Chapter Two

Basic architecture of the 8088/8086

Microprocessors

Outline

- *Feature of 8086/8088 microprocessor*
- *Internal architecture of the 8086/8088 microprocessors*
- *Memory address space and data organization*
- *Data types*
- *Segment registers and memory segmentation*
- *Pointer and index register*
- *Status and flag register*
- *The Stack*

Major Features of 8086 processor

1. The 8086 is a **16-bit microprocessor**. The term "16-bit" means that its arithmetic logic unit, **internal registers** and most of **its instructions** are designed to work with 16-bit binary words.
2. The 8086 has a **16-bit data bus**, so it can read data from or write data to memory and ports either 16 bits or 8 bits at a time. The 8088 has an 8-bit data bus.
3. The 8086 has a **20-bit address bus**, so it can directly access 2 power 20 or 10,48,576 (1Mb) memory locations. The 8088 also has a 20-bit address bus.
4. The Features of 8086 Microprocessor can generate **16-bit I/O address**, hence it can access 2 power 16 (65536)I/O ports.
5. The 8086 provides **fourtee**n 16-bit registers.
6. The 8086 has **multiplexed address** and **data bus** which reduces the number of pins needed, but does slow down the transfer of data (drawback).

Cntd…

7. The 8086 requires **one phase clock** with a 33% duty cycle to **provide optimized internal timing**.

8. The Features of 8086 Microprocessor is possible to perform **bit**, byte, **word** and block operations in 8086.

9. The Intel 8086 is designed to **operate in two modes**, namely the **minimum mode** and the **maximum mode.** When only **one 8086 CPU** is to be used in a microcomputer system, the 8086 is used in the **minimum mode of operation**.

10. The Intel 8086 **supports multiprogramming**. In multiprogramming, the code for two or more processes is in memory at the same time and is executed in a **time-multiplexed fashion**.

11. An interesting feature of the 8086 is that it fetches up to **six instruction bytes** (4 instruction bytes for 8088) from memory and **queue** stores them in order to speed up instruction execution.

12.The Features of 8086 **Microprocessor provides powerful instruction set** with the following **addressing modes:** Register, immediate, direct, indirect through an **index** or base.

Summary of feature of 8086 Architecture MP

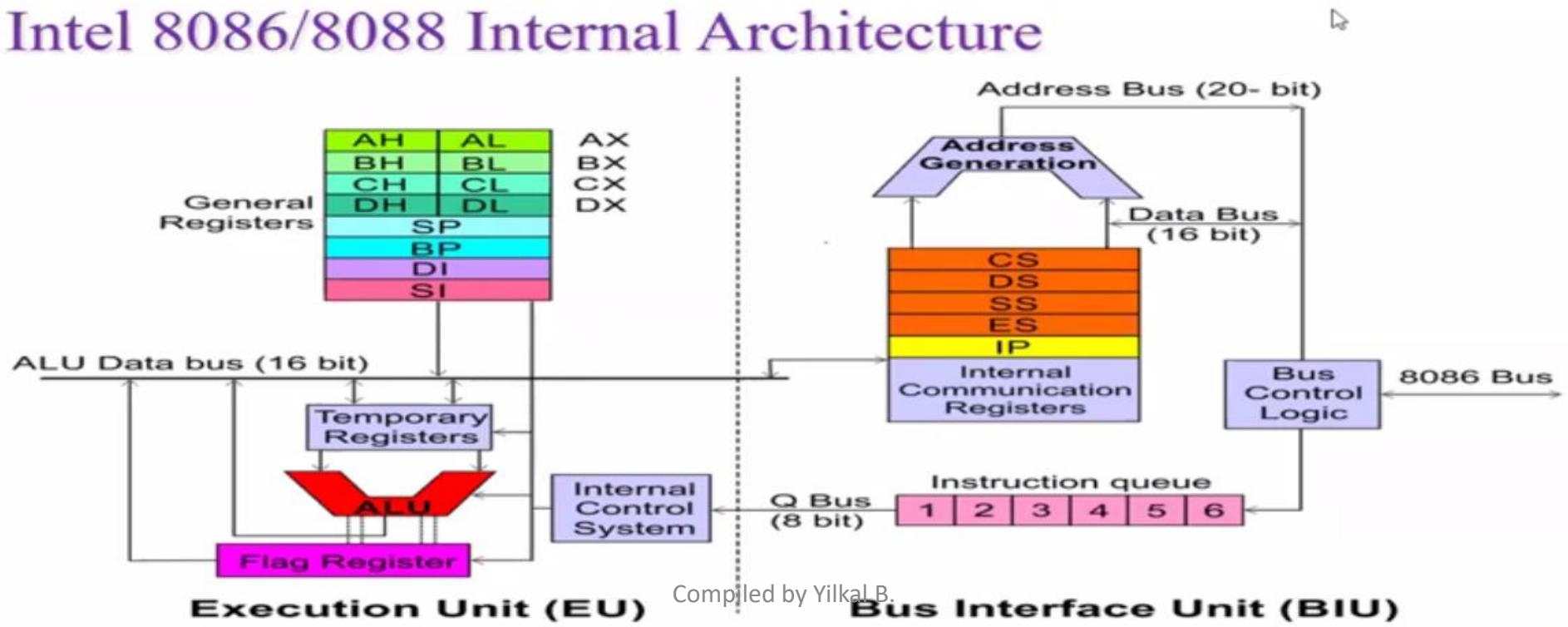
8086 Features

- 16-bit Arithmetic Logic Unit
- 16-bit data bus
- 20-bit address bus – 1,048,576 = 1 meg
- 16 I/O lines so it can access 64K I/O ports
- 16 bit flag
- It has 14 -16 bit registers
- Clock frequency range is 5-10 MHZ
- Designed by Intel
- Rich set of instructions
- 40 Pin DIP, Operates in two modes

Cntd…

- the 8086 internal architecture. It is internally divided **into two separate functional units**.
- These are **A. The Bus Interface Unit (BIU) and**

  B. The Execution Unit (EU)
- These two functional units can **work simultaneously** to increase system speed and hence the throughput.
- **Throughput** is a measure of number of instructions executed per unit time.

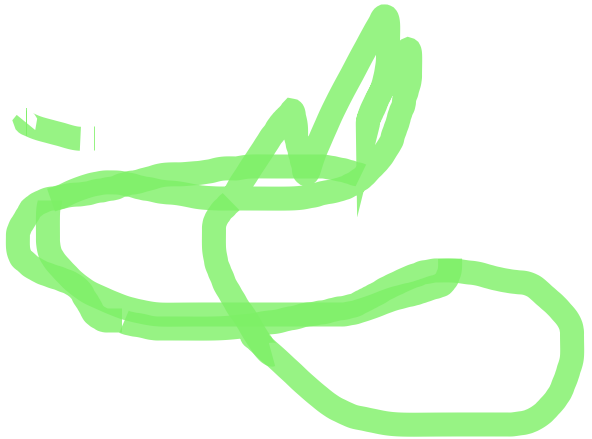


Cntd…

A. Bus Interface Unit (BIU): *This unit connects the 8086's interface to the outside(I/O).*

- *It provides* ***16-bit bi-directional data bus*** *and* ***20-bit address bus****.*
- ***The bus interface unit is responsible for performing all external bus operations***
- ***Function of BIU***

1. *It sends address of the memory or I/O*
2. *It* ***fetches*** *instruction from the memory*
3. *It* ***reads*** *data from port/memory*
4. *It* ***writes*** *data from port/memory*
5. *It supports instruction queuing*
6. *It provides the address relocation facilities*

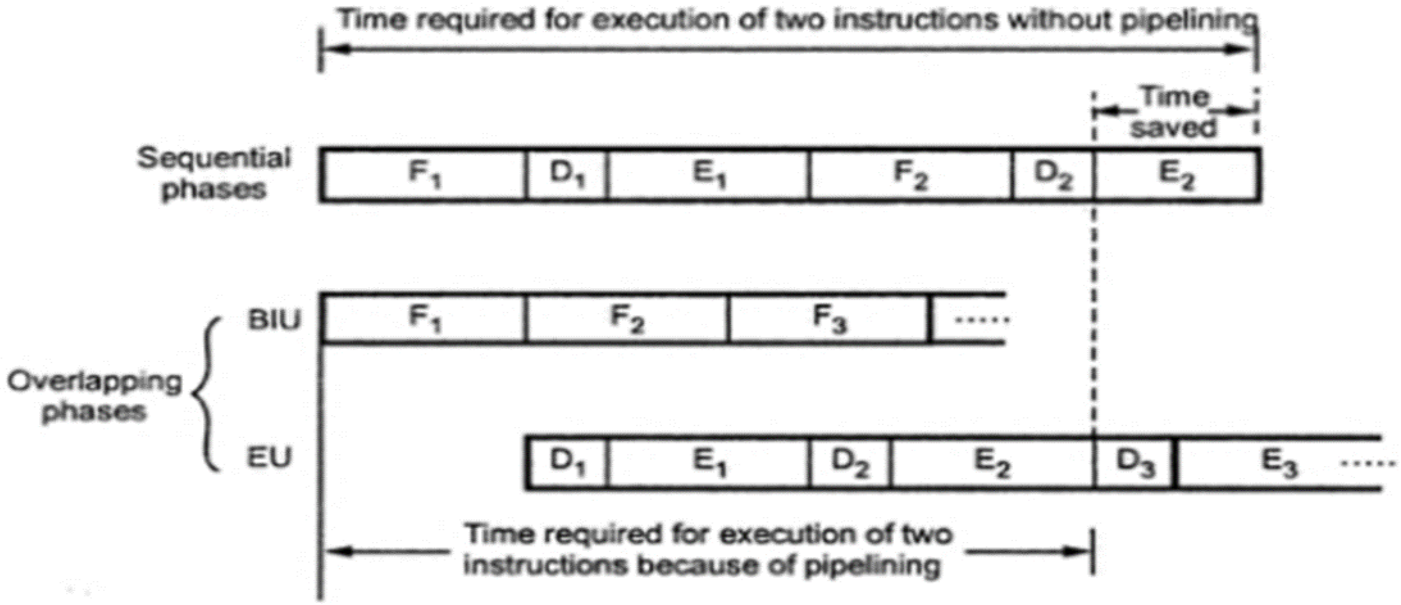


- *The BIU has to interact with memory and I/O devices in fetching the instructions and data required by the EU .*
- *To implement theses functions the BIU* ***contains*** *the* ***instruction queue****, segment registers, instruction pointers,* ***address summer*** *and* ***bus control logic.***

Cntd…

I. Instruction queue

- *To speedup program execution, BIU fetches* ***six instruction byte*** *ahead of time from the memory and kept in the group registers called* ***Queue.***
- *With help of the* ***queue,*** *it is possible to fetch the* ***next instruction*** *when current instruction is in execution.*



Pipelining

Cntd…

II. Segment Registers:

- ***A register*** *is one of a small set of* ***data holding places*** *that are part of the computer processor.*
- *A register may* ***hold an instruction****, a storage address, or any kind of data (such as a bit sequence or individual characters).*
- *Some instructions specify registers as part of the instruction*
- *A* ***register*** *that points to the base of the current segment being addressed is called* ***Segment register.***
- *The physical address of the 8086 Internal Architecture is* ***20-bits wide to access*** *1 Mbyte memory locations.*
- *However, its* ***registers*** *and* ***memory locations*** *which contain* ***logical addresses*** *are just 16-bits wide. Hence 8086 uses* ***memory segmentation.***

Cntd…

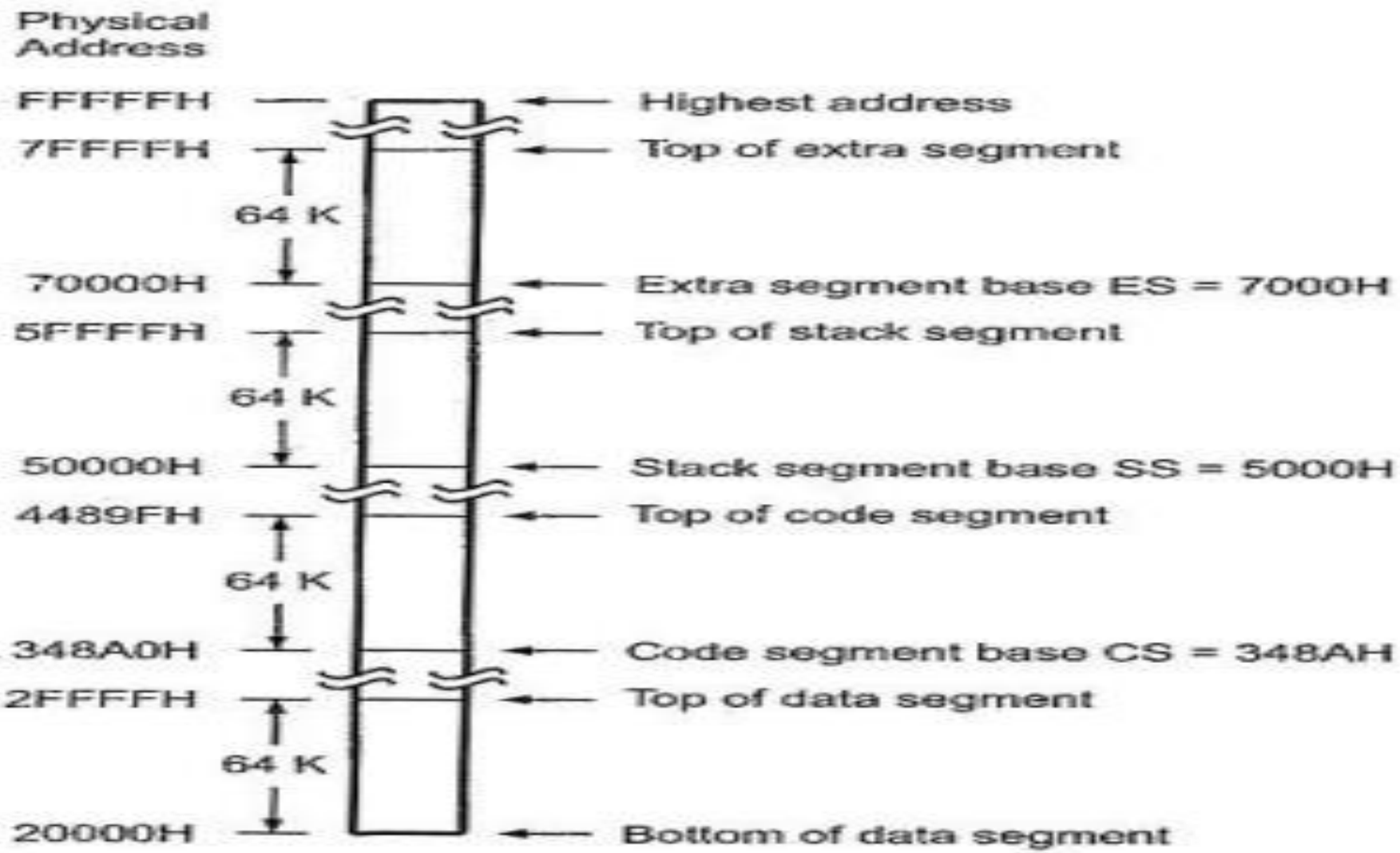
- *It treats the **1 Mbyte of memory as divided into segments**, with a maximum size of a segment as **64 Kbytes**.*
- *Thus any location within the segment can be accessed **using 16 bits.***
- *The 8086 Internal Architecture allows only **four active segments at a time.***
- *For the selection of the four active segments the 16-bit segment registers are provided within the BIU of the 8086.*

Cntd…

- ***These four registers are :***
  1. *Code segment (CS) register,*
  2. *the data segment (DS) register,*
  3. *the stack segment (SS) register, and*
  4. *the extra segment (ES) register.*

- *These are used to hold the upper 16-bits of the **starting addresses** of the **four memory segments,** on which 8086 works at a particular time.*
- *For example, the value in CS identifies the starting address of 64 K-byte segment known as **code segment**.*
- *By "**starting address**", we mean the lowest addressed byte in the **active code segment**.*
- ***The starting address is also known as base address or segment base.***

Cntd…


Physical Address
FFFFFH
Highest address
7FFFFH
Top of extra segment
64 K
70000H
Extra segment base ES = 7000H
5FFFFH
Top of stack segment
64 K
50000H
Stack segment base SS = 5000H
4489FH
Top of code segment
64 K
348A0H
Code segment base CS = 348AH
2FFFFH
Top of data segment
64 K
20000H
Bottom of data segment

Cntd…

- **Functions of Segment Registers**

  - The **CS register** holds the upper 16-bits of the **starting address** of the segment from which the BIU **is currently fetching the instruction code byte**.
  - The **SS register** is used for the upper 16-bits of the starting address for the program stack (**all stack related instructions will operate on stack**)
  - **ES register and DS** register are used to hold the upper 16-bits of the starting address of the **two memory segments which are used for data.**

- **Rules for Memory Segmentation:**

  1. The four segments can overlap for small programs.
  2. In a minimum system all four segments can start at the address **00000H.**
  3. The segment can **begin/start at any memory address** which is divisible by 16.

Cntd…

- ***Memory segmentation is*** *an OS memory management technique of division of a computer's primary memory into segments or sections.*
- ***Advantages of Memory Segmentation***

1. It ***allows the memory addressing capacity*** *to be 1 Mbyte even though the address associated with individual instruction is only 16-bit.*

2. It allows instruction ***code****, data, stack, and* ***portion of program to be more than 64 KB long*** *by using more than one code, data, stack segment, and extra.*

3. It facilitates use of separate memory areas for program, data and stack.

4. It permits a program or its data to be put in different areas of memory, each time the program is executed.

i.e. program can be relocated which is very useful in multiprogramming.

Cntd…

III. Instruction Pointer(IP):

- It is a 16-bit register used to **hold the address of the next instruction to be executed**.
- The **instruction pointer register** holds the 16-bit address of the next Code byte within the **code segment.**
- The value contained in the IP is referred to as **an offset.**
- This value must be offset from (added to) the **segment base address** in CS to produce the required 20-bit physical address

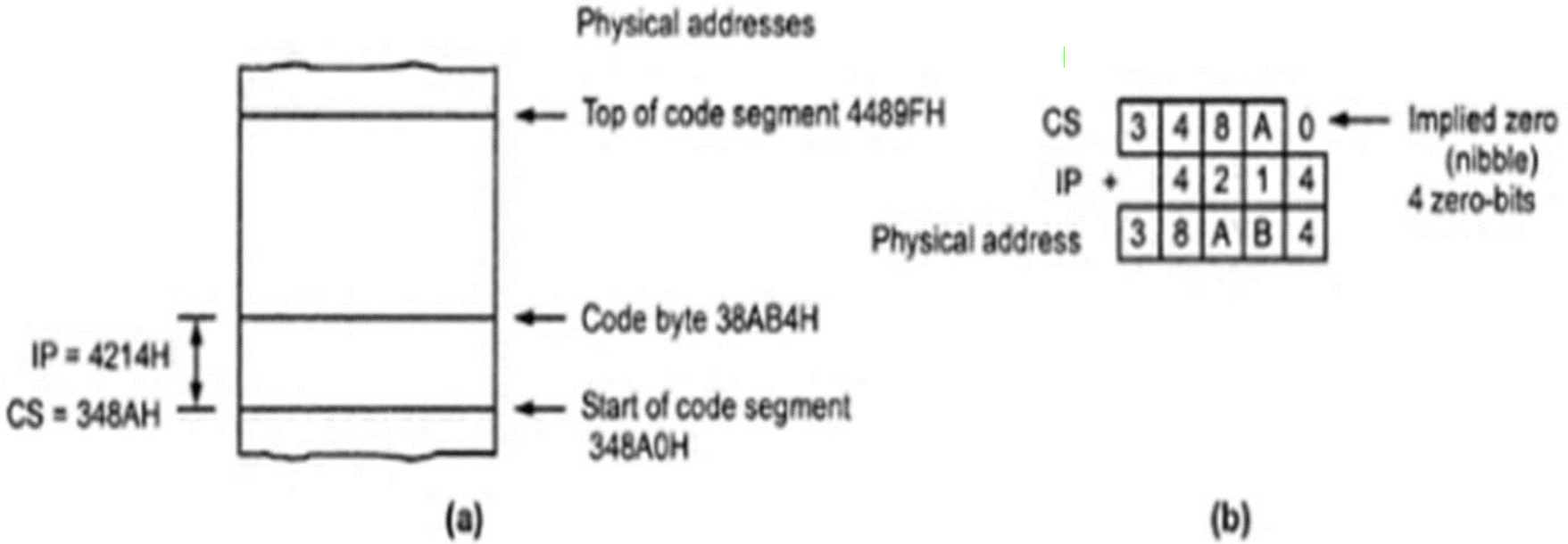
Generation of 20-bit Address

- ***20-bit physical address*** *is needed to access a specific memory location from any segment.*
- *To generate it, 8086 is using* ***contents of segment registers*** *and the* ***offset registers*** *associated with it.*

Example for address generation with any one segment register and corresponding register

- ***CS register*** *hold the 16-bit base address of the code segment*
- ***IP (Instruction Pointer)*** *is holds the 16-bit address of the next code byte within the code segment.*
- *The value contain in the IP is called as* ***an offset***
- *The content of the CS register is* ***shifted by 4 position*** *to the left by inserting 4 zero-bits. Now the CS content is* ***20-bit value***
- *This shifted value is offset(added to) the IP register content. It will generate the* ***20-bit physical address***

Cntd…



We have seen that how 20-bit physical address is generated within the code segment. In the similar way the 20-bit physical address is generated in the other segments. However, it is important to note that each segment requires particular segment register and offset register to generate 20-bit physical address.

Cntd…

B. Execution Unit (EU): *The execution unit is* ***gives the instruction to the BIU*** *from where to fetch the instructions or* ***data,*** *decodes instructions and* ***executes instructions.***

- ***It contains:***

  I. Control Circuitry

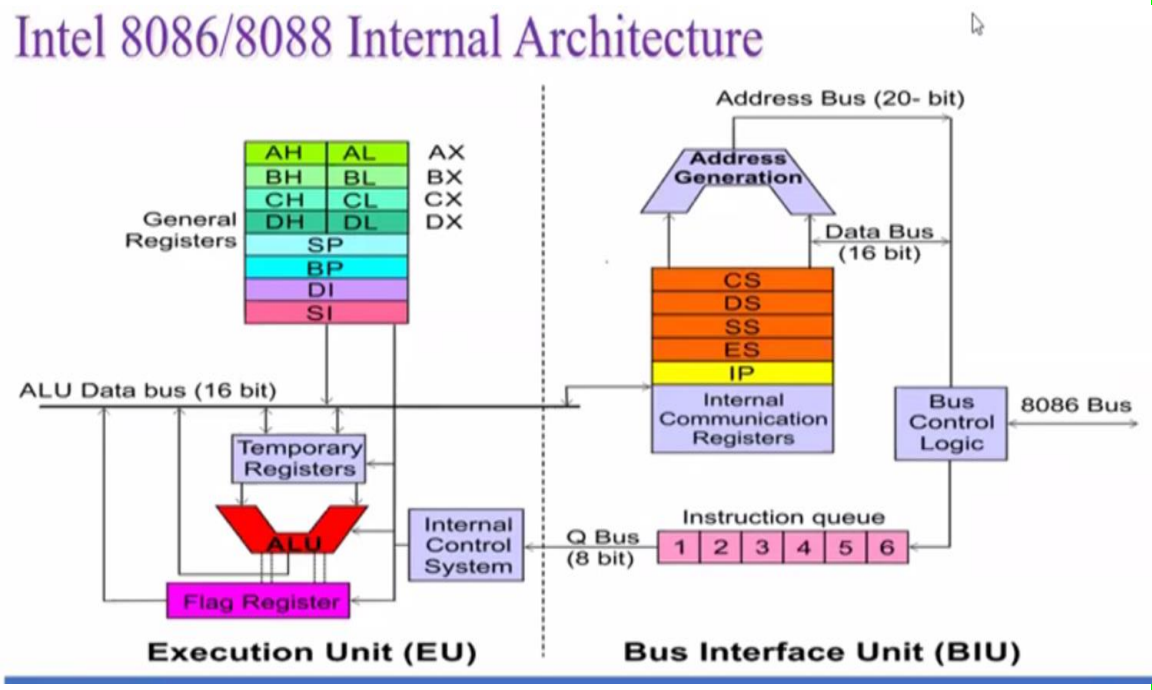
  II. Instruction decoder

  III. Arithmetic Logic Unit (ALU)

  IV. Flag register

  V. General purpose registers

  VI. Pointers and index registers



- *EU is responsible for* ***executing the instructions*** *of the programs and to carry out the required processing.*

Cntd…

I. Control Circuitry: It directs the internal operation in the EU

II. Instruction Decoder: It translate the instructions fetch from the memory into a series of actions which the EU performs.

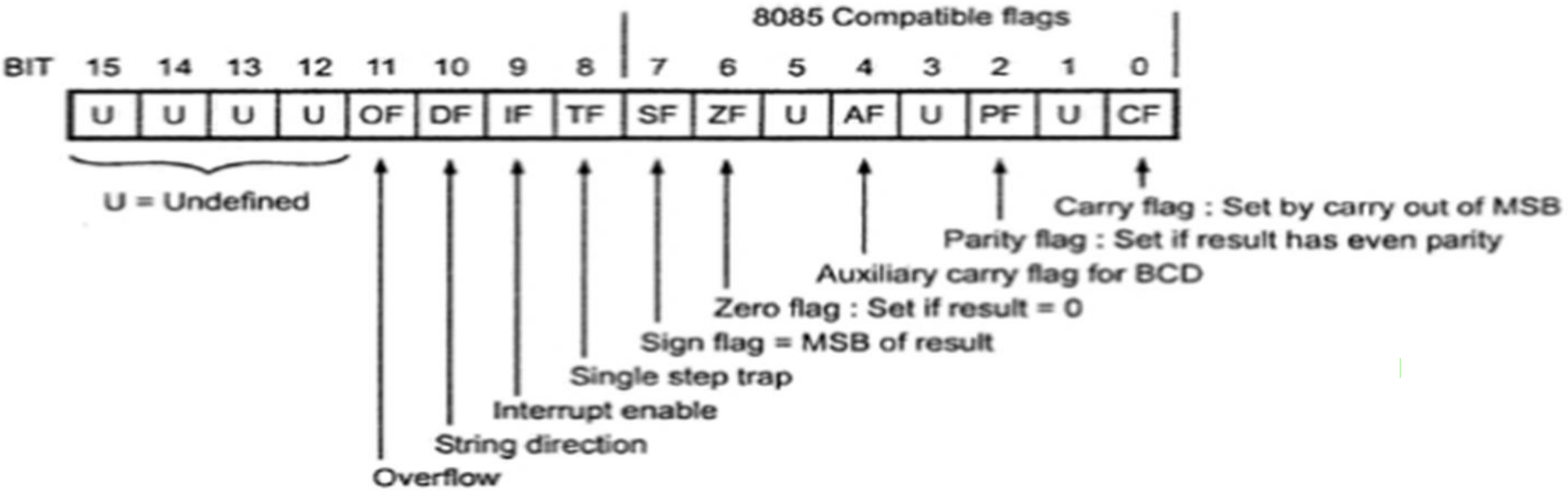
III. Arithmetic logical unit(ALU):

- It's a 16-bit wide
- It can add, subtract, AND, OR, XOR, increments, decrements, complements and shift binary numbers

Cntd…

IV. Flag Register:

- It helps to indicate some conditions produced at arithmetic or logic operation execution
- It also contains flag bit to control certain operation on execution unit.
- Flag register contains **9 actives flags** only shown below



8086 flag register bit pattern

Cntd…

1. Carry Flag(CF):

- *This flag is set to 1 in addition and subtraction process*
- *It will **set to 1**, If there is **a carry out of** MSB(Most Significant Bit) in addition process*
- *It will **set to 1,** If there is a **borrow is needed out** of the MSB in subtraction process*

2. Parity Flag(PF):

- *It is **set to 1,** if the result of byte or word operations contain **an even** number of 1's Otherwise this flag bit is 0*

3. Auxiliary Flag(AF):

- *This flag is set if there is **an overflow out of bit 3** (i.e., carry from lower nipple to higher nipple (D3 bit to D4 bit))*
- *This flag is used for BCD operation not for programmer*

Cntd…

4. Zero Flag (ZF):

- This flag sets if the result of operation in **ALU is zero**
- Flag is **reset** if the result is **nonzero**
- Its also set if certain register content become zero on increment or decrement operation of the register

5. Sign Flag (SF):

- This flag is set, if the value in the register is negative after the execution of arithmetic or logical operation
- Flag bit is zero then it is positive

Cntd…

6. Overflow Flag (OF):

- *This flag is set if the result is* ***out of range***
- *For addition, this flag is set when there is a* ***carry into the MSB and no carry*** *out of MSB or vice versa*
- *For subtraction, this flag is set when the MSB need a* ***borrow and there is no borrow*** *from the MSB or vise versa*

Give the contents of the flag register after execution of following addition.

$$\begin{array}{rcccc} & 0110 & 0101 & 1101 & 0001 \\ + & 0010 & 0011 & 0101 & 1001 \\ \hline & 1000 & 1001 & 0010 & 1010 \end{array}$$

Solution : SF = 1, ZF = 0, PF = 1, CF = 0, AF = 0 , OF = 1

Give the contents of the flag register after execution of following subtraction.

$$\begin{array}{rcccc} & 0110 & 0111 & 0010 & 1001 \\ - & 0011 & 0101 & 0100 & 1010 \\ \hline & 0011 & 0001 & 1101 & 1111 \end{array}$$

Solution : SF = 0, ZF = 0, PF = 1, CF = 0, AF = 1, OF = 0

Cntd…

The remaining three flags are used to control the operation of the processor

7. Trap Flag (TP):

- *If this flag is set, it will execute after each instruction is executed*
- *It can display various registers and memory variable contents on display after execution of the each instruction.*
- *So, the programmer can trace and correct the errors in the program*

8. Interrupt Flag:

- *This flag is used allow/prohibit the interrupt in program*
- *If* ***the flag set,*** *interrupt is allowed in program*

9. Direction Flag:

- *It is used with string instruction only*
- *If DF=0, the string is processed from the lowest address to the high address*
- *If DF=1, the string is processed from the high address to lowest address*

Cntd…

V. Multi Purpose Registers

- General registers are used for **temporary storage** and **manipulation of data and instructions**
- It has four 16-bit general purpose registers (Labeled AX, BX, CX, DX)
- The letter **X** is used to specify the complete **16-bit register**
- Each registers split into two 8-bit registers (L (lower byte)and H(higher byte))
- This type of registers used for **holding data,** variables and intermediate result temporarily
- **Accumulator register (AX)** is used as 16-bit accumulator (also AL is used as 8-bit register)
  - **Accumulator can be used for I/O operations and string manipulation**
- **Base register**(BX): is used as offset storage for generating physical address for some addressing modes
- It contains a data pointer used for based, based indexed or register indirect addressir

Cntd…

- ***Count register(CX)*** *consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX .*
- *Count register can be used as a counter in string manipulation and shift/rotate instructions1 word = 2bytes & 1 byte =8 bits*
- ***Data register(DX):*** *consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX .*
- *Data register can be used as a port number in I/O operations.*
- *It is also used with AX register along with DX for multiply and divide operations involving large values.*
- *In integer 32-bit multiply and divide instruction the DX register contains high order word and* ***AX register contains Lower order word of results****.*

Pointers and Index registers

- All segment registers are 16-bit wide but its needed to generate 20-bit address(physical address) on address bus
- To get 20-bit physical address one or more pointer or index registers are associated with each segment register
- Pointer Register IP is associated with **code segment**
- Pointer Register BP is associated with **data segment**
- Pointer Register SP is associated with **stack segment**
- These pointer registers holds the **offset value**
- The index registers DI and SI are used as a **general purpose registers** as well as for offset storage in the addressing modes.

Cntd…

- **Base Pointer (BP)**

- is a 16-bit register **pointing to data in stack segment.**
- BP register is usually used for based, based indexed or register indirect addressing.

- **Source Index (SI)**

- is a 16-bit register.
- SI is used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.

- **Destination Index (DI)**

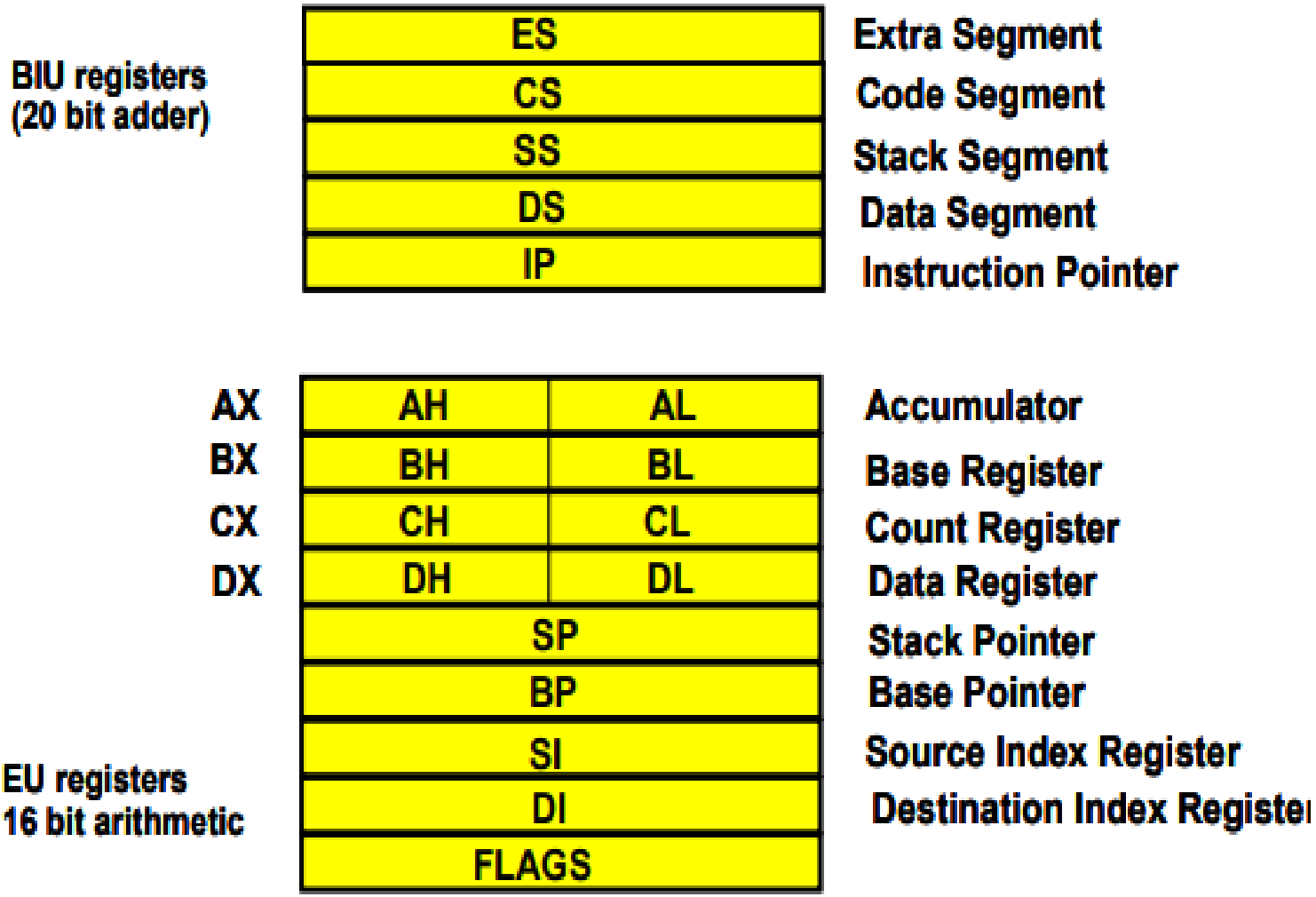
- is a 16-bit register.
- DI is used for indexed, based indexed and register indirect addressing
- The ES register points to the extra segment in which data is stored. String instructions always use ES and DI to determine the 20-bit physical address for the destination

Default and Alternate Register Assignments:

- *Table below shows that **some memory references and their default and alternate segment definitions.** For example, **instruction codes** can only be stored in the code segment with IP used as an offset. Similarly,*
- *for **stack operations** only SS and SP or BP registers can be used to give segment and offset addresses respectively.*
- *On the other hand, for accessing general data, string source; data pointed by BX and BP registers; it is possible to use alternate segments by using segment override prefix.*

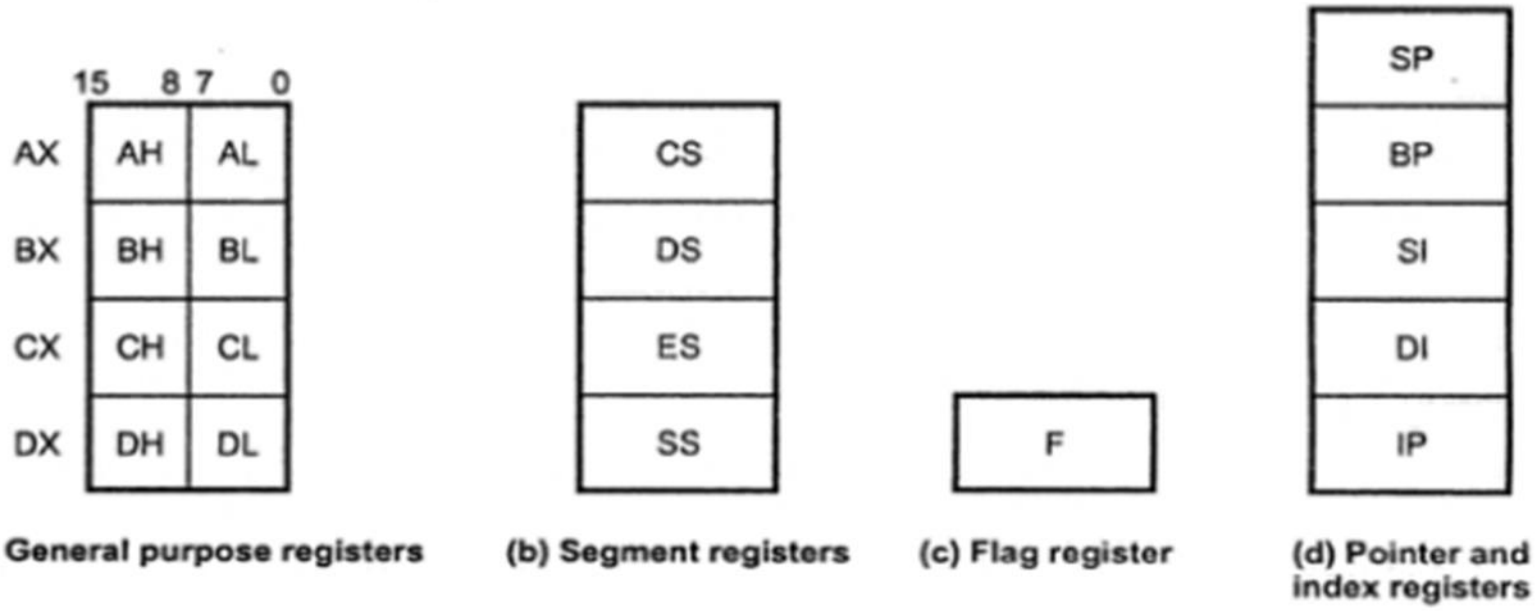
Type of Memory Reference	Default Segment	Alternate Segment	Offset (Logical Address)
Instruction fetch	CS	None	IP
Stack operation	SS	None	SP, BP
General data	DS	CS, ES, SS	Effective address
String source	DS	CS, ES, SS	SI
String destination	ES	None	DI
BX used as pointer	DS	CS, ES, SS	Effective Address
BP used as pointer	SS	CS, ES, DS	Effective Address

Cntd…


BIU registers
(20 bit adder)
ES
CS
SS
DS
IP
Extra Segment
Code Segment
Stack Segment
Data Segment
Instruction Pointer
AX
BX
CX
DX
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
FLAGS
Accumulator
Base Register
Count Register
Data Register
Stack Pointer
Base Pointer
Source Index Register
Destination Index Registe
EU registers
16 bit arithmetic

Programmer's Model

- 8086 has a powerful set of registers
- It includes:
  a. General purpose registers
  b. Segment registers
  c. Pointers and index registers
  d. Flag registers

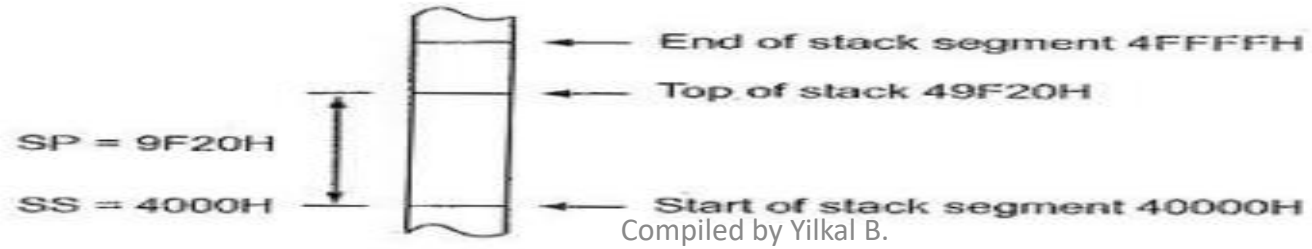


Registers Organization of 8086 (programmer's model)

Stack Pointer (SP):

- SP register contains the **16-bit offset** from the **start of the segment** to the top of stack.
- For stack operation, **physical address is produced** by **adding** the contents of **stack pointer register** to **the segment base address** in SS.
- The value of Data Segment Register (DS) is 4000H, To convert this 16-bit address into 20-bit,
- the BIU appends 0H to the LSBs of the address.
- After appending, the starting address of the Data Segment becomes 40000H. If the contents of SP are **9F20H** and SS are **4000H** then the physical address is calculated as follows.
- SS = 4000H after appending 0H, become SS = 40000H

SS	40000H
+ SP	9F20H
Physical address	49F20H



End of chapter Two

Any Question ???