Heaven's light is our guide

Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

Course Title: Peripherals & Interfacings

Lecture 1: Introduction to Microprocessor and it's Components

Microcomputer

- A microcomputer is a small, relatively inexpensive computer with a microprocessor as it's central processing unit (CPU).

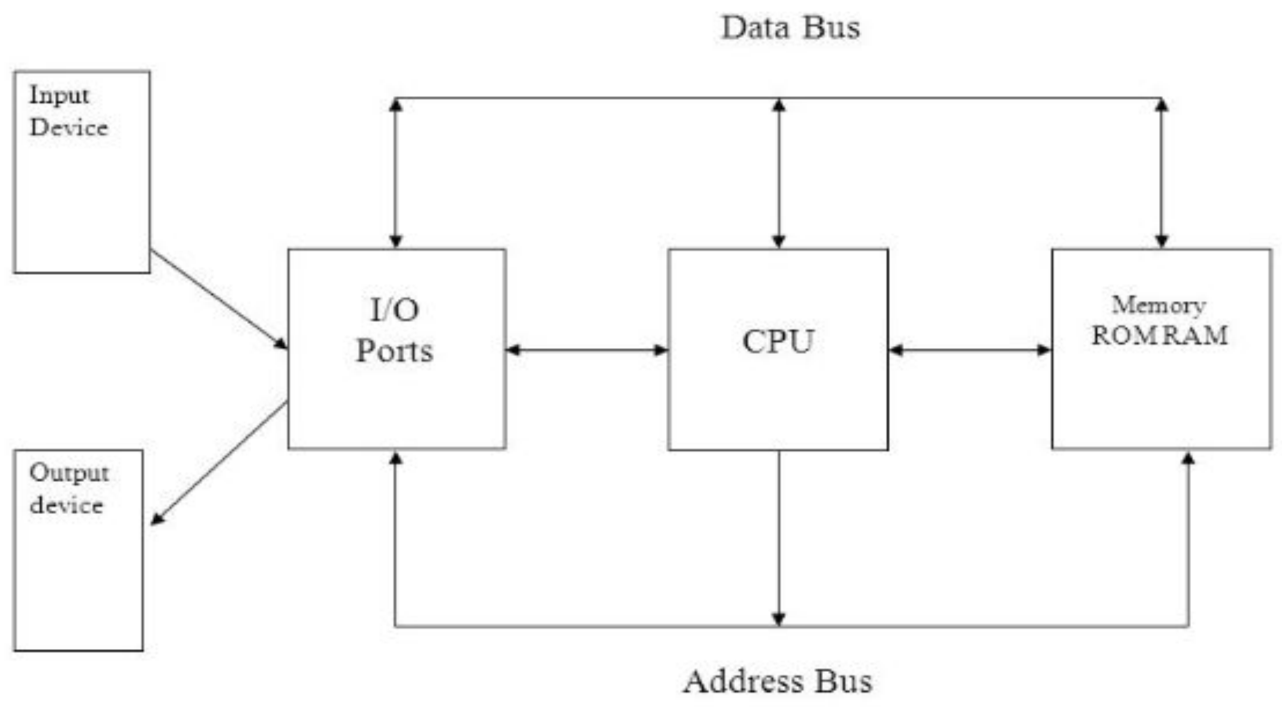


Fig 1: Block Diagram of a microcomputer

Bus

- Bus is a communication system that transfers data between components inside a computer, or between computers.
- All the peripherals are connected to microprocessor through Bus.

System Bus

- A system bus is a single computer bus that connects the major components of a computer system.

- There are three types of system bus:

1) Address bus
2) Data bus
3) Control bus

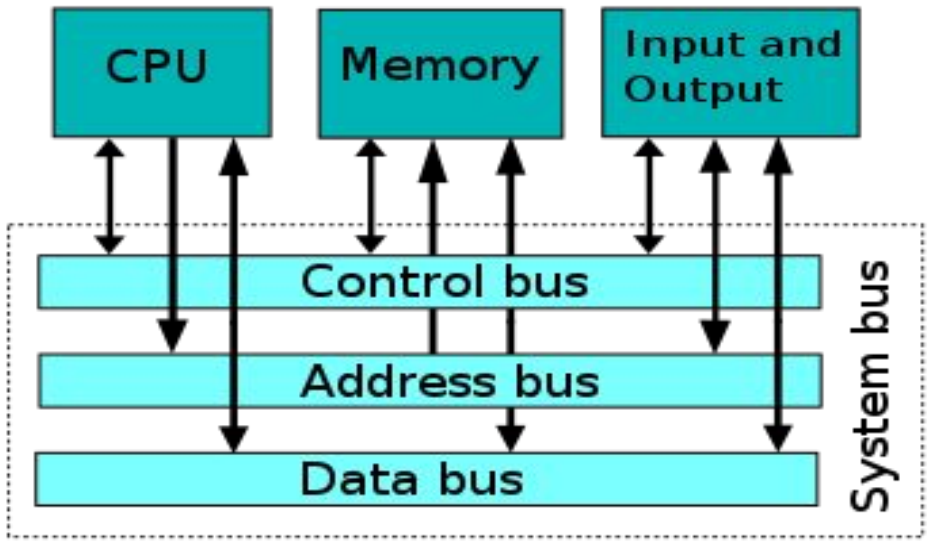


Fig 2: Interconnection of system bus

Address Bus

- It is a group of conducting wires which carries address only.
- Address bus is unidirectional.
- Because data flows only in one direction, from microprocessor to memory or from microprocessor to Input/output devices (that is, Out of Microprocessor).

Data Bus

- It is a group of conducting wires which carries Data only.
- Data bus is bidirectional.
- Because data flows in both directions, from microprocessor to memory or Input/ Output devices and from memory or Input/ Output devices to microprocessor.

Control Bus

- It is a group of conducting wires, which is used to generate timing and control signals to control all the associated peripherals.
- Microprocessor uses control bus to process data, that is what to do with selected memory location.
- Some control signals are Memory read, Memory write, I/O read, I/O Write, Opcode fetch etc.

Port

- A computer port is a connection point or interface between a computer and an external or internal device.
- Internal ports may connect such devices as hard drives, CD ROM, or DVD drives.
- External ports may connect modems, printers, mouse, and other devices.

Peripherals

- A peripheral device is an auxiliary device that connects to and works with the computer to either put information into it or get information out of it.
- A peripheral device may also referred as external peripheral, integrated peripheral, auxiliary component or input-output (I/O) device.
- Examples: keyboard, mouse, graphics card, microphone, loudspeaker, image scanner, printer etc.

Interface

- In terms of software, an interface is a program that allows a user to interact with computers in person or over a network.
- An example is Graphical User Interface (GUI).
- In computing, an interface is a shared boundary across which two or more separate components of a computer system exchange information.
- The exchange can be between computer software, hardware, peripheral devices, humans and combinations of these.

Computer Interfacing

- **Computer Interfacing:** It is an art of connecting computers and peripherals.
- **Microprocessor Interfacing:** Interfacing a microprocessor is to connect it with various peripherals to perform various operations to obtain a desired output.
- There are mainly two types of microprocessor interfacing:

1) Memory interfacing
2) I/O interfacing

Microprocessor Interfacing

1) **Memory interfacing:** Memory interfacing is used when the microprocessor needs to access memory frequently for reading and writing data stored in the memory.

2) **I/O interfacing:** I/O interfacing is received by connecting input and output devices with the microprocessor.

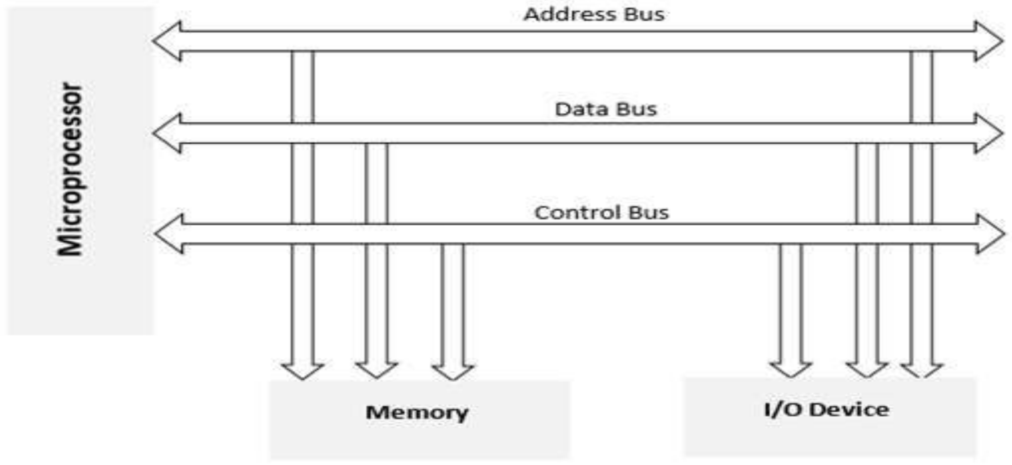


Fig 3: Block diagram of memory and i/o interfacing

Explanation of few Basic Terms

- **Address:** An Address is a pattern of 0’s and 1’s that represents a specific location in memory or a particular I/O device.
- Typical 8-bit microprocessors have 16 address lines, and, these 16 lines can produce unique 16-bit patterns from 0000 0000 0000 0000 to 1111 1111 1111 1111, representing 65,536 different address combinations.

Explanation of few Basic Terms

- **Addressing Mode:** Addressing mode is the manner in which the microprocessor determines the operand (data) and destination addresses during execution of an instruction.
- **Arithmetic Logic Unit:** An Arithmetic-logic unit (ALU) is a digital circuit that performs arithmetic and logic operations on two n-bit digital words.
- The value of n can be 4, 8, 16, 32, or 64.
- Typical operations performed by an ALU are addition, subtraction, AND, OR, and comparison of two n-bit digital words.
- The size of the ALU defines the size of the microprocessor. For example, a 32-bit microprocessor contains a 32-bit ALU.

Explanation of few Basic Terms

- **Bit:** The bit values are easily implemented in electronic and magnetic media by two-state devices whose states portray either of the binary digits 0 and 1.
- Examples of such two-state devices are a transistor that is conducting or not conducting, , a capacitor that is charged or discharged, and a magnetic material that is magnetized north to south or south to north.

Explanation of few Basic Terms

- **Bit Size:** Bit size refers to the number of bits that can be processed simultaneously by the basic arithmetic circuits of a microprocessor.
- **Clock:** The microprocessor requires synchronization among its components, and this is provided by a clock or timing circuits.

Explanation of few Basic Terms

- **Instruction set:** The instruction set of a microprocessor is a list of commands that the microprocessor is designed to execute.
- Typical instructions are ADD, SUBTRACT, and STORE.
- Individual instructions are coded as unique bit patterns which are recognized and executed by the microprocessor.
- If a microprocessor has 3 bits allocated to the representation of instructions, the microprocessor will recognize a maximum of eight, different instructions.
- The microprocessor will then have a maximum of eight instructions in its instruction set.

Explanation of few Basic Terms

- **Memory Management Unit:** Memory Management Unit allows programmers to write programs much larger than could fit in the main memory space available to the microprocessor.
- The programs are simply stored in a secondary device such as a hard disk.
- Portions of the programs are swapped into the main memory as needed for execution by the microprocessor.

Internal Architecture of 8086

Memory

Extra segment – It contains initialized and uninitialized global variables.
Code segment – It contains program section.
Stack segment - It contains local variables.
Data segment - It contains initialized and uninitialized global variables.

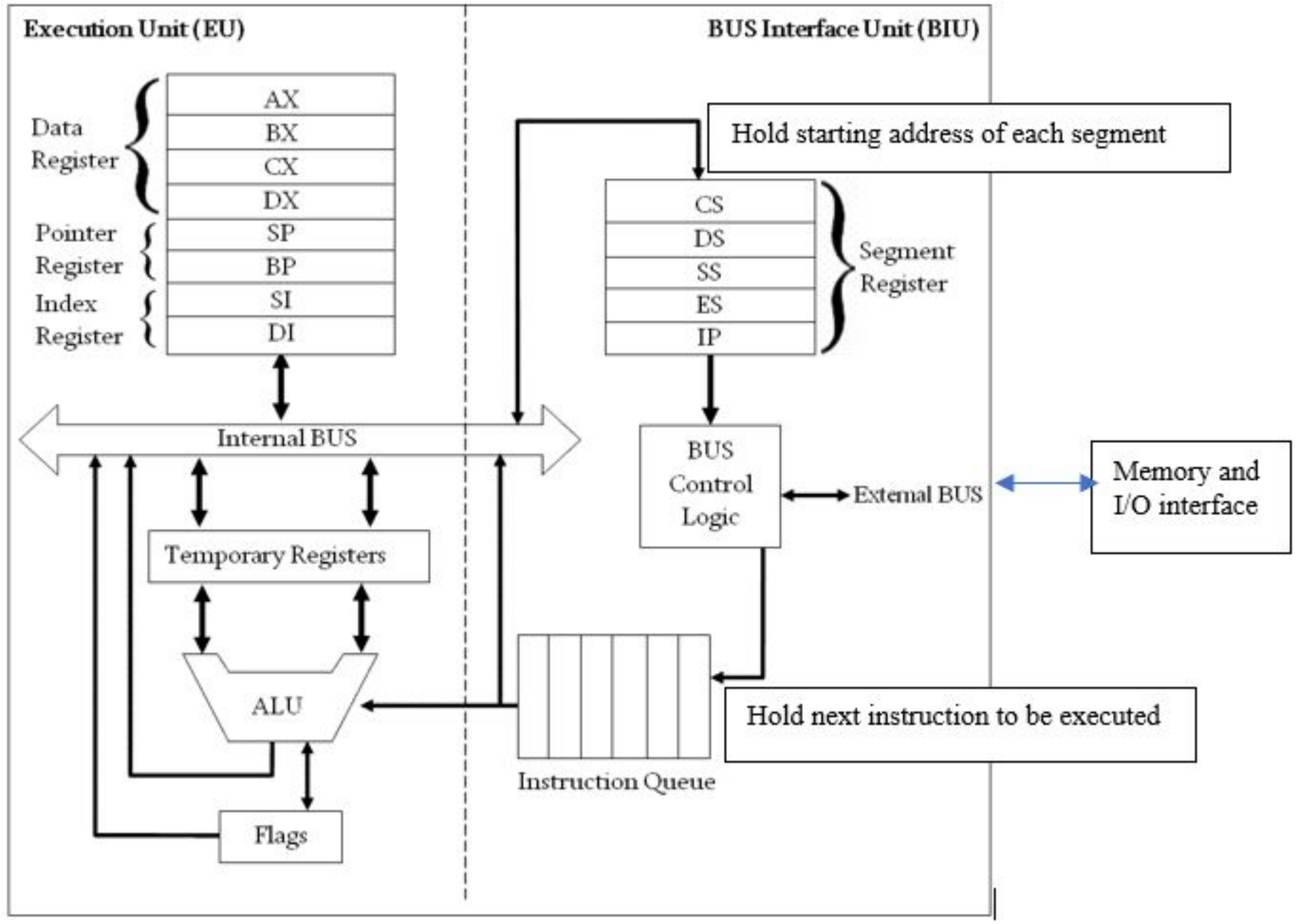


Fig 4: Internal architecture of 8086

Internal Architecture of 8086

- It has mainly two functional parts. Dividing the works between two units speeds up the processing.
- **Bus Interface Unit:** Main function of BIU is divided into three sections.

1) Calculating the 20-bit physical address.
2) Provide the interface with memory and I/O devices.
3) Provide pipelining by pre-fetching instructions and stored in the instruction queue.

- The BIU sends out addresses, fetches instructions from memory, reads data from ports and memory, and writes data to ports and memory.
- BIU handles all transfers of data and addresses on the buses for the execution unit.

Internal Architecture of 8086

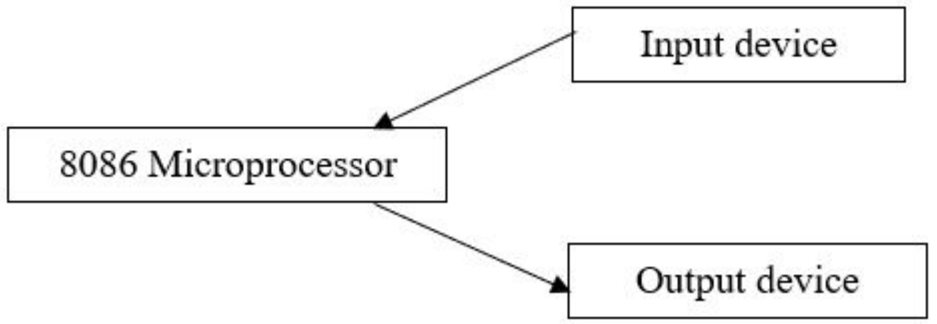
- **Execution Unit:** The execution unit of 8086 tells the BIU where to fetch instructions or data, decodes instructions, and executes instructions

1) Process the instructions or data provided by the BIU.
2) Stores the result in the internal registers.
3) Update the offset address of the registers and status flags.

I/O interfacing in Microprocessor

- ❖ Data is transferred from input-output devices to the microprocessor.
- ❖ There are three different ways in which data transfer can take place.

1) **Program controlled I/O**
2) **Interrupt program controlled I/O**
3) **Hardware controlled I/O**



I/O interfacing in Microprocessor

- **Program controlled I/O:** In program controlled I/O, the transfer of data is completely under the control of the microprocessor program.
- This means that the data transfer takes place only when an Input- Output Transfer Techniques instructions executed.
- The input-output instructions are written as **IN** and **OUT**.
- In most of the cases it is necessary to check whether the device is ready for data transfer or not.
- To check this, microprocessor polls the status bit associated with the I/O device.

I/O interfacing in Microprocessor

- **Interrupt program controlled I/O:** In interrupt program-controlled approach, when a peripheral is ready to transfer data, it sends an interrupt signal to the interrupt pin of the Microprocessor.
- When interrupted, the microprocessor stops the execution of the program and transfers the program control to an interrupt service routine which is written into the memory location.
- This interrupt service routine performs the data transfer.
- After the data transfer, it returns control to the main program at the point it was interrupted.

I/O interfacing in Microprocessor

- **Hardware controlled I/O:** To increase the speed of data transfer between memory and I/O, the hardware controlled I/O is used.
- It is commonly referred to as direct memory access (DMA).
- The hardware which controls this data transfer is commonly known as DMA controller.
- The DMA controller sends a HOLD signal to the microprocessor to initiate data transfer. In response to HOLD signal, microprocessor releases its data, address and control buses to the DMA controller.
- Then the data transfer is controlled at high speed by the DMA controller without the intervention of the microprocessor.
- After data transfer, DMA controller sends low on the HOLD pin, which gives the control of data, address, and control buses back to the microprocessor.
- This type of data transfer is used for large data transfers.

Program Controlled I/O Interfacing

- The instruction set contains instructions that transfer information to an I/O device and read information from an I/O device.
- It has 2 instructions for this purpose.

1) **IN:** Read information from an i/o device.
2) **OUT:** Transfer information to an i/o device.

- It has two bytes of instruction.
- One byte is for opcode and another byte is for port address of the input/output device.
- Both the instructions perform data transfer using accumulator (16-bit register-AX).

Program Controlled I/O Interfacing

- IN instruction, data is transferred from the particular port address/number to the accumulator.
- OUT instruction, data is transferred from the accumulator to the port address/number.
- The external i/o device of 8086 decodes the port number to find the address of the i/o device.

IN AL, 19h; 8 bits are saved to AL from I/O port 19h.

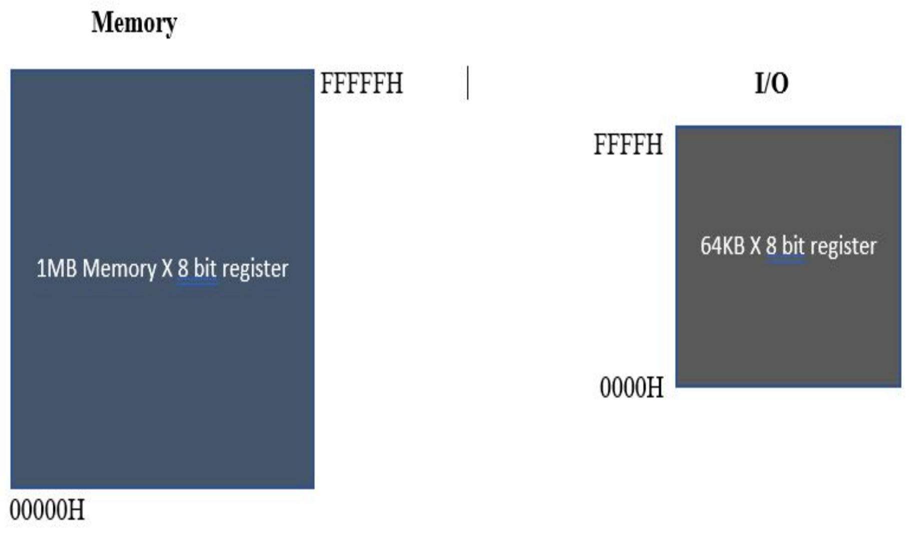
OUT 19h, AL; 16 bits are written to I/O port 19h.

- For 8-bit port number, the address line will be used A_0-A_7. Address range will be 00h to FFh. Remaining address line will have zeros. For 16-bit port number, the address line will be used A_0-A_{15}. Address range will be 0000h to FFFFh.

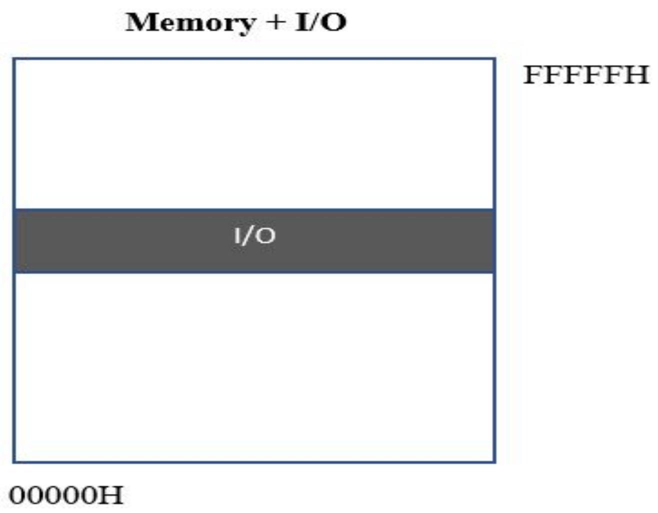
Program Controlled I/O Interfacing

- I/O devices are interfaced by using two methods.

1) I/O mapped I/O interfacing/Isolated/Direct.



2) Memory mapped I/O interfacing.



Program Controlled I/O Interfacing

Differences between I/O mapped I/O and memory mapped I/O:

I/O mapped I/O interfacing	**Memory mapped I/O interfacing**
1. I/O devices are separate from memory.	1. I/O device is treated as memory location.
2. IN and OUT instructions are used.	2. Memory instructions are used for data transfer.
3. I/O read and write instructions are used.	3. Memory read and write instructions are used.
4. Can use entire memory.	4. Overall memory size is reduced.
5. Address of I/O and memory will be separate.	5. Same address can be used for memory and I/O devices.
6. 16-bit address. Address range 0000h to FFFFh.	6. 20-bit address. Address range 00000h to FFFFFh.

Interfacing Memory with 8086 Microprocessor

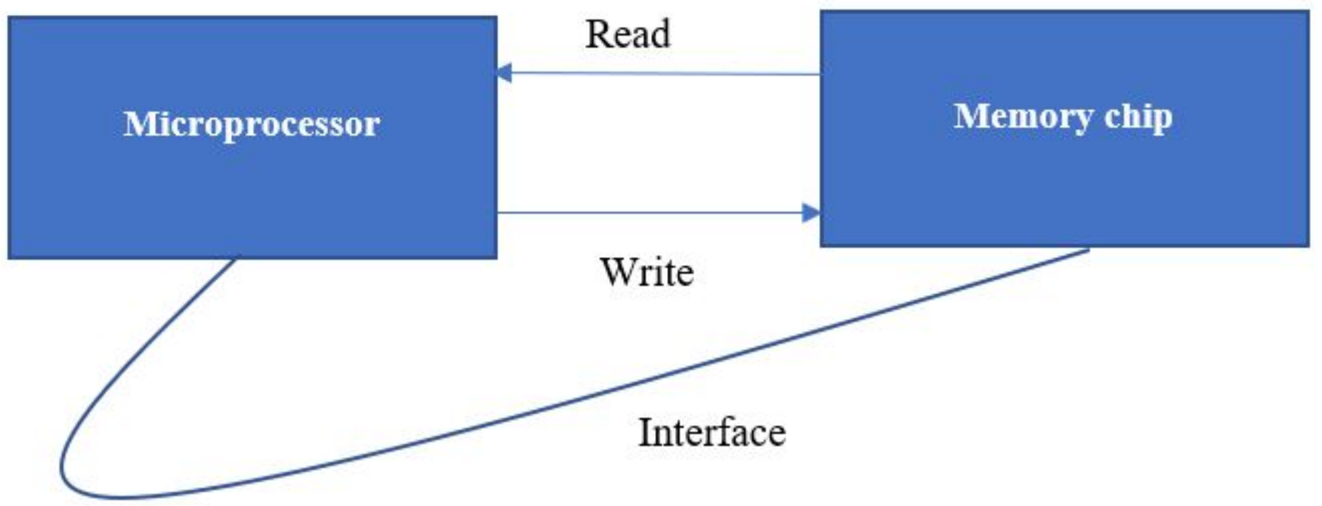
- **Memory** is nothing but a storage device for data, instructions and programs. Memory is an integral part of a microprocessor.

- Memory mainly two types:

1) **Read Only Memory (ROM):** Only read signal available. Program that does not need any changes. Manufacturer saved all types of instructions and information during manufacturing. It stores different types of system software, system programs and data. Examples: PROM, EPROM, EEPROM etc.

2) **Random Access Memory (RAM):** It only stores temporary data and process the user program. Both read and write signals are available. It is volatile.

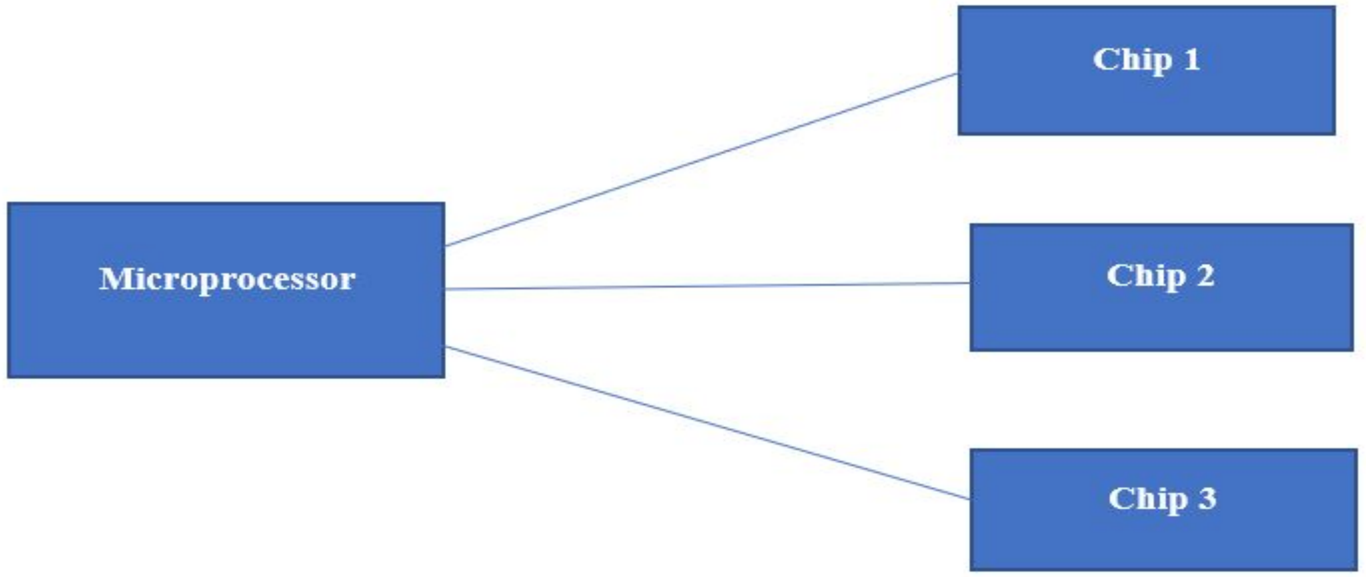
Interfacing Memory with 8086 Microprocessor



- For interfacing we need two signals, one for read and another for write.
- Microprocessor needs to access memory to get instruction codes and data.
- Instruction code means the program that are written to memory.
- Microprocessor will initiate read and write signals according to the operation.
- For read operation it has **MEMR** and for write operation it has **MEMW**.

Interfacing Memory with 8086 Microprocessor

- There can be many memory chips connected to the microprocessor.
- So, there should be a logic of selecting the memory chips that the microprocessor wants to communicate for read or write operation.
- To communicate Microprocessor will have **RD** and **WR** signals. Every chip has a **CS** (chip select) signal.



Interfacing Memory with 8086 Microprocessor

- To do the above job it is necessary to have a device or a circuit which performs this task reading or writing the memory and selecting the memory chips it is called interfacing device.
- It is linked with the memory so it can be called as **Memory Interfacing Device.**
- **Memory Interfacing Device:** It basically interfaces the memory with the microprocessor.
- The path of communication between memory and microprocessor is provided by the memory interfacing device.

Interfacing Memory with 8086 Microprocessor

❖ **Memory interfacing involves three different tasks:**

1) **To select the required chip (Chip Select).**

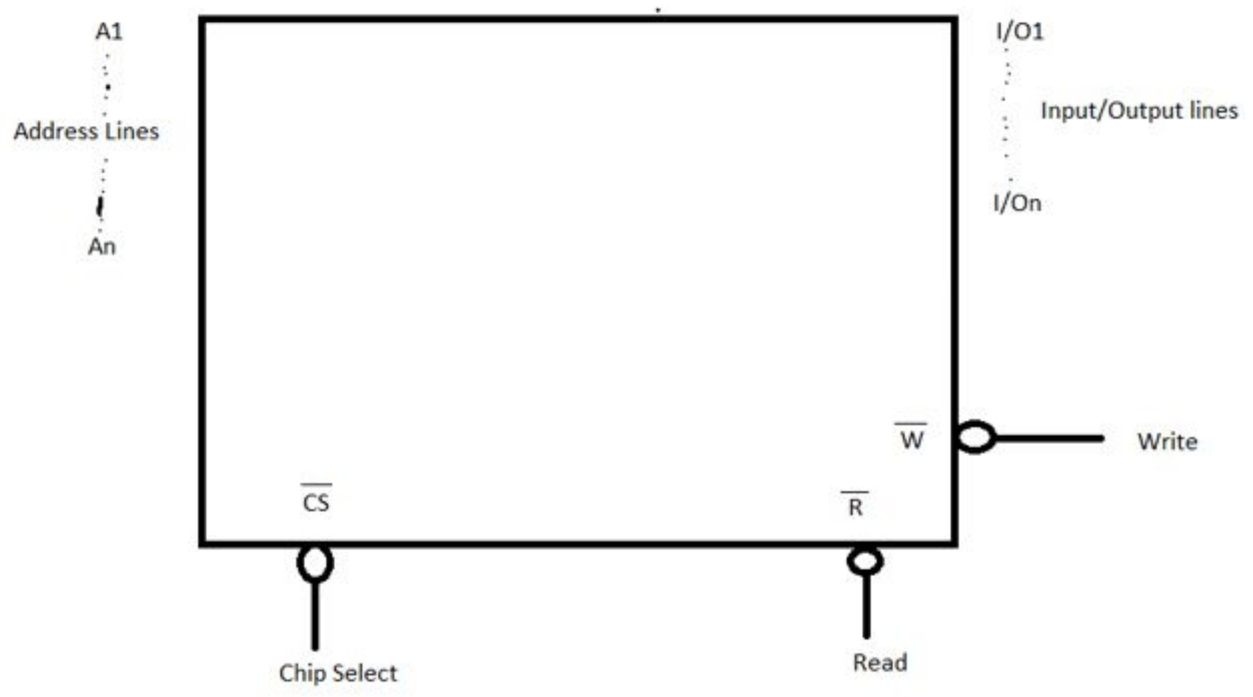
2) **Identify the required register** – Memory is composed of various memory registers. Each memory register has different memory location. Addresses are defined for various memory location.

3) **Must enable appropriate buffers** – Read, write signals and data, address buffer.

Interfacing Memory with 8086 Microprocessor

- **Memory device:** Generally, it has four components.

1) Address lines
2) Input-Output lines
3) Selection input
4) Control input



Interfacing Memory with 8086 Microprocessor

- **Address lines** select the memory location within the memory device A_0 ……$A_{n.}$ No of address lines indicate total memory capacity of the memory device. 20 address lines means total capacity of memory is 1 Megabytes.

- **Input-Output lines** Memory device may have separate i/o lines or common bi-directional i/o lines. Using these lines data can be transferred in any direction such as I/O_0……………….$I/O_{n.}$ Input-output lines carrying data into the memory or outside of the memory. So they also called data lines and denoted as D_0……………….$D_{n.}$ Size of each memory location depends on number of data lines. Suppose a memory 2K X 8 means memory size is 2 kilobytes and 8 bits can be stored in each memory location.

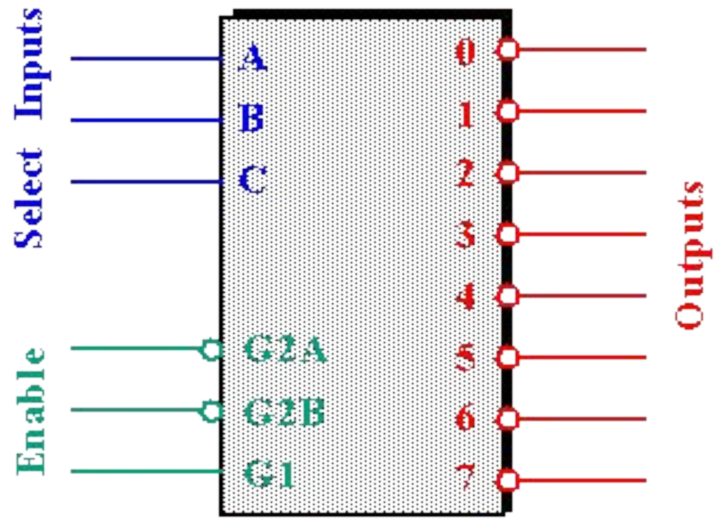
Interfacing Memory with 8086 Microprocessor

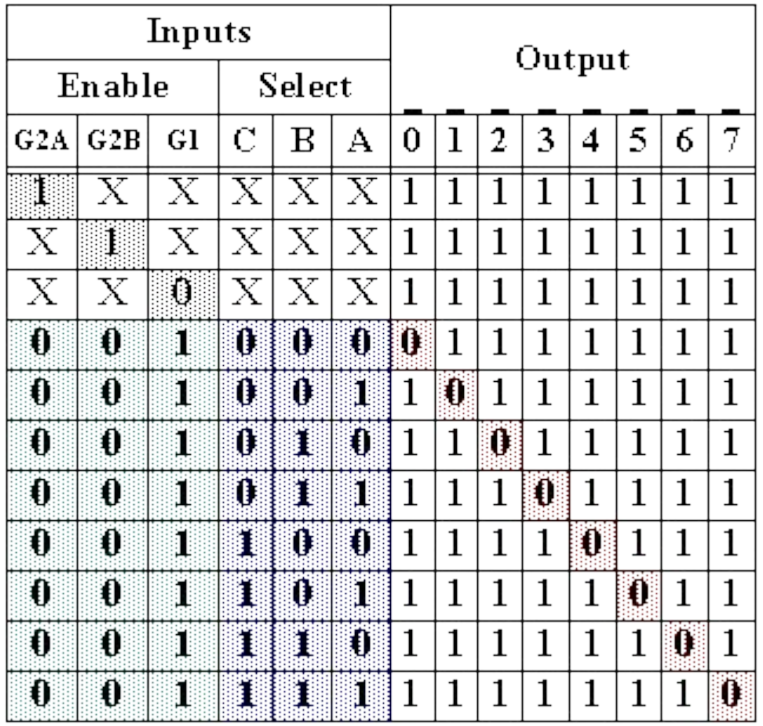
- **Chip Select** Memory device may contain one or more inputs which are used to select the memory device or enable it. Chip select (CS) means it will active at logic 0 (active low). If there are more than one chip select pins, then all of the pins should be activated at logic 0.

- **Control inputs** for ROM there is only one type of signal is available that is RD signal and it will be activated at logic 0. For RAM there is two types of signal available RD and WR signal, both are active low signal.

Interfacing Memory with 8086 Microprocessor

- **Memory Interfacing Device**: 3 to 8 line decoder
- Only one of eight outputs will active at any time.
- Selection inputs A, B and C select which output pin is going to be low or active.
- Here, G2A and G2B are active low means it will be active on logic 0.





Inputs						Output							
Enable			Select										
G2A	G2B	G1	C	B	A	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

Interfacing Memory with 8086 Microprocessor

- ❖ Interface a 64k X 8 EPROM to the 8086 microprocessor system using eight 8k X 8 EPROM chips.

Memory Interfacing

⊛ Solution:-

$$8K\times 8 = 2^3 \cdot 2^{10} \times 8$$

$$= 2^{13} \times 8 \text{ — Data lines } (D_0 - D_7)$$

Address lines $(A_0 - A_{12})$

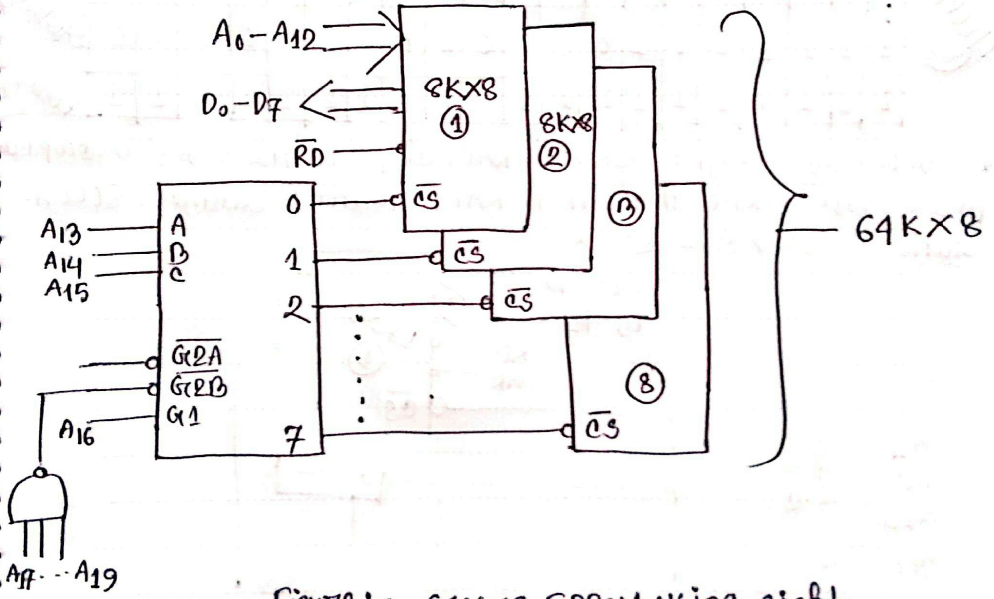


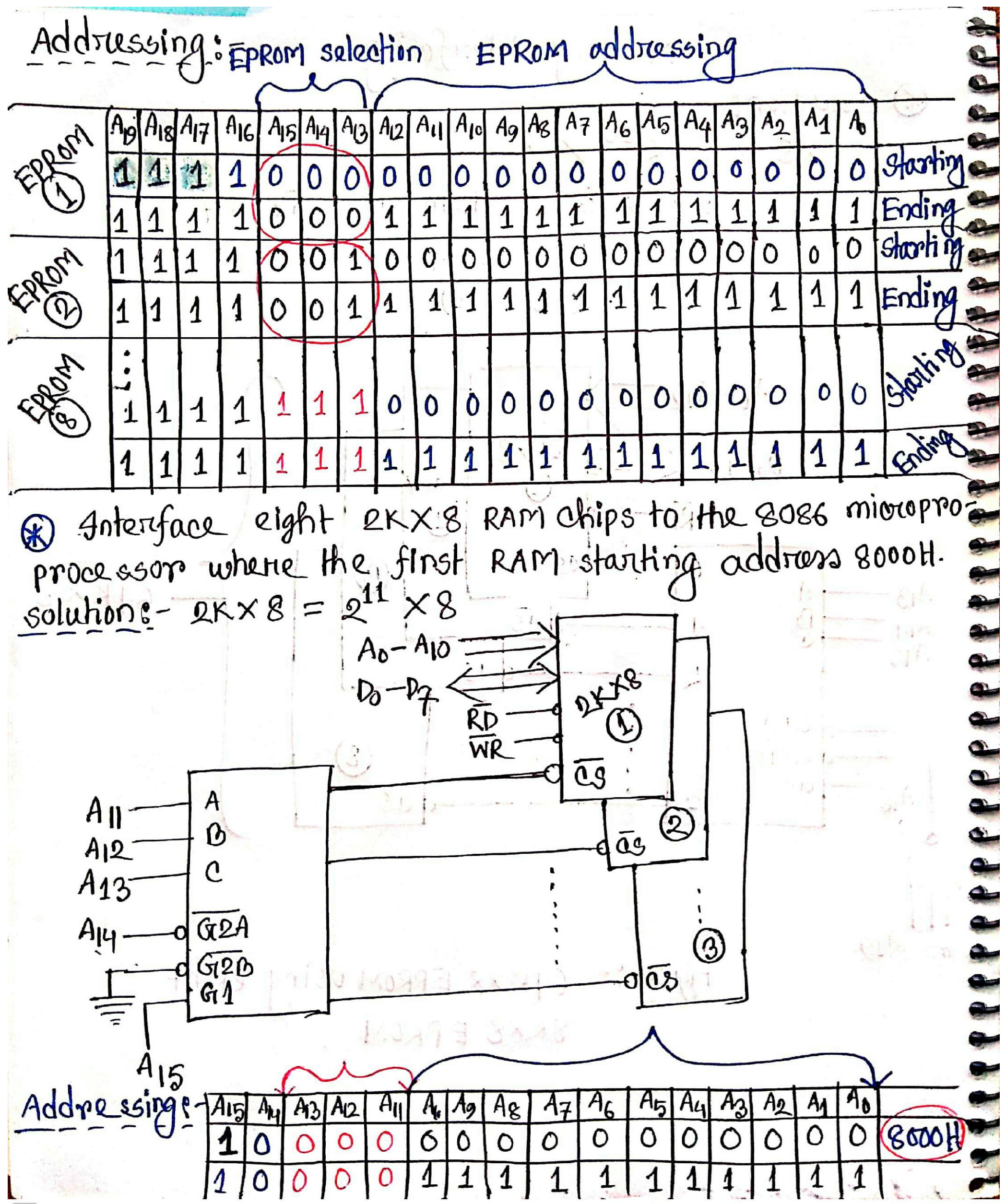
Figure:- 64K×8 EPROM using eight 8K×8 EPROM

Taven Trizedon MR

Scanned with CamScanner

Interfacing Memory with 8086 Microprocessor

- Interface eight 2k X 8 RAM chips to the 8086 microprocessor system where first RAM starting address will be 8000H.

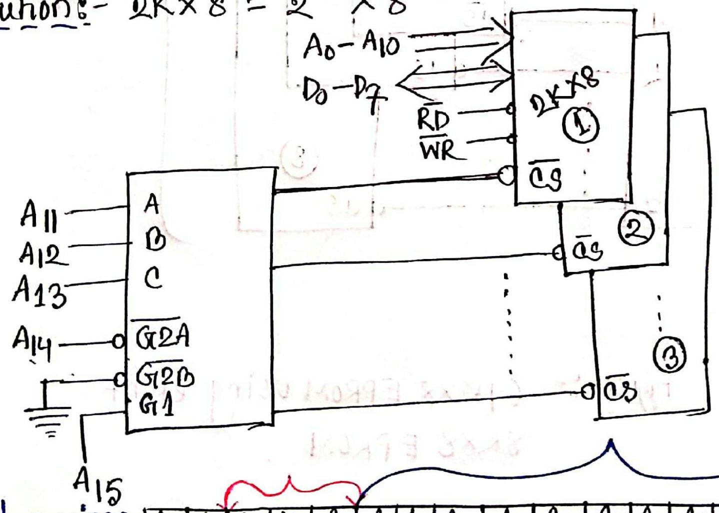


Addressing: EPROM selection EPROM addressing

	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	
EPROM (1)	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Starting
	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	Ending
EPROM (2)	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Starting
	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Ending
	⋮																				
EPROM (8)	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Starting
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Ending

(*) Interface eight 2K X 8 RAM chips to the 8086 microprocessor where the first RAM starting address 8000H.

solution:- $2K \times 8 = 2^{11} \times 8$



Addressing:-

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	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	

CS Scanned with CamScanner

Memory Banking in 8086

- The 8086 processor provides a 16 bits data bus.
- It is capable of transferring 16 bits in one cycle but each memory location is only of a byte (8 bits), therefore we need two cycles to access 16 bits (8 bit each) from two different memory locations.
- The solution to this problem is Memory Banking.
- Through Memory banking our goal is to access two consecutive memory locations in one cycle (transfer 16 bits).
- The memory chip is equally divided into two parts(banks).

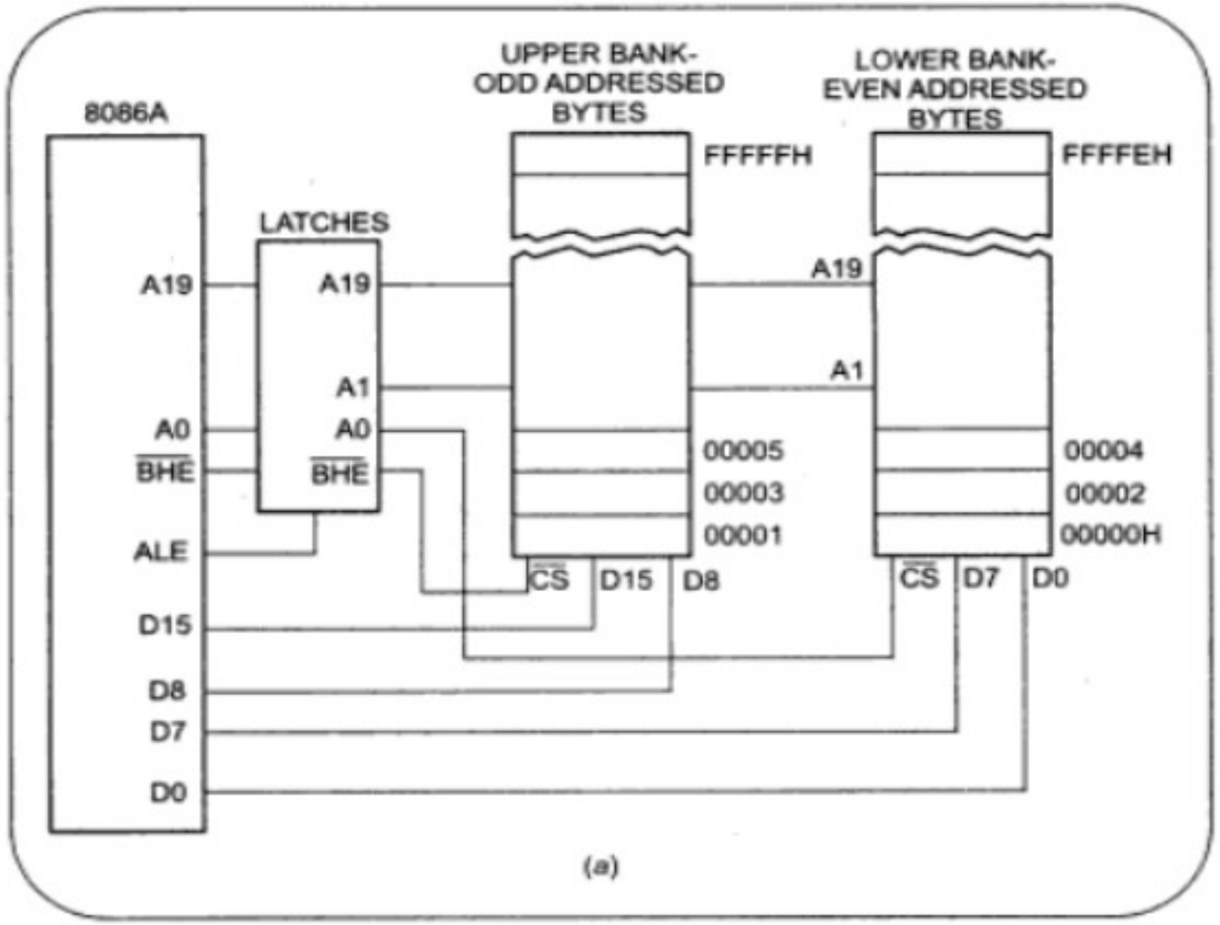
Memory Banking in 8086

- ❖ One of the banks contain even addresses called **Even bank** and the other contain odd addresses called **Odd bank**.
- ❖ Even bank always gives lower byte, so Even bank is also called **Lower bank** (LB).
- ❖ Odd bank is called a **Higher bank** (HB).
- ❖ Consider the instruction **MOV DS: WORD PTR [437AH], BX**. The word is actually written into two consecutive memory locations. If DS contains 0000H, the low byte of the word is written into **00437AH** and high byte is written into **00437BH**. To make it possible within one cycle the memory of 8086 is set up as two banks.

Memory Banking in 8086

Signals for read and write operation:

Address	Data type		A_0	Bus Cycles	Data Lines
Even	Byte	1	0	One	D_0-D_7
Even	Word	0	0	One	D_0-D15
Odd	Byte	0	1	One	D_8-D_{15}
Odd	Word	0	1	First	D_8-D_{15}
		1	0	Second	D_0-D_7



Memory Banking in 8086

- **Even Byte Addressing: MOV AH, DS: BYTE PTR [0000H].** If you read a byte or write a byte to 00000h, A_0 will be low and $\overline{BHE}$ will be high. Lower bank will be enabled, and high bank will be disabled. A byte will be transferred to or from the address location in the low bank D_0-$D_{7.}$ Then 8086 automatically transfer the byte of data from the lower data lines to AH.

- **Even word addressing: MOV AX, DS: WORD PTR [0000H].** To read a word from memory to AX both A0 and $\overline{BHE}$ will be low, so that both banks will be enabled. The low byte of the word will be transferred from address 00000H to the 8086 on D0-D7. The high byte of the word will be transferred from address 00001H to the 8086 on D8-D15.

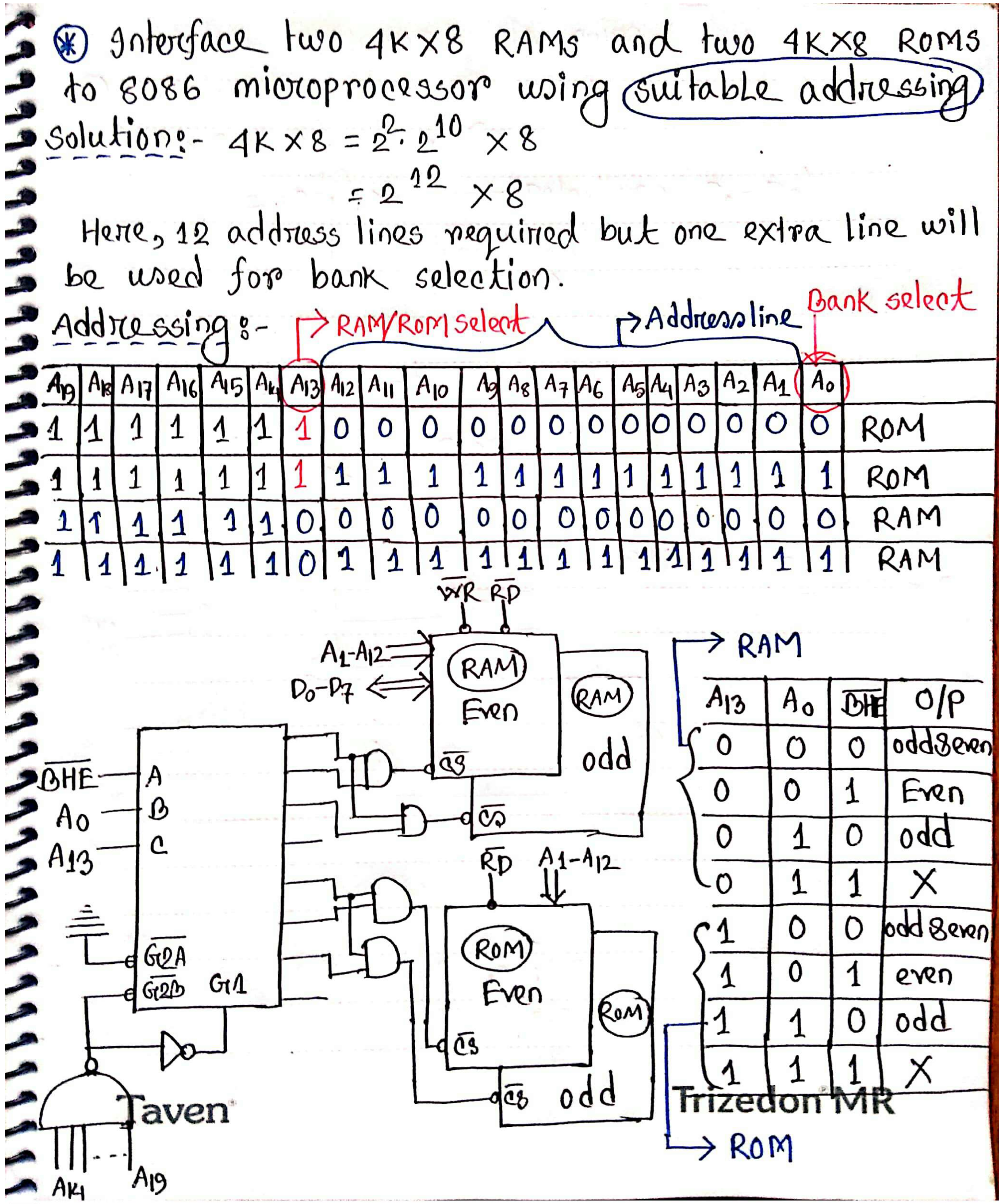
Memory Banking in 8086

- **Odd byte addressing: MOV AL, DS: BYTE PTR [0001H].** $\overline{BHE}$ will be low and A0 will be high. The byte will be transferred from memory address 00001H in the high bank to the 8086 on lines D8-D15. Then 8086 automatically transfer the byte of data from the lower data lines to AL.
- **Odd word addressing: MOV AX, DS: WORD PTR [0001H].** This instruction copies the low byte of a word from address 00001H to AL and high byte from address 00002H to AH. In this case 8086 requires two machine cycles. During the first machine cycle 8086 will output address 00001H, A0 high and $\overline{BHE}$ will be low. The byte from address 00001H will be read into the 8086 on lines D8-D15 and put in AL. During the second machine cycle the 8086 will send out address 00002H, $\overline{BHE}\ will\ be\ high$ and A0 will low. The byte from address 00002H will be read into the 8086 on lines D0-D7 and put in AH.

Memory Banking in 8086

- **Reason of using separate A0 and $\overline{BHE}$ signal:** If we want to perform any write operation with this instruction **MOV DS: BYTE PTR [0002H], AL.** The data from AL would be written to address 00002H for only enabling the lower bank by assigning A0 to low and $\overline{BHE}$ *to high*. If the upper bank also enabled then random data will be written from D8-D15 into address 00003H.

Memory Banking in 8086



✱ Interface two 4K×8 RAMs and two 4K×8 ROMs to 8086 microprocessor using suitable addressing

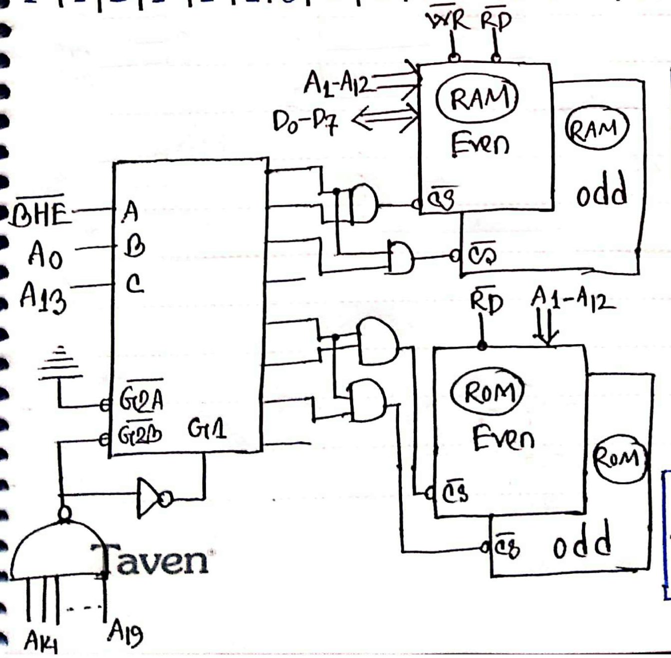
Solution:- $4K \times 8 = 2^2 \cdot 2^{10} \times 8$

$= 2^{12} \times 8$

Here, 12 address lines required but one extra line will be used for bank selection.

Addressing:- → RAM/ROM select → Address line Bank select

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	
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	ROM
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	ROM
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RAM
1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	RAM



→ RAM

A_{13}	A_0	$\overline{BHE}$	O/P
0	0	0	odd & even
0	0	1	Even
0	1	0	odd
0	1	1	X
1	0	0	odd & even
1	0	1	even
1	1	0	odd
1	1	1	X

→ ROM

Taven
Trizedon MR
Scanned with CamScanner