EE 390 : Digital System Engineering
Hand out 1 by Dr Sheikh Sharif Iqbal

Chapter 1 (as per text book)

Reference to text book: The 8088 and 8086 Microprocessors…. by Triebel and Singh

Types of Computers:

(1) Mainframe: - Invented by scientists of Bell Lab in 1950's (vacuum tubes). 800-transistors were used to built the 1st transistorized computer.
- Used to serve large number of users. Such as University data processing center, etc.……..)

(2) Minicomputer: - In 1960's, the advent of integrated circuits, a scalled-down version of mainframe computer, called minicomputer, become popular due to its low cost.
- Used to serve small and multi-user business environment.
- Consist of one computer-console and several user-terminals.

(3) Microcomputer: - In 1970's, microprocessors were used to built single-user PC's.
- The 1st generation of microprocessors were called 4004. It had 2250 transistors, 4-bit internal registers, 4-bit external data bus and operated at 0.108 MHz. Used in Calculators.
- With the use of Local Area Network (LAN) and File-server (*extends computational power and system resources to clients*), PC's are currently preferred over minicomputers.

(4) Micro-controllers: - The non-reprogrammable or dedicated use microprocessors are called embedded microprocessors or microcontrollers.
- Data controller (hard disk) and event controller (elevator)

Reprogrammable Microprocessors: **8086/8088** → 80286 → 80386 →→ Pentium

8080

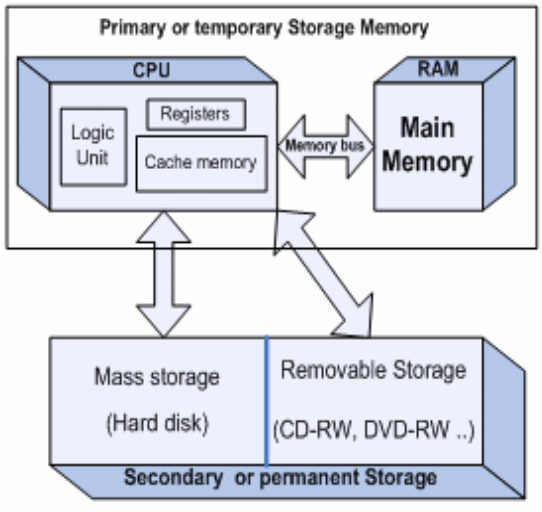
Embedded microprocessors or Microcontrollers: 8048/**8051** → 80186 →→ 80386EX

(5) Super-computers: In 1976, Cray-1, the 1st super-computer was built using ECL circuits. The processing speed was 130 MFLOPS.

(6) RISC Processors: -In early 1980, reduces instruction set computers were designed
- required less processor registers, clock cycles, addressing modes compared to CISC (complex instruction set) computers

Microcomputer: Operate in real-address mode (chip functions like 8086) or Protected address mode (multi program environment)

General Architecture or Microcomputer:



Microcomputer is an OPEN system.

- Microprocessor Unit: VLSI
- Input Unit
- Output unit
- Memory Unit
  - Primary ➔ Temporary
    - ROM ➔ EPROM, FLASH
    - RAM ➔ SRAM, DRAM
  - Secondary ➔ Permanent
    - Floppy ➔
    - Hard Disk ➔
    - CD ➔
- DVD ➔

8086 Microprocessor: - Invented in 1978 by Intel Corp.
- 16-bit internal registers
- 16-bit Data Bus
- 20-bit address bus
- support 1MB memory (8-bit)
- processing speed 4.77 MHz
- made of 29000 transistors

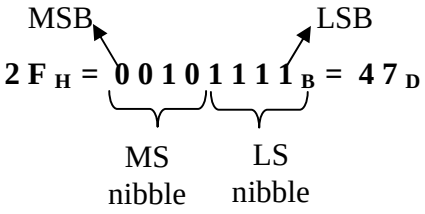
Figure 1-10: See page 14 of text book (4-th Edition)

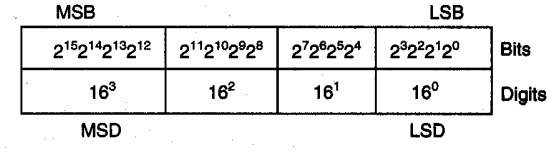
8088 Microprocessor: - Invented in 1979 by Intel Corp.
- 16-bit internal registers
- 8-bit Data Bus

Remaining specifications are same with 8086

Although 8088 processors are less efficient **but** also required less expensive memory module (one 8-bit bank) compared to 8086 (two 8-bit memory bank to store 16-bit data).

Review of Number systems:





Decimal number	Binary number	Hexadecimal number
0	00000000	00
1	00000001	01
2	00000010	02
3	00000011	03
4	00000100	04
5	00000101	05
6	00000110	06
7	00000111	07
8	00001000	08
9	00001001	09
10	00001010	0A
11	00001011	0B
12	00001100	0C
13	00001101	0D
14	00001110	0E
15	00001111	0F

EE 390 : Digital System Engineering

Hand out 2 by Dr Sheikh Sharif Iqbal

Chapter 2: **Data Organization:**

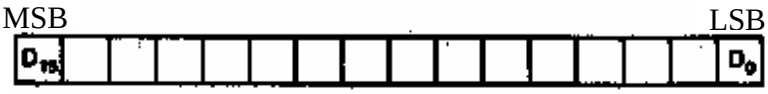
(7) Bit: One Binary digit. Example, '0' or '1'. Three Bit data ➔ '010_B' or '111_B'

(8) Nibble: Consist of Four Bits of binary data. Example, '**0**10**1**$_B$' or '**1**10**0**$_B$', where the leftmost bit is called Most-Significant-Bit (MSB) and the right most bit is called the Least-Significant-Bit (LSB).

(9) Byte: Consist of Eight Bits of binary data. Example, '0101**1101**$_B$' or '**1111**0000$_B$', where the leftmost nibble is called the Most-Significant-Nibble and the right most nibble is called the Least-Significant-Nibble.

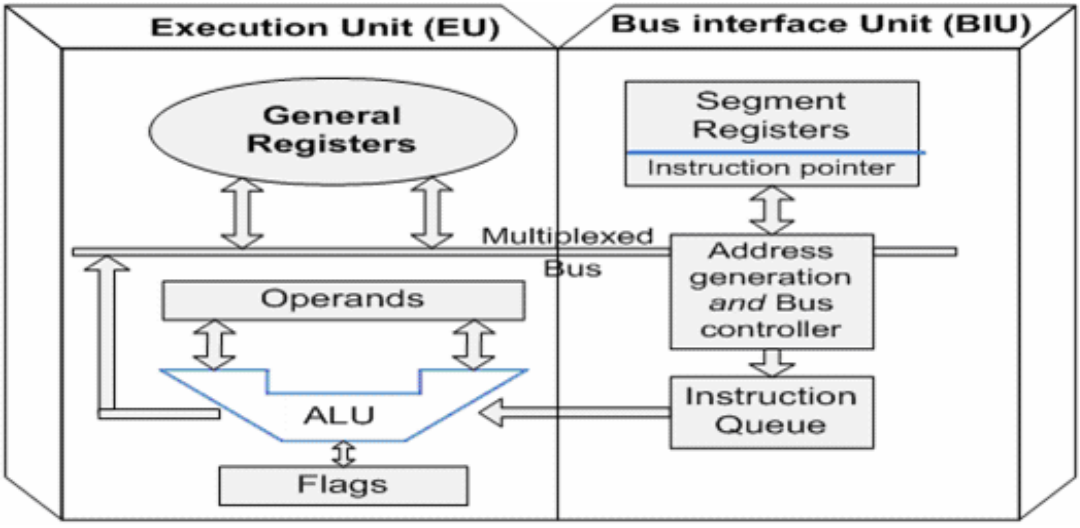
- Remember one nibble of binary data corresponds to one Hexadecimal digit. Example, '$1111_B = F_H$' or '$0101_B = 5_H$'

(10) Word: Consist of Sixteen Bits of binary data OR two consecutive bytes OR four hexadecimal digit (4 nibbles). Example, '**1111** 0000 0101 **1101**$_B$', where the leftmost byte is called the Most-Significant-Byte and the right most byte is called the Least-Significant-Byte.



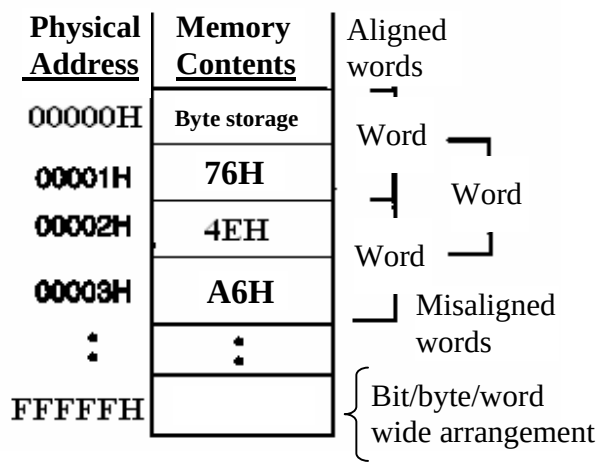
(11) Double-Word: Consist of 32 Bits of binary data OR two consecutive Words

Internal Architecture of 8088/8086 Microprocessor: Employ simultaneous operation



EU: Decode, Execute data/inst; flag control BIU: Instruction and data fetch and queue

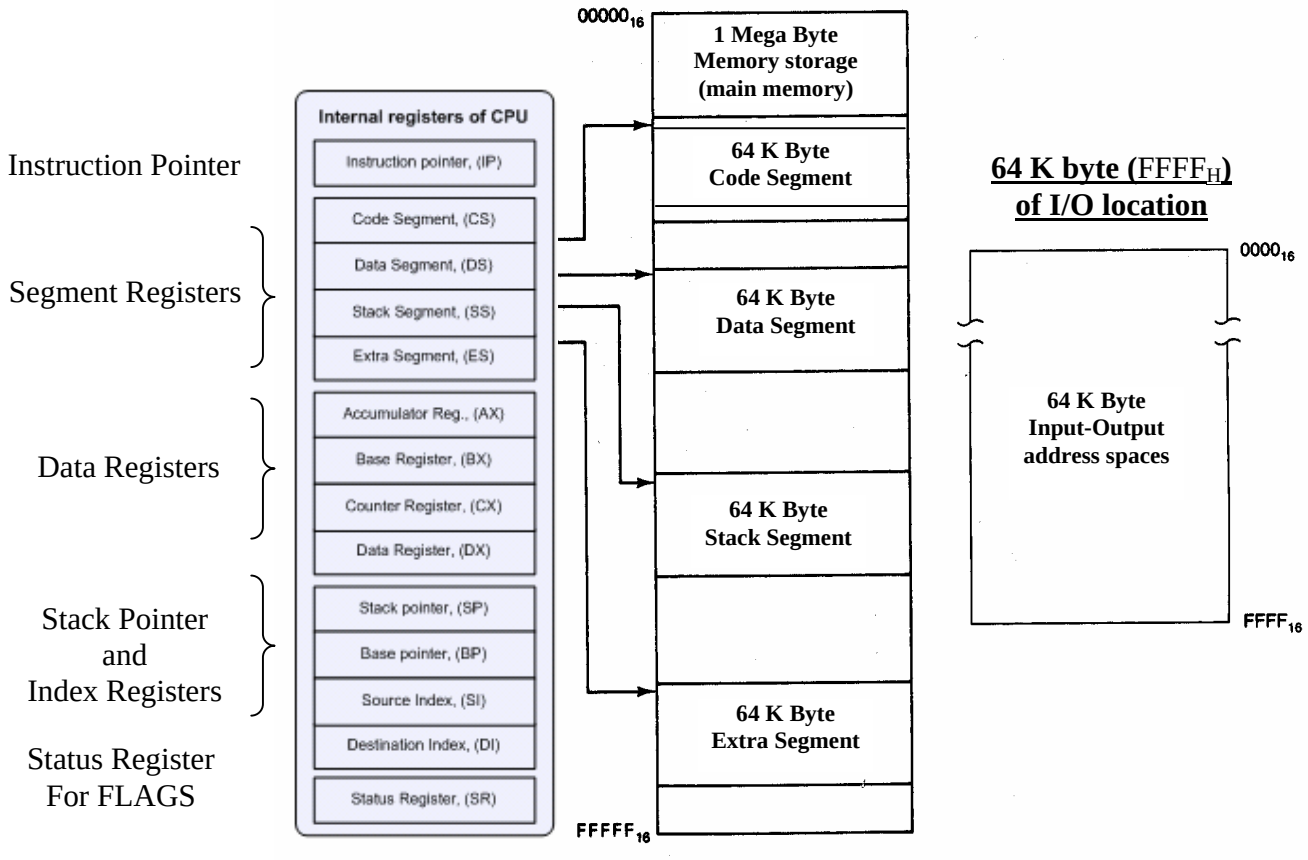
Data Organization in Main Memory: 8088/8086 Supports 1 MByte of Main Memory.



- Each Byte/Word of stored data is pointed by a 20-bit long physical address (PA).
- The 1 MByte memory location is divided into four 64 Kbyte Segments as 8086/8088 can access only four segments at any time.

  (a) Code Segment (*stores program codes*),
  (b) Data Segment (*stores Operands/Data*)
  (c) Extra Segment (*stores additional Data*)
  (d) Stack Segment (*stores temporary data*)
- Each Segment has a base-address which is pointed by the MPU-segment-register.
- Offset-address is pointed by pointers/index.

Software Model of 8088/8086: Operation of microprocessor from software point of view



16 Bit Internal Registers

One Million (mega) 8-Bit Storage
$FFFFF_H$=1MByte

16 bit Segment Register

Offset Register

20 bit Physical Address ➔

$(PA)_{\text{in Code-Segment memory}}$ = **CS***10 + **IP**; $(PA)_{\text{in Data-Segment}}$ = **DS***10 + **SI** = DS*10 + **DI**

$(PA)_{\text{in Extra-Segment memory}}$ = **ES***10 + **SI** = ES*10 + **DI**; $(PA)_{\text{in Stack-Segment}}$ = **SS***10 + **SP**;

20 Bit Physical Address pointing to Memory = Related Segment Register value * **10** + Offset Register value

EE 390 : Digital System Engineering
Hand out 3 by Dr Sheikh Sharif Iqbal

Chapter 2 (cont'd): **2.11. Generating a memory address:** (review)

20-bit **Physical address =** 10*16-bit segment **base address** + 16-bit **offset address**.

Note that the **lowest nibble** (or lowest hex digit) of the **base address** (*lowest--physical address of a segment*) should be "$\mathbf{0}_H$" ➔ $(PA)_{Seg\ Base} = 1234\mathbf{0}_H$

2.5 and 2.7 and 2.8 and 2.9: 16-bit Internal Registers:

(a) Code Segment Register (CS) ➔ stores "sixteen-bits starting from the MSB" of the base address of Code Segment (Or the four left-most hex digits (as the Least significant hex digit is "0_H")

* REMEMBER, VALUES OF 'CS' AND 'IP' COMBINES TO FORM THE (P.A.) POINTING IN CODE SEGMENT MEMORY LOCATIONS. (or P.A. = CS*10 + IP = CS:IP)

(b) Instruction Pointer register (IP) ➔ Is a 16-bit register that stores the offset address part of the Physical address, that points to the 64K Code Segment storage locations.

(c) Data Segment Register (DS) ➔ stores "sixteen-bits starting from the MSB" of the base address of Data Segment (Or the four left-most hex digits (as the Least significant hex digit is "$\mathbf{0}_H$")

* REMEMBER, VALUES OF 'DS' AND 'SI or 'DI' COMBINES TO FORM THE (P.A.) POINTING IN DATA SEGMENT MEMORIES. (or P.A. = DS*10 + SI or DI = DS:SI or DS:DI)

(d) Source Index register (SI) ➔ Is a 16-bit register that stores the offset address part of the Physical address, that points to the **source data** stored in the 64K Data or Extra Segment storage locations.

(e) Destination Index register (DI) ➔ Is a 16-bit register that stores the offset address part of the Physical address, that points to the **destination data** stored in the 64K Data or Extra Segment storage locations.

(f) Extra Segment register (ES) ➔ stores "sixteen-bits starting from the MSB" of the base address of Extra Segment (Or the four left-most hex digits (as the Least significant hex digit is "$\mathbf{0}_H$")

* REMEMBER, VALUES OF 'ES' AND 'SI or 'DI' COMBINES TO FORM THE (P.A.) POINTING IN EXTRA SEGMENT MEMORIES. (or P.A. = ES*10 + SI or DI = ES:SI or ES:DI)

(g) Stack Segment Register (SS) ➔ stores "sixteen-bits starting from the MSB" of the base address of Stack Segment (Or the four left-most hex digits (as the Least significant hex digit is "$\mathbf{0}_H$")

* REMEMBER, VALUES OF 'SS' AND 'SP or 'BP' COMBINES TO FORM THE (P.A.) POINTING IN DATA SEGMENT MEMORY LOCATIONS. (or P.A.= SS*10 + SP or ~~BP~~ = SS:SP)

Note: During the lab classes, you will see the segments are setup to overlap each other.

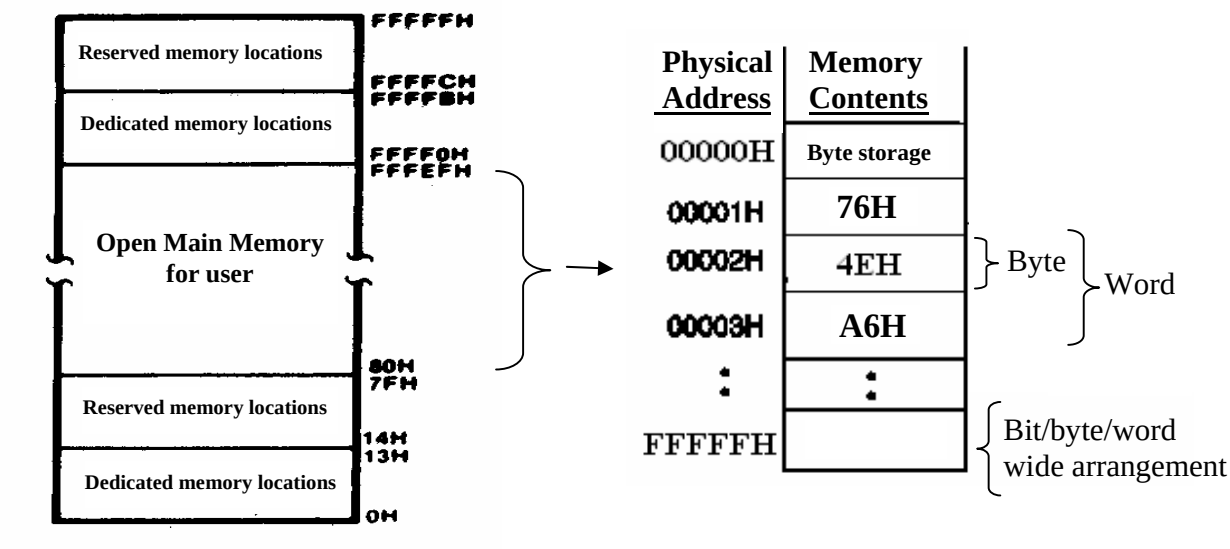
(h) Data register (AX, BX, CX, DX) ➔ are 16-bit registers used to perform addition, subtraction, multiplication, division between two pieces of data. Due to the special function, meant for these registers, they are called; AX = Accumulator register (used by MUL or DIV Instructions) , BX = Base register (used by XLAT Instructions) , CX = counter register (used by LOOP, SHIFT etc. Instructions) and DX = Data register (used also by MUL or DIV Instructions).

Example: **AX** = $23F5_H$ ➔ where the Least-Significant-Byte is stores in **AL** register (as AL = $F5_H$) AND Most-Significant-Byte is stores in **AH** register (as AH = 23_H)

Thus; $(AX)_{word} = (AH)_{byte}\,(AL)_{byte}$; $(BX)_{word} = (BH)_{byte}\,(BL)_{byte}$;

$(CX)_{word} = (CH)_{byte}\,(CL)_{byte}$; $(DX)_{word} = (DH)_{byte}\,(DL)_{byte}$;

2.3 and 2.6: Memory storage:



2.4:Data Types (a) **Integer**: Unsigned and Signed (MSB=>signed bit, '-3'= 2's comp. 3)

(b) **Binary Coded Decimal** (BCD): packed, unpacked. (c) **ASCII**: table of book **pg 30**.

Solve the examples and exercises of chapter 2. **PASS only the solutions of exercise problems.**

EE 390 : Digital System Engineering
Hand out 4 by Dr Sheikh Sharif Iqbal

Chapter 2 (cont'd): **2.10. Status Register for FLAG status:**

Bit 15	**....**	*TF*	*DF*	*IF*	**OF**	**SF**	**ZF**	**AF**	**PF**	**CF**

(a) Carry Flag (CF): Carry or Borrow occurred from the MSB of DATA. [CY=1 (set) or NC=0 (reset)]. Example: $10001111_B + 11110000_B = \mathbf{0}1111111_B$ and CF ➔ CY

(b) Parity FLAG (PF): Depends on the number of binary 1' in DATA. [PE=1 (set) or PO=0 (reset)]. Example: $10001111_B + 11110000_B = 01111111_B$ and PF ➔ PO

(c) Auxiliary FLAG (AF): Carry or Borrow from the MSNibble of DATA. [AC=1 (set) or NA=0 (reset)]. Ex: $10001111_B + 01001000_B = 1101\mathbf{0}111_B$ and AF ➔ AC

(d) Zero FLAG (ZF): Previous program-line results in zero DATA. [ZR=1 (set) or NZ=0 (reset)]. Example: $00000001_B - 00000001_B = \mathbf{0}_B$ and ZF ➔ ZR

(e) Sign FLAG (SF): Depends on MSB of signed DATA. [NG=1 (set) or PL=0 (reset)]. Example: $10000001_B + 00000001_B = \mathbf{1}0000010_B$ and SF ➔ NG

(f) Overflow FLAG (OF): Indicates that Signed DATA is out of range. [OV=1 (set) or NV=0 (reset)].

(g) Direction FLAG (DF): Auto-decrement or Auto Increment in address after execution of string operation. [DN=1 (set) or UP=0 (reset)].

(h) Trap FLAG (TF) ➔ operation mode and Interrupt FLAG (IF) ➔ interrupt request.

Chapter 3: **Software-The microcomputer program:**

- Study 8088 and 8086 and their memory sub-system from software point of view.
- Program ➔ Sequence of Commands of Instructions that defines microcomputer jobs. 8088 understands and performs operations for 117 basic instructions.
- Software ➔ Wide varity of programs that can be run by micro-computer (Languages, Operating systems, Application programs and diagnostics)
- Low level language (Execute faster …) and High level language (User friendly …..)

Assembly Language syntax → Lebel**: OpCode Oprand**; Comments

Source code ➔ **Assembler** (low level lang) / complier (high level lang) ➔ **Machine Code**

Label ➔ Address identifier; Opcode ➔ Operations to be performed ;
Operand ➔ Data on which operations are performed; Comments ➔ describes the operation. *Example:* L1:MOV AX,BX; Copy contents of BX register to AX register
Example 2: INC CX

Source operand *Destination operand*

Converting Assembly language Instruction to Machine Code:

Example: CMC ➔ $11110101_B = F5_H$

Addressing Mode: While executing instructions, Microprocessor can access operand data using different addressing modes such as, (1) Register operand addressing mode, (2) Immediate operand addressing mode, (3) Memory operand addressing mode.

Register operand Addressing modes :
(*See related figures in-chapter 3 of the book*)

Operands are registers only

Example: MOV DX,AX

- This operation will copy the contents of AX register into the DX registerx.
- If $AX=98F3_H$ and $DX=1111_H$, after executing the instruction, DX => $98F3_H$
- **Not allowed** instructions:

  MOV AL,BX

  MOV IP,AX

Instruction	Meaning	Format	Operation	Flags
MOV	copy	MOV D,S	(S) → (D)	Unaffected

Operands

Destination	Source
Memory	Accumulator
Accumulator	Memory
Register	Register
Register	Memory
Memory	Register
Register	Immediate
Memory	Immediate
Seg-reg	Reg16
Seg-reg	Mem16
Reg16	Seg-reg
Memory	Seg-reg

Immediate operand Addressing modes :
(*See related fig's in book)*

Source operands are data itself

* Example: MOV DH,**15**$_H$

- This operation will copy the immediate data of 15_H into the DH registerx.
- If $DX=1111_H$, after executing the instruction, DX => 1511_H.
- **Not allowed** instructions:

  MOV AL,$234A_H$

  MOV 15,AL

Memory Operand Addressing Modes: (*see book fig*)

- This mode access operand-data stored in Memory, by specifying its Physical-Address in different way.

  (a) Direct Memory addressing mode ➔ P.A. is specified directly in the instruction. Such as,

  MOV AL,[1234_H] ➔ MOV AX,[**DS**:1234_H]

  (b) Indirect Memory addressing mode ➔ P.A. is specified indirectly, using; (i) *Register indirect,* (ii) Based, (iii) Indexed, (iv) Based-indexed.

  Ex: MOV SI,1234_H and MOV AX,[SI] *

PA = Segment Base address : Offset address

PA = Segment base : Base + Index + Displacement

$$PA = \begin{Bmatrix} CS \\ SS \\ DS \\ ES \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \end{Bmatrix} + \begin{Bmatrix} SI \\ DI \end{Bmatrix} + \begin{Bmatrix} \text{8-bit displacement} \\ \text{16-bit displacement} \end{Bmatrix}$$

Solve the text book problems of chapter 3