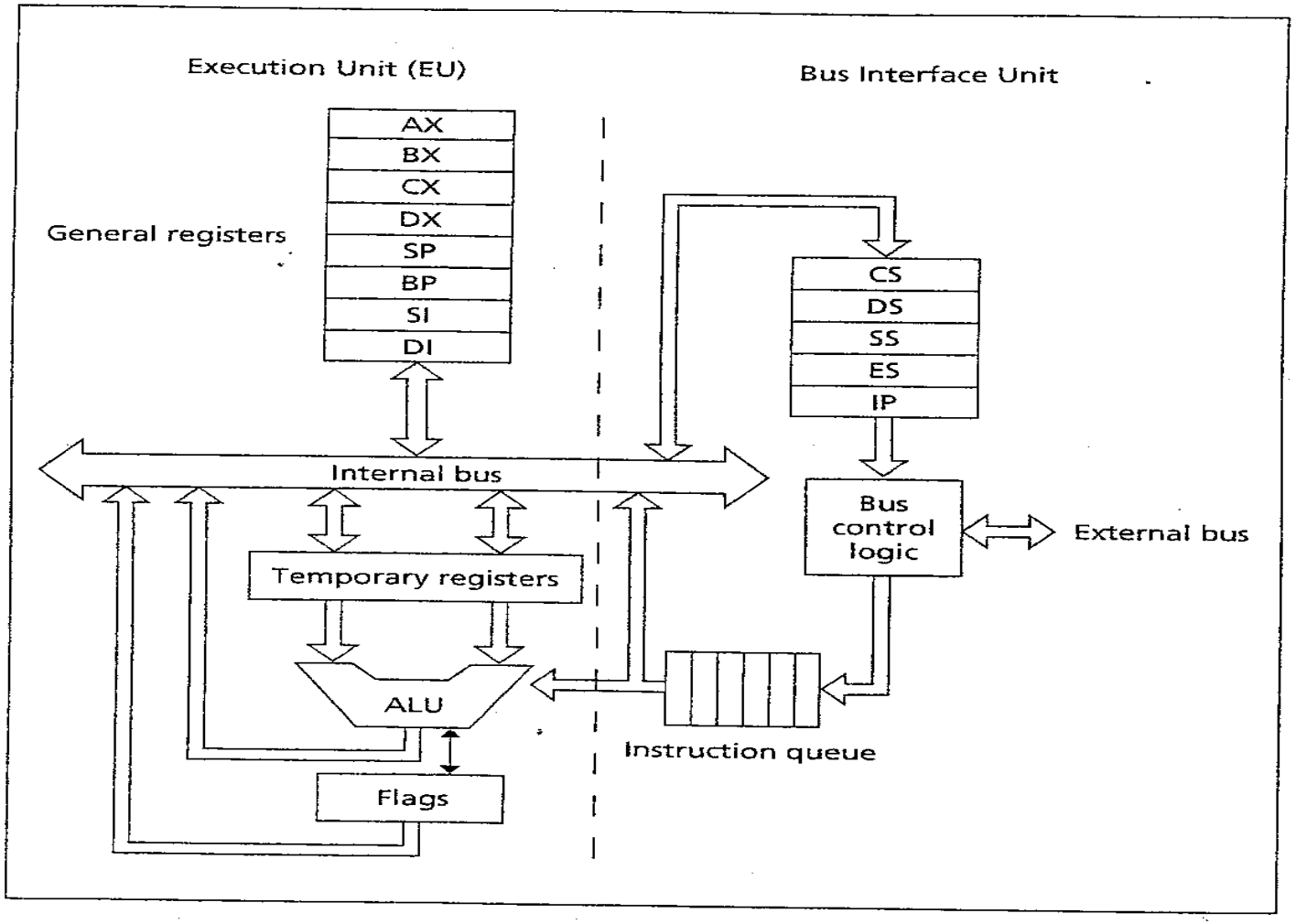
IBM-PC Organization

16-Bit Intel Processor Architecture

- A-16 bit microprocessor can operate on 16 bits of data at a time.
- 8086/8088 have the simplest structure
- 8086/8088 have the same instruction set, it forms the basic set of instructions for other Intel families.

Organization of the 8088/8086



Organization of the 8088/8086

2 main components:

- **Execution Unit (EU).**
- **Bus Interface Unit (BIU).**

EU: ALU + Registers (AX, BX, CX, DX, SI, DI, BP, and SP) + FLAGS register.

ALU: performs arithmetic & logic operations.

Registers: store data

FLAGS register: Individual bits reflect the result of a computation.

Organization of the 8088/8086

BIU: facilitates communication between the EU & the memory or I/O circuits.

Responsible for transmitting addresses, data, and control signals on the buses.

Registers (CS, DS, ES, SS, and IP) hold addresses of memory locations.

IP (instruction pointer) contain the address of the next instruction to be executed by the EU.

Organization of the 8088/8086

16-bit registers, 1M Bytes Memory

Registers:

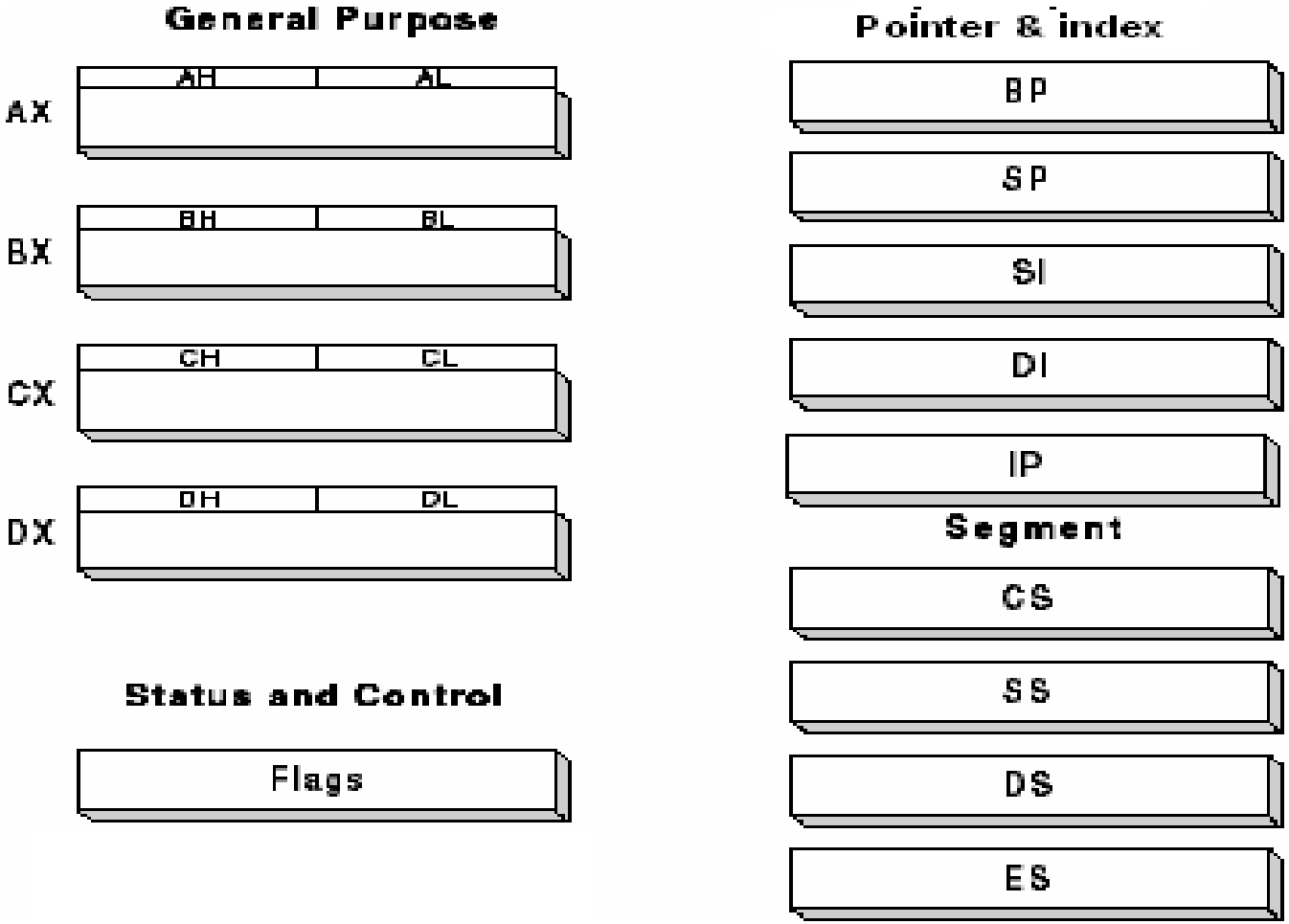
Information is stored in registers

Registers are classified according to the functions they perform

Registers

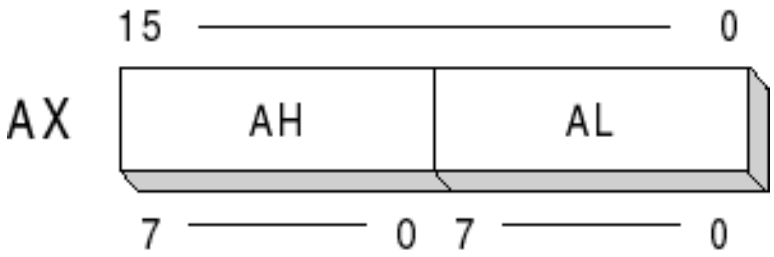
- **Data registers**: 4 general data registers hold data for an operation.
- **Address registers**: (segment, pointer and index registers) hold the address of an instruction or data.
- **Status register**: FLAG register keeps the current states of the processor.
- **14 16-bit registers**

Register


General Purpose
AX
AH
AL
BX
BH
BL
CX
CH
CL
DX
DH
DL
Status and Control
Flags
Pointer & Index
BP
SP
SI
DI
IP
Segment
CS
SS
DS
ES

General Data Register: Used for general data manipulation.

- They are 16-bit registers that can also be used as two 8 bit registers: low and high bytes can be accessed separately → more registers to use when dealing with byte-size data.
- In addition to being general-purpose registers, they perform special functions



AX (Accumulator)

- **Most efficient register for arithmetic, logic operations and data transfer: the use of AX generates the shortest machine code.**
- **In multiplication and division operations, one of the numbers involved must be in Al or AX**

BX (Base)

Can hold addresses (offset)

CX (Counter)

Counter for looping operations: loop counter, in REP instruction, and in the shift and rotate bits

DX (Data):

Used in multiply and divide, also used in I/O operations

The 8086 processor

The 8086 processor assign a 20-bit physical address to its memory locations.

2^{20} → 1 Mbytes

20 bits → 5 hex digits

first addresses: 00000, 00001,…,0000A,…FFFFF.

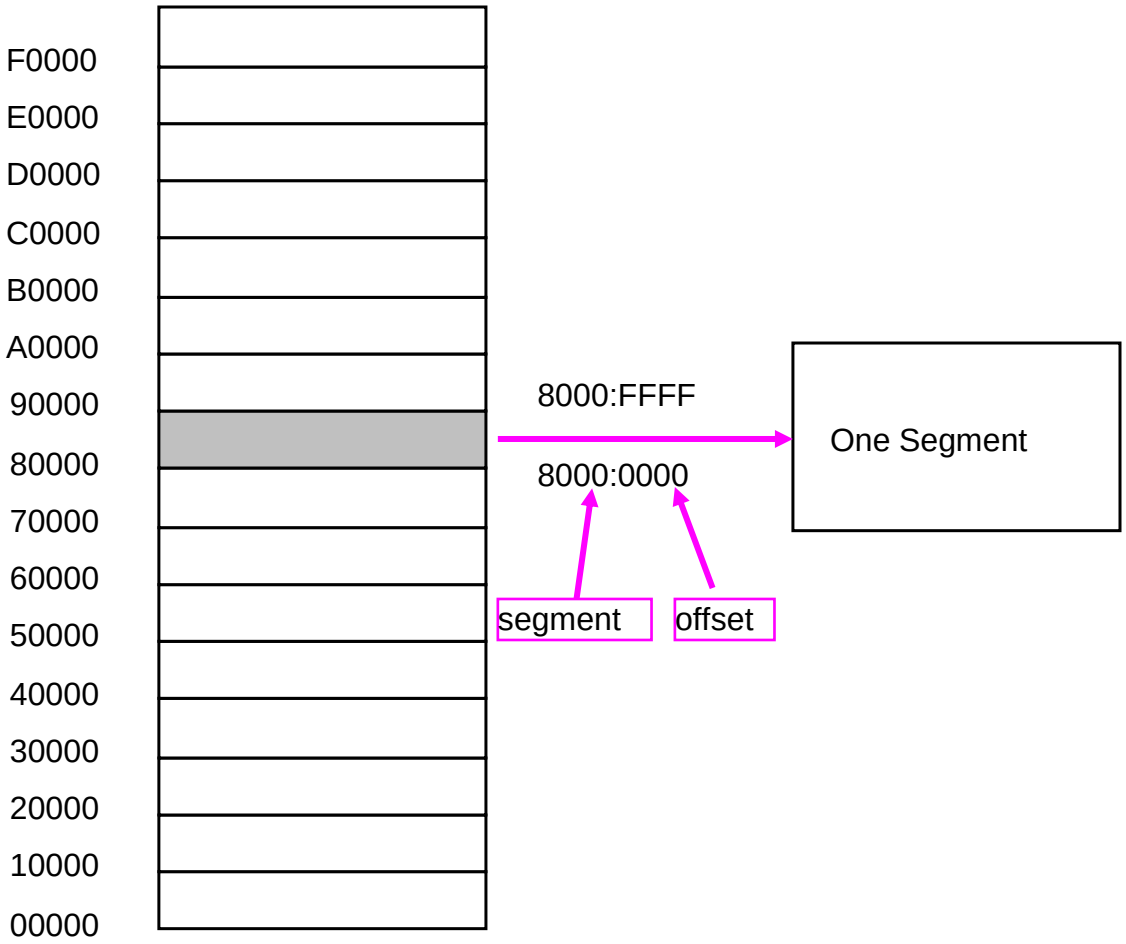
registers are 16-bits → can address only 2^{16} = 64 K Bytes.

→ Partition the memory into segments

Memory Segment

- Is a block of 2^{16} (64) K Bytes consecutive memory bytes.
- Each segment is identified by a 16-bit number called **segment number**, starting with 0000 up to FFFFh . Segment registers hold segment number.
- Within a segment, a memory location is specified by giving an **offset** (16-bit) = It is the number of bytes from the beginning of the segment (0→ FFFFh).

Memory Segment



Segment : Offset Address

- A memory location may be specified by a segment number and offset (logical address).

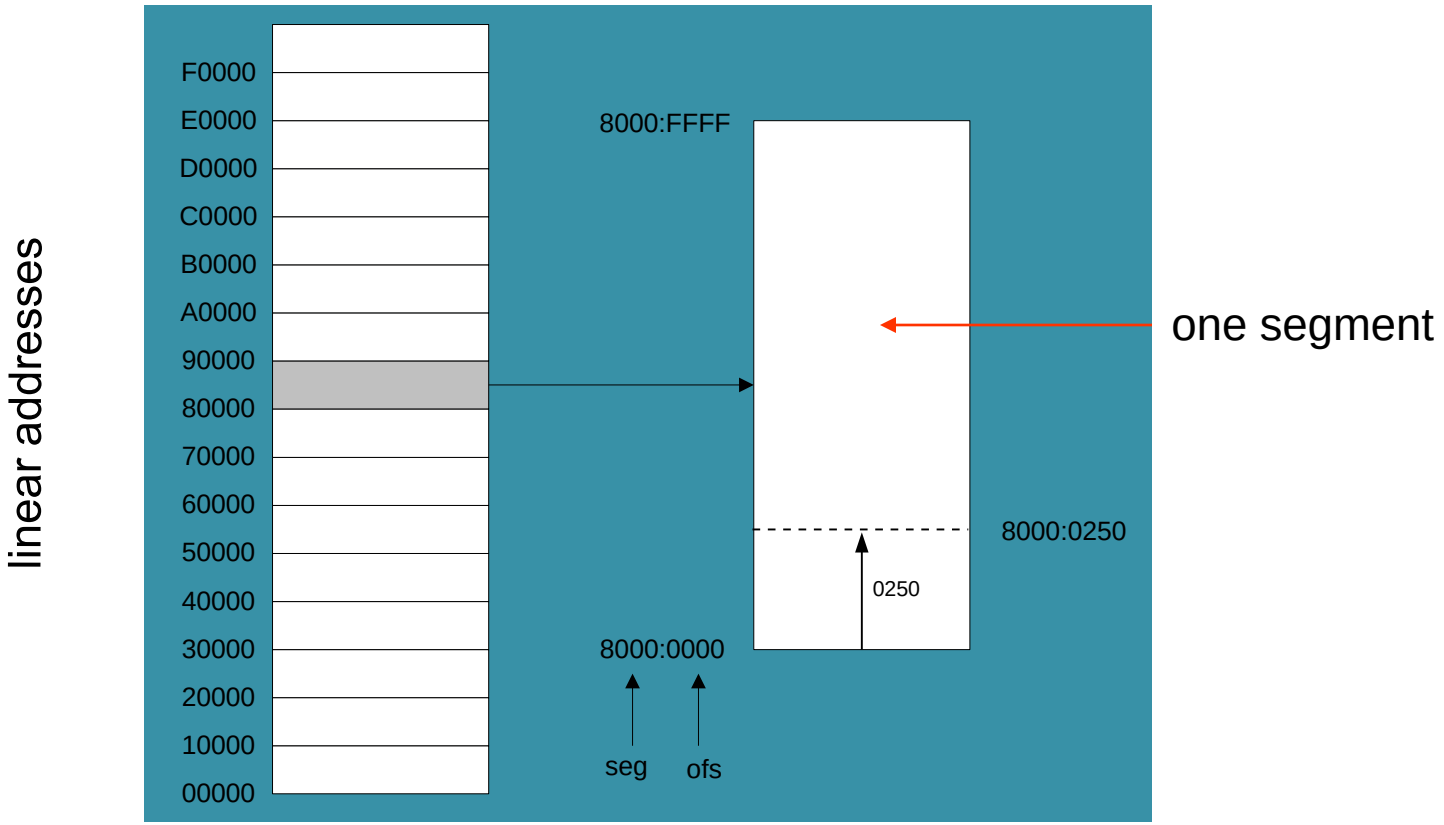
Example :

A4FB : 4872 h

- Offset： is the distance from the beginning to a particular location in the segment.

- Segment number：defines the starting of the segment within the memory space.

Segmented Memory



Start location of the segment must be 20 bits → the absolute address is obtained by appending a hexadecimal zero to the segment number , i.e. **multiplying by 16(10_h).**

Physical Address

Physical Address : is equal to

segment number X 10_h + Offset

Physical Address for A4FB : 4872

A4FB0

+

4872

A9822 **(20 bits)**

Location of Segments

Segment 0

starts at address 0000:0000 → 00000 h

ends at address 0000:FFFF → 0FFFF h

Location of Segments

Segment 1

starts at address 0001:0000 → 00010 h

ends at address 0001:FFFF → 1000F h

Overlapping between segments

Location of Segments

- The segments start every 10 = 16 bytes **(called Paragraph)** and the starting address of a segment always ends with a hex digit 0.
- **Paragraph boundary** is an address divisible by 16.

Solution

a) **Segment 1256 :**

offset = 1256A - 12560 = A

Address → **1256 : 000A**

b) **Segment 1240 :**

offset = 1256A - 12400 = 0016A

Address → **1240 : 016A**

Program Segments

- A typical machine language program consists of:
  - **instructions (CODES)**
  - **data**
  - **stack → is a data structure used by the processor to implement procedure calls.**

- Codes , data , and stack are loaded into different memory segments :
  - **Code segment CS : holds segment number of the code segment.**
  - **Data Segment DS : holds segment number of the data segment.**
  - **Extra Segment ES :extra segment : holds alternate segment number of the data segment.**
  - **Stack Segment SS : holds segment number of the stack segment.**

Program Segment

- **A program segment can occupy less than 64 Kbytes.**

- **Overlapping permits program segments that are less than 64 KB to be placed close together.**

- At any time, only those memory locations addressed by the 4 segment registers are accessible; → only 4 memory segments are active. However, the contents of a segment register can be modified by a program to address different segments.

Pointer and Index Registers SP, BP, SI, DI

- **Used for offset of data, often used as pointers. Unlike segment registers, they can be used in arithmetic and other operations.**

Pointer Registers

- **SP (Stack Pointer)**: Used with SS for accessing the stack segment.
- **BP (Base Pointer)**: Used with SS to access data on the stack. However, unlike SP, BP can be used to access data in other segments.

Index Registers

- **SI (Source Index):** Source of string operations. Used with DS (or ES).
- **DI (Destination Index):** Destination of string operation. Used with ES (or DS).

Instruction pointer

IP (Instruction pointer):

Points to the next instruction.

Used with CS.

Flags register

Flags: Bits specify status of CPU and information about the results of the arithmetic operations.

Bit Position

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	x	x	O	D	I	T	S	Z	x	A	x	P	x	C

O = Overflow
D = Direction
I = Interrupt
T = Trap
x = undefined

S = Sign
Z = Zero
A = Auxiliary Carry
P = Parity
C = Carry

Organization of the PC

- **A computer is made of: Hardware & software. Software controls the H/W operations.**
- **The purpose of the OS is to coordinate the operations of all the devices that make up the computer systems.**

Some of the OS functions

1) reading and executing the commands typed by the user.
2) performing I/O operations
3) generating error messages
4) managing memory and other resources.

Very popular O.S. for IBM PC is DOS.

- **DOS** manage only 1 M byte memory, does not support multitasking.
- **DOS** is a collection of routines that coordinates the operations of the computer. The routine that executes user command is COMMAND.COM.
- Information stored on disk is organized into **files**. A file has a name and an optional extension.

- The **BIOS** routines are used to perform I/O operations.
- **DOS** routines operate over the entire PC family.
- **BIOS** routines are machine specific.
- The compatibility of PC clones with the IBM PC depends on how well their **BIOS** routines match those of the IBM PC
- The addresses of **BIOS** routines (interrupt vectors) are placed in memory starting at 00000h.

I/O Ports Addresses

- I/O devices are connected to the computer through I/O circuits. Each of them contains several registers called ports.
- I/O ports have addresses ⟶ I/O addresses .
- 8086/8088 supports 64 KB of I/O ports. Example: keyboard controller: 60^{h} - 63^{h}

Start-up operation

- When PC is powered on CS is set to FFFFh & IP is set to 0000h. → PC executes the instruction with the address FFFF0h. This instruction transfers the control to the BIOS routines.
- BIOS loads the boot program.
- Boot program loads the OS and COMMAND.COM is given control