UNIT - II:

CPU-Organization:

- 8086 CPU Block diagram
- 8086 Pin diagram,
- concept of pipelining,
- Minimum and Maximum mode,
- Segment register and generation of 20 bits address,
- Concept of Address, Data, control and Systems bus,
- Types of flags.

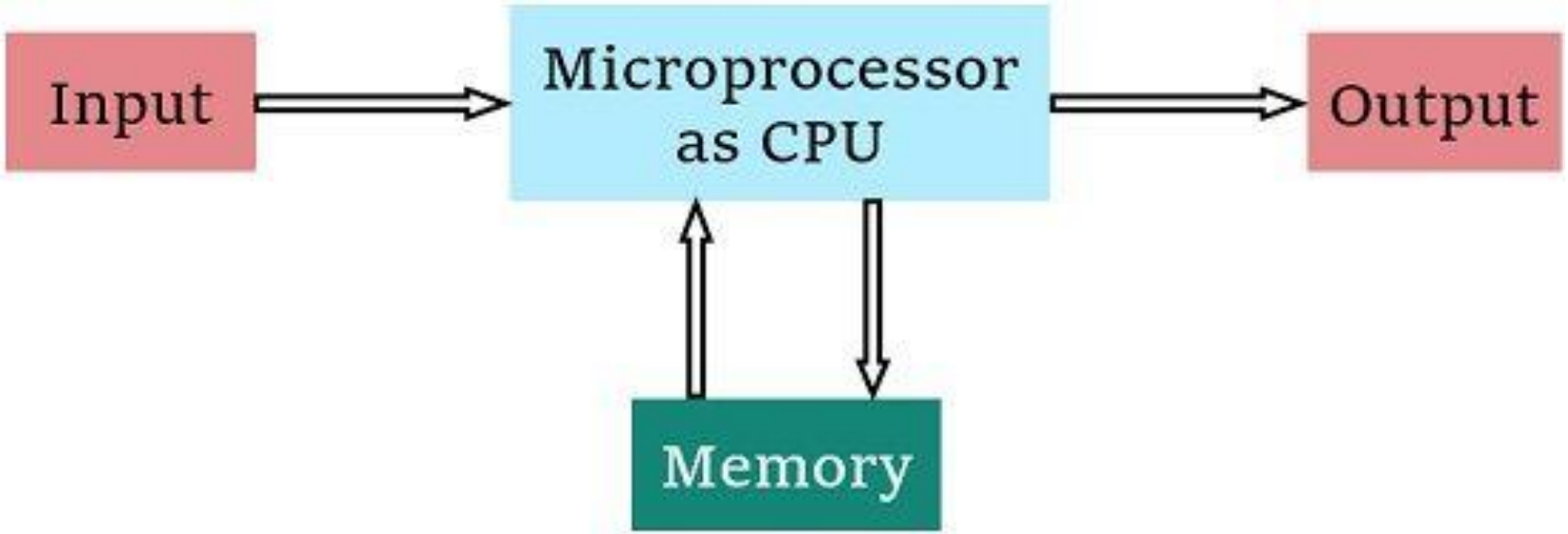
Text Book: Microprocessors and interfacing-Douglas V Hall, TMHE

Introduction

- 8086 is a 16-bit microprocessor.(Starting processor)
  - having 16-bit register set(can store 2^{16} different values).
  - have an ALU 16-bit wide
  - have16-bit bus
  - perform AL operations with 16-bit data in one cycle.
- Designed in 1978 by Intel.
- 8086 microprocessor has **20-bit address bus**.
  - able to access 2^{20} i.e., 1 MB address in the memory.
- the size of the data bus is 16-bit
  - it can carry 16-bit data at a time.

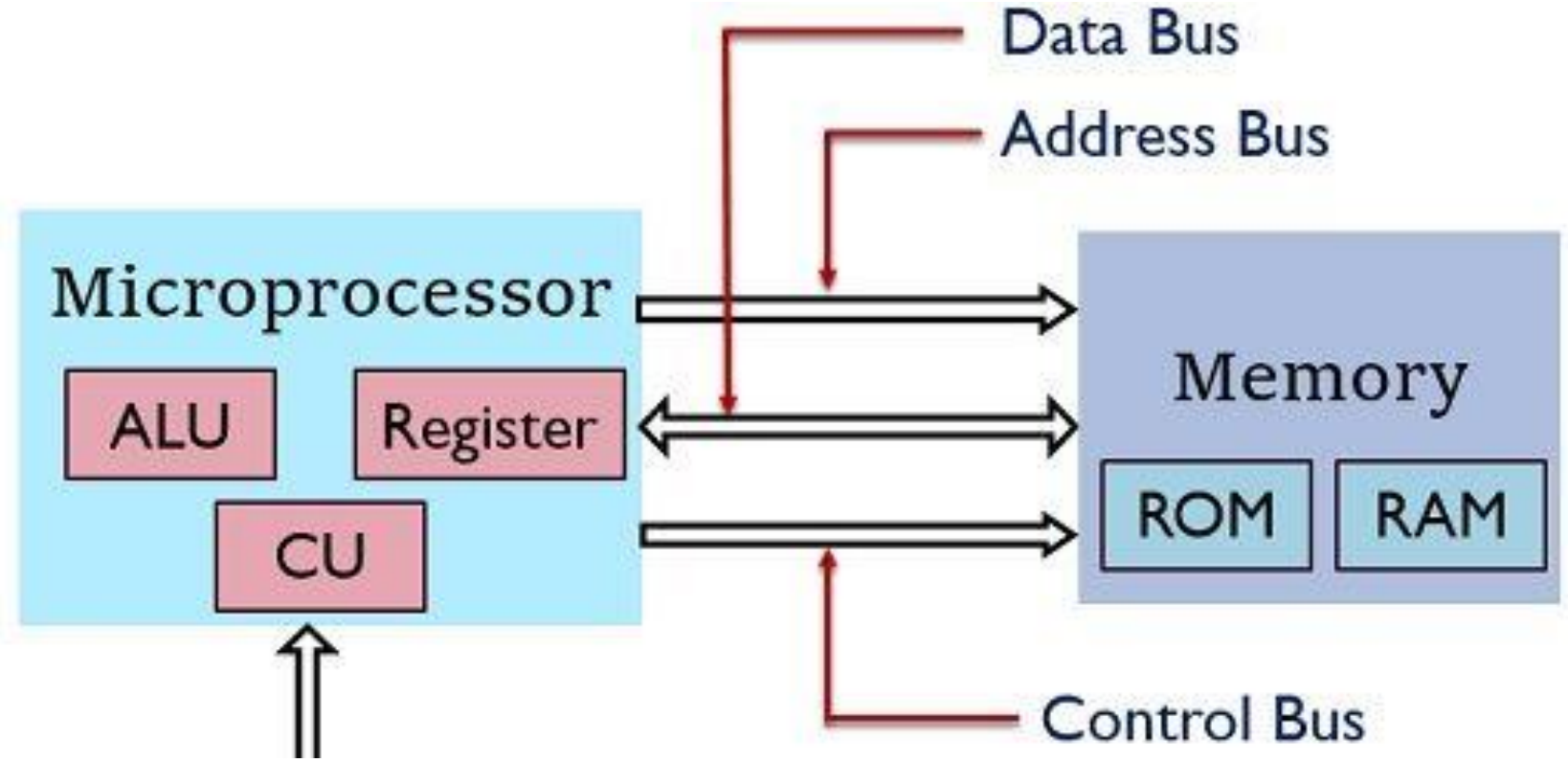
Microprocessor

A microprocessor is an IC that **executes arithmetic and logical functions** that are defined by a program inside the system. (Fetech→decode→execute)



Block Diagram of Computer with microprocessor as CPU

Working of Microprocessor

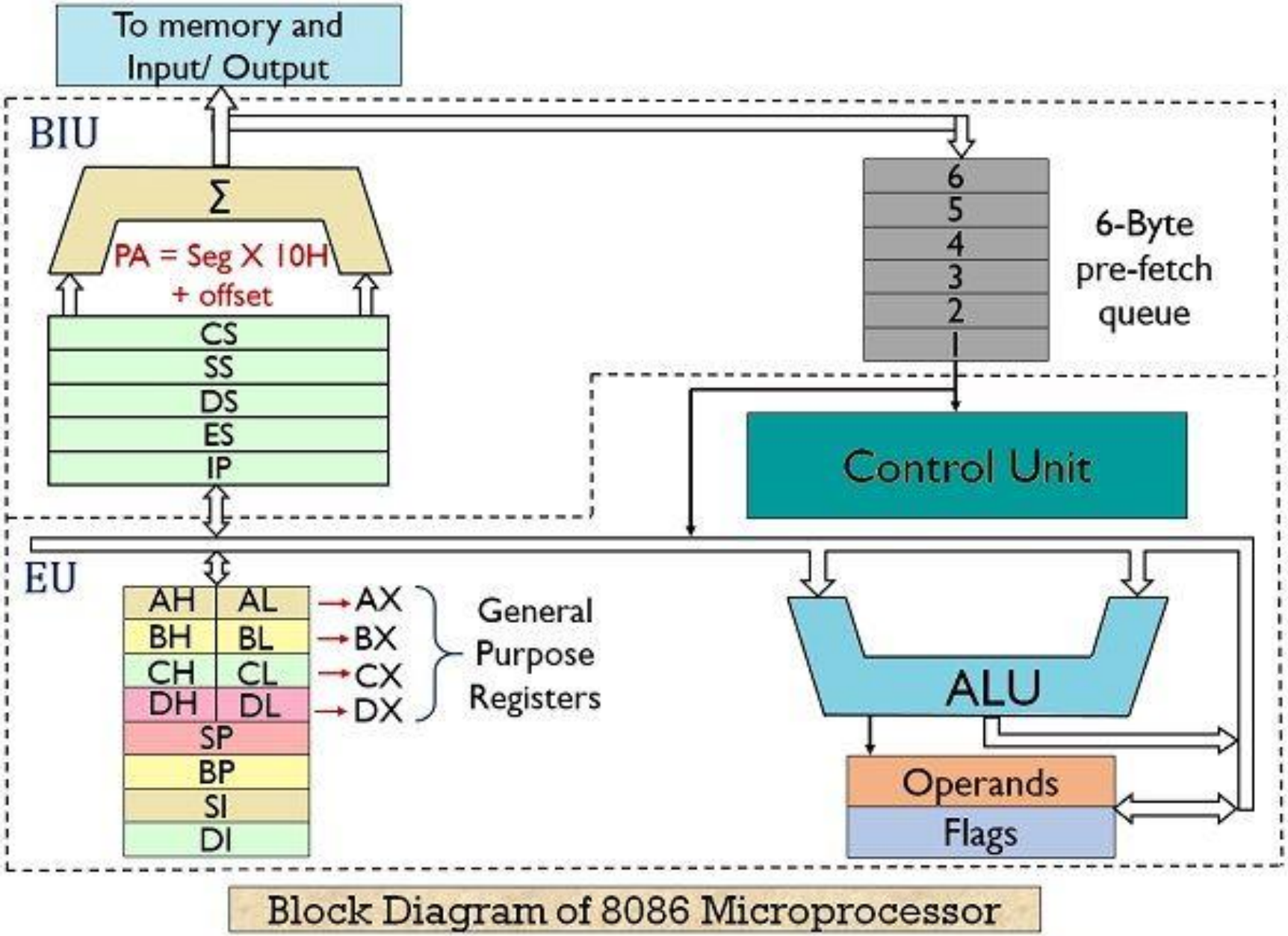


Applications of Microprocessor:

It finds numerous applications such as
in computer systems, in mobile phones, in remote controlling devices.
As well as in climate controlling and security applications etc.


D8086
L30936Ø7
INTEL ©1978

8086 – CPU – Block Diagram



Block Diagram of 8086 Microprocessor

8086 – Internal Architecture

The internal architecture of 8086 microprocessor is divided into 2 major parts, the **BIU** and **EU**.

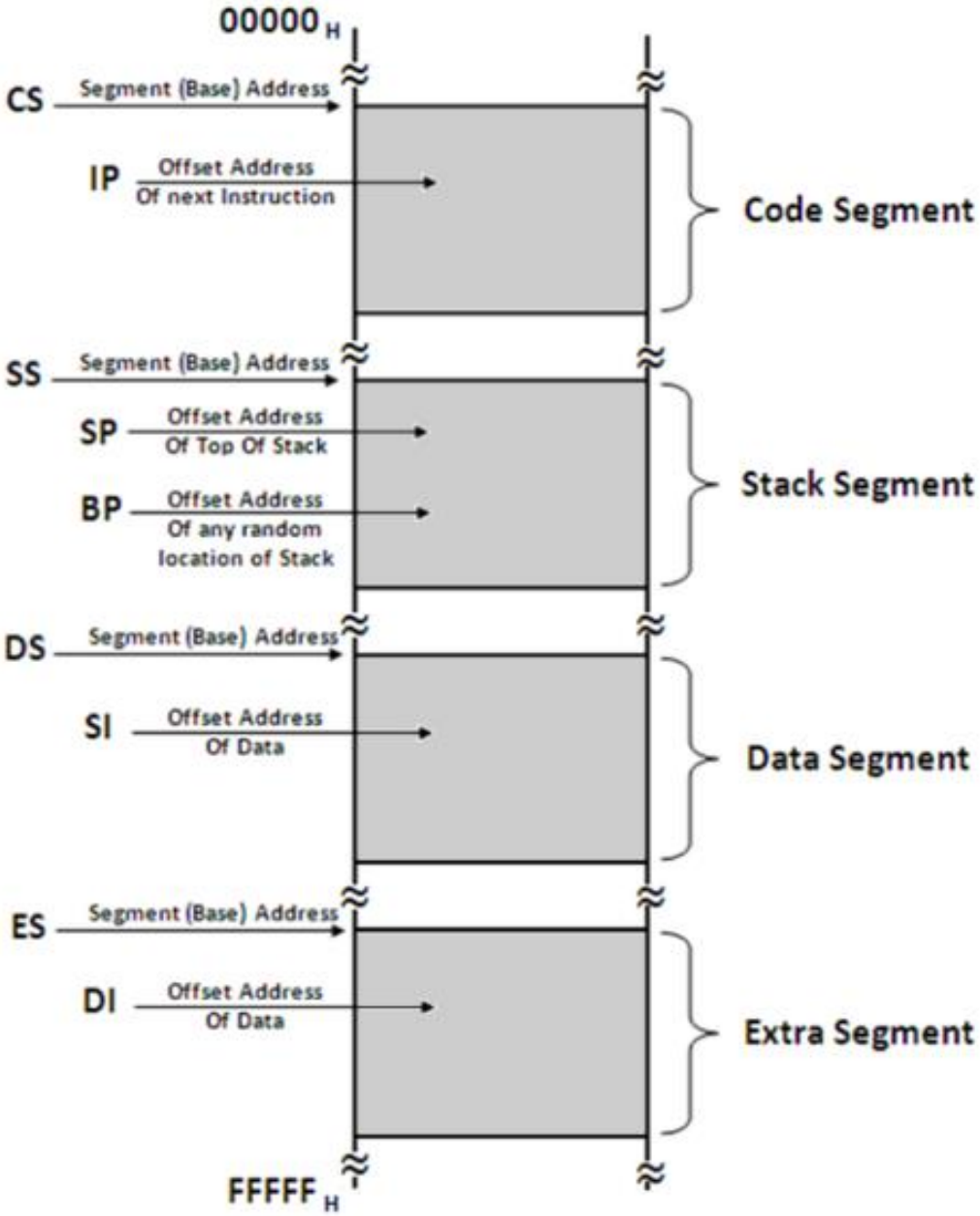
- **BIU**-Bus Interface Unit
  - Segment Registers
    - CS, DS, ES, SS, IP
  - Queue
- Memory Addressing
  - Segment Base
  - Offset

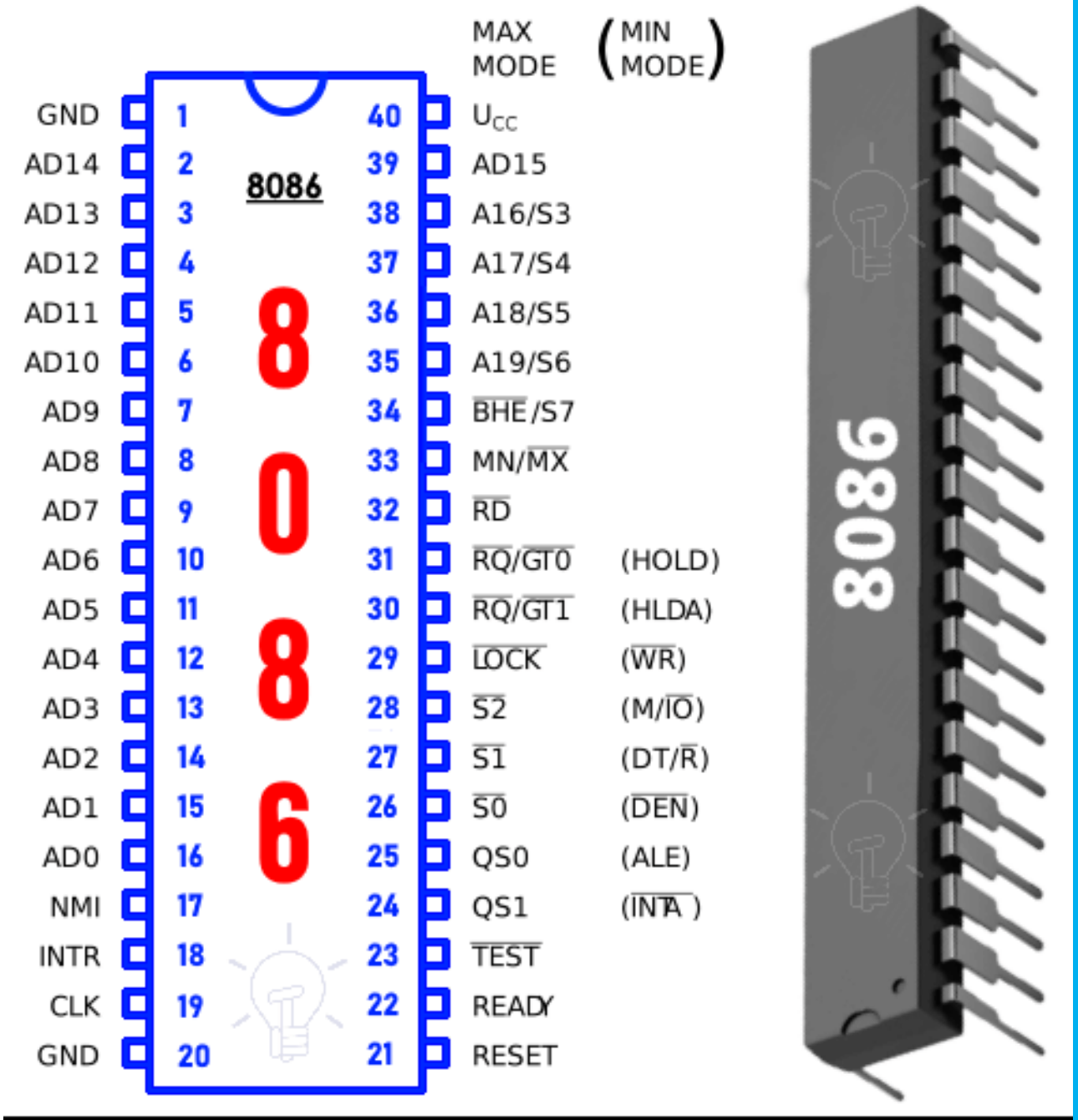
- **EU** - Execution Unit
  - General Purpose Registers
  - Pointer Registers
  - Index Registers
  - ALU
  - Flags

The BIU functions…

- BIU: **B**us **I**nterface **U**nit
- Fetches the sequenced instructions from the memory.
- Finds the physical address(PA) of that location in the memory where the instruction is stored.
- Manages the 6-byte pre-fetch queue where the pipelined instructions are stored.
- The BIU contains 4 segment registers.

MEMORY SEGMENTATION IN 8086





Pin Diagram of 8086 Microprocessor

1.First 16 bit mp.
2.available in 40pin DIP Chip.
3.each pin has got its own significance.
4.5V DC power supply for its operation.
5.has a 16-line data bus and 20-line address bus.
6. Operate in Multiplexed mode.

QS1,QS0 : Queue Status. These signals indicate the status of the internal 8086 instruction queue according to the table shown below

QS1	QS0	Status
0	0	No operation
0	1	First byte of op code from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

S2, S1, S0 : Status pins. These pins are active during T4, T1 and T2 states and is returned to passive state (1,1,1 during T3 or Tw (when ready is inactive). These are used by the 8288 bus controller for generating all the memory and I/O operation) access control signals. Any change in S2, S1, S0 during T4 indicates the beginning of a bus cycle.

S2	S1	S0	Characteristics
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive state

A16/S3, A17/S4, A18/S5, A19/S6 : The specified address lines are multiplexed with corresponding status signals.

A17/S4	A16/S3	Function
0	0	Extra segment access
0	1	Stack segment access
1	0	Code segment access
1	1	Data segment access

Concept of pipelining

- The process of fetching the next instruction when the present instruction is being executed is called as **pipelining**.
- **Pipelining** has become possible due to the use of queue.
  - BIU (Bus Interfacing Unit) fills in the queue until the entire queue is full.BIU restarts filling in the queue when at least two locations of queue are vacant.

When EU is busy in decoding and executing an instruction, the BIU fetches up to six instruction bytes for the next instructions. These bytes are called as the pre-fetched bytes and they are stored in a first in first out (FIFO) register set, which is called as a queue.

What is Pipelining?

- The greater performance of the CPU is achieved by instruction pipelining.
- 8086 microprocesor has two blocks

  BIU(BUS INTERFACE UNIT)

  EU(EXECUTION UNIT)

- 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.
- Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining.

Advantages of pipelining: **Performance improvement**

- The EU(execution unit)always reads the next instruction byte from the queue in BIU. This is faster than sending out an address to the memory and waiting for the next instruction byte to come.
- In short pipelining eliminates the waiting time of EU and speeds up the processing.
- The 8086 BIU will not initiate a fetch unless and until there are two empty bytes in its queue.
- 8086 BIU normally obtains two instruction bytes per fetch.

Disadvantages of pipelining:

- The execution time will generally be longer than the fetch time. Thus the fetch stage may have to wait for some time before it can empty the buffer.

- When conditional branch occurs, then the address of next instruction to be fetched become unknown. Then the execution stage have to wait while the next instruction is fetched.

- Conditional branch instructions

Assume that the instruction 3 is a conditional branch to instruction 15.

Until the instruction is executed there is no way of knowing which instruction will come next

The pipeline will simply loads the next instruction in the sequence and execute.

Branch is not determined until the end of time unit 7.

During time unit 8,instruction 15 enters into the pipeline.

No instruction complete during time units 9 through 12.

This is the performance penalty incurred because

General Architecture of Microprocessors

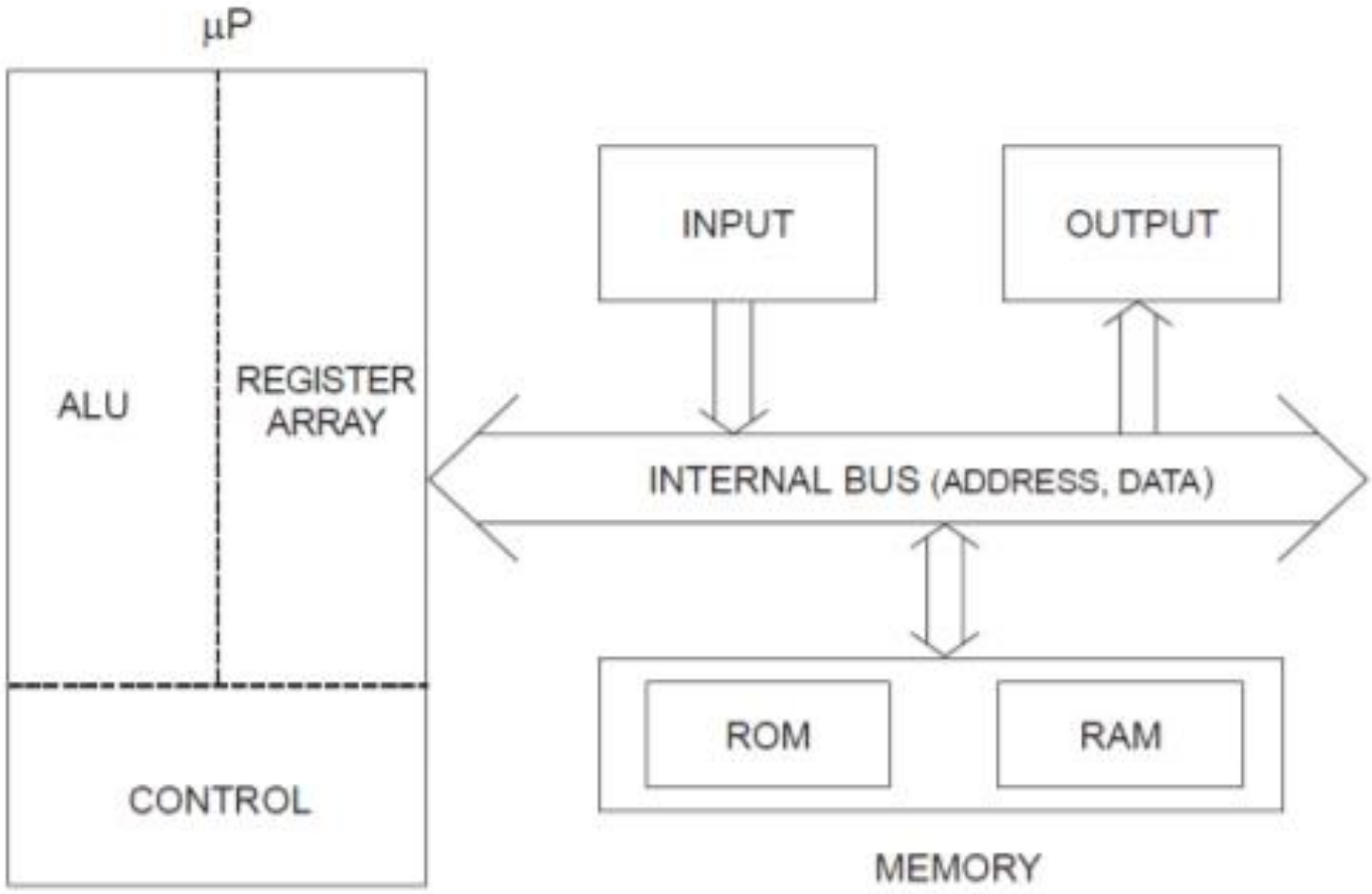


Figure 1.2 Architecture of Microprocessor

Minimum and Maximum mode

8086 Microprocessor will operate in two different mode, Pin:33 MN/MX will change mode

Minimum mode	Maximum mode
In minimum mode there can be only one processor i.e 8086	In maximum mode there can be multiple processor with 8086 like 8087, 8089
Pin:33 MN/MX is 1 to inicate minimum mode	Pin 33: MN/MX is 0 to indicate maximum mode
No multiprocessing→performance is slow	Multiprocessing→Very high performance
+5V to select minimum mode	0v(GND) to select maximum mode
HOLD & HOLDA signals are used for bus request with DMA controller	RQ/GT lines are used for bus request by other processors like 8087, 8089
ALE is given by 8086 as it is the only processor in the circuit	ALE is given by 8288 bus controller as there can be multiple processors in the circuit

Segment Registers:

- **Segmentation** is the process in which the main memory(1MB) of the computer is logically divided into different(4) segments.
- Each segment has its own base address.
- It is basically used to enhance the speed of execution of the computer system, so that the processor is able to fetch and execute the data from the memory easily and fast.
- Each segment contains(size) 64Kbyte of memory.

- Each segment is made up of contiguous memory locations.
- 8086 can access memory with address ranging from 00000 H to FFFFF H.
- It is an independent, separately addressable unit.

does not work the whole 1MB memory at any given time. However, it works only with four 64KB segments within the whole 1MB memory.

- Each of these segments are addressed by an address stored in corresponding segment register.
- These registers are 16-bit in size.
- Each register stores the base address (starting address) of the corresponding segment.
- Because the segment registers cannot store 20 bits, they only store the upper 16 bits.
- The 20-bit address of a byte is called its Physical Address.
- Offset is the displacement of the memory location from the starting location of the segment.

Example

The value of Data Segment Register (DS) is 2222 H.

- 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 22220H.

- If the data at any location has a logical address specified as: 2222 H : 0016 H
- Then, the number 0016 H is the offset.
- 2222 H is the value of DS.

- To calculate the effective address of the memory, BIU uses the following formula:

Effective Address = Starting Address of Segment + Offset

- To find the starting address of the segment, BIU appends the contents of Segment Register with 0H.
- Then, it adds offset to it.

Therefore:

EA = 22220 H

+ 0016 H

22236 H

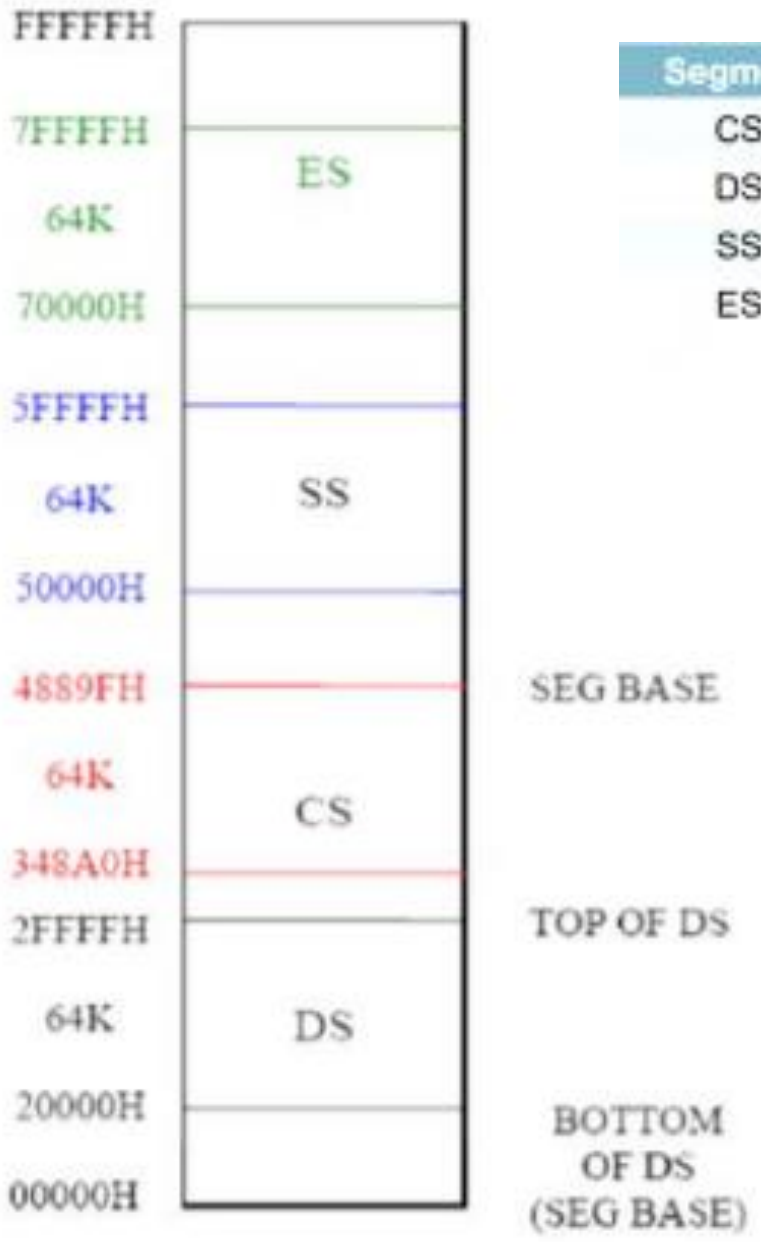
There are four segment registers.

- **Code segment (CS)** is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

Stack segment (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction. It is used for addressing stack segment of memory. The stack segment is that segment of memory, which is used to store stack data.

- **Data segment (DS)** is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions. It points to the data segment memory where the data is resided.

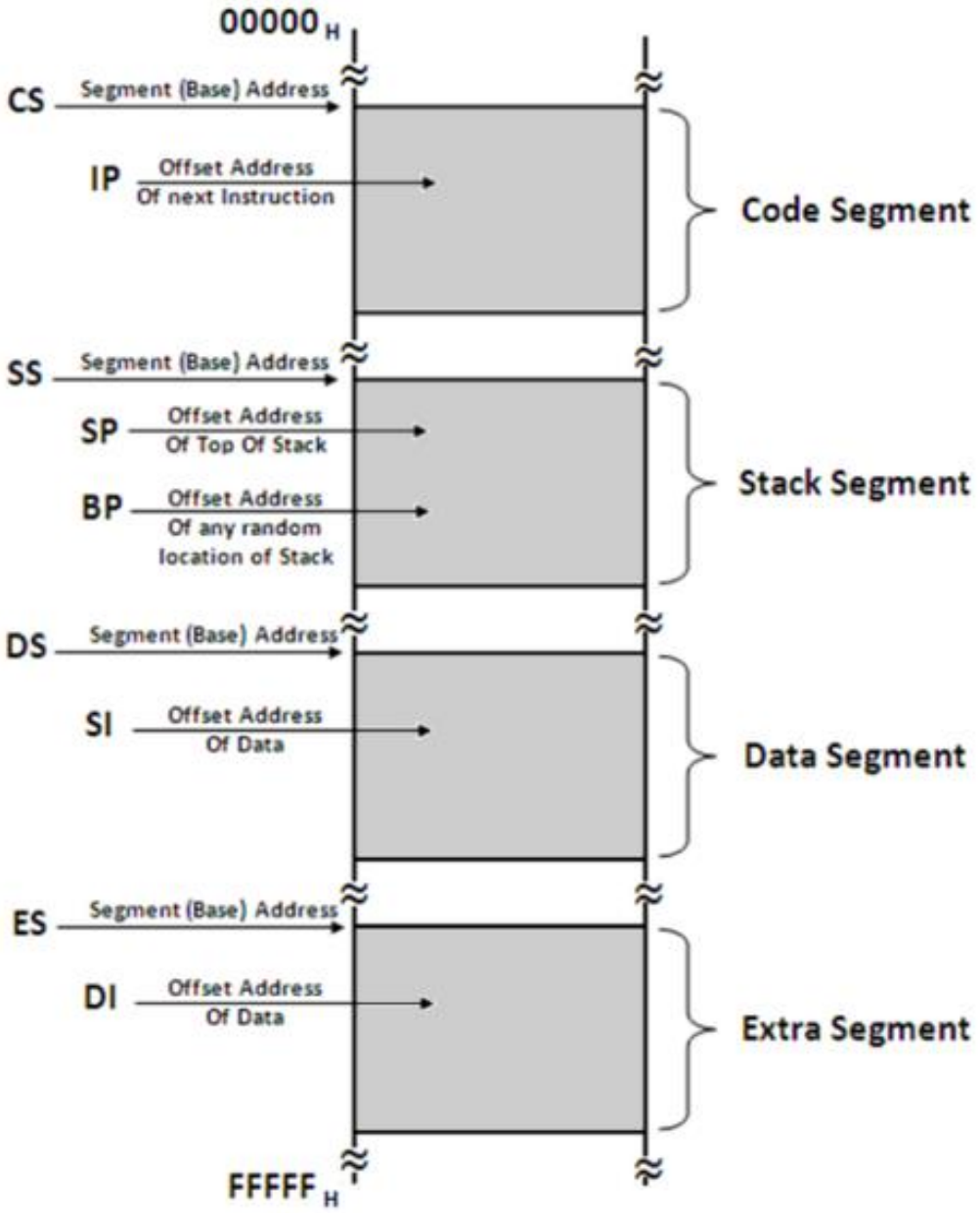
- **Extra segment (ES)** is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions. It also refers to segment which essentially is another data segment of the memory. It also contains data.



Segment	Offset Registers	Function
CS	IP	Address of the next instruction
DS	BX, DI, SI	Address of data
SS	SP, BP	Address in the stack
ES	BX, DI, SI	Address of destination data (for string operations)

Fig1.5. Memory segmentation

MEMORY SEGMENTATION IN 8086



Advantages of the Segmentation

- It provides a powerful memory management mechanism.
- Data related or stack related operations can be performed in different segments.
- Code related operation can be done in separate code segments.
- It allows to processes to easily share data.

Concept of Address, Data, control and Systems bus

- **BUS**: Collection of wires through which data is transmitted from one part of a computer to another is known as bus.
- A bus is a high-speed internal connection.
- Buses are used to send control signals and data between the processor and other components.

Three types of bus are used.

- **Address bus** - carries memory addresses from the processor to other components such as primary storage and input/output devices. The address bus is **unidirectional**.
- **Data bus** - carries the data between the processor and other components. The data bus is **bidirectional**.
- **Control bus** - carries control signals from the processor to other components. The control bus also carries the clock's pulses. The control bus is unidirectional.

•Bus connects all the internal components to the CPU and main memory.

The size of the bus known as width is important because it determines how much data can be transmitted at one time.

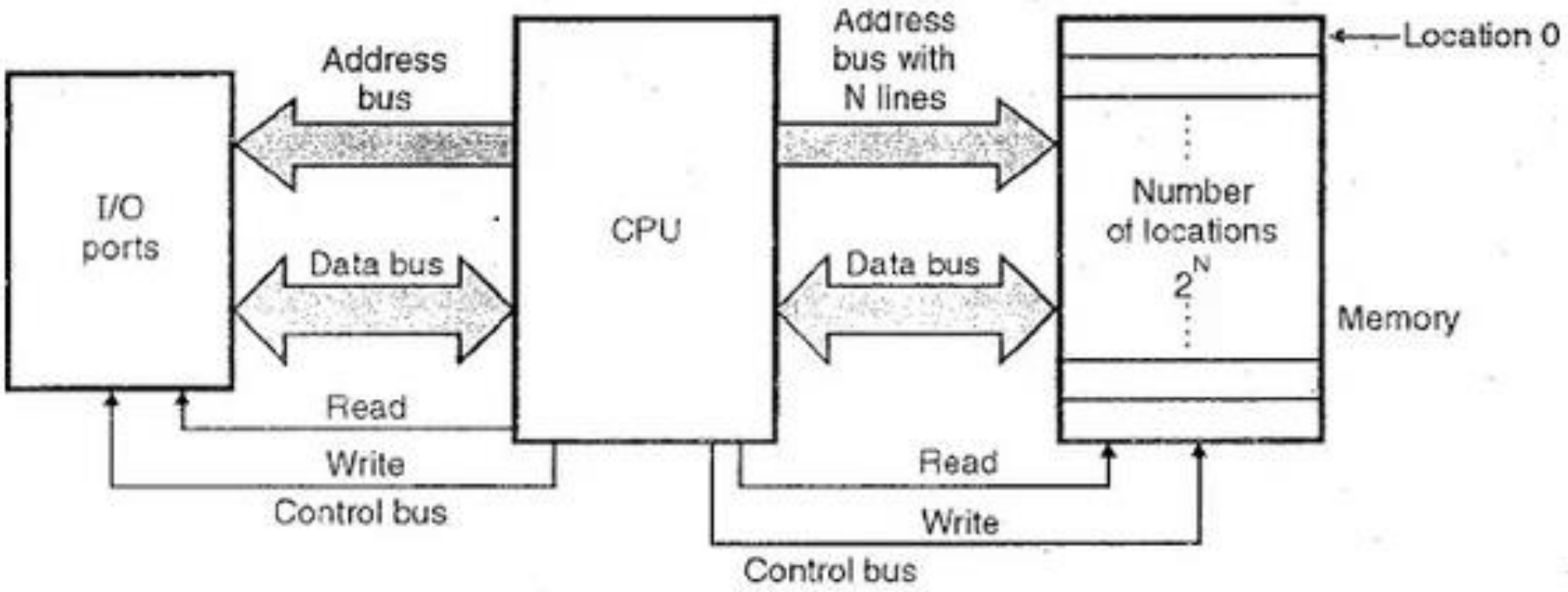
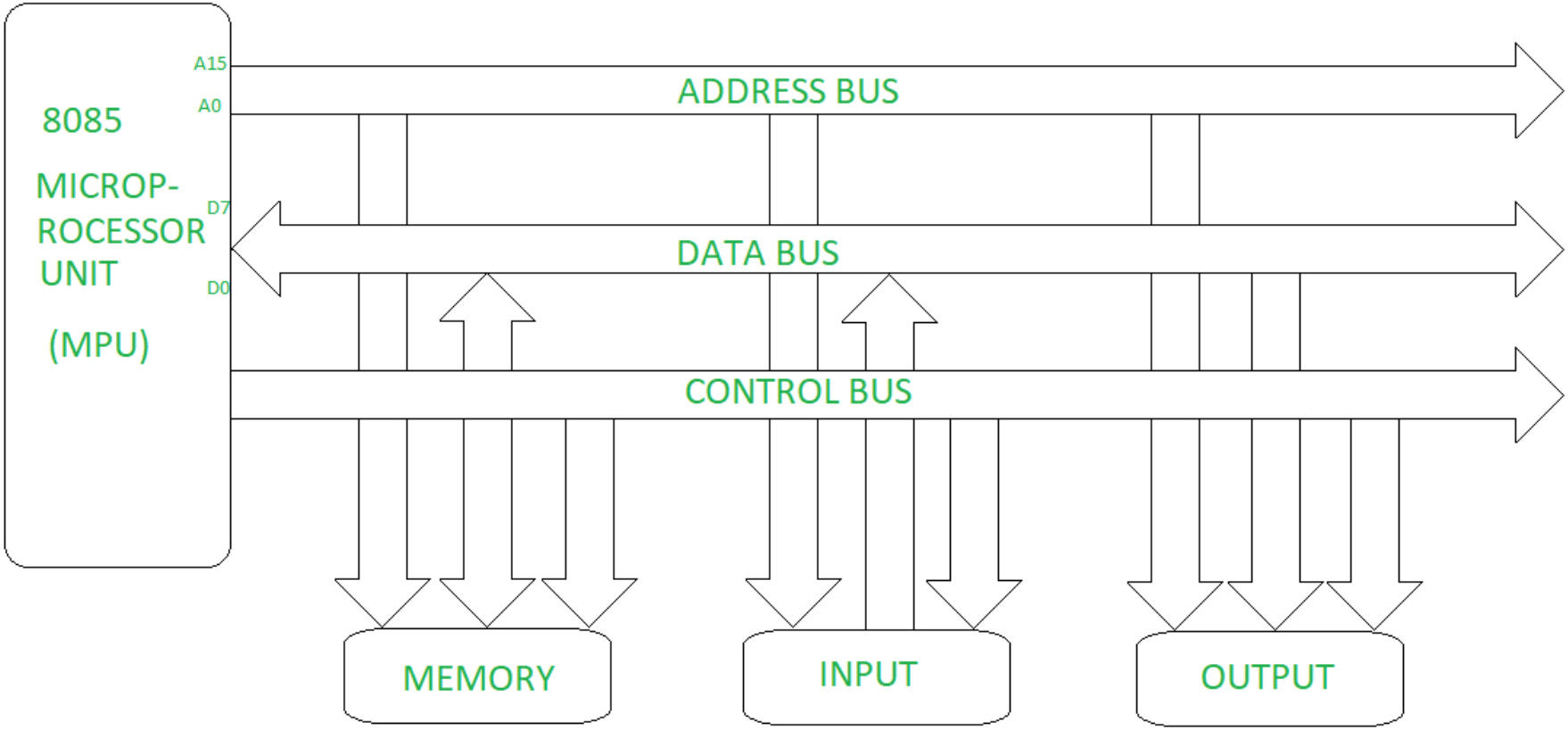


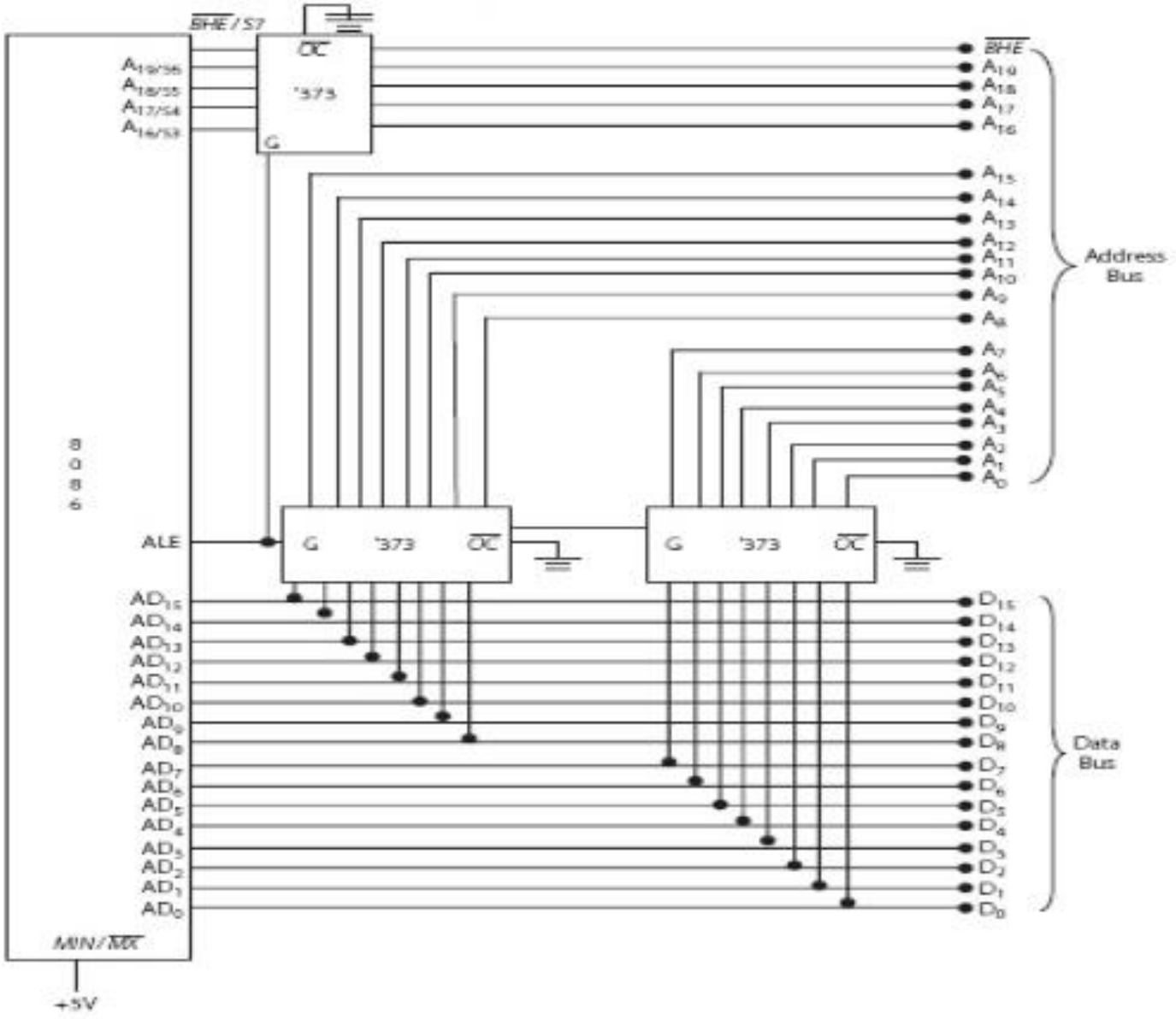
Fig. 1.2.3 : The three types of buses and their utility



Bus organization system of 8085 Microprocessor

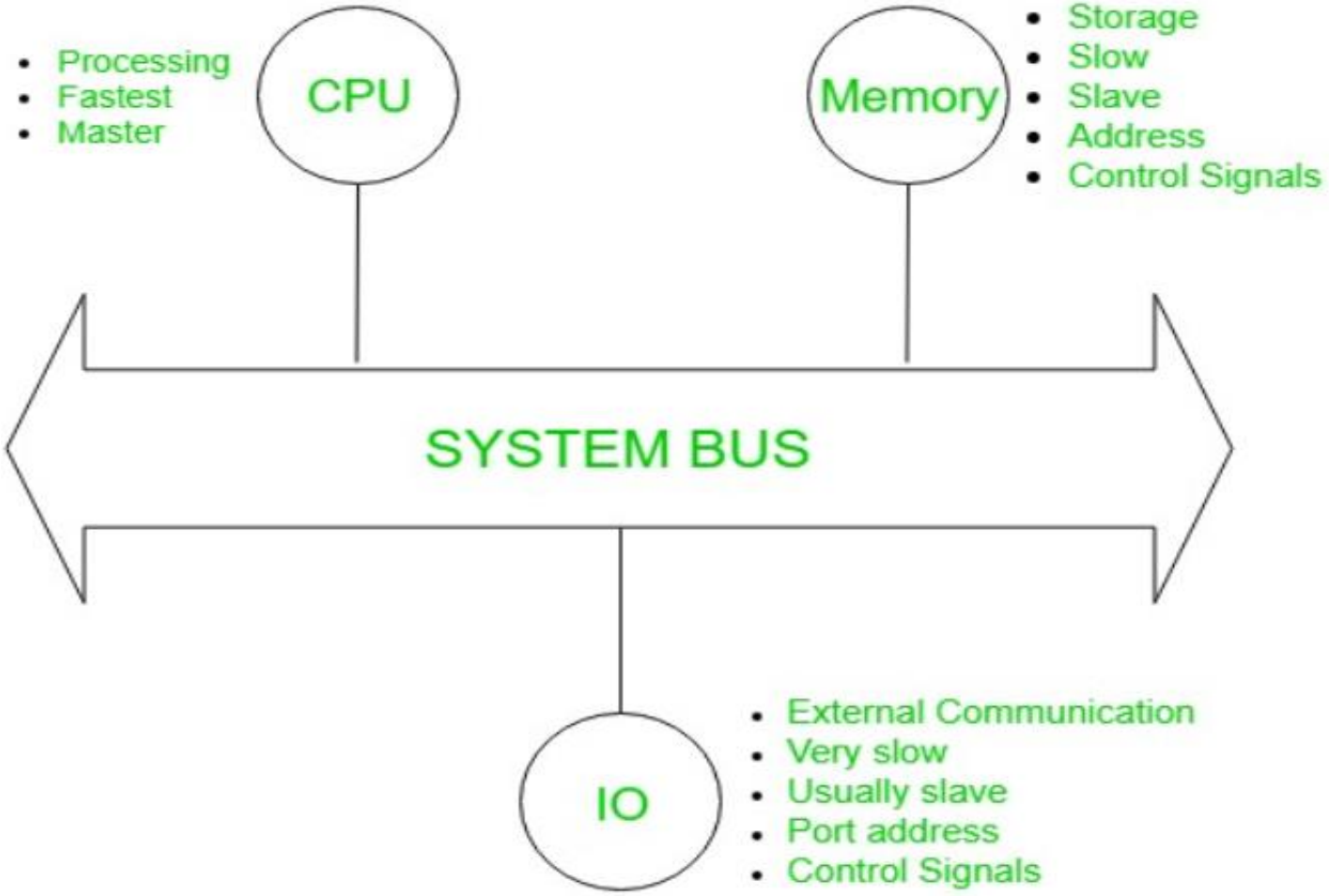
Buses transfer data in parallel.

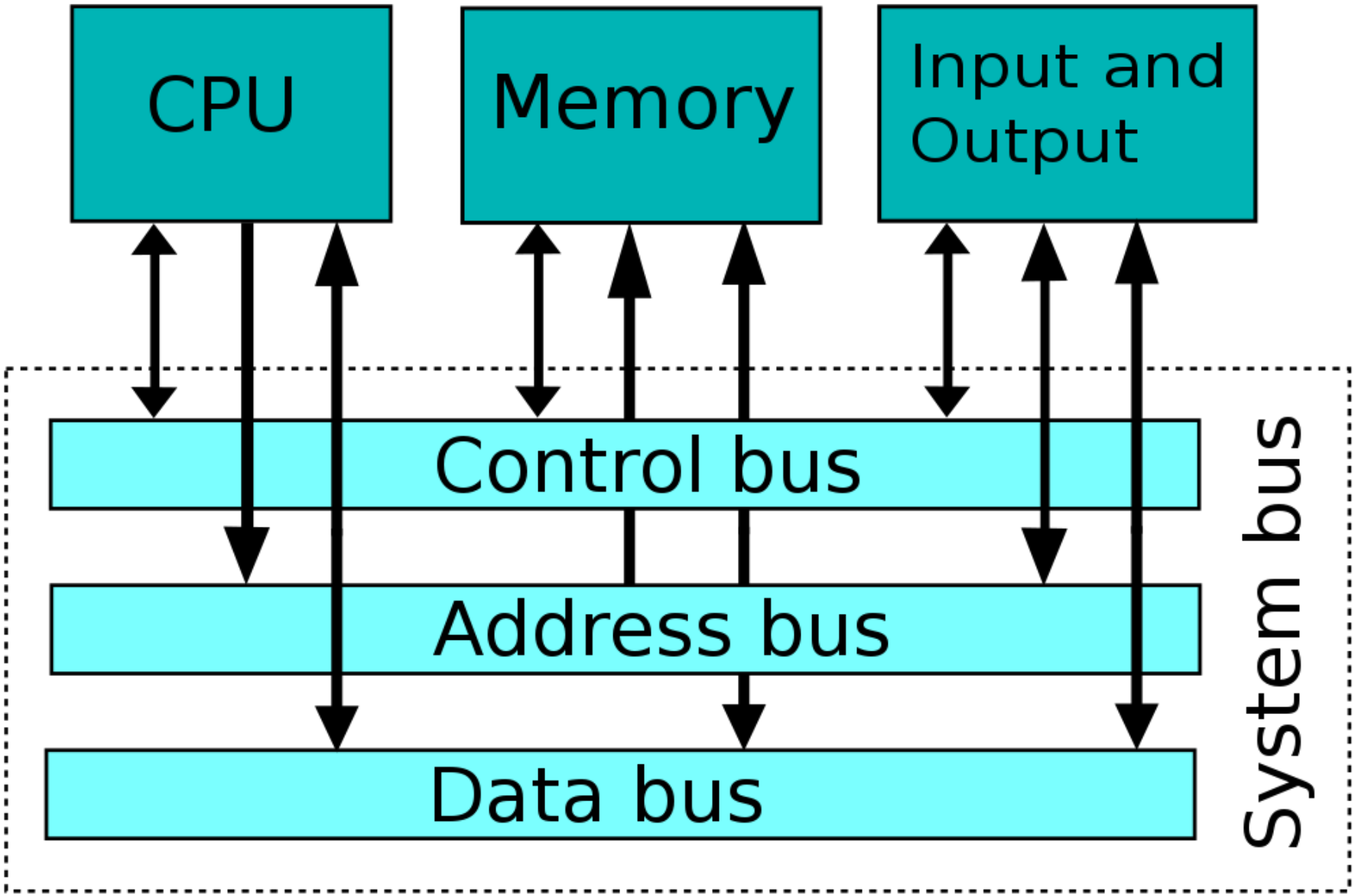
in a 16-bit bus, data are sent over 16 wires simultaneously.



System bus

- A bus which is used to provide communication between the major components of a computer is called a **System bus**.




CPU
Memory
Input and Output
Control bus
Address bus
Data bus
System bus

- Combining the functions of a data bus to carry information, an address bus to determine where it should be sent, and a control bus to determine its operation. the technique was developed to reduce costs and improve modularity,

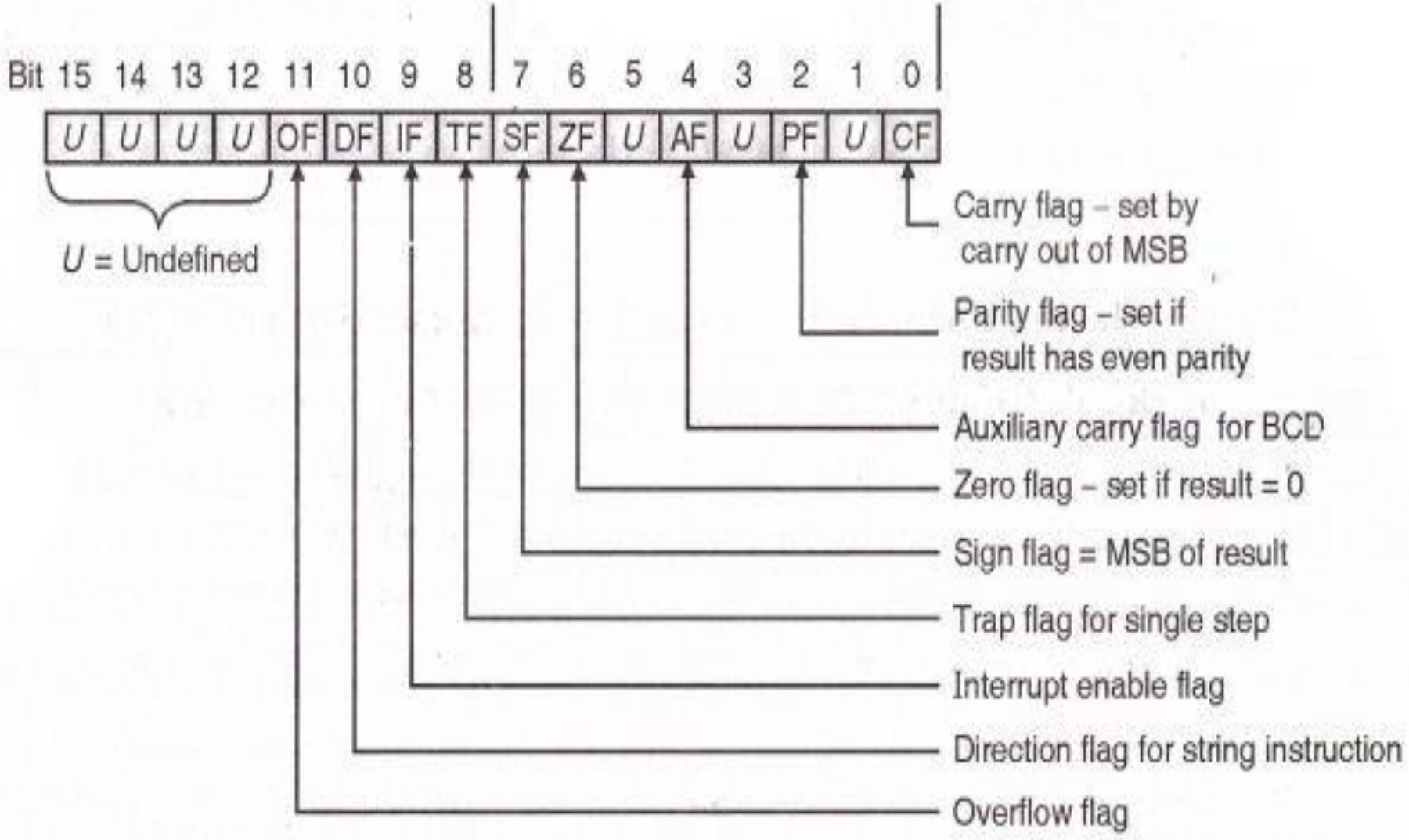
Types of flags

(Flag register of 8086microprocessor)

- The Flag register is a Special Purpose Register. Depending upon the value of result after any arithmetic and logical operation the flag(flip-flop) bits become **set =1** or **reset=0**.

8086 has 16-bit flag register, and there are 9 valid flag bits. The format of flag register is like below.

Bits	D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Flags					O	D	I	T	S	Z		AC		P		CY



8086 flag register format

It consists of 9 active flags out of 16. The remaining 7 flags marked 'U' are undefined flags. These 9 flags are of two types:

- 6 Status flags
- 3 Control flags

The 6 status flags are:

- **Sign Flag (SF)**
- **Zero Flag (ZF)**
- **Auxiliary Cary Flag (AF)**
- **Parity Flag (PF)**
- **Carry Flag (CF)**
- **Overflow Flag (OF)**

The 3 Control flags are:

- **Trap flag (TF)-**
- **Interrupt enable flag (IF)**
- **Direction Flag(DF)**

Status flags:

Carry flag (CY)-

- It is set whenever there is a carry or borrow out of the MSB of a result. D7 bit for an 8 bit operation and D15 bit for a 16 bit operation.

Parity flag (PF)-

- It is set if the result has even parity. If parity is odd, PF is reset.
- This flag is normally used for data transmission errors.

Auxiliary carry flag (AC)-

- It is set if a carry is generated out of the lower nibble.
- It is used only in 8 bit operations like DAA and DAS.

Zero flag (ZF)-

- It is set if the result is zero.

Sign flag (SF)-

- It is set if the MSB of the result is 1. For signed operations such a number is treated as negative.

Overflow flag (OF)-

- It will be set if the result of a signed operation is too large to fit in the number of bits available to represent it.
- It can be checked using the instruction INTO (Interrupt on Overflow).

Control flags:

Trap flag (TF)-

- It is used to set the trace mode i.e. start single stepping mode.
- Here the microprocessor is interrupted after every instruction so that the program can be debugged.

Interrupt enable flag (IF)-

- It is used to mask (disable) or unmask (enable) the INTR interrupt.
- If user sets IF flag, the CPU will recognize external interrupt requests. Clearing IF disables these interrupts.

Direction flag (DF)-

- If this flag is set, SI and DI are in auto-decrementing mode in string operations.

Flag Register and ADD instruction

The flag bits affected by the ADD instructions are: CF, PF, AF, ZF, SF and OF. The OF will be studied in Chapter 6.

Ex: Show how the flag register is affected by the addition of 38H and 2FH.

```
Solution:   MOV  BH,38H        ;BH=38H
            ADD  BH,2FH        ;BH = BH + 2F =  38 + 2F= 67H
```

```
    38          0011  1000
+   2F        + 0010  1111
    67          0110  0111
```

CF = 0 since there is no carry beyond d7

PF = 0 since there is odd number of 1`s in the result

AF = 1 since there is a carry from d3 to d4

ZF = 0 since the result is not zero

SF = 0 since d7 of the result is zero

Ex: Show how the flag register is affected by the following addition

Solution:

```
MOV  AX,34F5H      ;AX =34F5H
ADD  AX,95EBH      ;AX = CAE0H
```

```
  34F5      0011 0100 1111 0101
+ 95EB      1001 0101 1110 1011
  CAE0      1100 1010 1110 0000
```

CF = 0 since there is no carry beyond d15
PF = 0 since there is odd number of 1s in the lower byte
AF = 1 since there is a carry from d3 to d4
ZF = 0 since the result is not zero
SF = 1 since d15 of the result is 1

- Note that the MOV instructions have no effect on the flag (Explain on the existing example)