UNIT 2

The 8086 Microprocessor: Architecture, Register organization, 8086 signal description, Physical memory organization, Minimum and Maximum

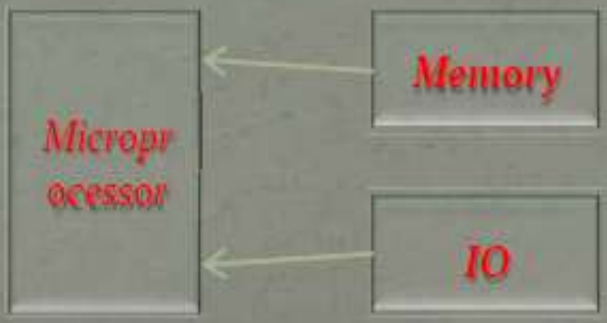
mode system and timing diagrams, Addressing modes, 8086 Instruction Set and Assembler Directives, Assembly Language example programs,

Stack structure of 8086, Interrupt structure of 8086, Interrupt vector table, Procedures and macros.

What is a microprocessor?

A Microprocessor is a multipurpose, programmable logic device that *reads binary instructions* from a storage device called memory, *accepts binary data* as inputs and *processes data* according to those instructions, and *provides results* as output.

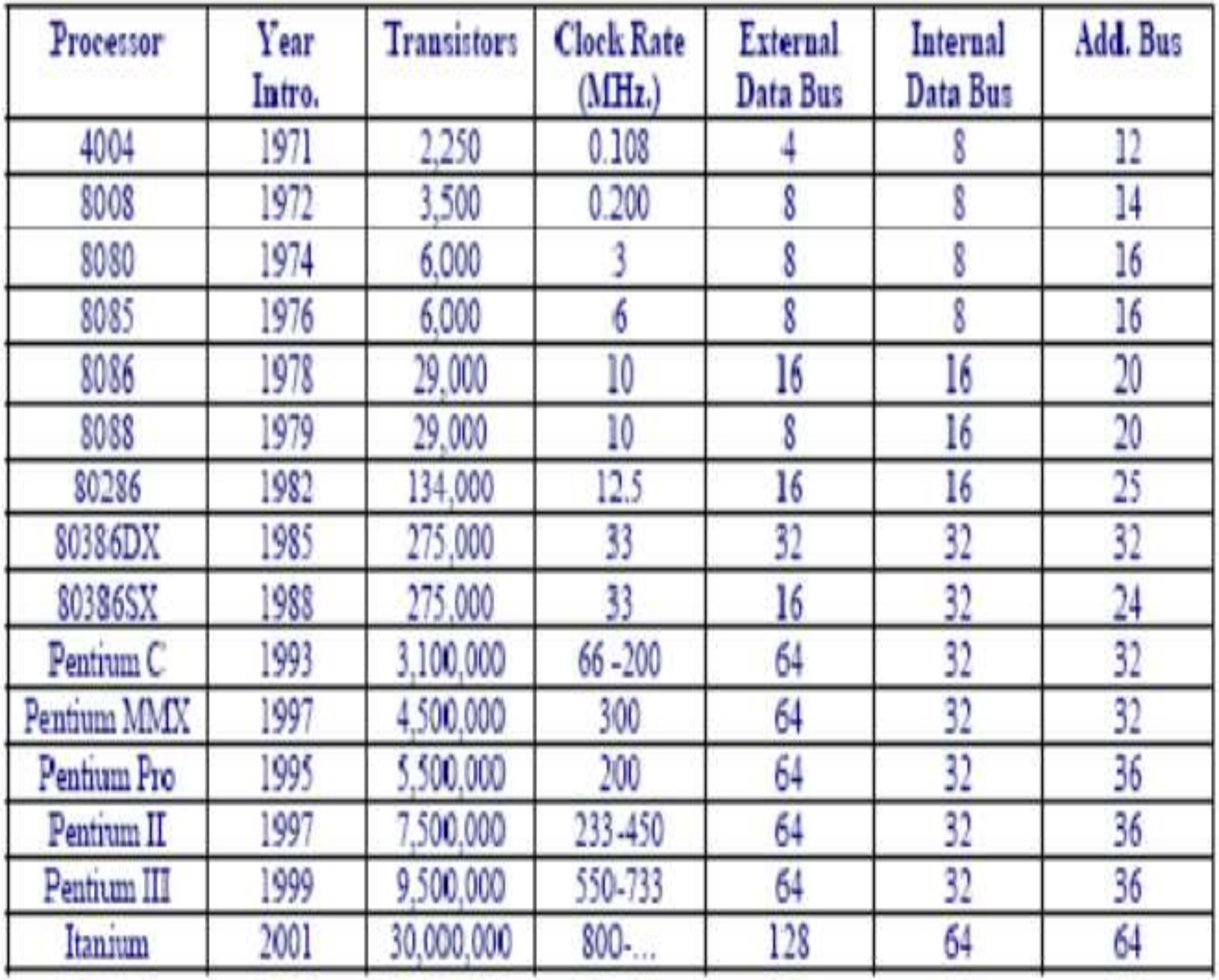
A Programmable machine



Micropr
ocessor

Memory

IO



Processor	Year Intro.	Transistors	Clock Rate (MHz.)	External Data Bus	Internal Data Bus	Add. Bus
4004	1971	2,250	0.108	4	8	12
8008	1972	3,500	0.200	8	8	14
8080	1974	6,000	3	8	8	16
8085	1976	6,000	6	8	8	16
8086	1978	29,000	10	16	16	20
8088	1979	29,000	10	8	16	20
80286	1982	134,000	12.5	16	16	25
80386DX	1985	275,000	33	32	32	32
80386SX	1988	275,000	33	16	32	24
Pentium C	1993	3,100,000	66 -200	64	32	32
Pentium MMX	1997	4,500,000	300	64	32	32
Pentium Pro	1995	5,500,000	200	64	32	36
Pentium II	1997	7,500,000	233-450	64	32	36
Pentium III	1999	9,500,000	550-733	64	32	36
Itanium	2001	30,000,000	800-...	128	64	64

Evolution of Intel's 80X86 Family Microprocessors

D8086
L3093607
INTEL ©1978

8086- Architecture: Features

- *It is a 16-bit µp.*
- *8086 has a 20 bit address bus can access up to 2^{20} memory locations (1 MB).*
- *It can support up to 64K I/O ports.*
- *It provides 14, 16 -bit registers.*
- *Word size is 16 bits and double word size is 4 bytes.*
- *It has multiplexed address and data bus AD0- AD15 and A16 – A19.*

- ***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.***

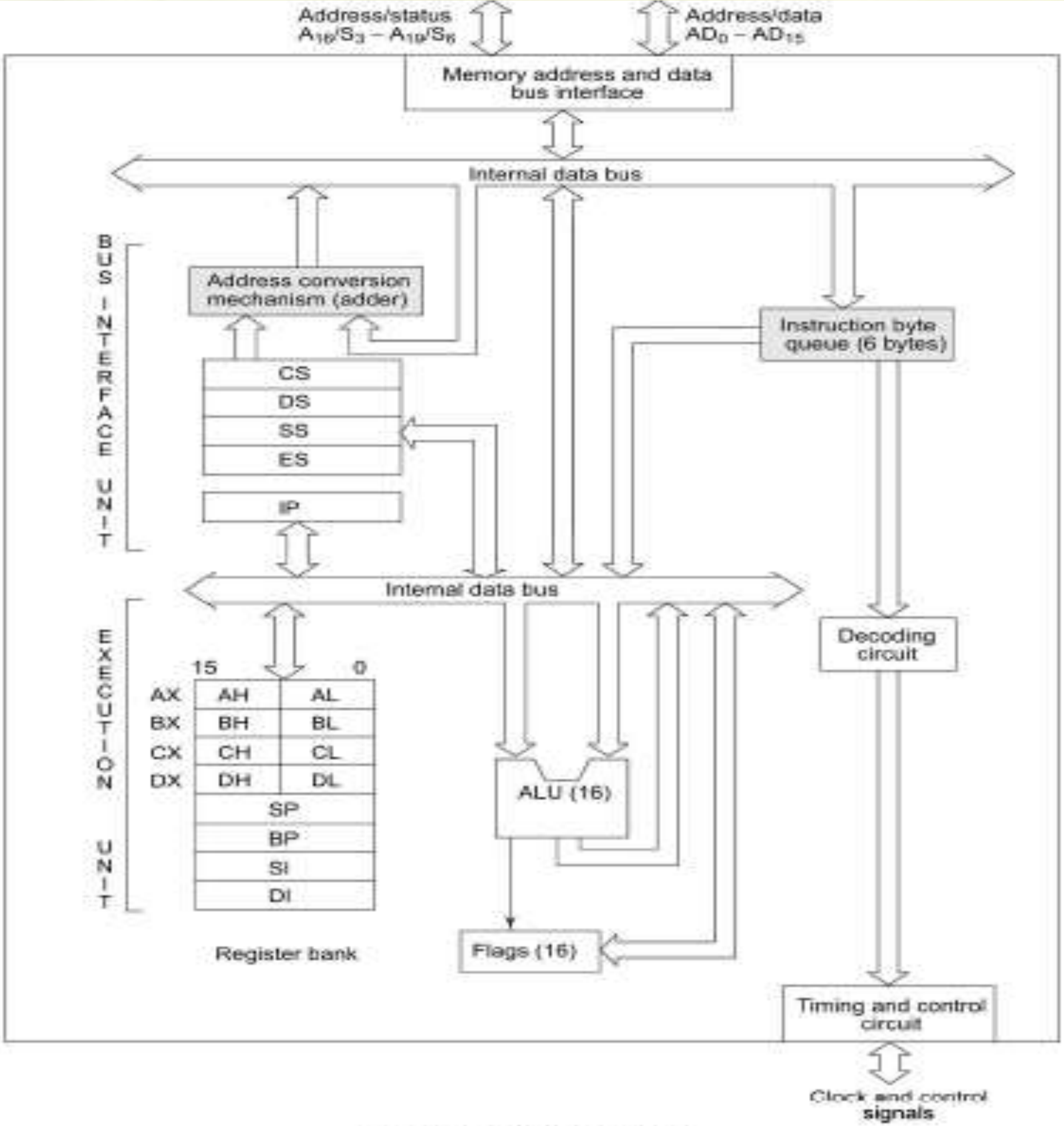


Fig. 1.2 ***8086 Architecture***

Internal architecture of 8086

- 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 byte queue.

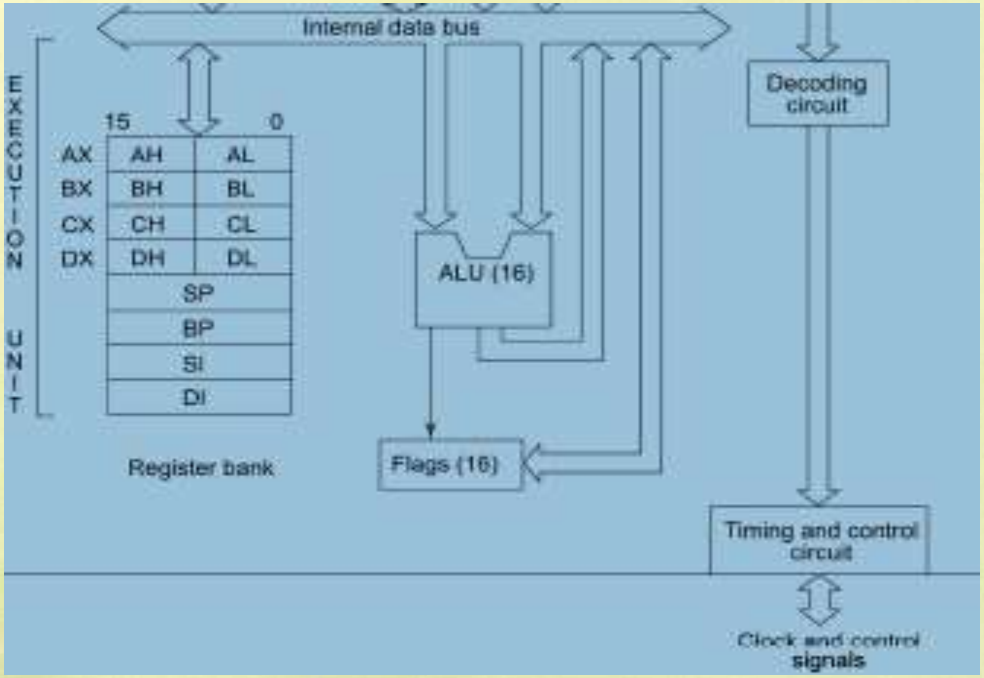
- Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining. This results in efficient use of the system bus and system performance.

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

- EU contains Timing and Control circuit, Decoding circuit, ALU, General data registers, Pointer and Index registers, Flag register.

EXECUTION UNIT

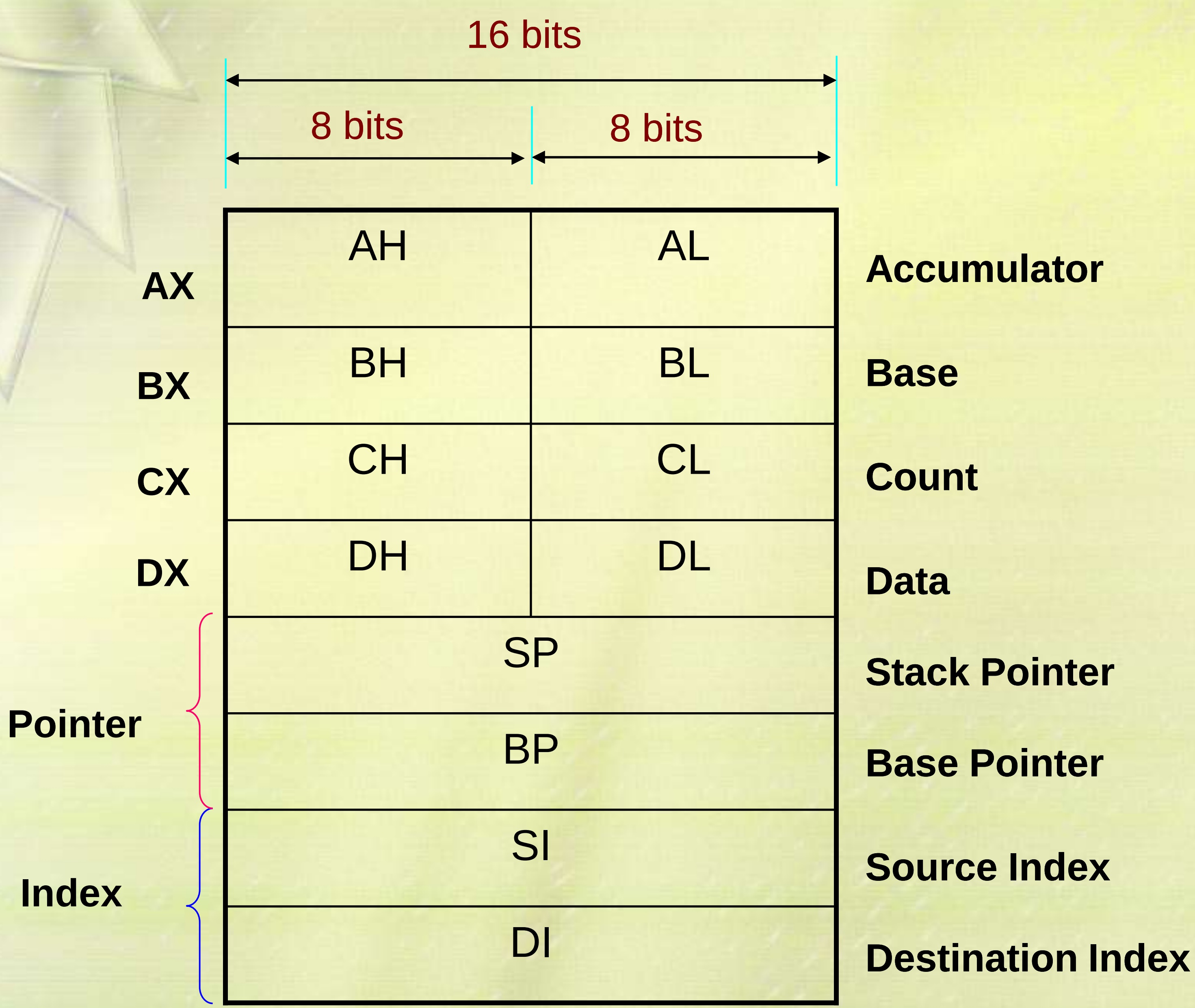
- Decodes instructions fetched by the BIU
- Generate control signals,
- Executes instructions.



The main parts are:

- Timing and Control Circuit
- Decoding circuit
- ALU

EXECUTION UNIT – General Purpose Registers



16 bits

8 bits | 8 bits

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

EXECUTION UNIT – 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)

Pointer And Index Registers

- used to keep offset addresses.
- Used in various forms of memory addressing.
- In the case of SP and BP the default reference to form a physical address is the Stack Segment.
- 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

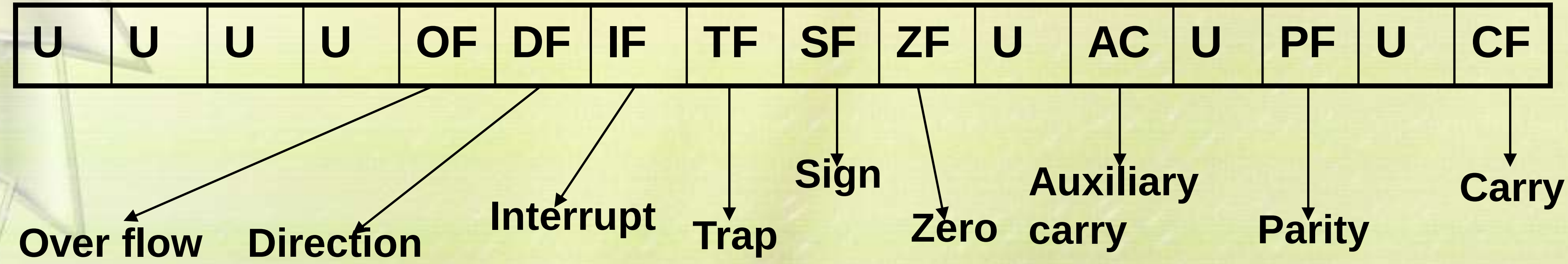
- SI: Source Index register
  - is required for some string operations
  - When string operations are performed, the SI register points to memory locations in the data segment which is addressed by the DS register. Thus, SI is associated with the DS in string operations.

- 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 extra 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

EXECUTION UNIT – 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



U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AC	U	PF	U	CF

Over flow

Direction

Interrupt

Trap

Sign

Zero

Auxiliary carry

Parity

Carry

U - Unused

EXECUTION UNIT – Flag Register

Flag	Purpose
Carry (CF)	Holds the carry after addition or the borrow after subtraction.
Parity (PF)	PF=0;odd parity, PF=1;even parity.
Auxiliary carry (AC)	Holds the carry (half – carry) after addition or borrow after subtraction between bit positions 3 and 4 of the result (for example, in BCD addition or subtraction.)
Zero (ZF)	Shows the result of the arithmetic or logic operation. Z=1; result is zero.
Sign (SF)	Holds the sign of the result after an arithmetic/logic instruction execution. S=1; negative.

Flag	Purpose
Trap (TF)	A control flag. If this flag is set, the processor enters the single step execution mode.
Interrupt (IF)	A control flag. Controls the operation of the INTR (interrupt request) IF=0; INTR pin disabled. IF=1; INTR pin enabled.
Direction (DF)	A control flag. It selects either the increment or decrement mode for DI and /or SI registers during the string instructions.
Overflow (OF)	Overflow occurs when signed numbers are added or subtracted. An overflow indicates if the result of a signed operation is large enough to be accommodated in a destination register.

Execution unit – 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

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

ZF = 0; the result is not zero

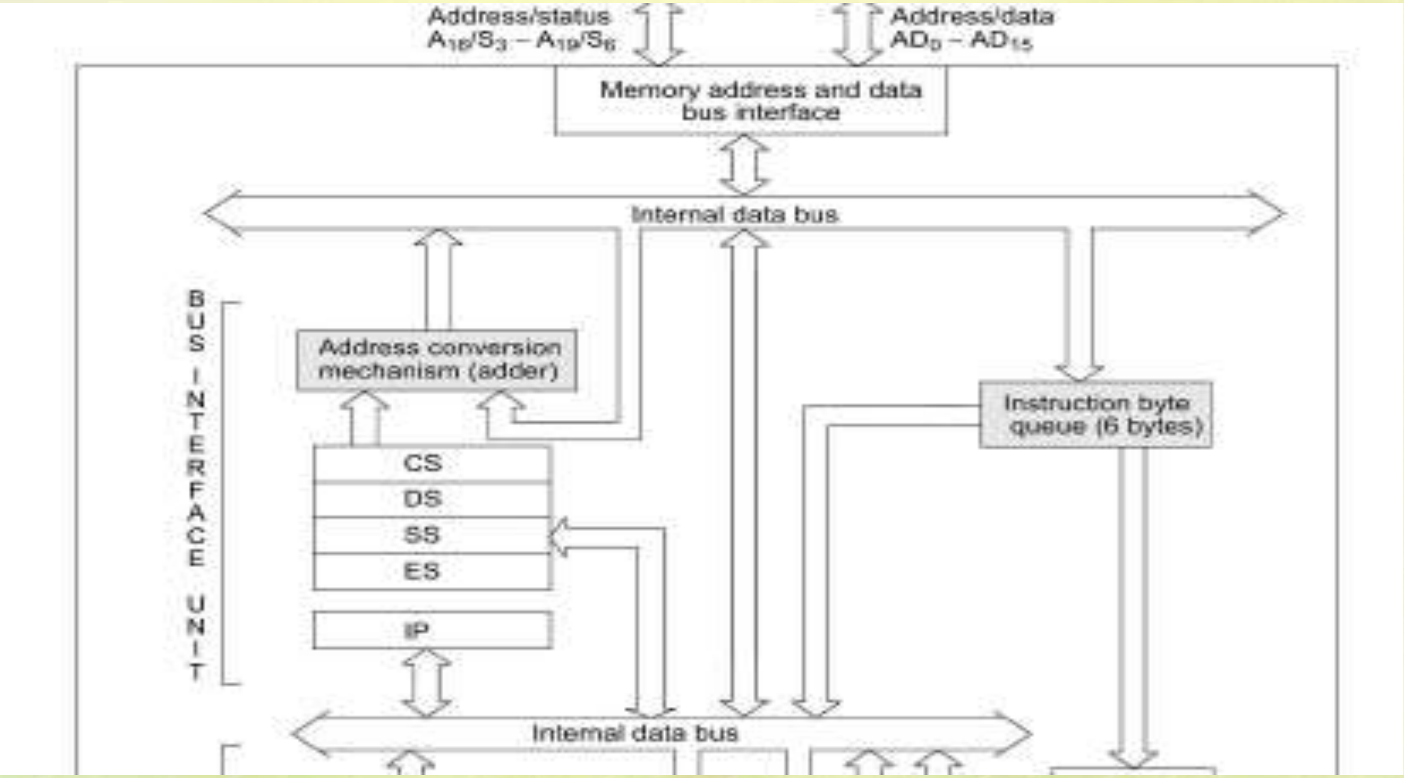
SF = 1; bit seven is one

OF = 1; the sign bit has changed

BUS INTERFACE UNIT (BIU)

Contains

- **6-byte Instruction Queue**
- **The Segment Registers (CS, DS, ES, SS).**
- **The Instruction Pointer (IP).**
- **The Address conversion mechanism (adder)**


Address/status
$A_{16}/S_3 - A_{19}/S_6$
Address/data
$AD_0 - AD_{15}$
Memory address and data bus interface
Internal data bus
BUS INTERFACE UNIT
Address conversion mechanism (adder)
Instruction byte queue (6 bytes)
CS
DS
SS
ES
IP
Internal data bus

The Queue Operation

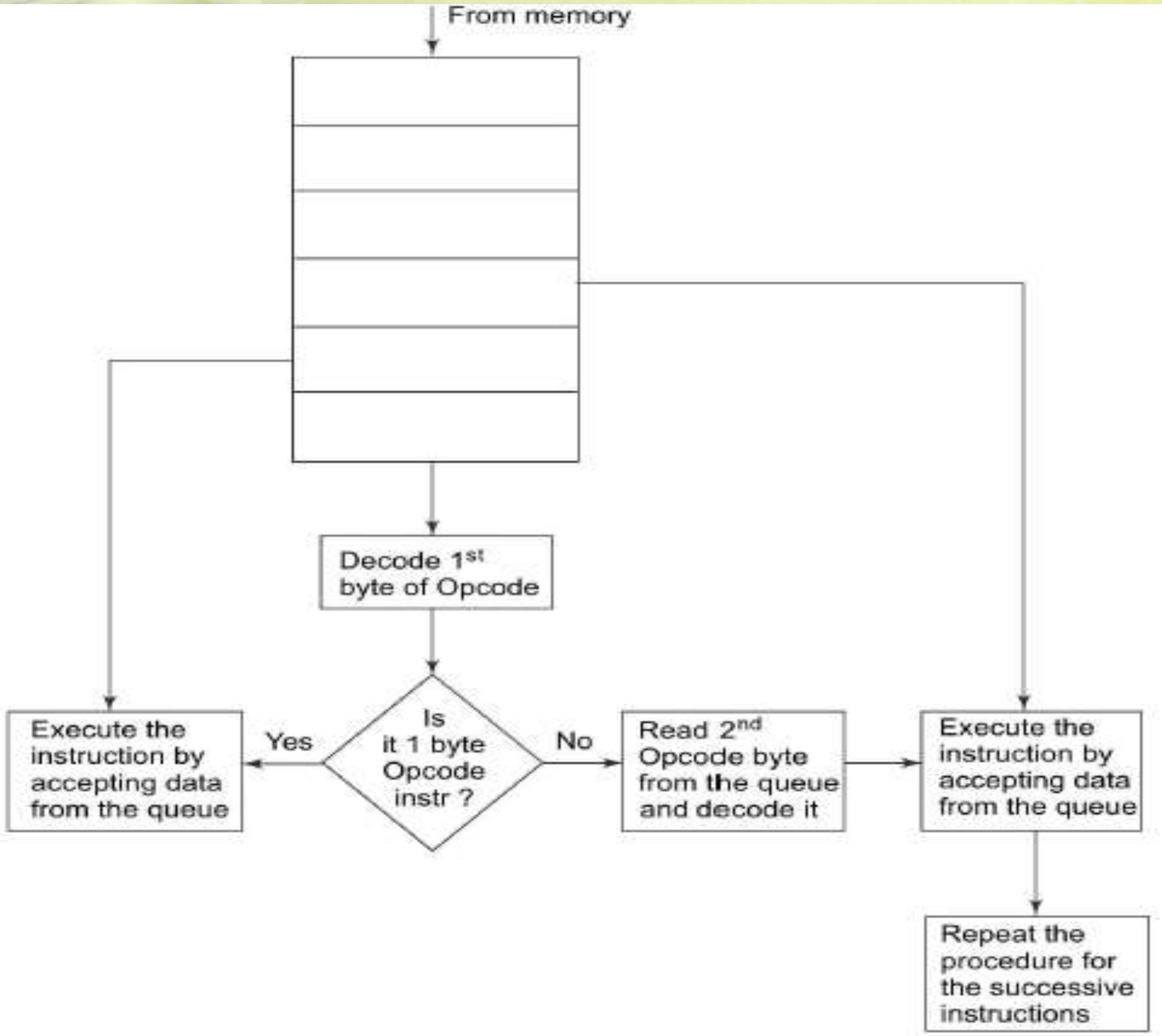
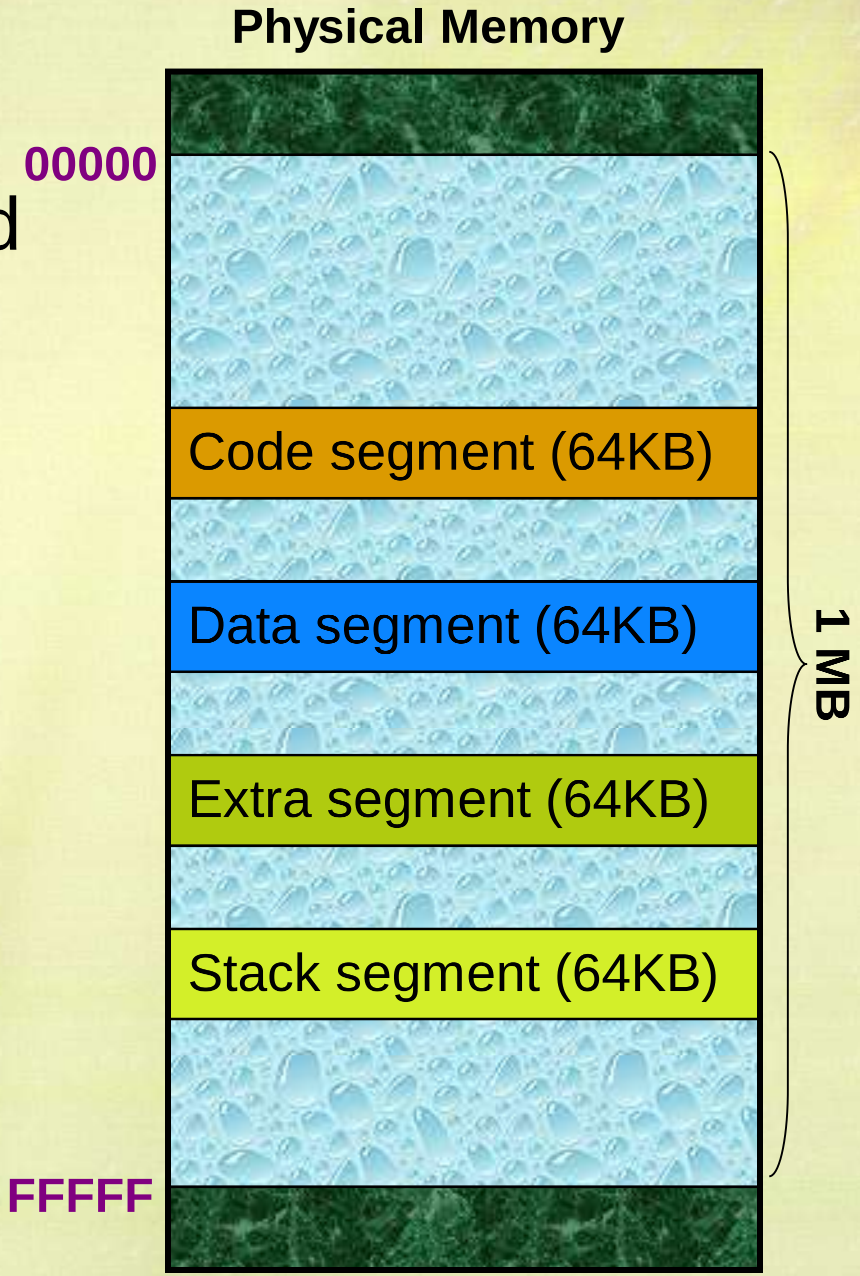

From memory
Decode 1st byte of Opcode
Is it 1 byte Opcode instr ?
Yes
No
Execute the instruction by accepting data from the queue
Read 2nd Opcode byte from the queue and decode it
Execute the instruction by accepting data from the queue
Repeat the procedure for the successive instructions

Fig. 1.6 ***The Queue Operation***

Segmented Memory

- The memory in an 8086 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.



- 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.**

- 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

Advantages of Segmented memory Scheme

- 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 is done .

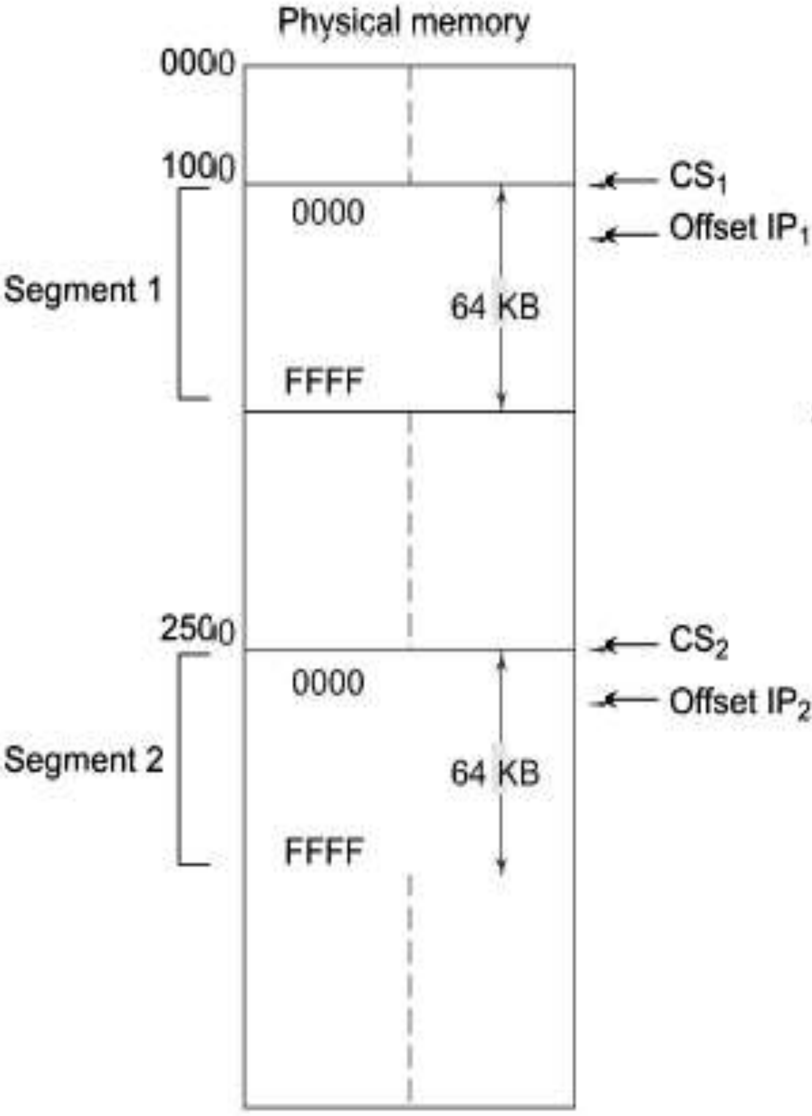


Fig. 1.3(a) *Non-overlapping Segments*

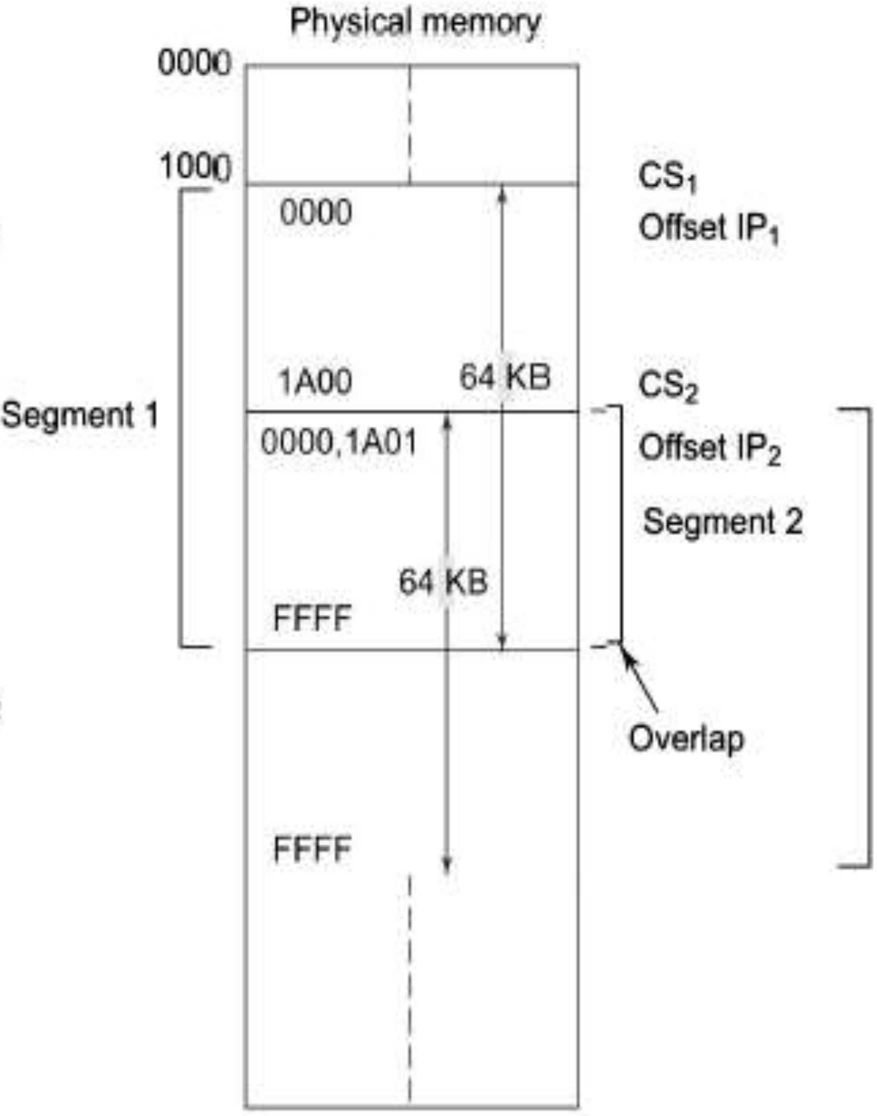
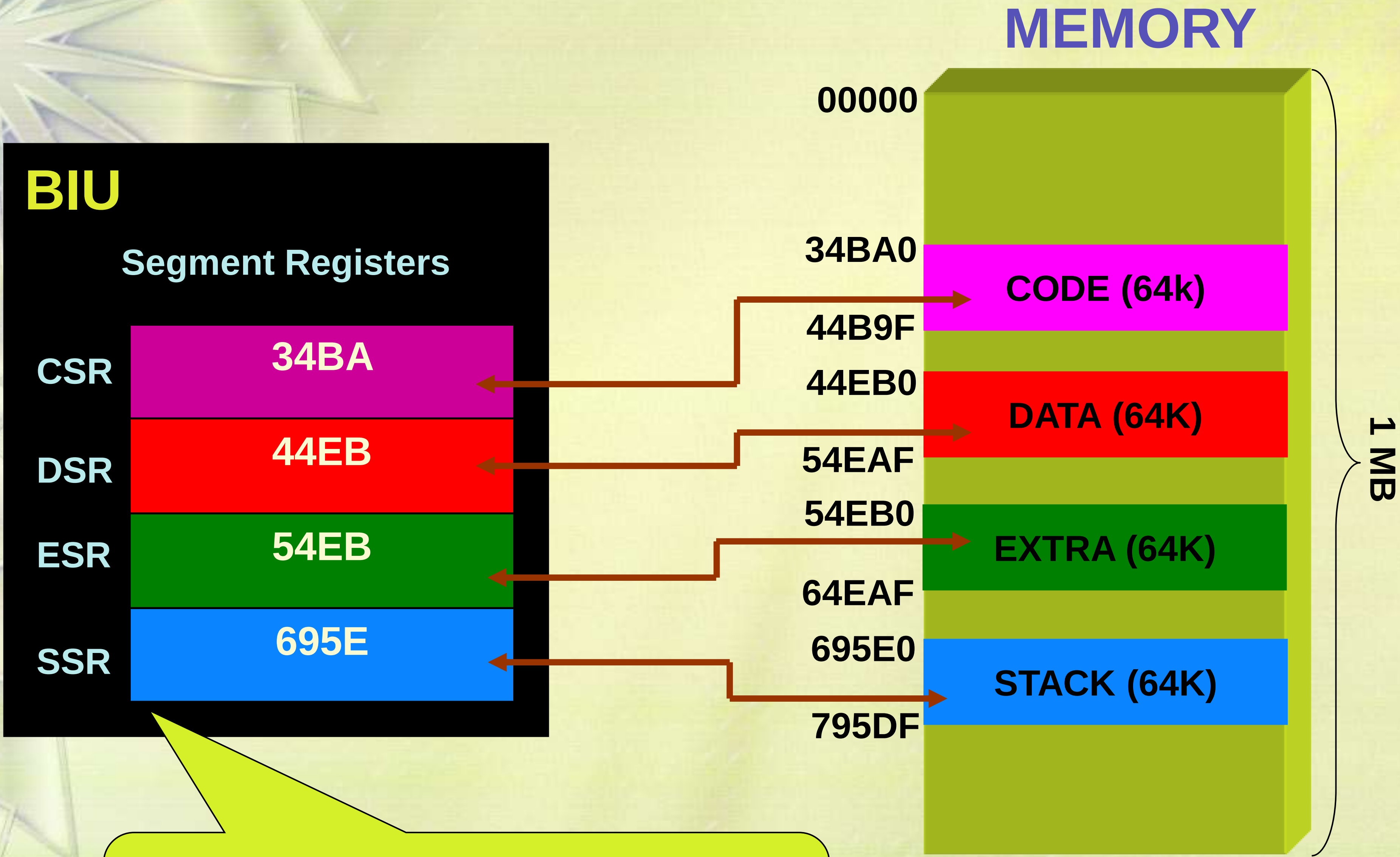


Fig. 1.3(b) *Overlapping Segments*

Segment registers

- 8086 processor have 4 segment registers
- Code Segment register (CS), Data Segment register (DS), Extra Segment register (ES) and Stack Segment (SS) register.
- All are 16 bit registers.
- Each of the Segment registers store the upper 16 bit address of the starting address of the corresponding segments.



MEMORY

BIU

Segment Registers

CSR	34BA
DSR	44EB
ESR	54EB
SSR	695E

00000	
34BA0	CODE (64k)
44B9F	
44EB0	DATA (64K)
54EAF	
54EB0	EXTRA (64K)
64EAF	
695E0	STACK (64K)
795DF	

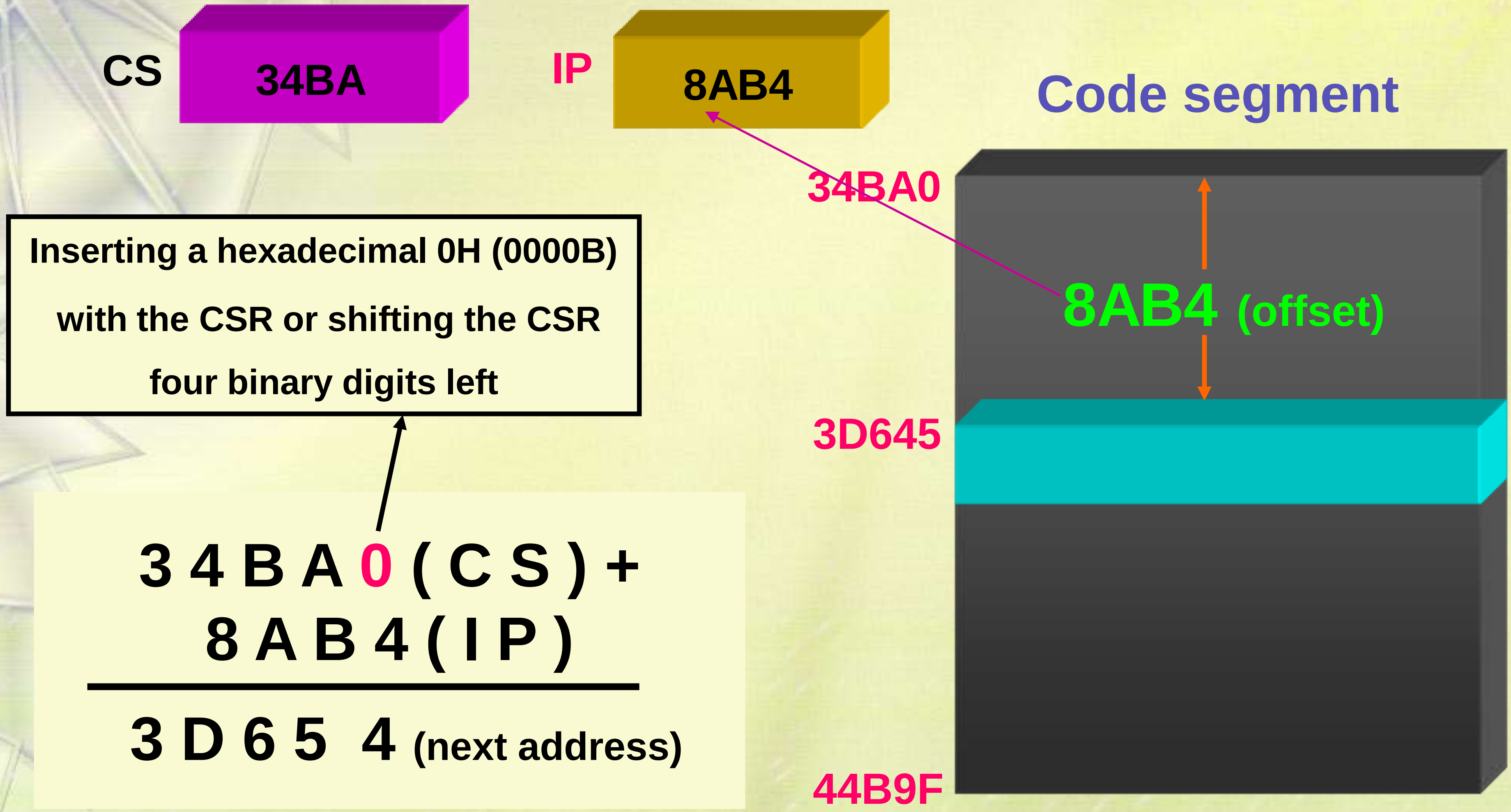
1 MB

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

Instruction pointer & 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.**
- **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.**

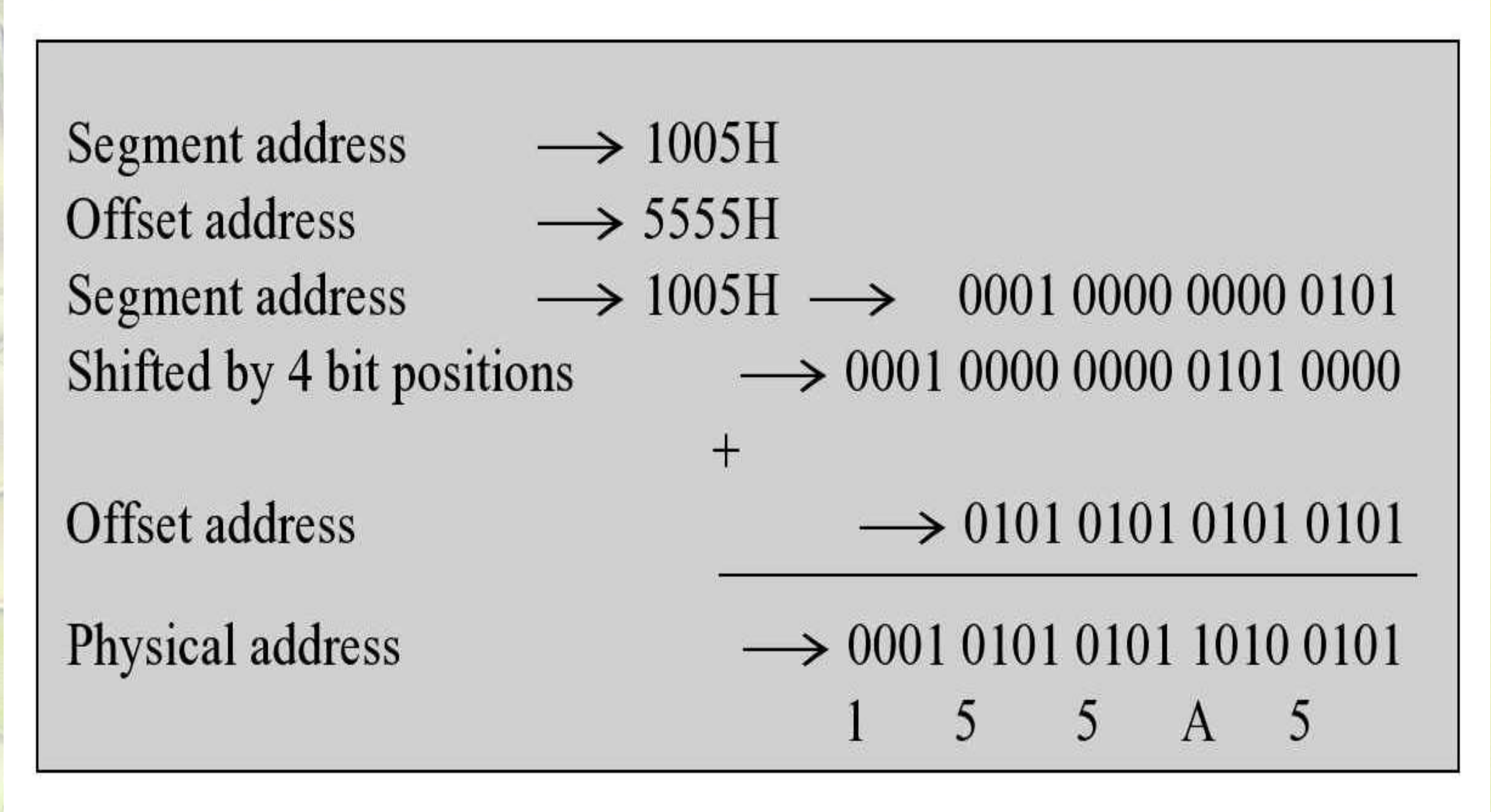
- **The following examples shows the CS:IP scheme of address formation:**



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

34BA0 (CS) +
8AB4 (IP)

3D654 (next address)

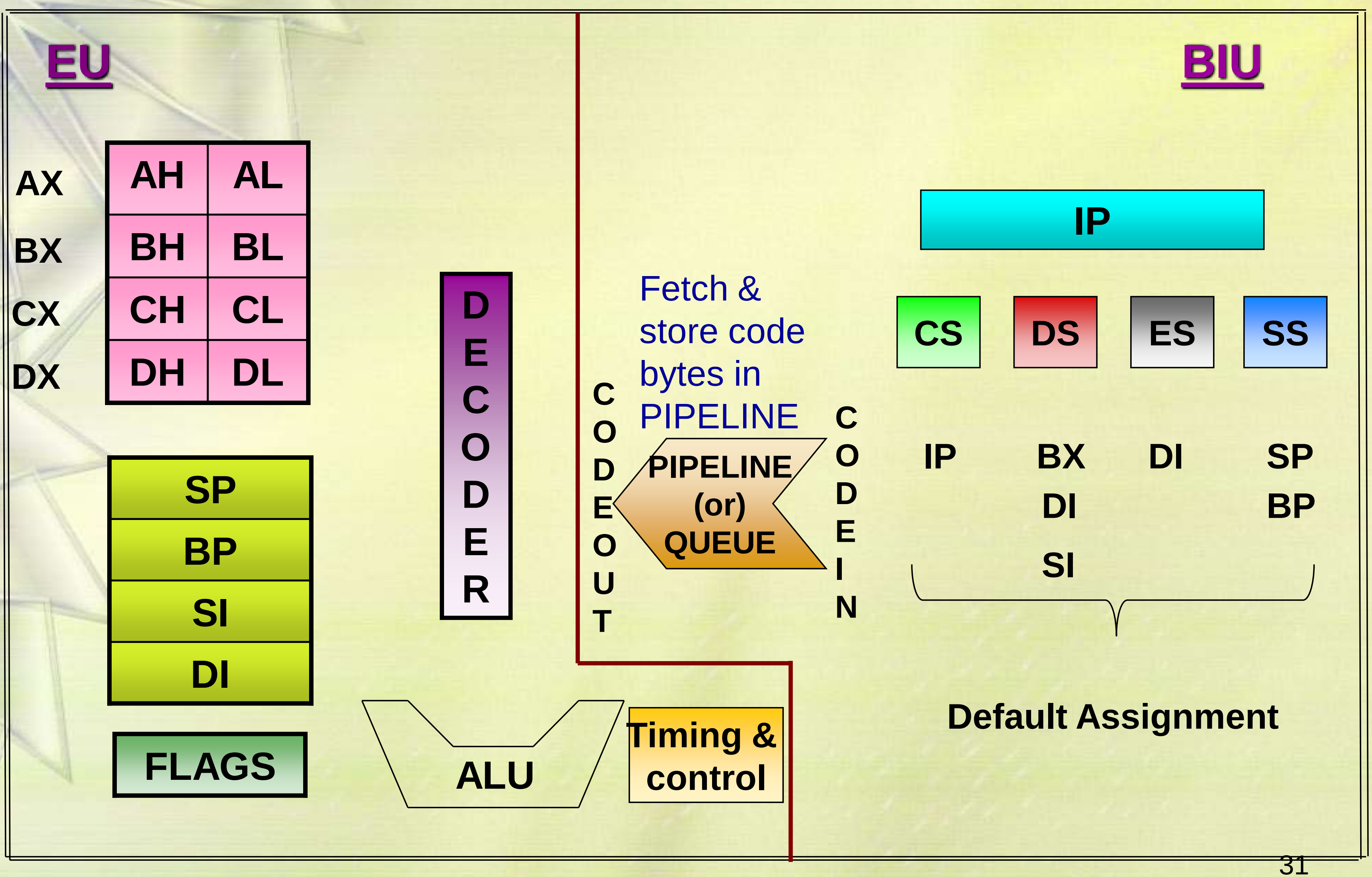


```
Segment address              → 1005H
Offset address               → 5555H
Segment address              → 1005H →      0001 0000 0000 0101
Shifted by 4 bit positions          → 0001 0000 0000 0101 0000
                                  +
Offset address                           → 0101 0101 0101 0101
                                 ------------------------------
Physical address                    → 0001 0101 0101 1010 0101
                                       1     5    5    A    5
```

Segment and Address register combination

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

Summary of Registers & Pipeline of 8086 µP



EU

AX AH AL

BX BH BL

CX CH CL

DX DH DL

SP

BP

SI

DI

FLAGS

DECODER

ALU

Timing & control

CODE OUT

Fetch & store code bytes in PIPELINE

PIPELINE (or) QUEUE

CODE IN

BIU

IP

CS DS ES SS

IP BX DI SI DI SP BP

Default Assignment