217529- Internet of Things

Unit Number: **2**

Unit Name: **Communication Interface**

Unit Outcomes: CO2

Develop the skill set to build IoT system and sensor interfacing

Syllabus

- Basic Peripherals & their interfacing with8086/8088,
- Semiconductor Memory Interfacing-Dynamic RAM Interfacing-Interfacing I/O ports-PIO 8255
- Modes of operation-interfacing Analog-Digital Data converter-stepper motor interfacing

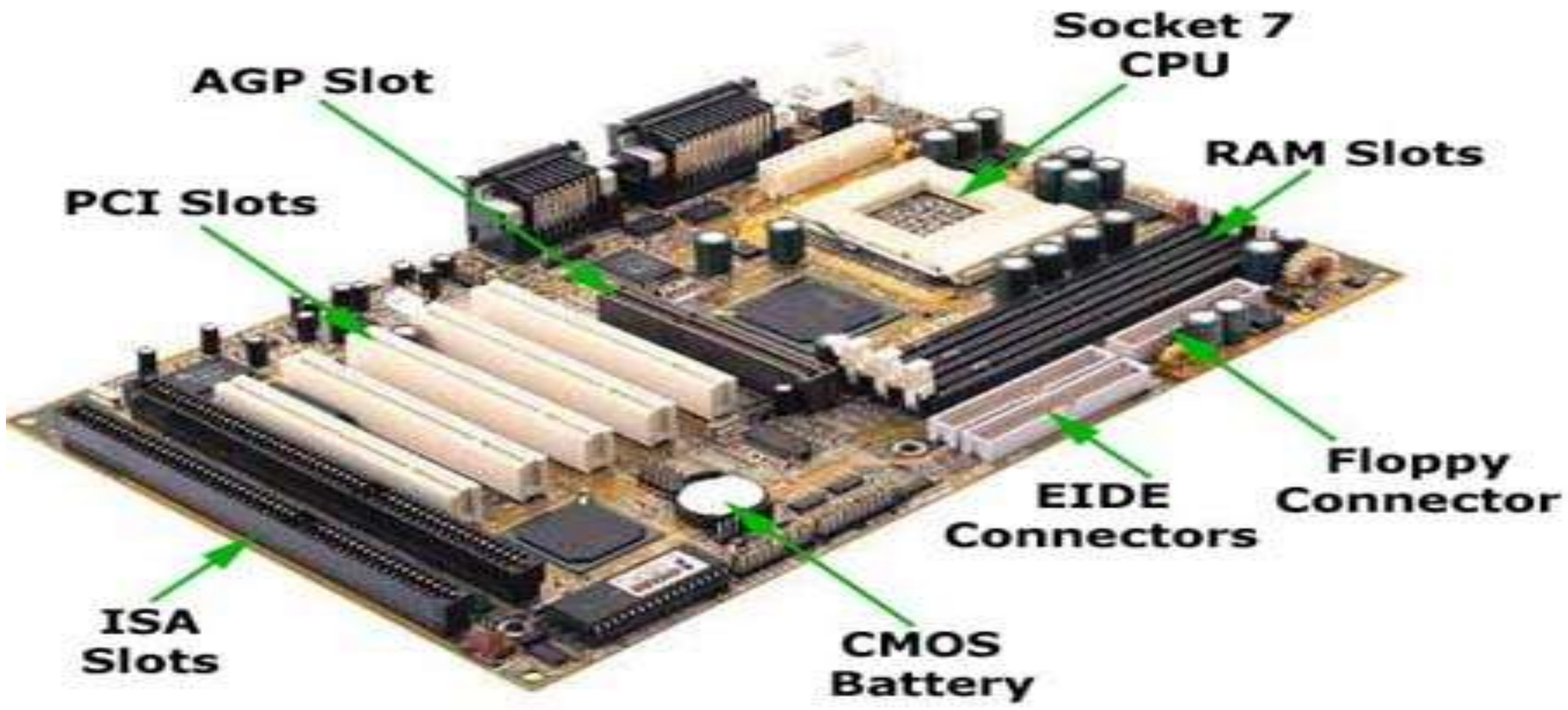
Introduction of Microprocessor (µP)

- A microprocessor is a central processing unit or the brain of a com

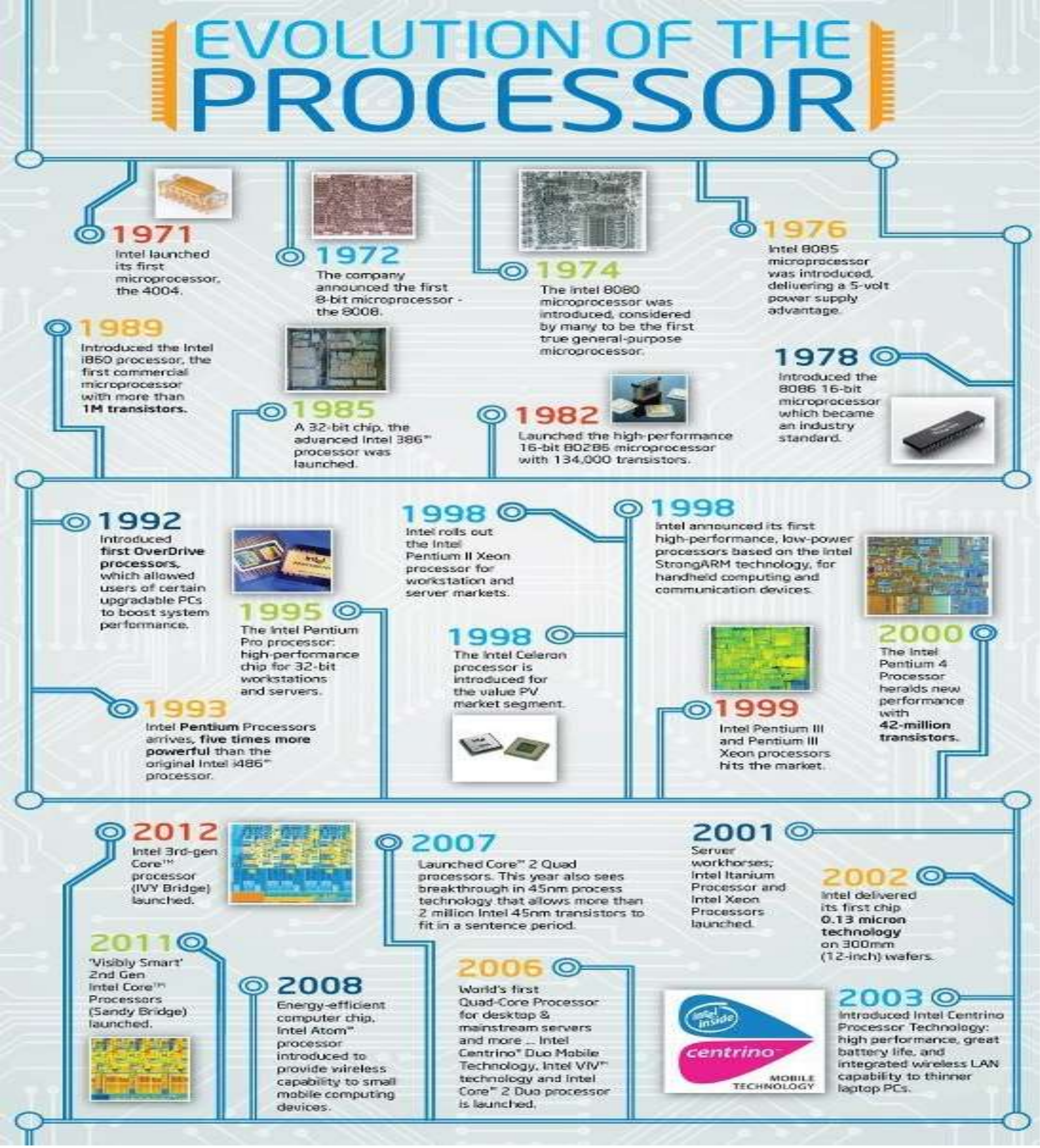
  inside a single Integrated circuit (IC).
- It is made up of millions of semiconductor transistors, diodes & resistors and it is responsible for any arithmetic or logical operation.
- The microprocessor is the clock-driven digital integrated circuit that is built up by using VLSI technology and this reduces the overall cost and power of the processor. It is a digital device capable of processing any binary data given to it.
- It accepts the input in binary form, processes them as per the instruction stored in the memory, and performs the arithmetic logic and sequential digital logic operation.
- The Central Processing Unit (CPU) 'controls' what the computer does and is responsible for performing calculations and data processing. It also handles the movement of data to and from system memory.

Introduction of Microprocessor (up)

- The central processing unit is the processor that controls all the functions of the computer system. Fitted in socket 7 in following diagram. Top of it fan is fitted.



- It is a kind of integrated circuit (IC) unit which combines all the basic functions of a central processing unit (CPU) of the computer.
- It is a programmable unit that is fabricated on the silicon chip and it consists of an ALU unit, clock, control unit, and register array which accepts the input in binary form (0's and 1's) and delivers the output after processing the input data as per the instructions fetched into the memory unit.



EVOLUTION OF THE PROCESSOR

1971 Intel launched its first microprocessor, the 4004.

1972 The company announced the first 8-bit microprocessor - the 8008.

1974 The Intel 8080 microprocessor was introduced, considered by many to be the first true general-purpose microprocessor.

1976 Intel 8085 microprocessor was introduced, delivering a 5-volt power supply advantage.

1989 Introduced the Intel i860 processor, the first commercial microprocessor with more than **1M transistors.**

1985 A 32-bit chip, the advanced Intel 386™ processor was launched.

1982 Launched the high-performance 16-bit 80286 microprocessor with 134,000 transistors.

1978 Introduced the 8086 16-bit microprocessor which became an industry standard.

1992 Introduced **first OverDrive processors,** which allowed users of certain upgradable PCs to boost system performance.

1995 The Intel Pentium Pro processor: high-performance chip for 32-bit workstations and servers.

1993 Intel **Pentium** Processors arrives, **five times more powerful** than the original Intel i486™ processor.

1998 Intel rolls out the Intel Pentium II Xeon processor for workstation and server markets.

1998 The Intel Celeron processor is introduced for the value PV market segment.

1998 Intel announced its first high-performance, low-power processors based on the Intel StrongARM technology, for handheld computing and communication devices.

1999 Intel Pentium III and Pentium III Xeon processors hits the market.

2000 The Intel Pentium 4 Processor heralds new performance with **42-million transistors.**

2012 Intel 3rd-gen Core™ processor (IVY Bridge) launched.

2011 'Visibly Smart' 2nd Gen Intel Core™ Processors (Sandy Bridge) launched.

2008 Energy-efficient computer chip, Intel Atom™ processor introduced to provide wireless capability to small mobile computing devices.

2007 Launched Core™ 2 Quad processors. This year also sees breakthrough in 45nm process technology that allows more than 2 million Intel 45nm transistors to fit in a sentence period.

2006 World's first Quad-Core Processor for desktop & mainstream servers and more ... Intel Centrino® Duo Mobile Technology, Intel VIV™ technology and Intel Core™ 2 Duo processor is launched.

2001 Server workhorses; Intel Itanium Processor and Intel Xeon Processors launched.

2002 Intel delivered its first chip **0.13 micron technology** on 300mm (12-inch) wafers.

2003 Introduced Intel Centrino Processor Technology: high performance, great battery life, and integrated wireless LAN capability to thinner laptop PCs.

- The Processor is the heart of a Computer , the development of processors began in Late 1960s

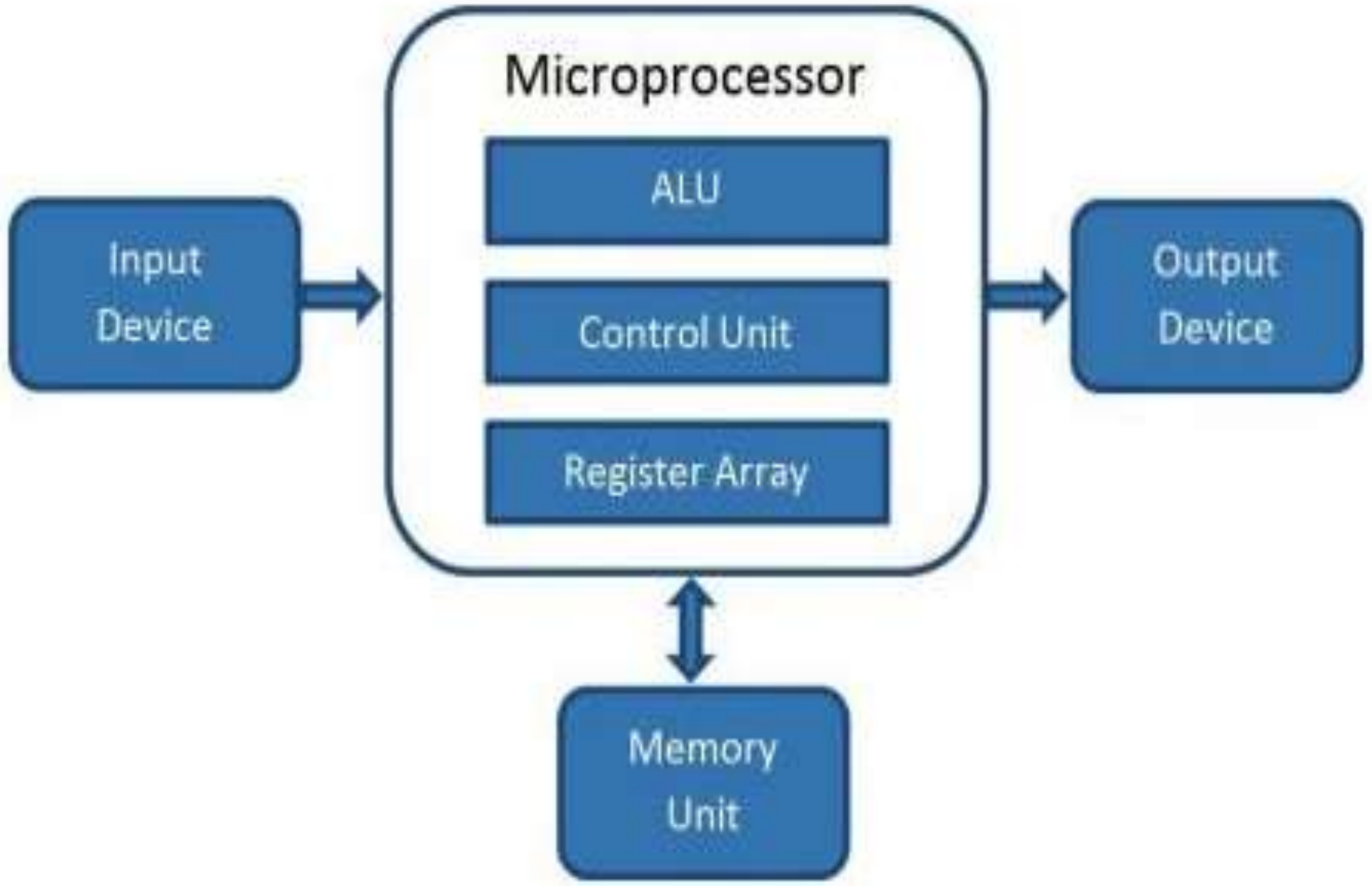
- Intel came out with the first ever Processor 4004 in 1971 which was a 4 bit microprocessor and now we have advanced so much that we use microprocessors like i3 , i5 and i7.

- In the 1980s, Arcade Mania came out by Namco igniting a novel trend. The Osborne 1 Laptop of 1981 had five screens with 10.7kgs of weight and a processor. The gaming business also took in the processor with Nintendo NES in 1986.

- The era of the personal computer in 1991 led to the wide use of microprocessors followed by music players like MP3 and iPod in 1997 and 2001 respectively. Early 2000 saw the launch of the blackberry Smartphone and Microsoft Windows Tablet.

- The notebook came out in 2008 with better media and internet content. Digital Signage like internet devices, automobiles, etc. also uses high-end microprocessors since 2011. Ultrabook uses the latest technology since 2011 with a high-performance computing experience.

Block Diagram of Microprocessor (µP)



- So basically a microprocessor takes input from input devices (keyboard, mouse) process it as per instructions given in the memory and produces output (Monitor).

- Memory unit is a component of a computer system. It is used to store data, instructions and information.

- All processing is going to done by Microprocessor("µP").
  - A microprocessor consists of an ALU, control unit and register array.

- ALU performs arithmetic and logical operations on the data received from an memory. ALU fetch the all data from memory unit. After fetching microprocessor decode the instructions. After decoding it execute the instruction.

- Control unit controls the instructions and flow of data within the computer. The control unit converts the input into control signals and then sent to the processor and directs the execution of a program. The operations that have to performed are directed by the processor on the computer.

- Register array There are various types of registers in 8085/8086.These registers are used for storage ,manipulating data and instructions. It consists of registers identified by letters like B, C, D, E, H, L, and accumulator.

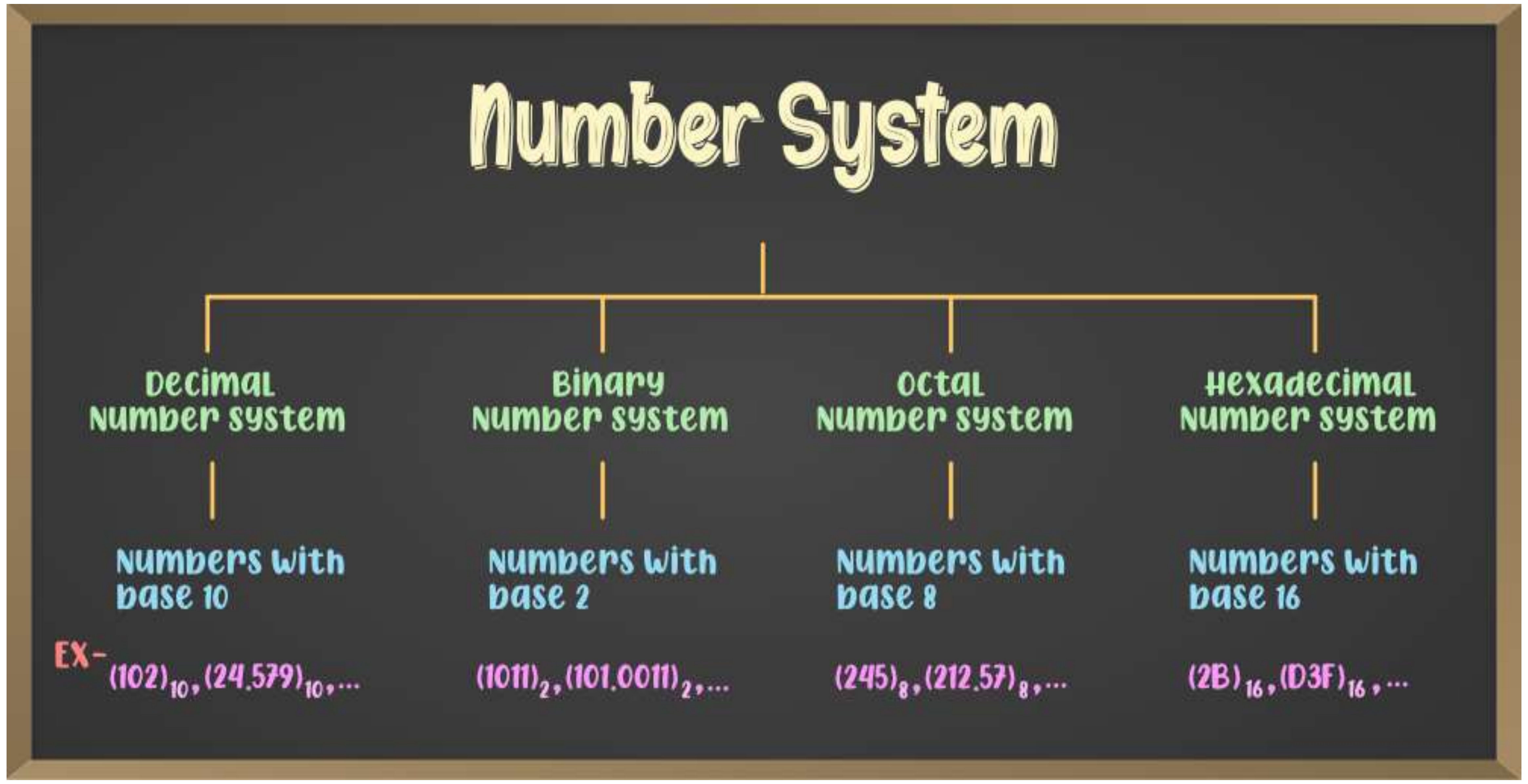
Understand what is decoding

- Lets understand what is decoding. for example: what is the language used to write following instruction.
  - a=b+c   Higher level language is used to write this kind of instruction.
  - ADD B, C;   Assembly level language Which we are going to used
  - 0110111 Machine language, Lower level Language, Object code, Binary Lang

- When we compile the program then HLL code is converted into Machine level language.
  - For example C program saved file using .c extension, once we compile another file is created i.e. .bin or .obj
- Now when Microprocessor("μP") fetch the instruction in binary format.
- Now when we creating assembly language program in 8085/8086 program then it saved as extension .ASM and these program given to assembler to convert code in Machine Level Language.
- After decoding Microprocessor("μP") Binary code or opcode (unique code of instruction)

What is Number System:

- Number system defines a set of values used to represent quantity.
  - Quantity represent using numbers like 1,2,3 …….
  - There are four number systems that a computer supports. They are



Number System

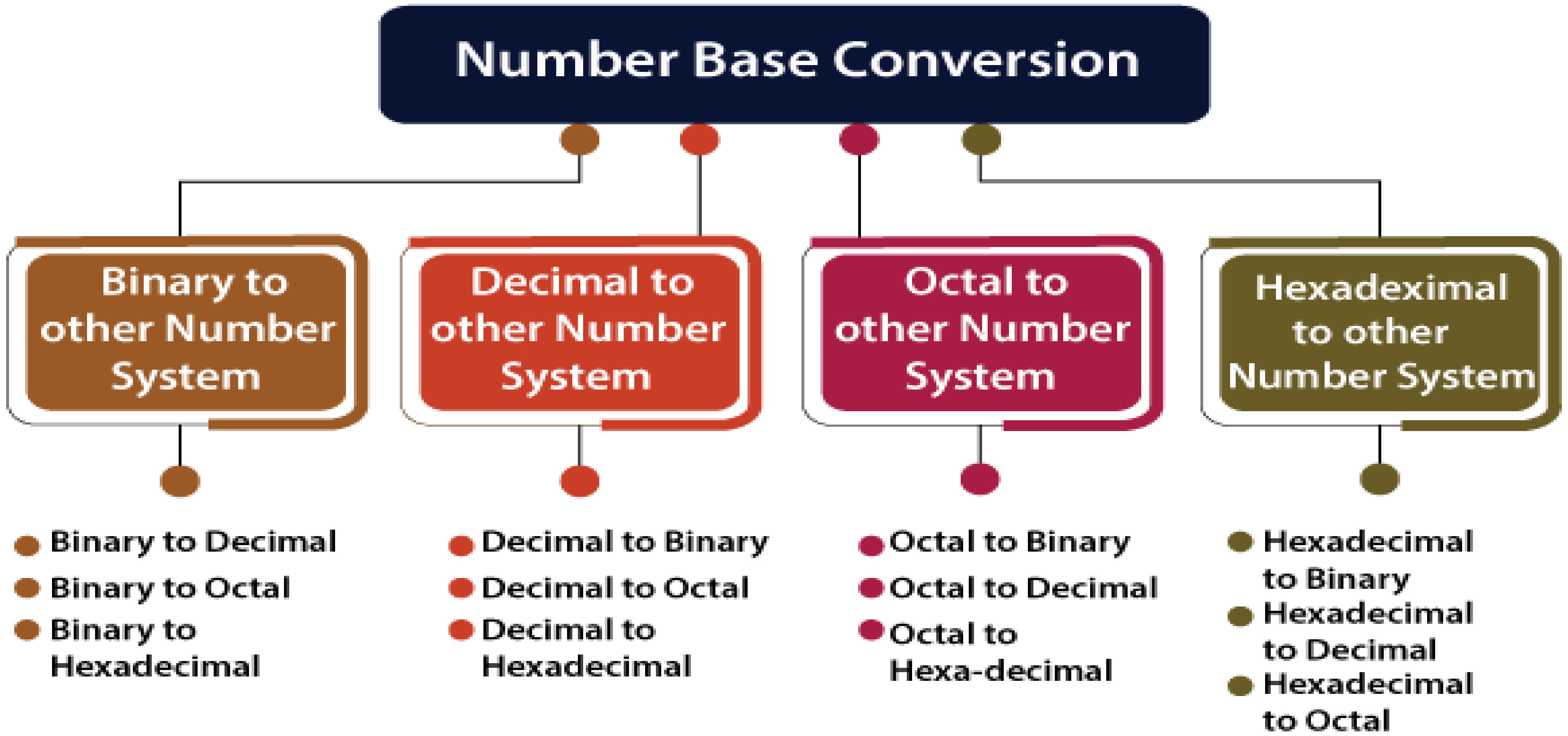
Decimal Number system — Numbers with base 10 — EX- $(102)_{10}$, $(24.579)_{10}$, ...

Binary Number system — Numbers with base 2 — $(1011)_2$, $(101.0011)_2$, ...

Octal Number system — Numbers with base 8 — $(245)_8$, $(212.57)_8$, ...

Hexadecimal Number system — Numbers with base 16 — $(2B)_{16}$, $(D3F)_{16}$, ...

- We have four types of number systems so each one can be converted into the remaining three systems. There are the following conversions possible in Number System



Number Base Conversion

Binary to other Number System
- Binary to Decimal
- Binary to Octal
- Binary to Hexadecimal

Decimal to other Number System
- Decimal to Binary
- Decimal to Octal
- Decimal to Hexadecimal

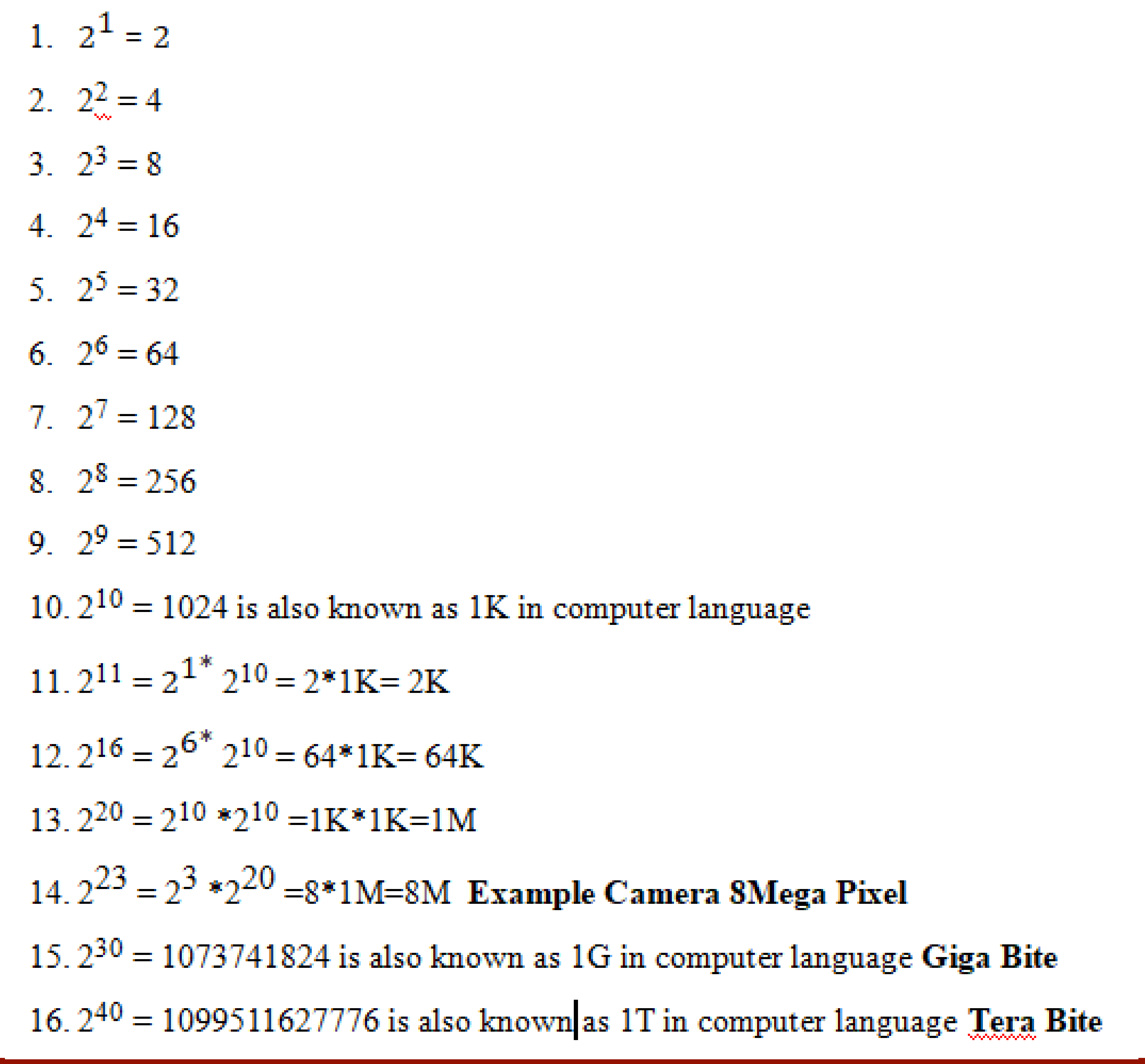
Octal to other Number System
- Octal to Binary
- Octal to Decimal
- Octal to Hexa-decimal

Hexadeximal to other Number System
- Hexadecimal to Binary
- Hexadecimal to Decimal
- Hexadecimal to Octal

Cont..

- In this unit we are going to used Hexadecimal Number. Hex is going to convert into binary in machine. So we already know the conversion process of ***HEX to Binary***.
- For example
  - 35H=00110101
  - 41H=01000001
  - 72H=01110010
  - FFH= 11111111
- To convert this number we need 8bits. Therefore it is know as 8bit number.
- Range of 8 bit number is 00 to FF. 8 bit number know as bit
- Range of 16 bit number is 0000 to FFFF 16 bit number know as word.
- Example Identify how many bit numbers are: 25, 62, 1234, 4000, 51FC, 5721, 21, FF

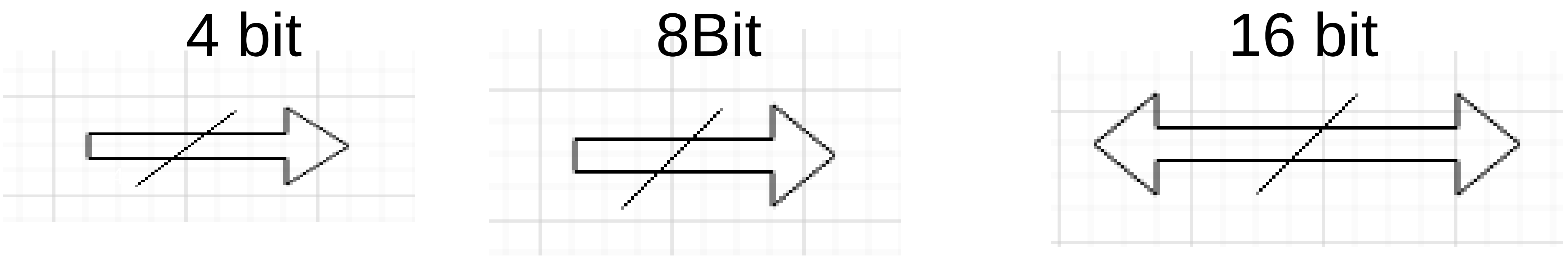
Cont.. Lets understand concept of power of 2



1. $2^1 = 2$
2. $2^2 = 4$
3. $2^3 = 8$
4. $2^4 = 16$
5. $2^5 = 32$
6. $2^6 = 64$
7. $2^7 = 128$
8. $2^8 = 256$
9. $2^9 = 512$
10. $2^{10} = 1024$ is also known as 1K in computer language
11. $2^{11} = 2^{1*}\, 2^{10} = 2*1K= 2K$
12. $2^{16} = 2^{6*}\, 2^{10} = 64*1K= 64K$
13. $2^{20} = 2^{10} * 2^{10}$ =1K*1K=1M
14. $2^{23} = 2^3 * 2^{20}$ =8*1M=8M **Example Camera 8Mega Pixel**
15. $2^{30} = 1073741824$ is also known as 1G in computer language **Giga Bite**
16. $2^{40} = 1099511627776$ is also known as 1T in computer language **Tera Bite**

Buses

- A bus organization is a group of conducting wires which carries information, all the peripherals are connected to microprocessor through the bus. A system bus is nothing just a group of wires to carry bits.
- Bus is a common channel through which bits from any sources can be transferred to the destination.
- Each wire carries just one bit, so the number of wires determines the largest data WORD the bus can transmit: a bus with eight wires can carry only 8-bit data words, and hence defines the device as an 8-bit device. Following fig shows bus and bit identification



- Types of Bus in the microprocessor are:-
  - Address Bus
  - Data Bus
  - Control Bus

Cont..

- There are four types of **internal buses**.
  - **Data Bus (bidirectional)** : This bus system sends computer information or instructions to the output device. The data bus allows data to travel between the microprocessor (CPU) and memory (RAM). . Length of Data bus in 8086 microprocessor is 8 bit (that is, two hexadecimal Digits0, ranging from 00H to FFH.
  - **Address Bus** : This bus system determines the correct location of memory and the data is received or received from it.
    - The addresses bus is unidirectional because of data flow in one direction, from the microprocessor to memory or from the microprocessor to input/out devices. Length of Address bus of 8085 microprocessor is 16 bit (That is, four hexadecimal digits), ranging from 0000H to FFFF H. Address Bus is used to perform the first function, identifying a peripheral or a memory location. In any operation address bus is used first.
  - **Control Bus** : This bus carries the signal from the CPU and converts it into various parts of the computer (Keyboard, Mouse, Disk drive, Printer).
  - The control bus carries the control signals to control all the associated peripherals, the microprocessor uses control bus to process data, that is what to do with selected memory location signals are
      - :- a. memory card read b.memory write c. input/output,write.
  - Power Bus : These buses provide electricity to various parts of the computer.

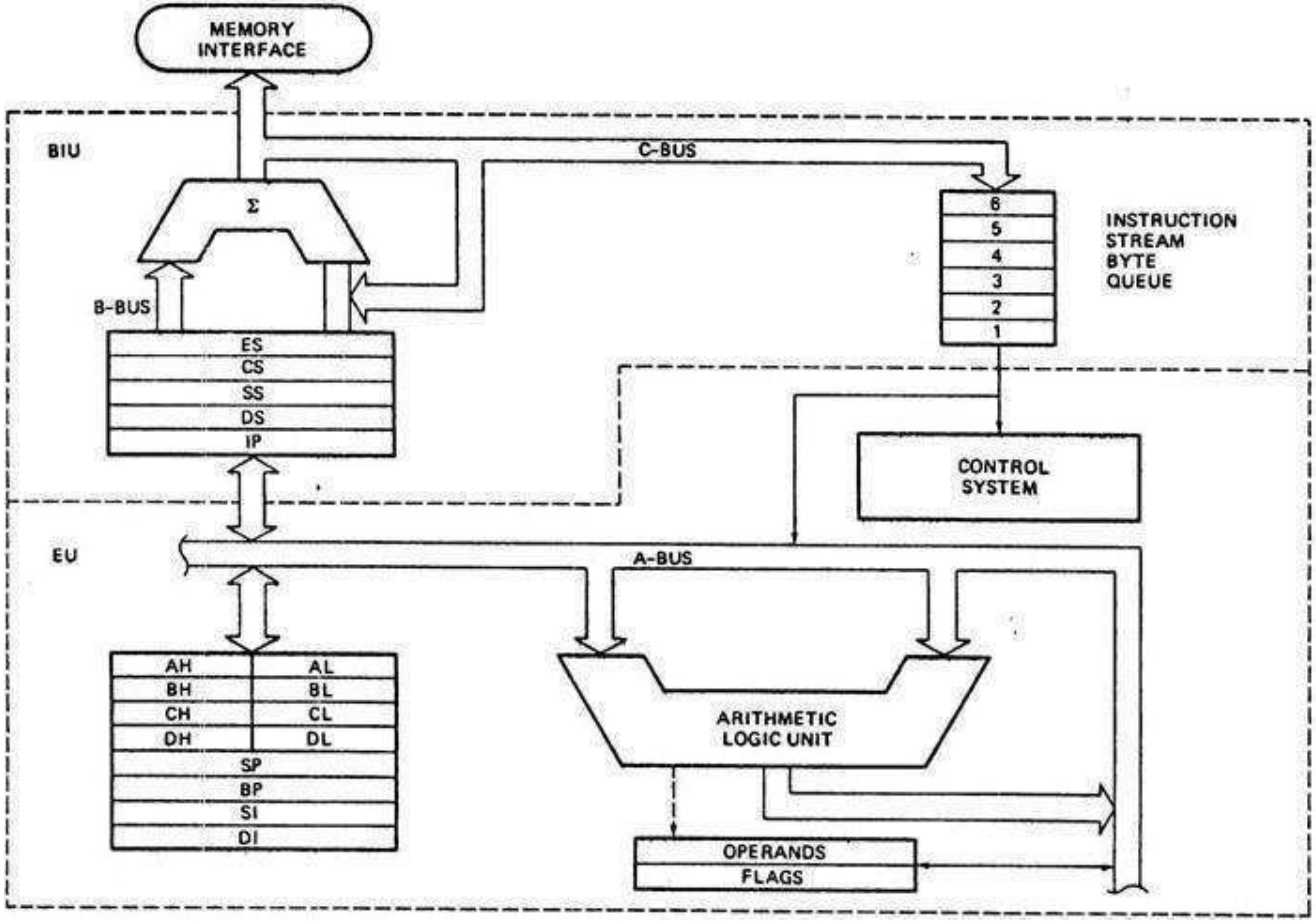
Cont..

- **External Bus**
  - There are several slots aligned in the motherboard, many of these slots have been placed in the pins.
  - Internal buses are connected to these slots and external buses are connected to the internal bus in that place, so that the connections between motherboards and other parts are connected.
  - This external buses (cabal) are used to connect between the peripheral device and the internal device, and also used for connect internal buses and internal device.

Buses used in reading and writing operations.

- Writing operation
  - For example address bus decide location 1000 in memory, Data bus decide write 75 in 1000 address then control bus write that data into address 1000.
- Reading Operation:
  - For example If we want to read data from location 1000. then first address bus is used to find address then location is selected. Now control bus is going to used for read operation, and finally data bus used to transfer that data to microprocessor.
- The system bus works by combining the functions of the three main buses: namely, the data, address and control buses. System bus characteristics are dependent on the needs of the processor, the speed, and the word length of the data and instructions.
- The size of a bus, also known as its width, determines how much data can be transferred at a time and indicates the number of available wires. A 32-bit bus, for example, refers to 32 parallel wires or connectors that can simultaneously transmit 32 bits.

Internal Architecture of 8086



- The 8086 microprocessor is divided into two functional units:
  - EU (Execution Unit)
  - BIU (Bus Interface Unit).

- The BIU (Bus Interface Unit) fetches instructions, reads data from memory and ports, and writes data memory and I/O ports.

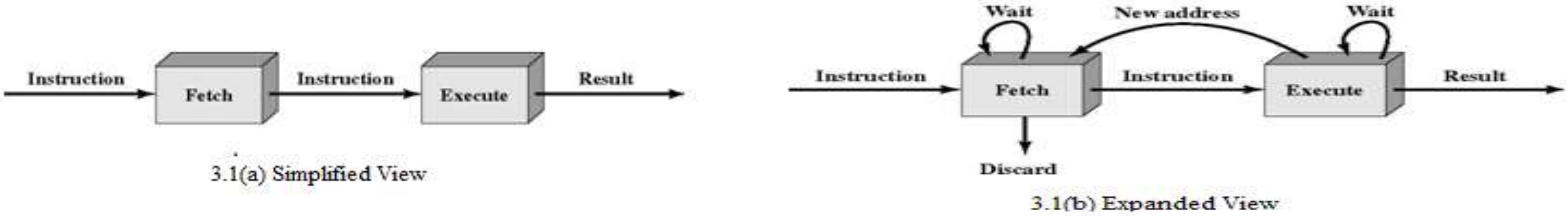
- EU (Execution Unit) receives program instruction codes and data from the BIU, execute these instructions and stores the results either in the general registers or output them through the BIU.

- EU has no connections to the system buses. It receives and outputs all

What is Pipelining?

- 8085 fetch the one instruction, get inside the processor, execute it, finished it. After that it fetch the next instruction.

- But in 8086 it will fetch the one instruction from memory, while executing that instruction it fetch another instruction side by side. That the reason 8086 architecture divide into two function unit.



3.1(a) Simplified View

3.1(b) Expanded View

3.1 Two-Stage Instruction Pipeline

- Pipelining organizes the execution of the multiple instructions simultaneously. Microprocessor instruction pipelining is a hardware implementation that allows multiple instructions to be simultaneously processed through the instruction cycle. This is enabled by the instruction cycle itself as it divides the operations that have to be performed on each instruction into standalone phases (e.g decode, fetch, execute).

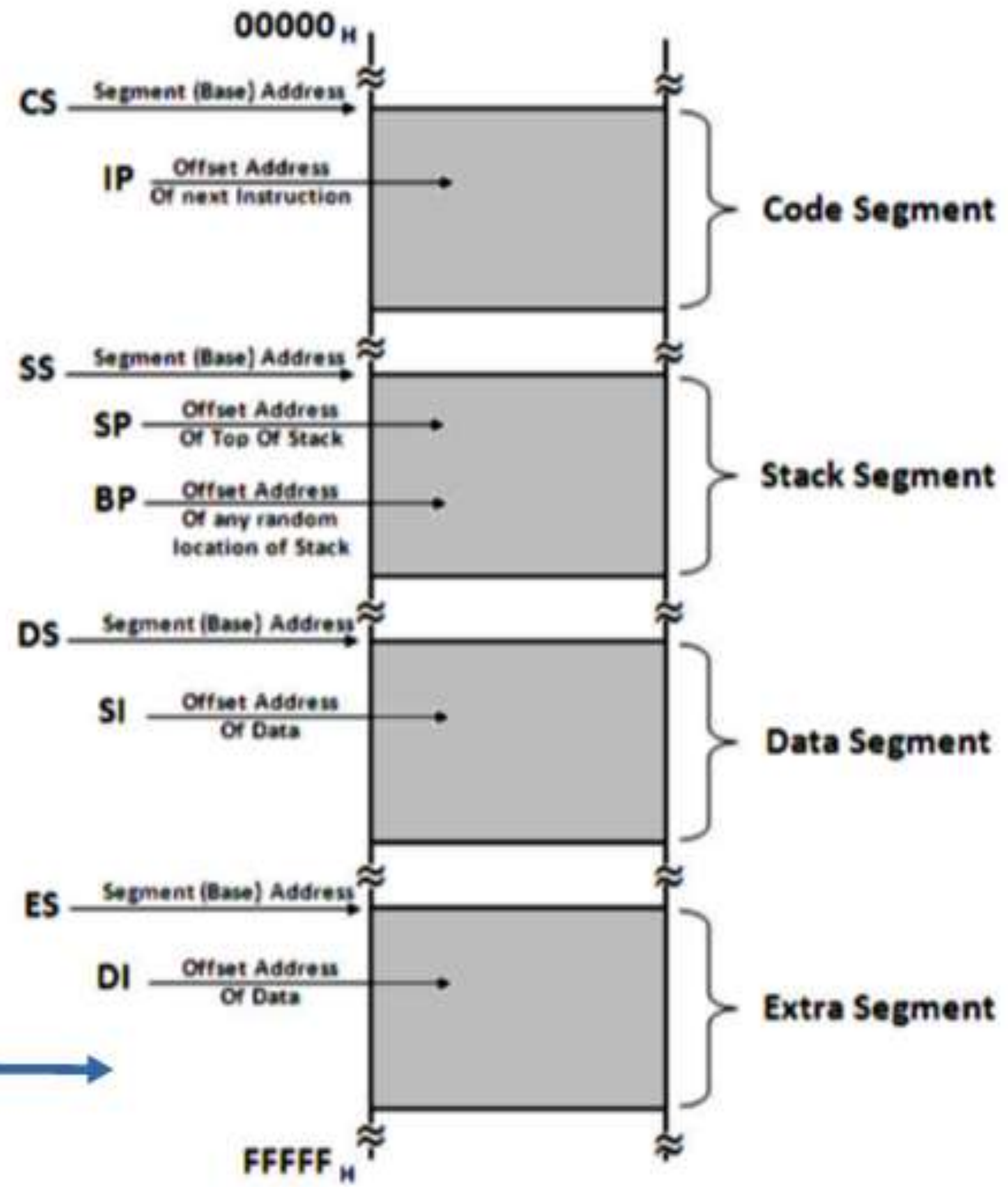
BIU of 8086 microprocessor

- **Segment Register**
- Segment registers are basically memory pointers located inside the CPU.
- Segmentation means dividing the memory into logically different parts called segments. 8086 has a 20-bit address bus, hence it can access 1MB memory (220 = 210 *210 =1K*1K=1M).
- It is not possible to work with a 20 bit address as it is not a byte compatible number i.e. (20 bits is two and a half bytes).
- To access the information in machine we used virtual address. Because we created folder and keep information. To avoid working with this incompatible number, we create a virtual model of the memory.
- Here the memory is divided into 4 segments:
    - Code, Stack Data and Extra.
    - The max size of a segment is 64KB and the minimum size is 16 bytes.

A Segment Address is the starting address of a segment.

Offset address tells the **memory location within the segment**.

Memory Supported by8086



CS, SS, DS and ES give you Segment Address. And IP, SP, BP, SI and DI give us offset address.

All x86 segment registers are 16 bits in size, irrespective of the CPU

Cont..

- **CS, code segment register**. Machine instructions exist at some offset into a code segment. The segment address of the code segment of the currently executing instruction is contained in CS. 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.

- **DS, data segment register** . Variables and other data exist at some offset into a data segment. There may be many data segments, but the CPU may only use one at a time, by placing the segment address of that segment in register DS. 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.

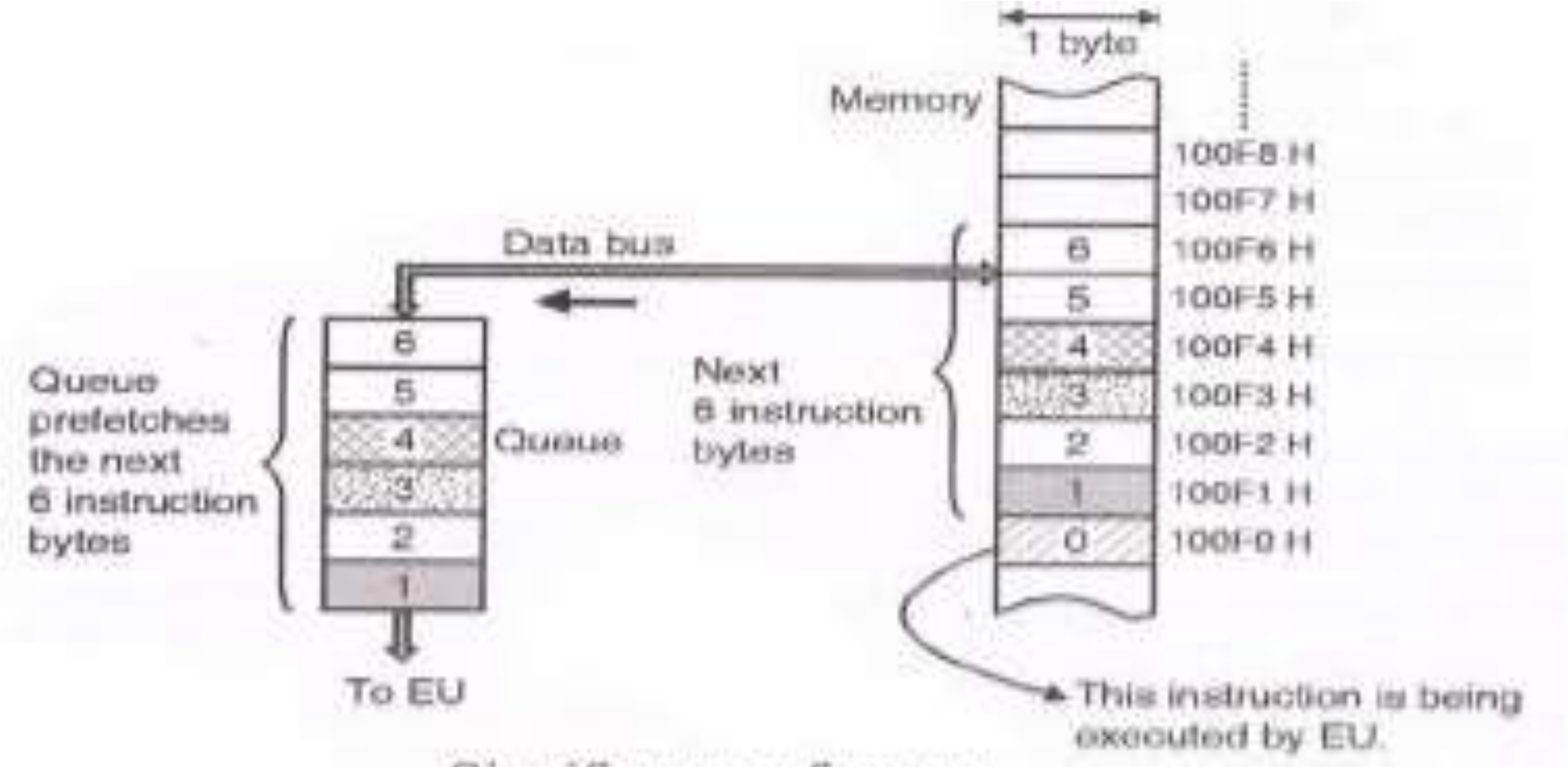
- **SS, stack segment register**. The stack is a very important component of the CPU used for temporary storage of data and addresses. Therefore, the stack has a segment address, which is contained in register SS. By default, the processor assumes that all data referenced by the stack pointer (SP) and 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.

- **ES, extra segment register**. The extra segment is exactly that: a spare segment that may be used for specifying a location in memory. 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.

Cont.. (IP and IQ)

- **Instruction Pointer (IP):**
  - Instruction Pointer holds the 16 bit address or offset of the next code byte within the code segment.
  - The value contained in the Instruction Pointer called as Offset because the value must be added to the segment base address in CS to produce the required 20 bit address.
  - CS register contains the Upper 16 bit of the starting address of the code segment in the 1 Mbyte address range the instruction pointer contains a 16 bit offset which tells wherein that 64 Kbyte code segment the next
  - instruction byte has to be fetched from.
- **Instruction Queue**
  - BIU gets upto 6 bytes of next instructions bytes and stores them in the instruction queue. 8086 Queue is only Six Byte long:
    - This is because the longest instruction in the instruction set of 8086 is six byte long.
    - Hence with a six byte long queue it is possible to pre-fetch even the longest instruction in the main program.

Cont..



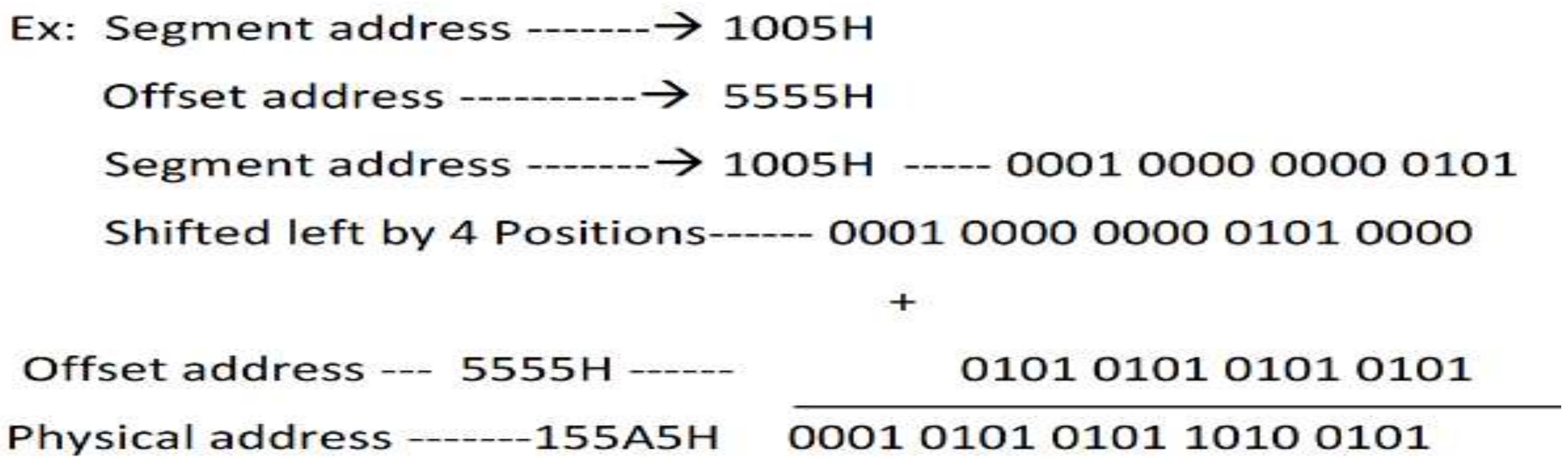
Significance of queue

- 1. As shown in the above figure, while the EU is busy in decoding the instruction corresponding to memory location 100F0, the BIU fetches the next six instruction bytes from locations 100F1 to 100F6 numbered as 1 to 6.
- 2. These instruction bytes are stored in the 6 byte queue on the first in first out (FIFO) basis.
- 3. When EU completes the execution of the existing instruction and becomes ready for the next instruction, it simply reads the instruction bytes in the sequence 1, 2.... from the Queue.
- 4. Thus the Queue will always hold the instruction bytes of the next instructions to be executed by the EU.
- 5. Queue refilling is done when there are 2 bytes are available. 8086 is an 16 bit processor, it transfer 16 bit i.e. 2 bytes in one cycle. That's the reason when Queue has a room for at least two bytes then BIU refill the queue.
- *Note: 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.*

Cont.. Physical Address

- In computing, physical address refers to a memory address or the location of a memory cell in the main memory. It is used by both hardware and software for accessing data. Software, however, does not use physical addresses directly; instead, it accesses memory using a virtual address. A hardware component known as the memory management unit (MMU) is responsible for translating a virtual address to a physical address.

- **Physical Address = Address of segment * 10H + Offset Address**
  - For example: CS=1000H and IP=2345H then calculate Physical address?
    - Physical Address= 1000H * 10H + 2345H = 10000+2345 = 12345H
  - Example 2: CS=4042H and IP=0580H then calculate Physical address?
    - Physical Address= 4042H * 10H + 0580H = 40420H + 0580H= 409A0H
  - Another method to calculate Physical Address using Binary conversion is as follows



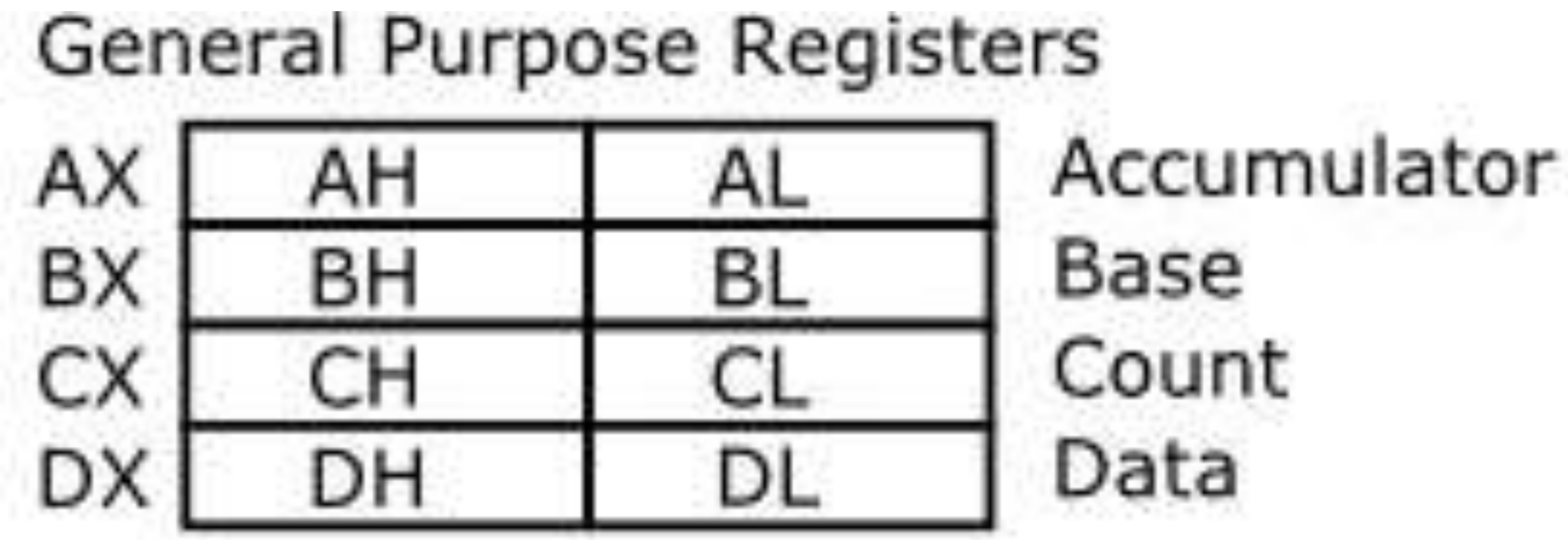
```
Ex:  Segment address -------→ 1005H
     Offset address ---------→  5555H
     Segment address -------→ 1005H  ----- 0001 0000 0000 0101
     Shifted left by 4 Positions------ 0001 0000 0000 0101 0000
                                                 +
Offset address ---  5555H ------            0101 0101 0101 0101
                                       ___________________________
Physical address -------155A5H      0001 0101 0101 1010 0101
```

EU (Execution Unit):

- The Execution unit tells the BIU where to fetch instructions or data from decodes instructions and Executes instructions.
- EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.
- Execution unit receives program instruction codes and data from the BIU, executes them and stores the results in the general registers.
- **ALU (Arithmetic and Logic Unit):** The EU unit contains a circuit board called the Arithmetic and Logic Unit. The ALU can perform arithmetic, such as; +,-,×,/ and logic such as OR, AND, NOT operations.
- **Pointers and index registers.**
- The pointers contain within the particular segments. The pointers IP, BP, SP usually contain offsets within the code segments, data and stack segments respectively
  - Stack Pointer (SP) is a 16-bit register pointing to program stack in stack segment.
  - Base Pointer (BP) is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.
  - Source Index (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.
  - Destination Index (DI) is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

Cont.. General purpose register

- There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16bit data. The valid register be used individually to store 8-bit data and can be used in pairs to store 16bit data.

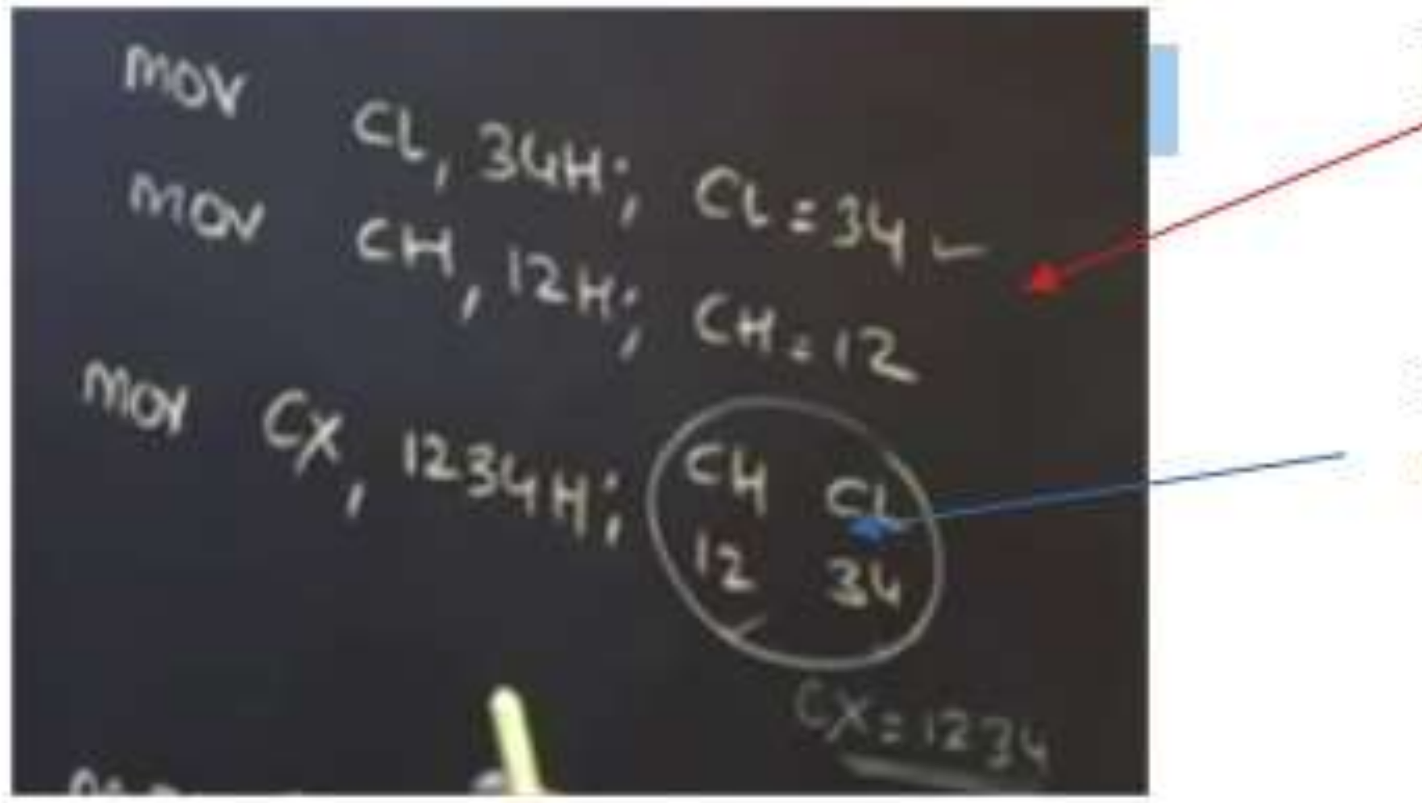


General Purpose Registers

AX	AH	AL	Accumulator
BX	BH	BL	Base
CX	CH	CL	Count
DX	DH	DL	Data

- The AX register can be accessed as a 16-bit register, or as AH an 8-bit register (H referring to the high-order byte of AX), and AL an 8-bit register (L referring to the low-order byte of AX). Similarly BX can be accessed as BH and BL, CX can be accessed as CH and CL, DX can be accessed as DH and DL.

- For example: copy data into CX register

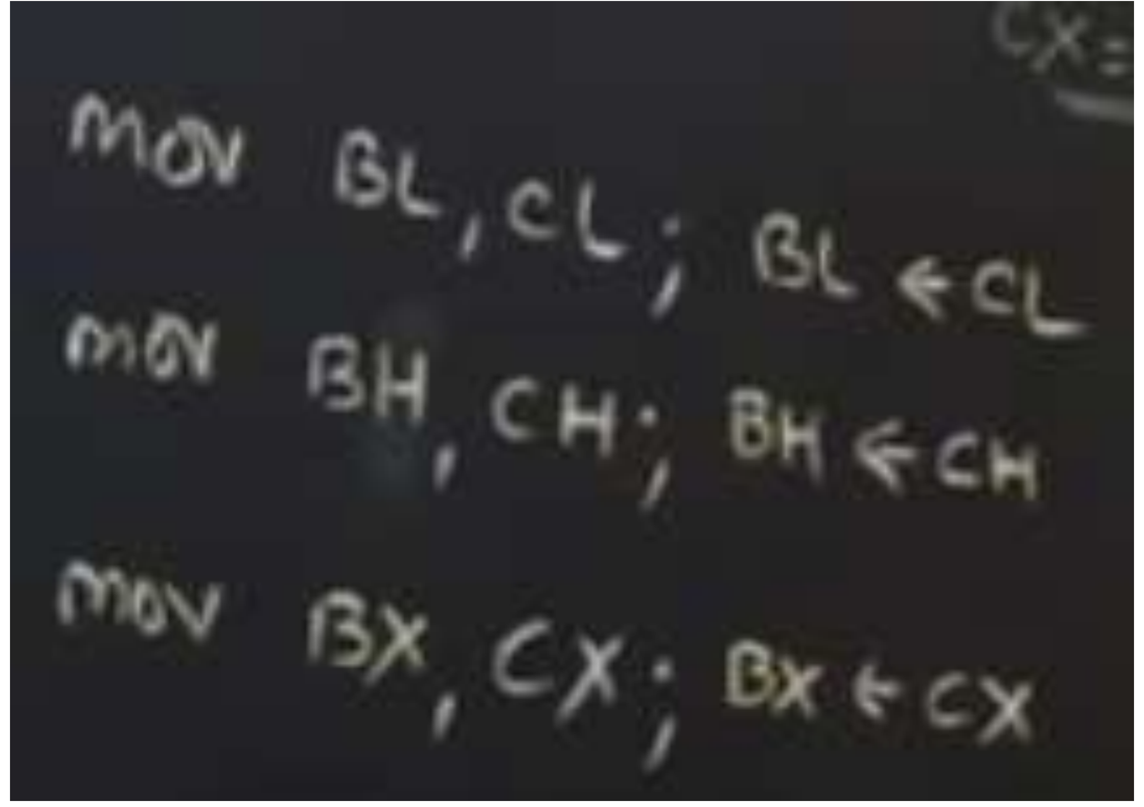


```
mov  CL, 34H;  CL=34
mov  CH, 12H;  CH=12
mov  CX, 1234H;  CH CL
                 12 34
                        CX=1234
```

Moving 8 bit data

Moving 16 bit data then 8 bit saved in CL and other in CH

Cont..



CX=

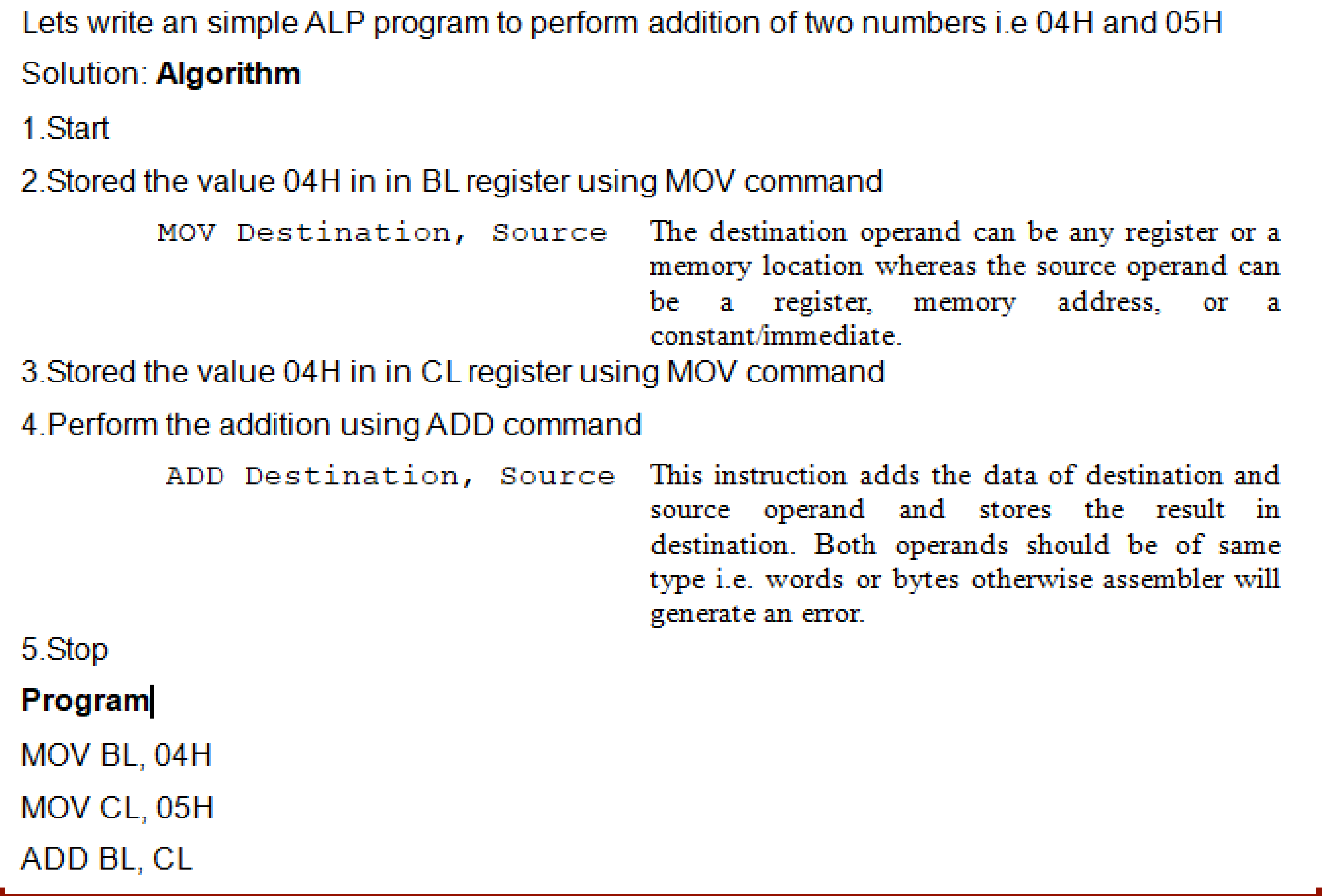
MOV BL, CL; BL ← CL

MOV BH, CH; BH ← CH

MOV BX, CX; BX ← CX

- Another example copy the data from one register to another
- The four index registers are used primarily with operations

related to the memory addressing modes of the 8086 microprocessor



Lets write an simple ALP program to perform addition of two numbers i.e 04H and 05H

Solution: **Algorithm**

1.Start

2.Stored the value 04H in in BL register using MOV command

`MOV Destination, Source`	The destination operand can be any register or a memory location whereas the source operand can be a register, memory address, or a constant/immediate.

3.Stored the value 04H in in CL register using MOV command

4.Perform the addition using ADD command

`ADD Destination, Source`	This instruction adds the data of destination and source operand and stores the result in destination. Both operands should be of same type i.e. words or bytes otherwise assembler will generate an error.

5.Stop

Program

```
MOV BL, 04H
MOV CL, 05H
ADD BL, CL
```

Control Unit:

- The control unit in 8086 microprocessor produces control signal after decoding the opcode to inform the general purpose register to release the value stored in it. And it also signals the ALU to perform the desired operation.

- Note: Here Control unit Decode opcode and add an operand. For example ADD BL, CL whole opcode is going to decode. for this instruction ADD BL, 05H here opcode only created for ADD BL and 05H is operand going to add

- Control unit having one incoming and one outgoing line, Here incoming line used when opcode so control unit decode it. And second outgoing line send operand so operation going to perform.

- **Operands register**

- It is a temporary register and is used by the processor to hold the temporary values at the time of operation.

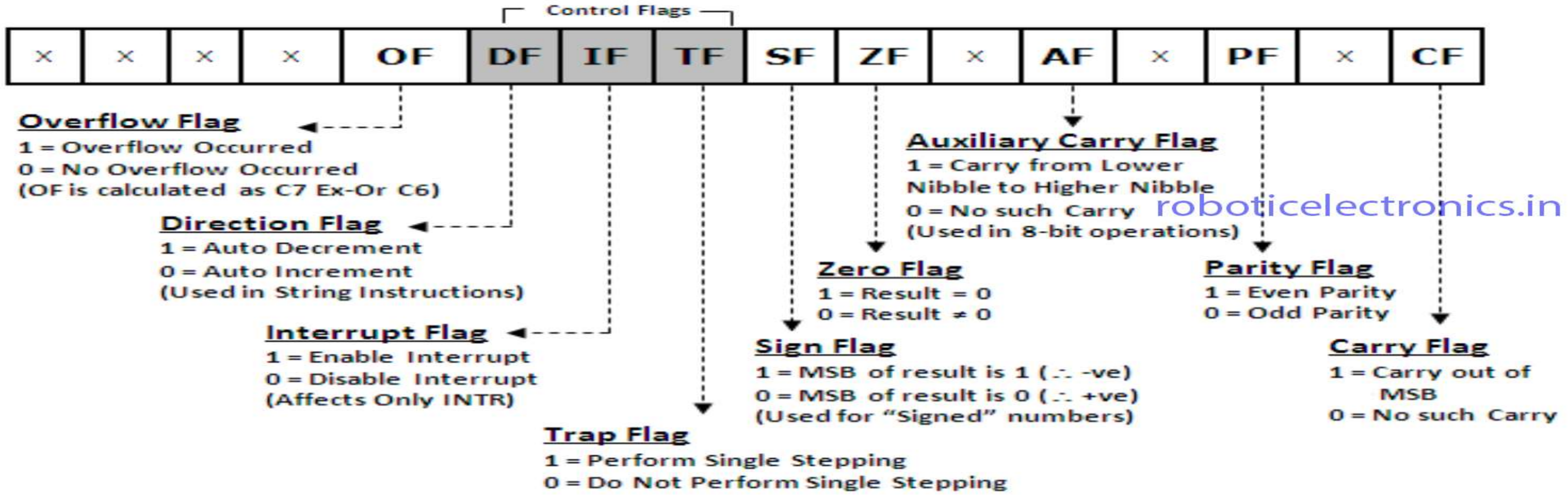
- The reason behind two separate sections for BIU and EU in the architecture of 8086 is to perform fetching and decoding-executing simultaneously

- for example: XCHG BX, CX this line is used to exchange value of BX and CX register.

- To do this operation processor need and temporary register that is operand register. First CX register saved value to temp register then BX value copy in CX register and then temp register value copy in BX register.

Flags:

- Flag register holds the status of the result generated by the ALU. It has several flags that show the different conditions or status of the result.
- It is an16 bit flag register is used in 8086. It is divided into two parts .
    - Condition code or status flags
    - Machine control flags
- Figure below shows the details of the 16 bit flag register of 8086 CPU. It consists of 9 active flags out of 16. The remaining 7 flags marked 'U' are undefined flags.



Overflow Flag
1 = Overflow Occurred
0 = No Overflow Occurred
(OF is calculated as C7 Ex-Or C6)

Direction Flag
1 = Auto Decrement
0 = Auto Increment
(Used in String Instructions)

Interrupt Flag
1 = Enable Interrupt
0 = Disable Interrupt
(Affects Only INTR)

Trap Flag
1 = Perform Single Stepping
0 = Do Not Perform Single Stepping

Sign Flag
1 = MSB of result is 1 (∴ -ve)
0 = MSB of result is 0 (∴ +ve)
(Used for "Signed" numbers)

Zero Flag
1 = Result = 0
0 = Result ≠ 0

Auxiliary Carry Flag
1 = Carry from Lower Nibble to Higher Nibble
0 = No such Carry
(Used in 8-bit operations)

Parity Flag
1 = Even Parity
0 = Odd Parity

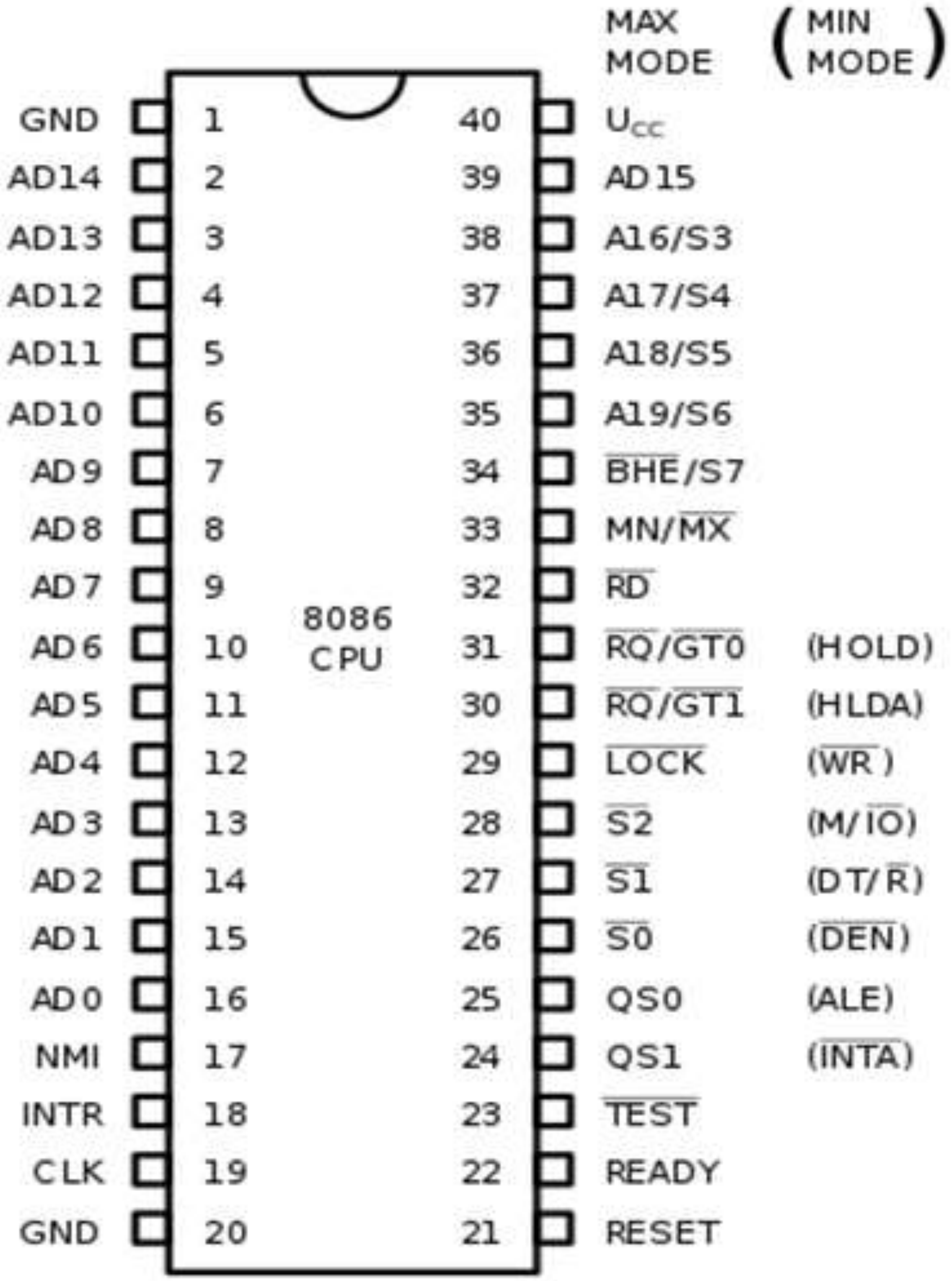
Carry Flag
1 = Carry out of MSB
0 = No such Carry

Cont..

- **SF- Sign Flag:** This flag is set, when the result of any computation is negative. For signed computations the sign flag equals the MSB of the result.
- **ZF- Zero Flag:** This flag is set, if the result of the computation or comparison performed by the previous instruction is zero.
- **PF- Parity Flag:** This flag is set to 1, if the lower byte of the result contains even number of 1's.
- **CF- Carry Flag:** This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction.
- **AF-Auxilary Carry Flag:** This is set, if there is a carry from the lowest nibble, i.e, bit three during addition, or borrow for the lowest nibble, i.e, bit three, during subtraction.
- **OF- Over flow Flag:** This flag is set, if an overflow occurs, i.e, if the result of a signed operation is large enough to accommodate in a destination register. The result is of more than 7-bits in size in case of 8-bit signed operation and more than 15-bits in size in case of 16-bit sign operations, and then the overflow will be set.
- **TF- Tarp Flag:** If this flag is set, the processor enters the single step execution mode. The processor executes the current instruction and the control is transferred to the Trap interrupt service routine.
- **IF- Interrupt Flag:** If this flag is set, the mask able interrupts are recognized by the CPU, otherwise they are ignored.
- **D- Direction Flag:** This is used by string manipulation instructions. If this flag bit is '0', the string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e., auto decrementing mode.

8086 Pin diagram

- **Definition: 8086 is a 16-bit microprocessor and was created by Intel in 1978. Like the pin configuration of 8085 microprocessor, the 8086 microprocessor also contains 40 pins dual in line.**



- The minimum mode is selected by applying logic 1 to the MX/MN input. It is typically used for smaller single microprocessor systems.

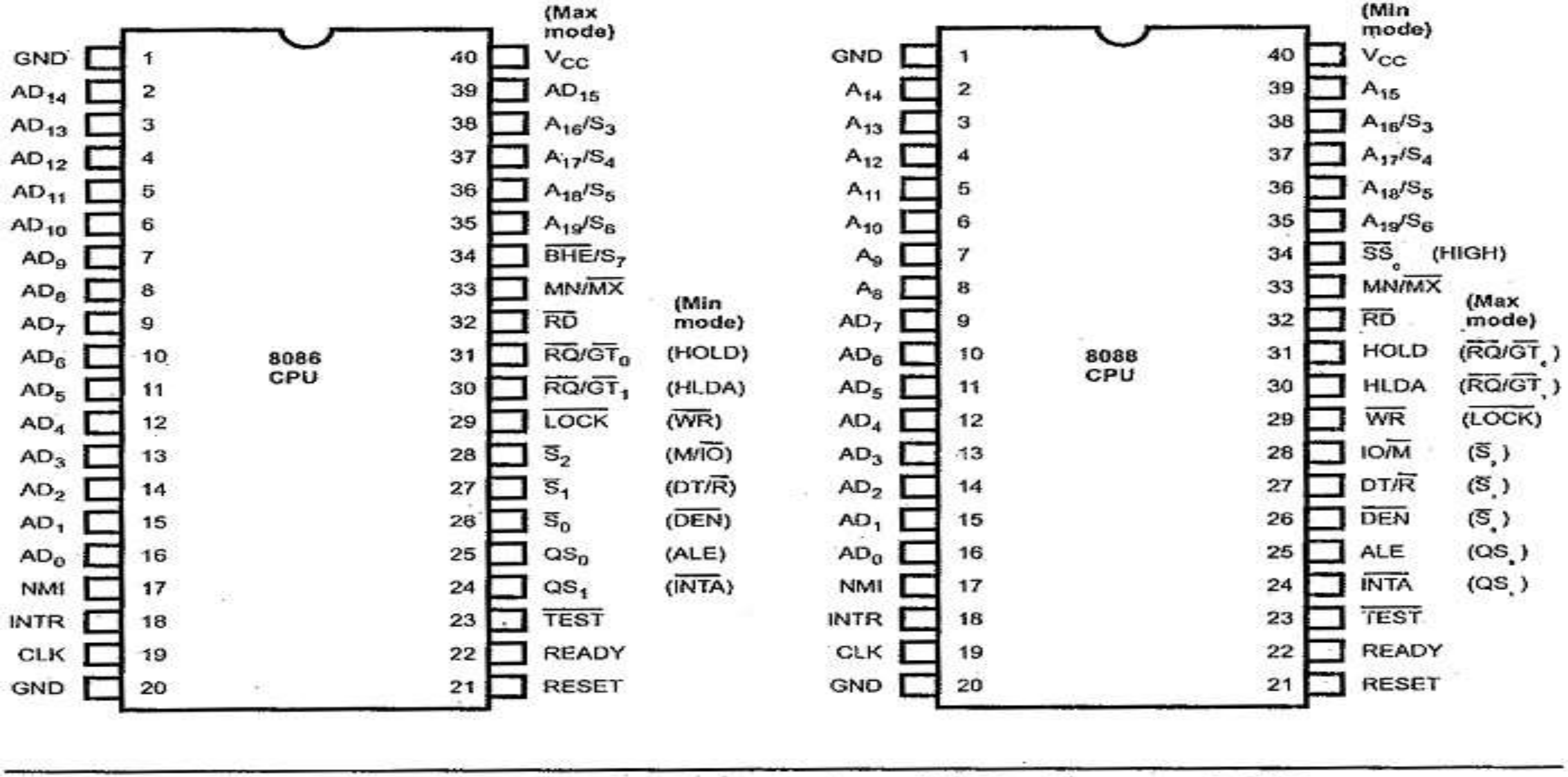
- The maximum mode is selected by applying logic 0 to the MX/MN input. It is typically used for larger multiple microprocessor systems.

- Depending on the mode of operation selected, the assignments for a number of the pins on the microprocessor package are changed. The pin functions specified in parentheses pertain to the maximum-mode

- Address bus size is 20 bit and Data bus is 16 bit. Address bus recognized

- Pin description

8088



(a) Pin diagram of 8086 Fig. 10.1 (b) Pin diagram of 8088

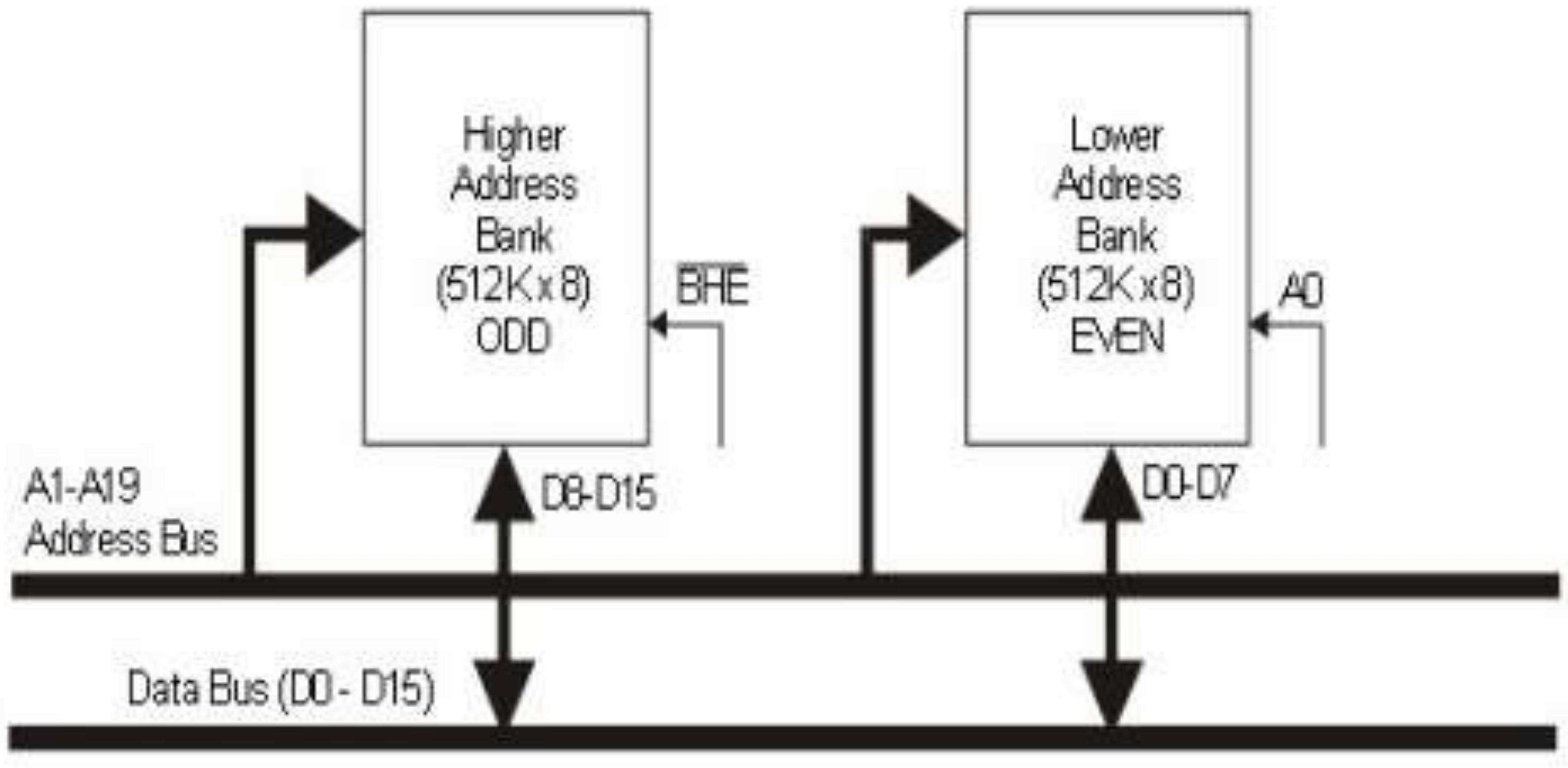
- The 8086 is a 16-bit microprocessor with a 16-bit data bus, and the 8088 is a 16-bit microprocessor with an 8-bit data bus. The pin-out shows, the 8086 has pin connections AD0-AD15 and the 8088 has pin connections AD0-AD7. There is one more minor difference in one of the control signals. The 8086 has an M/IO pin, and the 8088 has an 10/M pin. The only hardware difference appears on pin 34 of both chips : on the 8086 it is a BHE/S7 pin, while on the 8088 it is a SS0 pin.

8086 Memory Banking

- The 8086 microprocessor uses a 20-bit address to access memory. With 20-bit address the processor can generate 2^20 = 1 Mega address.

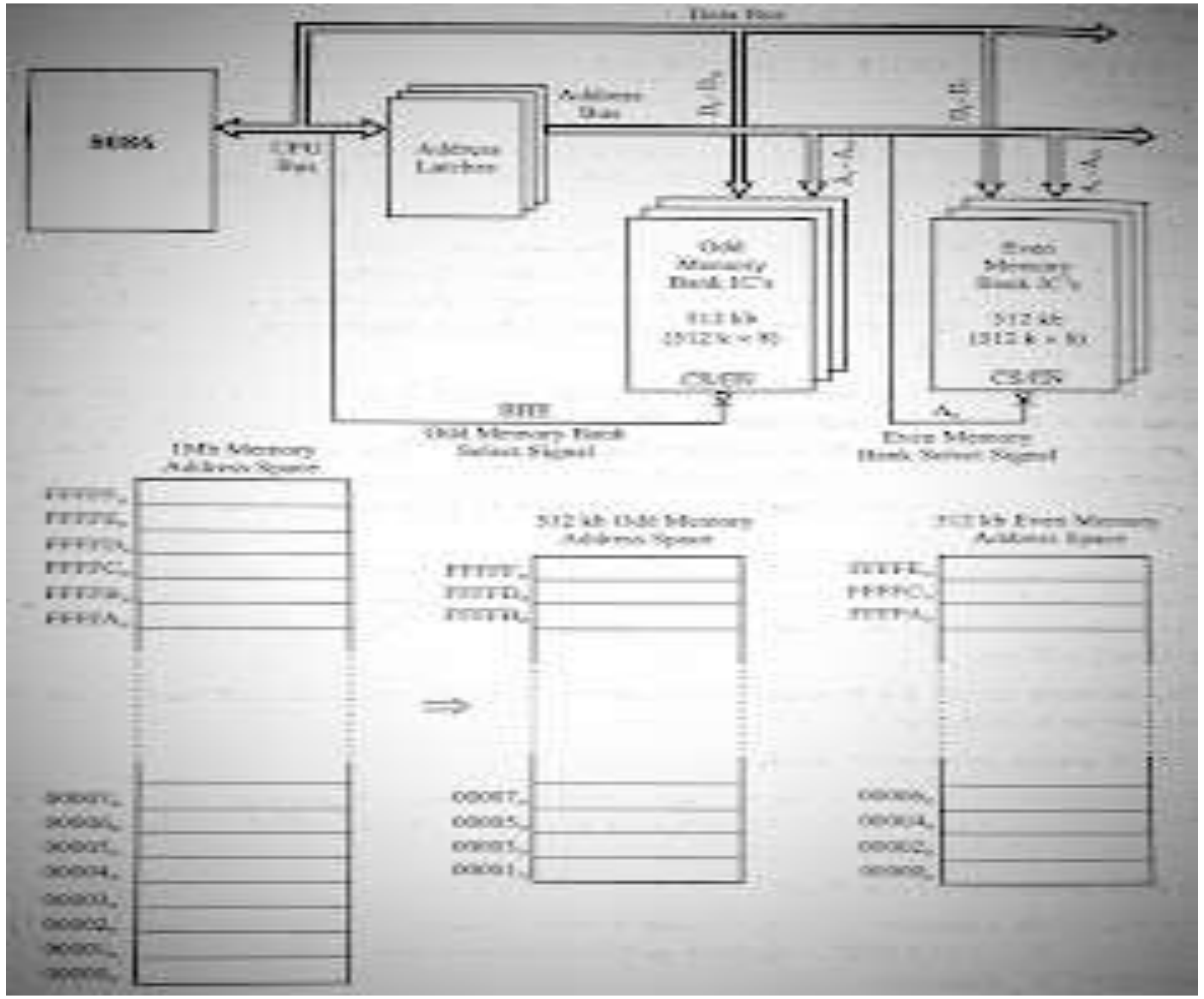
- The basic memory word size of the memories used in the 8086 system is 8-bit or 1-byte (i.e., in one memory location an 8-bit binary information can be stored). Hence, the physical memory space of the 8086 is 1Mb (1 Mega-byte).

- For the programmer, the 8086 memory address space is a sequence of one mega-byte in which one location stores an 8-bit binary code/data and two consecutive locations store 16-bit binary code/data. But physically (i.e., in the hardware), the 1Mb memory space is divided into two banks of 512kb (512kb + 512kb = 1Mb). The two memory banks are called Even (or Lower) bank and Odd (or Upper) bank. The organization of even and odd memory banks in the 8086-based system is shown in Figure.



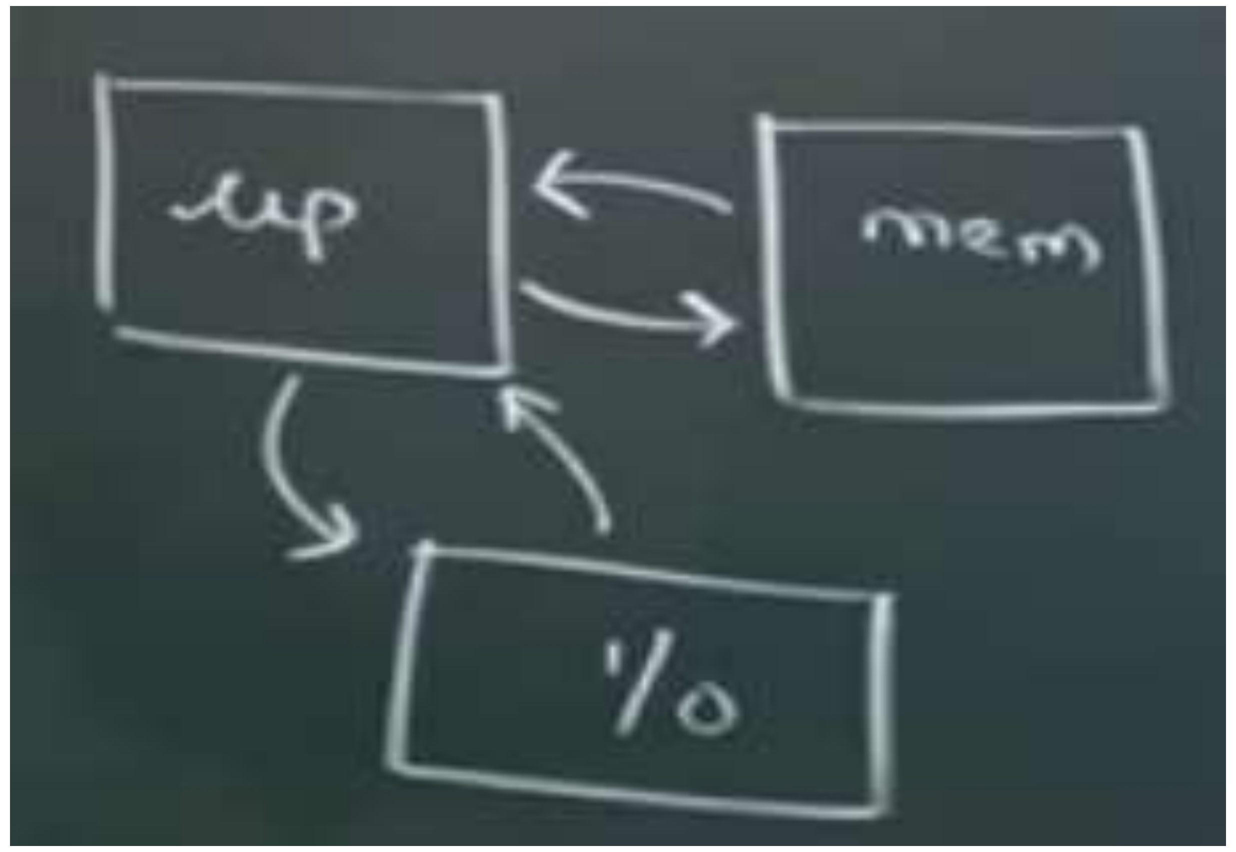
Cont..

- The data lines D0-D7 are connected to even bank and the data lines D8 -D15 are connected to odd bank. The even memory bank is selected by the address line Ao and the odd memory bank is selected by the control signal BHE. The memory banks are selected when these signals are low(active low). Any memory location in the memory bank is selected by the address line A1 to A19.
- The organization of memory into two banks and providing bank select signals allows the programmer to read/write the byte (8-bit) operand in any memory address through 16-bit data bus. It also allows the programmer to read/write the word (16-bit) operand starting from even address or odd address.



Interfacing With 8086/8088

- **Interface is the path for communication between two components. Interfacing is of two types, memory interfacing and I/O interfacing.**



- Memory Interfacing with 8086/8088: [size of memory is 1MB as Address bus 20 bit]
- Memories are made up of **registers**. Each register in the memory is one storage location. Each location **is identified by an address**. Generally, the total number of bits that a memory can store is its capacity. Most of the types the capacity is specified in terms of bytes (group of eight bits).
- Each register consists of storage elements (flip-flops or capacitors in semiconductor memories and magnetic domain in magnetic storage), each of which stores one bit of data.

Cont..

- A storage element is called a cell.
- It available in many sizes like 1K, 32K, 64K etc....

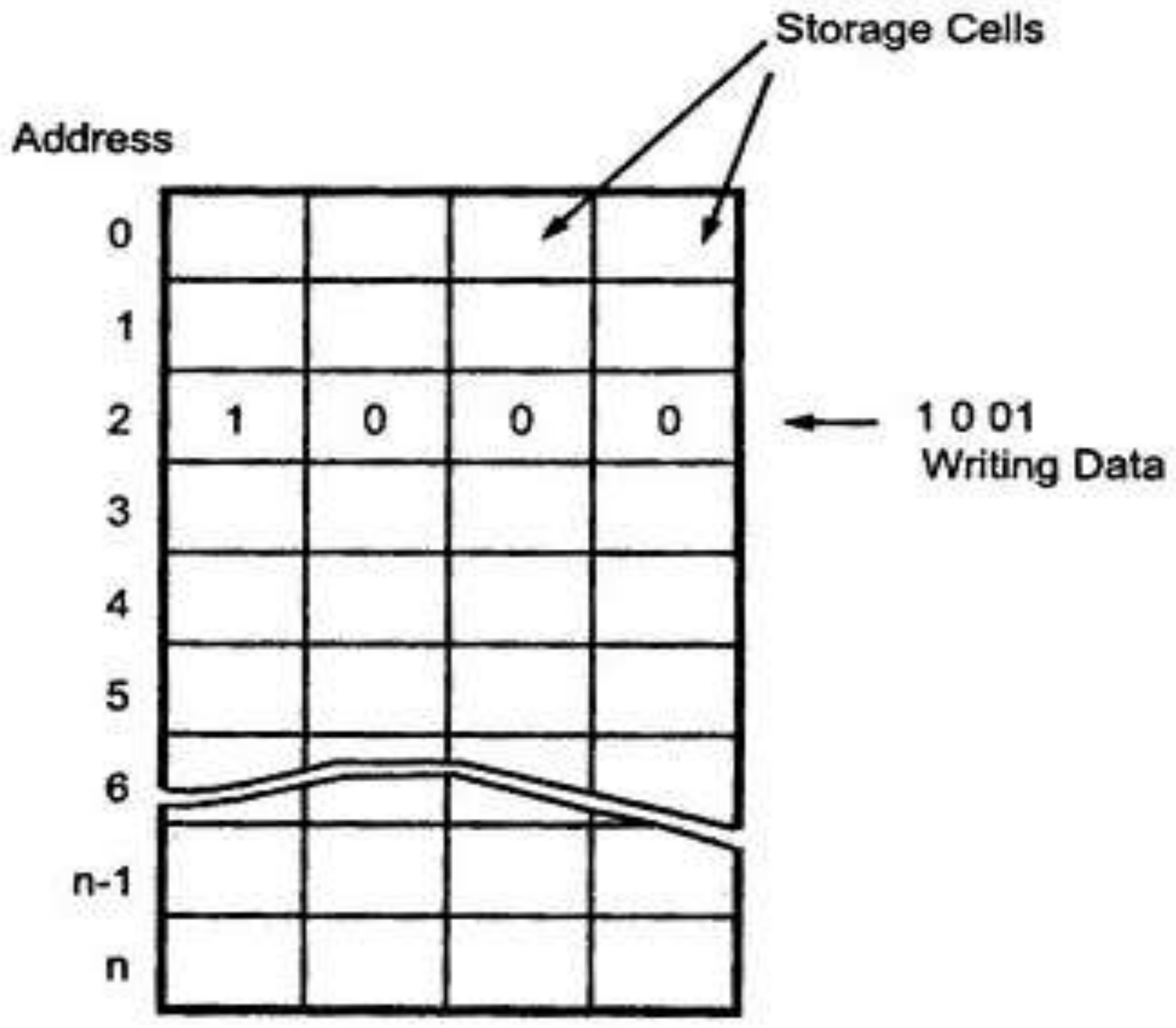


Fig. 3.67 (a) Write operation

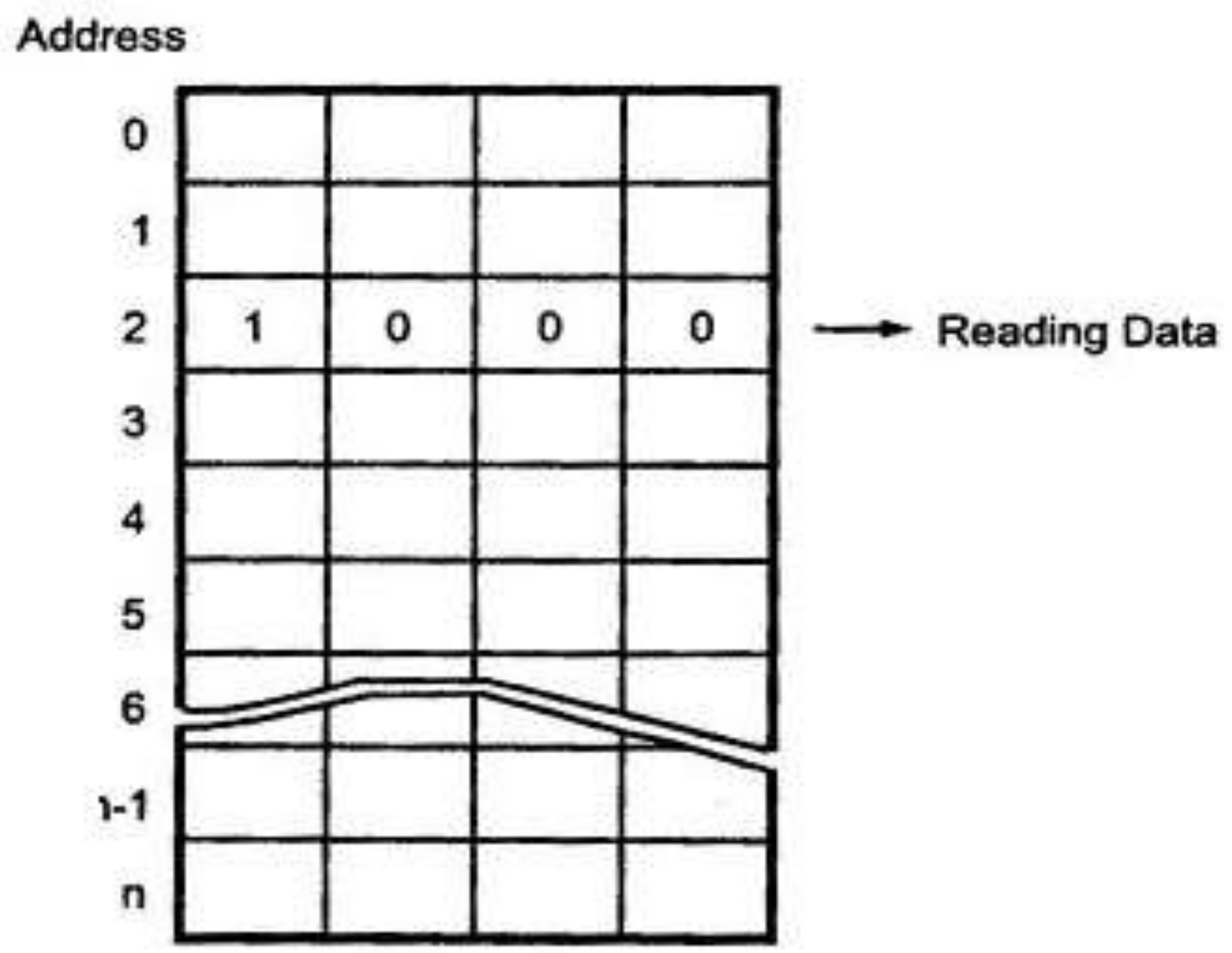
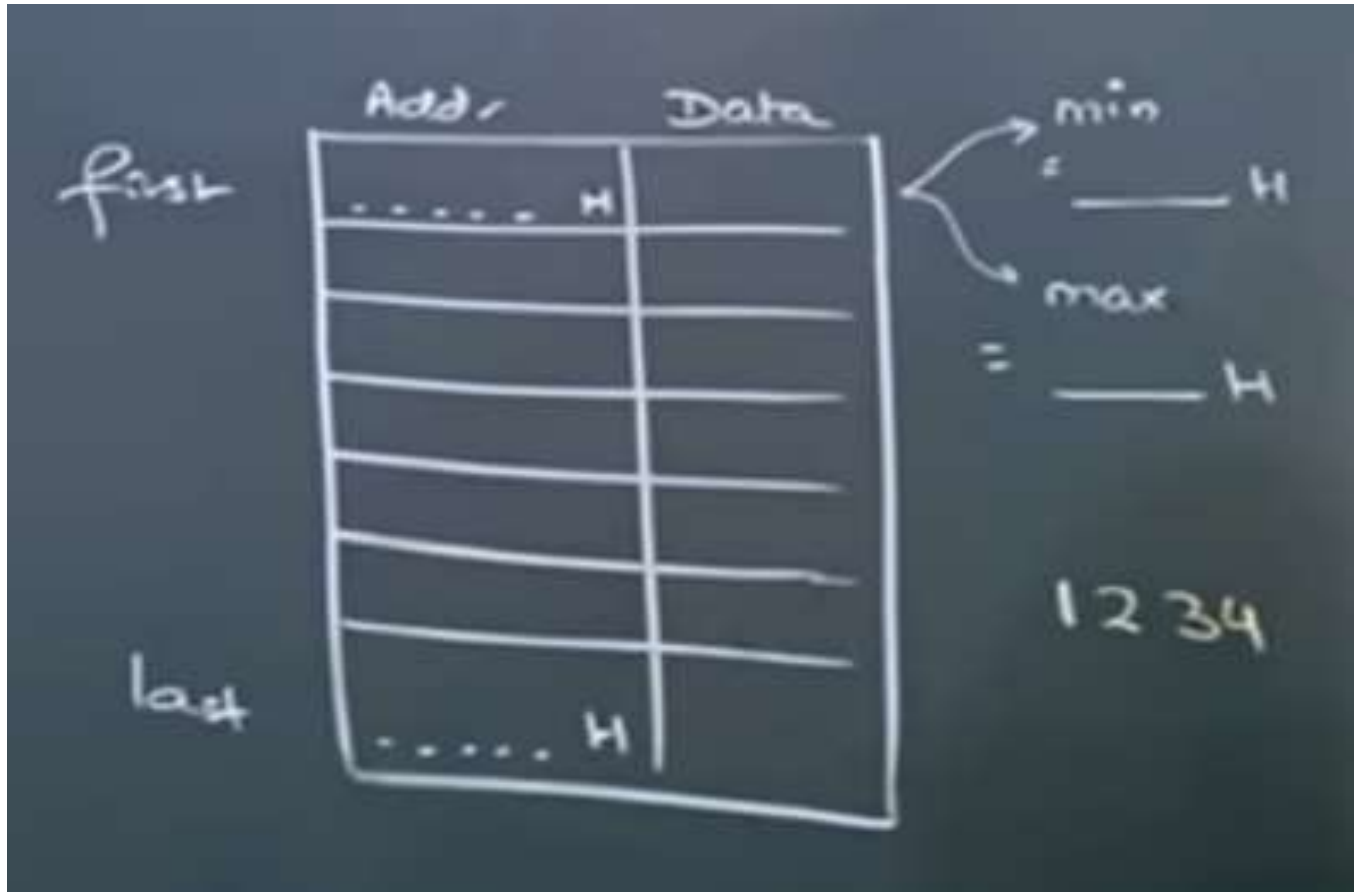


Fig. 3.67 (b) Read operation

- The data stored in a memory by a process called writing and are retrieved from the memory by a process called reading.
- **Memory Representation in 8086**



Cont..

What is the size of address bus?

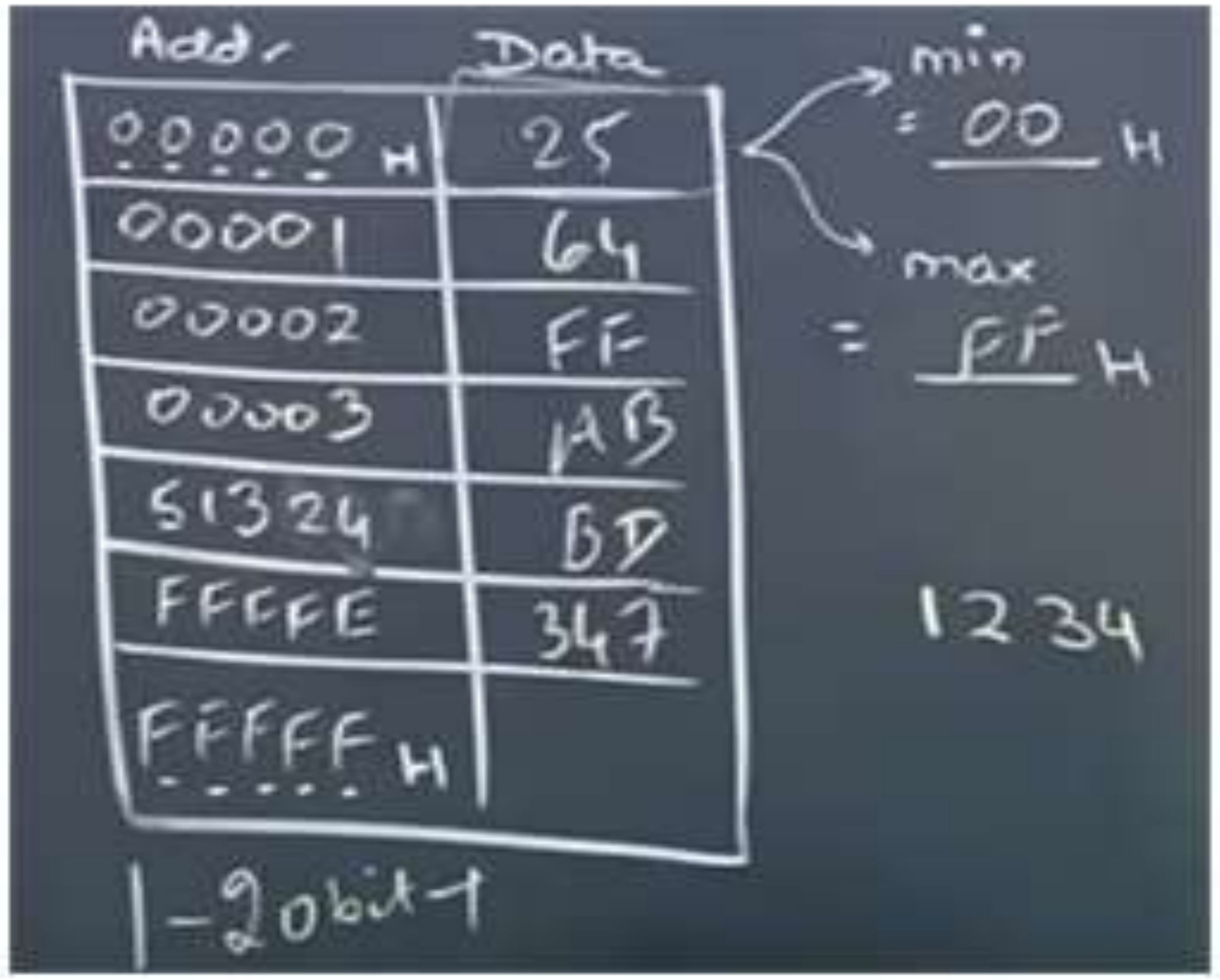
20 bit i.e. $2^{20} = 2^{10} * 2^{10}$ =1K*1K=1MB

So what is the Starting point of memory and Ending Point of memory?

00000H and FFFFFH

What is the minimum and maximum value of data can be saved in one location?

Min value in Hex = 00H and Max Value in Hex is = FFH because 1 memory location carry only 1 byte i.e 8 Bit data.



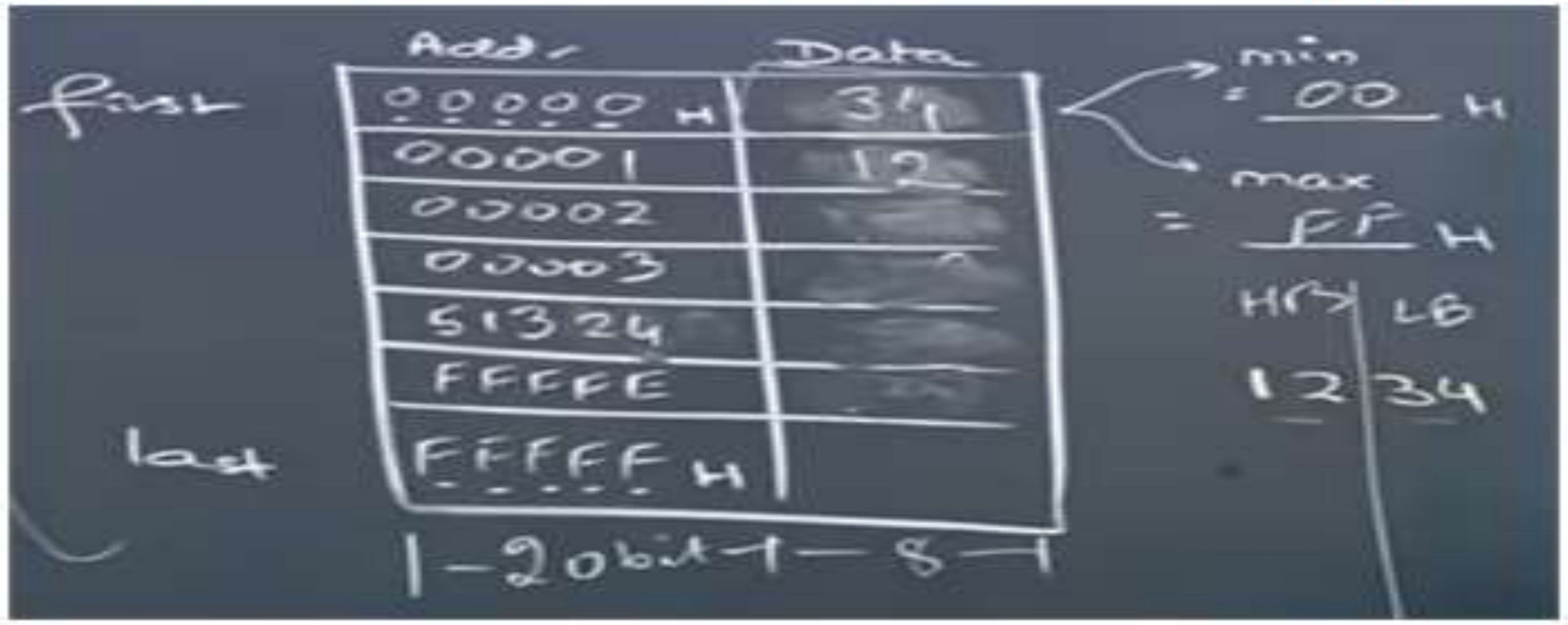
Addr	Data
00000 H	25
00001	64
00002	FF
00003	AB
51324	BD
FFFFE	347
FFFFF H	

347 is invalid storage in memory because 1 memory location carry only 1 byte i.e 8 Bit data.

Cont..

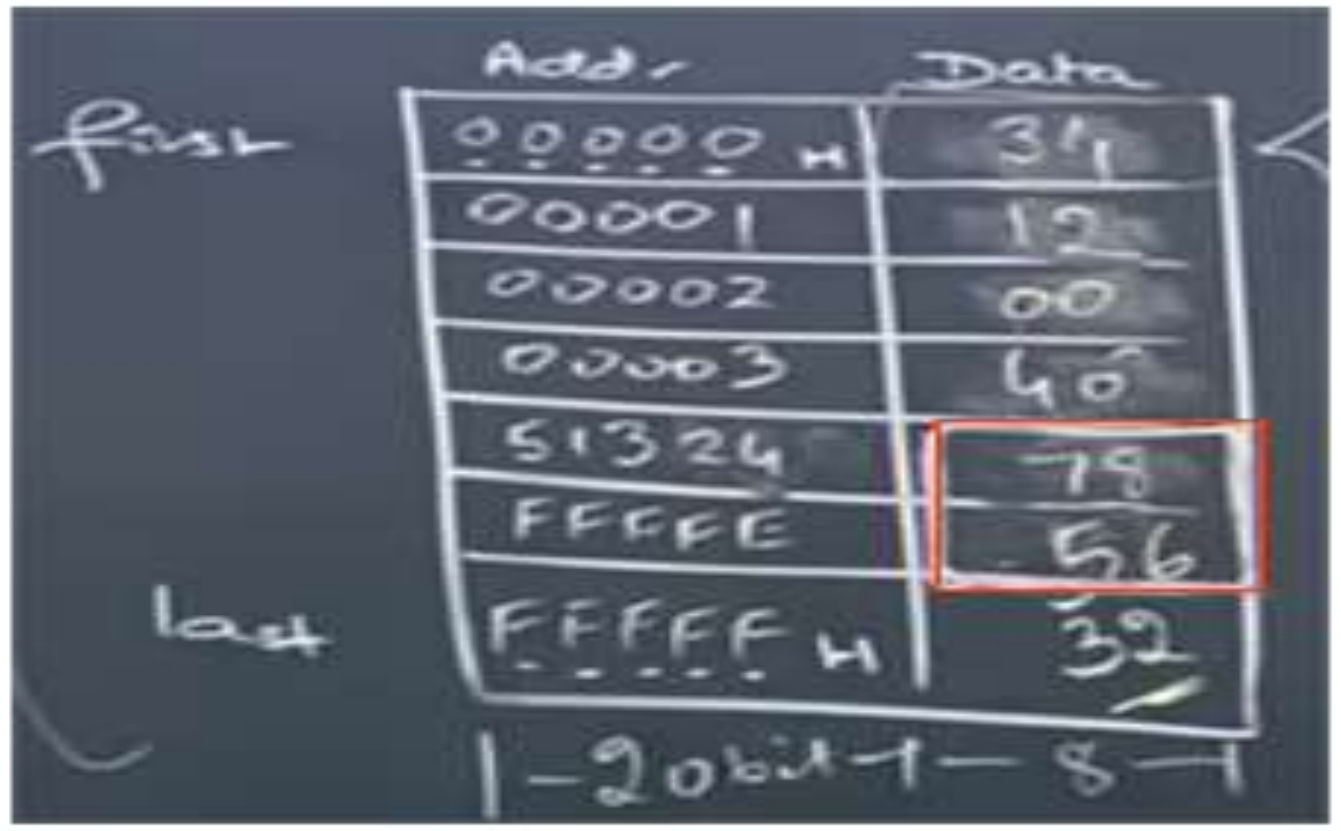
How the 1234 going to store in memory?

As 1234 is 16 bit number then 1234 need two memory locations to store. It is going to store in following way in memory as it matter in reading and writing process. Here **Little Endian Rule** is followed to save this data in memory. Lower byte is going to saved at lower address and Higher byte is going to saved higher address. Fig shows the saving of 1234 in memory



Add.	Data
00000 H	34
00001	12
00002	
00003	
51324	
FFFFE	
FFFFF H	

Reading the data from memory is Higher Address first and Lower Address second. Reading of following two memory location is 5678H

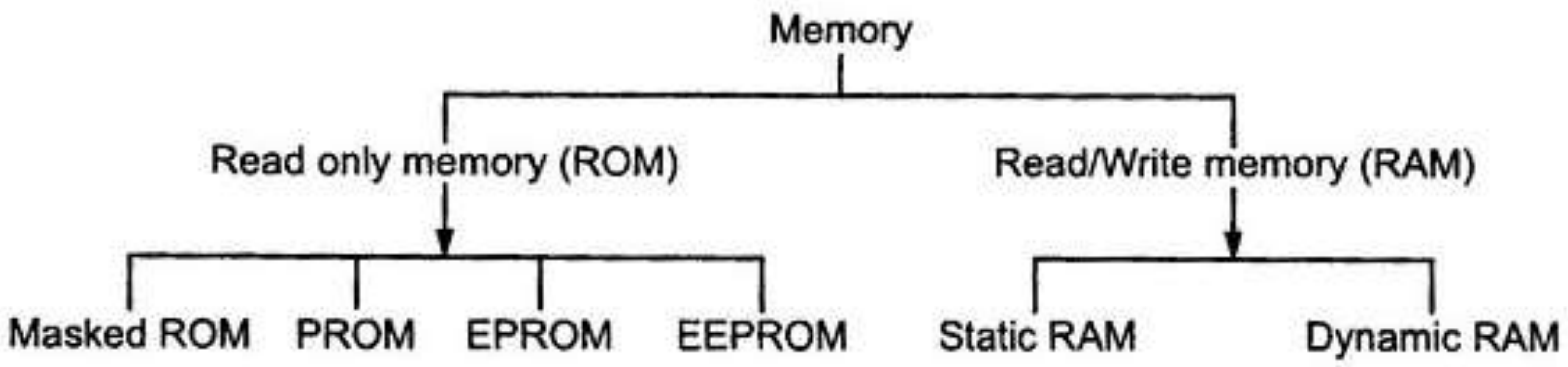


Add.	Data
00000 H	34
00001	12
00002	00
00003	40
51324	78
FFFFE	56
FFFFF H	32

Figure shows the classification of semiconductor memories.

Semiconductor Memory

- The semiconductor memory is classified as volatile and non-volatile memory.
- Volatile memories are those memories that store the data temporarily. More specifically we can say that data is stored in volatile memory only till the duration power supply to the IC is ON. And once the supply gets OFF then the stored data gets lost.
- As against in non-volatile type of memory, the data retained in the memory even if the power supply is OFF. Thus we can say that in non-volatile memory the data is stored on a permanent basis.



Read-Only Memory (ROM):

- **ROM: It stands for Read-Only Memory. It is a memory array that is permanently programmed by the manufacturer or programmer only once. Hence its data cannot be changed by the processor once it is programmed.**

- Thus the processor can only read the data present in this memory hence called read-only memory or fixed memory.
- The process of loading the data in the ROM is known as programming. The way in which ROM is programmed further classifies it.
- **Custom programmed ROM (ROM):** These memories are programmed by the manufacturer at the time of its fabrication. Hence is stored permanently.
- **Programmable ROM (PROM):** PROM’s are programmed by the user but these offers re-programmability after erasing previously loaded contents. There exist two methods by which the contents of the ROM are erased.

Cont..

- **EPROM:** It stands Erasable Programmable Read-Only Memory. The EPROM offers re-programming, by erasing the previously stored data by making use of ultraviolet rays. The memory-erasing time lies between 10 to 30 minutes.

- Basically, the electrons in the isolated gate of MOS transistor of memory cells get removed when irradiated with ultraviolet rays. Thereby allowing removal of stored data in the memory cell through the control gate.

- Further, in order to reprogram the EPROM, the memory chip is inserted in the PROM programmer socket. A PC provides interfacing to the PROM programmer and the programmer installs the information to be loaded in the chip from the personal computer.

- **EEPROM:** It is an abbreviation used for electrically erasable programmable read-only memory. Initially, the data in E2PROM is erased by applying external voltage at the erase pin of the chip. But this somewhat increases the complexity of the overall system.

- So, the latest versions provide incorporation of supply voltage within the chip.

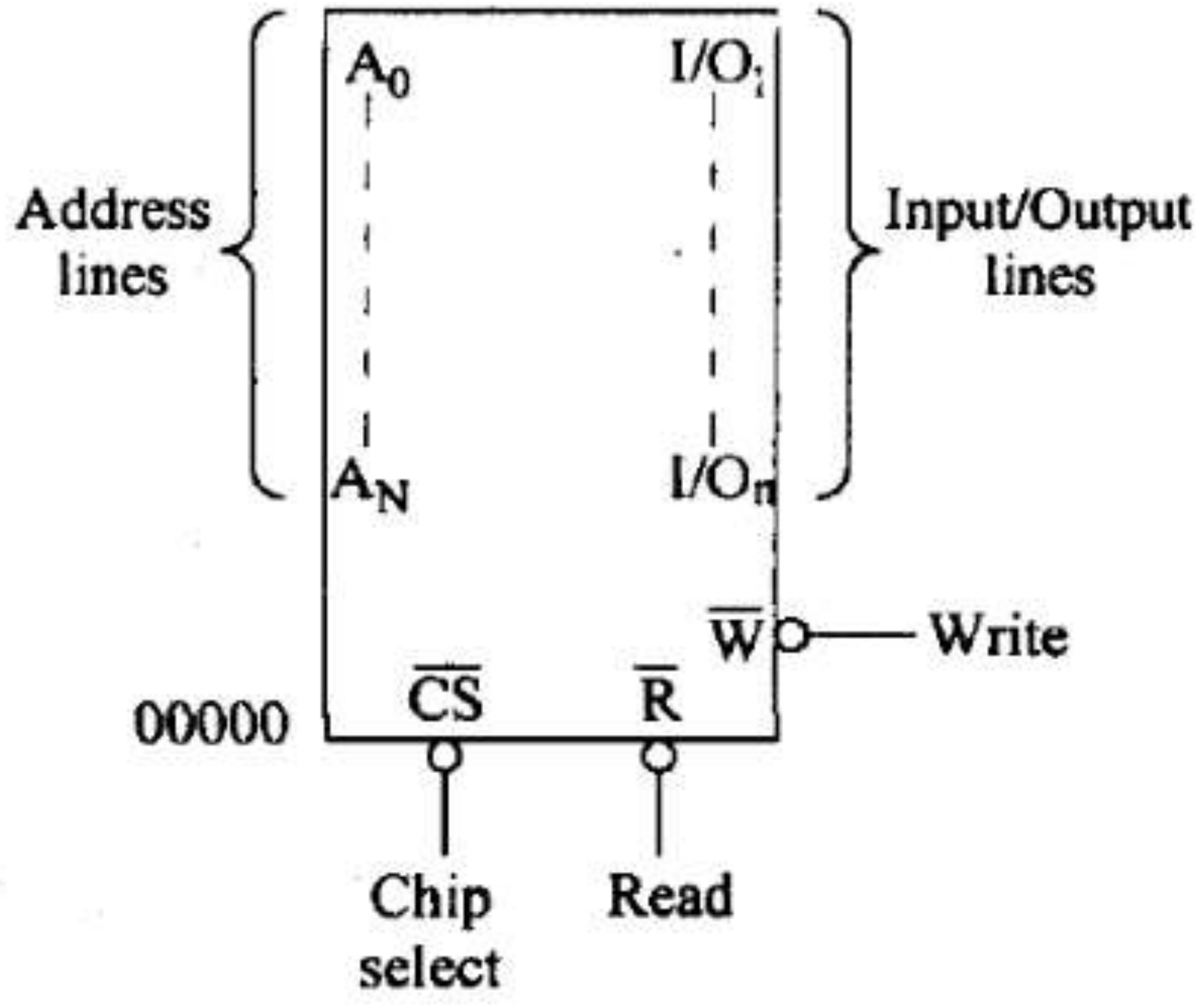
Random Access Memory(RAM):

- RAM stands for random access memory, and it's one of the most fundamental elements of computing. RAM is a temporary memory bank where your computer stores data it needs to retrieve quickly. RAM keeps data easily accessible so your processor can quickly find it without having to go into long-term storage to complete immediate processing tasks.

- Every computing device has RAM, whether it's a desktop computer (running Windows, MacOS, or Linux), a tablet or smartphone (running Android or iOS), or even an IoT computing device (like a smart TV).
- Nearly all computers have a way of storing information for longer-term access, too. But the memory needed to run the process you're currently working on is stored and accessed in your computer's RAM.

Cont..

- RAM can be classified into two types, namely Static RAM and Dynamic RAM.
- **SRAM:** It is an abbreviation for Static Random Access Memory. SRAM has an array of flip-flops that are used to store the data. The memory cells consist of flip flops that hold the data until the power supply is on.
- **DRAM:** It is an acronym for Dynamic Random Access Memory. It is also a read/write memory that stores the data in the form of charges in the capacitor and transistor pair present in the memory cell. However, the stored charge takes some milliseconds to get dissipated therefore periodic refreshing of the elements is required, in order to have random access feature.
- RAM contains temporary data and software programs generally for different applications.
- Fig shows the schematic representation of memory



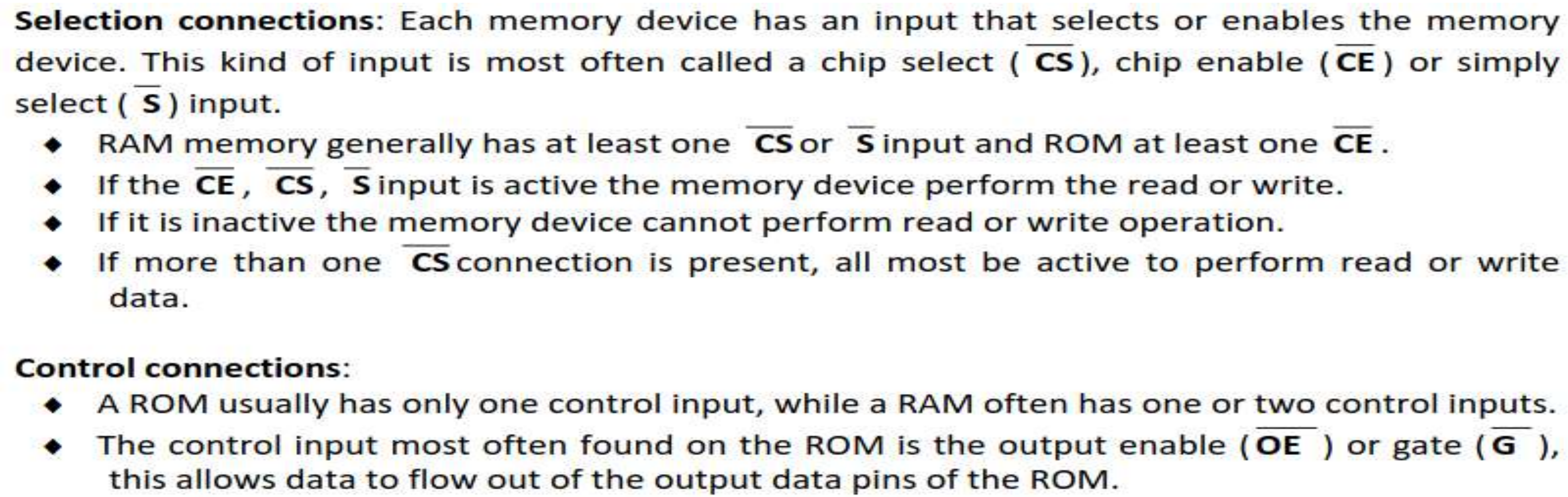
Cont..

- Memory device must contain address lines , Input, output lines, selection input and control input to perform read or write operation.
- The number of address lines indicates the total memory capacity of the memory device. A 1K memory requires 10 address lines A0-A9. Similarly a 1MB requires 20 lines A0-A19 (in the case of 8086).
- The memory devices may have separate I/O lines or a common set of bidirectional I/O lines. Using these lines data can be transferred in either direction.
- These lines are labeled as I/O ... I/On or DO ....Dn. The size of a memory location is dependent upon the number of data bits.
- If the numbers of data lines are eight ,then 8 bits or 1 byte of data can be stored in each location. Similarly if numbers of data bits are 16, then the memory size is 2 bytes. For example 2K x 8 indicates there are 2048 memory locations and each memory location can store 8 bits of data.
- Memory devices may contain one or more inputs which are used to select the memory device or to enable the memory device. This pin is denoted by CS (Chip select) or CE (Chip enable). When this pin is at logic '0' then only the memory device performs a read or a write operation. If this pin is at logic ‘1’ the memory chip is disabled. If there are more than one CS input then all these pins must be activated to perform read or write operation.
- All memory devices will have one or more control inputs. When ROM is used , OE (output enable) pin allows data to flow out of the output data pins. To perform this task both CS and OE must be active. A RAM contains one or two control inputs. They are R /W or RD and WR . If there is only one input R/W then it performs read operation when R/W pin is at logic 1. If it is at logic 0 it performs write operation.

MEMORY INTERFACE USING RAMS, EPROMS AND EEPROMS

- **The general procedure of static memory interfacing with 8086 is briefly described as follows:**
  - Arrange the available memory chips so as to obtain 16-bit data bus width. The upper 8-bit bank is called ‘odd address memory bank’ and the lower 8-bit bank is called ‘even address memory bank’.
  - Connect available memory address lines of memory chips with those of the microprocessor and also connect the memory RD and WR inputs to the corresponding processor control signals. Connect the 16-bit data bus of the memory bank with that of the microprocessor 8086.
  - The remaining address lines of the microprocessor, BHE and A0 are used for decoding the required chip select signals for the odd and even memory banks. CS of memory is derived from the O/P of the decoding circuit.
- The address map of the system should be continuous as far as possible, i.e. there should be no windows in the map. A memory location should have a single address corresponding to it, i.e. absolute decoding should be preferred, and minimum hardware should be used for decoding. Mostly, linear decoding are used to minimise the required hardware.

Cont..

- Pin connections common to all memory devices are:
- Address connections: All memory devices have address inputs that select a memory location within the memory device. Address inputs are labeled from A0-An
- Data connections: ll memory devices have a set of data outputs or input/outputs. Today many of them have bi-directional common I/O pins.
- 

  Selection connections: Each memory device has an input that selects or enables the memory device. This kind of input is most often called a chip select ($\overline{\textbf{CS}}$), chip enable ($\overline{\textbf{CE}}$) or simply select ($\overline{\textbf{S}}$) input.

  - RAM memory generally has at least one $\overline{\textbf{CS}}$ or $\overline{\textbf{S}}$ input and ROM at least one $\overline{\textbf{CE}}$.
  - If the $\overline{\textbf{CE}}$, $\overline{\textbf{CS}}$, $\overline{\textbf{S}}$ input is active the memory device perform the read or write.
  - If it is inactive the memory device cannot perform read or write operation.
  - If more than one $\overline{\textbf{CS}}$ connection is present, all most be active to perform read or write data.

  Control connections:

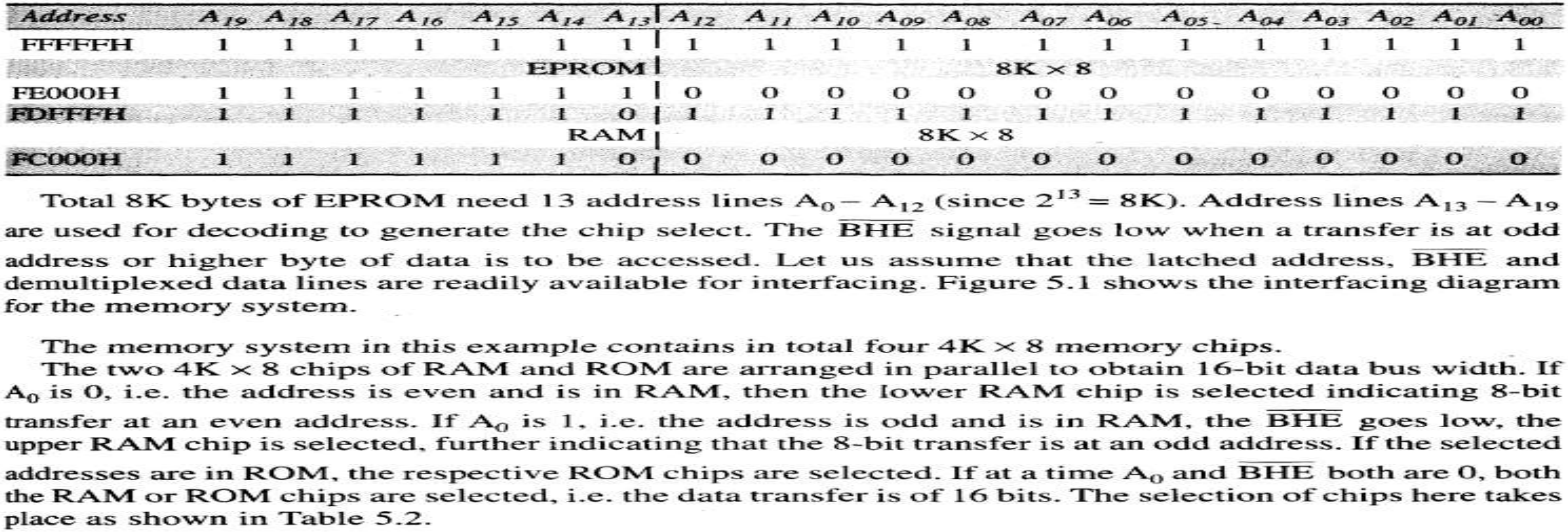
  - A ROM usually has only one control input, while a RAM often has one or two control inputs.
  - The control input most often found on the ROM is the output enable ($\overline{\textbf{OE}}$) or gate ($\overline{\textbf{G}}$), this allows data to flow out of the output data pins of the ROM.

Lets understand the example: Interface 8KB of EPROM and 8KB of RAM with 8086.

- Step 1: We have to select appropriate memory location out of 1MB if location not given.
- So what is the Starting point of memory and Ending Point of memory?

00000H and FFFFFH because we have 20 bit Address line

- As we know that whenever microprocessor reset then IP and CS are initialized to from address FFFF0H. Hence this address must be lie in the EPROM. The address of the RAM may be selected any where in the 1MB address space of 8086.



Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
							EPROM						8K × 8							
FE000H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
FDFFFH	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
							RAM					8K × 8								
FC000H	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

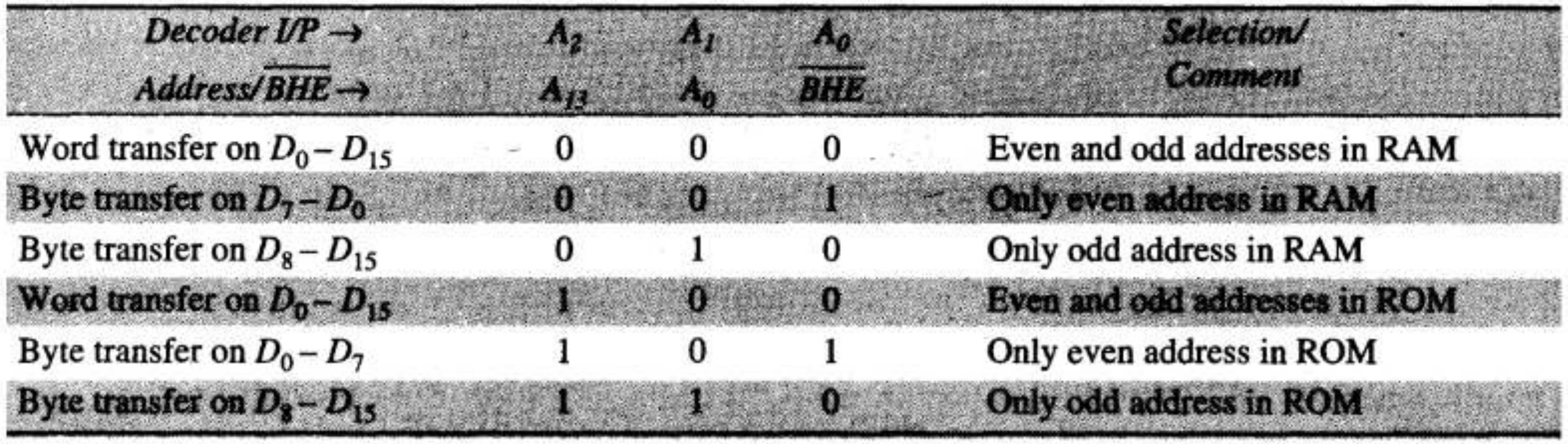
Total 8K bytes of EPROM need 13 address lines $A_0 - A_{12}$ (since $2^{13} = 8K$). Address lines $A_{13} - A_{19}$ are used for decoding to generate the chip select. The $\overline{BHE}$ signal goes low when a transfer is at odd address or higher byte of data is to be accessed. Let us assume that the latched address, $\overline{BHE}$ and demultiplexed data lines are readily available for interfacing. Figure 5.1 shows the interfacing diagram for the memory system.

The memory system in this example contains in total four 4K × 8 memory chips.

The two 4K × 8 chips of RAM and ROM are arranged in parallel to obtain 16-bit data bus width. If A_0 is 0, i.e. the address is even and is in RAM, then the lower RAM chip is selected indicating 8-bit transfer at an even address. If A_0 is 1, i.e. the address is odd and is in RAM, the $\overline{BHE}$ goes low, the upper RAM chip is selected, further indicating that the 8-bit transfer is at an odd address. If the selected addresses are in ROM, the respective ROM chips are selected. If at a time A_0 and $\overline{BHE}$ both are 0, both the RAM or ROM chips are selected, i.e. the data transfer is of 16 bits. The selection of chips here takes place as shown in Table 5.2.

Cont..

Table 5.2 *Memory Chip Selection for Problem 5.1*



Decoder I/P →	A_2	A_1	A_0	*Selection/*
Address/$\overline{BHE}$ →	A_{13}	A_0	$\overline{BHE}$	*Comment*
Word transfer on $D_0 - D_{15}$	0	0	0	Even and odd addresses in RAM
Byte transfer on $D_7 - D_0$	0	0	1	Only even address in RAM
Byte transfer on $D_8 - D_{15}$	0	1	0	Only odd address in RAM
Word transfer on $D_0 - D_{15}$	1	0	0	Even and odd addresses in ROM
Byte transfer on $D_0 - D_7$	1	0	1	Only even address in ROM
Byte transfer on $D_8 - D_{15}$	1	1	0	Only odd address in ROM

Application of 74LS138, a 3 to 8 lines decoder

1. G1, G2A and G2B are enable signals
2. To enable 74LS138, G1 should be high and G2A and G2B should be low
3. A, B and C are select lines
4. Y0, Y1, ...Y7 are output lines
5. An output lines goes low when it is selected, Other output lines remain high

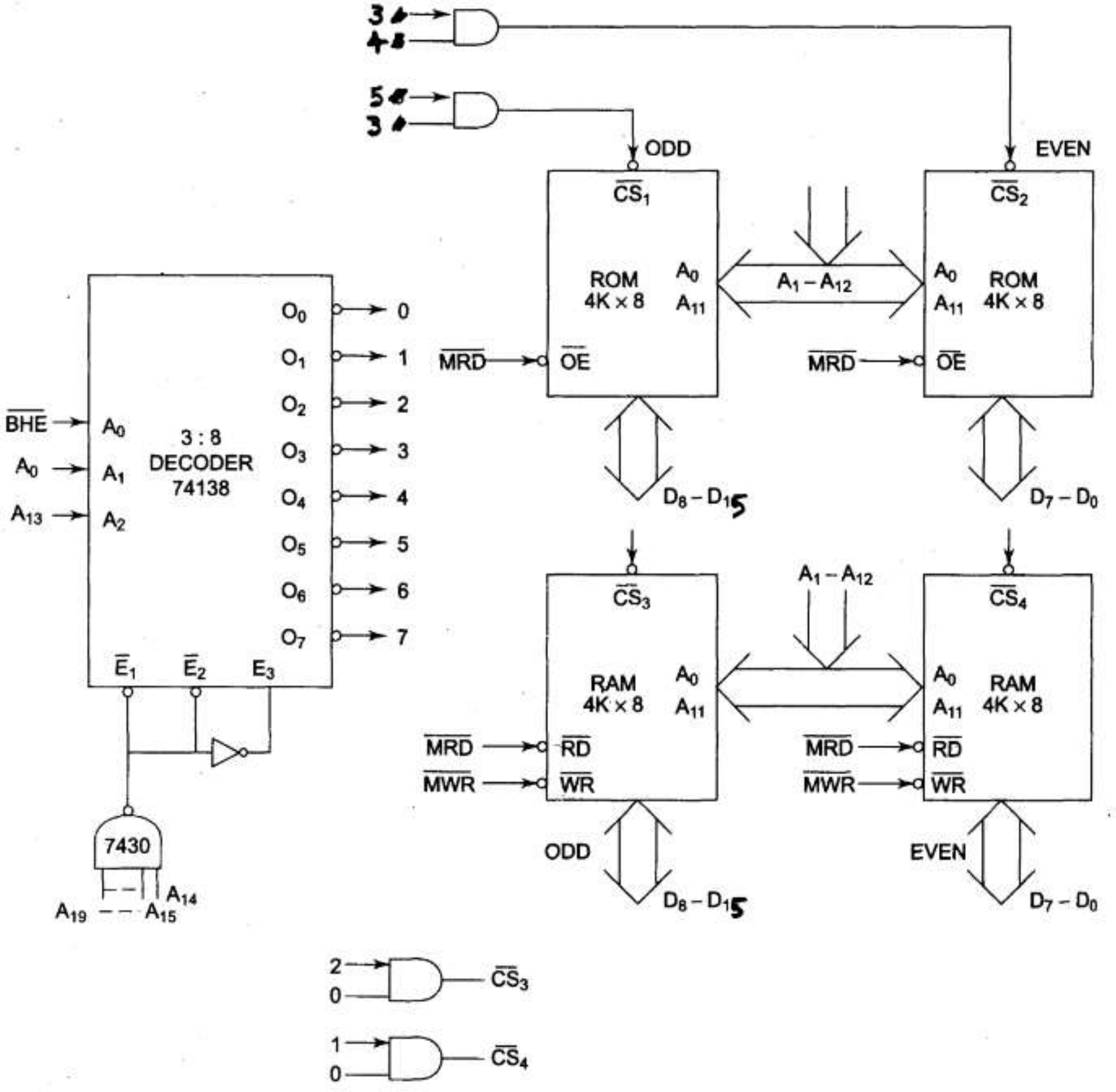
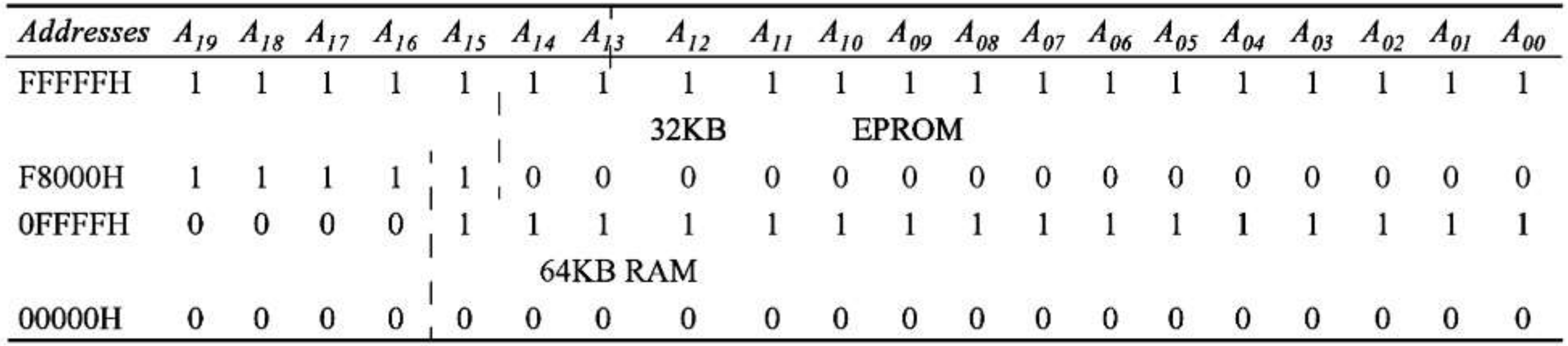


Fig. 5.1 *Interfacing Problem 5.1*

Example 2

- : It is required to interface two chips of 16K x 8 ROM and two chips of 32K x 8 RAM with 8086. Select the EPROM address suitably. The RAM address must start at 00000h
- Solution:
- The last address in the map of 8086 is FFFFFh. After resetting, the processor starts from FFFF0h..
- Hence this address must lie in the address range of EPROM.
- Address Mapping

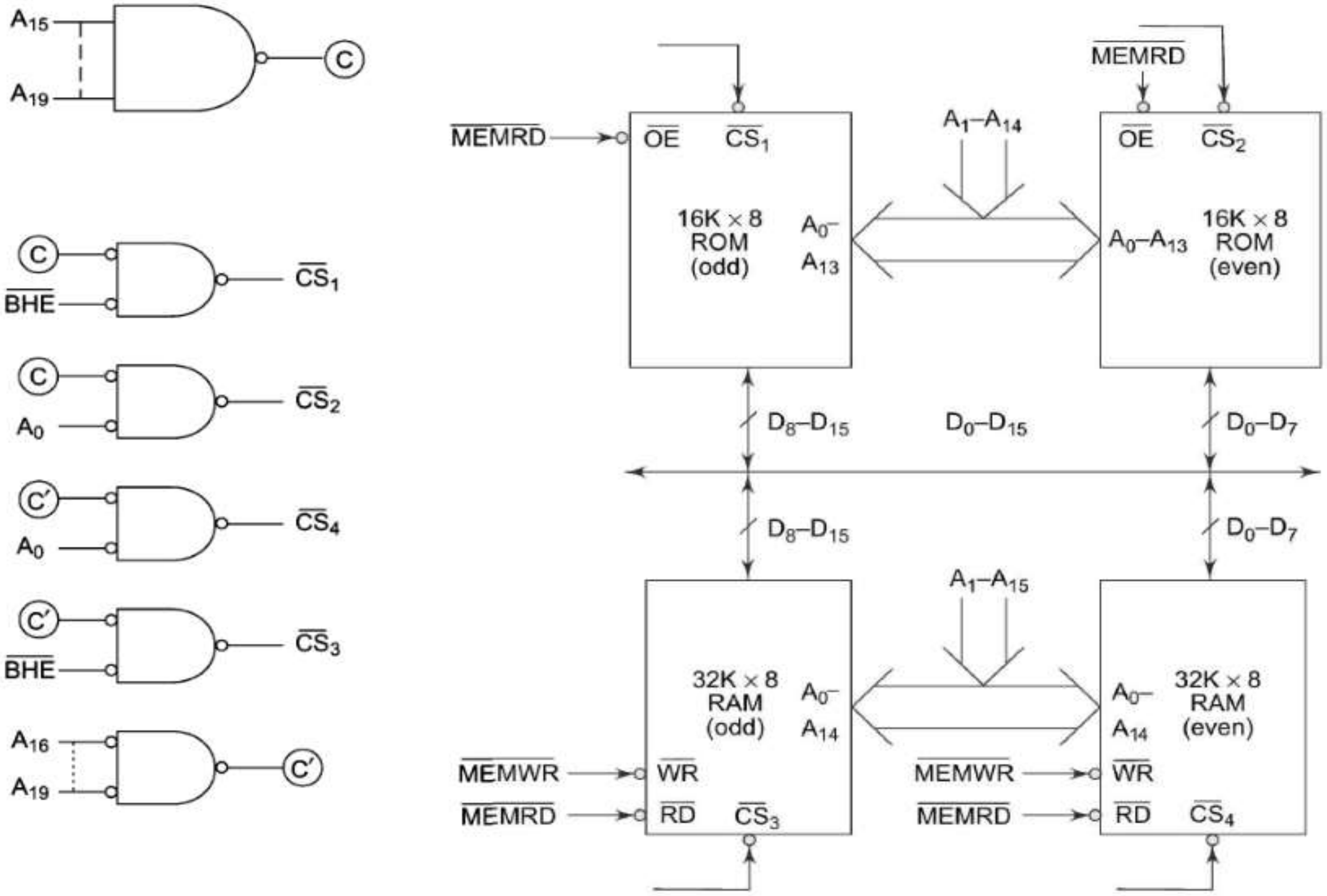


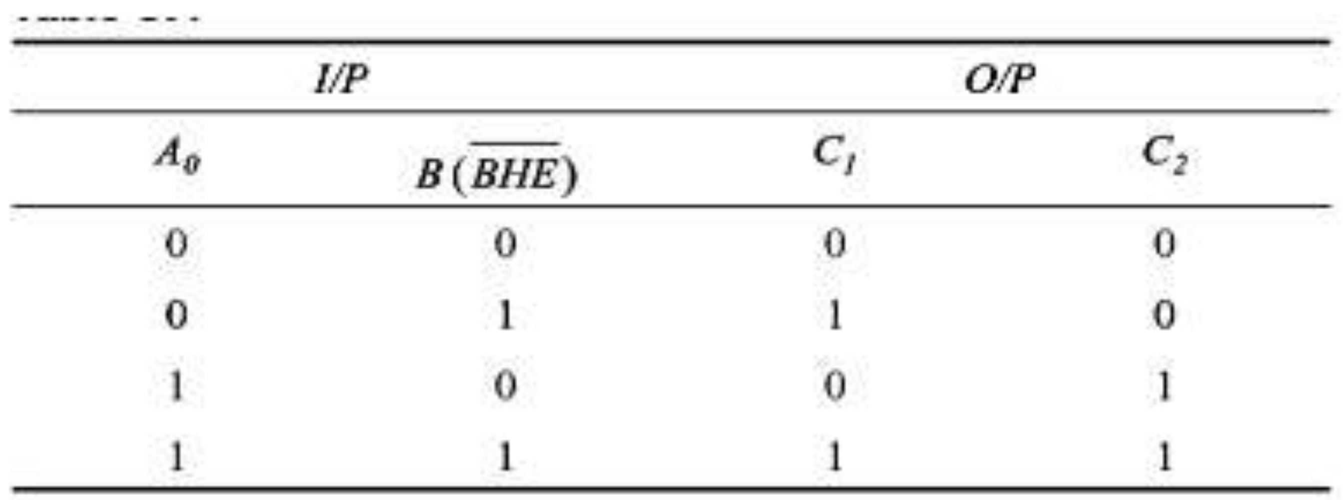
Addresses	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								32KB			EPROM									
F8000H	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0FFFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
							64KB RAM													
00000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- For 32 KB, number of address lines required -15 A0-A14
- For 64 KB, number of address lines required -16 A0-A15
- It is better not to use a decoder to implement the above map because it is not continuous . I.e.. There is some unused address space between the last RAM address (0FFFFh) and the first EPROM address (F8000h)

Cont..

Interfacing Diagram





I/P		O/P	
A_0	$B(\overline{BHE})$	C_1	C_2
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

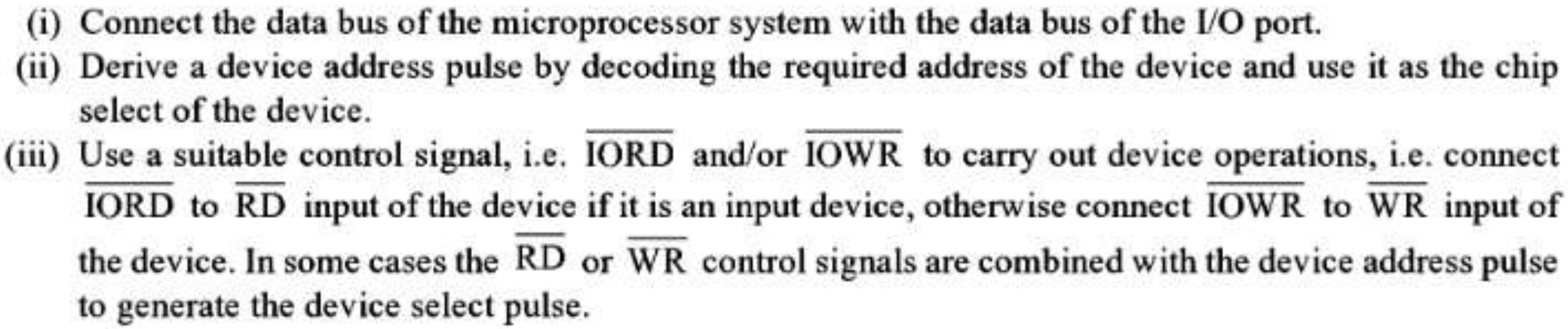
For 32 KB, number of address lines required -15 A0-A14, hence A15-A19 are used for chip selection

For 64 KB, number of address lines required -16 A0-A15
hence A16-A19 are used for chip selection

BHE*	A_0	Characteristics
0	0	Whole word
0	1	Upper byte from / to odd address
1	0	Lower byte from / to even address
1	1	None

I/O Interfacing:

- I/O ports are the devices through which the microprocessor communicates with other devices or external data sources /destinations.
- Input activity, is the activity that enables the microprocessor to read data from external devices, for example keyboard, joysticks, mouser etc. The devices are known as input devices as they feed data into a microprocessor system.
- Output activity transfers data from the microprocessor to the external devices, for example CRT display, 7 segment displays, printer, etc, the devices that accept the data from a microprocessor system are called output devices.
- Steps in Interfacing an I/O Device
- The following steps are performed to interface a general I/O device with a CPU:



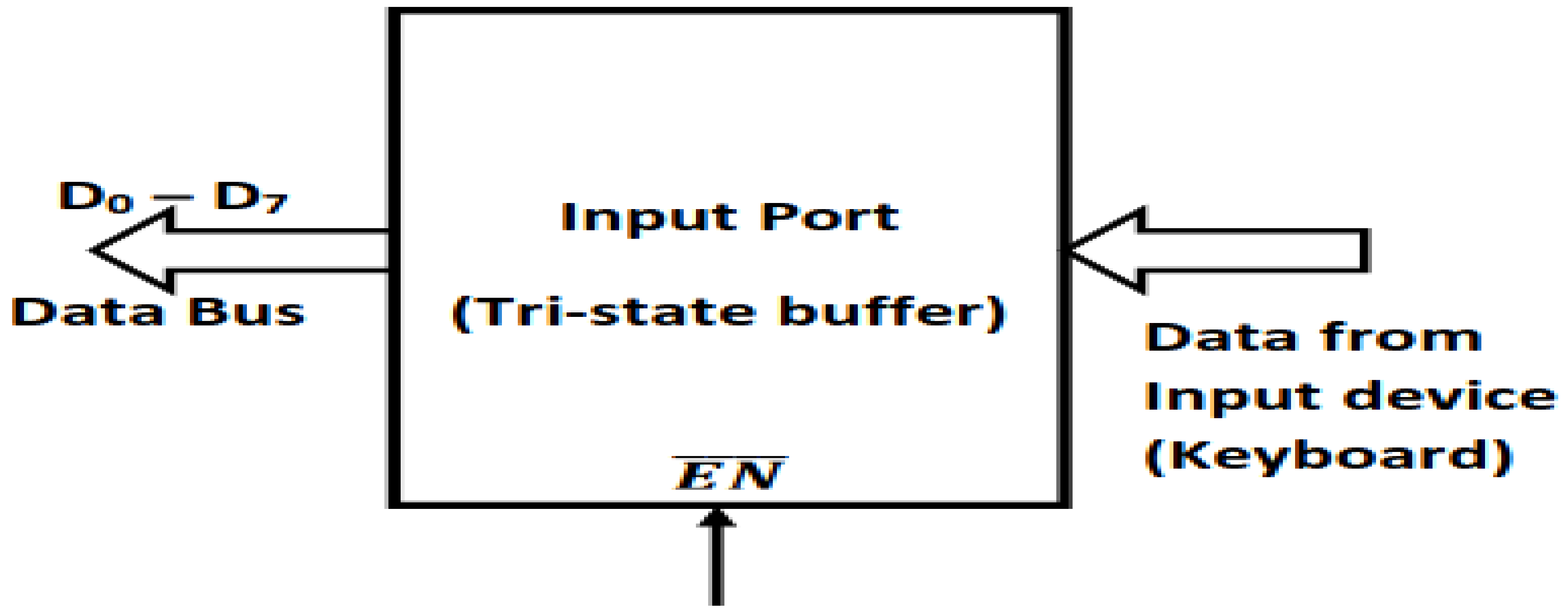
(i) Connect the data bus of the microprocessor system with the data bus of the I/O port.

(ii) Derive a device address pulse by decoding the required address of the device and use it as the chip select of the device.

(iii) Use a suitable control signal, i.e. $\overline{\text{IORD}}$ and/or $\overline{\text{IOWR}}$ to carry out device operations, i.e. connect $\overline{\text{IORD}}$ to $\overline{\text{RD}}$ input of the device if it is an input device, otherwise connect $\overline{\text{IOWR}}$ to $\overline{\text{WR}}$ input of the device. In some cases the $\overline{\text{RD}}$ or $\overline{\text{WR}}$ control signals are combined with the device address pulse to generate the device select pulse.

Input Port

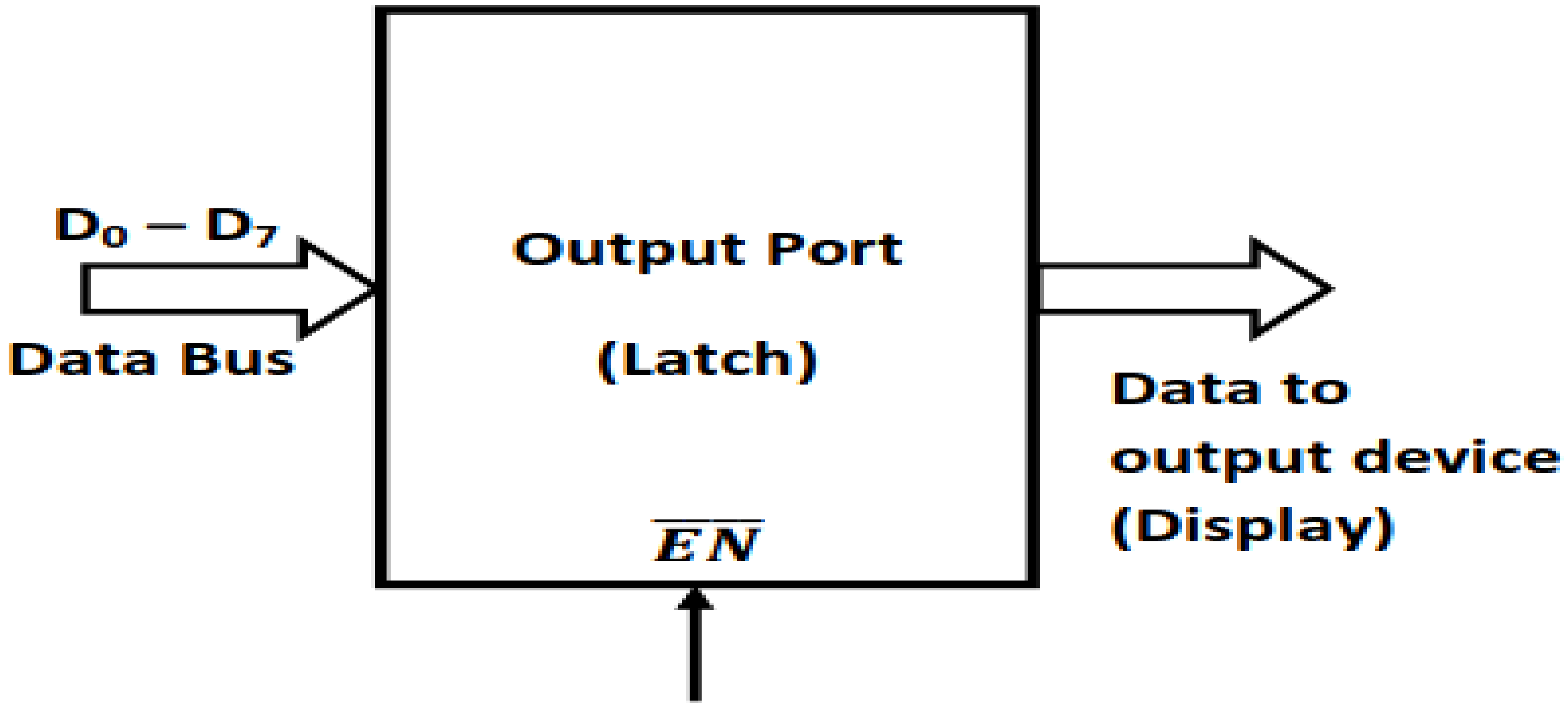
- It is used to read data from the input device such as keyboard. The simplest form of input port is a buffer. The chip 74LS244 contains eight buffers and is used as an 8-bit input port. The input device is connected to the microprocessor through the buffer shown in the figure below.



- When microprocessor wants to read data from the input device (keyboard), the control signals from the microprocessor activates the buffer by Enable input of the buffer through the enable pin. . Once the buffer is enabled, data from the device is available on the data bus. Microprocessor reads $(\overline{CS})$ this data by initiating read command

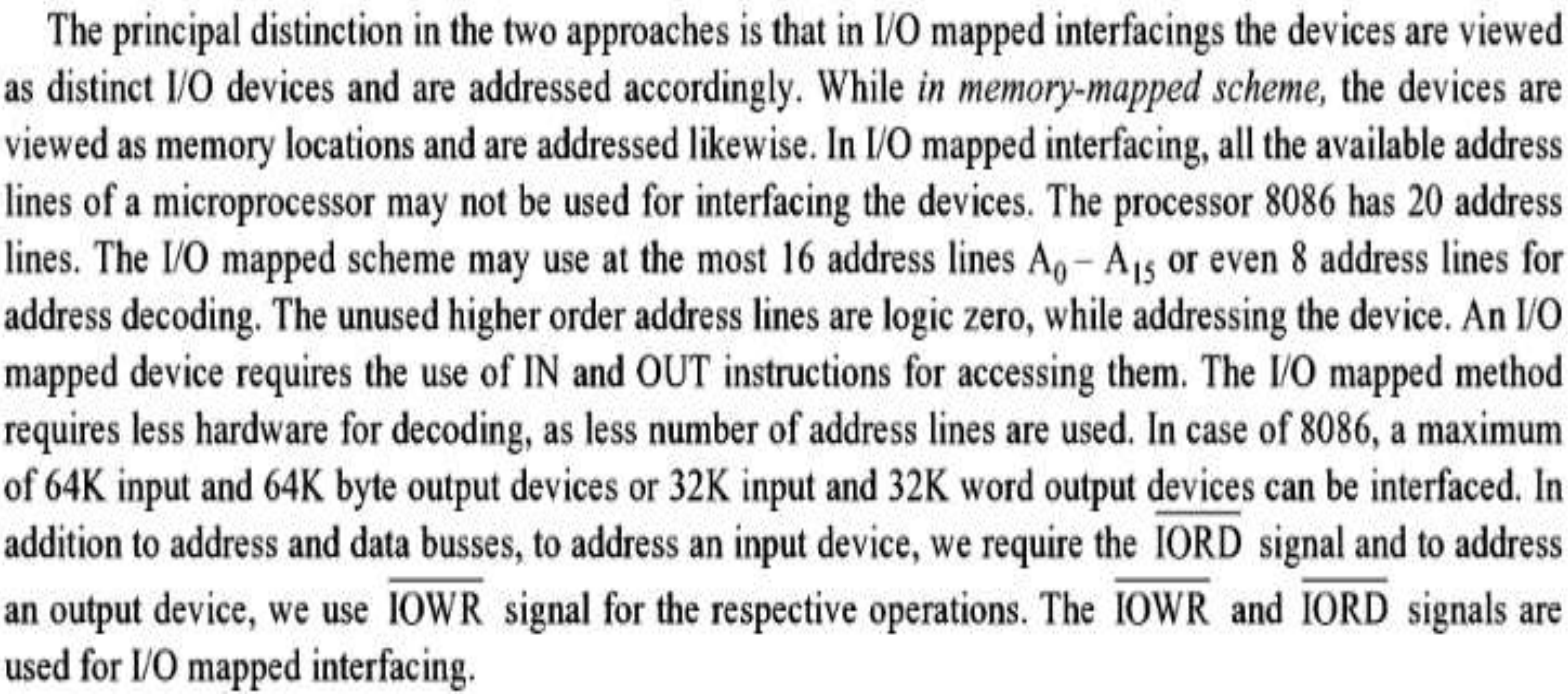
Output Port

- It is used to send data from the microprocessor to the output device such as display. The simplest form of the output port is a latch. The chip 74LS373 as shown below, contains eight latches (D flip-flops) and is used as an 8-bit output port. The output device is connected to the 8086µP through the 74LS373 latch.



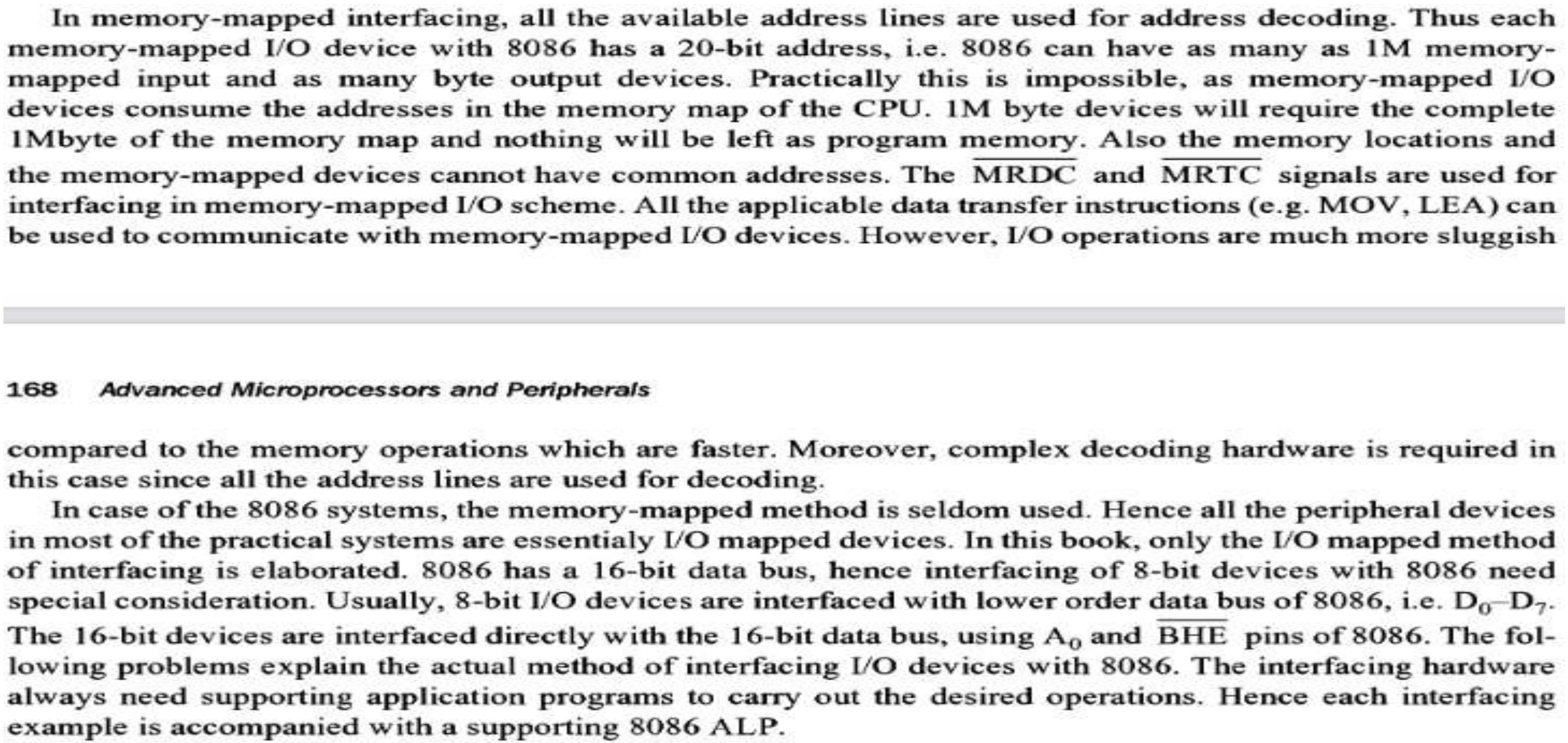
- The output device is connected to the microprocessor through latch as shown in the figure. When microprocessor wants to send data to the output device it puts the data on the data bus and activates the clock signal of the latch (EN)

I/O mapped I/O



The principal distinction in the two approaches is that in I/O mapped interfacings the devices are viewed as distinct I/O devices and are addressed accordingly. While *in memory-mapped scheme,* the devices are viewed as memory locations and are addressed likewise. In I/O mapped interfacing, all the available address lines of a microprocessor may not be used for interfacing the devices. The processor 8086 has 20 address lines. The I/O mapped scheme may use at the most 16 address lines $A_0 - A_{15}$ or even 8 address lines for address decoding. The unused higher order address lines are logic zero, while addressing the device. An I/O mapped device requires the use of IN and OUT instructions for accessing them. The I/O mapped method requires less hardware for decoding, as less number of address lines are used. In case of 8086, a maximum of 64K input and 64K byte output devices or 32K input and 32K word output devices can be interfaced. In addition to address and data busses, to address an input device, we require the $\overline{\text{IORD}}$ signal and to address an output device, we use $\overline{\text{IOWR}}$ signal for the respective operations. The $\overline{\text{IOWR}}$ and $\overline{\text{IORD}}$ signals are used for I/O mapped interfacing.

Memory mapped I/O

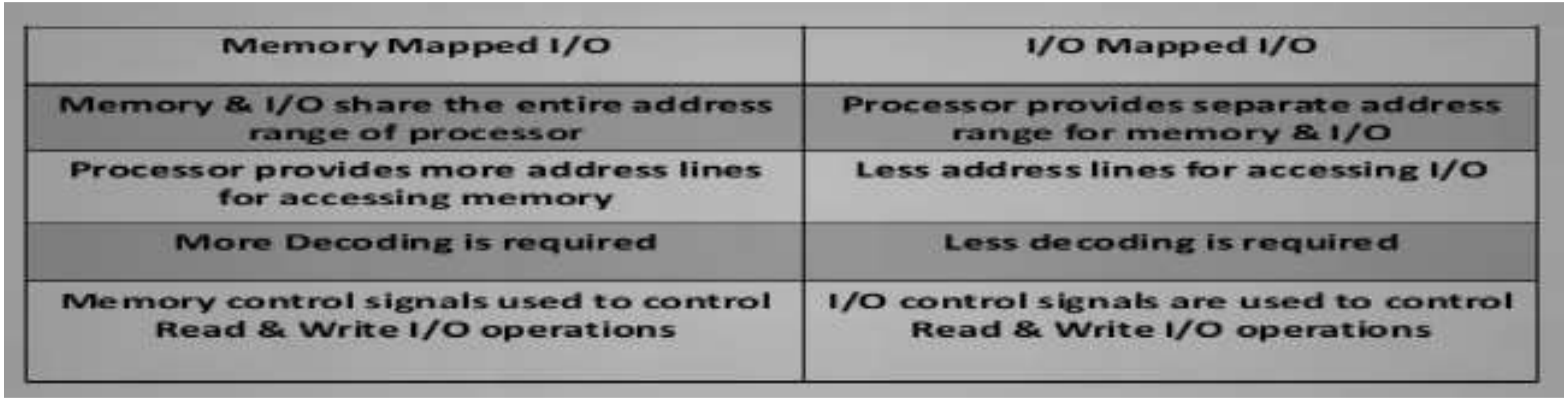


In memory-mapped interfacing, all the available address lines are used for address decoding. Thus each memory-mapped I/O device with 8086 has a 20-bit address, i.e. 8086 can have as many as 1M memory-mapped input and as many byte output devices. Practically this is impossible, as memory-mapped I/O devices consume the addresses in the memory map of the CPU. 1M byte devices will require the complete 1Mbyte of the memory map and nothing will be left as program memory. Also the memory locations and the memory-mapped devices cannot have common addresses. The $\overline{MRDC}$ and $\overline{MRTC}$ signals are used for interfacing in memory-mapped I/O scheme. All the applicable data transfer instructions (e.g. MOV, LEA) can be used to communicate with memory-mapped I/O devices. However, I/O operations are much more sluggish

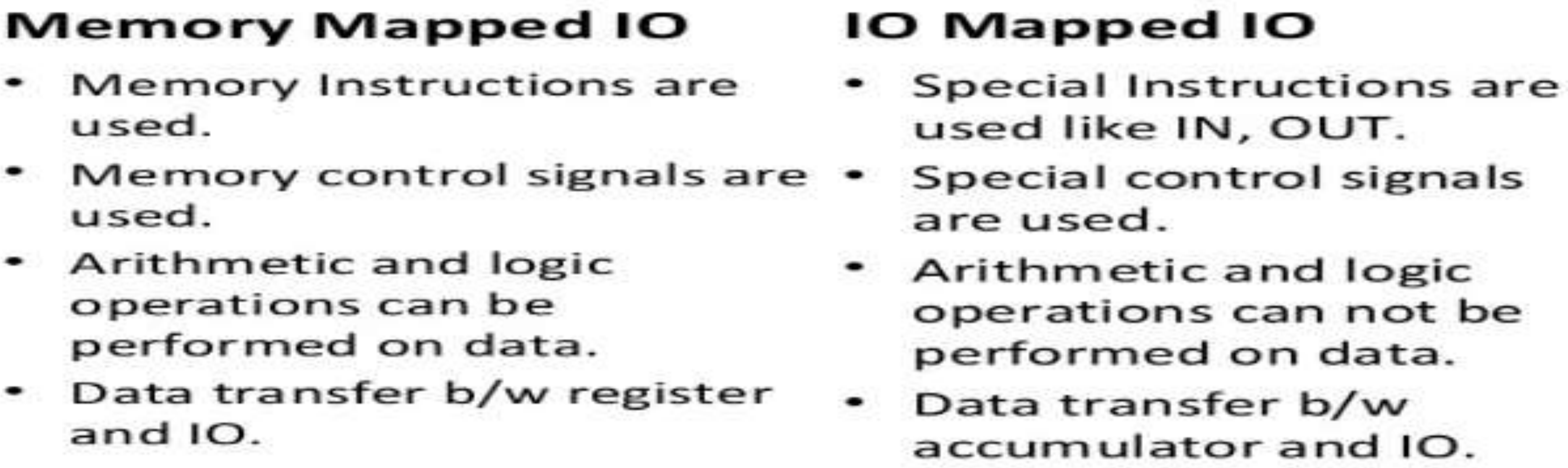
compared to the memory operations which are faster. Moreover, complex decoding hardware is required in this case since all the address lines are used for decoding.

In case of the 8086 systems, the memory-mapped method is seldom used. Hence all the peripheral devices in most of the practical systems are essentialy I/O mapped devices. In this book, only the I/O mapped method of interfacing is elaborated. 8086 has a 16-bit data bus, hence interfacing of 8-bit devices with 8086 need special consideration. Usually, 8-bit I/O devices are interfaced with lower order data bus of 8086, i.e. D_0–D_7. The 16-bit devices are interfaced directly with the 16-bit data bus, using A_0 and $\overline{BHE}$ pins of 8086. The following problems explain the actual method of interfacing I/O devices with 8086. The interfacing hardware always need supporting application programs to carry out the desired operations. Hence each interfacing example is accompanied with a supporting 8086 ALP.

Difference between Memory mapped I/O and I/O mapped I/O



Memory Mapped I/O	I/O Mapped I/O
Memory & I/O share the entire address range of processor	Processor provides separate address range for memory & I/O
Processor provides more address lines for accessing memory	Less address lines for accessing I/O
More Decoding is required	Less decoding is required
Memory control signals used to control Read & Write I/O operations	I/O control signals are used to control Read & Write I/O operations



Memory Mapped IO

- Memory Instructions are used.
- Memory control signals are used.
- Arithmetic and logic operations can be performed on data.
- Data transfer b/w register and IO.

IO Mapped IO

- Special Instructions are used like IN, OUT.
- Special control signals are used.
- Arithmetic and logic operations can not be performed on data.
- Data transfer b/w accumulator and IO.

Example

- Interface an input port 74LS245 to read the status of the switches SW1 to SW8. the switches when shorted, input a ‘1’ else input a ‘0’ to the microprocessor system. Store the status in register BL. The address of the port is 0740H
- Sol: The hardware interface circuit is shown in figure.

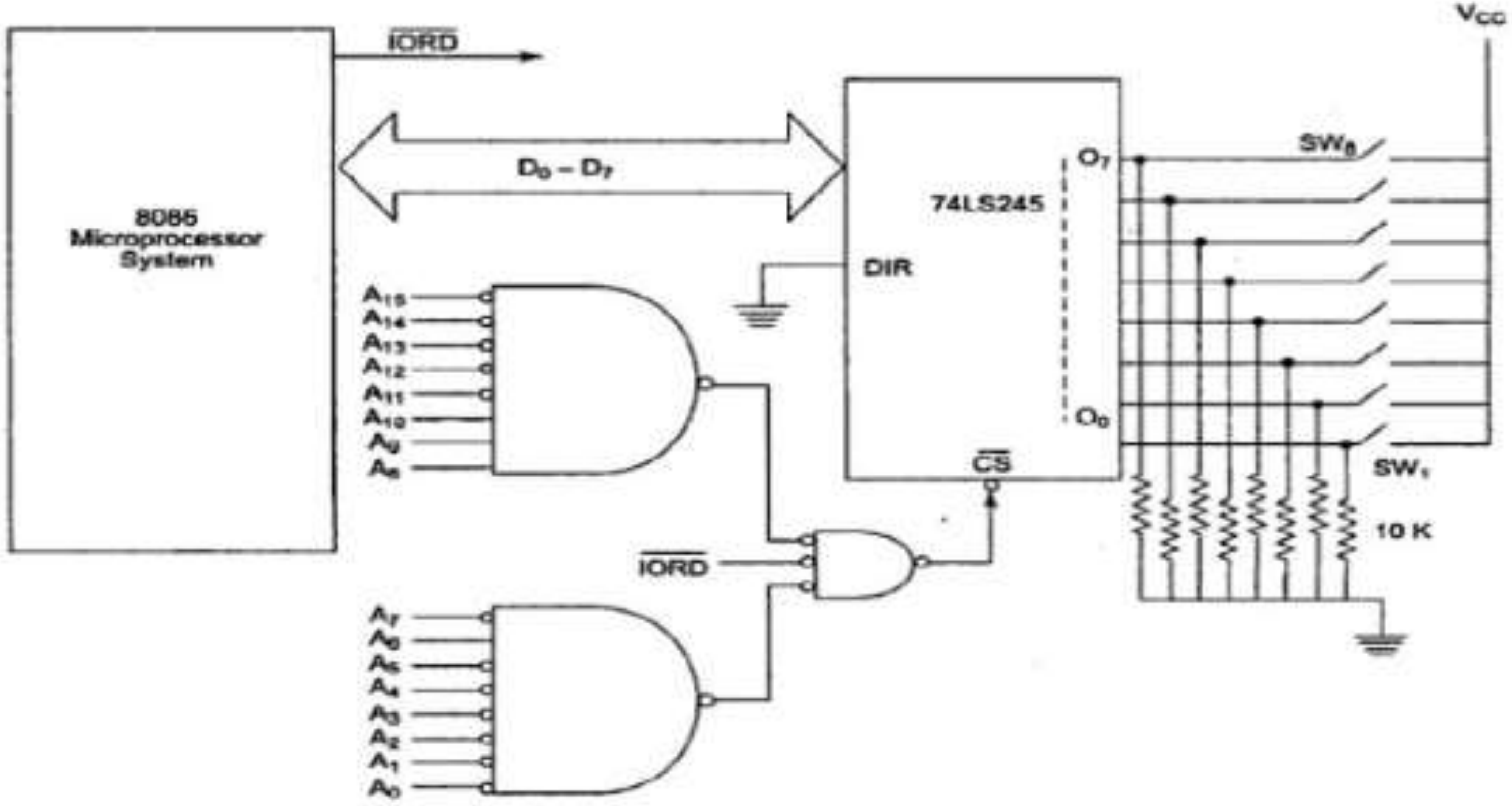


Fig. Interfacing Input Port 74LS245

The address, control and data lines are assumed to be readily available at the microprocessor system The ALP is given as follows:

```
MOV BL, 00H ; clear BL for status
MOV DX, 0740H ; 16-bit Port address in DX
IN AL,DX ; Read Port 0740H for switch positions.
MOV BL, AL ; Store status of switches from AL into BL
HLT ; Stop
```

Here LSB bit of BL corresponds to the status of SW1 and likewise the MSB of BL corresponds to the status of SW8.

Cont..

- **Problem: Design an interface of input port 74LS245 to read the status of switches SW1 to SW8 and output port 74LS373 with 8086. Display the number of key that is pressed with the help of output port on 7 segment display. Write an ALP for this task. The input port address is 08H and output port address is 0AH.**

- **Solution: Status of the switches is first read into the AL. Displaying the shorted switch number in the 7 segment display. Instead of using 16 address lines, one may use only A3– A0.**

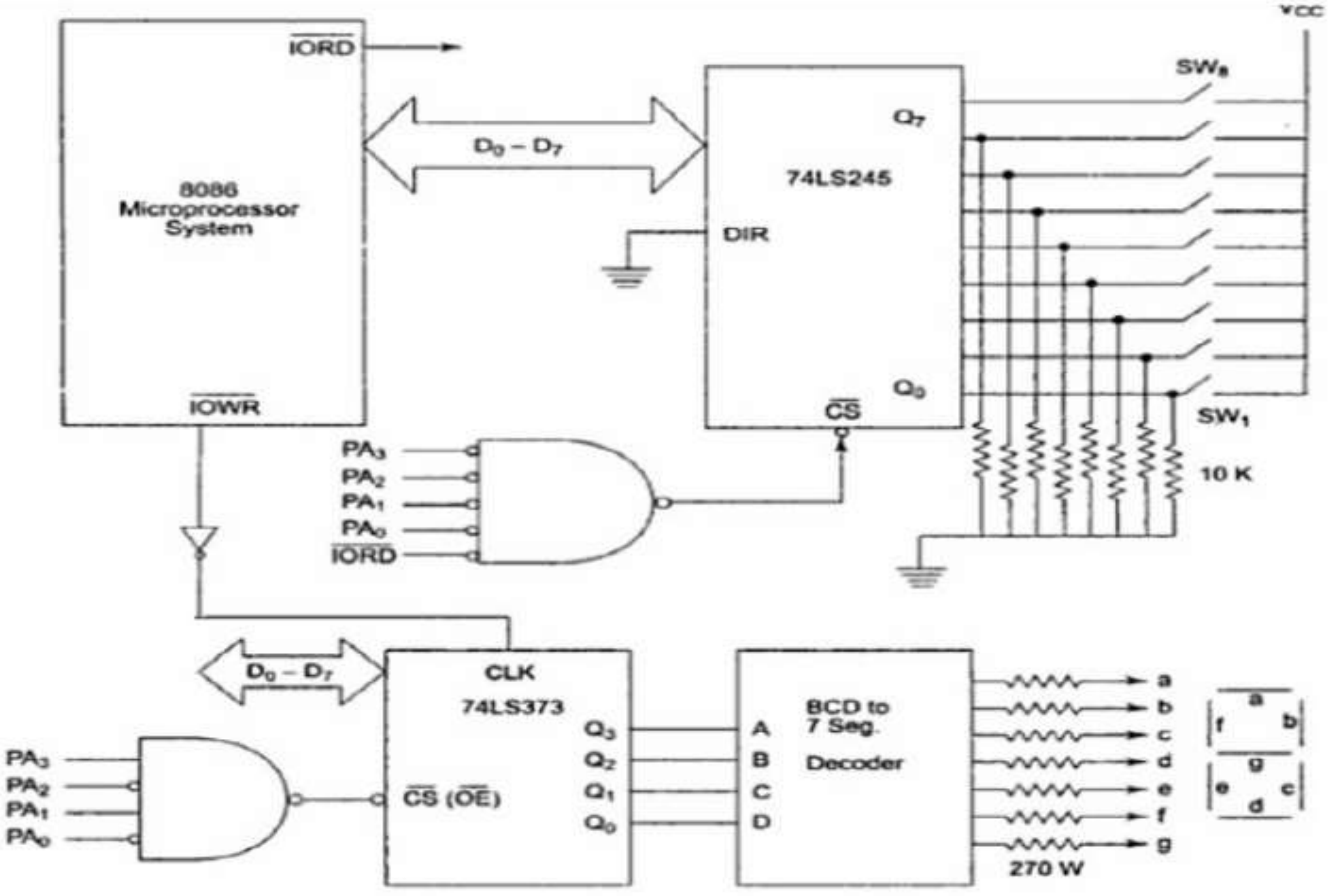
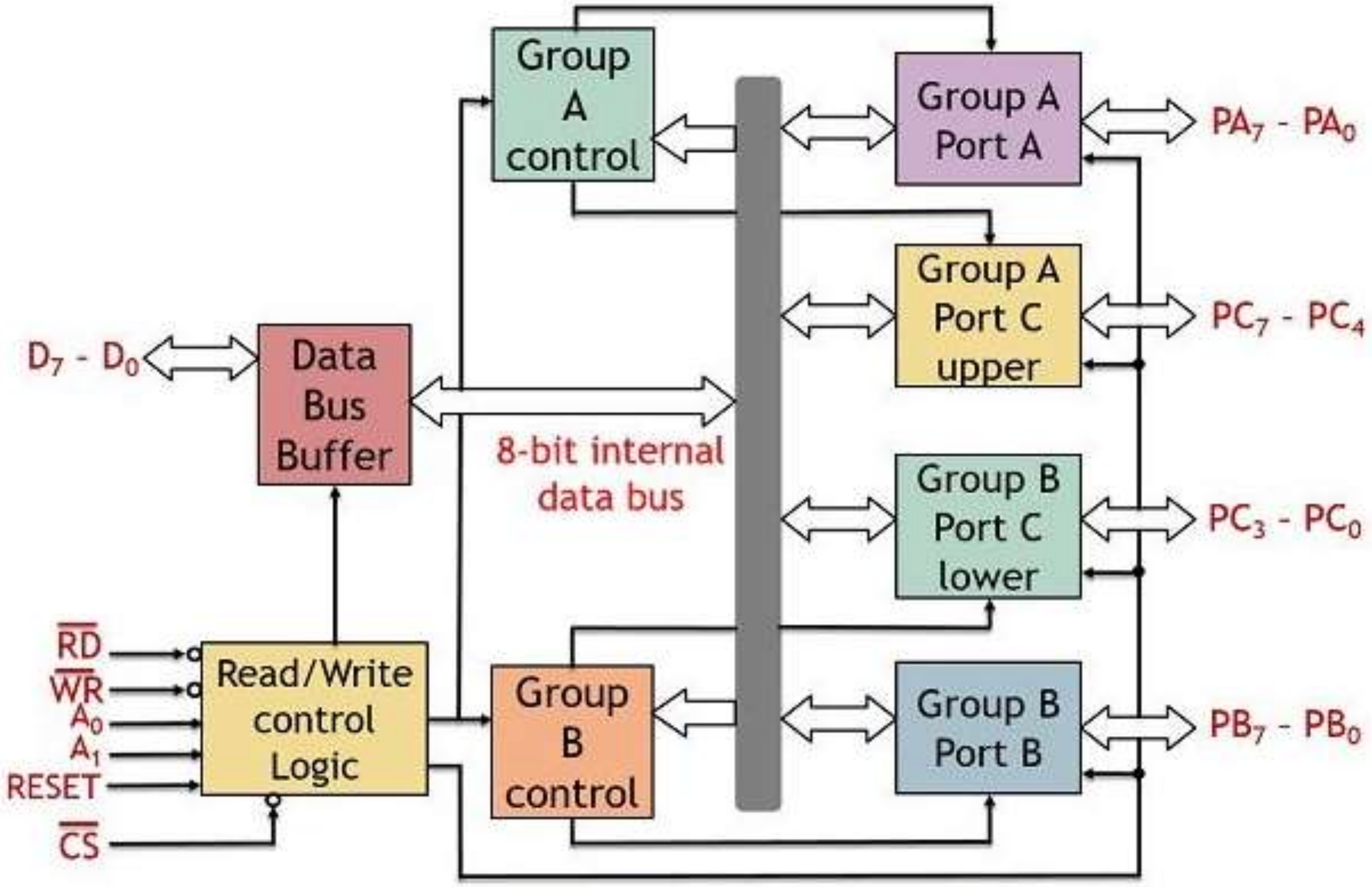


Fig. *Interfacing Switches and Displays for Problem*

8255 PPI (Programmable Peripheral interface)

- **It is a programmable peripheral interface, which means it is a programmable device used to interface I/O devices with the processor. In reality, we are not supposed to connect I/O devices directly with the data bus of the processor ,instead there should be some device to which I/O ports should be there to connect I/O devices**

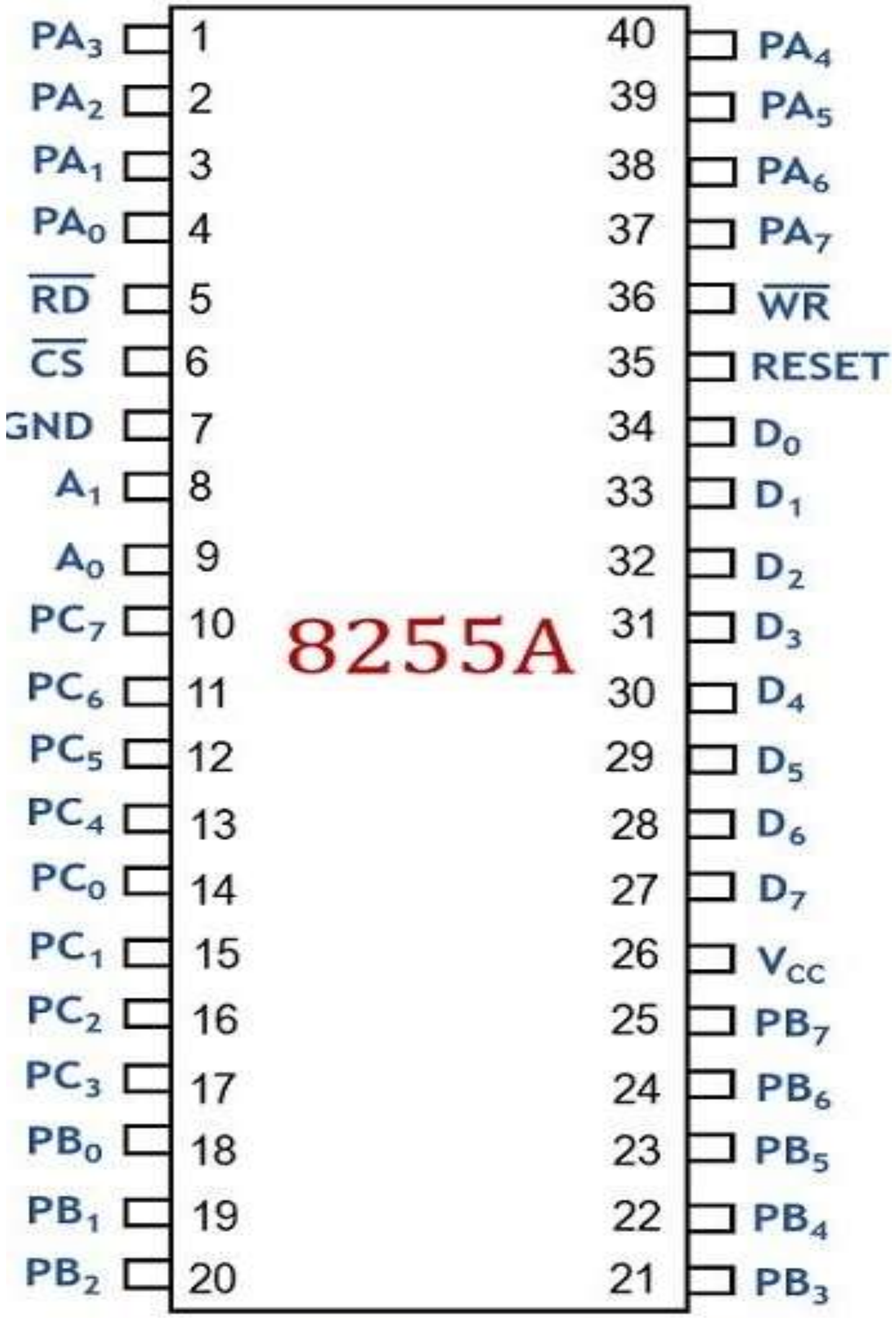
- Fig shows the internal 8255 block diagram of 8255 Pin Diagram Microprocessor. It consists of data bus buffer, control logic and Group A and Group B controls



Architecture of 8255 PPI

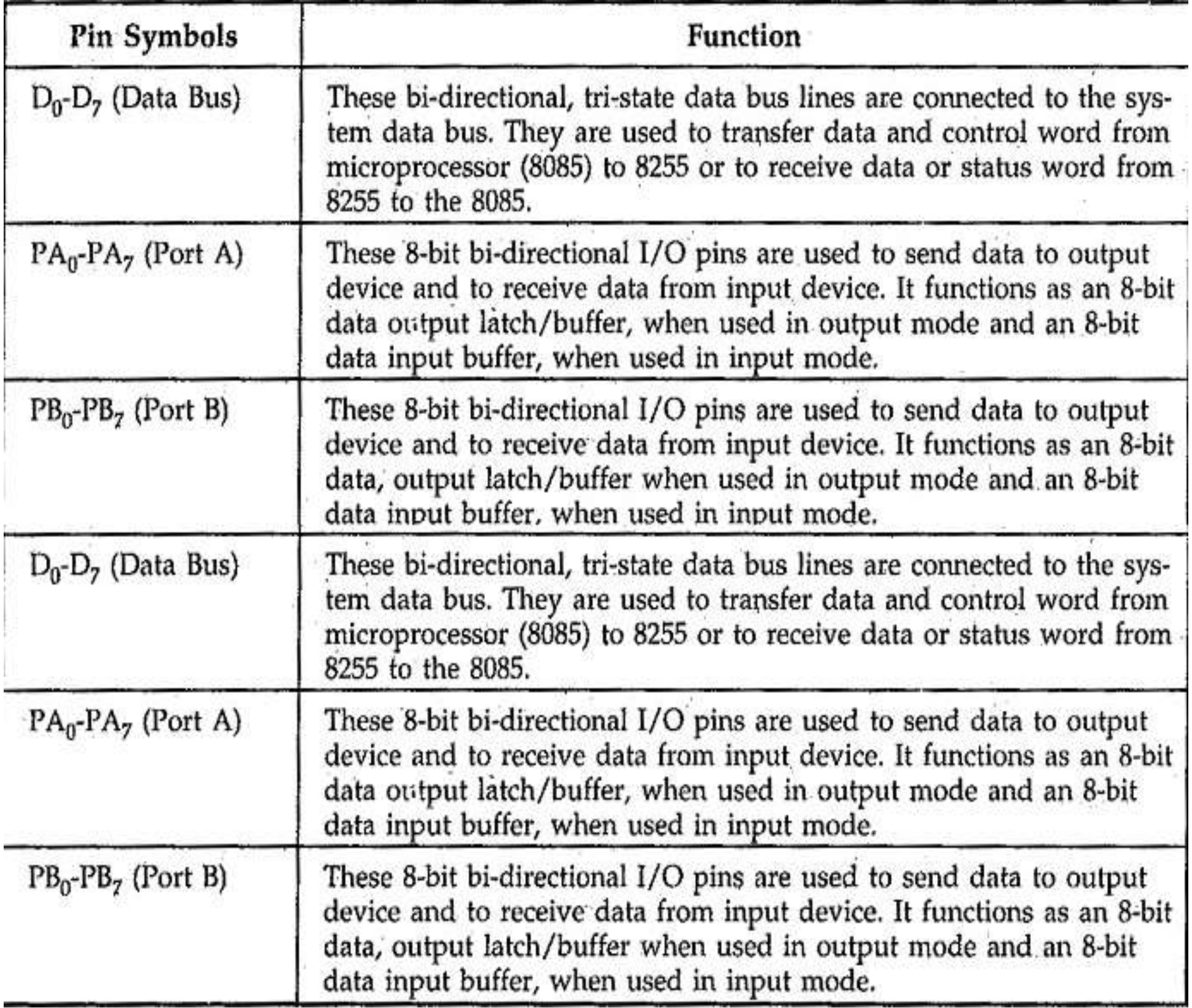
Pin Diagram of 8255 PPI

- This microprocessor includes 40-pins like
- PA7-PA0, PC7-PC0, PC3-PC0, PB0-PB7, RD, WR,
- CS, A1 & A0,D0-D7 and RESET.



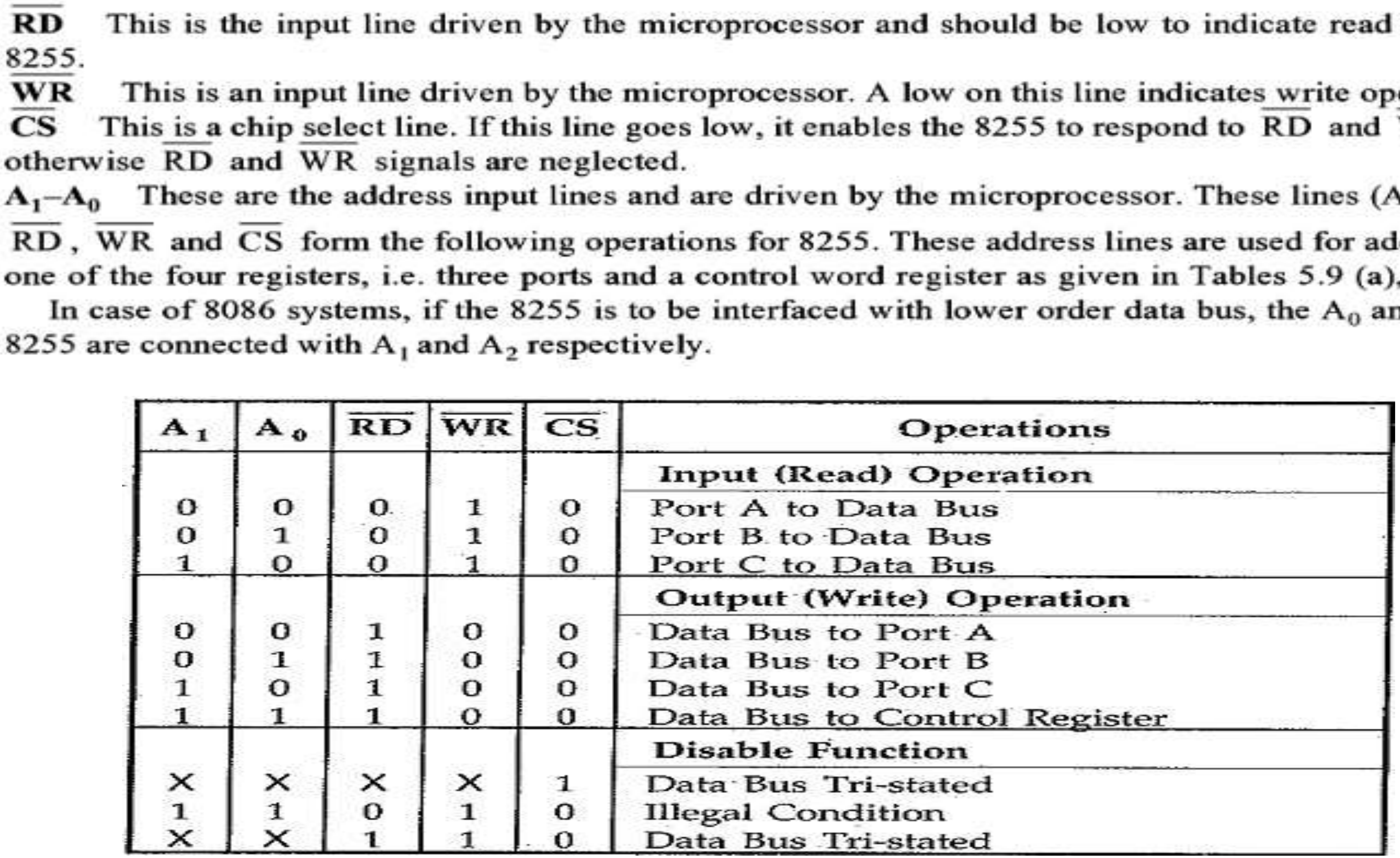
Pin Diagram of 8255 PPI

Cont..



Pin Symbols	Function
D_0-D_7 (Data Bus)	These bi-directional, tri-state data bus lines are connected to the system data bus. They are used to transfer data and control word from microprocessor (8085) to 8255 or to receive data or status word from 8255 to the 8085.
PA_0-PA_7 (Port A)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. It functions as an 8-bit data output latch/buffer, when used in output mode and an 8-bit data input buffer, when used in input mode.
PB_0-PB_7 (Port B)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. It functions as an 8-bit data, output latch/buffer when used in output mode and an 8-bit data input buffer, when used in input mode.
D_0-D_7 (Data Bus)	These bi-directional, tri-state data bus lines are connected to the system data bus. They are used to transfer data and control word from microprocessor (8085) to 8255 or to receive data or status word from 8255 to the 8085.
PA_0-PA_7 (Port A)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. It functions as an 8-bit data output latch/buffer, when used in output mode and an 8-bit data input buffer, when used in input mode.
PB_0-PB_7 (Port B)	These 8-bit bi-directional I/O pins are used to send data to output device and to receive data from input device. It functions as an 8-bit data, output latch/buffer when used in output mode and an 8-bit data input buffer, when used in input mode.

Cont..



- $\overline{\textbf{RD}}$ This is the input line driven by the microprocessor and should be low to indicate read operation to 8255.

  $\overline{\textbf{WR}}$ This is an input line driven by the microprocessor. A low on this line indicates write operation.

  $\overline{\textbf{CS}}$ This is a chip select line. If this line goes low, it enables the 8255 to respond to $\overline{RD}$ and $\overline{WR}$ signals, otherwise $\overline{RD}$ and $\overline{WR}$ signals are neglected.

  $\mathbf{A_1}$**–**$\mathbf{A_0}$ These are the address input lines and are driven by the microprocessor. These lines ($A_1 - A_0$) with $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ form the following operations for 8255. These address lines are used for addressing any one of the four registers, i.e. three ports and a control word register as given in Tables 5.9 (a), (b) and (c).

  In case of 8086 systems, if the 8255 is to be interfaced with lower order data bus, the A_0 and A_1 pins of 8255 are connected with A_1 and A_2 respectively.

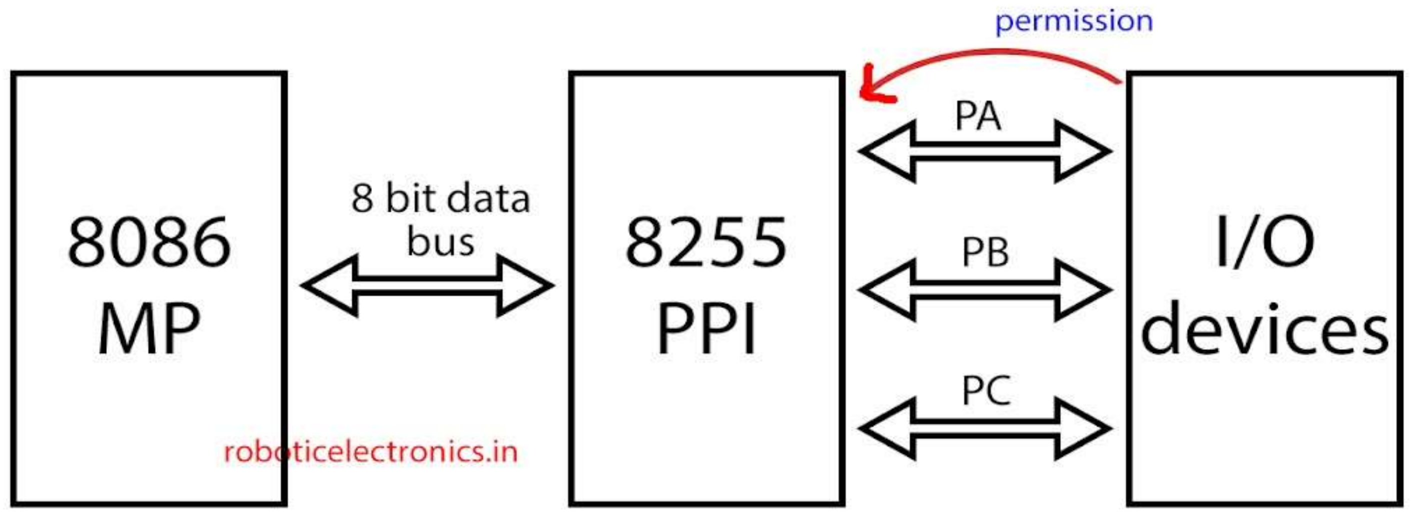
A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Operations
					Input (Read) Operation
0	0	0	1	0	Port A to Data Bus
0	1	0	1	0	Port B to Data Bus
1	0	0	1	0	Port C to Data Bus
					Output (Write) Operation
0	0	1	0	0	Data Bus to Port A
0	1	1	0	0	Data Bus to Port B
1	0	1	0	0	Data Bus to Port C
1	1	1	0	0	Data Bus to Control Register
					Disable Function
X	X	X	X	1	Data Bus Tri-stated
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus Tri-stated

Modes of Operation

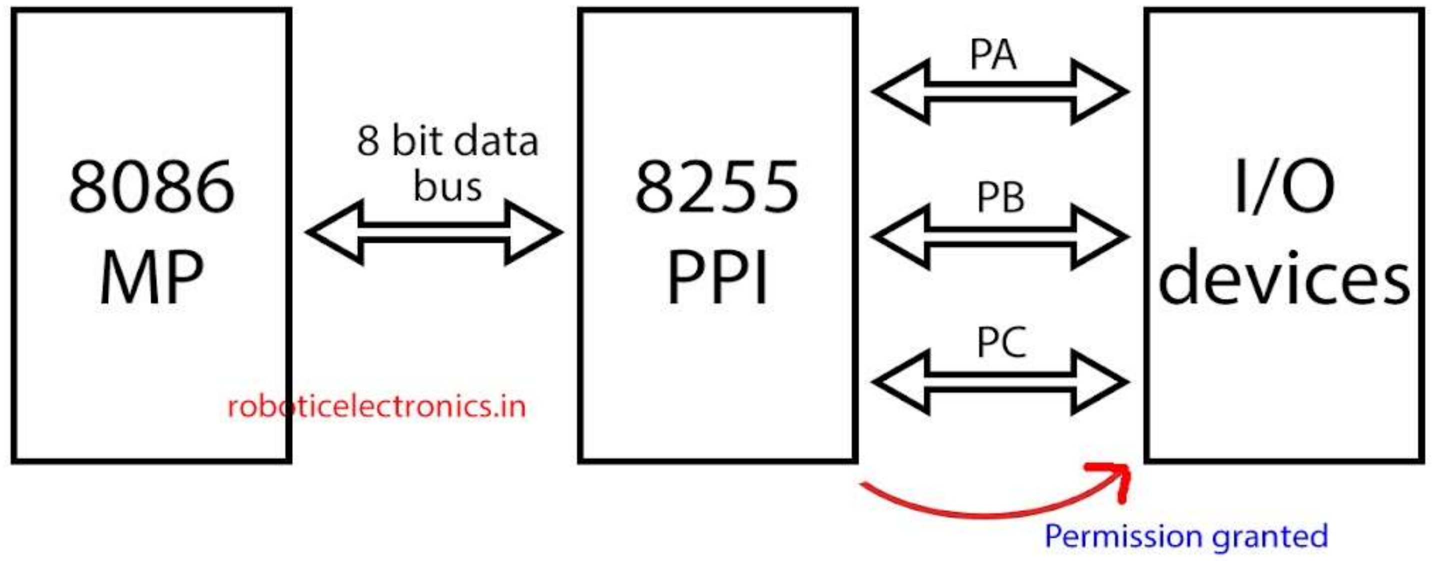
- **As we have already discussed that 8255 has two modes of operation.**
  - **Bit Set-Reset mode**
  - **I/O mode**

8255 Working

- **As already mentioned 8255 has 3 data ports namely Port – A (PA) and PB, PC.**
- Assuming the connected device to be an input device. Initially, the Input device seeks for permission from PPI so that it can send data.

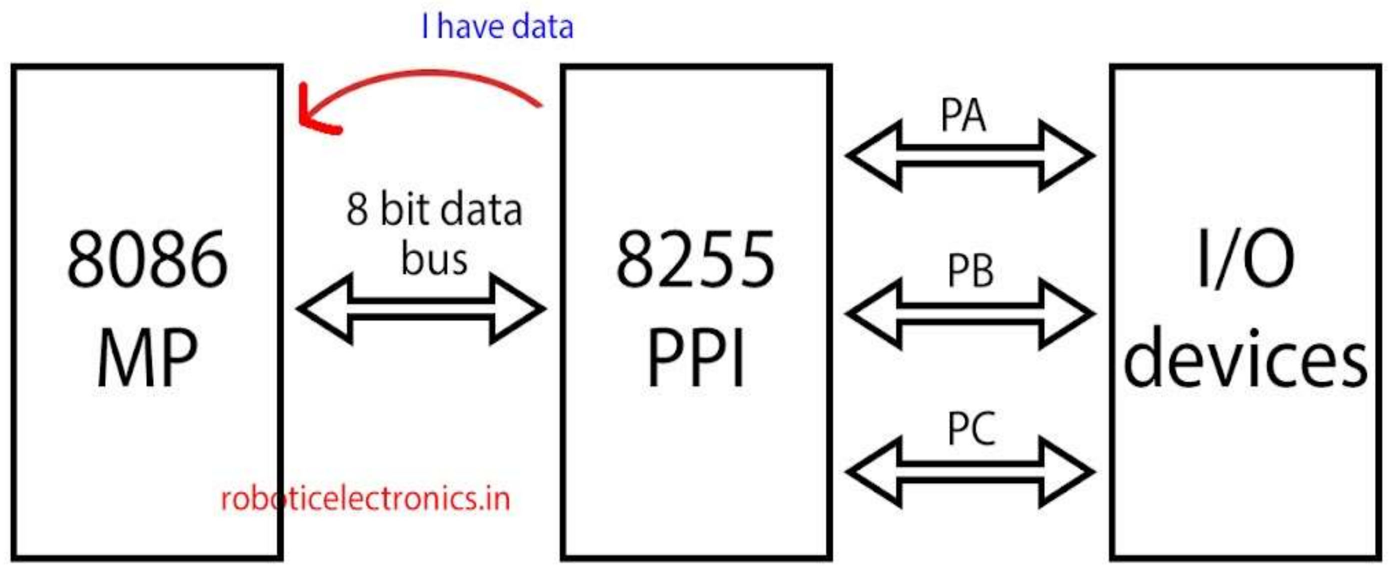


- PPI permits Input devices to send data, only when there is no left data in 8255 which should be sent to the 8086 processor. If there is some previous data left in 8255, which is still not sent to 8086, then it doesn’t permit Input device.



Cont..

- **Once 8255 permits input device, data is received and stored in temporary registers in 8255. Once 8255 holds some data, which should be sent to 8086, then it send signal to 8086**



- **Whenever 8086 is free to receive the data, then 8086 sends back a signal , after that data transmission happens between 8255 and 8086. If 8086 do not becomes free upto long time, which means 8255 has some value in it which is still not sent to 8086, so 8255 does not permit the Input device to send any data because the existing data will be overwritten.**
- All the signal in the above diagrams represented using red curved arrow are known as handshake signals. This process of data transmission is known as handshaking.

8255 Interfacing with 8086

- Fig Shows the 8255 Interfacing with 8086 Microprocessor and Interfacing 8255 with 8085 Microprocessor in I/O mapped I/O technique. Here RD and WR signals are activated when IO/M signal is high, indicating I/O bus cycle. Reset out signal from 8085 is connected to the RESET signal of the 8255

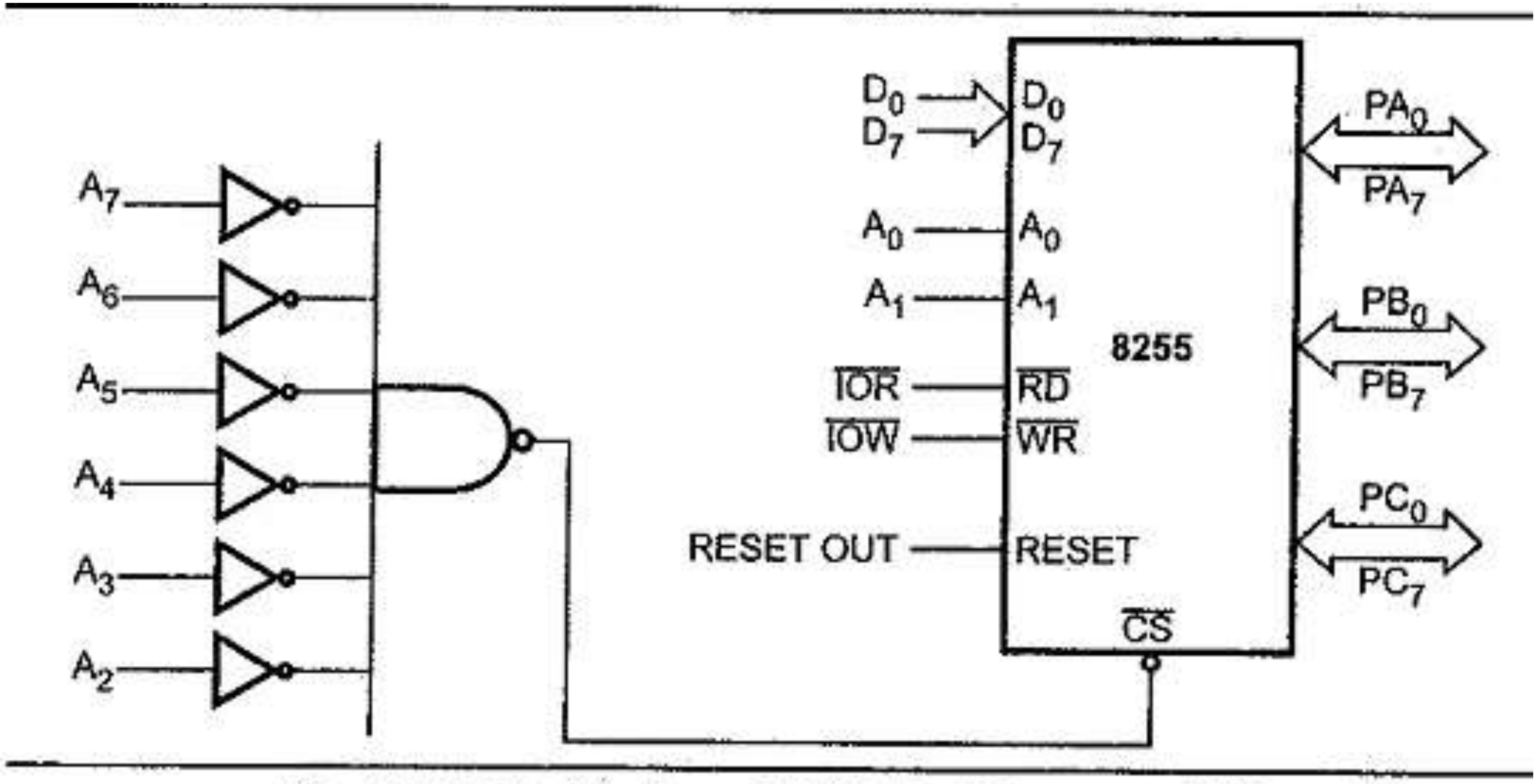
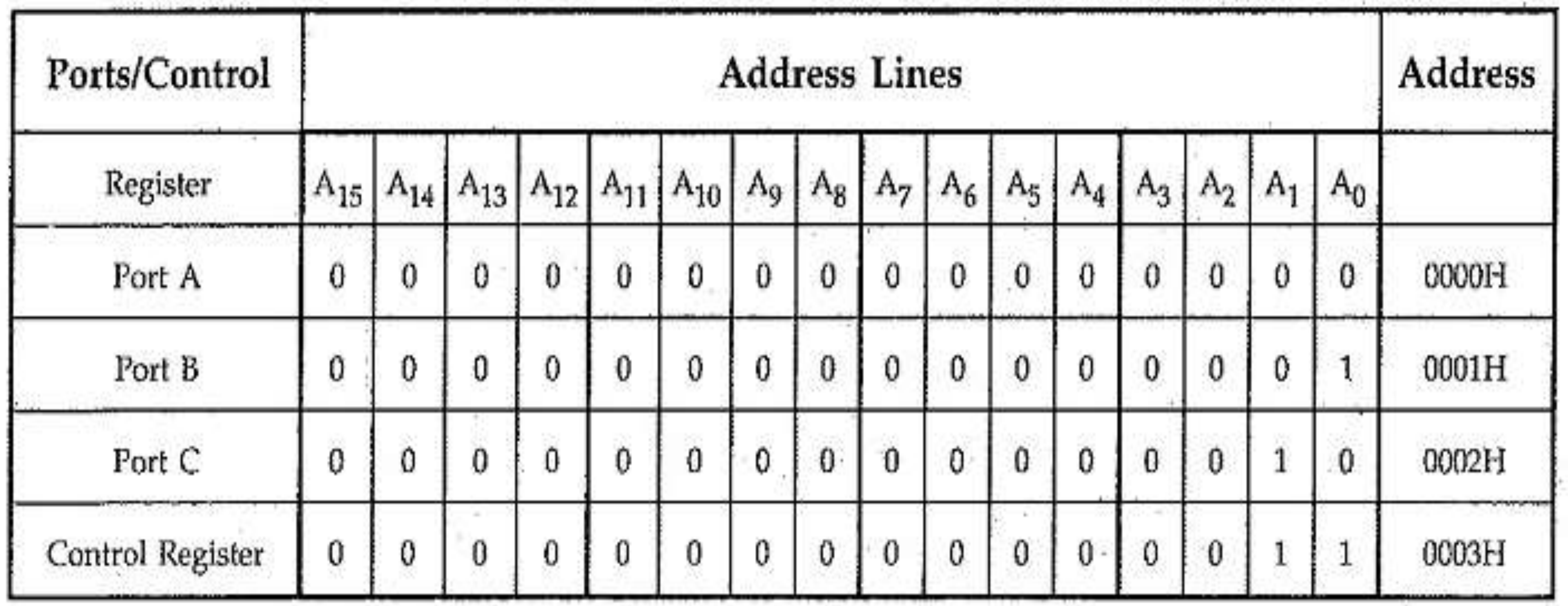


Fig. 14.17 Interfacing of 8255 in I/O mapped I/O

Interfacing 8255 In Memory Mapped I/O

- Fig shows the interfacing of 8255 with 8085 in memory mapped I/C technique. Here RD and WR signals are activated when 10/M signal is low, indicating memory bus cycle. To get absolute address, all remaining address lines (A15 – A2) are used to decode the address for 8255. Other signal connections are same as in I/C mapped I/O.



Ports/Control	Address Lines																Address
Register	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Port A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
Port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001H
Port C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002H
Control Register	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0003H

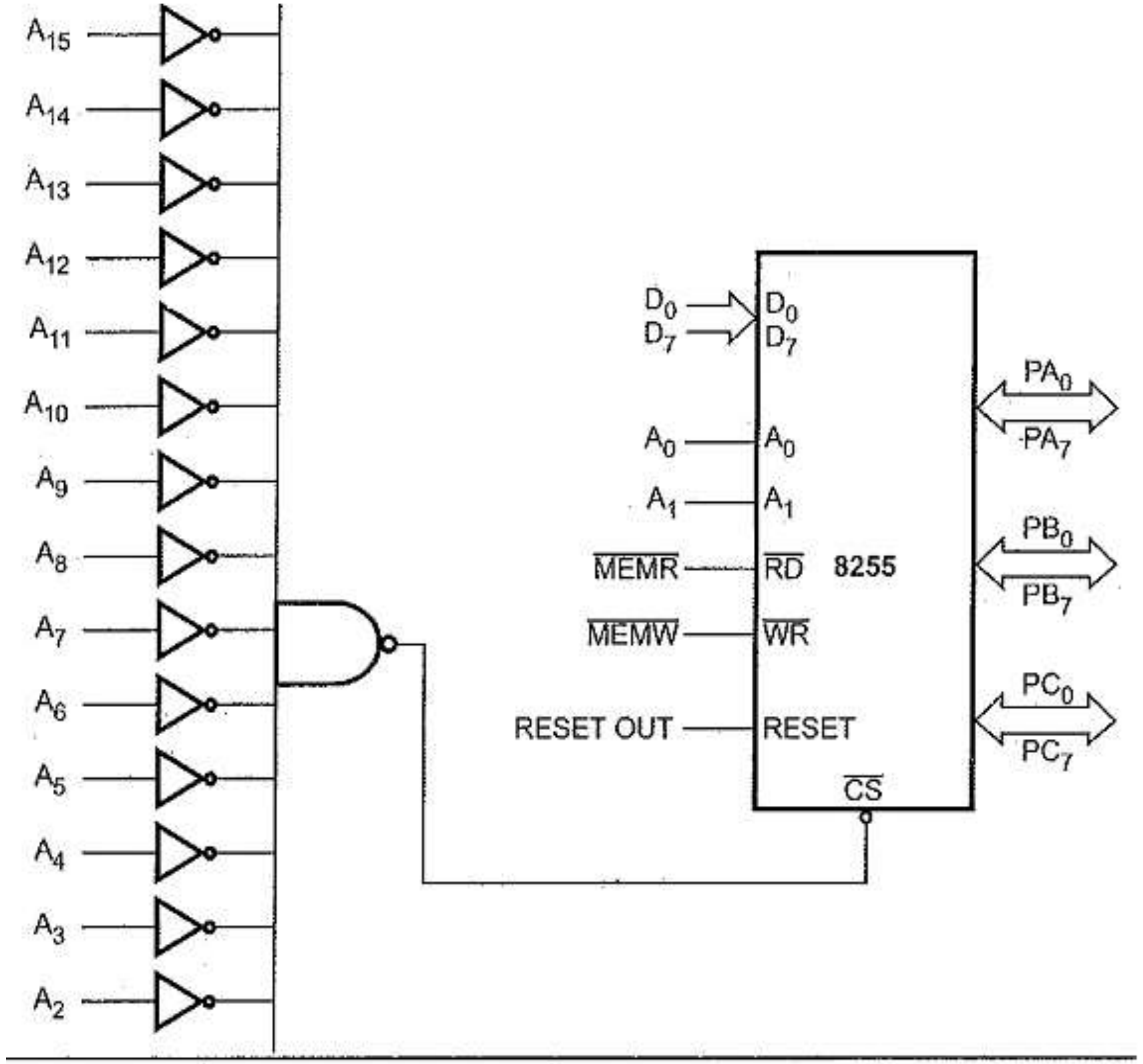
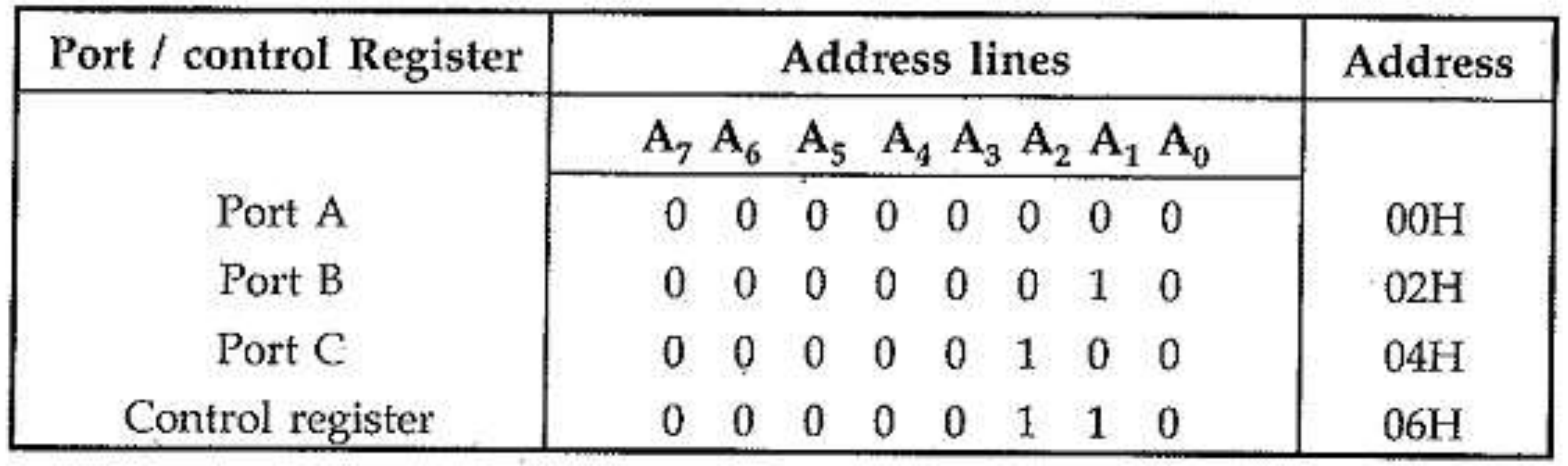


Fig. 14.18 Interfacing 8255 in memory mapped I/O

8255 Interfacing with 8086 in I/O Mapped I/O Mode

- **Fig shows the 8255 Interfacing with 8086 in I/O mapped I/O technique. Here, RD and WR signals are activated when M/IO signal is low, indicating I/O bus cycle. Only lower data bus (D0 — D7) is used as 8255 is 8-bit device. Reset out signal from clock generator is connected to the Reset signal of the 8255. In case of interrupt driven I/O INTR signal (PC3 or PC0) from 8255 is connected to INTR input of 8088.**

I/O Map :



Port / control Register	Address lines								Address
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Port A	0	0	0	0	0	0	0	0	00H
Port B	0	0	0	0	0	0	1	0	02H
Port C	0	0	0	0	0	1	0	0	04H
Control register	0	0	0	0	0	1	1	0	06H

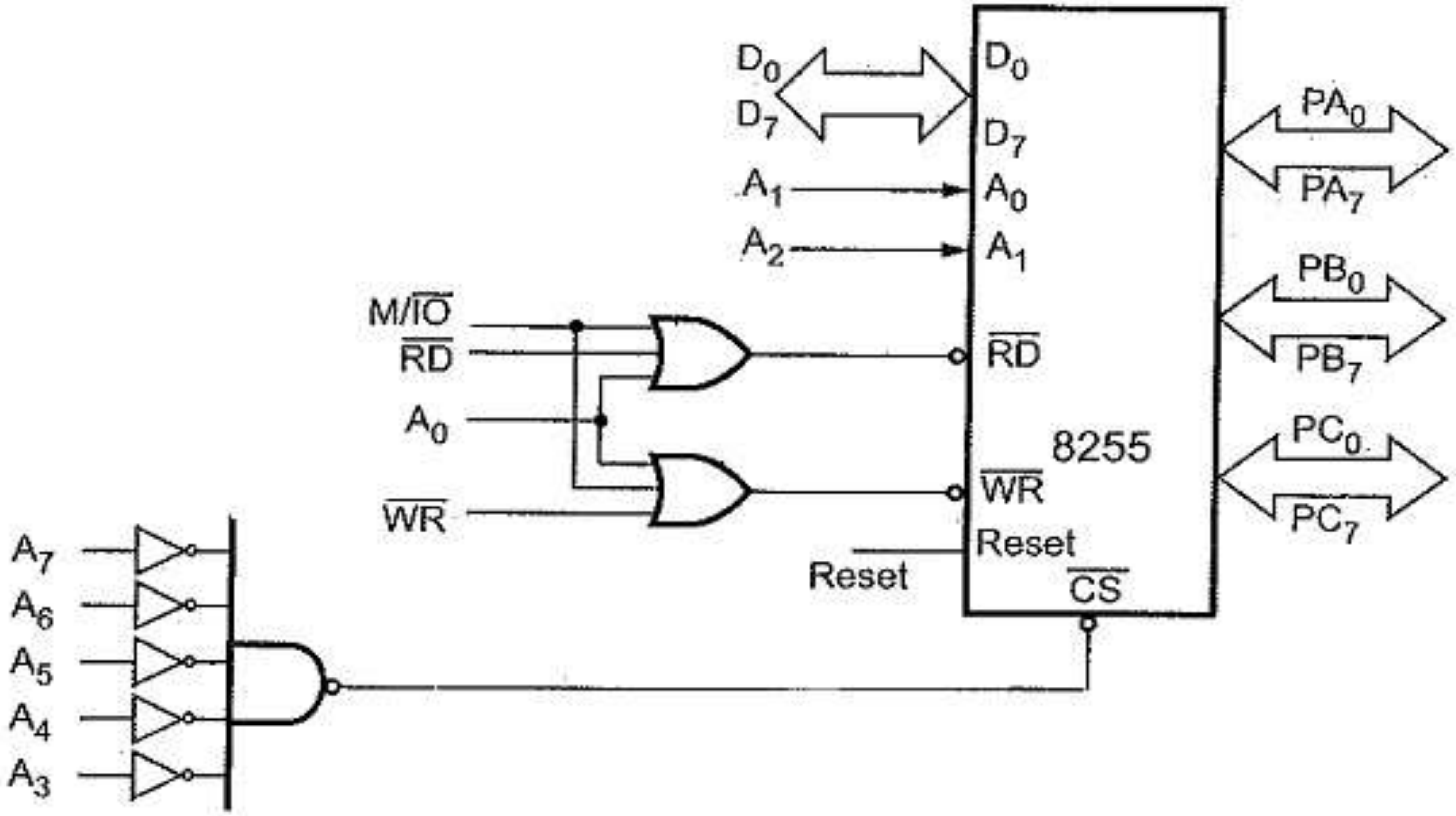
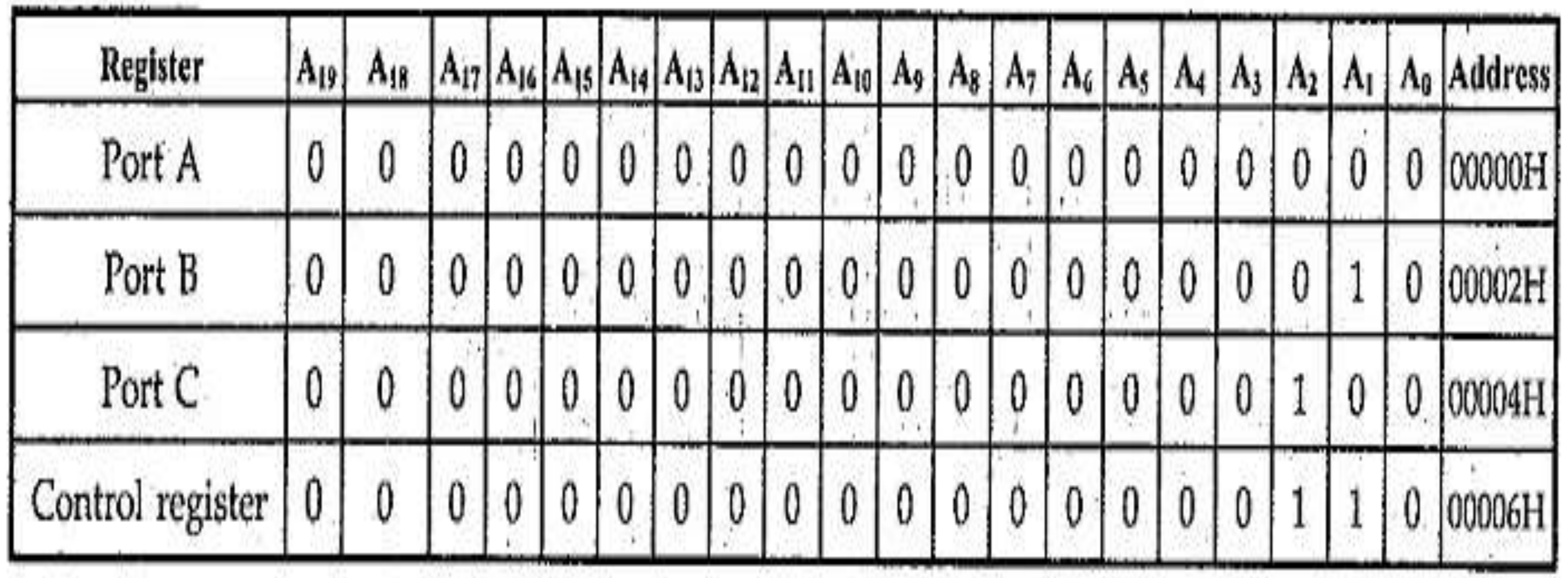


Fig. 14.19 I/O mapped I/O

8255 Interfacing with 8086 in Memory Mapped I/O

- In this type of I/O interfacing, the 8086 uses 20 address lines to identify an I/O device; an I/O device is connected as if it is a memory register. The 8086 uses same control signals and instructions to access I/O as those of memory. Fig. 14.20 shows the 8255 Interfacing with 8086 in memory mapped I/O technique. Here RD and WR signals are activated when M/IO signal is high, indicating memory bus cycle. Address lines A0 – A1 are used by 8255 for internal decoding. To get absolute address, all remaining address lines (A3 – A19) are used to decode the address for 8255. Other signal connections are same as in I/O napped I/O.

I/O Map :



Register	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address
Port A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000H
Port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	00002H
Port C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	00004H
Control register	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	00006H

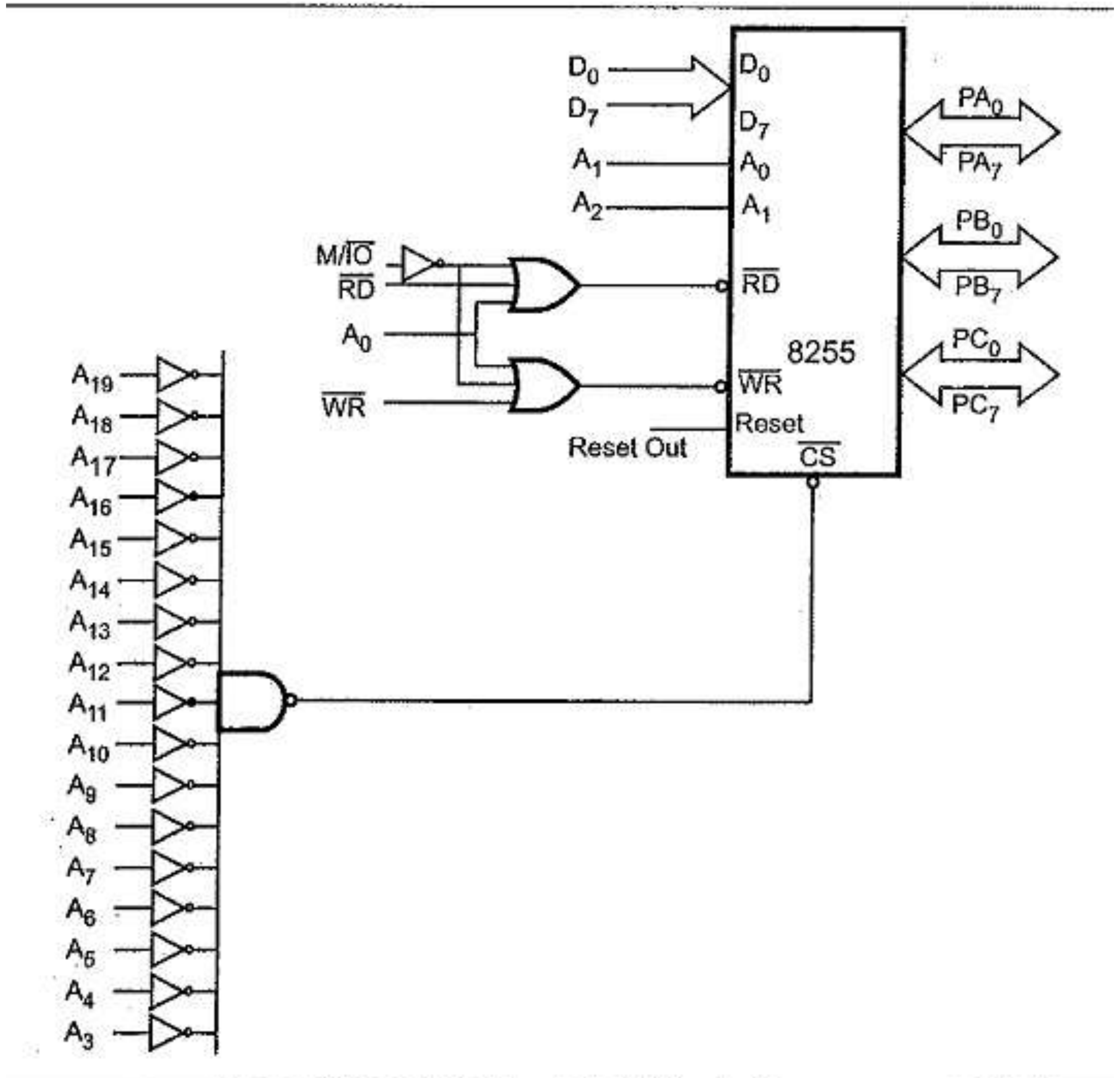


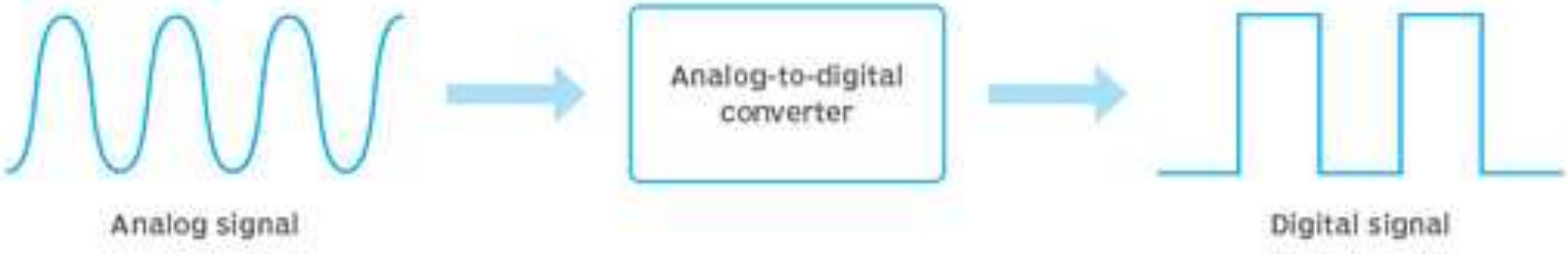
Fig. 14.20 Memory mapped I/O

Analog-to-digital conversion (ADC) 0808/0809

- A converter that is used to change the analog signal to digital is known as an analog to digital converter or ADC converter.

- An analog-to-digital converter changes an analog signal that's continuous in terms of both time and amplitude to a digital signal that's discrete in terms of both time and amplitude. The analog input to a converter consists of a voltage that varies among a theoretically infinite number of values. Examples are sine waves, the waveforms representing human speech and the signals from a conventional television camera.

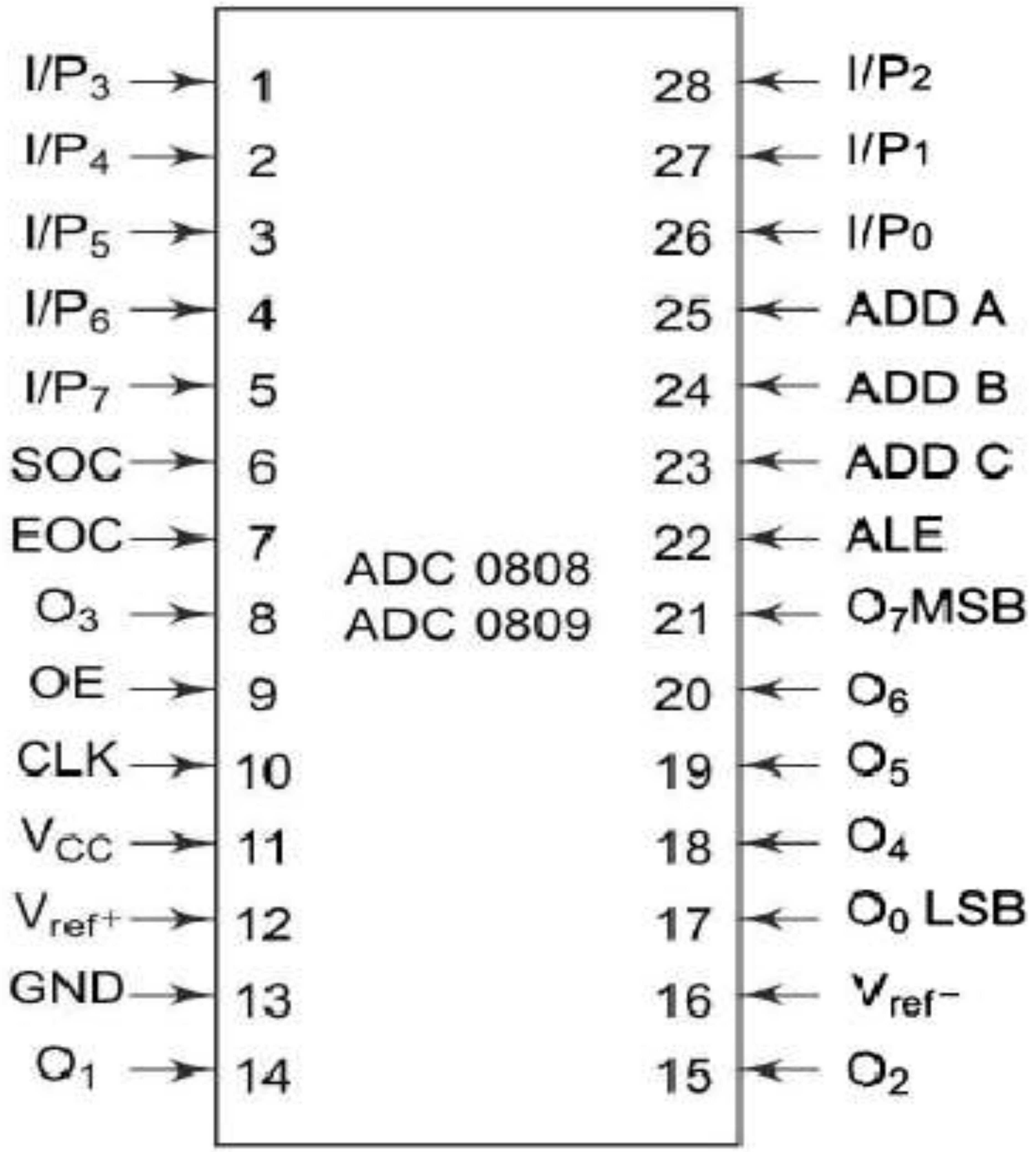
- The output of the analog-to-digital converter has defined levels or states. The number of states is almost always a power of two -- that is, 2, 4, 8, 16, etc. The simplest digital signals have only two states and are called binary. All whole numbers can be represented in binary form as strings of ones and zeros.



ADC interfacing

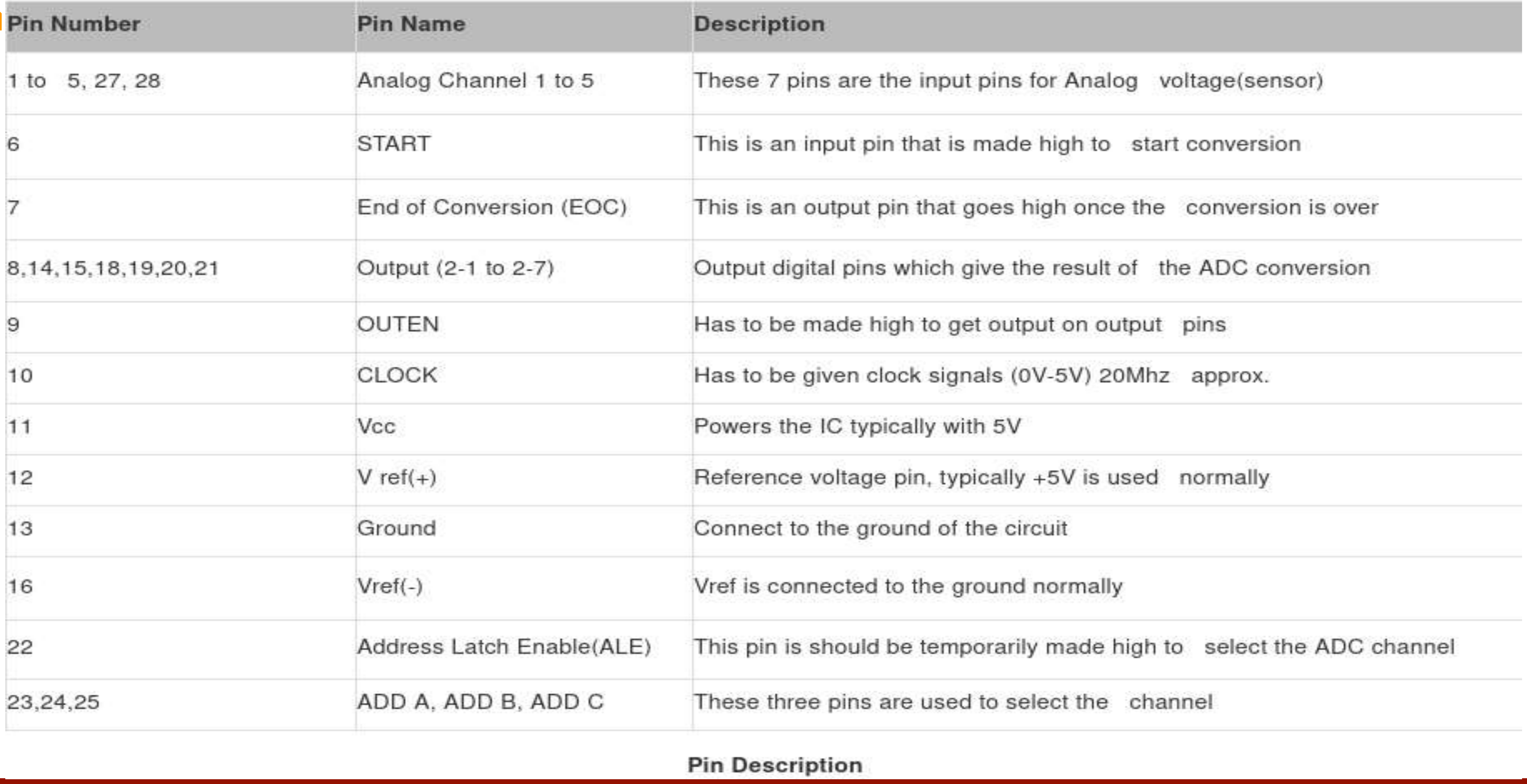
- General algorithm for ADC interfacing contains the following steps:
- 1. Ensure the stability of analog input, applied to the ADC.
- 2. Issue start of conversion pulse to ADC
- 3. Read end of conversion signal to mark the end of conversion processes.
- 4. Read digital data output of the ADC as equivalent digital output.
- 5. Analog input voltage must be constant at the input of the ADC right from the start of conversion till the end of the conversion to get correct results. This may be ensured by a sample and hold circuit which samples the analog signal and holds it constant for a specific time duration. The microprocessor may issue a hold signal to the sample and hold circuit.
- 6. If the applied input changes before the complete conversion process is over, the digital equivalent of the analog input calculated by the ADC may not be correct.

ADC Pin Dig..



I/P_0–I/P_7	Analog inputs
ADD A, B, C	Address lines for selecting analog inputs
O_7–O_0	Digital 8-bit output with O_7 MSB and O_0 LSB
SOC	Start of conversion signal pin
EOC	End of conversion signal pin
OE	Output latch enable pin, if high enable output
CLK	Clock input for ADC
V_{CC}, GND	Supply pins +5V and GND
V_{ref+} and V_{ref-}	Reference voltage positive (+5 Volts maximum) and Reference voltage negative (0V minimum)

ADC Pin Dig..



Pin Number	Pin Name	Description
1 to 5, 27, 28	Analog Channel 1 to 5	These 7 pins are the input pins for Analog voltage(sensor)
6	START	This is an input pin that is made high to start conversion
7	End of Conversion (EOC)	This is an output pin that goes high once the conversion is over
8,14,15,18,19,20,21	Output (2-1 to 2-7)	Output digital pins which give the result of the ADC conversion
9	OUTEN	Has to be made high to get output on output pins
10	CLOCK	Has to be given clock signals (0V-5V) 20Mhz approx.
11	Vcc	Powers the IC typically with 5V
12	V ref(+)	Reference voltage pin, typically +5V is used normally
13	Ground	Connect to the ground of the circuit
16	Vref(-)	Vref is connected to the ground normally
22	Address Latch Enable(ALE)	This pin is should be temporarily made high to select the ADC channel
23,24,25	ADD A, ADD B, ADD C	These three pins are used to select the channel

Pin Description

Interface ADC 0808 with 8086 using 8255 PPI

- Fig shows the ADC 0808 connection with 8255 PPI

Solution Figure 5.39 shows the interfacing connections of ADC0808 with 8086 using 8255. The analog input I/P$_2$ is used and therefore address pins A,B,C should be 0,1,0 respectively to select I/P$_2$. The OE and ALE pins are already kept at +5V to select the ADC and enable the outputs. Port C upper acts as the input port to receive the EOC signal while port C lower acts as the output port to send SOC to the ADC. Port A acts as a 8-bit input data port to receive the digital data output from the ADC. The 8255 control word is written as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Control word
1	0	0	1	1	0	0	0	= 98 H

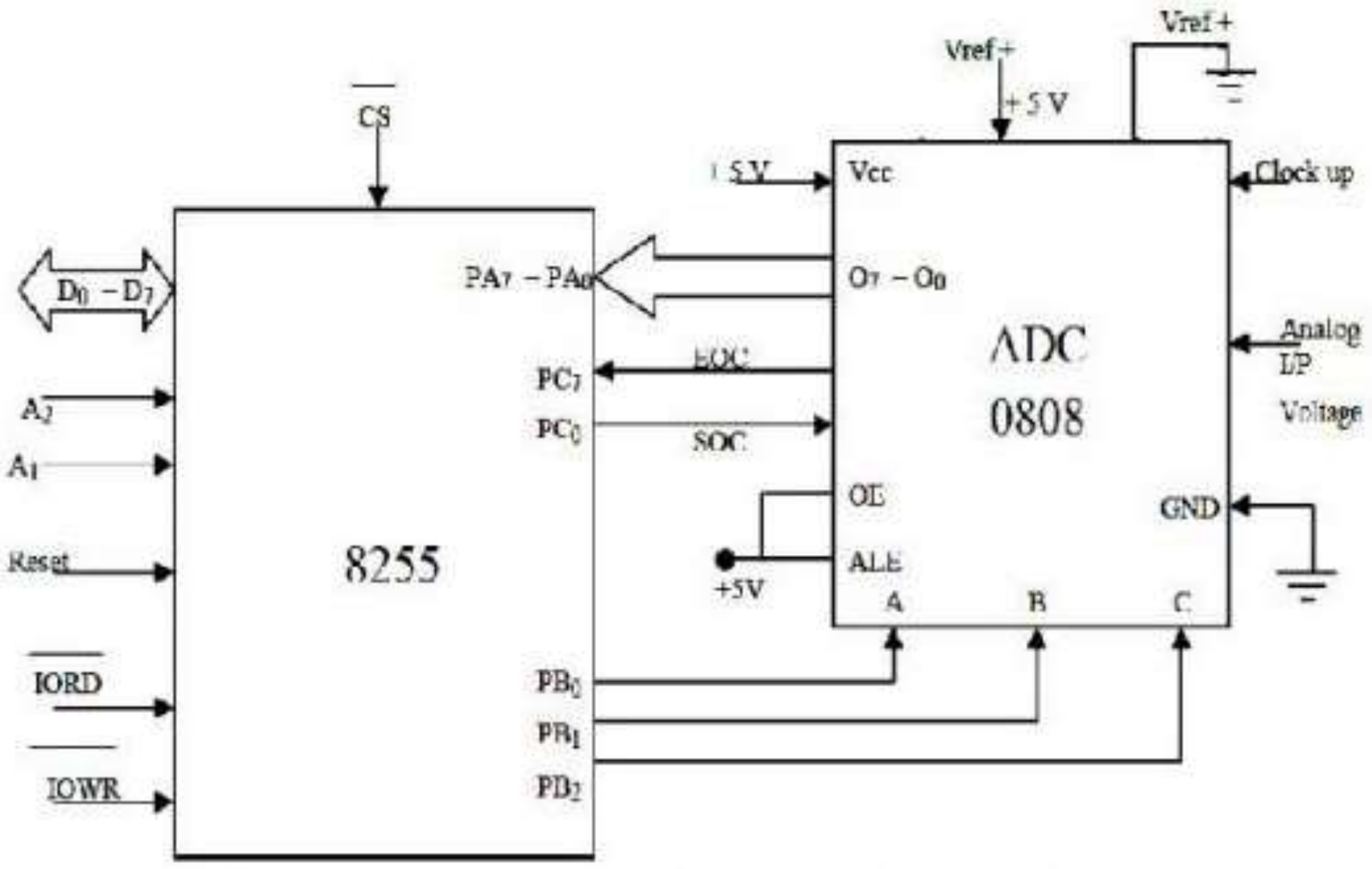
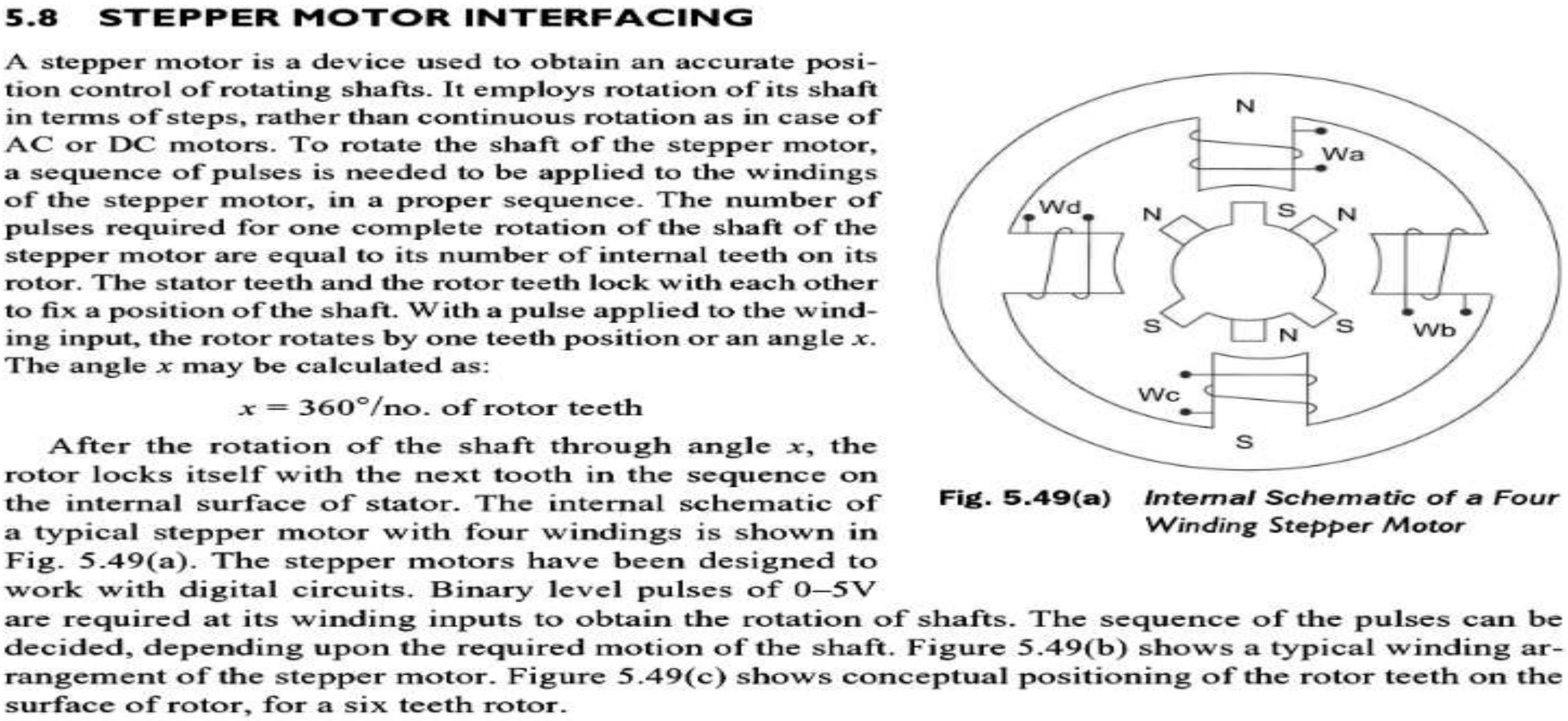


Fig. 5.4 Interfacing ADC with 8255 of microcontroller

Cont..



N
Wa
Wd
N
S
N
S
N
S
Wb
Wc
S

- ## 5.8 STEPPER MOTOR INTERFACING

A stepper motor is a device used to obtain an accurate position control of rotating shafts. It employs rotation of its shaft in terms of steps, rather than continuous rotation as in case of AC or DC motors. To rotate the shaft of the stepper motor, a sequence of pulses is needed to be applied to the windings of the stepper motor, in a proper sequence. The number of pulses required for one complete rotation of the shaft of the stepper motor are equal to its number of internal teeth on its rotor. The stator teeth and the rotor teeth lock with each other to fix a position of the shaft. With a pulse applied to the winding input, the rotor rotates by one teeth position or an angle x. The angle x may be calculated as:

$$x = 360°/\text{no. of rotor teeth}$$

After the rotation of the shaft through angle x, the rotor locks itself with the next tooth in the sequence on the internal surface of stator. The internal schematic of a typical stepper motor with four windings is shown in Fig. 5.49(a). The stepper motors have been designed to work with digital circuits. Binary level pulses of 0–5V are required at its winding inputs to obtain the rotation of shafts. The sequence of the pulses can be decided, depending upon the required motion of the shaft. Figure 5.49(b) shows a typical winding arrangement of the stepper motor. Figure 5.49(c) shows conceptual positioning of the rotor teeth on the surface of rotor, for a six teeth rotor.

Fig. 5.49(a) *Internal Schematic of a Four Winding Stepper Motor*

Cont..

❑

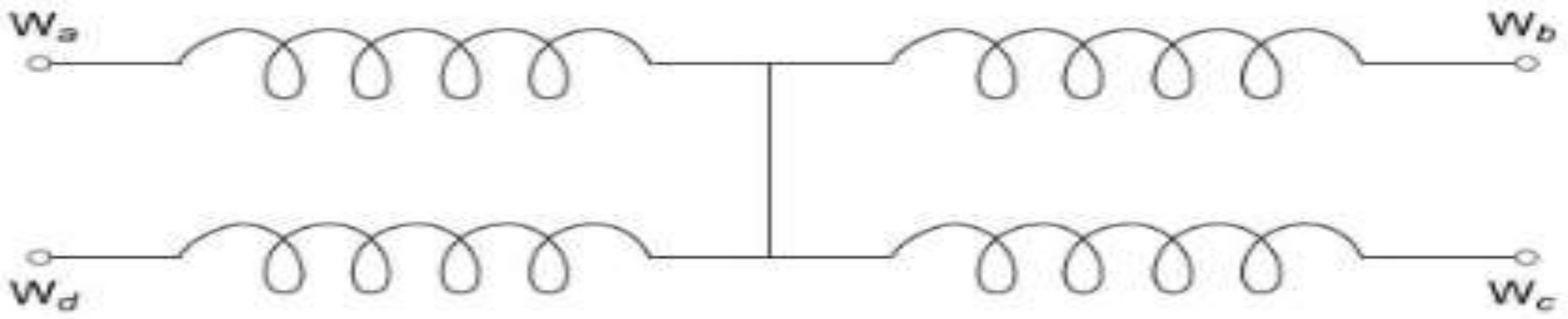


Fig. 5.49(b) *Winding Arrangement of a Stepper Motor*

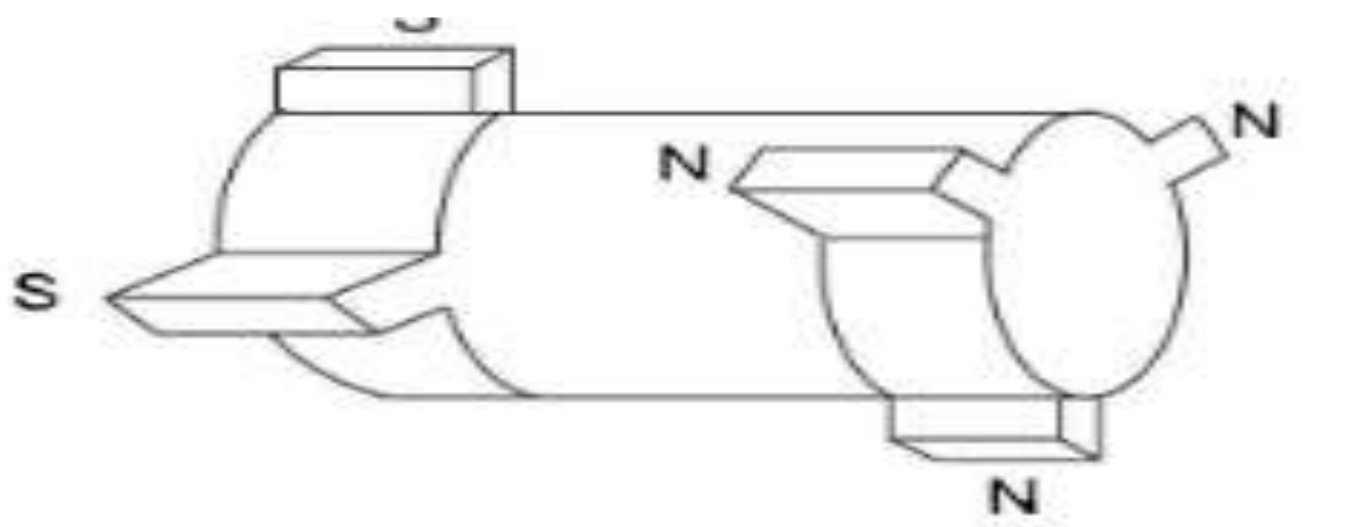


Fig. 5.49(c) *A Stepper Motor Rotor*

The circuit for interfacing a winding W_n with an I/O port is given in Fig. 5.50. Each of the windings of a stepper motor need this circuit for its interfacing with the output port. A typical stepper motor may have parameters like torque 3 kg-cm, operating voltage 12 V, current rating 0.2 A and a step angle 1.8°, i.e. 200 steps/revolution (number of rotor teeth).

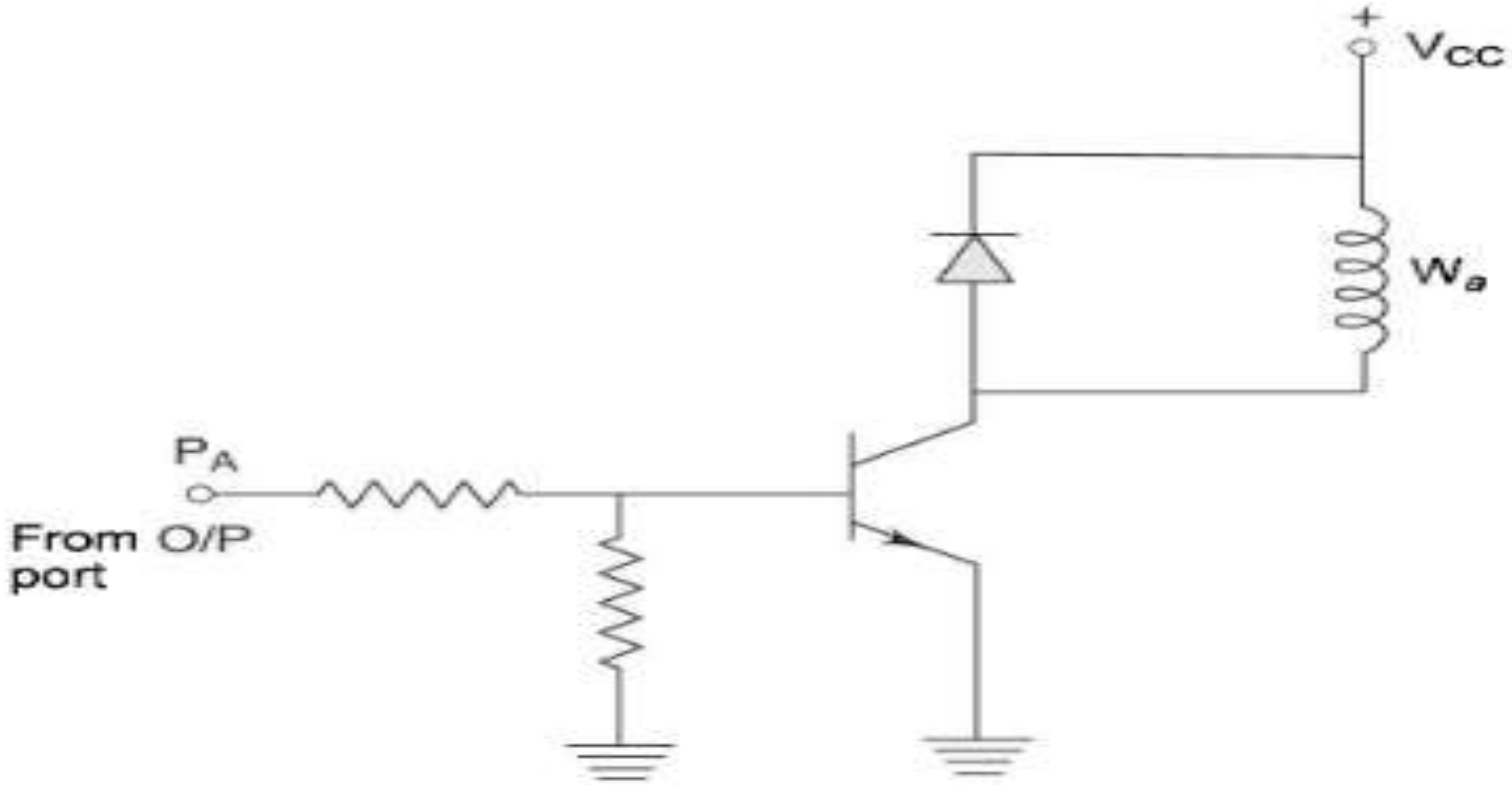
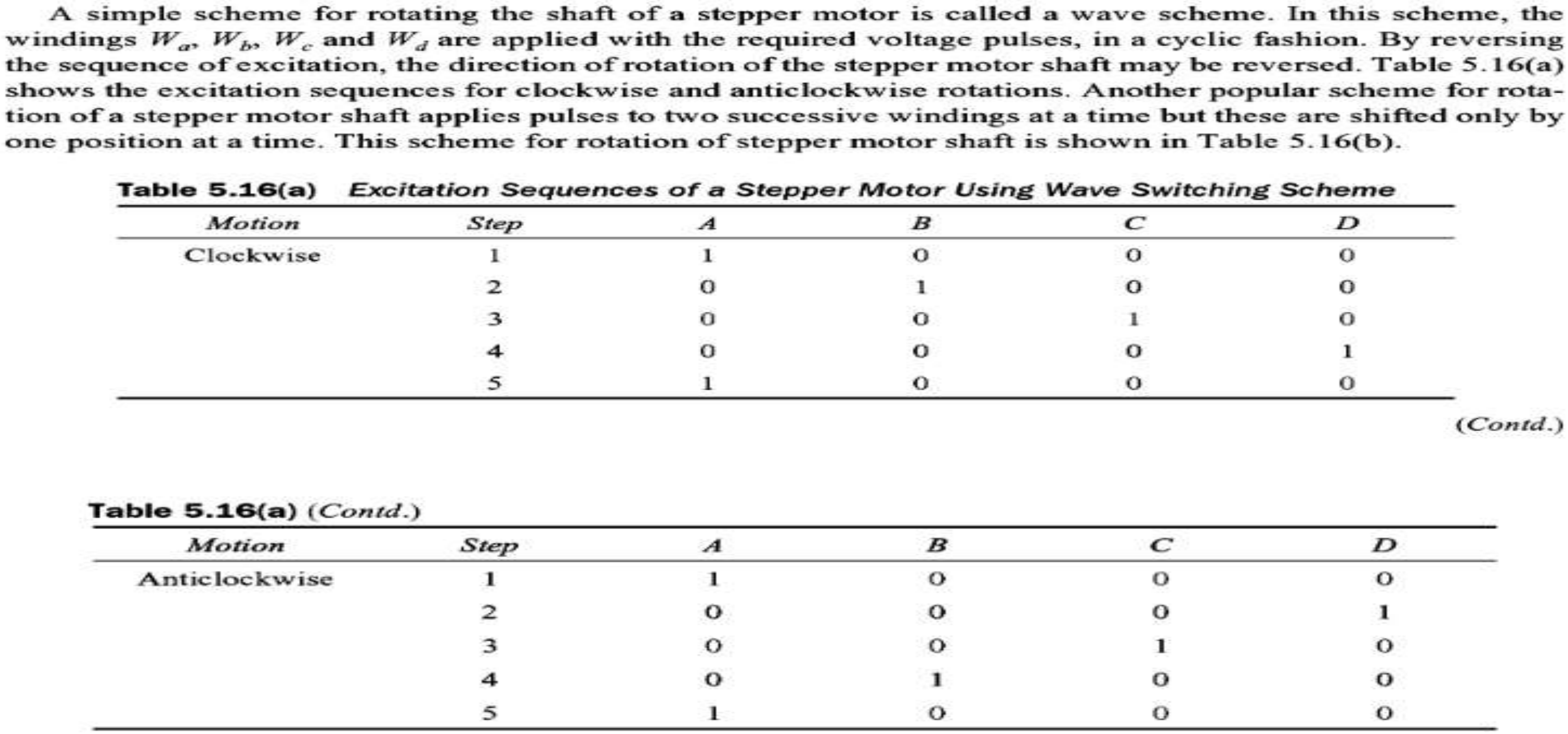


Fig. 5.50 *Interfacing Stepper Motor Winding W_a*

Cont..

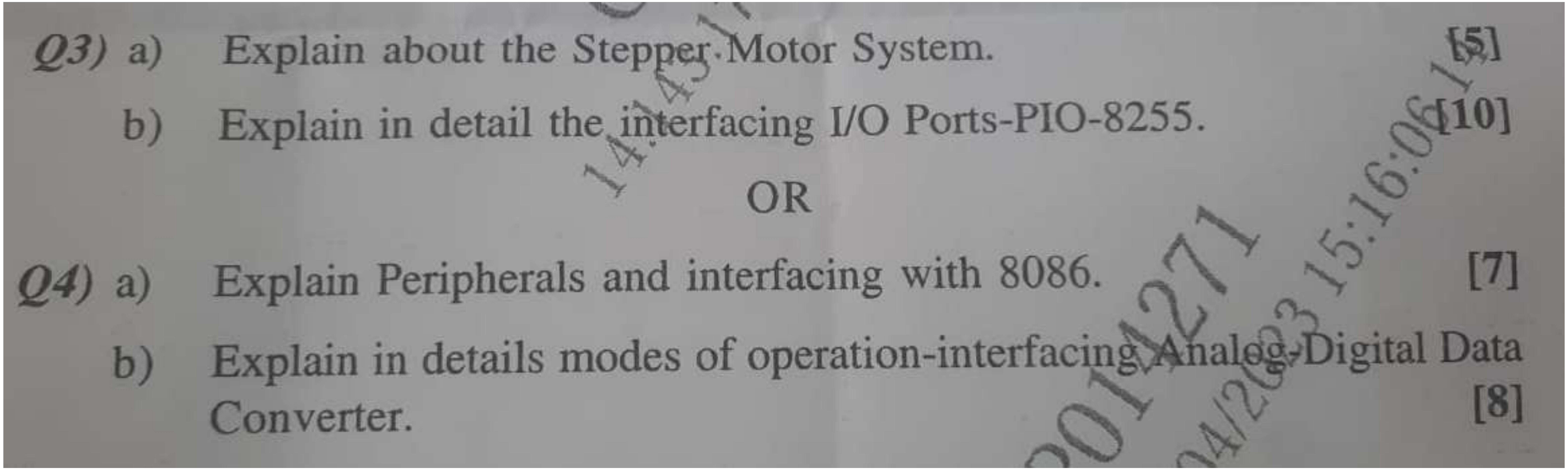


- A simple scheme for rotating the shaft of a stepper motor is called a wave scheme. In this scheme, the windings W_a, W_b, W_c and W_d are applied with the required voltage pulses, in a cyclic fashion. By reversing the sequence of excitation, the direction of rotation of the stepper motor shaft may be reversed. Table 5.16(a) shows the excitation sequences for clockwise and anticlockwise rotations. Another popular scheme for rotation of a stepper motor shaft applies pulses to two successive windings at a time but these are shifted only by one position at a time. This scheme for rotation of stepper motor shaft is shown in Table 5.16(b).

Table 5.16(a) *Excitation Sequences of a Stepper Motor Using Wave Switching Scheme*

Motion	*Step*	*A*	*B*	*C*	*D*
Clockwise	1	1	0	0	0
	2	0	1	0	0
	3	0	0	1	0
	4	0	0	0	1
	5	1	0	0	0
Anticlockwise	1	1	0	0	0
	2	0	0	0	1
	3	0	0	1	0
	4	0	1	0	0
	5	1	0	0	0

SPPU QP April 2023



Q3) a) Explain about the Stepper Motor System. [5]

b) Explain in detail the interfacing I/O Ports-PIO-8255. [10]

OR

Q4) a) Explain Peripherals and interfacing with 8086. [7]

b) Explain in details modes of operation-interfacing Analog-Digital Data Converter. [8]