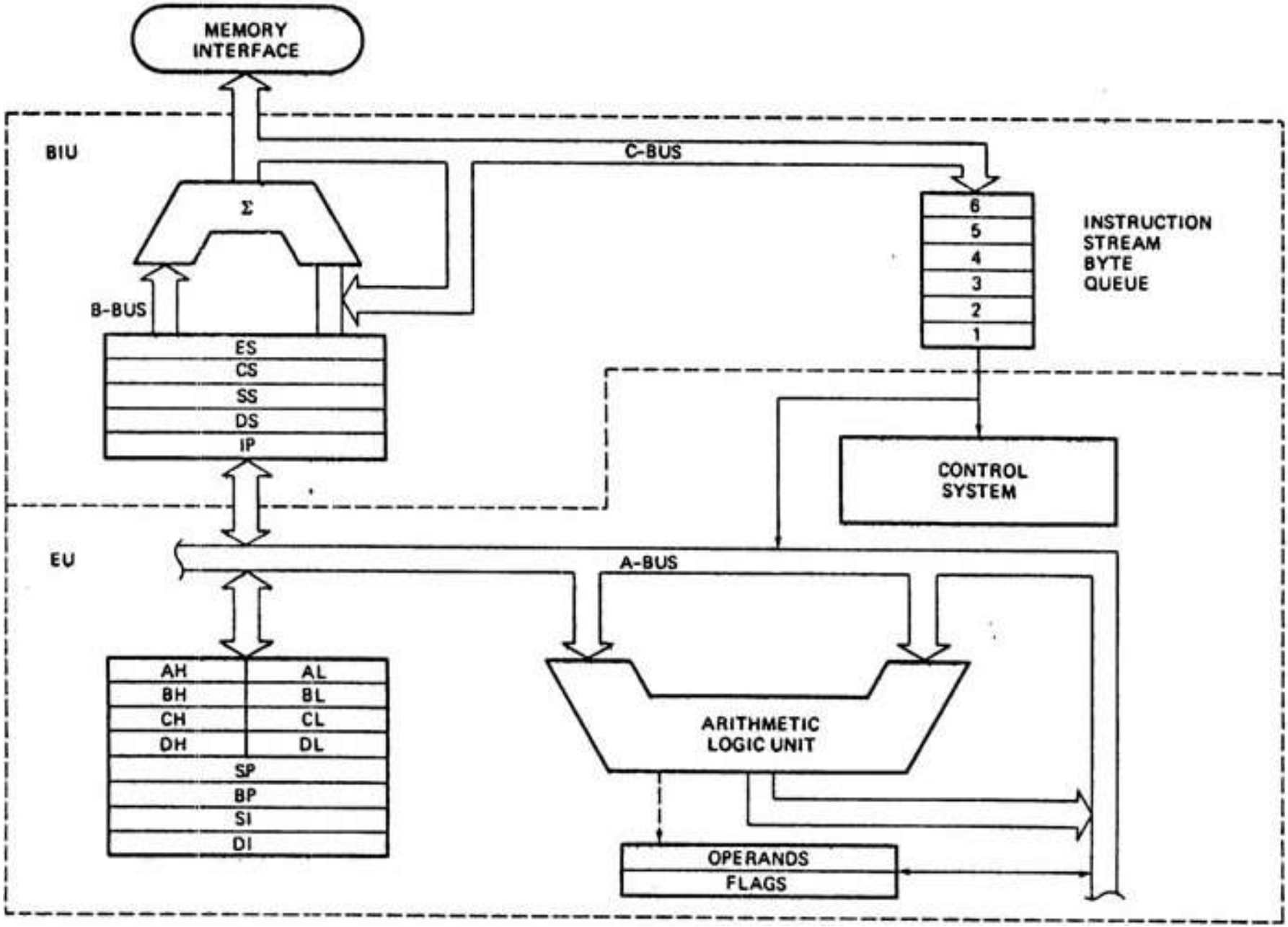
Introduction to 8086/8088 Microprocesser

- General Facilities
- BIU and EU
- Data Registers
- Segment Registers
- Index Registers
- Pointer Registers
- Flag Register
- Memory Addressing
- Physical Memory Address Calculations.

8086 – Introduction

- 16 bit µp with 20 bit address bus & 16 bit data bus
- Can address up to 2^{20} = 1 MB memory directly
- Can read or write 8 or 16 bit data
- 8088 - 8 bit data bus
- Internal architecture has two main units
- BIU - Bus Interface Unit
- EU – Execution Unit


MEMORY INTERFACE
BIU
C-BUS
Σ
B-BUS
6
5
4
3
2
1
INSTRUCTION STREAM BYTE QUEUE
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

- BIU and EU work simultaneously for instruction execution and form 2 stage instruction pipeline.
- BIU – has bus interface logic, segment registers, memory addressing logic and 6 byte instruction queue.
- While EU is busy in instruction execution, BIU fetches instruction from memory and stores them in instruction queue.

- If EU executes a jump instruction
  - Program control to be transferred to another location then BIU
    - Resets the queue
    - Fetches instruction from new address
    - Passes the instruction to EU
    - Starts refilling the queue from new location
    - Process is called pipeline flushing.

BIU of 8088

- 8088 – 4 byte instruction queue
- Each time there is empty space of 2 bytes or more 8086 BIU startes filling the queue.
- 8088 BIU fills the queue when there is empty space of 1 byte.

Execution Unit (EU)

- Contains – instruction decoder, ALU, general purpose registers, pointer registers, index registers, flag register and control circuitary to execute instruction.
- EU is resposible for-
  - Execution of instructions
  - Providing address to BIU for fetching data/instruction.
  - Manipulating various registers as well as flag register.
- EU is almost completly isolated from outside world.
- BIU performs all bus operations for EU.
- Overlapped fetching and execution of instruction is shown

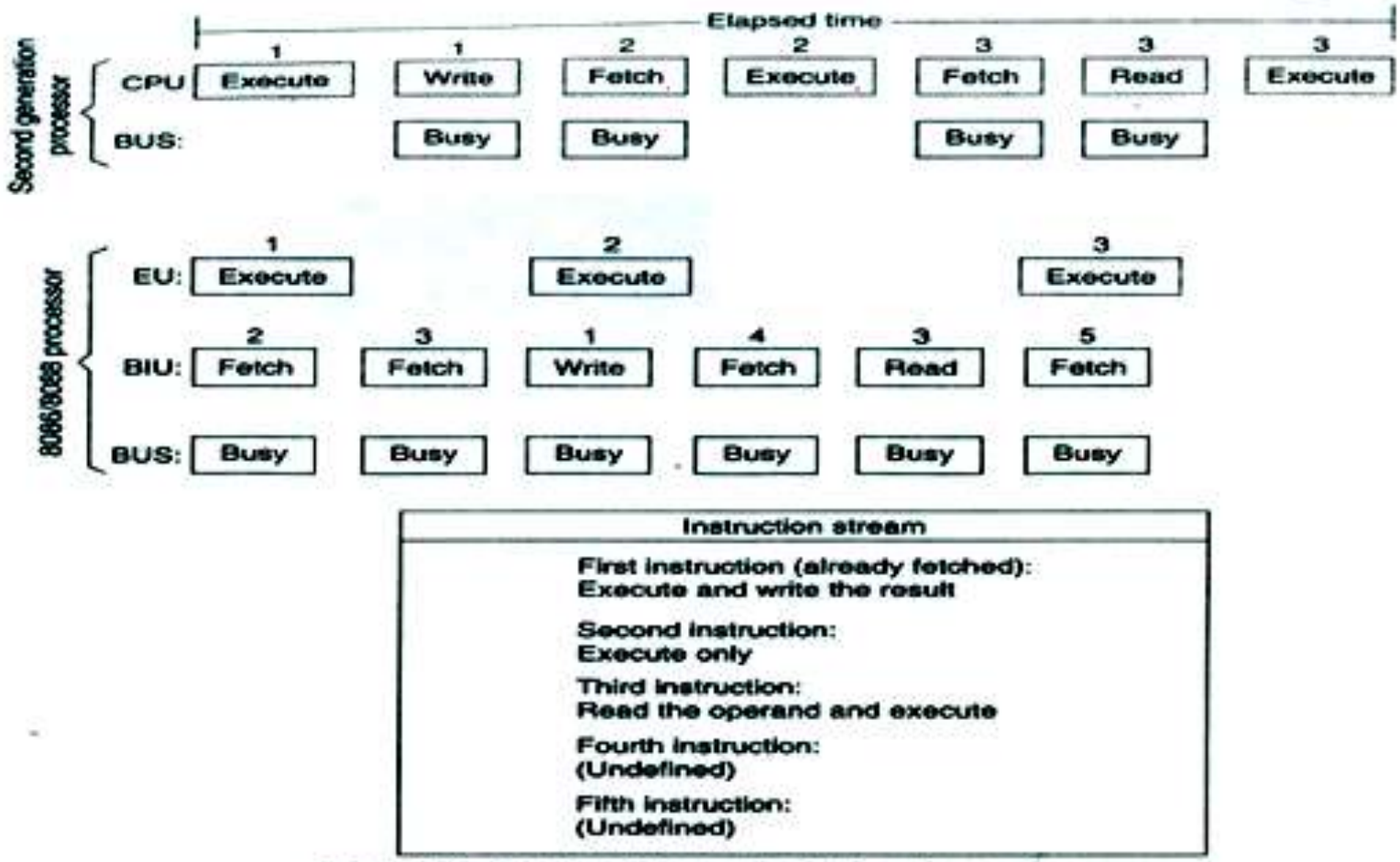


Figure 5.2 Overlapped instruction fetch and execution.

It is assumed that

- Two µps i.e 2nd generation and 8086 execute the instruction in same no. of clock cycles.
- Due to overlapping of fetch and excute, 8086/8088 is able to excute 3 instruction in lesser time but also able to fetch 4th and 5th instruction.
- **Register Set-** 8086 has fourteen – 16 bit registers
    - Data Registers
    - Segment Registers
    - Pointer & index register
    - flag register

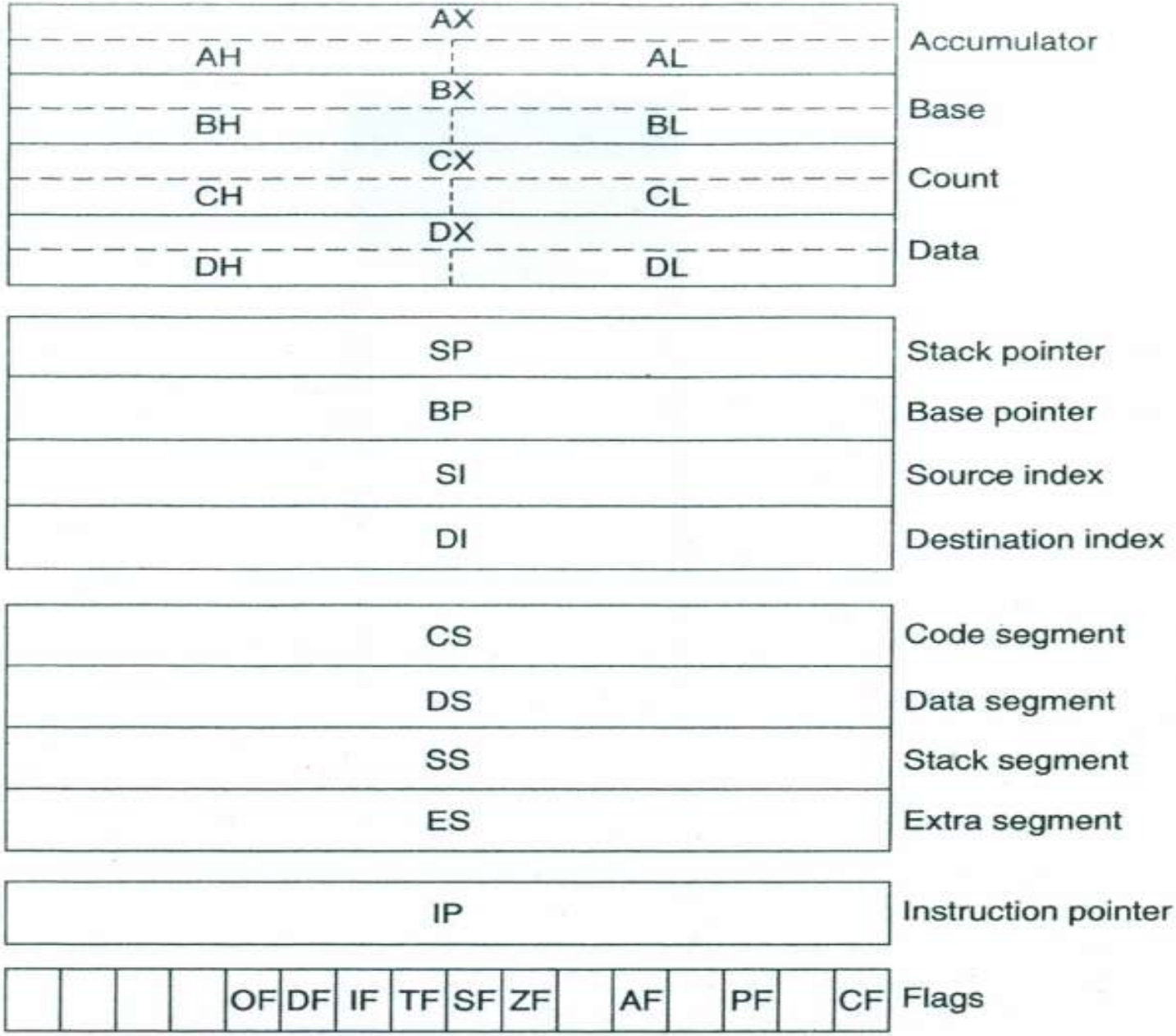
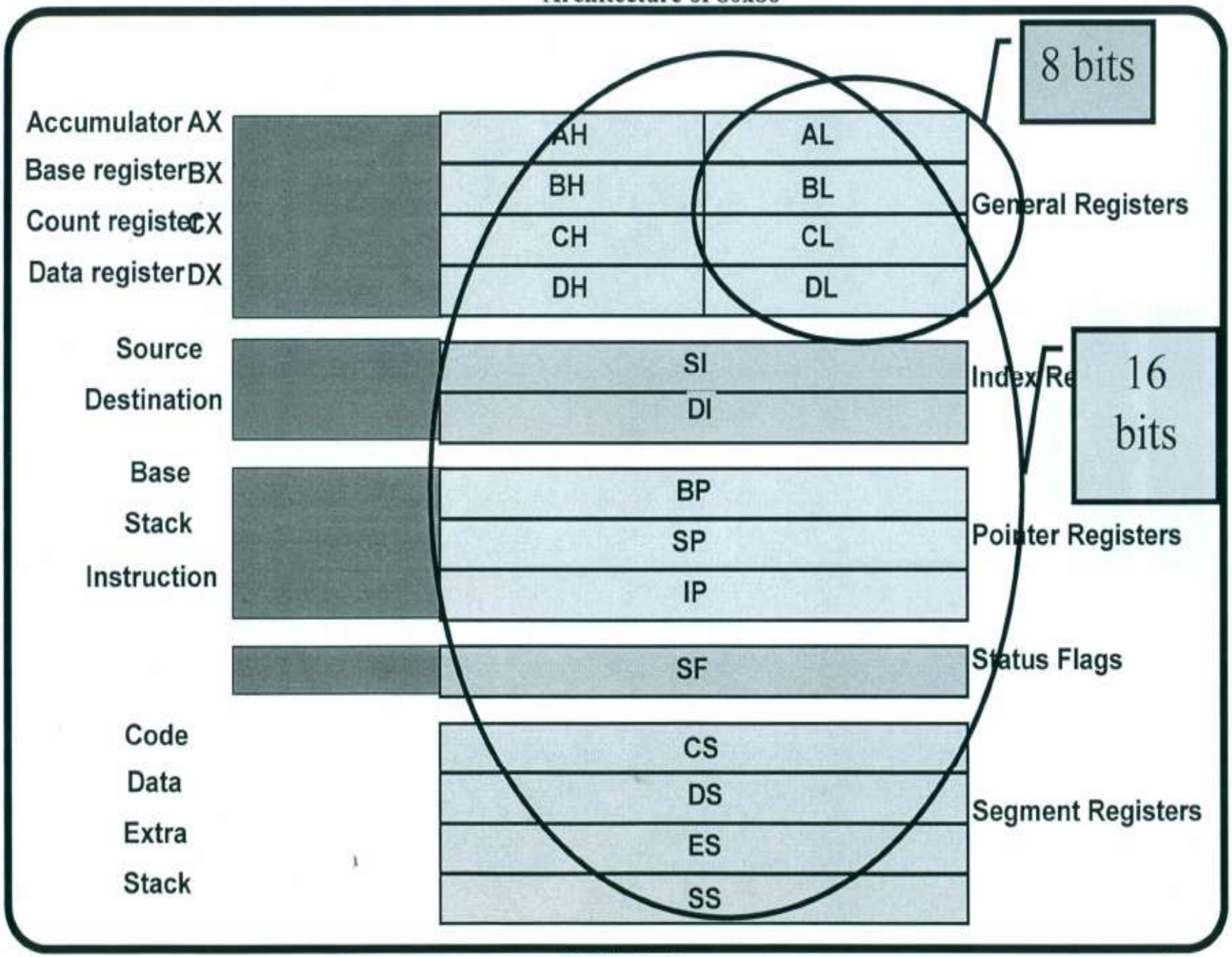


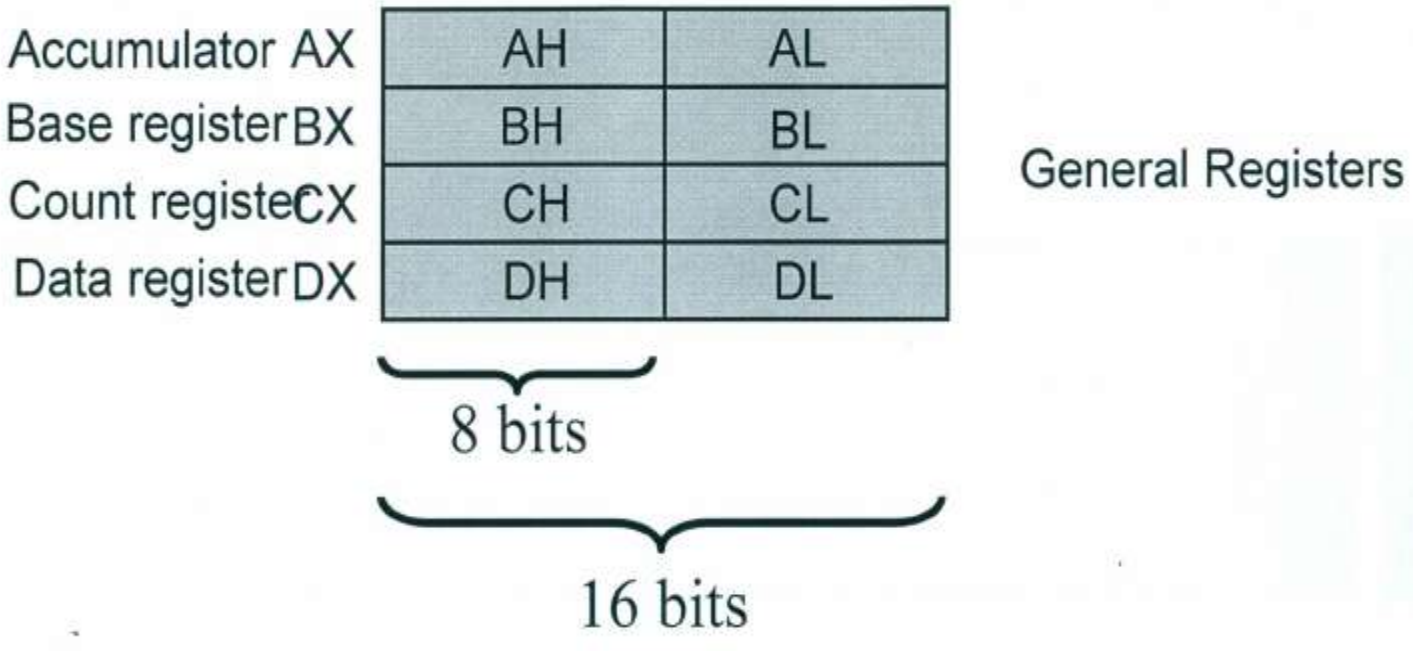
Figure 5.3 Intel 8086/8088 register set.

- Data Registers- Four sixteen bit AX,BX,CX,DX
  - Or 8 – 8 bit registers
  - AH AL BH BL CH CL DH DL
- AX- Primary Accumulator
  - Input-Output operations through AX(or AL)
  - Several string operations require one operand to be in AX (or AL).
  - 16 bit divide or multiply - AX has one word operand (AL for 8 bit operation).
  - 32 bit multiply or divide – AX holds lower order word operand.


8 bits
Accumulator AX
Base register BX
Count register CX
Data register DX
AH
AL
BH
BL
CH
CL
DH
DL
General Registers
Source
Destination
SI
DI
Index Re
16 bits
Base
Stack
Instruction
BP
SP
IP
Pointer Registers
SF
Status Flags
Code
Data
Extra
Stack
CS
DS
ES
SS
Segment Registers

- BX – GPR (Genral Purpose Register) + Base Register
- CX – GPR + Count Register in multi iteration instruction.
- DX – GPR + used in I/O instructions, multiply & divide instructions.

General Registers



Many instructions in a computer program involve moving data into and out of these general registers.

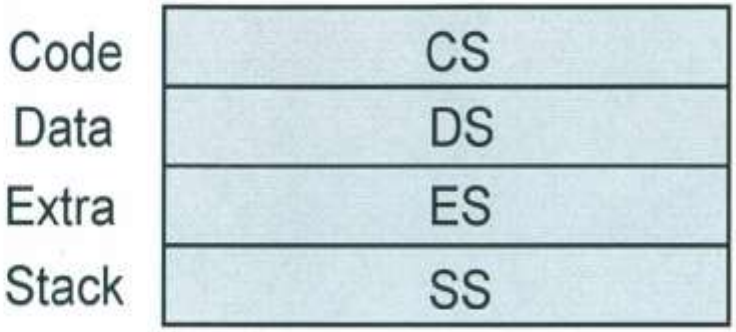
Segment Registers

- concept of memory segmentation.

  Memory divided into no. of parts called segments.

- In 8086, 1 MB physical memory can be divide into 4 types of segments.
- Starting address of each segment is placed in 16 bit register
- CS – Code Segment Register.
- DS – Data Segment Register.
- SS – Stack Segment Register.
- ES – Extra Segment Register.
- Each segment has memory space of 64k byte

Segment Registers



Segment Registers

- Segment registers contain segment addresses.
- The Code segment contains the program being executed.
- The Data segment contains data (variables).
- The Extra segment can be used for anything.
- The Stack segment contains the stack.
- Each segment is of 64KB.

The advantage of using segment register are that they;

1. Allow the memory capacity to be 1 M even though the address associated with the individual instruction are only 16 bit wide.
2. Allow the instruction ,data, or stack portion of a program to be more than 64k bytes long by using more than one code, data, or stack segment
3. Facilitate the 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.

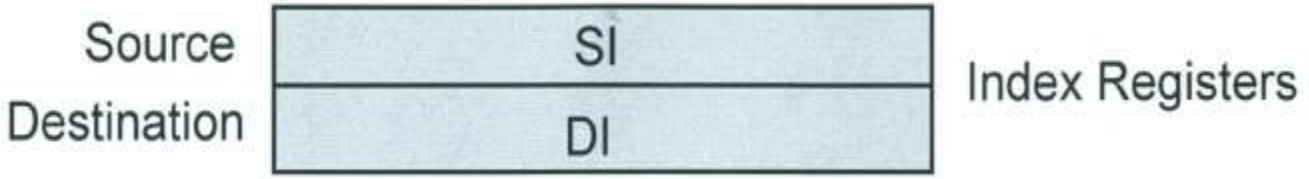
Index Registers

SI – Source Index Register

DI – Destination Index Register

- May be used as GPR. However main purpose is to store address offset in case of – Indexed, Base Indexed, and Relative Base Indexed addressing modes

Index Registers



- Index registers may be used to store 16-bit data.

- Index registers are also used as pointers to offset addresses in memory.

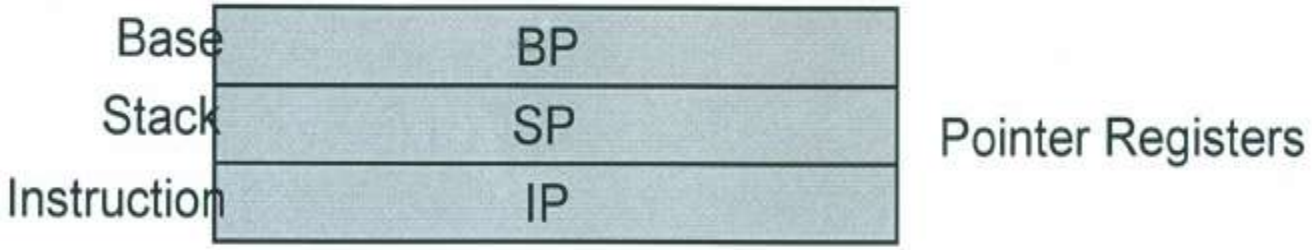
- **`MOV AL,[SI]`**

means move into AL the byte in the data segment at the offset address that is in SI.

Pointer Registers

- Stack Pointer(SP) - provides access to stack top. Holds offset address of stack top.
- Instruction Pointer(IP) - holds offset address of next instruction to be executed. Like program counter.
- Base Pointer (BP) - GPR. Main purpose is to provide indirect access to data in stack.

Pointer Registers



The Instruction Pointer, IP, contains the offset address of the next instruction to be executed.

The Stack Pointer, SP, contains the offset address of the top of the stack.

The Base Pointer, BP, is used to access data in the stack segment.

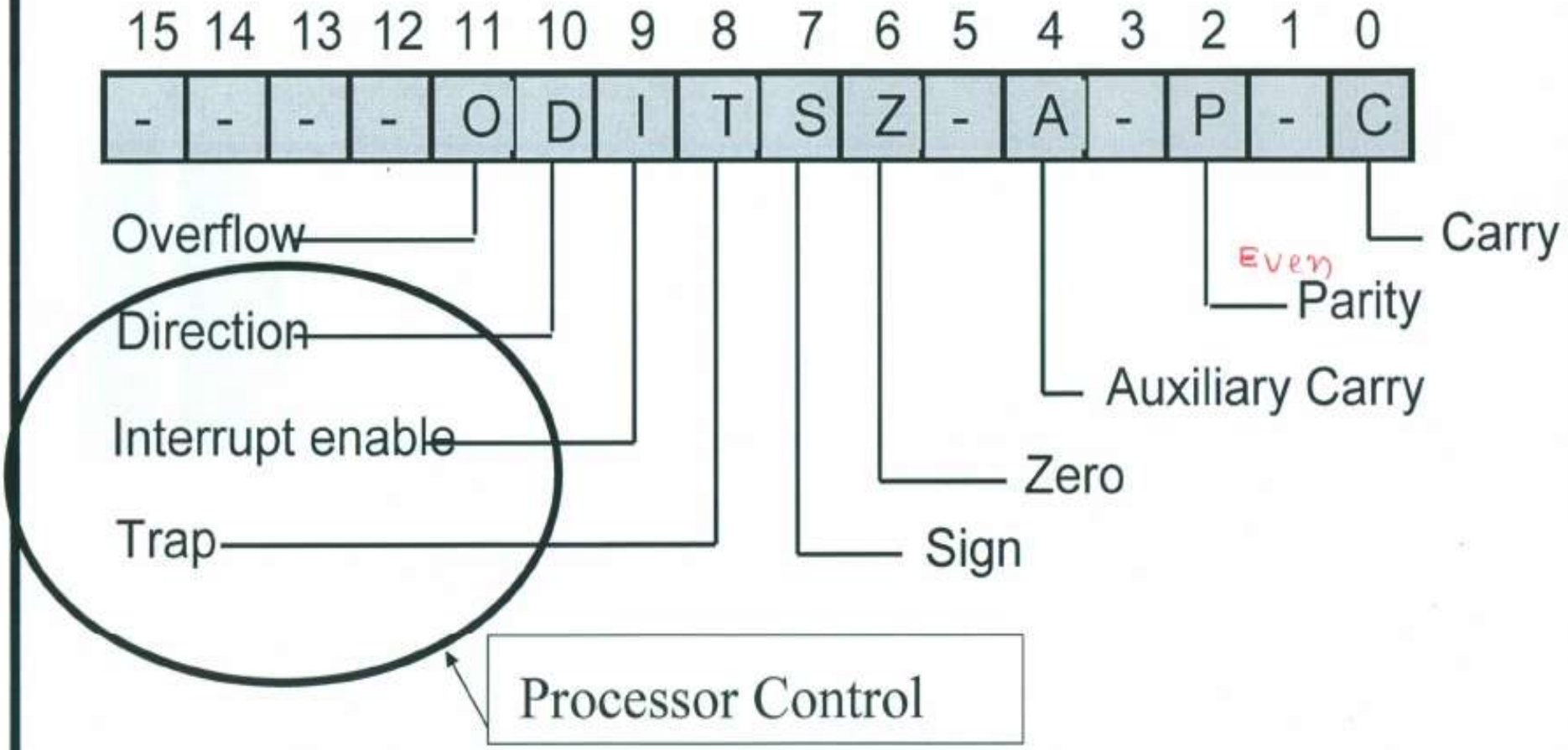
- Flag Register - also called program status word (PSW)

9 active Flags

6 status flags

3 control flags

The Status Register



Status Flag –

- Carry – Arithmetic operation results is carry out of MSB or Borrow in Subtraction. Also used in shift & Rotate.
- Parity – Even parity in byte operation or lower byte of word operation
- Auxiliary Carry – set if there in carry out of lower nibble to higher nibble in 8 bit quantity or lower byte to higher byte in 16 bit quantity. Used in BCD operations.
- Zero – set when result of operation is zero.

- Sign – set when after arithmetic or logic operation, MSB of result = 1 indicates that result is negative.
- Overflow – used to detect magnitude overflow in signed arithmetic.
    - For addition operation – flag is set when there is carry in to MSB but no carry out of MSB or vice versa.
    - For subtraction – flag is set when MSB needs a borrow but there is no borrow from MSB or vice versa.

Control Flags –

- Direction – Used with string operations. When set (i.e = 1) then string operation in auto decrement mode (i.e right to left) when reset (i.e = 0) then string operation in auto increment mode (left to right)

Interrupt Enable –

- When = 1, enables 8086 to recognize external interrupts. When = 0, all maskable interrupts are disabled. No effect on non maskable interrupts.
- Trap flag – when = 1, sets processor in single step mode for debugging.

Memory Addressing – Real Mode

- Real Mode – limited to 1 MB of memory i.e real memory.
- Segment Register – 16 bits in length.
- Memory address – 20 bits.
- Segment register content are appended with 0000 to make it 20 bit address i.e segment register contents are multiplied by 16
- Now the address represent the starting address of segment in memory
- Let us say CS = 1234H
- After appending with 0000(=0H) we get 12340H as starting address of code segment in memory.
- The size of a segment i.e 64kb i.e FFFFH
- The address range of code segment 12340H to (12340 + FFFF) = 2233FH

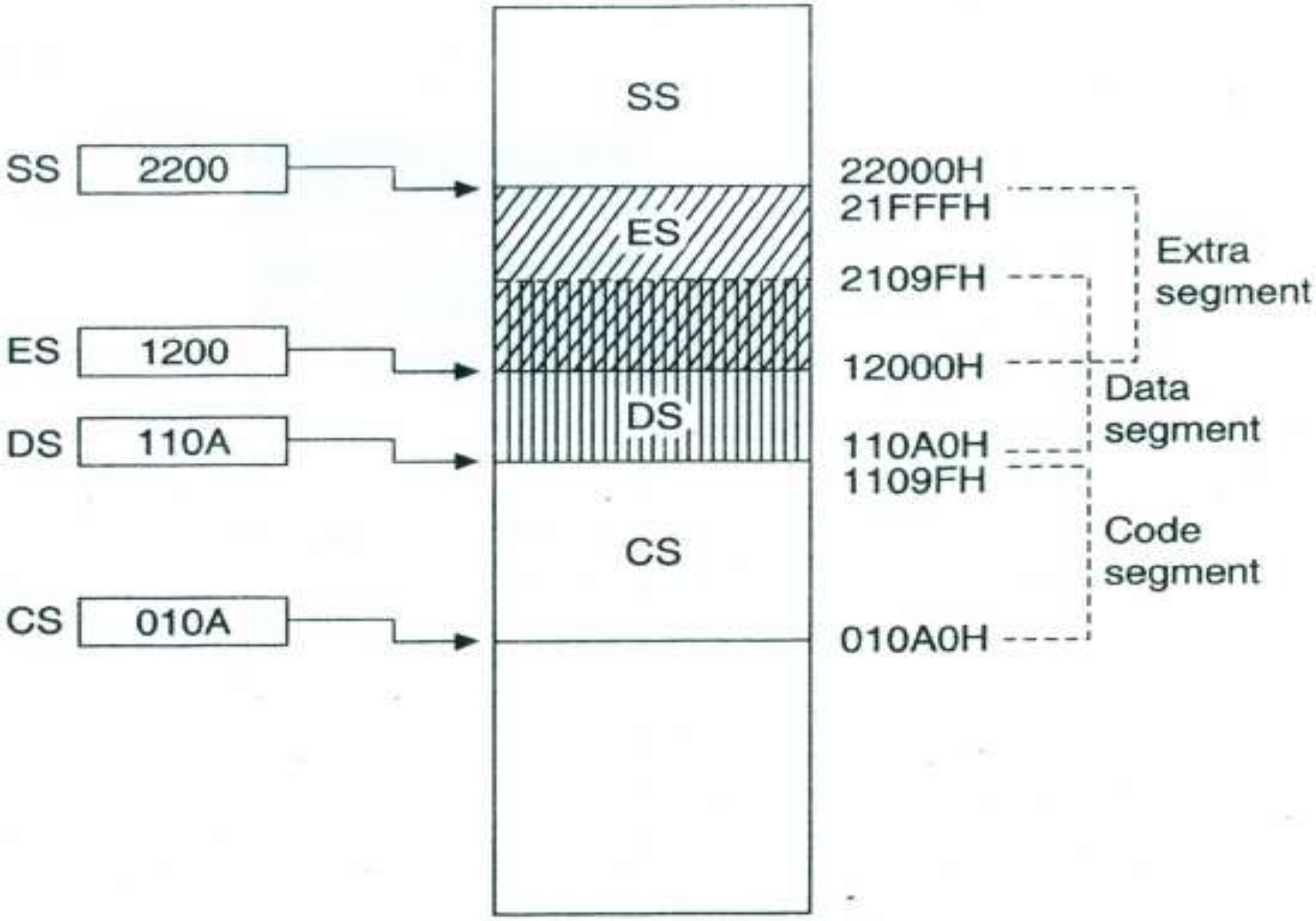
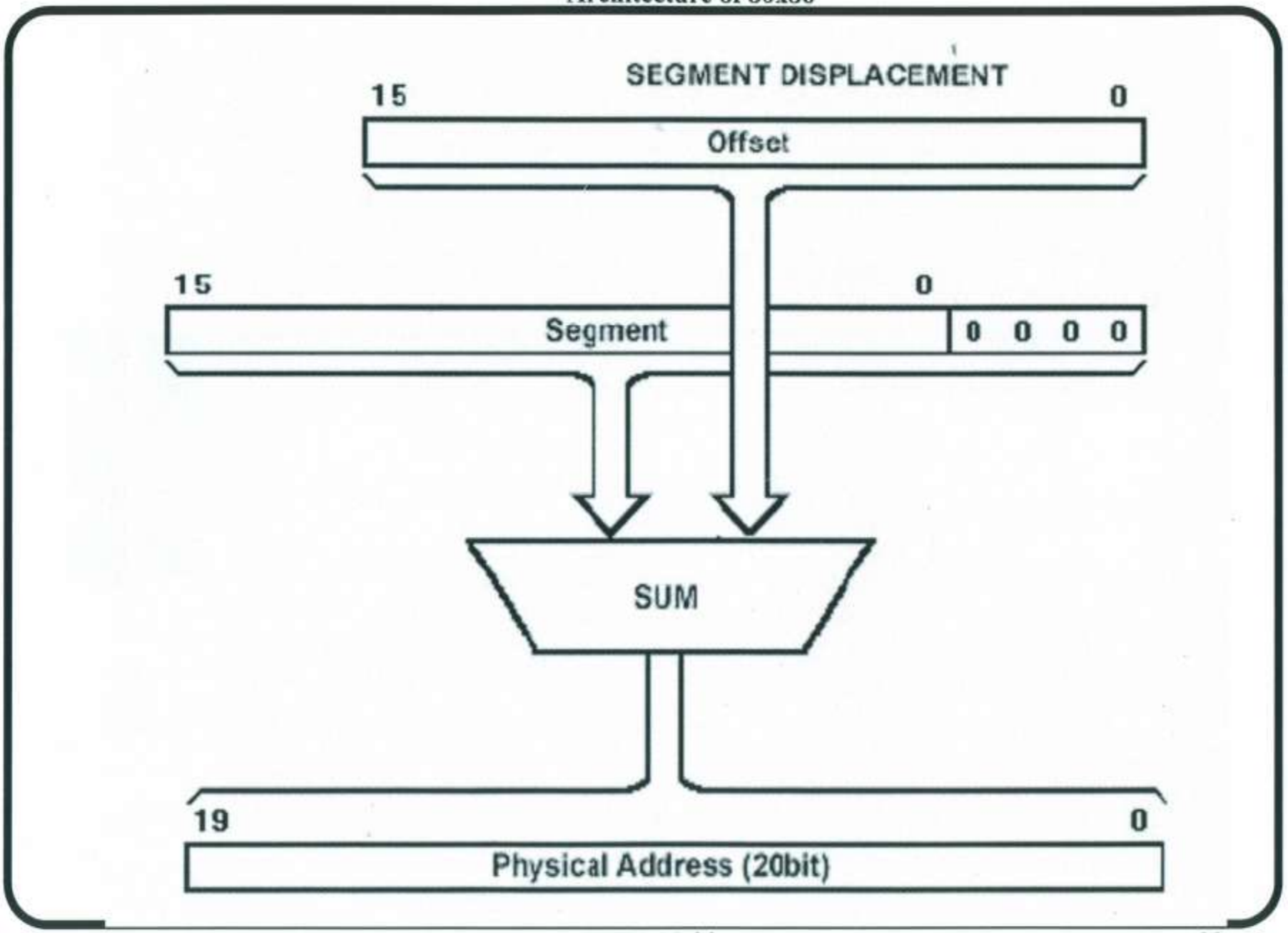


Figure 5.5 An example of memory segmentation.

- A segment must thus start at 16 byte boundary [since last 4 bits = 0000]
- Two segment can overlap with each other partially if segment address have been defined like that.
- There may be more then one segment of a particular type. For that segment register contents are dynamically changed.
- All real mode memory addresses consist of segment address plus an offset address.
- Offset address – increment from starting of segment.

- Thus segment address defines the starting address of 64k byte memory segment.
- Offset address selects a location within 64k byte memory segment.
- Offset address is also known as logical address.
- Physical address is actual memory location of operands = segment address appended with 0H + offset address = (segment address) x 16 + offset address.


SEGMENT DISPLACEMENT
15
0
Offset
15
0
Segment
0 0 0 0
SUM
19
0
Physical Address (20bit)

- Let us take an example.

-> Segment register = 1234H

-> Offset = 0022H

Physical address ?

Segment starting address = 12340H

+ offset = 0022H

Physical address = 12362H

Exercise 1 – calculate last address of segment, given the segment register values as 1234H, 2300H and AB00H.

Exercise 1

Show the physical ending addresses of each segment located by the following segment register values.

1234H = 12340H + FFFFH = 2233FH

2300H = 23000H + FFFFH = 32FFFH

AB00H = AB000H + FFFFH = BAFFFH

Exercise -2 – Given physical address of memory location and segment register content, find the logical address i.e offset

Exercise 2:

What would be the offset required to map to physical address location $002C3_{16}$ if the contents of the corresponding segment register are $002A_{16}$?

Solution

The offset address is obtained by shifting the contents of segment register left by four bit positions can then subtracting from the physical address. Shifting left gives

$$002A0_{16}$$

Now subtracting, we get the value of offset:

$$002C3_{16} - 002A0_{16} = 0023_{16}$$

Exercise 3:

Determine the offset register content for the following

Segment Register	Memory location	Offset Register
DS = 1000h	12000h	DI = 2000h
DS = 2000h	21002h	SI = 1002h
DS = A000h	A1000h	BX = 1000h
CS = 3456h	3F12Dh	IP = ABCDh
SS = 2300h	26200h	BP = 3200h
SS = 2900h	2CA00h	SP = 3A00h

Note –

- 8086 and 8088 operate only in real mode
- 80286 – 80486 – Pentium – operate in real or protected mode.
- Real mode operation allows microprocessor to address only first 1 MB of memory space even if it is 80486 µp.
- Memory segmentation allows program to be relocated in any portion of memory.

- There is a set combination of segment register and offset register known as default combination .

Segment	offset	Special purpose
CS	IP	Instruction address
SS	SP or BP	stack address
DS	BX,DI,SI	Data address
	An 8 or 16 bit number	
ES	DI for string instruction	string destination address