Welcome to the World of Assembly Language Programming

Base Slide Prepared by
Madhusudan Basak
Lecturer
Department of CSE, BUET

Modified by
Abdus Salam Azad, Assistant Professor, CSE, BUET

References

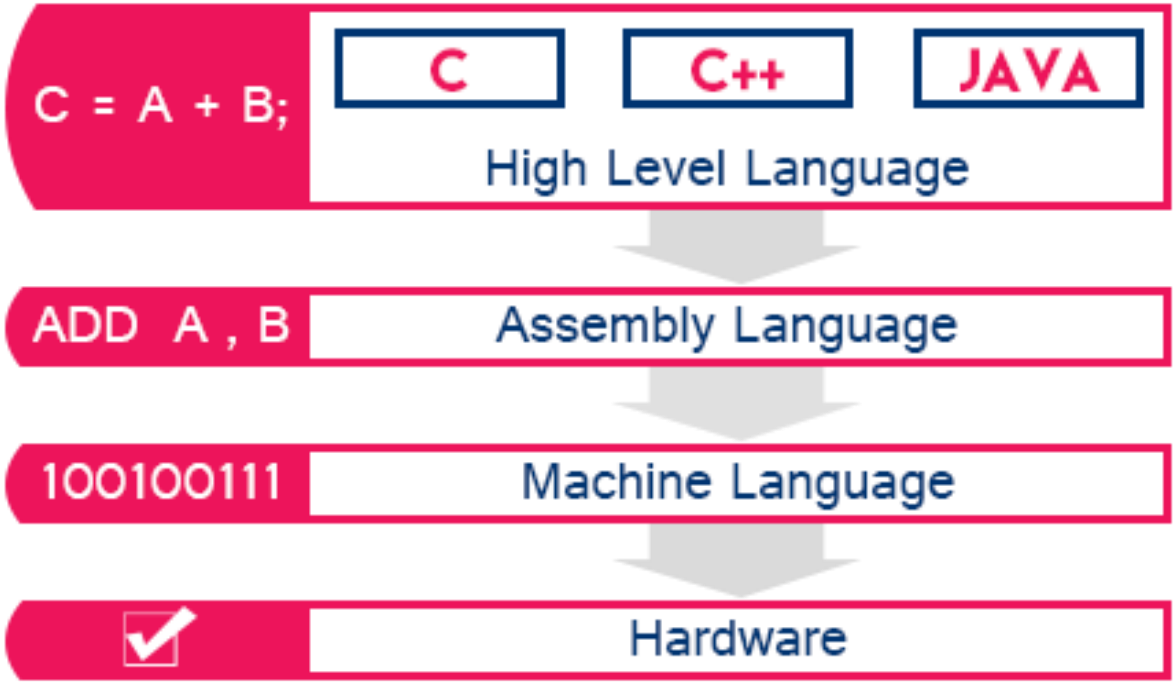
- **Assembly Language Programming and Organization of the IBM PC**

  --- Ytha Yu and Charles Marut

- Assembly Language for the IBM-PC

  --- Kip R. Irvine

What Is Assembly Language ?



Definition (Simplified) from wiki

- a low-level programming language for a computer, in which there is a very strong (generally one-to-one) correspondence between the language and the architecture's machine code instructions.

  – Each assembly language is specific to a particular computer architecture --- Platform Dependent

Why use it ?

- **Direct hardware manipulation**
  - Gives you complete control over the system's resources
- **Performance and efficiency**
- Access to specialized processor instructions

Disadvantages ☹

- Hard and tedious
- **Non-portable(machine dependent)**
- Bug-prone
- Difficult to debug

A question ?

- In what kind of situations, assembly language will be the only/best solution?

- Drivers and communication with custom hardware/electronics.
- An Compiler
- Optimizations

Chapter 1

Microcomputer Systems

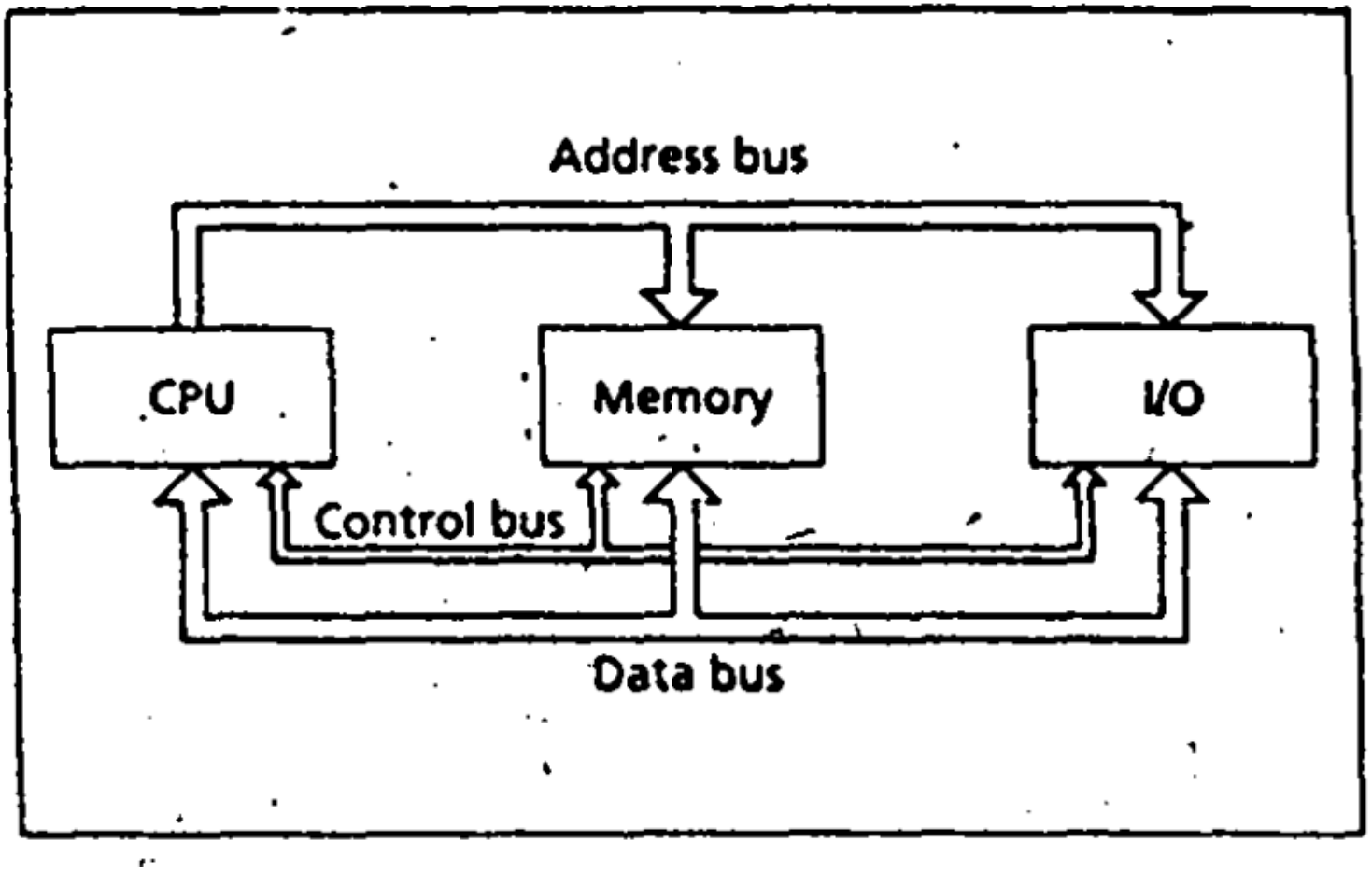
The Components of a Microcomputer System

- Functionally three parts
  - CPU
  - Memory circuits
  - I/O circuits

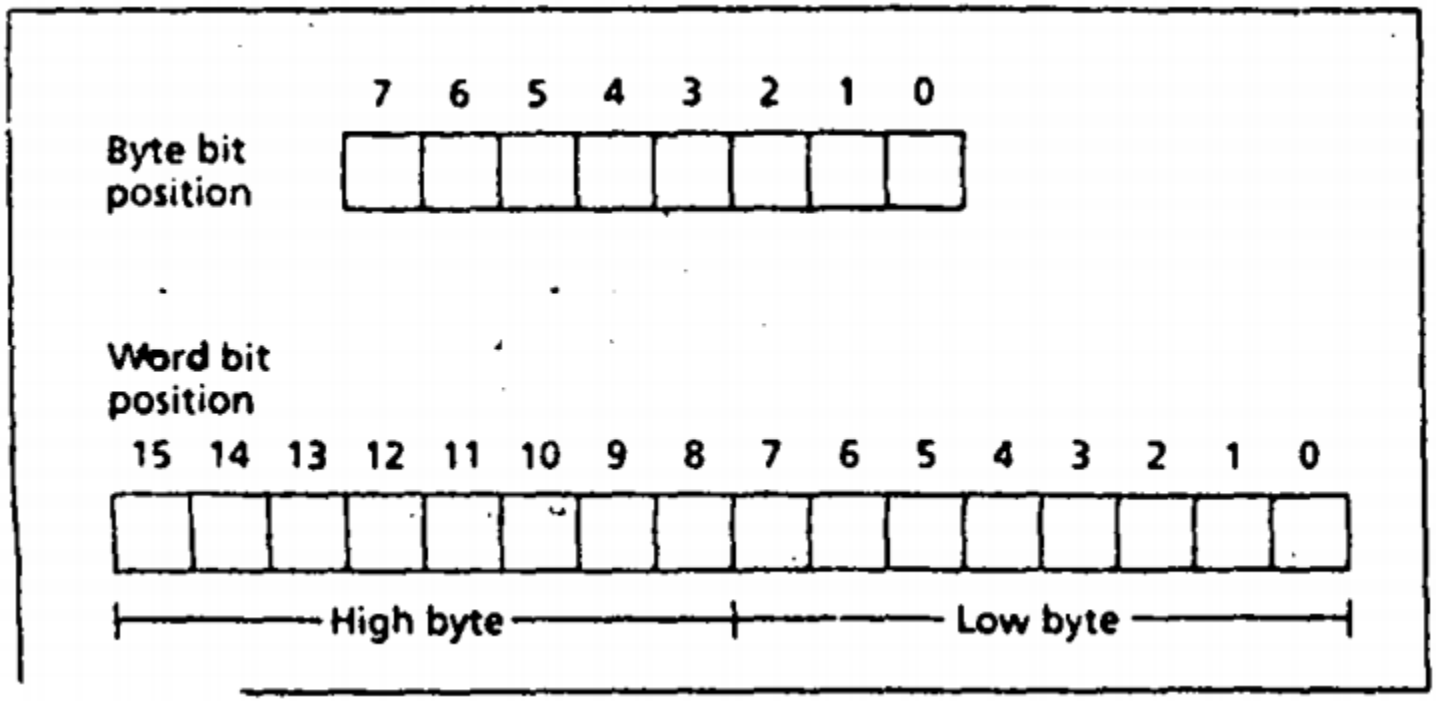
Bus

- A set of wires or connections with which a processor communicates with memory and l/O circuits
- There are three kind of buses:
  - address bus
  - data bus,
  - control bus

Bus Connections



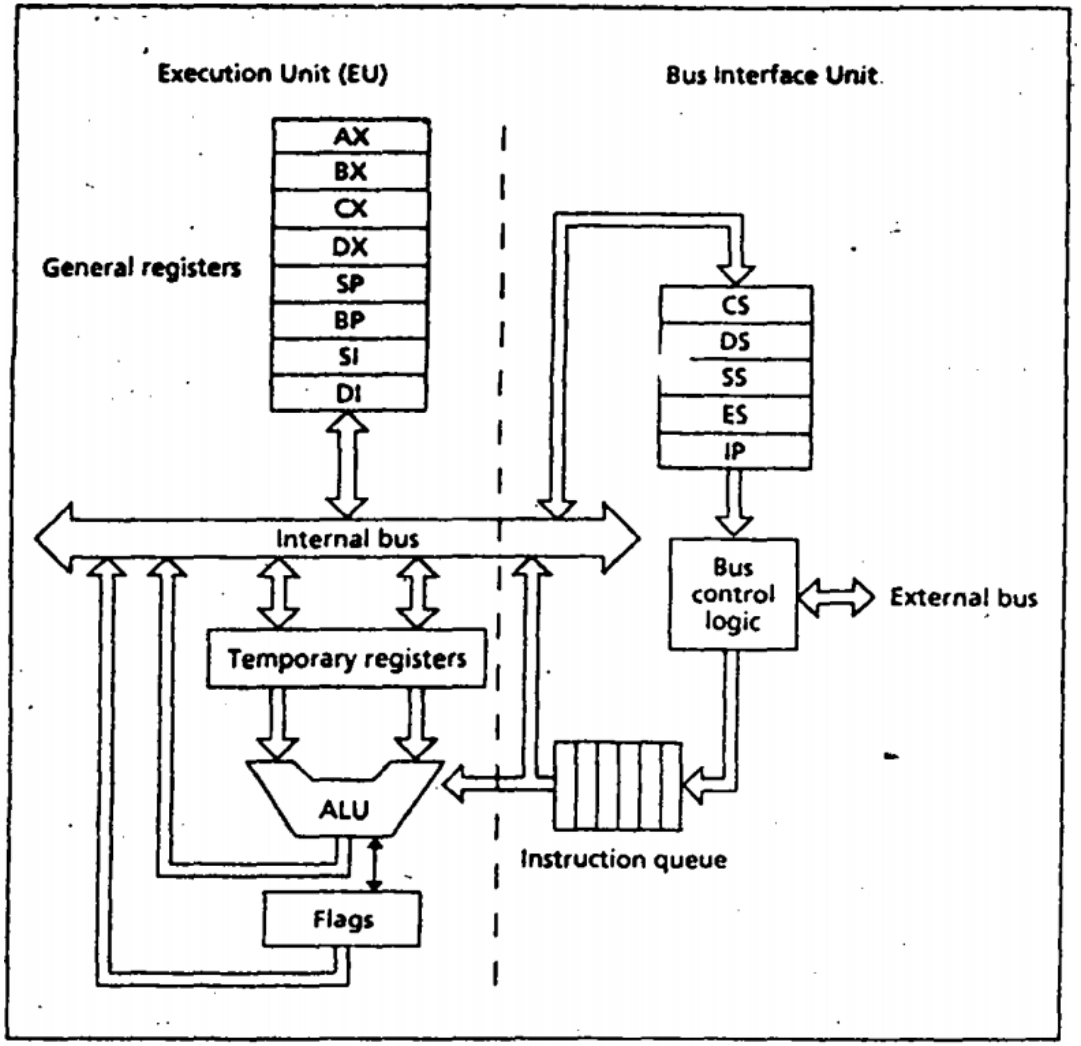
Memory - Bytes and Words



CPU

- Two main components
  - Execution Unit(EU)
  - Bus Interface Unit(BIU)

Intel 8086 Microprocessor Organization



Execution Unit

- The purpose of the execution unit (EU} is to execute Instructions.
  - It contains a circuit called the arithmetic and logic unit (ALU).
    - The ALU can perform arithmetic (+, - , x *, I)* and logic (AND, OR, NOT} operations.
- The data for the operations are stored in registers
- **A register is like a memory location except that *we* normally refer to it by a name rather than a number.**
- The EU has eight registers for storing data; their names arc AX, BX, CX, DX, SI, DI, BP, and SP.

Bus Interface Unit

- The Bus Interface unit (BIU) facilitates communication between the EU and the memory or I/0 circuits.
- It is responsible for transmitting addresses, data, and control signals on the buses.
- Its registers are named CS, DS, ES, SS, and IP.
  - They hold addresses of memory locations.
  - The IP (instruction pointer) contains the address of the next instruction to be executed by the EU.

Instruction Execution

- Fetch
  1. Fetch an instruction from memory.
  2. Decode the instruction to determine the operation.
  3. Fetch data from memory if necessary.
- Execute
  4. Perform the operation on the data.
  5. Store the result in memory if needed

Chapter 3

ORGANIZATION OF THE IBM PERSONAL COMPUTERS

Organization of the 8086/8088 Microprocessors

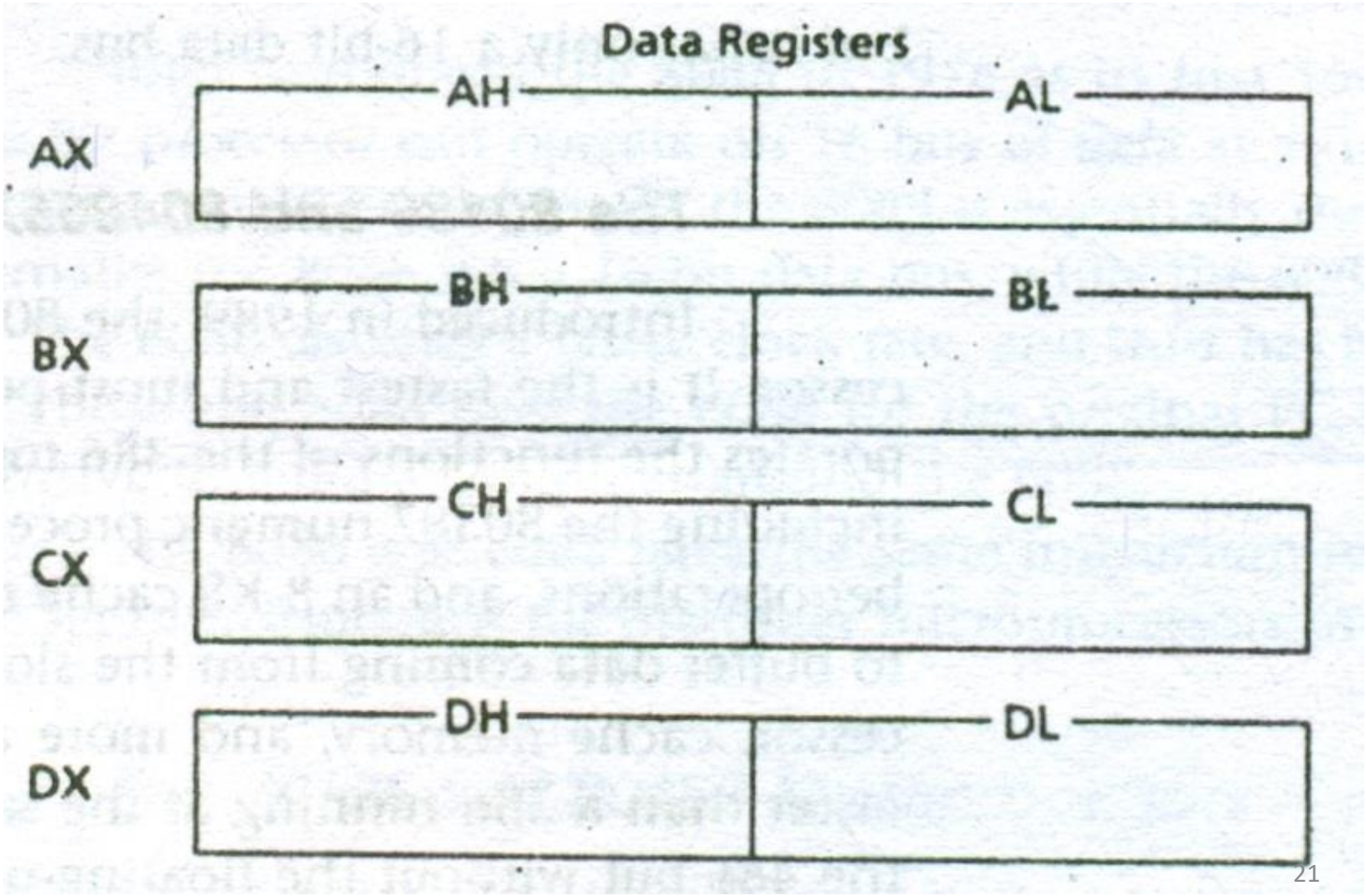
- Registers
  - There are fourteen 16 bit registers
  - 3 Types of registers
    - Data Registers – holds data for operation
    - Address Registers – holds addresses of instruction or data
    - Status Register – holds current status of processor

Data Registers

- AX (Accumulator Register)
- BX (Base Register)
- CX (Count Register)
- DX (Data Register)

- Each one can be used by parts
  - e.g., AX has two equal parts
    - The higher part (8 bit) is referred as AH and
    - The lower part – AL
    - both of them can be used independently

Data Registers



Memory Organization

- 8086 features 1 20 bit physical address for memory
  - 2^20 == 1MB Memory is supported
- However, all registers are of 16 bit
- The memory is divided (non-disjoint) into segments

Segment

- Each segment consists of a block of 2^16 consecutive bytes in the memory
  - 64KB in size
  - Each memory location in a memory segment is addressable by a 16 bit **offset**
- Each segment is identified with a 16 bit **segment number**
  - Hence there are 2^16 segments
  - Numbered from 0000h to FFFFh

Logical Address and Physical Address

- Each Location in memory can be addressed by specifying a **segment number and and offset within** the segment
  - Written as segment:offset
  - Called the logical address
- To find the physical address from a logical address
  - Shift left the *segment number* by four bit, then add *offset*
  - *segment*10h + offset*

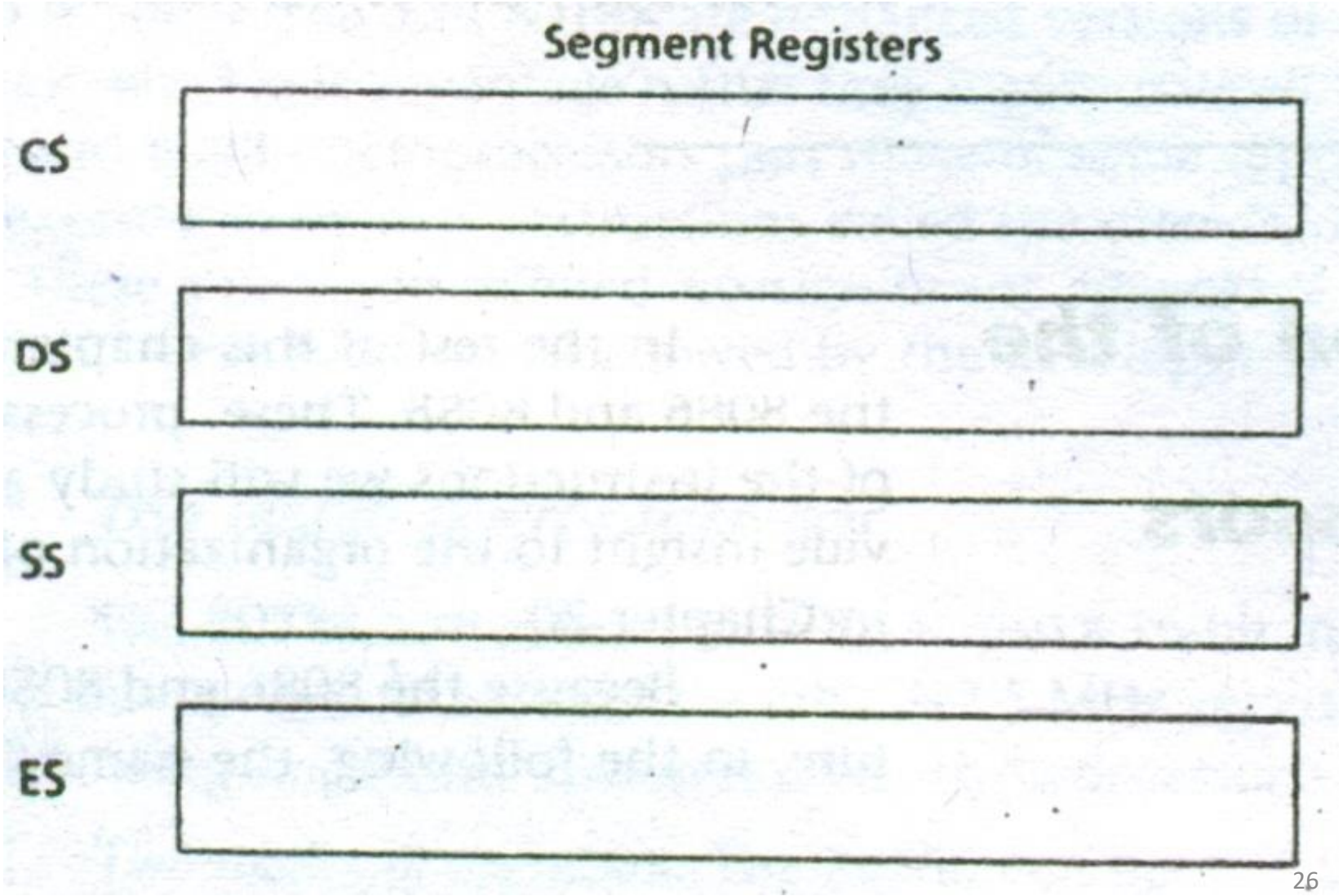
- If you have a segment A4FBh and offset 4872h. Then what is the 20 bit physical address?

A4FB0h

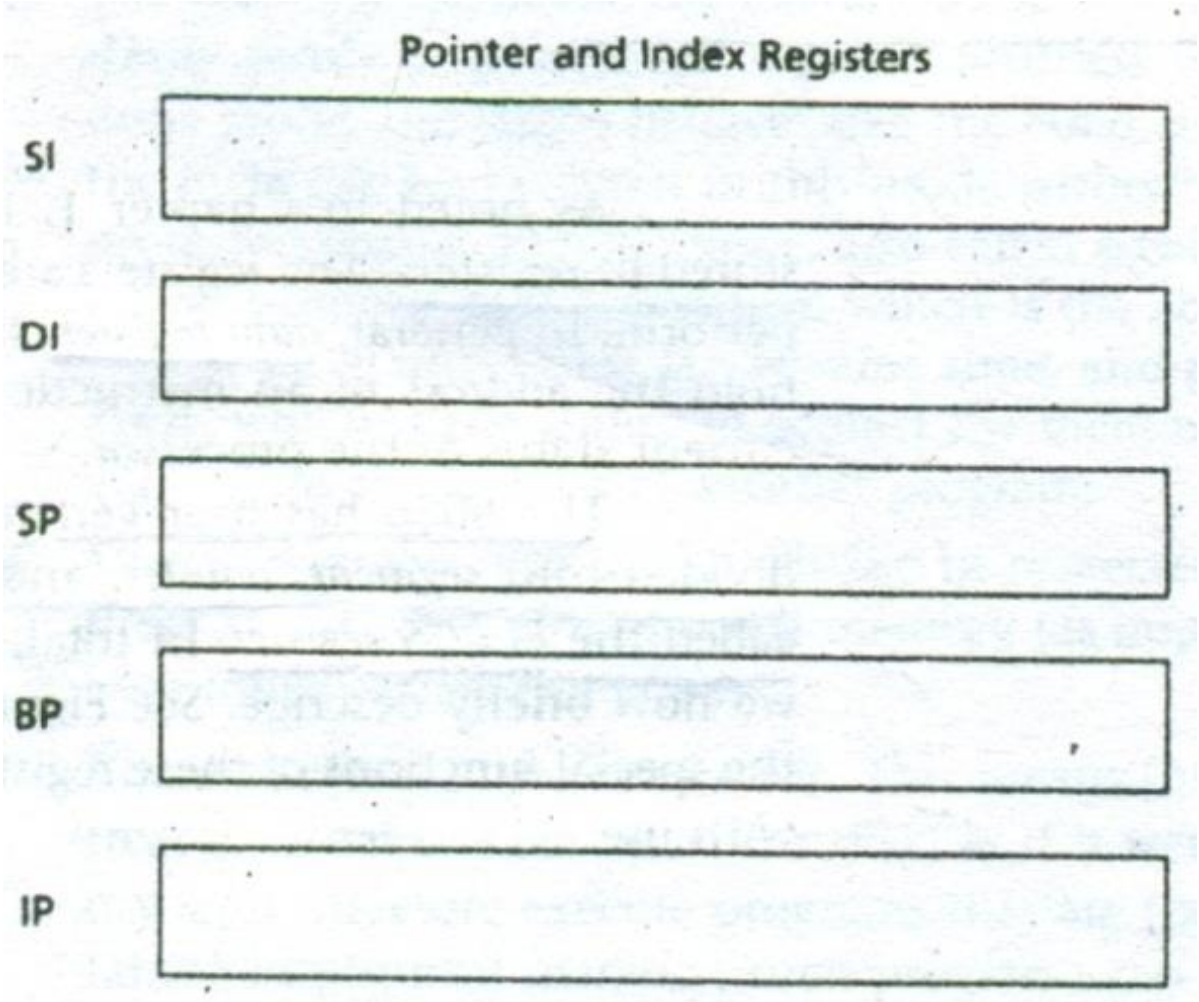
+4872h

A9822h (20 bit physical address)

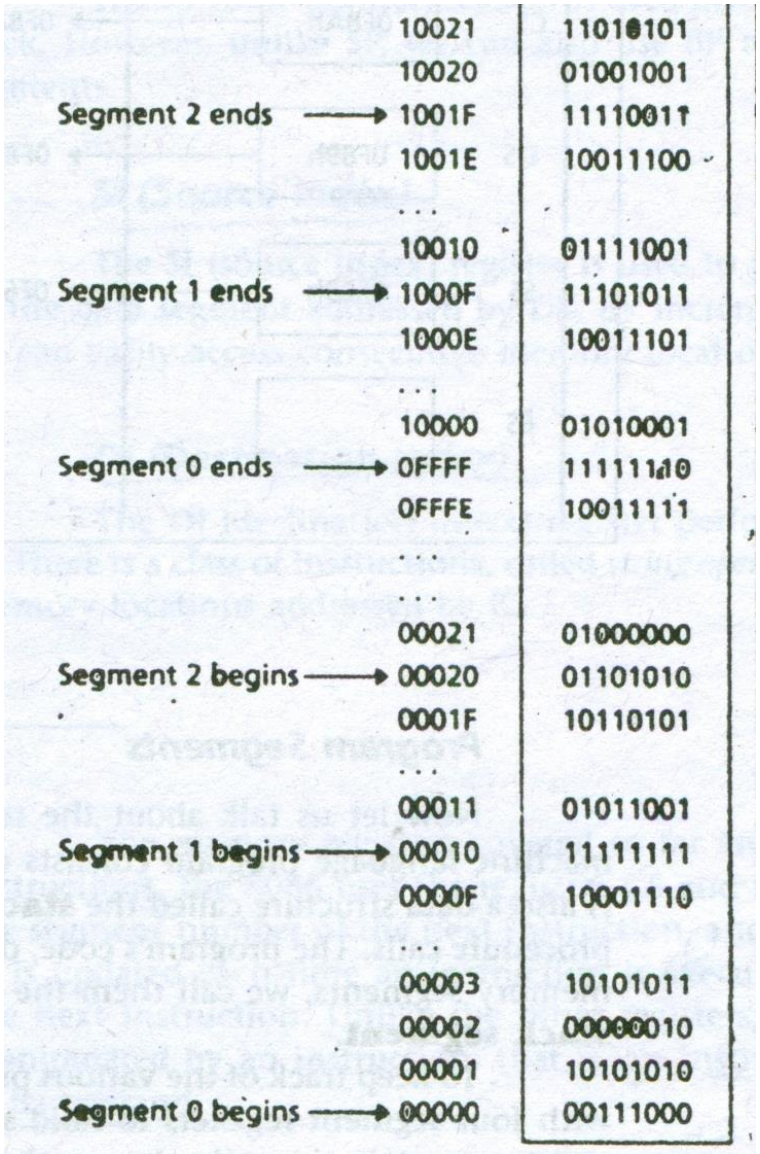
Segment Registers(Address Registers)



Pointer and Index Registers, Instruction pointers (Address Registers)



Overlapping Scenario



Overlapping Scenario

Example 3.1 For the memory location whose physical address is specified by 1256Ah, give the address in segment:offset form for segments 1256h and 1240h.

Solution: Let X be the offset in segment 1256h and Y the offset in segment 1240h. We have

$$1256Ah = 12560h + X \text{ and } 1256Ah = 12400h + Y$$

and so

$$X = 1256Ah - 12560h = Ah \text{ and } Y = 1256Ah - 12400h = 16Ah$$

thus

$$1256Ah = 1256:000A = 1240:016A$$

Physical and Logical address calculation

Example 3.2 A memory location has physical address 80FD2h. In what segment does it have offset BFD2h?

Solution: We know that

$$\text{physical address} = \text{segment} \times 10\text{h} + \text{offset}$$

Thus

$$\text{segment} \times 10\text{h} = \text{physical address} - \text{offset}$$

in this example

$$\begin{array}{l} \text{physical address} = 80\text{FD2h} \\ \underline{\quad - \text{offset} = \ \ \text{BFD2h}} \\ \text{segment} \times 10\text{h} = 75000\text{h} \end{array}$$

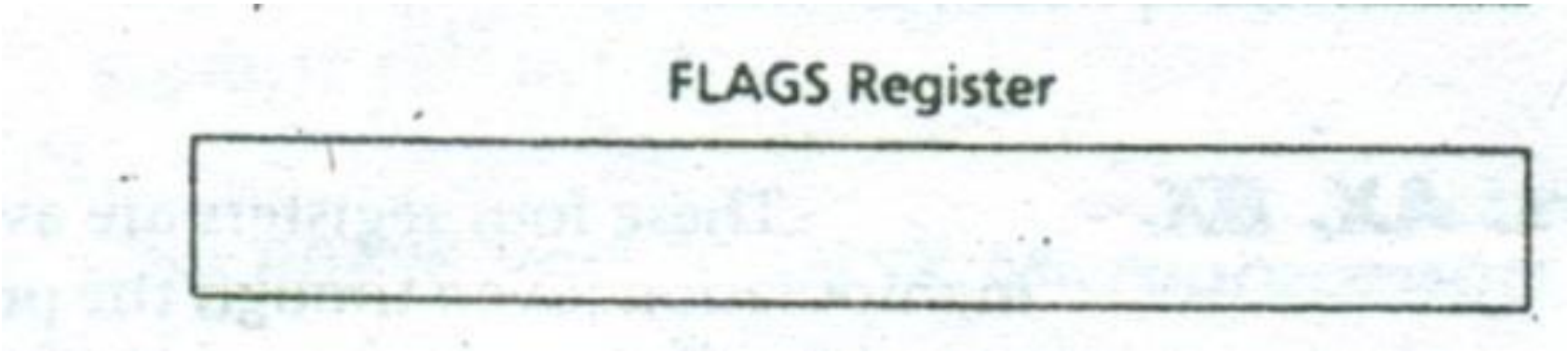
So the segment must be 7500h.

Segment Registers

- Code, Data, and Stack are maintained in different segments
- Stack is used by the processor to implement procedure calls.

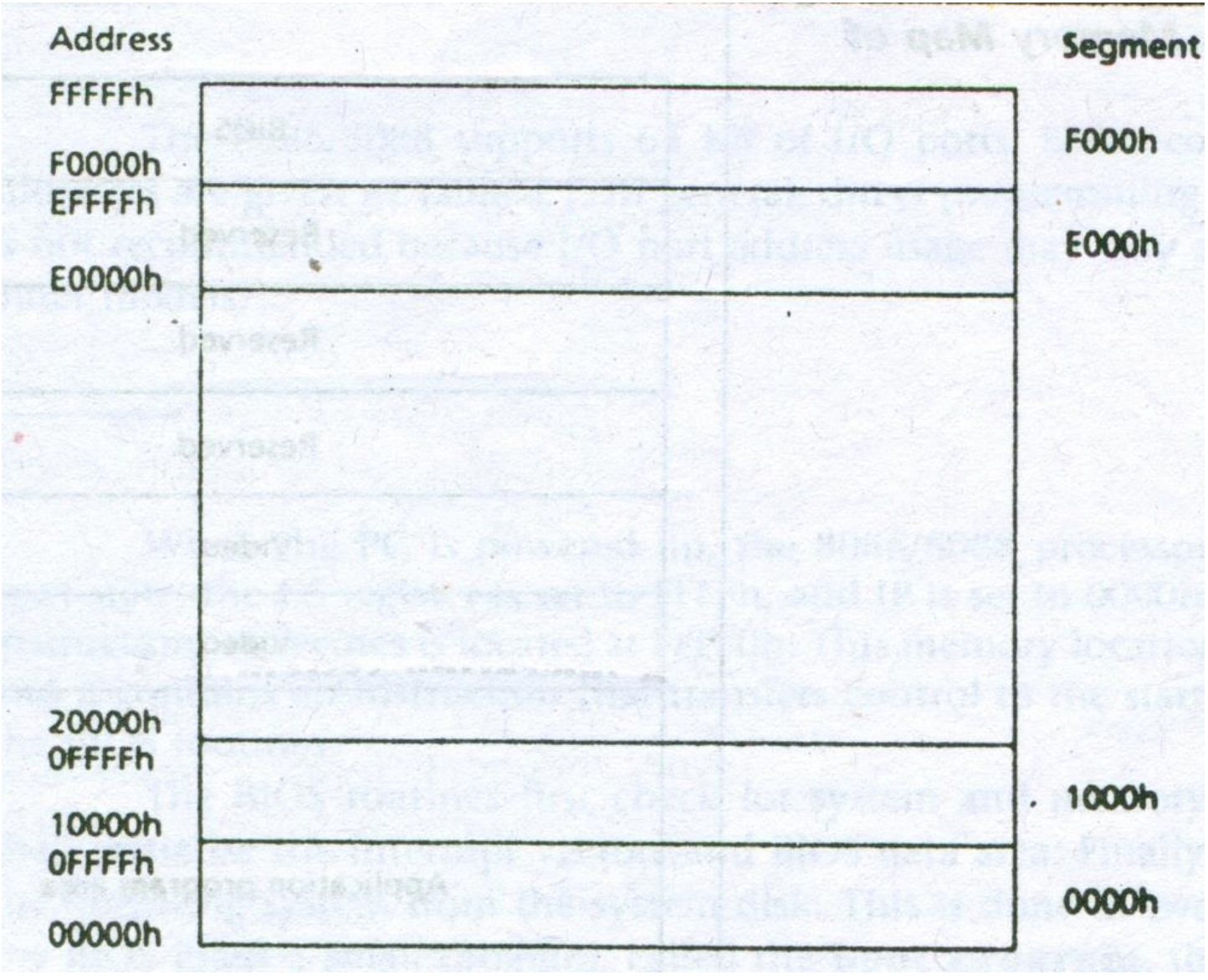
Task	Segment Registers	Pointer and Index Registers
Code	CS	IP
Data	DS	SI
	ES	EI
Stack	SS	SP
		BP

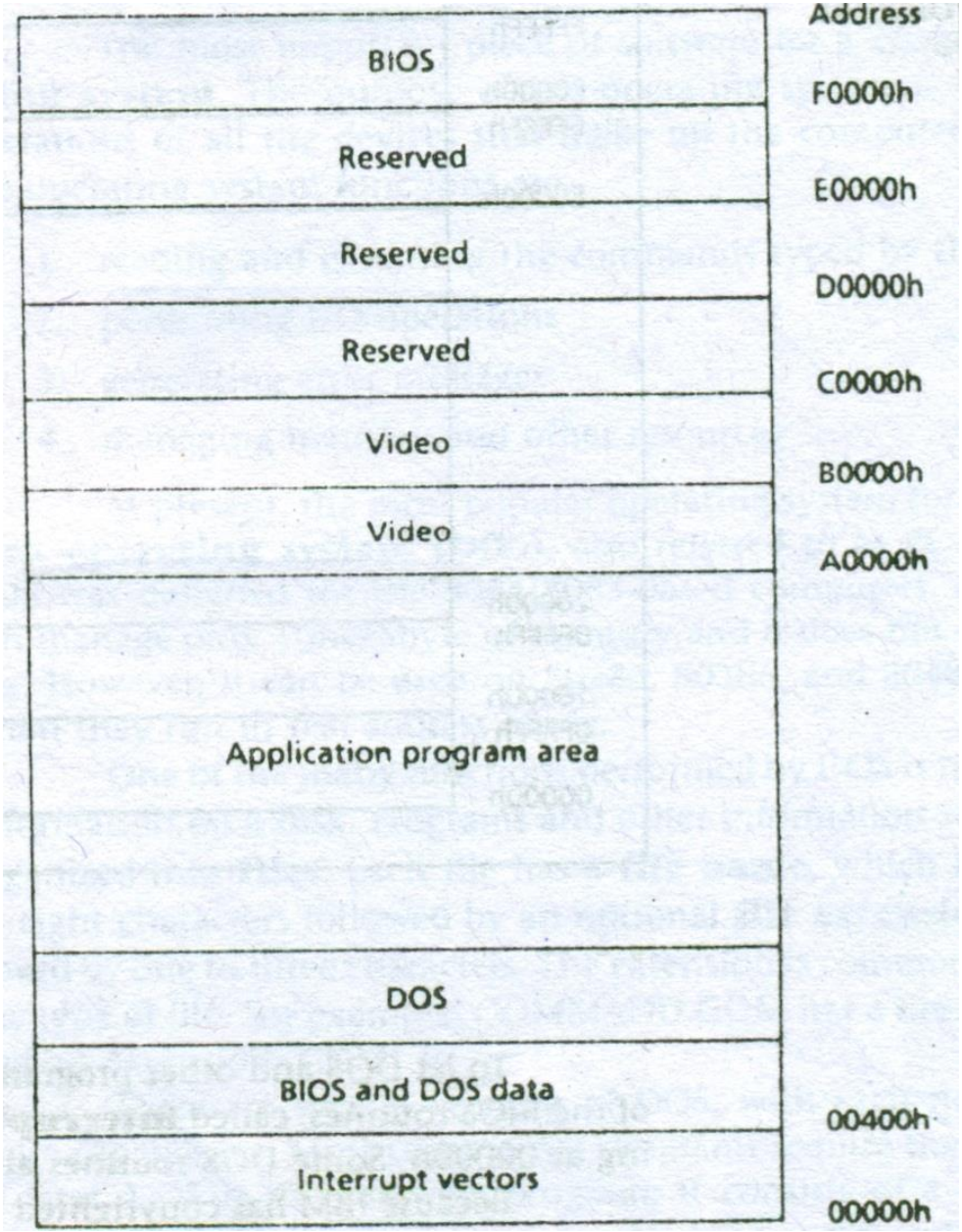
FLAGS Register(Status Register)



Memory Organization of the PC

- 8086/8088 has only 1MB of memory
- Not all memory for application programs
- Interrupt Vector, Video Display Memory etc are needed
- IBM fixed all the positions and allowed all to live happily


Address
Segment
FFFFFh
F0000h
F000h
EFFFFh
E0000h
E000h
20000h
0FFFFh
10000h
1000h
0FFFFh
00000h
0000h


Address
BIOS
F0000h
Reserved
E0000h
Reserved
D0000h
Reserved
C0000h
Video
B0000h
Video
A0000h
Application program area
DOS
BIOS and DOS data
00400h
Interrupt vectors
00000h

I/O ports addresses

- The 8086/8088 supports· 64 KB of I/0 ports
- Interrupt controller (20h-21h)
- Keyboard controller (60h-63h)
- etc...


Thank you!