MICROPROCESSOR ARCHITECTURE

UOP S.E.COMP (SEM-I)

8086 MICROPROCESSOR ARCHITECTURE

Prof.P.C.Patil
Department of Computer Engg
Matoshri College of Engg.Nasik
pcpatil18@gmail.com.

Introduction

Introduction

- Before getting into 8086 lets 1st define microprocessor.
- In simple words, a microprocessor is an electronic device which computes on the given input similar to CPU of a computer.
- It is made by fabricating millions (or billions) of transistors on a single chip.

History

History

- Microprocessor journey started with a 4-bit processor called 4004,
- It was made by Intel corporation in 1971.
- it was 1st single chip processor.
- then the idea was extended to 8-bit processors like 8008, 8080 and then 8085 (all are Intel products).
- 8085 was a very successful one among the 8-bit processors,
- However its application is very limited bcoz of its slower computing speed and other quality factors.

History

- Some years later Intel came up with its 1st 16-bit processors 8086.
- At the same time other manufacturers were also making processors like 68000 (by motorola), Zilog z-80, General instrument PIC16X, MOS Technology 6502, etc...
- In 1979 Intel released a modified version of 8086 as 8088.

History

- Next intel started updating 80x86 series by introducing 80286, 80386, 80486, pentium and then pentium series.
- After 80486, the next processor in series was to be said 80586,
- But Intel named it as pentium bcoz of its copyright problem.
- Further updation in pentium resulted in pentium-I, pentium-II, etc..

Features

Features

- It is a 16-bit µp.
- It has a 20 bit Address bus can access up to 2^{20} (10,48,576) memory locations (1 MB).
- It has 16-Bit Data Bus
- 16-Bit Words are stored in two consecutive memory locations .
- It is possible to perform Bit, Byte, Word and block operations.
- It performs Arithmetic and Logical operations on Bit, Byte, Word and Decimal Numbers

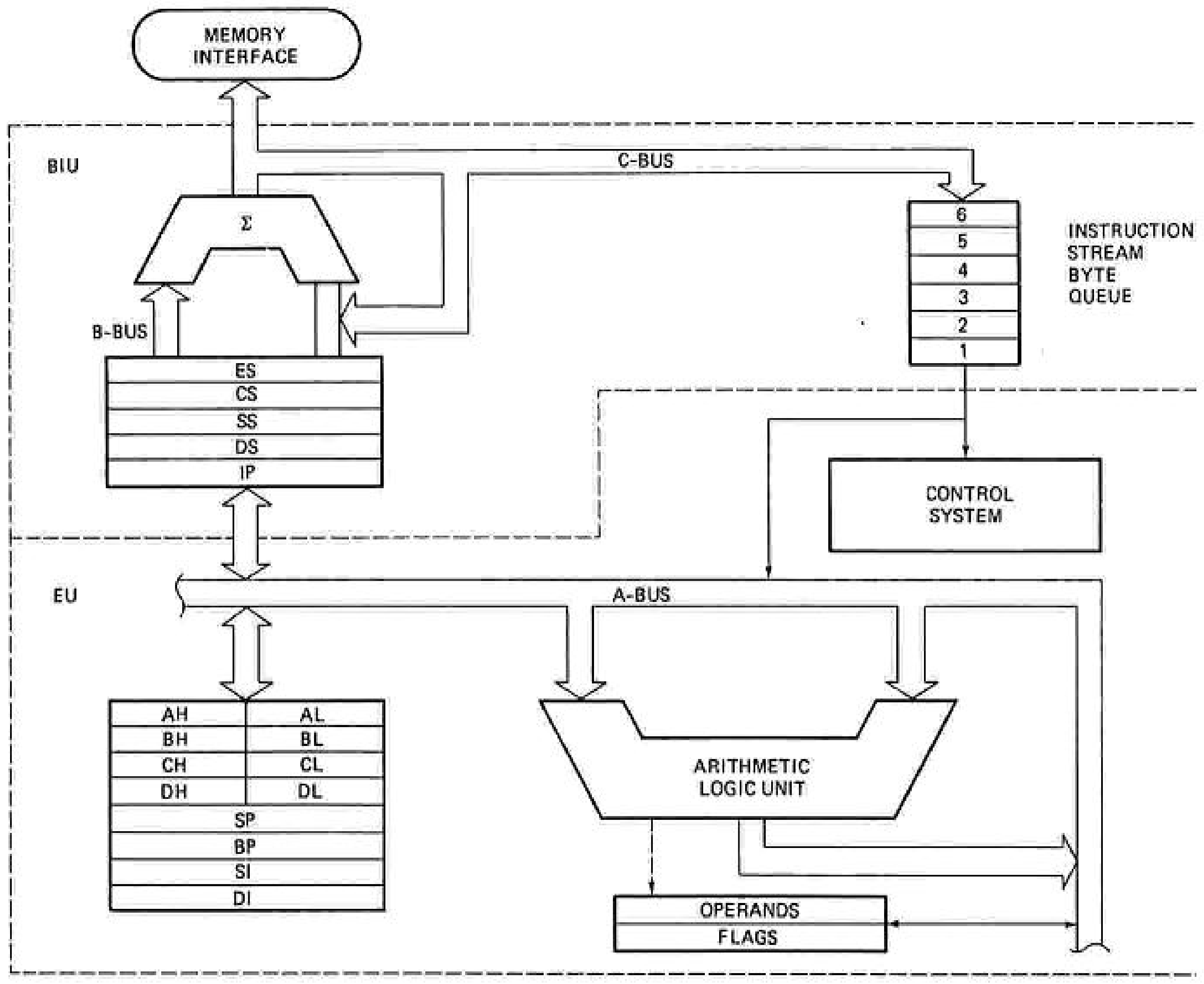
Features

- It can support up to 64K I/O ports.
- It provides Fourteen, 16 -bit registers.
- Word size is 16 bits.
- It has multiplexed address and data bus AD0-AD15 and A16 – A19.
- It requires single phase clock with 33% duty cycle to provide internal timing.

Features

- 8086 is designed to operate in two modes, Minimum and Maximum.
- It can prefetches up to 6 instruction bytes from memory and queues them in order to speed up instruction execution.
- It requires +5V power supply.
- A 40 pin dual in line package.
- Address ranges from 00000H to FFFFFH
- Memory is byte addressable - Every byte has a separate address.

8086 Architecture


MEMORY INTERFACE
BIU
C-BUS
Σ
INSTRUCTION STREAM BYTE QUEUE
6
5
4
3
2
1
B-BUS
ES
CS
SS
DS
IP
CONTROL SYSTEM
EU
A-BUS
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
ARITHMETIC LOGIC UNIT
OPERANDS
FLAGS

- 8086 has two blocks BIU and EU.
- The BIU handles all transactions of data and addresses on the buses for EU.
- The BIU performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands. The instruction bytes are transferred to the instruction queue.
- EU executes instructions from the instruction system byte queue.

Architecture

- Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining. This results in efficient use of the system bus and system performance.

- BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.

- EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.

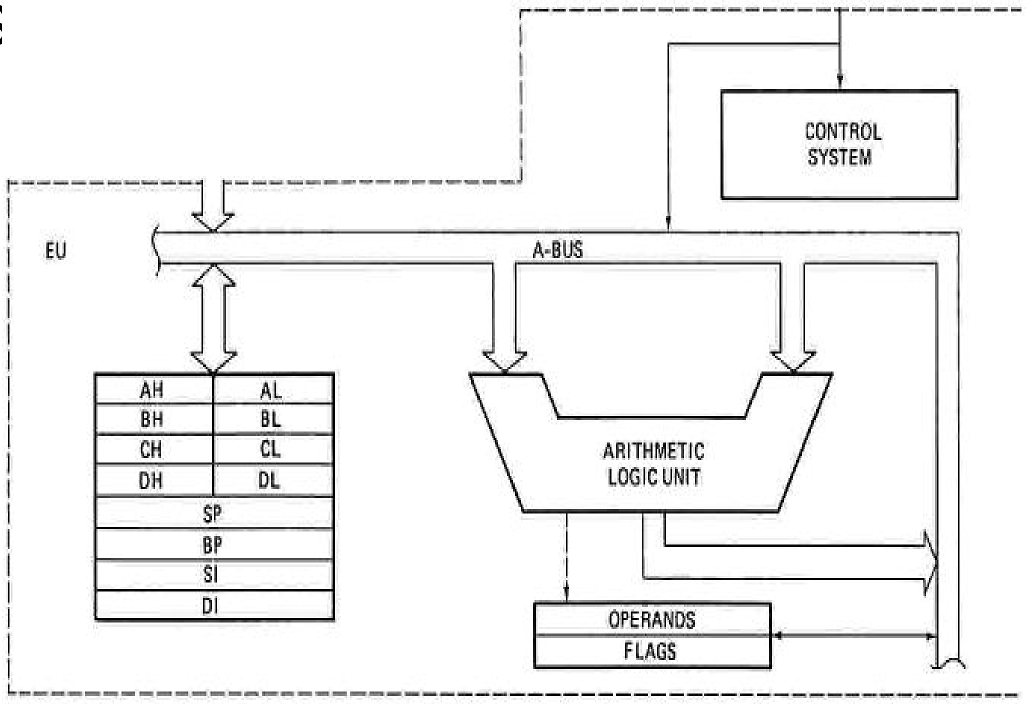
Execution Unit

Execution Unit

- Decodes instructions fetched by the BIU
- Generate control signals,
- Executes instructions

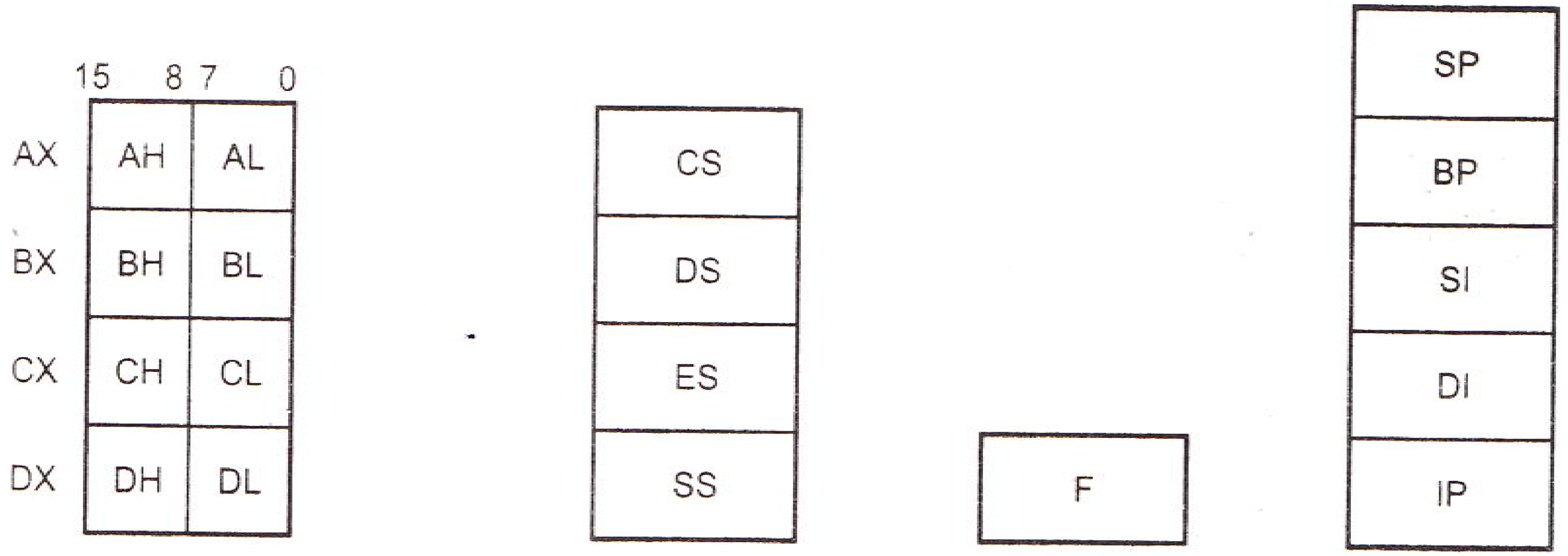
The main parts are:

- Control Circuitry
- Instruction decoder
- ALU

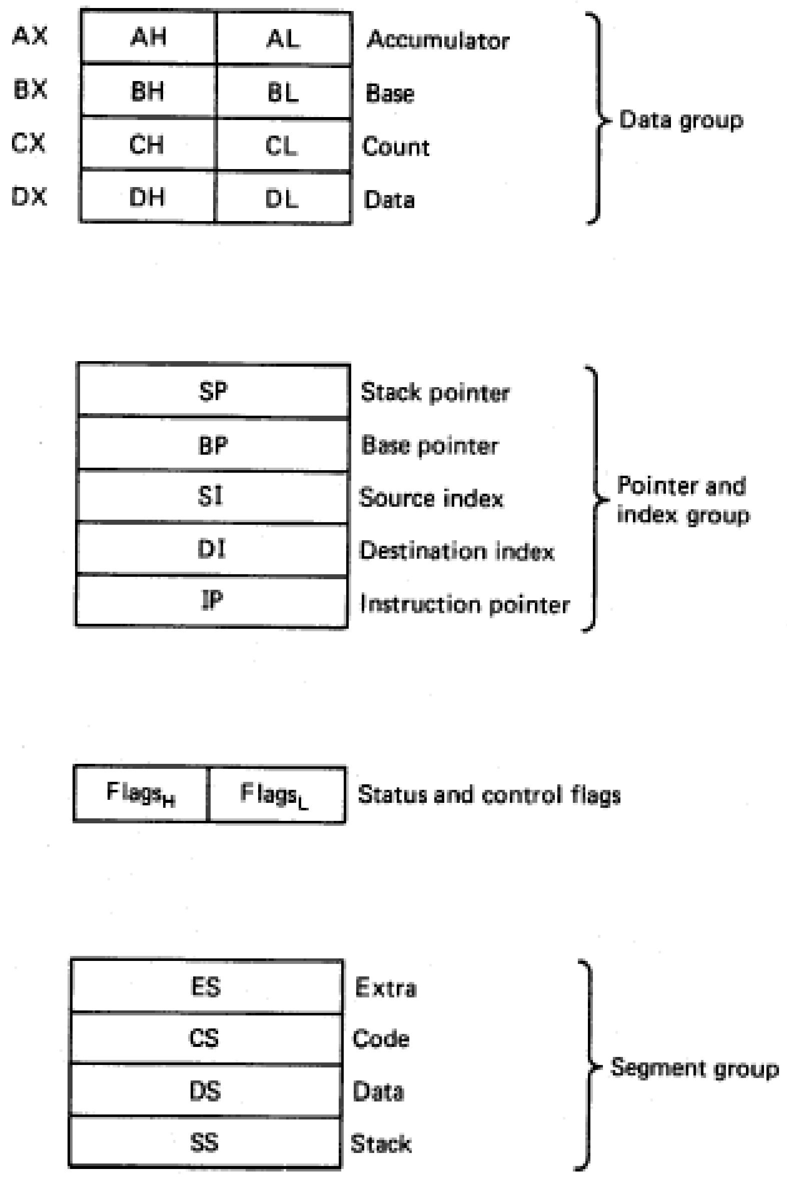


Registers

Registers



(a) General purpose registers (b) Segment registers (c) Flag registers (d) Pointer and index registers



AX, BX, CX and DX are two bytes wide and each byte can be accessed separately

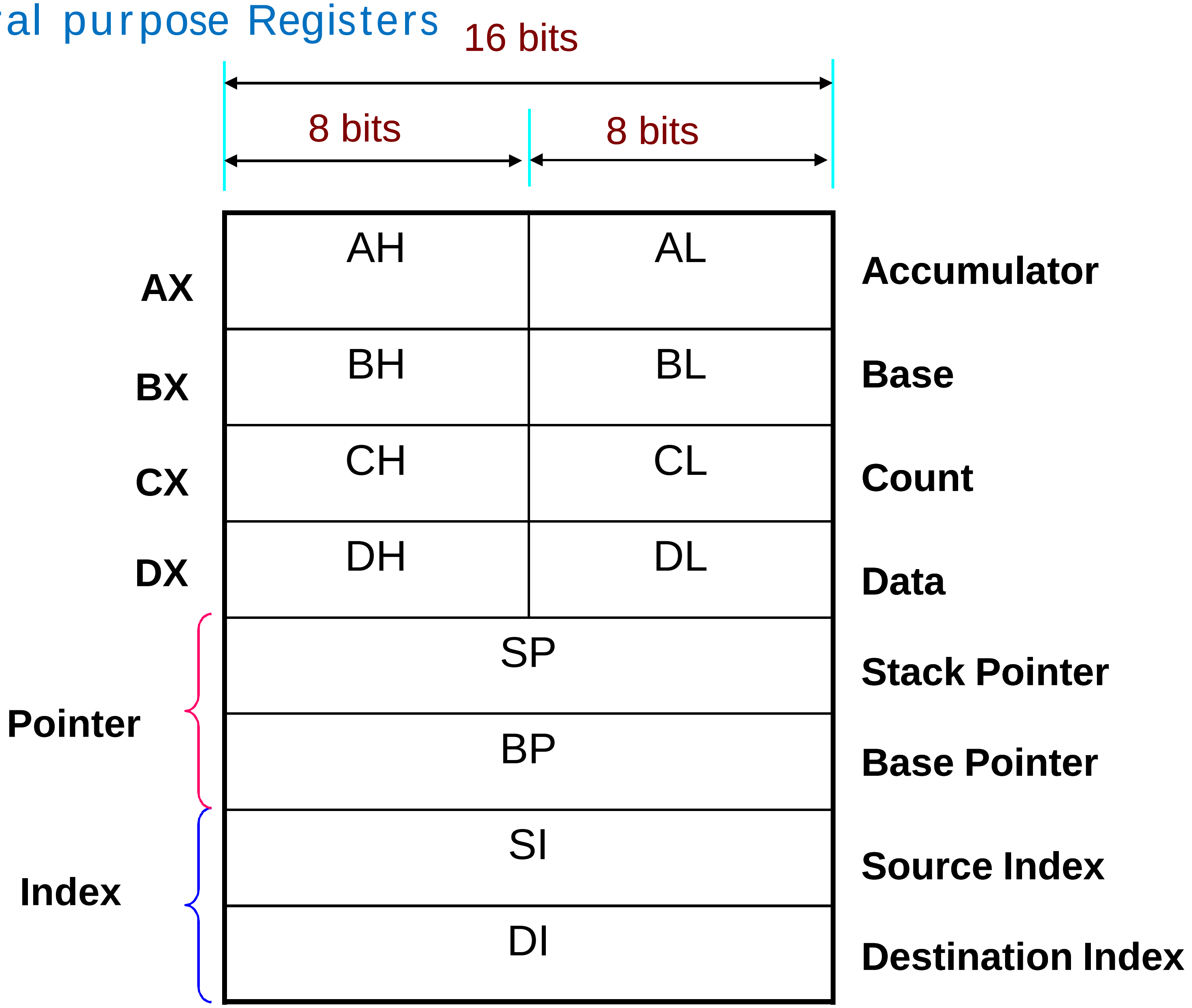
These registers are used as memory *pointers*.

Flags will be discussed later

Segment registers are used as base address for a segment in the 1 M byte of memory

General Purpose Registers

General purpose Registers



Register	Purpose
AX	Word multiply, word divide, word I /O
AL	Byte multiply, byte divide, byte I/O, decimal arithmetic
AH	Byte multiply, byte divide
BX	Store address information
CX	String operation, loops
CL	Variable shift and rotate
DX	Word multiply, word divide, indirect I/O (Used to hold I/O address during I/O instructions. If the result is more than 16-bits, the lower order 16-bits are stored in accumulator and higher order 16-bits are stored in DX register)

General purpose Registers

- Instructions execute faster if the data is in a register
- 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, and DL are the low bytes
- Data Registers are general purpose registers but they also perform special functions

AX

- Accumulator Register
- Preferred register to use in arithmetic, logic and data transfer instructions because it generates the shortest Machine Language Code
- Must be used in multiplication and division operations
- Must also be used in I/O operations

General purpose Registers

- **BX**
  - Base Register
  - Also serves as an address register
- **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

Pointers and Index Registers

Pointers and Index Registers

- Used to Keep offset addresses.
- Used in various forms of memory addressing.
- The index registers (SI & DI) and the BX generally default to the Data segment register (DS).

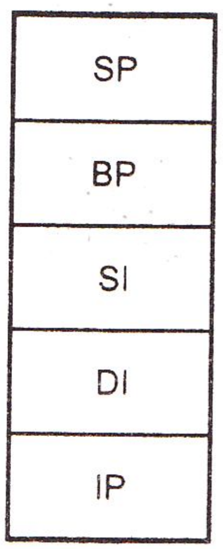
SP: Stack pointer

– 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



(d) Pointer and index registers

Pointers and Index Registers

- SI: Source Index register
  - – is required for some string operations
  - – When string operations are performed, the SI register points to memory locations in the data segment which is addressed by the DS register. Thus, SI is associated with the DS in string operations.

- DI: Destination Index register
  - – is also required for some string operations.
  - – When string operations are performed, the DI register points to memory locations in the data segment which is addressed by the ES register. Thus, 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

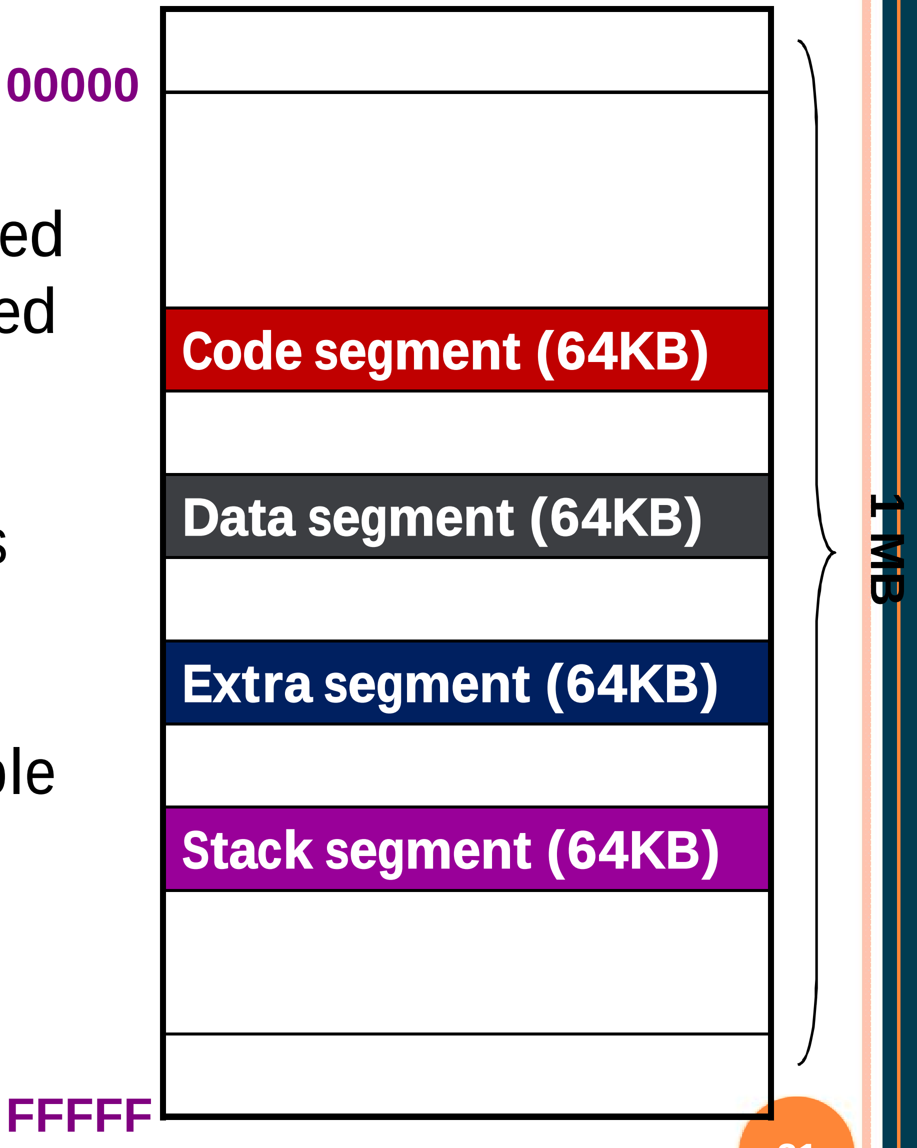
Segment Registers & Segmentation

Segment Register and Segmentation

- Physical address of 8086 is 20 Bit wide (To access 1 Mb memory locations)
- BUT- But the registers and memory locations which contains logical address are 16 Bit
- Hence segmentation is required

Segment Register and Segmentation

- The memory in an 8086/88 based system is organized as segmented memory.
- The CPU 8086 is able to address 1Mbyte of memory.
- The Complete physically available memory may be divided into a number of logical segments.



Segment Register and Segmentation

- The size of each segment is 64 KB
- A segment is an area that begins at any location which is divisible by 16.
- A segment may be located any where in the memory
- Each of these segments can be used for a specific function.

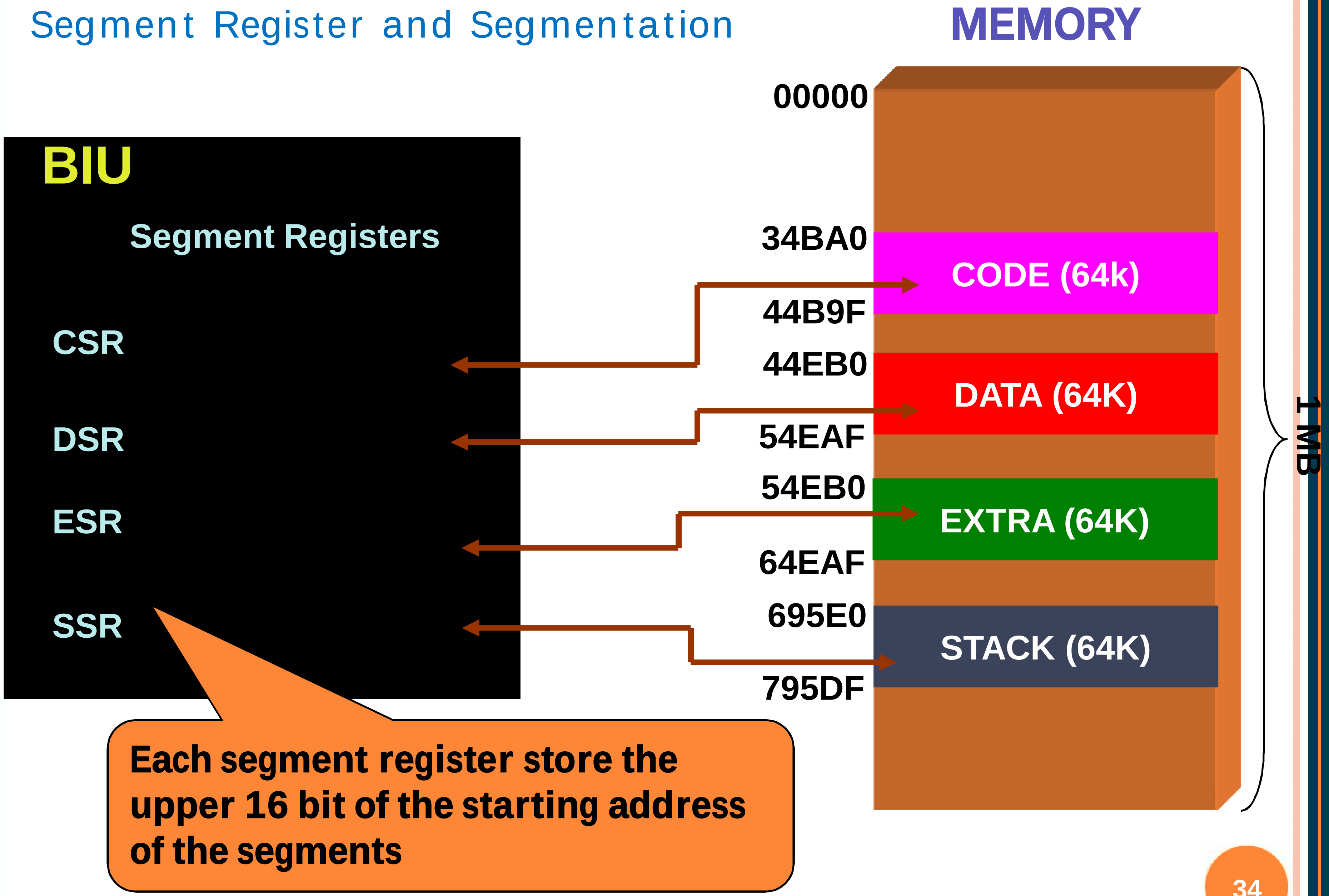
  - Code segment is used for storing the instructions.
  - The stack segment is used as a stack and it is used to store the return addresses.
  - The data and extra segments are used for storing data byte.

*** In the assembly language programming, *more than one data/ code/ stack segments* can be defined. But *only one segment of each type can be accessed* at any time.**

Segment Register and Segmentation

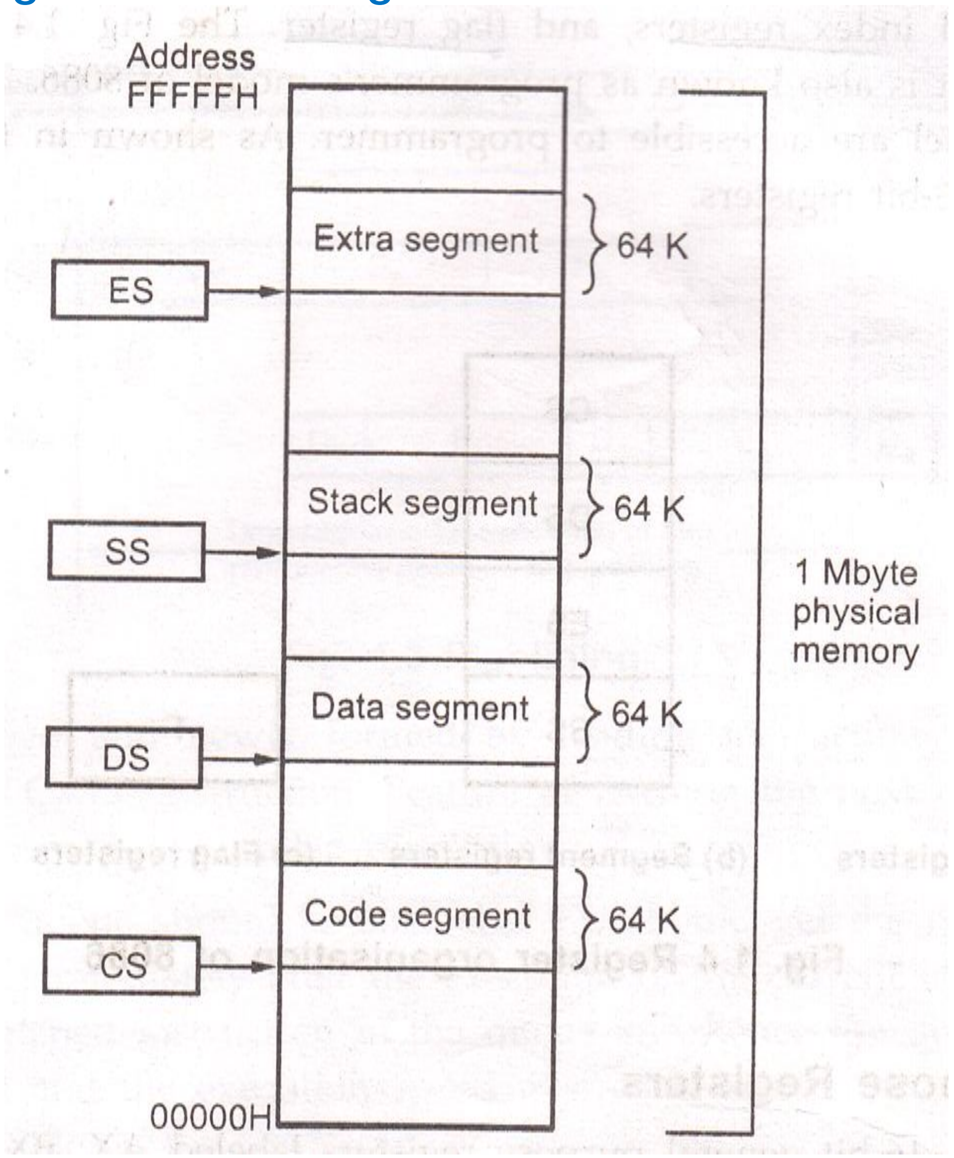
- The 4 segments are Code, Data, Extra and Stack segments.
- A Segment is a 64kbyte block of memory.
- The 16 bit contents of the segment registers in the BIU actually point to the starting location of a particular segment.
- Segments may be overlapped or non-overlapped
- Each of the Segment registers store the upper 16 bit address of the starting address of the corresponding segments.

Segment Register and Segmentation



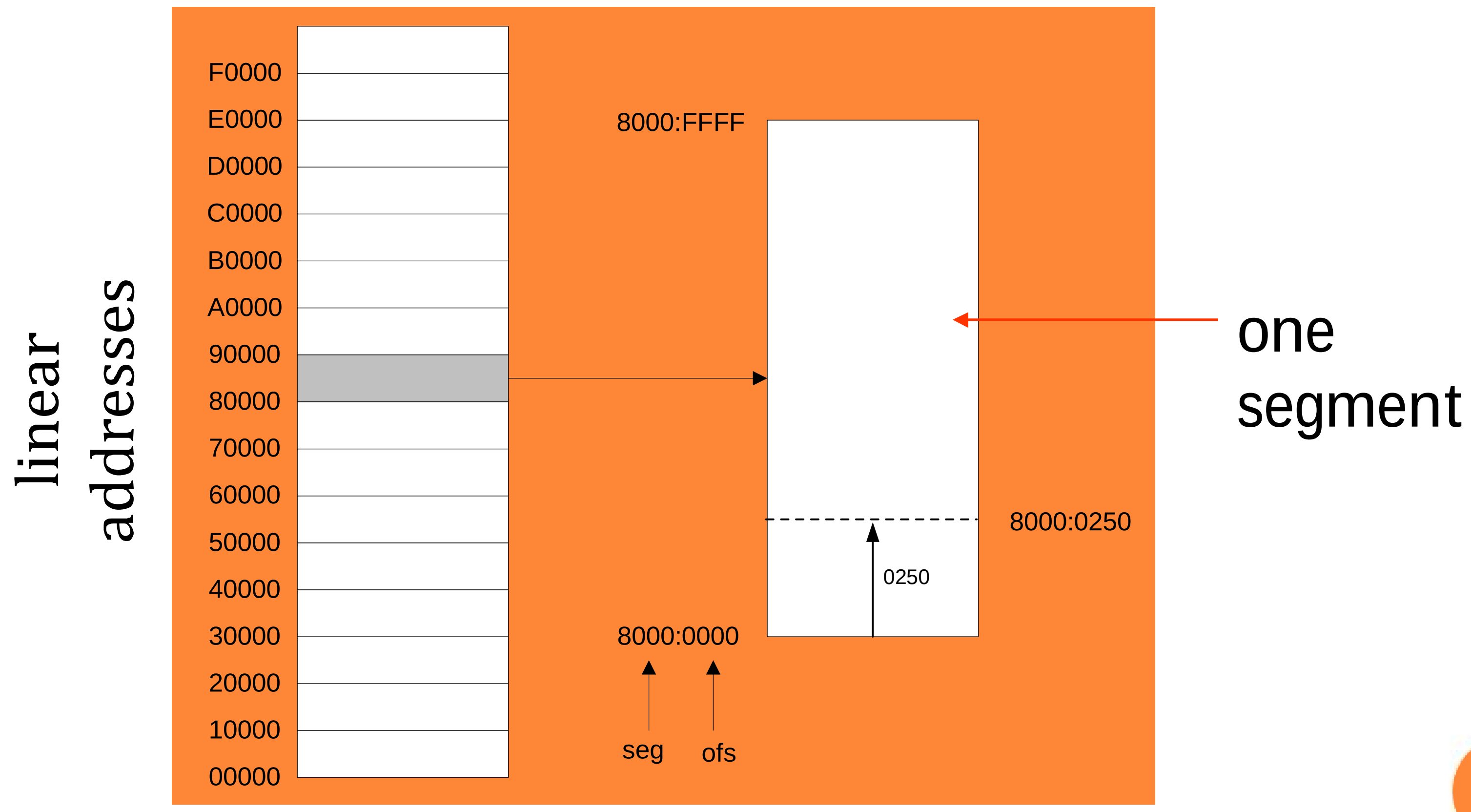
Each segment register store the upper 16 bit of the starting address of the segments

Segment Register and Segmentation



Segmented Memory

Segmented memory addressing: absolute (linear) address is a combination of a 16-bit segment value added to a 16-bit offset



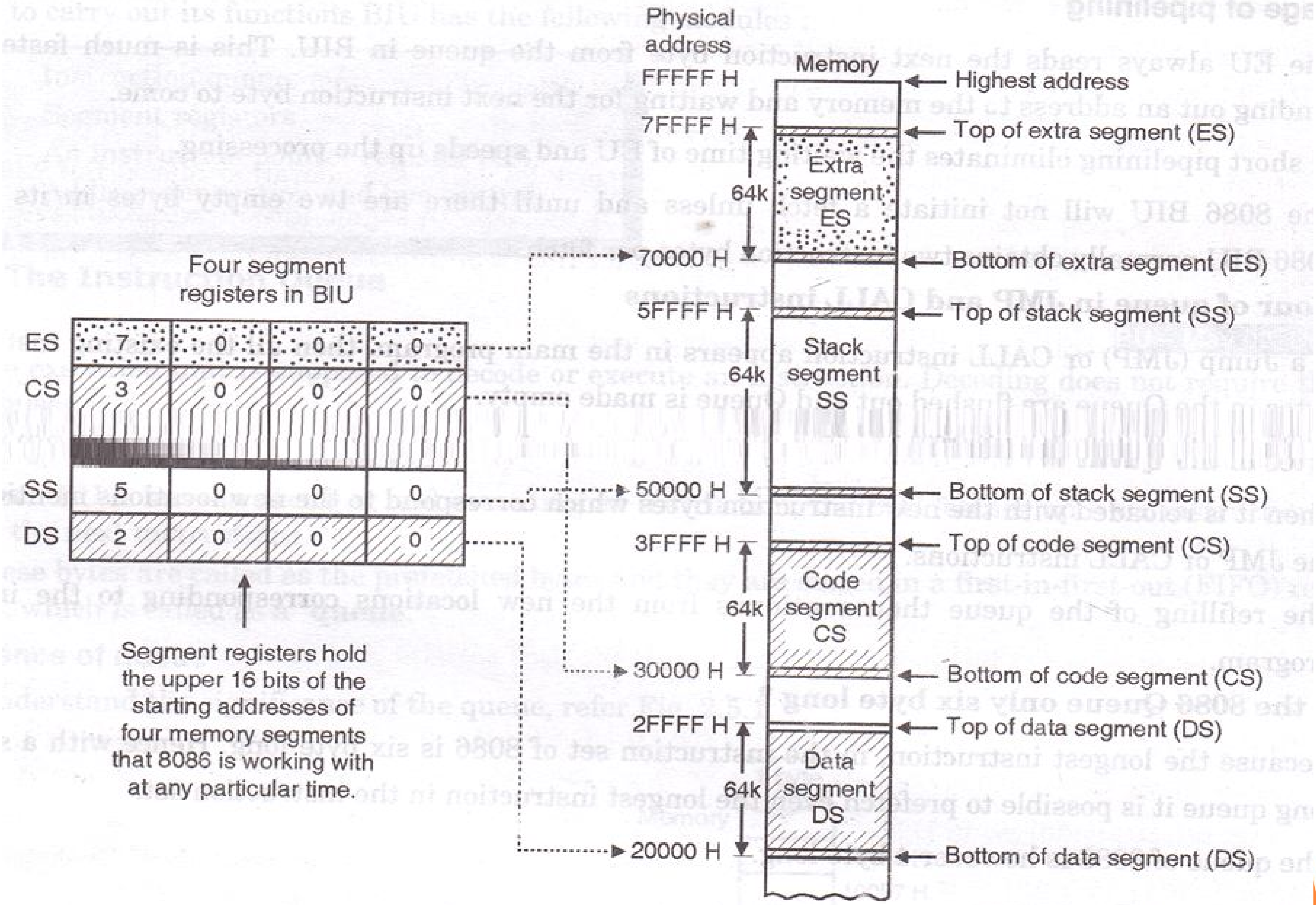
Advantages of Segmentation

Advantages of Segmentation

- Provides powerful memory management
- Supports modular S/W design
- Easily implement Object Oriented Program
- Allow two processes to easily share data
- Allows to extend address ability of a processor
- Possible to separate memory areas
- Possible to increase memory size of code data or stack segment
- Possible to write a program which is Memory independent or Dynamic relocatable.

Advantages of Segmentation

- Allows the memory capacity to be 1Mb although the actual addresses to be handled are of 16 bit size.
- Allows the placing of code, data and stack portions of the same program in different parts (segments) of the memory, for data and code protection.
- Permits a program and/or its data to be put into different areas of memory each time program is executed, i.e. provision for relocation may be done .
- The segment registers are used to allow the instruction, data or stack portion of a program to be more than 64Kbytes long. The above can be achieved by using more than one code, data or stack segments.



Physical address
Memory
FFFFF H
Highest address
7FFFF H
Top of extra segment (ES)
64k
Extra segment ES
70000 H
Bottom of extra segment (ES)
Four segment registers in BIU
5FFFF H
Top of stack segment (SS)
ES 7 0 0 0
CS 3 0 0 0
64k
Stack segment SS
SS 5 0 0 0
50000 H
Bottom of stack segment (SS)
DS 2 0 0 0
3FFFF H
Top of code segment (CS)
64k
Code segment CS
30000 H
Bottom of code segment (CS)
Segment registers hold the upper 16 bits of the starting addresses of four memory segments that 8086 is working with at any particular time.
2FFFF H
Top of data segment (DS)
64k
Data segment DS
20000 H
Bottom of data segment (DS)

- Thus every location within the segment can be accessed using 16 bits.
- 8086 allows only four active segments provided by BIU

Physical Address Generation

Instruction Poinetr and Summing block

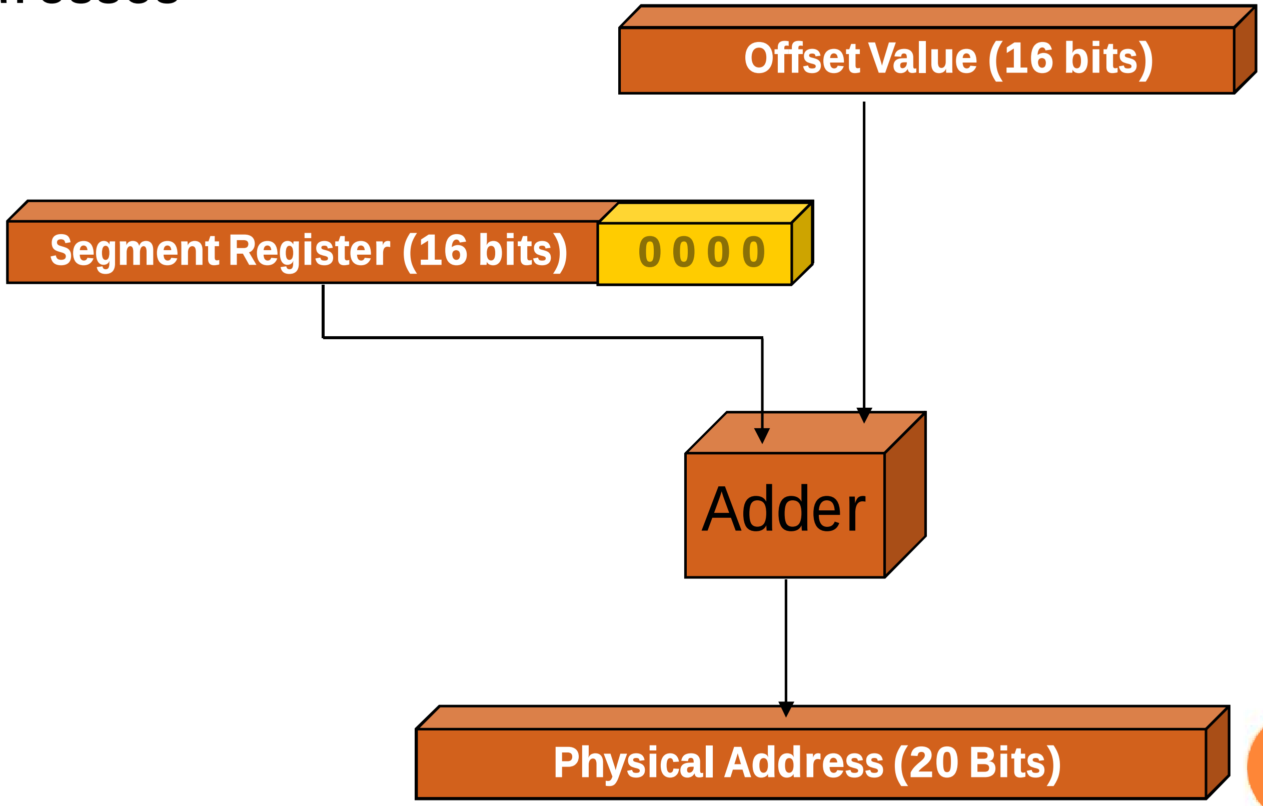
- The *instruction pointer* register contains a *16-bit offset address of instruction* that is to be executed next.
- The **IP** always references the *Code segment register (CS).*
- The value contained in the instruction pointer is called as an *offset* because this value must be added to the *base address* of the *code segment*, which is available in the CS register to find the *20-bit physical address*.

Instruction Poinetr and Summing block

- The *value of the instruction pointer* is incremented after executing *every instruction*.
- To form a *20bit 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.

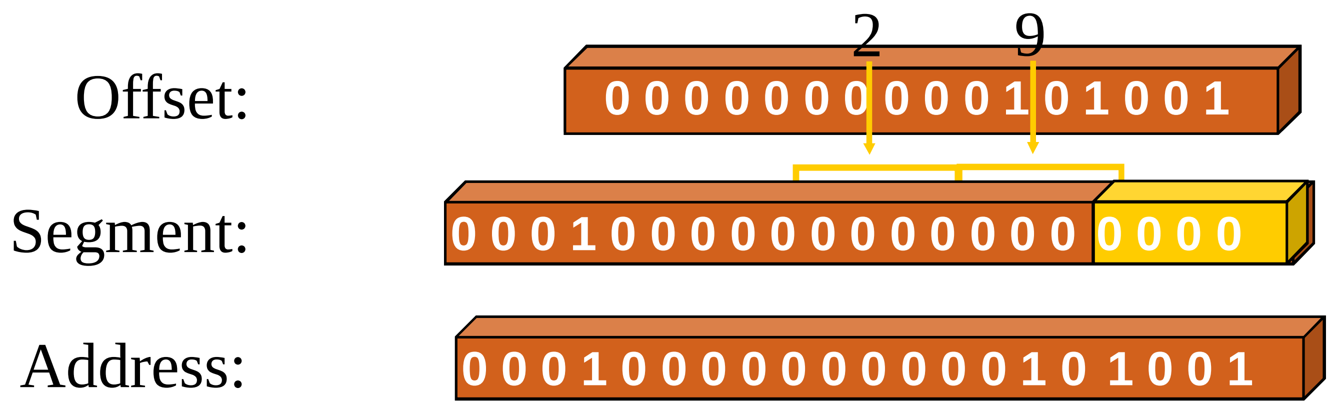
Physical Address generation

- The BIU has a dedicated adder for determining physical memory addresses



Example:Physical Address generation

- If the data segment starts at location 1000h and a data reference contains the address 29h where is the actual data?



Physical 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

```
  A4FB0
+ 4872
  A9822
```

Example

•If DS=7FA2H and the offset is 438EH

a) Calculate the physical address

7FA20 + 438E = 83DAE

b) calculate the lower range

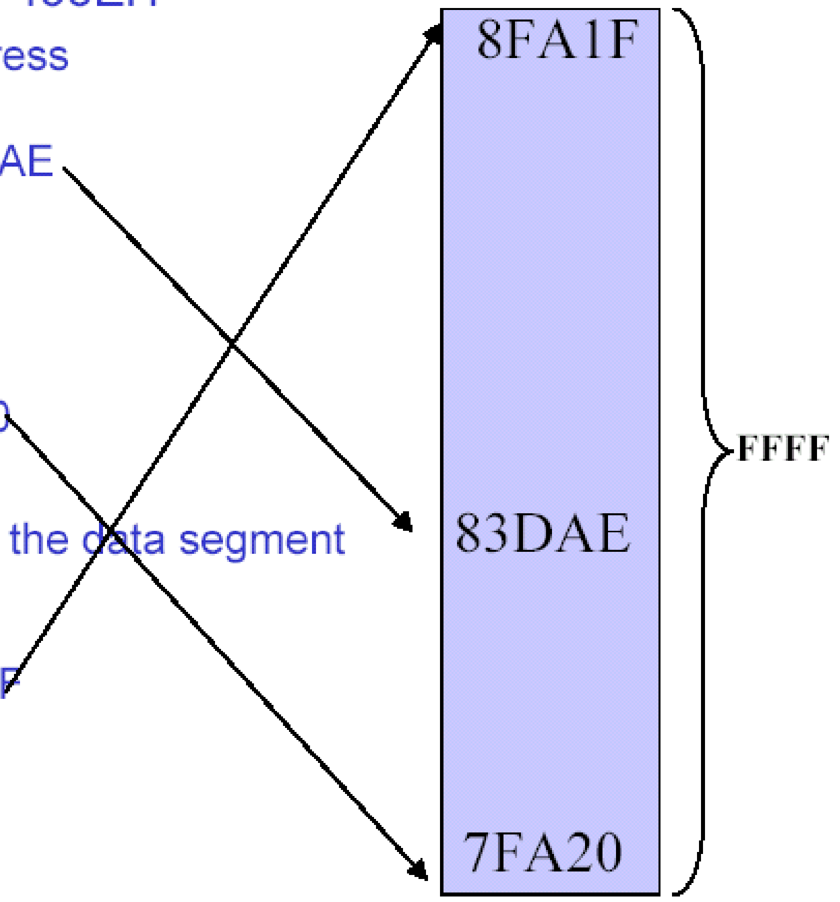
7FA20 + 0000 = 7FA20

c) Calculate the upper range of the data segment

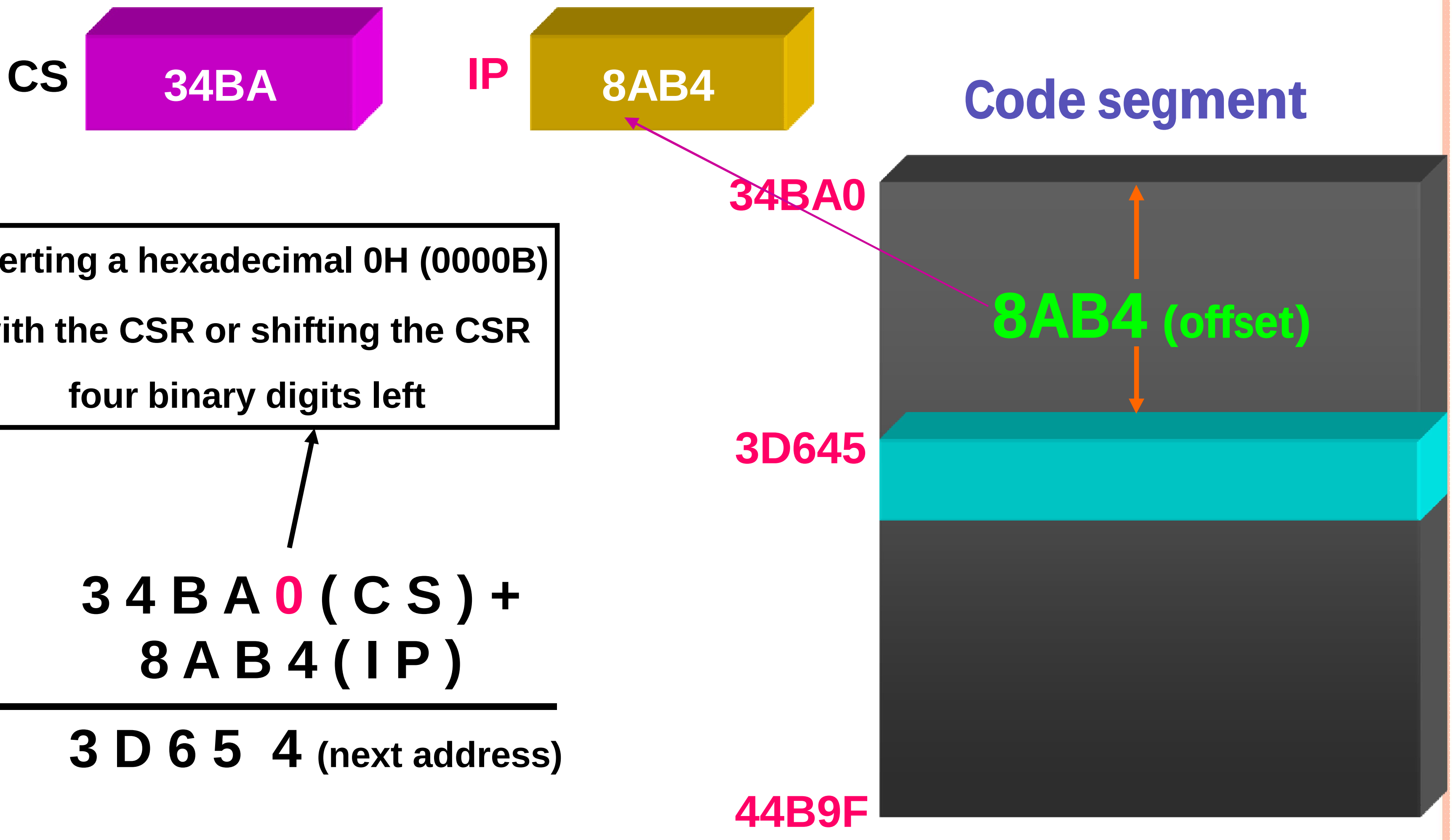
7FA20 + FFFF = 8FA1F

d) Show the logical Address

7FA2:438E



Physical Address generation



Inserting a hexadecimal 0H (0000B) with the CSR or shifting the CSR four binary digits left

3 4 B A 0 (C S) +
8 A B 4 (I P)

3 D 6 5 4 (next address)

Segment and Index register combination

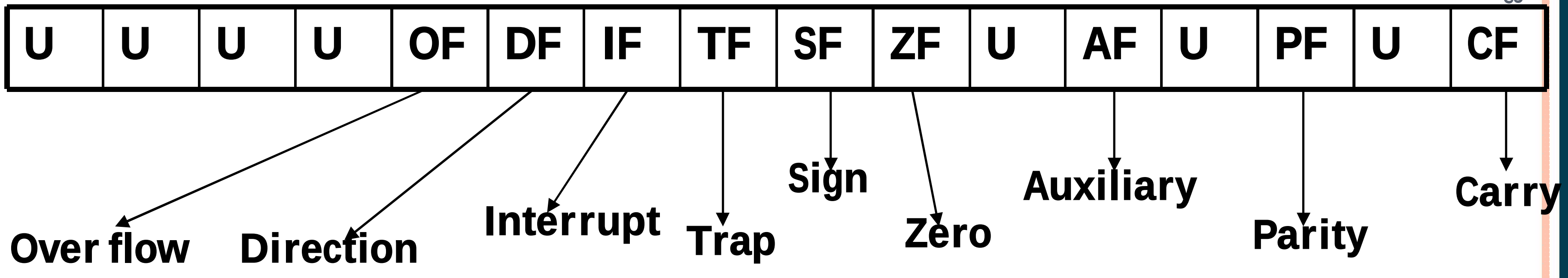
- CS:IP
- SS:SP SS:BP
- DS:BX DS:SI
- DS:DI (for other than string operations)
- ES:DI (for string operations)

Flag Registers

Flag Register

- A flag is a flip flop which indicates some conditions produced by the execution of an instruction or controls certain operations of the EU .
- In 8086 The EU contains
  - a 16 bit flag register
  - 9 of the 16 are active flags and remaining 7 are undefined.
  - 6 flags indicates some conditions- status flags
  - 3 flags –control Flags

Flag Register



U - Unused

Flag Register

Flag	Purpose
Carry (CF)	Holds the carry after addition or the borrow after subtraction. Also indicates some error conditions, as dictated by some programs and procedures .
Parity (PF)	PF=0;odd parity, PF=1;even parity.
Auxiliary (AF)	Holds the carry (half – carry) after addition or borrow after subtraction between bit positions 3 and 4 of the result (Lower nibble to higher nibble)(Used for BCD operation and is not available for programmer.)
Zero (ZF)	Shows the result of the arithmetic or logic operation is zero or not. (Z=1; result is zero. Z=0; The result is 0)
Sign (SF)	Holds the sign of the result after an arithmetic/logic instruction execution. (S=1: Negative, S=0 : Positive)

Flag	Purpose
Trap (TF)	(Single Steping) Enables the trapping through an on-chip debugging feature.
Interrupt (IF)	(Allow/Prohibit the interruption of program) Controls the operation of the INTR (interrupt request) I=0; INTR pin disabled. I=1; INTR pin enabled.
Direction (DF)	It selects either the increment or decrement mode for DI and /or SI registers during the string instructions. (DF=0 : String is processed from begining DF=1: String is processed from High address to low address)
Overflow (OF)	Overflow occurs when signed numbers are added or subtracted. An overflow indicates the result has exceeded the capacity of the Machine
Sign (SF)	Holds the sign of the result after an arithmetic/logic instruction execution. (S=1: Negative, S=0 : Positive)

Flag Register

- Six of the flags are status indicators reflecting properties of the last arithmetic or logical instruction.
- For example, if register AL = 7Fh and the instruction ADD AL,1 is executed then the following happen

AL = 80h

CF = 0; there is no carry out of bit 7

PF = 0; 80h has an odd number of ones

AF = 1; there is a carry out of bit 3 into bit 4

ZF = 0; the result is not zero

SF = 1; bit seven is one

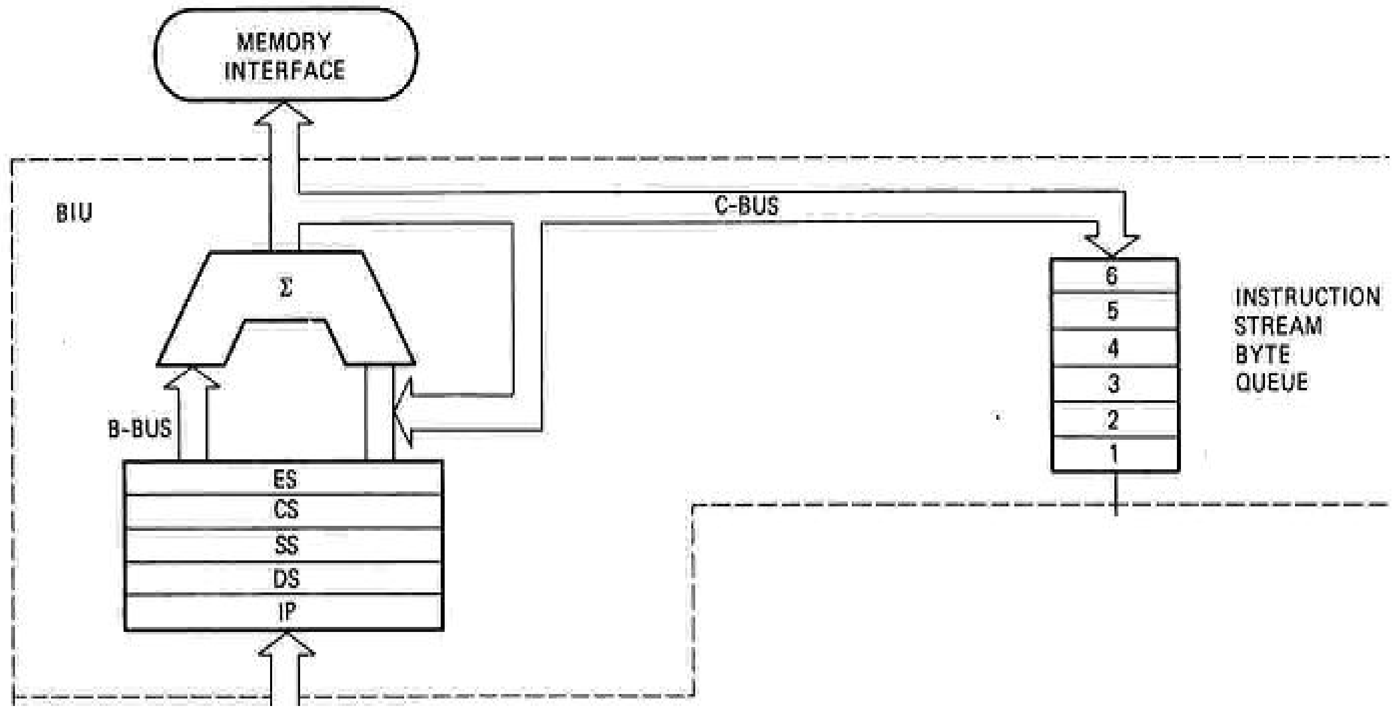
OF = 1; the sign bit has changed

BIU

BIU

Contains

- 6-byte Instruction Queue (Q)
- The Segment Registers (CS, DS, ES, SS).
- The Instruction Pointer (IP).
- The Address Summing block (Σ)



The Queue

Queue

- The BIU uses a mechanism known as an instruction stream queue to implement a *pipeline architecture.*

- This queue permits pre-fetch of up to 6 bytes of instruction code. Whenever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by pre-fetching the next sequential instruction.

Queue

- These pre-fetching instructions are held in its **FIFO queue**. With its **16** bit data bus, the BIU fetches two instruction bytes in a single memory cycle.
- After a byte is loaded at the input end of the queue, it automatically shifts up through the **FIFO** to the empty location nearest the output.
- The **EU accesses the queue from the output end**. It reads one instruction byte after the other from the output of the queue.
- The intervals of no bus activity, which may occur between bus cycles are known as ***Idle state*.**