocessor to access the system's shared bus space.

Access to the shared bus for the remote bus master is accomplished via a bus arbiter. The bus arbiter functions to resolve priority between bus masters and allows only one device at a time to access the shared bus.

The 8086 and 8088 microprocessors use the 8289-bus arbiter. The 80286 uses the 82289-bus arbiter and the 80386/80486 use the 82389. The Pentium and Pentium Pro directly support a multiuser environment.

# EXERCISE

## *SECTION-I (MULTIPLE CHOICE QUESTIONS)*

1. Following two signals are used in DMA operation.
   (a) INTR & INTA (b) HOLD & HLDA
   (c) WAIT & READY (d) HOLD & INTA
2. Following is a DMA controller IC
   (a) 8086 (b) 8051 (c) 8237 (d) 8255
3. In a DMA read operation the data is transferred
   (a) from I/O to memory (b) from memory to I/O
   (c) from memory to memory (d) from I/O to I/O
4. In a DMA write operation the data is transferred
   (a) from I/O to memory (b) from memory to I/O
   (c) from memory to memory (d) from I/O to I/O
5. Which method bypasses the microprocessor for data transfer?
   (a) DMA (b) Interrupt (c) Handshaking (d) Polling
6. DMA stands for
   (a) Data Memory Access (b) Data Memory Allocation
   (c) Direct Memory Allocation (d) Direct Memory Access
7. Which microprocessor pin is used to acknowledge a DMA operation?
   (a) READY (b) HOLD (c) HLDA (d) WAIT
8. Number of channels available in 8237 DMA controller
   (a) 2 (b) 4 (c) 3 (d) 5
9. The pin of microprocessor is used to request DMA process.
   (a) WR (b) RD (c) ALE (d) HOLD
10. The bus that is immediately available to the microprocessor is called
    (a) Remote bus (b) Local bus (c) Shared bus (d) All of these

**ANSWER KEY**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. | b | 2. | c | 3. | b | 4. | a | 5. | a |
| 6. | d | 7. | c | 8. | b | 9. | d | 10. | b |

## *SECTION-II (SHORT QUESTIONS)*

1. Define Direct Memory Access operation.
2. Describe need of DMA operation.
3. Which two signals are used in DMA operation?
4. What is the advantage of DMA operation?
5. What is the effect on buses of microprocessor during DMA operation?

6. What is the function of current address register in DMA controller?
7. Define shared bus system.
8. What is the advantage of Shared Bus System?
9. What is the function of the 8289 bus arbiter?
10. What is meant by local bus?

## ***SECTION-III (LONG QUESTIONS)***

1. Draw the block diagram of DMA Controller and describe its operation.
2. Explain operation of shared bus system with help of diagram.

# CHAPTER 06
# BUS INTERFACE

## INTRODUCTION TO BUS INTERFACES

A computer bus is a common pathway through which information is connected from one component to another. A computer bus is a set of parallel conductors, which may be conventional wires, copper tracks on a Printed Circuit Board, or microscopic aluminum trails on the surface of a silicon chip.

**Types of Computer Bus**

A modern-day computer system can be viewed as comprising just two types of buses. These are the System Bus and the I/O Bus or Expansion Bus.

**System Bus**

The system bus or internal bus is a pathway composed of cables and connectors used to carry data between a computer microprocessor and the main memory. The bus combines the functions of a data bus to carry information, an address bus to determine where it should be sent, and a control bus to determine its operation.

**I/O Bus or Expansion Bus**

This is a bus made up of the electronic pathways that connect the different external devices, such as a printer, storage, monitors, keyboard, mouse, etc. Thus I/O bus connects various peripheral devices to the CPU – these are connected to the system bus via a 'bridge' implemented in the processor's chipset.

Other names for the I/O bus include expansion bus, external bus or host bus. An expansion bus typically comprises a series of slots on the motherboard into which cards are inserted. The common expansion bus types include:

**Expansion Bus Types**

- **ISA –** Industry Standard Architecture
- **EISA –** Extended Industry Standard Architecture

- **MCA –** Micro Channel Architecture
- **VESA –** Video Electronics Standards Association
- **PCI –** Peripheral Component Interconnect
- **PCMCIA –** Personal Computer Memory Card Industry Association
- **AGP –** Accelerated Graphics Port
- **SCSI –** Small Computer Systems Interface.
- **USB –** Universal Serial Bus
- **FireWire**

## 6.1 INDUSTRY STANDARD ARCHITECTURE (ISA BUS)

This is the most common type of early expansion bus, which was designed for use in the original IBM PC. The IBM PC-XT used an 8-bit bus design. This means that the data transfers take place in 8-bit chunks across the bus. The ISA bus ran at a clock speed of 4.77 MHz

For the 80286-based IBM PC-AT, an improved bus design that could transfer 16-bits of data at a time was announced. The 16-bit version of the ISA bus is sometimes known as the AT bus (AT-Advanced Technology).

The improved AT bus also provided a total of 24 address lines, which allowed 16MB of memory to be addressed. The AT bus was backward compatible with its 8-bit predecessor and allowed 8-bit cards to be used in 16-bit expansion slots.


Fig. 6.1 Five 16-bit and one 8-bit ISA slots on a motherboard

| 8-Bit ISA card (XT-Bus) | 16-Bit ISA Card (AT-Bus) |
|---|---|
| 8-bit data interface | 16-bit data interface |
| 4.77 MHZ bus | 8-MHZ bus |
| 62-pin connector | 62-pin connector |
| | 36-pin AT extension connection |

## 6.2 EXTENDED ISA & VESA LOCAL BUSES

**Extended Industry Standard Architecture**

EISA was developed by a group of manufactures. It was designed to use a 32-bit data path and provided 32 address lines giving access to 4GB of memory. EISA offered a disk-based setup for the cards, but it still ran at 8MHz in order for it to be compatible with ISA.

If an ISA card is placed in an EISA slot it will use only the top row of connectors, whereas a full EISA card uses both rows.

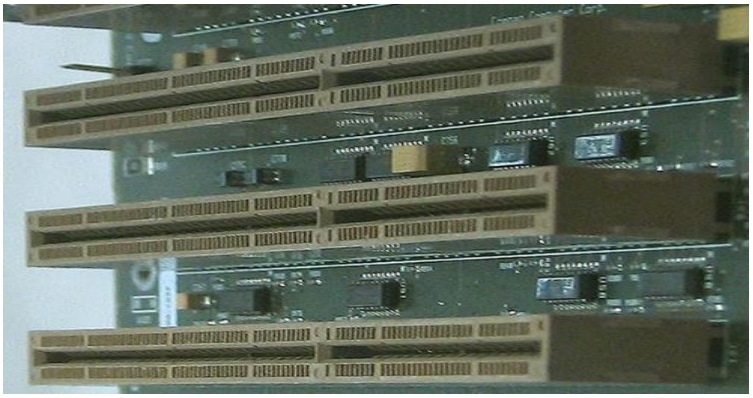


Fig. 6.2

EISA provided much better performance than the ISA equivalent. Apart from being able to transfer 4 bytes of data simultaneously, it offers bus mastering; a technology that placed a mini-processor on each expansion card. These mini-processors controlled much of the transfer allowing the CPU to perform other tasks.

**Video Electronics Standards Association (VESA Bus)**

It is also known as the Local bus or the VESA-Local bus. VESA was invented to solve the problem of proprietary technology where different manufacturers were attempting to develop their own buses. It was developed with the main aim of providing enhanced video performance on computers. It was thus aimed at standardizing PC's video specifications.

Fig. 6.3 Seven ISA slots. Six are 16-bit ISA (longer – with middle black sections), three additionally have a VLB slot (leftmost brown sections)

The VL Computer bus provided a 32-bit data path and run at 25 or 33MHZ. It ran at the same clock frequency as the host CPU. But this became a problem as processor speeds increased. The faster the peripherals are required to run, the more expensive they are to manufacture.
It was difficult to implement the VL-Bus on newer chips such as the 486s and the new Pentiums and so eventually the VL-Bus was superseded by PCI.

**Features of the VESA Local Bus Card**

- 32-bit interface
- 62/36-pin connector
- 90+20 pin VESA local bus extension
- The VESA design was backward compatible with the older ISA cards.

## 6.3 PERIPHERAL COMPONENT INTERCONNECT (PCI) BUS

PCI brought about changes in the design of the bus and its communication with the processor bus. It was developed by Intel and launched as the expansion bus for the Pentium processor in 1993. It is a local bus like VESA meaning that it connects the CPU, memory, and peripherals to a wider, faster data pathway.

PCI supports both 32-bit and 64-bit data width; therefore it is compatible with 486s and Pentiums. The bus data width is equal to the processor, for instance, a 32-bit processor would have a 32 bit PCI bus and operate at 33MHz.
PCI was used in developing Plug and Play (PnP). All PCI cards support plug and play specifications. This means you plug a new card into the computer, power it on and it will “self-identify” and “self-specify” and start working without manual configuration using jumpers.

Unlike VESA, PCI supports bus mastering i.e. the bus has some processing capability and therefore the CPU spends less time processing data. Most PCI cards are designed for 5V, but there are also 3V and dual-voltage cards. Keying slots are used to differentiate 3V and 5V cards and slots to ensure that a 3V card is not slotted into a 5V socket and vice versa.

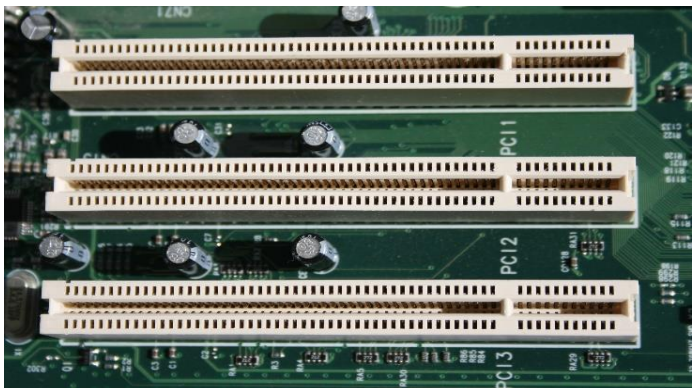



Fig. 6.4 Three 32-bit PCI slots

**The PCI Express Bus**

The PCI Express bus comes in several versions (1X, 2X, 4X, 8X, 12X, 16X and 32X), which provide throughputs of between 250 Mb/s and 8 Gb/s, or close to 4 times the peak throughput of AGP 8X ports. Because its manufacturing cost is that similar to that of the AGP port, the PCI Express bus will progressively replace the former.

## 6.4 THE UNIVERSAL SERIAL BUS (USB)

A Universal Serial Bus (USB) is a latest interface that enables communication between devices and a host controller such as a personal computer (PC) or smartphone. The universal serial bus connects external peripherals such as a mouse, keyboards, joysticks, printers, flash drives, scanners, and digital cameras to the computer. Because of its wide variety of uses, including support for electrical power, the USB has replaced a wide range of interfaces like the parallel and serial port. A single USB port can connect up to 127 peripheral devices. This computer bus also supports Plug-and-Play installation and hot plugging.

**USB CONNECTORS**

There are several types of USB connectors. In the past the majority of USB cables were one of two types, type A and type B. Later the USB C Type was introduced to provide a better data transfer performance. As a result, there are many USB connector types: USB Type A, USB Type B, Mini-A, Mini-B, Micro-A, Micro-B and Micro-AB. Types A and B have 4 pins within the connector, Mini and Micro A and B connectors generally have five pins.

The USB 2.0 standard is type A; it has a flat rectangle interface that inserts into a hub or USB host which transmits data and supplies power. A keyboard or mouse are common examples of a type A USB connector.

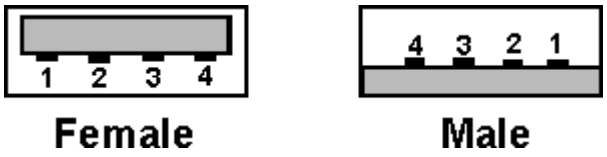



Fig. 6.5

A type B USB connector is square with slanted exterior corners. It is connected to an upstream port that uses a removable cable such as a printer. The type B connector also transmits data and supplies power.

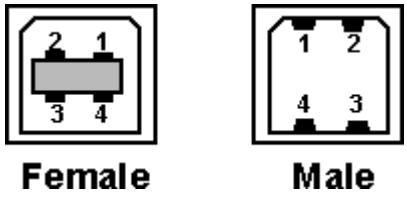



Fig. 6.6

The basic USB pinouts for the connectors Type-A & B are given in the table below.

| PIN | COLOUR | SIGNAL |
|---|---|---|
| 1 | Red | $V_{BUS}$ (4.75 – 5.25 V) |
| 2 | White | Data - |
| 3 | Green | Data + |
| 4 | Black | Ground |

Today, newer connectors have replaced old ones, such as the Mini-USB (or Mini-B), that has been abandoned in favor of the Micro-USB and USB-C cables. Micro-USB cables are usually used for charging and data transfer between smartphones, video game controllers, and some computer peripherals. Micro-USB are being slowly replaced by type-C connectors, which are becoming the new standard for Android smartphones and tablets.

**Various USB Data Transfer Rates**

**USB 1.0**

- Low speed: 1.5 Mbps (Megabits per second)
- Full speed: 12 Mbps (Megabits per second)

**USB 2.0**

- High Speed

- Throughput up to 60 MB per second (480 Mbits per second)
- Support battery charging

**USB 3.0**

- Very High Speed
- 10 times faster than USB 2.0
- Supports data rate up to 512 MB per second (4 Gbit per second)
- Uses two unidirectional data paths one to receive and the other to transmit data

**USB 3.1**

- USB 3.1 Gen 1 supports speeds of up to 5Gbit/s,
- USB 3.1 Gen 2 supports speeds of up to 10Gbit/s.

## 6.5 ACCELERATED GRAPHICS PORT COMPUTER BUS (AGP)

The need for high quality and very fast performance of video on computers led to the development of the Accelerated Graphics Port (AGP). The AGP port is connected to the CPU and operates at the speed of the processor bus. This means that video information can be sent more quickly to the card for processing.

The AGP uses the main PC memory to hold 3D images. In effect, this gives the AGP video card an unlimited amount of video memory. To speed up the data transfer, Intel designed the port as a direct path to the PC's main memory. The data transfer rate ranges from 264 Mbps to 528mbpn, 800 Mbps up to 1.5 Gbps.

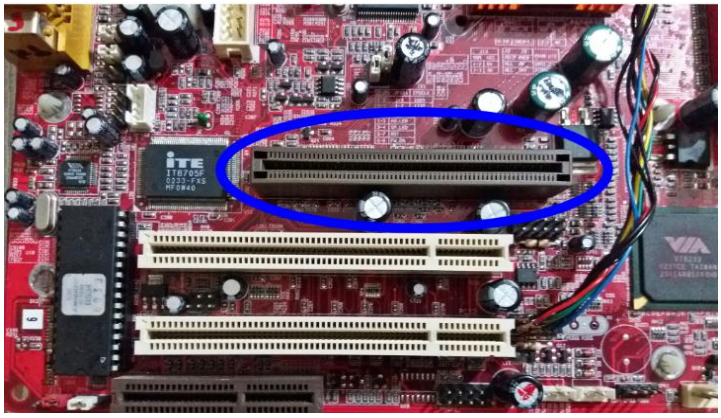


Fig. 6.7 AGP (Accelerated Graphics Port)

# EXERCISE

## *SECTION-I (MULTIPLE CHOICE QUESTIONS)*

1. ISA bus supports …… bits data.
   (a) 8 (b) 16 (c) 24 (d) 32
2. PCI bus operates at;
   (a) 8 MHz (b) 16 MHz (c) 33 MHz (d) 128 MHz
3. ISA bus operates at;
   (a) 4 MHz (b) 8 MHz (c) 12 MHz (d) 16 MHz
4. The speed of USB 3.0 is approximately greater than USB 2.0.
   (a) 3 times (b) 5 times (c) 10 times (d) 15 times
5. PCI bus supports bits data;
   (a) 16 (b) 32 (c) 64 (d) Both b & c
6. Data transfer speed of USB 3.0 is
   (a) 3200 Mbps (b) 4800 Mbps (c) 6800 Mbps (d) 7560 Mbps
7. ISA card is not used for
   (a) EPROM (b) Flash (c) RAM (d) None of these
8. EISA bus is used in bit standard;
   (a) 8 (b) 16 (c) 32 (d) All of these

**ANSWER KEY**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | b | 2. | c | 3. | b | 4. | c |
| 5. | d | 6. | b | 7. | c | 8. | c |

## *SECTION-II (SHORT QUESTIONS)*

1. Name different interfaces used in computer.
2. Why ports are needed in the process of interfacing?
3. Define ISA (Industry Standard Architecture) Bus.
4. Write applications of ISA bus.
5. Define EISA bus.
6. What is the advantage of EISA bus over ISA bus?
7. Define PCI (Peripheral Components Interconnect) Bus.
8. What is the application of Accelerated Graphic Port?
9. What is the purpose of Universal Serial Bus?
10. How many USB can be connected through a USB bus?

## *SECTION-III (LONG QUESTIONS)*

1. Discuss Extended ISA bus and VESA local buses.
2. Discuss Universal Serial Bus in detail.

# CHAPTER 07
# THE PENTIUM MICROPROCESSORS

## 7.1 SUMMARY OF GROWTH FROM 808186 TO 80486.

### The 80186 Microprocessor

The Intel 80186 was an improved version of the 8086 microprocessor and introduced in 1982. 80186 is a 16-bit microprocessor with a 16-bit data bus and a 20-bit address bus. It has a programmable peripheral device integrated into the same package. The instruction set of the 80186 is a superset of the instruction set of the 8086. The term super-set means that all of the 8086 instructions will execute properly on an 80186, but the 80186 has 10 new instruction types. The 80188 variant, with an 8-bit external data bus was also available.

- 16-bit Microprocessor
- 1MB Main Memory
- Initially 6 MHz Clock Frequency

### The 80286 Microprocessor

The 80286 microprocessor (also a 16-bit architecture microprocessor) was introduced in 1983, almost identical to the 8086 and 8088 except it addressed a 16M byte memory system. The instruction set of the 80286 was also almost identical to the 8086 and 8088 except for a few instructions. The clock speed of the 80286 was increased, so it executed some instructions in as little as 250 ns (4.0 MIPs).

- 16-bit Microprocessor
- 16MB Main Memory
- 4.0 MIPS (250nS / 8MHz)

### The 80386 Microprocessor

In 1986, Intel released the 80386 microprocessor. It was Intel's first practical 32-bit microprocessor that contained a 32-bit data bus and a 32-bit memory address. Through these 32-bit buses, the 80386 addressed up to 4G bytes of memory. The instruction set of the 80386 microprocessor was upward compatible with the earlier 8086, 8088, and 80286 microprocessors.

- 32-bit Microprocessor
- 4GB Main Memory
- 12-33MHz
- Memory Management Unit added
- Variations: DX, EX, SL, SLC (Cache) and SX

### THE 80486 MICROPROCESSOR

In 1989, Intel released the 80486 microprocessor, which incorporated an 80386-like microprocessor, an 80387-like numeric coprocessor, and an 8K byte cache memory system into one integrated package. The internal structure of the 80486 was modified from the 80386 so about half of its instructions executed in 25ns (50 MIPs).

- 32-bit Microprocessor
- 4GB Main Memory
- 20-50MHz later 66 and 100MHz
- Variations: SX, DX2, DX4

## 7.2 INTRODUCTION TO THE PENTIUM MICROPROCESSORS

### THE PENTIUM MICROPROCESSOR

The Pentium processor is normally used as a 5th-generation personal computer microprocessor invented by Intel Corporation in 1993. It was a 32-bit superscalar microprocessor similar to the 80386 and 80486 microprocessors. A special category of microprocessors through which more than one instruction gets executed in one clock cycle is called superscalar processors. This microprocessor was originally labeled the P5 or 80586. This processor operates with 5V. The two introductory versions of the Pentium operated with a clocking frequency of 60 MHz and 66 MHz and a speed of 110 MIPs. The highest clock rating provided by this microprocessor was 233 MHz. Another difference was that the cache size was increased to 16K bytes. The Pentium contains an 8K byte instruction cache and an 8K byte data cache. This microprocessor includes approximately 3.1 million transistors, a 32-bit address bus that provides 4 GB of physical memory space. The data bus width increased from the 32-bits found in the 80386 and 80486 to a full 64-bits. This processor includes 2- instruction units.

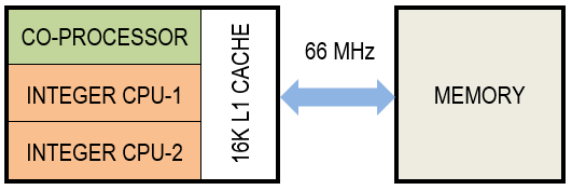



Fig. 7.1

- 32-bit Microprocessor, 64-bit data bus and 32-bit address bus
- 4GB Main Memory
- 60, 66, 90 MHz
- Double-clocked Pentium at 120 MHz and 133 MHz
- Fastest version produced 233 MHz
- 16 KB L1 cache
- Dual Integer processors

## THE PENTIUM PRO MICROPROCESSOR

The Pentium Pro processor is a 6th generation x86 silicon transistor developed by Intel in 1995. This processor is represented by P6. This processor operates at 3.1 V – 3.3 V. The clock speed of the Pentium-Pro processor runs at 150 MHz, 166 MHz, 180 MHz, or 200 MHz with a 60 MHz or 66 MHz external bus CLK. Pentium-Pro includes 5.5 million transistors. Pentium-Pro includes a 2×8 Kbyte L1 cache & a 256 Kbyte L2 cache. The address bus of this processor is 36-bit. This processor includes 6- instruction units. This processor includes the architecture of a 12-stage decoupled super pipeline that utilizes an instruction pool.

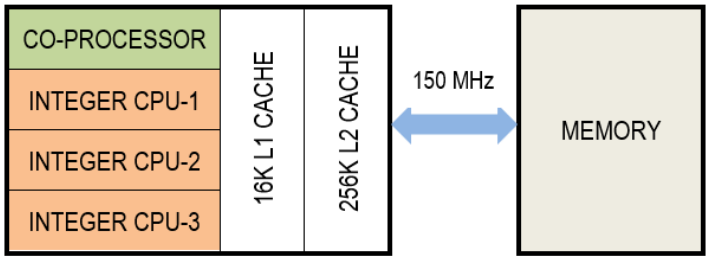



Fig. 7.2

- 32-bit Microprocessor, 64-bit data bus and 36-bit address bus
- 64GB Main Memory
- Starts at 150MHz
- 16 KB L1 cache; 256 KB L2 cache
- Three Integer processors

## THE PENTIUM-II MICROPROCESSOR

After the Pentium Pro processor, Pentium II was developed by Intel in 1997. Pentium II is manufactured based on P6 micro-architecture and it is a sixth-generation x86 compatible microprocessors. This processor core contains 7.5 million transistors and it is an improved version of the first P6-generation core of the Pentium Pro. All features of the Pentium Pro have been incorporated with Pentium II. This processor has a larger cache. As this processor can operate at 2.8 V, the power consumption is reduced significantly. Pentium II can support MMX instructions for enhanced floating-point operation. The features of Pentium II processor are given below:

The L2 cache size is increased to 512 KB. The L2 cache operates at half of the processor's clock frequency, whereas the L2 cache of Pentium Pro operates at the same frequency as the processor.

The Pentium-II has a 32 KB L1 cache. This processor has separate 16 KB L1 data and 16 KB L1 instruction caches.

The Pentium-II is cheaper to manufacture because of the separate slower L2 cache memory. The improved 16-bit performance and MMX support make it suitable for Windows operating systems and multimedia applications.

The Pentium-II is packaged in a slot-based module rather than a CPU socket.

The Pentium-II processor clock speeds are 233, 266, 300, 350, 400 or 450 MHz with a 66 MHz front side bus.

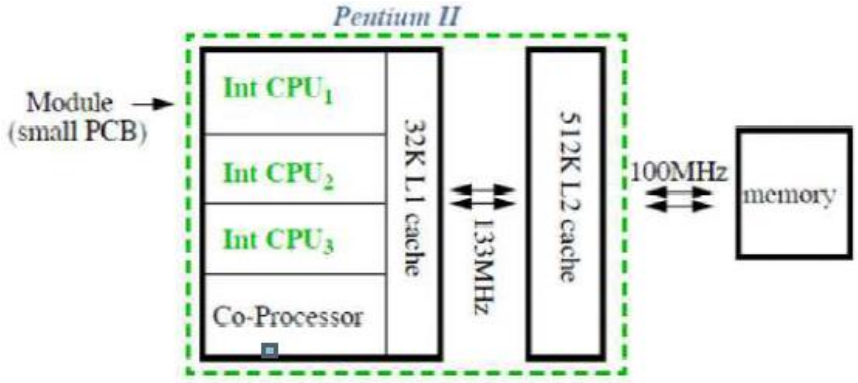



Fig. 7.3

- 32-bit Microprocessor, 64-bit data bus and 36-bit address bus
- 64GB Main Memory
- Starts at 266MHz
- 32 KB L1 cache; 512 KB L2 cache
- Three Integer processors

## THE PENTIUM-III MICROPROCESSOR

After the Pentium II, the next version of the Pentium processor is Pentium III. This processor was developed by Intel in 1999. Pentium III is a sixth-generation x86-compatible microprocessor and it was manufactured based on P6 micro-architecture. This processor core contains 9.5 to 28 million transistors. All features of the Pentium II have been added with Pentium III. This processor is used for high-performance desktop computers and servers which can operate Windows NT, Unix and Windows 98 operating systems. This processor is suitable for audio and video processing, image processing, and Internet and multimedia applications. The new features of the Pentium III processor are given below:

Pentium III can operate in multiple branch prediction algorithms. 70 new instructions are added to the Pentium III for multimedia and advanced image processing.

Pentium III has eight 64-bit Intel MMX registers and 57 MMX instructions for multimedia applications.

It has an on-die 512 Kbyte L2 cache. This processor is available at operating frequencies of 450 MHz to 1.4 GHz with 100 MHz to 133 MHz front side bus. Pentium III can operate at 2.0 V to 1.45 V.

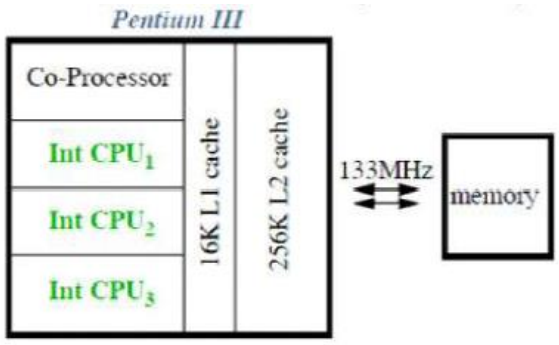



Fig. 7.4

- 32-bit Microprocessor, 64-bit data bus and 36-bit address bus
- 64GB Main Memory
- Starts at 800MHz
- 32 KB L1 cache; On-chip 256 KB L2 cache
- Three Integer processors

## THE PENTIUM-IV MICROPROCESSOR

Pentium 4 processor is the seventh generation processor. This processor was developed in 2000 by Intel. This processor core contains 42 million transistors. Pentium 4 operates at 1.40 V to 1.25 V. Clock speed of Pentium 4 varies from 1.3 GHz to 3.8 GHz.

It supports faster system bus at 400 MHz to 1066 MHz with 3.2 GB/s of bandwidth. Pentium 4 has 12 Kb L1 cache for code and 8 Kb for data. It has on-die 512 Kb L2 cache.

The instruction set of Pentium 4 processor is compatible with x86 (i386), x86-64, MMX, SSE, SSE2, and SSE3 instructions. These instructions include 128-bit SIMD integer arithmetic and 128-bit SIMD double-precision floating-point operations.

- 1.4 to 1.9 GHz; Latest 3.20 and 3.46 GHz
- 1MB/512KB/256KB L2 cache
- Specialized for streaming video, game and DVD application

## 7.3 SPECIAL PENTIUM REGISTERS

The Pentium is essentially the same microprocessor as the 80386 and 80486, except that some additional features and changes to the control register set have occurred. This section highlights the differences between the 80386 control register structure and the flag register.

### PENTIUM CONTROL REGISTER

Fig. shows the control register structure for the Pentium microprocessor. A new control register, CR4, has been added to the control register array.

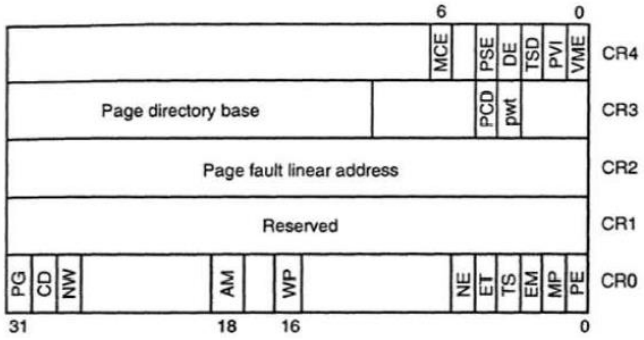



Fig. 7.5

Following is a description of the new control bits and the new control register CR4:

**CD –** Cache disable controls the internal cache. If CD = 1, the cache will not fill with new data for cache misses, but it will continue to function for cache hits. If CD = 0, misses will cause the cache to fill with new data.

**NW –** Not write-through selects the mode of operation for the data cache. If NW = 1, the data cache is inhibited from cache write-through.

**AM –** Alignment mask enables alignment checking when set. Note that alignment checking occurs only for protected mode operation when the user is at privilege level 3.

**WP –** Write protect protects user level pages against supervisor level write operations. When WP = I, the supervisor can write to user level segments.

**NE –** Numeric error enables standard numeric coprocessor error detection. If NE= I, the FERR pin becomes active for a numeric coprocessor error. If NE= 0, any coprocessor error is ignored.

**VME –** Virtual mode extension enables support for the virtual interrupt flag in protected mode. If VME = 0, virtual interrupt support is disabled.

**PVI –** Protected mode virtual interrupt enables support for the virtual interrupt flag in protected mode.

**TSD –** Time stamp disable controls the RDTSC instruction. Debugging extension enables 1/0 breakpoint debugging extensions when set.

**DE –** Debugging Extension is used to enable I/O break point.

**PSE –** Page size extension enables 4M-byte memory pages when set.

**MCE –** Machine check enable enables the machine checking interrupt.

### THE EXTENDED FLAG (EFLAG) REGISTER

E-Flag register has been changed in the Pentium microprocessor. Four new flag bits have been added to this register to control or indicate conditions about some of the new features in the Pentium.

**ID –** The identification flag is used to test for the CPUID instruction. If a program can set and clear the ID flag, the processor supports the CPUID instruction.

**VIP –** Virtual interrupt pending indicates that a virtual interrupt is pending.

**VIF –** Virtual interrupt is the image of the virtual interrupt flag IF used with VIP.

**AC –** Alignment check indicates the state of the AM bit in control register 0.

## 7.4 PENTIUM MEMORY MANAGEMENT

The memory-management unit within the Pentium is upward compatible with the 80386 and 80486 microprocessors. The main change is in the paging unit and a new system memory management mode.

## PAGING UNIT

The paging mechanism functions with 4K-byte memory pages or with a new extension available to the Pentium with 4M-byte memory pages.

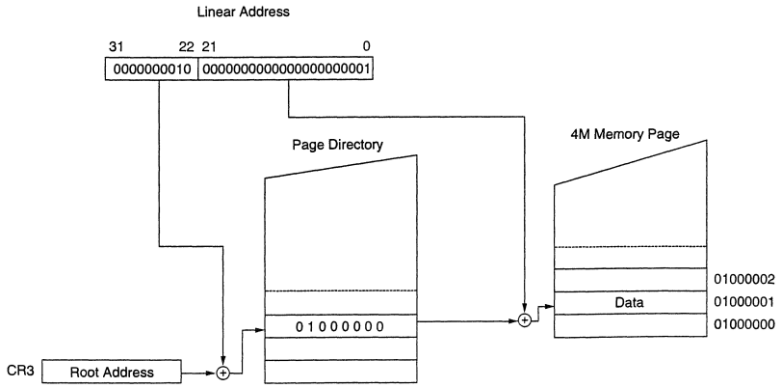



Fig. 7.6

The size of the paging table structure can become large in a system that contains a large memory. In the Pentium, with the new 4M-byte paging feature, this is dramatically reduced to just a single page table.

The main difference between 4K paging and 4M paging is that in the 4M paging scheme there is no page table entry in the linear address.

In Figure, the leftmost 10-bits of the linear address select an entry in the page directory (just as with 4K pages). Unlike 4K pages, there are no page tables; instead, the page directory ad- dresses a 4M-byte memory page.

## MEMORY-MANAGEMENT MODE

The system memory-management mode (SMM) is on the same level as protected mode, real mode, and virtual mode.

Access to the SMM is accomplished via a new external hardware interrupt applied to the SMI pin on the Pentium. When the SMM interrupt is activated, the processor begins executing system level software in an area of memory called the system management RAM or SMRAM.

The SMI interrupt disables all other interrupts normally handled by user applications and the operating system. A return from the SMM interrupt is accomplished with a new instruction. RSM re- turns from the memory-

management mode interrupt and returns to the interrupted program at the point of the interruption.

## 7.5 NEW PENTIUM INSTRUCTIONS

The Pentium contains only one new instruction that functions with normal system software; the remainder of the new instructions are added to control the memory-management mode feature and to serialize instructions. The operation of new instructions are as follows.

**CMPXCHG8B –** The CMPXCHG8B instruction compares the number 64-bit stored in EDX and EAX with the contents of a 64-bit memory location or register pair.

– The CPUID instruction reads the CPU identification code and other information from the Pentium.

**RDTSC –** The RDTSC instruction reads the time stamp counter into EDX: EAX. The time stamp counter counts CPU clocks from the time the microprocessor is reset, where the time stamp counter is cleared to zero.

**RDMSR and WRMSR –** The RDMSR and WRMSR instructions allow the model-specific registers to be read or written. The model-specific registers are unique to the Pentium and are used to trace, check performance, test, and check for machine errors.

**RSM –** The RSM instruction returns form a memory-management mode interrupt.

# EXERCISE

## *SECTION-I (MULTIPLE CHOICE QUESTIONS)*

1. The clock speed of 80386 microprocessor is …….
   (a) 12-33 MHz (b) 66 MHz (c) 80 MHz (d) 100 MHz
2. The cache memory of Pentium processor is
   (a) 2 KB (b) 4 KB (c) 8 KB (d) 16 KB
3. 80386 microprocessor can address …… memory.
   (a) 1MB (b) 2MB (c) 2GB (d) 4GB
4. The data bus of Pentium-II processor is …….. bits.
   (a) 16 (b) 32 (c) 64 (d) 128
5. 80486 has two versions. One is 486DX and other is
   (a) 486SX (b) 486PX (c) 486BX (d) 486NX
6. Memory management was first introduced in …….. microprocessor
   (a) 80186 (b) 80286 (c) 80386 (d) 80486
7. Pentium-II processor has main memory
   (a) 16 GB (b) 32 GB (c) 64 GB (d) 128 GB
8. 80186 microprocessor was initially designed for;
   (a) 6 MHz (b) 12 MHz (c) 32 MHz (d) 66 MHz

**ANSWER KEY**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | a | 2. | d | 3. | d | 4. | c |
| 5. | a | 6. | b | 7. | c | 8. | a |

## *SECTION-II (SHORT QUESTIONS)*

1. List salient features of 80186/80188 microprocessors.
2. Write main characteristics of 80386 Processor.
3. Write main characteristics of 80486 Processor.
4. Write main characteristics of Pentium IV Processor.
5. Write main characteristics of Pentium III Processor.
6. What is the function of VIP Flag in Pentium Processor?
7. What are special Pentium Registers?
8. Define paging unit.
9. What is a memory management unit?
10. Enlist the new Pentium Instructions
11. Write the function of instruction CMPXCHG8B.

12. Write the function of instruction “RSM”.

## ***SECTION-III (LONG QUESTIONS)***

1. Discuss Pentium Memory Management.
2. Describe the functions of four new flags of Pentium processor.
3. Enlist new Pentium instructions and describe function of each instruction.
4. Discuss summary of growth from 80186 to 80486.

# CHAPTER 08
# THE MICROCONTROLLER

## 8.1 SINGLE CHIP MICROPROCESSOR

A single chip that contains the processor (CPU), non-volatile memory (flash memory or ROM) for the program, volatile memory (RAM) for processing the data, a clock and an I/O control unit is called Microcontroller. Microcontroller units (MCUs) are available in numerous sizes and architectures. The MCU was actually the first "system-on-chip" (SoC) but that term was coined later when more components were added.

Putting a CPU, ROM, RAM, and data input/output circuitry into a single IC will make a single-chip microprocessor. Single-chip microprocessors come compact and cheap, but do not allow users to choose built-in functions at their option. Single-chip microprocessors are also known as "Microcontroller Units (MCUs)," because they are made of a single IC.

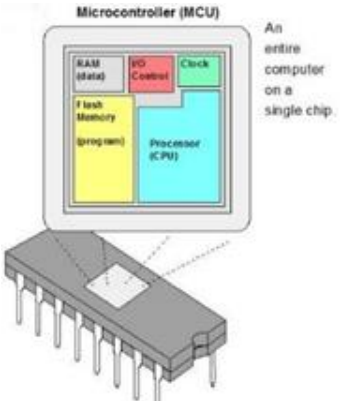


Fig. 8.1

**Types of Microcontrollers**

Microcontrollers are divided into various categories based on memory, architecture, bits and instruction sets. Following is the list of their types –

**1. Based on Bit Configuration**

- 8-bit microcontroller − This type of microcontroller is used to execute arithmetic and logical operations like addition, subtraction, multiplication division, etc. For example, Intel 8031 and 8051 are 8 bits microcontroller.
- 16-bit microcontroller − This type of microcontroller is used to perform arithmetic and logical operations where higher accuracy and performance is required. For example, Intel 8096 is a 16-bit microcontroller.

- 32-bit microcontroller − This type of microcontroller is generally used in automatically controlled appliances like automatic operational machines, medical appliances, etc.

**2. Based on the Memory Configuration**

- External memory microcontroller − This type of microcontroller is designed in such a way that they do not have a program memory on the chip. Hence, it is named as external memory microcontroller. For example: Intel 8031 microcontroller.
- Embedded memory microcontroller − This type of microcontroller is designed in such a way that the microcontroller has all programs and data memory, counters and timers, interrupts, I/O ports are embedded on the chip. For example: Intel 8051 microcontroller.

**3. Based on the Instruction Set Configuration**

- CISC − CISC stands for complex instruction set computer. It allows the user to insert a single instruction as an alternative to many simple instructions.
- RISC − RISC stands for Reduced Instruction Set Computers. It reduces the operational time by shortening the clock cycle per instruction.

## COMPARISON OF MICROCONTROLLER & MICROPROCESSOR

A microcontroller differs from a microprocessor in many ways. The first and most important difference is its functionality. In order the microprocessor may be used, other components such as memory or components for data transfer must be added to it. Even though the microprocessor is considered to be a powerful computer machine, the weak point is that it is not adjusted to communication to peripheral environment. Simply, in order to communicate with peripheral environment, the microprocessor must use specialized circuits added as external chips. It means in short that microprocessors are the pure heart of the computers. The comparison of microprocessor and microcontroller is presented as follows:

| *MICROPROCESSORS* | *MICROCONTROLLERS* |
|---|---|
| Microprocessor performs the function of a central processing unit (CPU) on to a single integrated | Microcontroller can be considered as a small computer which has a processor and some other |

| | |
|---|---|
| circuit (IC). | components. |
| Microprocessors are mainly used in designing general purpose systems. | Microcontrollers are used in automatically controlled devices. |
| Microprocessors are basic components of personal computers. | Microcontrollers are generally used in embedded systems. |
| Computational capacity of microprocessor is very high. Hence can perform complex tasks. | Less computational capacity when compared to microprocessors. Usually used for simpler tasks. |
| A microprocessor-based system can perform numerous tasks. | A microcontroller-based system can perform single or very few tasks. |
| Microprocessors have integrated Math Coprocessor. Complex mathematical calculations which involve floating point can be performed with great ease. | Microcontrollers do not have math coprocessors. They use software to perform floating point calculations which slows down the device. |
| The main task of microprocessor is to perform the instruction cycle repeatedly. This includes fetch, decode and execute. | In addition to performing the tasks of fetch, decode and execute, a microcontroller also controls its environment based on the output of the instruction cycle. |
| In order to build or design a system (computer), a microprocessor has to be connected externally to some other components like Memory (RAM and ROM) and Input / Output ports. | The IC of a microcontroller has memory (both RAM and ROM) integrated on it along with some other components like I / O devices and timers. |
| The overall cost of a system built using a microprocessor is high. This is because of the requirement of external components. | Cost of a system built using a microcontroller is less as all the components are readily available. |
| Power consumption is high. | Power consumption is less. |
| The clock frequency is very high usually in the order of Giga Hertz. | Clock frequency is less usually in the order of Mega Hertz. |
| Instruction throughput is given higher priority than interrupt | In contrast, microcontrollers are designed to optimize interrupt |

| | |
|---|---|
| latency. | latency. |
| Have few bit manipulation instructions | Bit manipulation is powerful and widely used feature in microcontrollers. |

## APPLICATIONS OF 8051 MICROCONTROLLER

Some of the applications of 8051 mainly used in daily life & industrial applications are as follows:

- Light sensing and controlling devices
- Temperature sensing and controlling devices
- Fire detections and safety devices
- Automobile applications
- Defense applications

**Some industrial applications of micro controller and its applications**

- Industrial instrumentation devices
- Process control devices

**Measurement applications**

- Voltmeter applications
- Measuring and revolving objects
- Current meter objects
- Hand held metering system

**Applications in Embedded Systems**

The applications of 8051 microcontroller involves in 8051 based projects. The list of 8051 projects is listed below.

- Arduino Managed High Sensitive LDR based Power Saver for Street Light Control System
- The Temperature Humidity Monitoring System of Soil Based on Wireless Sensor Networks using Arduino
- RFID based Electronic Passport System for Easy Governance using Arduino
- Arduino based DC Motor Speed Control
- Arduino Based Line Following Robot
- GSM based Electricity Energy Meter Billing with Onsite Display
- Android Phone Speech Recognition Sensed Voice Command based Notice Board Display
- Parking Availability Indication System
- Voice Controlled Home Appliances

- Remote Control Home Appliances
- Solar Highway Lighting System with Auto Turn Off in Daytime
- 8051 Microcontroller based Wireless Energy Meter

## 8.2 INTRODUCTION TO MICROCONTROLLERS

A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip as shown in fig. 8.2.

A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor. The temporary information that the microcontroller receives is stored in its data memory, where the processor accesses it and uses instructions stored in its program memory to decode and apply the incoming data. It then uses its I/O peripherals to communicate and enact the appropriate action.

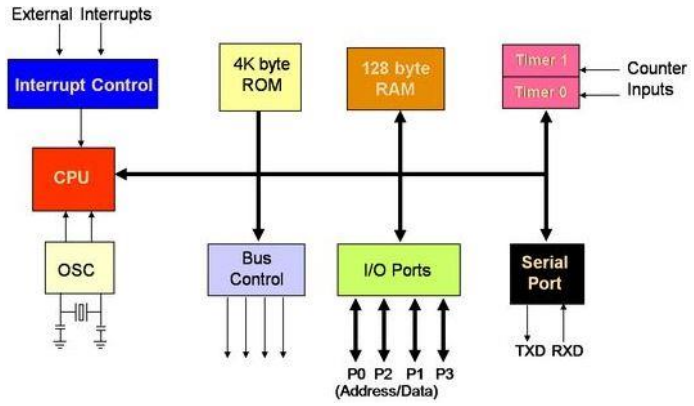



Fig. 8.2

The core elements of a microcontroller are:

**The Processor (CPU) --** A processor responds to various instructions that direct the microcontroller's function. This involves performing basic arithmetic, logic, I/O and data transfer operations.

**Memory --** A microcontroller's memory is used to store the data that the processor receives and uses to respond to instructions. A microcontroller has two main memory types:

Program memory, which stores long-term information about the instructions that the CPU carries out. Program memory is non-volatile memory.

Data memory, which is required for temporary data storage while the instructions are being executed. Data memory is volatile.

**I/O peripherals --** The input and output devices are the interface for the processor to the outside world. The input ports receive information and send it to the processor. The processor receives that data and sends the necessary instructions to output devices that execute tasks external to the microcontroller.

The processor, memory and I/O peripherals are the defining elements of the microprocessor, there are other elements that are frequently included. Other supporting elements of a microcontroller include:

**Analog to Digital Converter (ADC) --** An ADC is a circuit that converts analog signals to digital signals. It allows the processor at the center of the microcontroller to interface with external analog devices, such as sensors.

**Digital to Analog Converter (DAC) --** A DAC performs the inverse function of an ADC and allows the processor at the center of the microcontroller to communicate its outgoing signals to external analog components.

**System bus --** The system bus is the connective wire that links all components of the microcontroller together.

**Serial port --** The serial port is one example of an I/O port that allows the microcontroller to connect to external components. It has a similar function to a USB or a parallel port but differs in the way it exchanges bits.

## 8.3 8051 INTERNAL RAM & REGISTERS

The Data Memory or RAM of the 8051 Microcontroller stores temporary data and intermediate results that are generated and used during the normal operation of the microcontroller. Original Intel's 8051 Microcontroller had 128 bytes of internal RAM. But almost all modern variants of 8051 Microcontroller have 256 bytes of RAM. The first 128 bytes i.e., memory addresses from 00H to 7FH is divided into three different areas as shown in fig. 8.3.

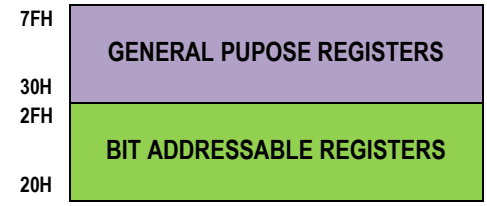



Fig. 8.3

**WORKING REGISTERS**

The first 32 bytes from addresses 00H to 1FH are organized as four banks as shown in the fig. 8.4. Each bank has eight registers R0 to R7. Each Register can be addressed in two ways: either by name or by address.

Only one bank can be accessed at a time. To address the register by name, first the corresponding Bank must be selected. In order to select the bank, we have to use the RS0 and RS1 bits of the Program Status Word (PSW) Register. RS0 and RS1 are 3rd and 4th bits in the PSW Register.

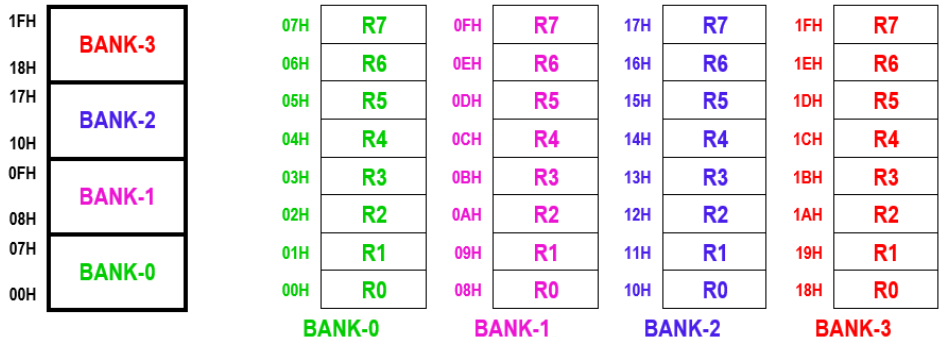



Fig. 8.4

The RS1 and RS0 is used like this to select the register banks –

| RS1 | RS0 | Register Bank |
|---|---|---|
| 0 | 0 | Register Bank 0 |
| 0 | 1 | Register Bank 1 |
| 1 | 0 | Register Bank 2 |
| 1 | 1 | Register Bank 3 |

When RESET pin is activated, RS0 and RS1 will be cleared (i.e., RS0=0 and RS1=0). So Bank0 is selected.

**BIT ADDRESSABLE REGISTERS**

Bit addressable area can store or remove one bit of data as well as one byte data. The 8051 has the next 16 bytes of the RAM i.e., from 20H to 2FH are bit addressable memory locations. There are totally 128 bits that can be

addressed individually using 00H to 7FH or the entire byte can be addressed as 20H to 2FH.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2FH | 7FH | 7EH | 7DH | 7CH | 7BH | 7AH | 79H | 78H |
| 2EH | 77H | ..... | ..... | ..... | ..... | ..... | ..... | 70H |
| 2DH | 6FH | ..... | ..... | ..... | ..... | ..... | ..... | 68H |
| 2CH | 67H | ..... | ..... | ..... | ..... | ..... | ..... | 60H |
| 2BH | 5FH | ..... | ..... | ..... | ..... | ..... | ..... | 58H |
| 2AH | 57H | ..... | ..... | ..... | ..... | ..... | ..... | 50H |
| 29H | 4FH | ..... | ..... | ..... | ..... | ..... | ..... | 48H |
| 28H | 47H | ..... | ..... | ..... | ..... | ..... | ..... | 40H |
| 27H | 3FH | ..... | ..... | ..... | ..... | ..... | ..... | 38H |
| 26H | 37H | ..... | ..... | ..... | ..... | ..... | ..... | 30H |
| 25H | 2FH | ..... | ..... | ..... | ..... | ..... | ..... | 28H |
| 24H | 27H | ..... | ..... | ..... | ..... | ..... | ..... | 20H |
| 23H | 1FH | ..... | ..... | ..... | ..... | ..... | ..... | 18H |
| 22H | 17H | ..... | ..... | ..... | ..... | ..... | ..... | 10H |
| 21H | 0FH | ..... | ..... | ..... | ..... | ..... | ..... | 08H |
| 20H | 07H | 06H | 05H | 04H | 03H | 02H | 01H | 00H |

Fig. 8.5

A bit addressable area of 16 bytes from byte addresses 20H to 2FH in internal RAM as shown in fig. 8.5, forming a total of 128 addressable bits (i.e., 16-byte location × 8 bits). Each bit can be accessed by its bit addresses from 00H to 7FH for the byte address location 20H to 2FH in RAM. For example 32H is the bit 2 of the internal RAM location 26H.

**GENERAL PURPOSE REGISTERS**

The final 80 bytes of the internal RAM area from 30H to 7FH location are used for general purpose data storage. It is addressable by bytes. The advantage of using this memory location is that their data access is faster compared to off-chip RAM.

**Note:** These lower 128 bytes of RAM can be addressed directly or indirectly.

## SPECIAL FUNCTION REGISTERS

The upper 128 bytes of the RAM i.e., memory addresses from 80H to FFH is allocated for Special Function Registers (SFRs). SFRs control specific functions of the 8051 Microcontroller. Some of the SFRs are I/O Port Registers (P0, P1, P2 and P3), PSW (Program Status Word), A (Accumulator), IE (Interrupt Enable), PCON (Power Control), etc.

21 addresses in the SFR area can be used in this microcontroller. Out of these 21 locations, 11are bit-addressable SFR locations. The remaining 10 bits are not bit-addressable.

The bit addressable SFRs are like below -

| Register | Byte address | Bit-address |
|---|---|---|
| P0 (Port 0) | 80H | 80H to 87H |
| P1 (Port 1) | 90H | 90H to 97H |
| P2 (Port 2) | A0H | A0H to A7H |
| P3 (Port 3) | B0H | B0H to B7H |
| PSW | D0H | D0H to D7H |
| Register A (Accumulator) | E0H | E0H to E7H |
| Register B | F0H | F0H to F7H |
| TCON (Timer Control) | 88H | 88H to 8FH |
| SCON (Serial Control) | 98H | 98H to 9FH |
| IE (Interrupt Enable) | A8H | A8H to AFH |
| IP (Interrupt Priority) | B8H | B8H to BFH |

The remaining 10 not bit-addressable are as below -

| Register | Byte address |
|---|---|
| SP (Stack Pointer) | 81H |
| DPL (Data Pointer Low) | 82H |
| DPH (Data Pointer High) | 83H |
| PCON (Power Control) | 87H |
| TMOD (Timer Mode) | 89H |
| TL0 (Timer0 Low) | 8AH |
| TL1 (Timer1 Low) | 8BH |
| TH0 (Timer0 High) | 8CH |
| TH1 (Timer1 Low) | 8DH |
| SBUF (Serial Buffer) | 99H |

SRFs Memory addresses are only direct addressable. Even though some of the addresses between 80H and FFH are not assigned to any SFR, they cannot be used as additional RAM area.

## 8.4 8051 INTERRUPT SYSTEM

It is a subroutine calls that given by the microcontroller when some other program with high priority is request for acquiring the system buses than interrupt occur in current running program.

Interrupts provide a method to postpone or delay the current process, performs a sub-routine task and then restart the standard program again.

**Types of interrupt in 8051 Microcontroller**

Let's see the five sources of interrupts in 8051 Microcontroller:

- Timer 0 overflow interrupt - TF0
- External hardware interrupt - INT0
- Timer 1 overflow interrupt - TF1
- External hardware interrupt - INT1
- Serial communication interrupt - RI/TI

The timer and serial interrupts are internally produced by the microcontroller, whereas the external interrupts are produced by additional interfacing devices or switches that are externally connected with the microcontroller. These external interrupts can be level triggered or edge triggered. When interrupt occur then the microcontroller executes the interrupt service routine. Therefore the memory location corresponds to interrupt enables it. Consider the interrupt corresponding to the memory location is shown in the interrupt vector table below.

| Interrupt Number | Interrupt Description | Address |
|---|---|---|
| 0 | EXTERNAL INT 0 | 0003h |
| 1 | TIMER/COUNTER 0 | 000Bh |
| 2 | EXTERNAL INT 1 | 0013h |
| 3 | TIMER/COUNTER 1 | 001Bh |
| 4 | SERIAL PORT | 0023h |

## 8.5 8051 INSTRUCTION SET

Writing a Program for any Microcontroller consists of giving commands to the Microcontroller in a particular order in which they must be executed in order to perform a specific task. The commands to the Microcontroller are known as a Microcontroller's Instruction Set.

An Instruction Set is unique to a family of computers. 8051 Microcontroller Instruction Set is also called as the MCS-51 Instruction Set.

As the 8051 family of Microcontrollers are 8-bit processors, the 8051 Microcontroller Instruction Set is optimized for 8-bit control applications. As a typical 8-bit processor, the 8051 Microcontroller instructions have 8-bit Opcodes. The 8051 Microcontroller instruction set can have up to 28 = 256 Instructions.

## INSTRUCTION SET (THE MCS-51 INSTRUCTION SET)

There are 49 Instruction Mnemonics in the 8051 Microcontroller Instruction Set and these 49 Mnemonics are divided into five groups.

Based on the operation they perform, all the instructions in the 8051 Microcontroller Instruction Set are divided into five groups. They are:

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Boolean or Bit Manipulation Instructions
- Program Branching Instructions

We will now see about these instructions briefly.

## DATA TRANSFER INSTRUCTIONS

The Data Transfer Instructions are associated with transfer of data between registers or external program memory or external data memory. The Mnemonics associated with Data Transfer are given below.

| Mnemonic | Description |
|---|---|
| MOV | Move Data |
| MOVC | Move Code |
| MOCX | Move External Data |
| PUSH | Move Data to Stack |
| POP | Copy Data from Stack |
| XCH | Exchange Data between two Registers |
| XCHD | Exchange Lower Order Data between two Registers |

## ARITHMETIC INSTRUCTIONS

Using Arithmetic Instructions, you can perform addition, subtraction, multiplication and division. The arithmetic instructions also include increment by one, decrement by one and a special instruction called Decimal

Adjust Accumulator. The Mnemonics associated with the Arithmetic Instructions of the 8051 Microcontroller Instruction Set are:

| Mnemonic | Description |
|---|---|
| ADD | Addition without Carry |
| ADDC | Addition with Carry |
| SUBB | Subtract with Carry |
| INC | Increment by 1 |
| DEC | Decrement by 1 |
| MUL | Multiply |
| DIV | Divide |
| DA A | Decimal Adjust the Accumulator (A Register) |

The arithmetic instructions have no knowledge about the data format i.e., signed, unsigned, ASCII, BCD, etc. Also, the operations performed by the arithmetic instructions affect flags like carry, overflow, zero, etc. in the PSW Register.

## LOGICAL INSTRUCTIONS

The next group of instructions are the Logical Instructions, which perform logical operations like AND, OR, XOR, NOT, Rotate, Clear and Swap. Logical Instruction are performed on Bytes of data on a bit-by-bit basis.
Mnemonics associated with Logical Instructions are as follows:

| Mnemonic | Description |
|---|---|
| ANL | Logical AND |
| ORL | Logical OR |
| XRL | Ex-OR |
| CLR | Clear Register |
| CPL | Complement the Register |
| RL | Rotate a Byte to Left |
| RLC | Rotate a Byte and Carry Bit to Left |
| RR | Rotate a Byte to Right |
| RRC | Rotate a Byte and Carry Bit to Right |
| SWAP | Exchange lower and higher nibbles in a Byte |

## BOOLEAN OR BIT MANIPULATION INSTRUCTIONS

As the name suggests, Boolean or Bit Manipulation Instructions deal with bit variables. We know that there is a special bit-addressable area in the RAM and some of the Special Function Registers (SFRs) are also bit addressable. The Mnemonics corresponding to the Boolean or Bit Manipulation instructions are:

| Mnemonic | Description |
|---|---|
| CLR | Clear a Bit (Reset to 0) |
| SETB | Set a Bit (Set to 1) |
| MOV | Move a Bit |
| JC | Jump if Carry Flag is Set |
| JNC | Jump if Carry Flag is Not Set |
| JB | Jump if specified Bit is Set |
| JNB | Jump if specified Bit is Not Set |
| JBC | Jump if specified Bit is Set and also clear the Bit |
| ANL | Bitwise AND |
| ORL | Bitwise OR |
| CPL | Complement the Bit |

These instructions can perform set, clear, and, or, complement etc. at bit level.

## PROGRAM BRANCHING INSTRUCTIONS

The last group of instructions in the 8051 Microcontroller Instruction Set is the Program Branching Instructions. These instructions control the flow of program logic. The mnemonics of the Program Branching Instructions are as follows.

| Mnemonic | Description |
|---|---|
| LJMP | Long Jump (Unconditional) |
| AJMP | Absolute Jump (Unconditional) |
| SJMP | Short Jump (Unconditional) |
| JZ | Jump if A is equal to 0 |
| JNZ | Jump if A is not equal to 0 |
| CJNE | Compare and Jump if Not Equal |
| DJNZ | Decrement and Jump if Not Zero |
| NOP | No Operation |

| | |
|---|---|
| LCALL | Long Call to Subroutine |
| ACALL | Absolute Call to Subroutine (Unconditional) |
| RET | Return from Subroutine |
| RETI | Return from Interrupt |
| JMP | Jump to an Address (Unconditional) |

All these instructions, except the NOP (No Operation) affect the Program Counter (PC) in one way or other. Some of these instructions has decision making capability before transferring control to other part of the program.

## 8.6 MICROCONTROLLERS ON THE 8051 FAMILY

### OVERVIEW OF THE 8051 FAMILY:

8051 microcontroller was initially designed by Intel Corporation in 1981. Features of 8051 made it extremely popular in market. Because of it's popularity and high demand Intel allowed other manufacturers to fabricate and market different variants of 8051 with a condition that all these variants should be code compatible with 8051. This resulted in a lot of variants of 8051 in market, among which 8052 and 8031 are the most popular ones. Therefore, 8052 and 8031 are considered as the family members of 8051.

### 8052 MICROCONTROLLER

8052 is the super set of 8051 as it has all the features of 8051 with an extra timer and an extra RAM of 128 bytes. Therefore, 8052 has a total of 256 bytes of RAM and 3 timers in all. Also all the programs written for 8051 will run on 8052 as 8052 is super set of 8051, but it's reverse is not true.

### 8031 MICROCONTROLLER

8031 is referred to as ROM-less microcontroller chip because it has 0 K byes of on-chip ROM. For it's operation, 8031 requires external ROM which aids it in fetch and execute operations. Apart from this, it shares almost all the features of 8051.

### COMPARISON OF 8051 WITH ITS OTHER FAMILY MEMBERS:

Following table highlights the main characteristics of distinction between 8051, 8052 and 8031.

| Characteristic | 8051 | 8052 | 8031 |
|---|---|---|---|
| RAM | 128 bytes | 256 bytes | 128 bytes |
| ROM (on-chip) | 4 KB | 8 KB | 0 KB |

| | | | |
|---|---|---|---|
| Number of Timer | 2 | 3 | 2 |
| Interrupt Sources | 6 | 8 | 6 |
| Serial Port | 1 | 1 | 1 |
| Number of I/O Ports | 32 | 32 | 32 |

## EXERCISE

### *SECTION-I (MULTIPLE CHOICE QUESTIONS)*

**1. The microcontrollers are used in;**

a. Computers, laptops, televisions
b. Printers, Refrigerators
c. Microwave Ovens
d. All of the above

**2. The speed of the microcontrollers is;**

a. High
b. Slow
c. Very High
d. Better

**3. The 8-bit microcontrollers are used to execute;**

**a.** Arithmetic operations only
b. Logical operations only
c. Both a and b
**d.** None of the above

**4. _____ is an example of 16 bit microcontroller**

a. 8031 microcontroller
b. 8051 microcontroller
c. 8096 microcontroller
d. None of the above

**5. The type of microcontroller is designed in such a way that they don't have a program memory on the chip is called as _____**

a. External memory microcontroller
b. Embedded memory microcontroller
c. Both a and b
d. None of the above

**6. _____ is an example of external memory microcontroller**

a. Intel 8031 microcontroller
b. 8051 microcontroller
c. 8096 microcontroller
d. None of the above

**7. In embedded memory microcontroller has _____**

a. Programs, data
b. Counters, timers
c. Input/output ports
d. All of the above

**8. _____ is the first microcontroller designed by intel**

a. Intel 8031 microcontroller
b. 8051 microcontroller
c. 8096 microcontroller
d. None of the above

**9. The 8051 microcontroller is a _____ bit microcontroller**

a. 8 bit microcontroller
b. 16 bit microcontroller
c. 32 bit microcontroller
d. None of above

**10. The 8051 microcontroller has_____**

a. 8 bit data Bus
b. 16 bit data Bus
c. 32 bit Data Bus
d. None of the above

## ANSWER KEY

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. | **d** | 2. | **b** | 3. | **c** | 4. | **c** | 5. | **a** |
| 6. | **a** | 7. | **d** | 8. | **b** | 9. | **a** | 10. | **a** |

## ***SECTION-II (SHORT QUESTIONS)***

1. Define microcontroller.
2. Define microprocessor.
3. Enlist two differences between a microprocessor & microcontroller.
4. Give three applications of microcontrollers?
5. How much memory space is built in 8051 microcontroller?
6. Name the interrupts of 8051 microcontroller.
7. Describe the sizes of 8051 microcontroller?
8. How much timers /counters are available with 8051 microcontroller?

9. Name three groups of instruction set of 8051 microcontroller?
10. Enlist various addressing modes of 8051 microcontroller.

## *SECTION-III (LONG QUESTIONS)*

1. Describe the detailed comparison of microprocessor and microcontrollers.
2. Describe the architecture of 8051 microcontroller.
3. Explain the various addressing modes of 8051 microcontroller.

## TEXT /REFERENCE BOOKS /Websites :

1. BarryB.Brey ··The Intel Microprocessors (8086/8088, 80186. 80286. 80386. 80486)".

2. Microprocessor Principles l & Applications By Gilmore

3. Microprocessors - Douglas V. Hall

4. www.allaboutcircuits.com