8086 MICROPROCESSOR

K. RADHIKA

Associate Professor

Department of ECE

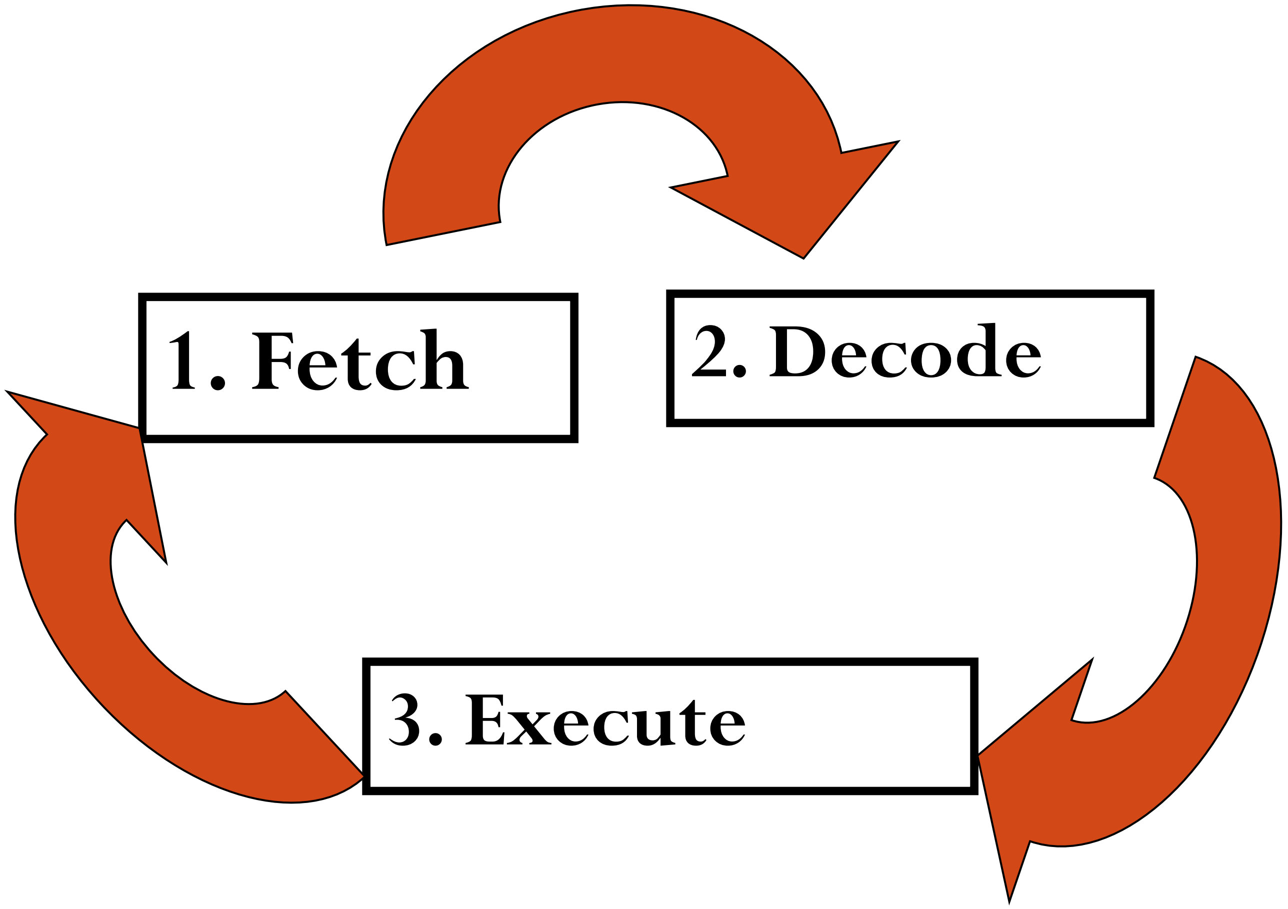
Jyothishmathi Institute of Technology & Science

INTRODUCTION TO 8086

Overview of Microcomputer Systems

- Two principal components
  1. Hardware
     - CPU,
     - Timing circuits
     - Memory Units
     - Input / Output Subsystems
     - Bus control Logic
     - System Bus
  2. Software
     - System Software
     - User Software

Operation of mP



8086 ARCHITECTURE

Features

- 16-bit microprocessor , most of instructions are designed to work with 16 bit binary words
- Data bus : 16 bit (read data from or write data to memory and ports either 16 bits or 8 bits at a time
- Address bus : 20 bit (can address upto 2^{20}=1048576 = 1MB memory locations). [Address range : 00000H to FFFFFH]
- It provides 14, 16-bit registers.
- It has multiplexed Address and Data bus AD0- AD15 and A16 – A19.

8086 Architecture (continued…)

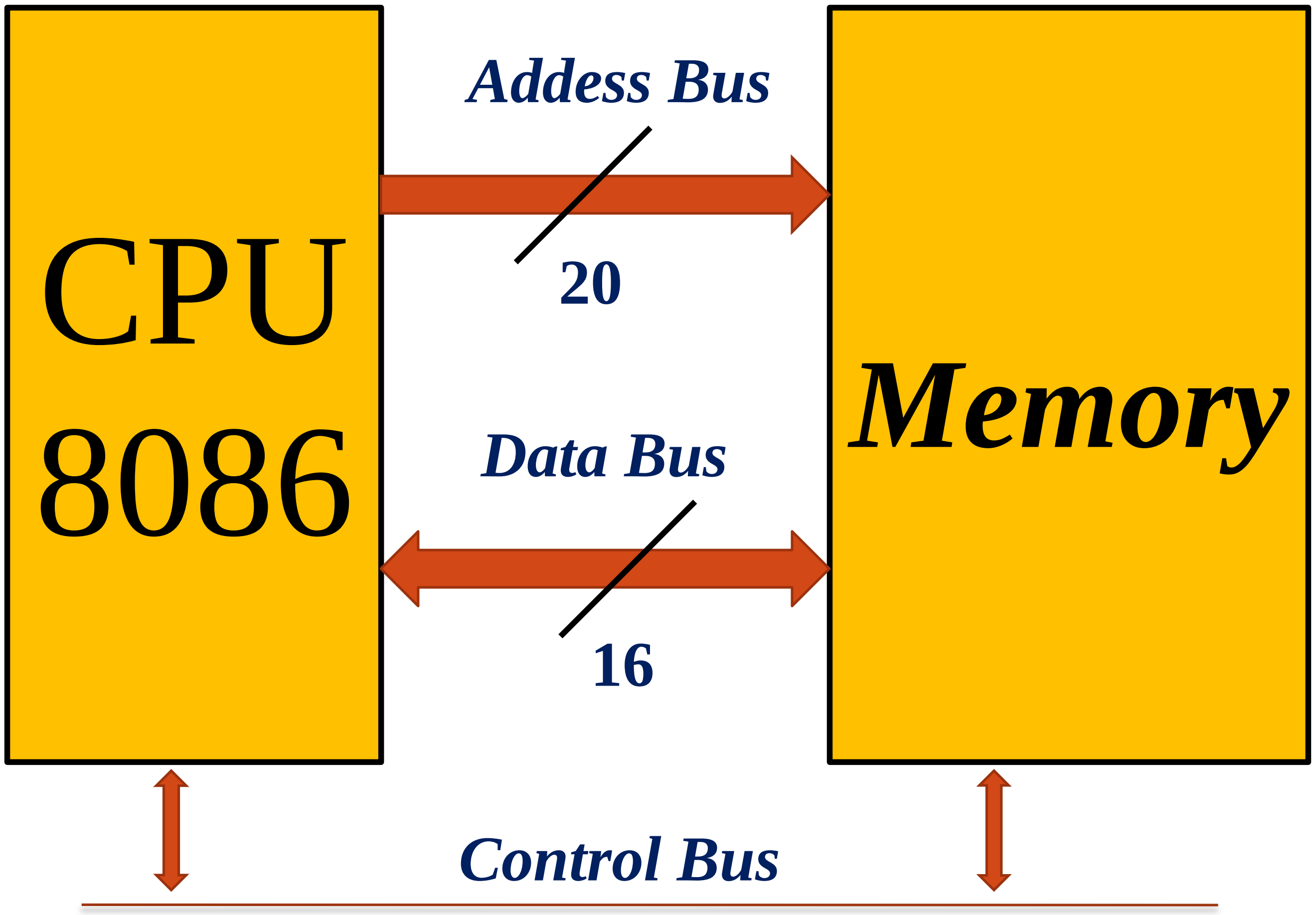
8086 Features

- **Has 'Look ahead' feature to increase the throughput. (i.e) It can prefetch upto 6 bytes from memory and queues them in order to speed up instruction execution.**
- **40 pin dual in line package**
- **Supply Voltage: +5V ; Clock Speed: 5MHz, 8MHz, 10MHz**

- **8086 is designed to operate in two modes, Minimum and Maximum.**

8086 Architecture (continued…)

CPU-MEMORY INTERFACE



8086 Architecture

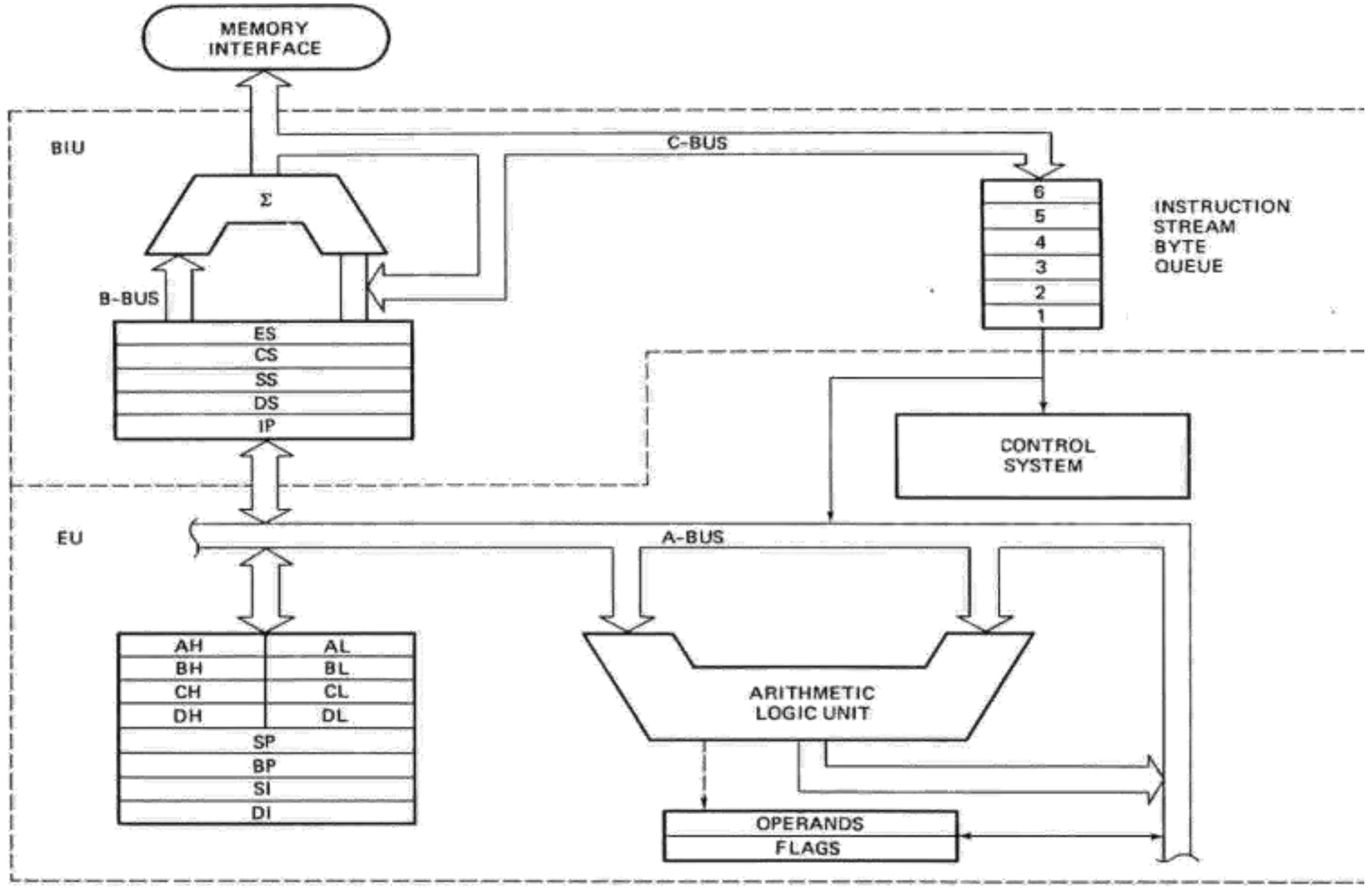
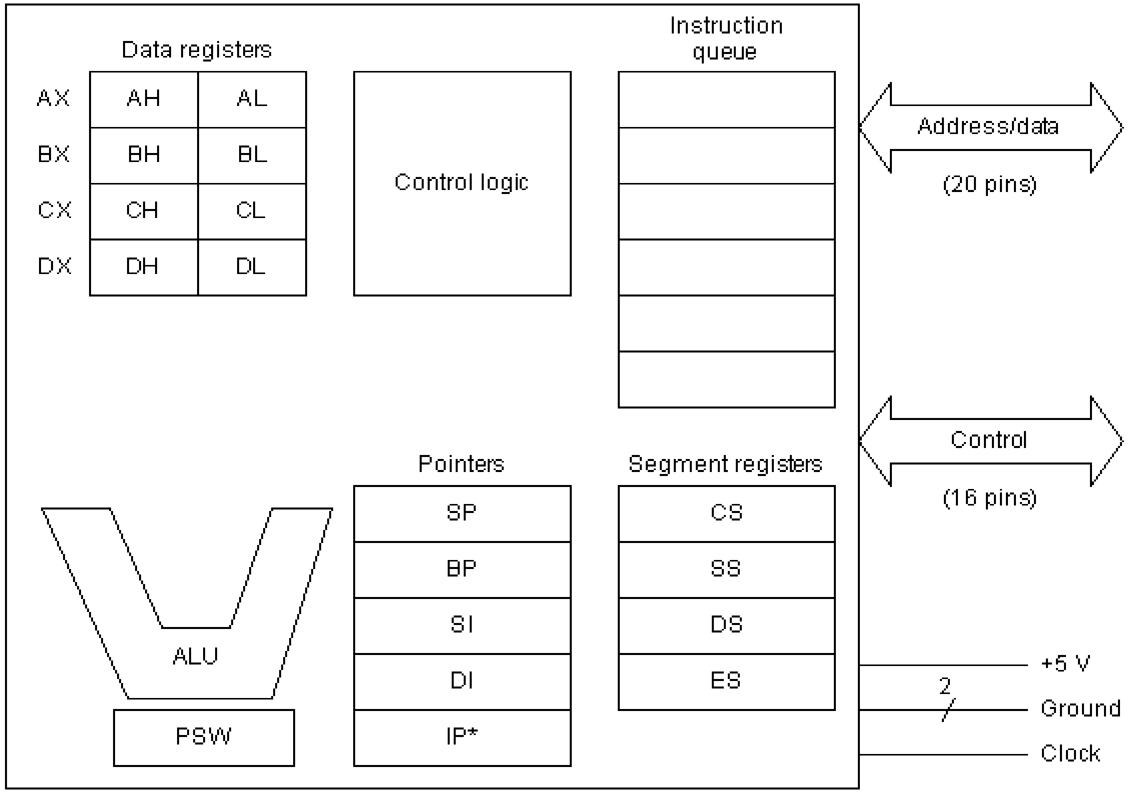


FIGURE 2-7 8086 internal block diagram. (*Intel Corp.*)

8086 Architecture (continued…)

8086 ARCHITECTURE

Figure 2-2 8086's internal configuration.



*For the 8086 the program counter is called the instruction pointer (IP).

In addition to serving as arithmetic registers, the BX, CX and DX registers play special addressing, counting, and I/O roles:

- BX may be used as a base register in address calculations.
- CX is used as an implied counter by certain instructions.
- DX is used to hold the I/O address during certain I/O operations.

8086 Architecture (continued…)

8086 ARCHITECTURE

The 8086 has two parts:

- **the Bus Interface Unit (BIU)**
  - **Fetches instructions,**
  - **Reads and Writes data, and**
  - **computes the 20-bit address for memory operands**
  - **Transfers instruction bytes into the 6 byte FIFO queue**

- **the Execution Unit (EU)**
  - **Decodes and**
  - **Executes the instructions using the 16-bit ALU.**

Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining.

8086 Architecture (continued…)

BIU Registers

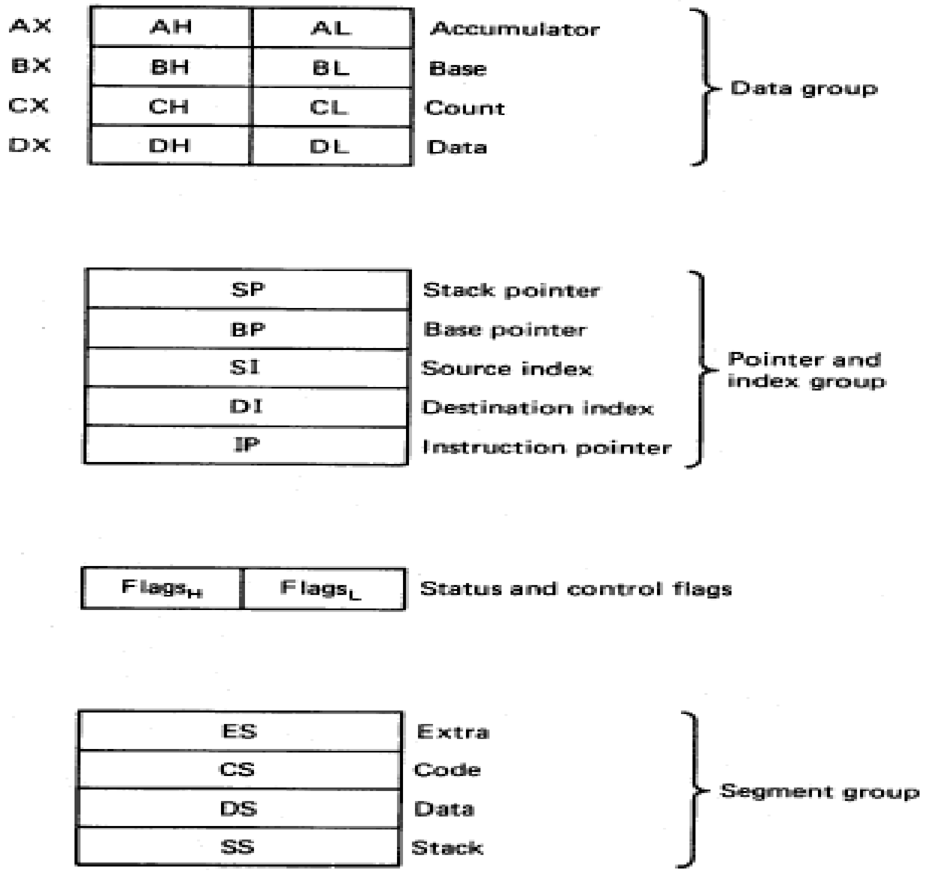
CS	Code Segment
DS	Data Segment
ES	Extra Segment
SS	Stack Segment
IP	Instruction Pointer

EU Registers

AX	AH	AL	Accumulator
BX	BH	BL	Base Register
CX	CH	CL	Count Register
DX	DH	DL	Data Register
	SP		Stack Pointer
	BP		Base Pointer
	SI		Source Index Register
	DI		Destination Index Register
	Flags		Flag Register

The 8086 Microprocessor: Registers

In total there are Fourteen 16-bit registers in an 8086



8086 Architecture (continued…)

The 8086 Microprocessor: Registers

Registers

- Registers are in the CPU and are referred to by specific names
- Data registers
  - Hold data for an operation to be performed
  - There are 4 data registers (AX, BX, CX, DX)
- Address registers
  - Hold the address of an instruction or data element
  - Segment registers (CS, DS, ES, SS)
  - Pointer registers (SP, BP, IP)
  - Index registers (SI, DI)
- Status register
  - Keeps the current status of the processor
  - The status register is called the FLAG register

8086 Architecture (continued…)

Data Registers: AX, BX, CX, DX

- **Instructions execute faster if the data is in a register**
- **Data Registers are general purpose registers but they also perform special functions**
- **AX, BX, CX, DX are the data registers**
- **Low and High bytes of the data registers can be accessed separately**
  - **AH, BH, CH, DH are the high bytes**
  - **AL, BL, CL, DL are the low bytes**

- **AX**
  - Accumulator Register
  - Used in Arithmetic, Logic and Data Transfer instructions
  - Used in Multiplication and Division operations
  - Used in I/O operations
- **BX**
  - Base Register
  - Also serves as an address register
  - Used in array operations
  - Used in Table Lookup operations (XLAT)
- **CX**
  - Count register
  - Used as a Loop Counter
  - Used in shift and rotate operations
- **DX**
  - Data register
  - Used in Multiplication and Division
  - Also used in I/O operations

Pointer and Index Registers

- **Contains the offset addresses of memory locations**
- **Can also be used in Arithmetic and other operations**
- **SP: Stack pointer**
  - **Used with SS to access the stack segment**
- **BP: Base Pointer**
  - **Primarily used to access data on the stack**
  - **Can be used to access data in other segments**

8086 Architecture (continued…)

Pointer and Index Registers

- **SI: Source Index register**
  - **is required for some string operations**
  - **SI is associated with the DS in string operations.**
- **DI: Destination Index register**
  - **is also required for some string operations.**
  - **DI is associated with the ES in string operations.**

The SI and the DI registers may also be used to access data stored in arrays

Segment Registers - CS, DS, SS and ES

- **Are Address registers**
- **Stores the memory addresses of instructions and data**
- **Memory Organization**
  - **20 bit address line addresses 1 MB of memory**
  - **Each byte in memory has a 20 bit address**
  - **Addresses are expressed as 5 hex digits from 00000 - FFFFF**
  - **Problem: 20 bit addresses are TOO BIG to fit in 16 bit registers!**
  - **Solution: Memory Segment**
    - **A segment number is a 16 bit number**
    - **Segment numbers range from 0000 to FFFF**
    - **Block of 64K (65,536) (i.e 2^{16})consecutive memory bytes**
    - **Within a segment, a particular memory location is specified with an offset**
    - **An offset also ranges from 0000 to FFFF**

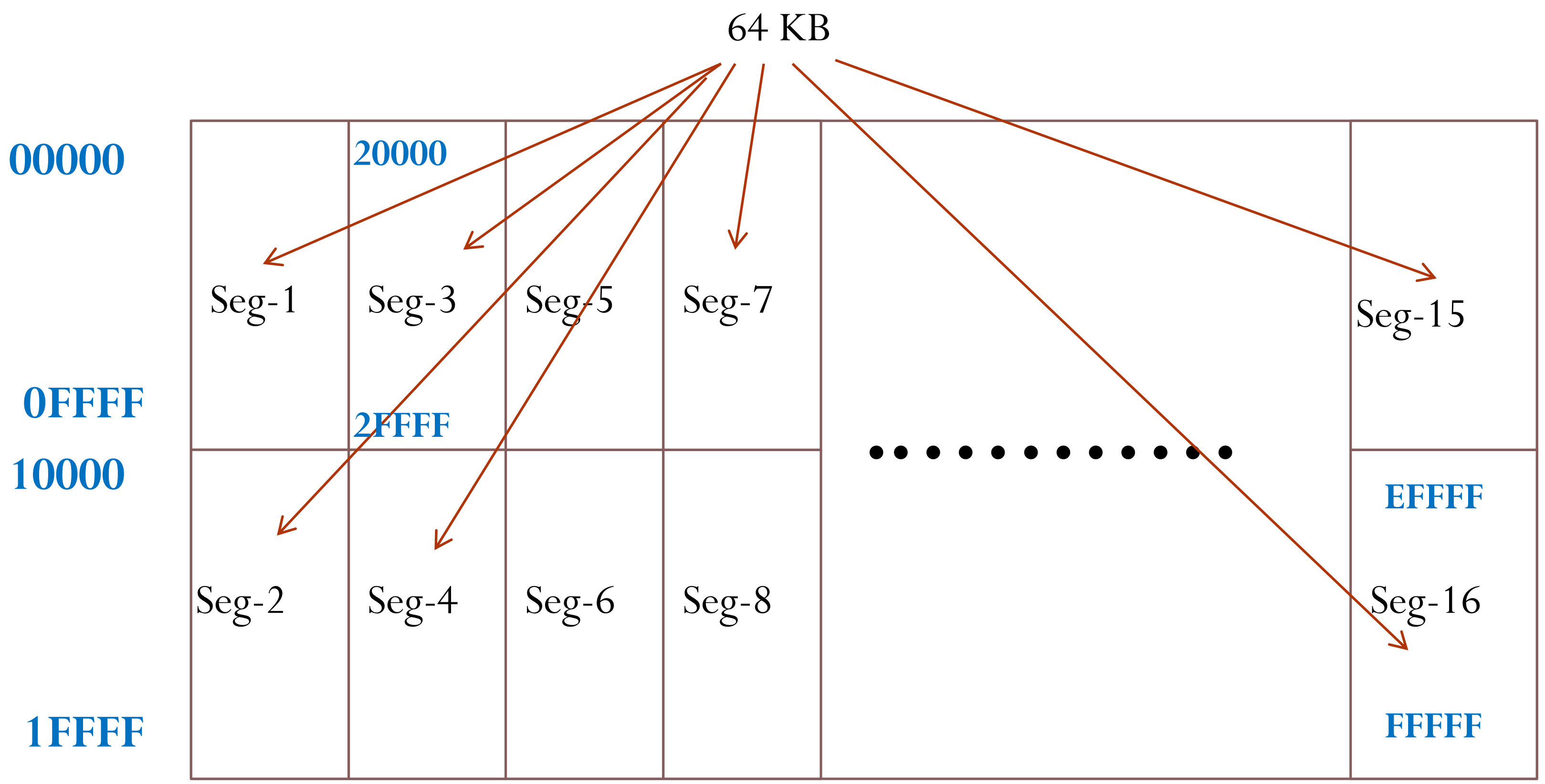
Segmented Memory Architecture

- Memory space is divided into overlapping segments
- Each segment is of 64 Kbytes
- Segment address begins at an address that is divisible by 16_{10} or 10_{16}
- Segment register contains the starting address of a segment.
- 16 bit words will be stored in two consecutive memory locations.
  - If first byte of the data is stored at an even address, 8086 reads the entire word in one operation.
  - Example if the 16 bit data 2607H is stored at even address 00520H, then for instruction MOV BX, [00520],
  - 8086 reads the first byte and stores the data in BL and reads the second byte and stores the data in BH.

BL ← (00520) BH ←(00521)

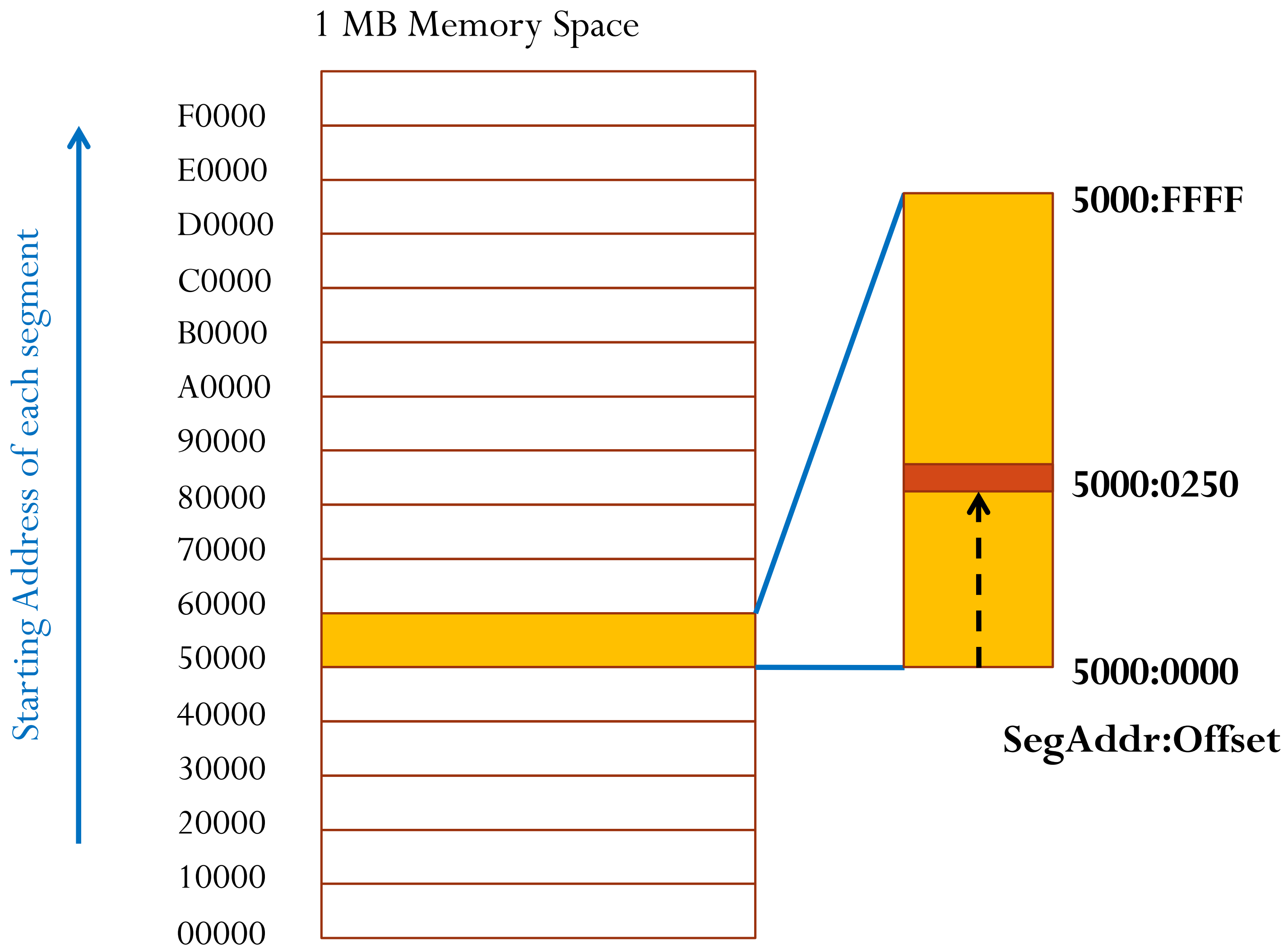
- If the first byte of data is stored at an odd address, 8086 needs two operation to read the 16 bit data.
- Example if the 16 bit data F520H is stored at odd address 00521H, then for instruction MOV BX, [00521],
- In first operation, 8086 reads the 16 bit data from the 00520 location and stores the data of 00521 location in register BL and discards the data of 00520 location.
- In second operation, 8086 reads the 16 bit data from the 00522 location and stores the data of 00522 location in register BH and discards the data of 00523 location.

1MB Memory Space divided into non-overlapping segments



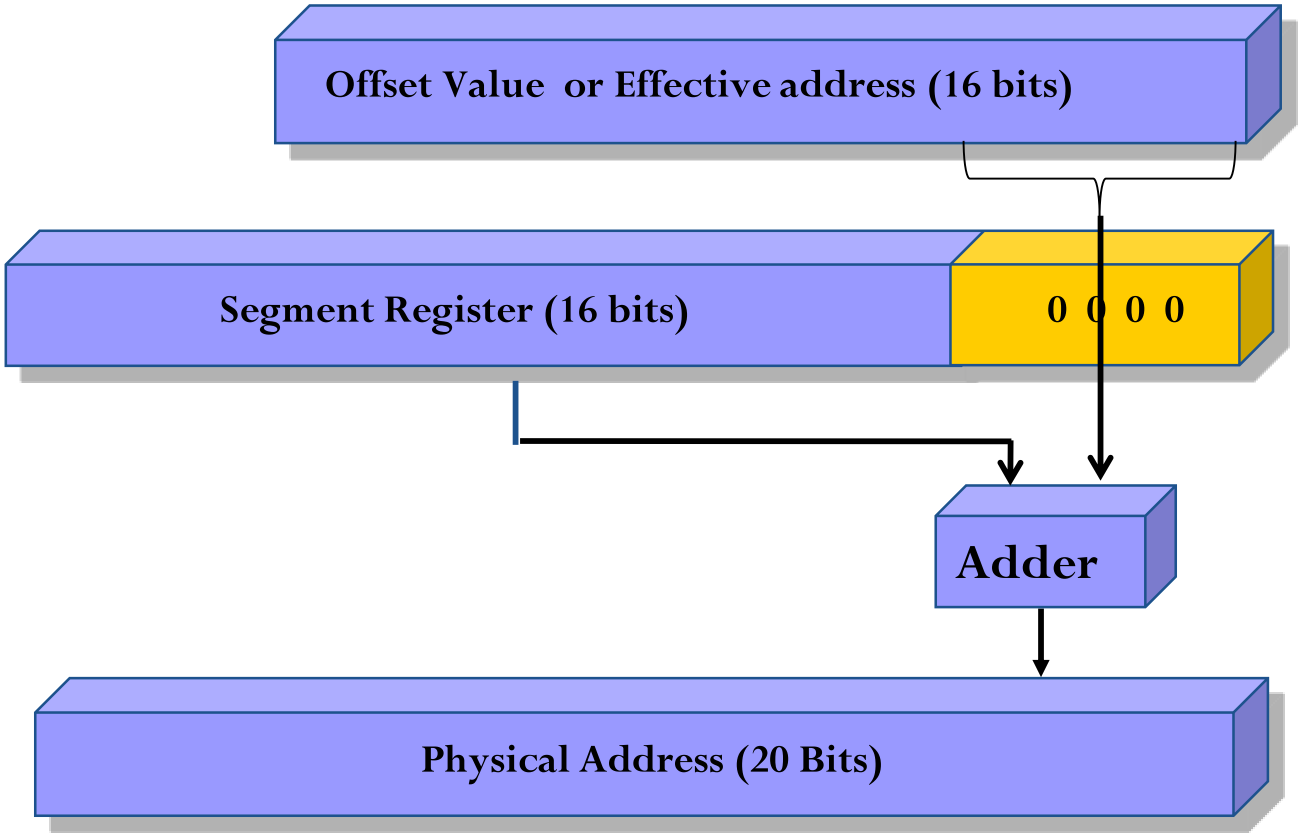
Segmented memory addressing:

Absolute Address = Four bit left shifted16-bit segment value added to a 16-bit offset



Physical Memory Address Generation

- **The BIU has a dedicated adder for determining Physical memory addresses**


Offset Value or Effective address (16 bits)
Segment Register (16 bits)
0 0 0 0
Adder
Physical Address (20 Bits)

Physical Memory Address Generation

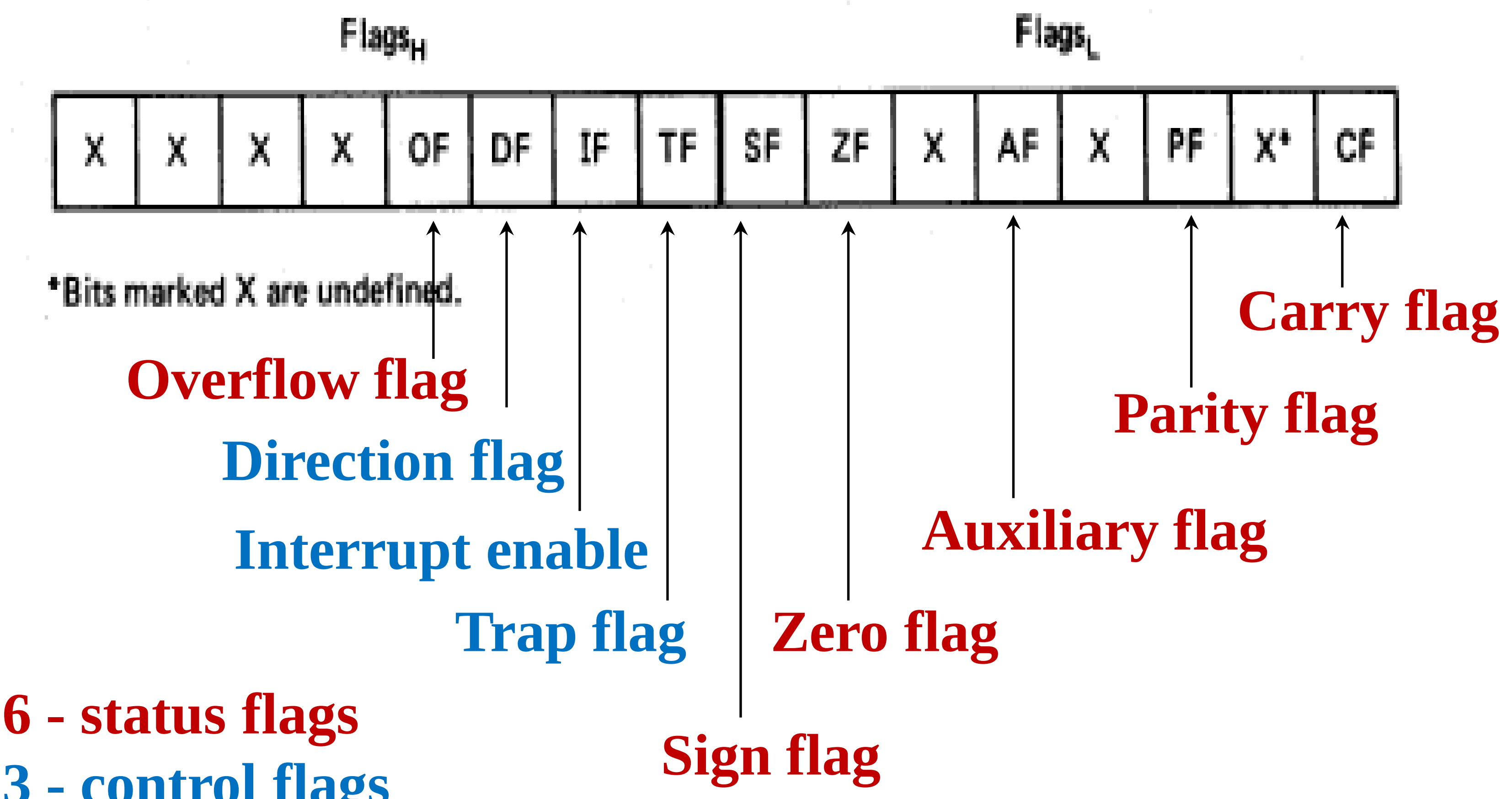
- **Logical Address is specified as Segment:Offset**
- **Physical address is obtained by shifting the segment address 4 bits to the left and adding the offset address**
- **Thus the physical address of the logical address A4FB:4872 is**

$$\begin{array}{rcr} \text{A4FB0} & \rightarrow & 1010\ \ 0100\ \ 1111\ \ 1011\ \ 0000 \\ +\ \underline{\quad 4872} & \rightarrow & 0100\ \ 1000\ \ 0111\ \ 0010 \\ \hline \text{A9822} & & 1001\ \ 1001\ \ 1000\ \ 0010\ \ 0010 \\ \hline \end{array}$$

Advantages of using Segment Registers

1. Even though addresses associated with the instructions are 16 bits only, allows the memory capacity to be 1MB
2. More than one Code, Data or Stack segment can be used for programs more than 64KB long.
3. Facilitates, use of separate memory areas for a program, its data and the stack.
4. Permit a program and/or its data to be put into different areas of memory each time the program is executed.

Flags



6 - status flags
3 - control flags

Flags

- Status or Conditional flags:
  - These are set according to the results of the arithmetic or logic operations.
  - Need not be altered by the user.

- Control flags:
  - Used to control some operations of the MPU.
  - These flags are to be set by the user, in order to achieve some specific purposes.

Status or Conditional or Condition Code Flags

- **CF (carry) Contains carry from leftmost bit following arithmetic, also contains last bit from a shift or rotate operation.**
- **PF (parity) Indicates the number of 1 bits that result from an operation.(1=even)**
- **AF (auxiliary carry) Contains carry out of bit 3 into bit 4 for specialized arithmetic (BCD).**
- **ZF (zero) Indicates when the result of arithmetic or a comparison is zero. (1=yes)**
- **SF (sign) Contains the resulting sign of an arithmetic operation (1=negative)**
- **OF (overflow) Indicates overflow of the leftmost bit during arithmetic.**

Control flags:

- **DF (direction) Indicates left or right for moving or comparing string data.**

- **IF (interrupt) Indicates whether external interrupts are being processed or ignored.**

- **TF (trap) Permits operation of the processor in single step mode.**

Example

- **Assume that the previous instruction performed the following addition,**

```
  0010  0011  0100  0101
  0011  0010  0001  1001
  ----------------------
  0101  0101  0101  1110
  ----------------------
```

SF= 0   ZF= 0   AF= 0

PF= 0   CF= 0   OF= 0

```
  0101  0100  0011  1001
  0100  0101  0110  1010
  ----------------------
  1001  1001  0101  0011
  ----------------------
```

SF= 1   ZF= 0   AF= 1

PF= 1   CF= 0   OF= 0