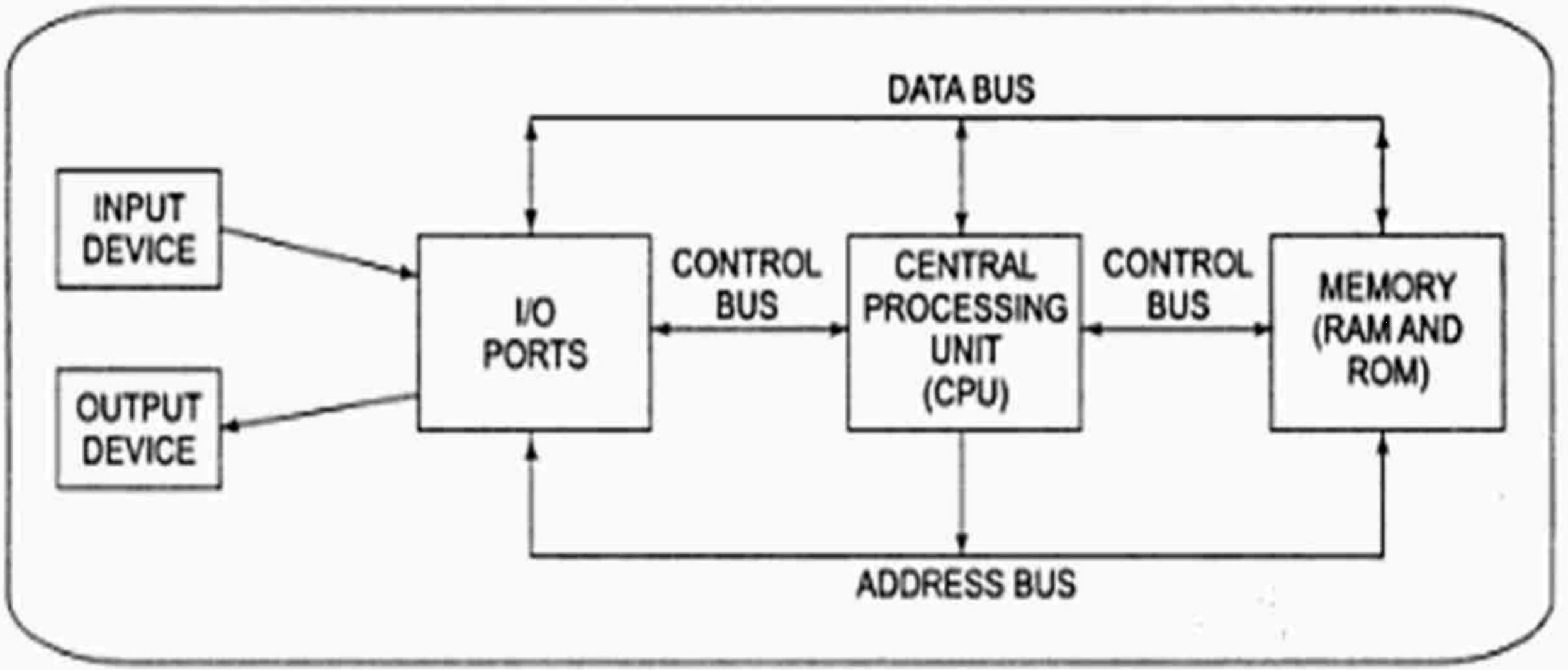
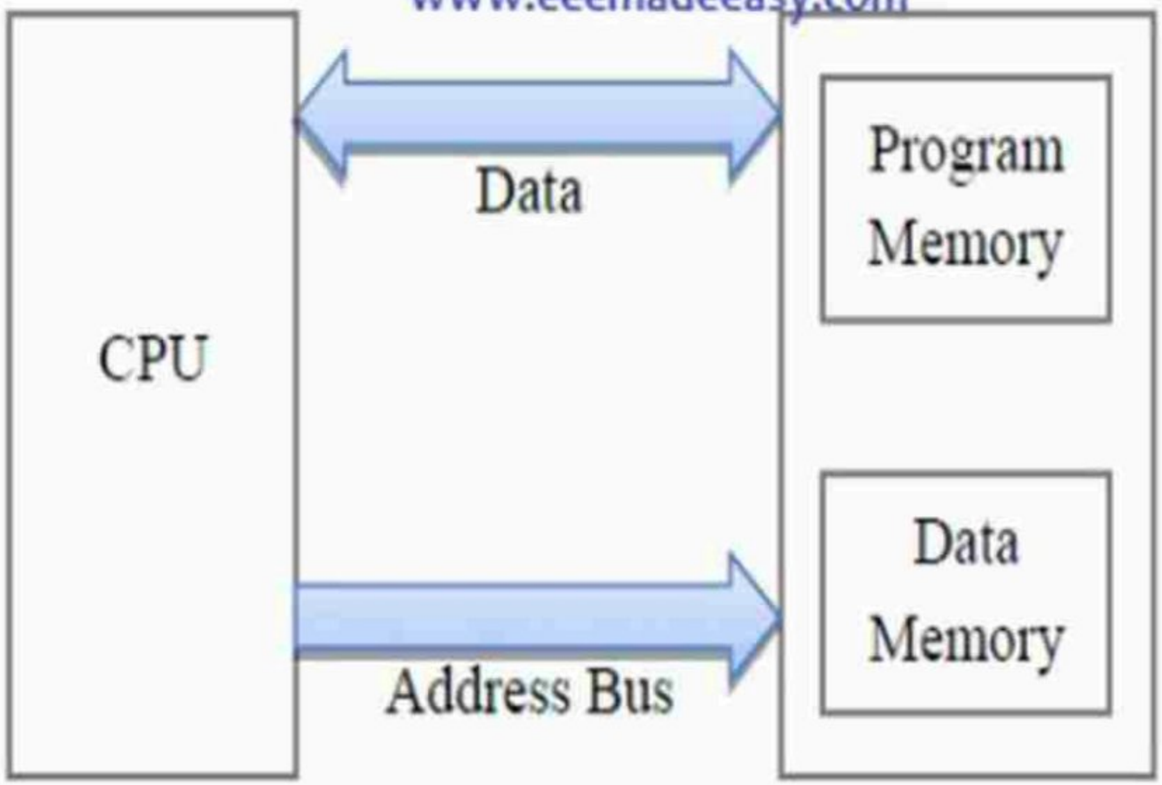
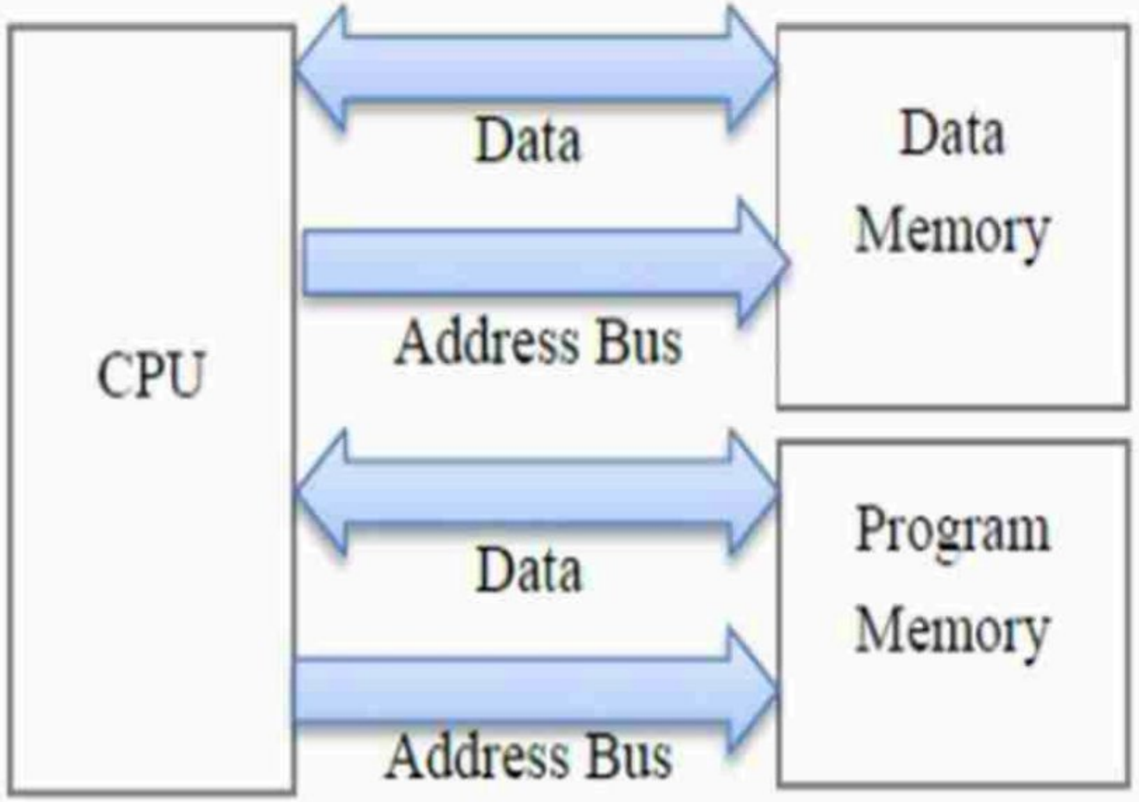


Von-Neumann (Princeton architecture)



Harvard architecture

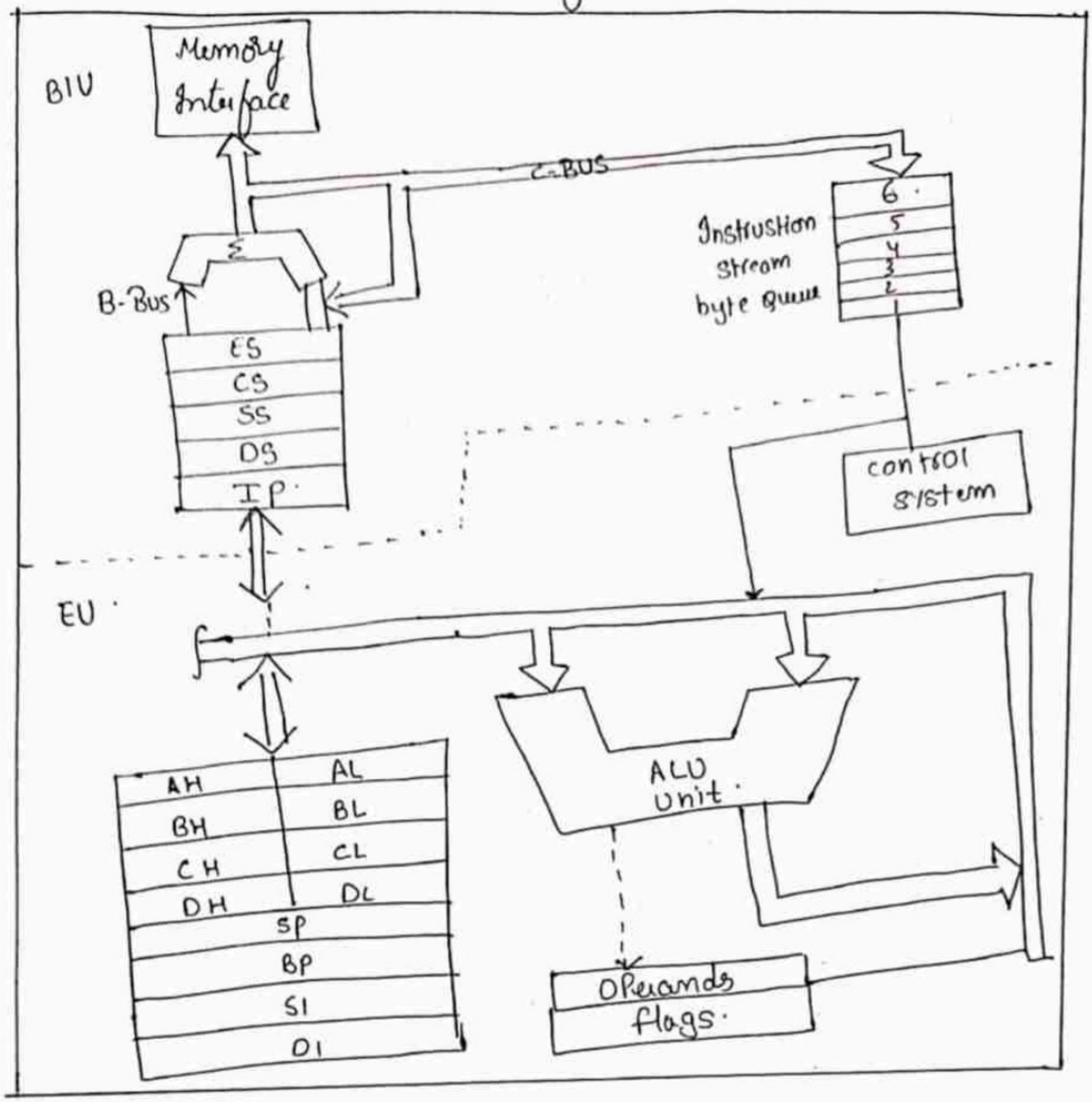


1. **Central Processing Unit (CPU)**:
   - The CPU is the brain of the computer and executes instructions stored in memory.
   - It consists of an Arithmetic Logic Unit (ALU) for mathematical and logical operations and a Control Unit (CU) for managing the execution of instructions.
2. **Memory**:
   - Memory is used to store data and instructions that the CPU needs to process.
   - There are two main types of memory:
     - **RAM (Random Access Memory)**: Provides fast, temporary storage for data and program code.
     - **ROM (Read-Only Memory)**: Contains permanent firmware or instructions that cannot be modified.
3. **Input/Output (I/O) Devices**:
   - These include peripherals like keyboards, mice, monitors, printers, and network adapters.
   - I/O devices allow users to input data and receive output from the computer.
4. **Storage Devices**:
   - Hard drives (HDDs), solid-state drives (SSDs), and optical drives are used for lon storage of data and programs.

RISC vs CISC Processors/Controllers

RISC	CISC
Instruction takes one or two cycles	Instruction takes multiple cycles
Only load/store instructions are used to access memory	In additions to load and store instructions, memory access is possible with other instructions also
Instructions executed by hardware	Instructions executed by the micro program
Fixed format instruction	Variable format instructions
Few addressing modes	Many addressing modes
Few instructions	Complex instruction set
Most of the have multiple register banks	Single register bank
Highly pipelined	Less pipelined
Complexity is in the compiler	Complexity in the microprogram

* Architecture of 8086 *



Explanation.

8086 CPU is divided into 2 Parts (Functional units).

① Bus Interface unit

② Execution unit

Dividing the work between these two units speeds up the processor.

① BIU :- Fetches Address, Instructions from memory
reads data, writes data to memory
handles all data transfers and address on bus for execution.

→ BIU has direct link with memory so memory can be accessed either by Segment Registers, Instruction Pointer or Instruction Queue for fetching up the Instructions

→ These Instructions are sent to control unit for execution
The control unit takes helps of registers, ALU for execution

→ Instruction Queue:
Contains set of Instructions to be executed
It has 6 instructions in advance which are fetched and stored
whenever a Instruction completes its execution the next Instruction is executed by control unit.

→ Segment Registers:
4 types with each 16 bit
CS — code segment
DS — Data segment
SS — stack segment
ES — extia segment

→ Instruction Pointer:
contains address of next instruction that is to be executed.

(2) Execution Unit (EU)

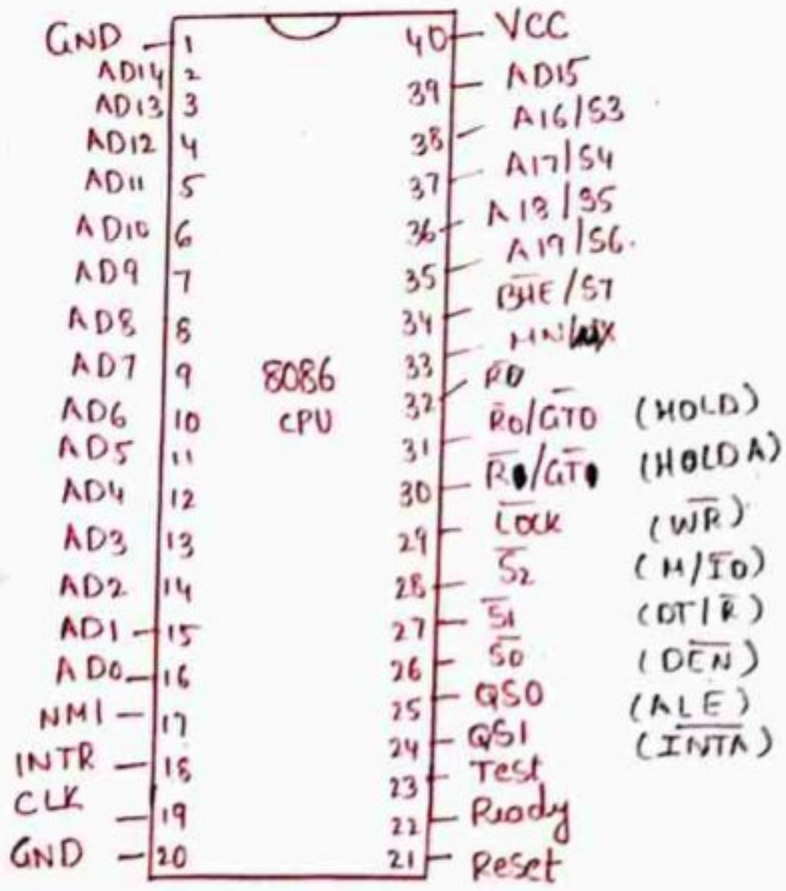
→ control unit: executes Instructions
: main component of any Processor.

→ ALU: Mathematical and logical Operations are Performed

→ General Purpose Registers, Pointers & Index Registers [write from Reg diagram]

→ Operands: used with Instructions

Pin Diagram of 8086 :-



Power Supply :- +5V DC Supply at Pin-40.

Ground :- 2 Pins (1 & 20)

Clock Signal :- Pin-19 for providing timing signals to processor for Operations (6-10 MHz)

Address/Data bus :- AD0 - AD15 . carries 16 bit address.
↓
AD0 - AD7 → carries low order byte data
AD8 - AD15 → Higher order byte data

Address / status bus :- (A16-A19)/(S3-S6)
carries 4 bit address.

S7/BHE :- Bus high enable - 34 Pin.
Transfer of data using data bus.

Read :- 32 Pin - Signal for read Operation.

Write :- 29 Pin - Signal for write Operation to memory or output or Input devices. depends on M/IO.

MN/MX :- Min / Max mode Pin-33
It Indicates what mode the 8086 is Operating
when = 1 - Min mode
0 - Max mode.

Ready :- Pin-22
acknowledgement signal from I/O devices that data is transferred.
Ready = 1 :- Device is ready to transfer data
= 0 :- Wait state.

Reset :- causes processor to immediately terminate Present activity and restart execution. (Pin-21)

INTR :- Pin-18
Interrupt request Signal.
INTR = 1 → Interrupt Present
= 0 → Interrupt absent

NMI : Pin 17
definitely should be served by Micro Processor

INTA :- Pin-24 (Interrupt Acknowledgement)
Signal given by microProcessor after acknowledgement given to Interrupt

ALE (Address latch enable) (Pin-25)
A positive pulse is generated indicates Valid address on address/ data lines

DEN (Data Enable): Pin-26.
acts as transreciever

DT/R (Data Transmit / Receive) Pin-27
Gives Direction of Data flow
DT/R = 1 → Data is transmitted
= 0 → Data is recieved.

M/IO : Pin-28
M/IO = 1 → Indicates IO Opeation
0 → Indicates memory Opeation

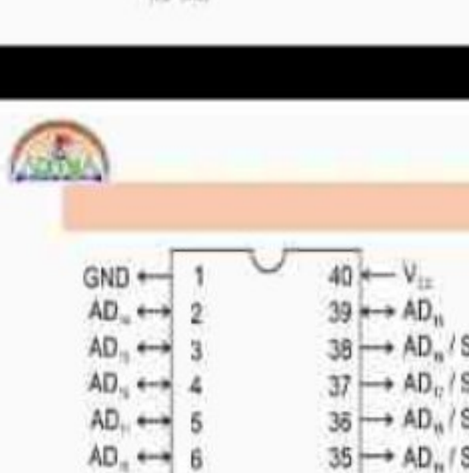
HLDA : Pin-30
Hold the Acknowledgement of HOLD signal.

HOLD : Pin-31
Indicates external devices are requesting to access the addres / data buses.

➤Minimum or maximum mode operations are decided by the pin MN/ MX(Active low).

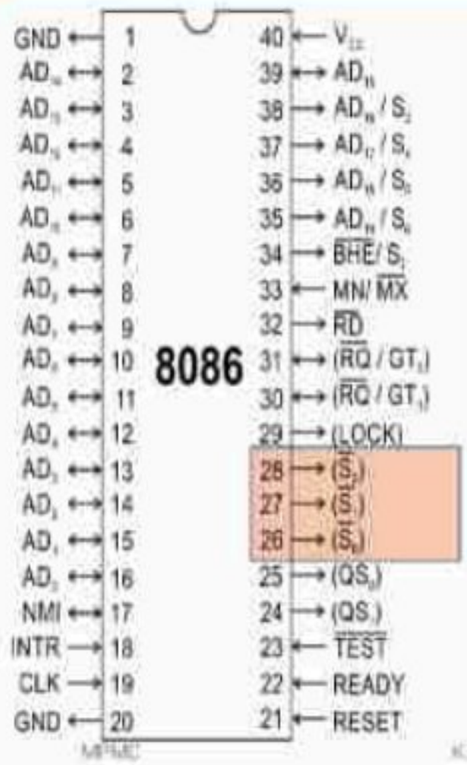
➤When this pin is high 8086 operates in minimum mode otherwise it operates in Maximum mode

MINIMUM MODE OF 8086

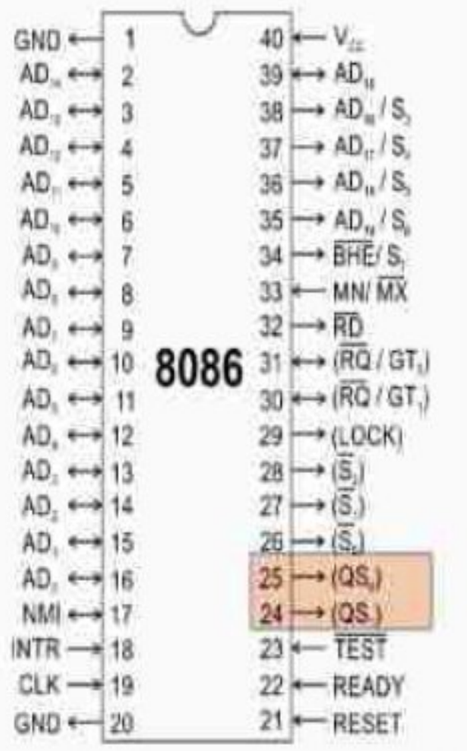


DT/$\overline{R}$	(Data Transmit/ Receive) Output signal from the processor to control the direction of data flow through the data transceivers
$\overline{DEN}$	(Data Enable) Output signal from the processor used as out put enable for the transceivers
ALE	(Address Latch Enable) Used to de-multiplex the address and data lines using external latches
M/$\overline{IO}$	Used to differentiate memory access and I/O access. For memory reference instructions, it is high. For IN and OUT instructions, it is low.
$\overline{WR}$	Write control signal; asserted low Whenever processor writes data to memory or I/O port
$\overline{INTA}$	(Interrupt Acknowledge) When the interrupt request is accepted by the processor, the output is low on this line.
HOLD	Input signal to the processor form the bus masters as a request to grant the control of the bus. Usually used by the DMA controller to get the control of the bus.
HLDA	(Hold Acknowledge) Acknowledge signal by the processor to the bus master requesting the control of the bus through HOLD. The acknowledge is asserted high, when the processor accepts HOLD.

MAXIMUM MODE OF 8086



$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Indication
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive



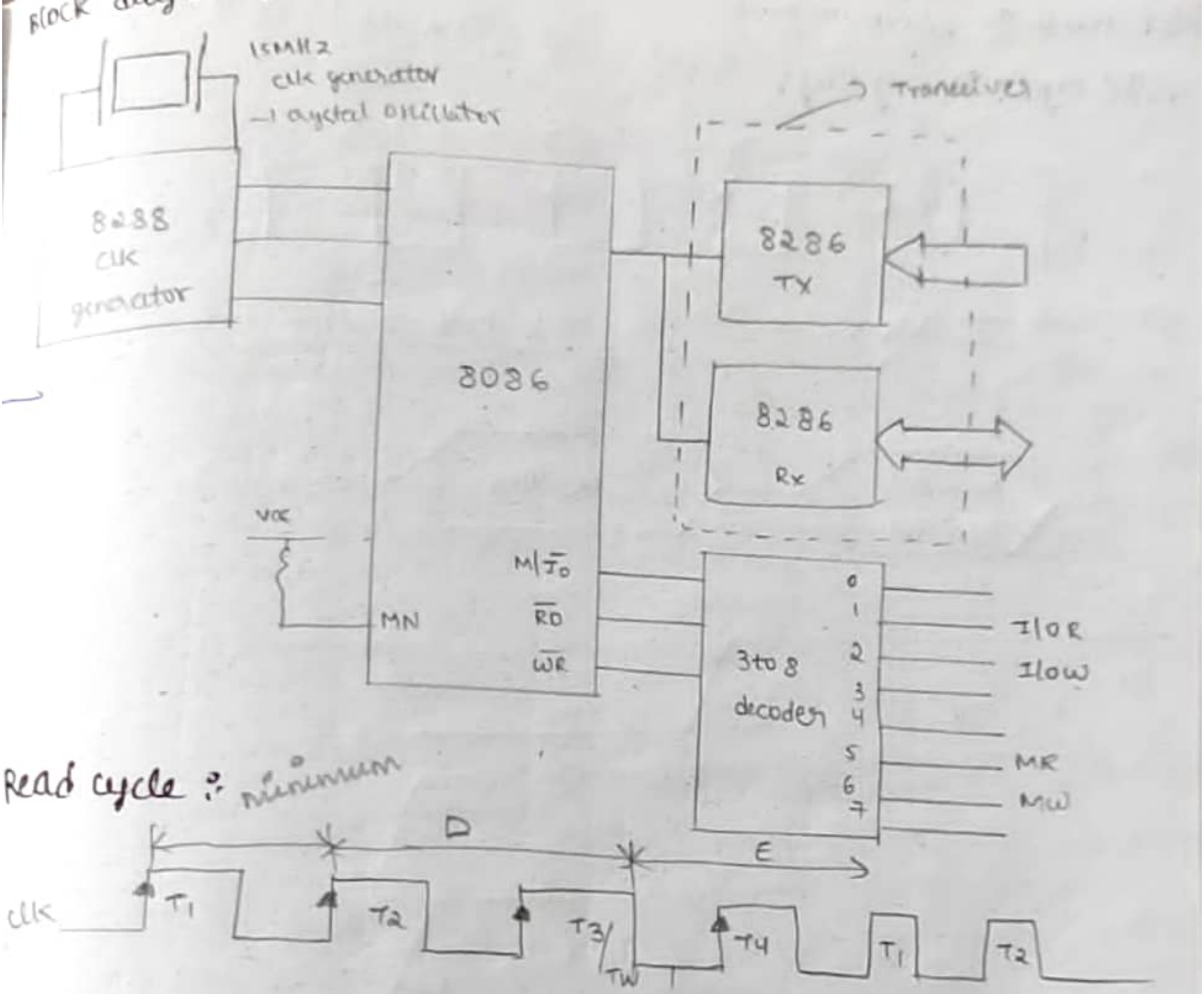
QS_1, QS_0

➤(Queue Status) The processor provides the status of queue in these lines.
➤The queue status can be used by external device to track the internal status of the queue in 8086.
➤The output on QS_0 and QS_1 can be interpreted as shown in the table.

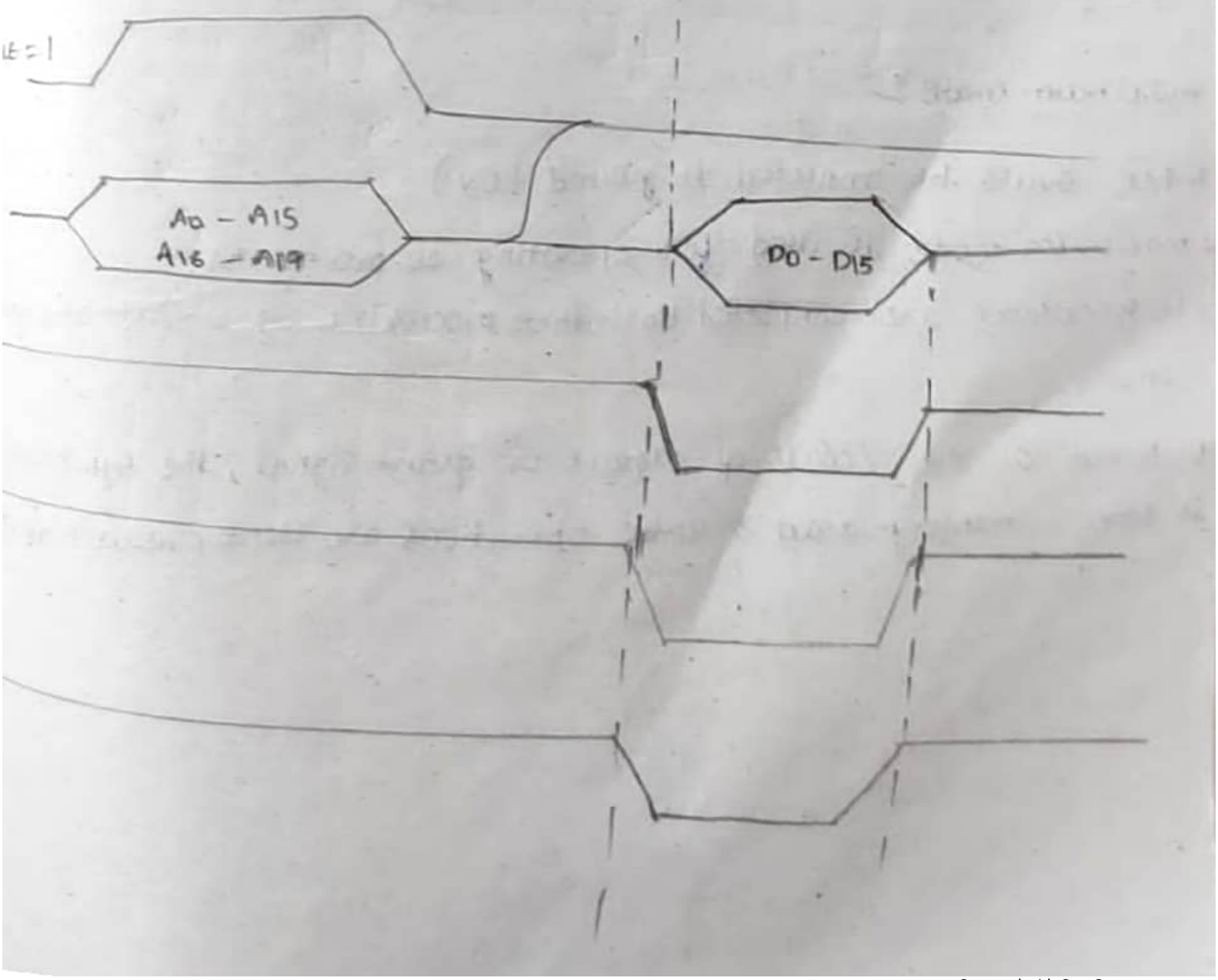
QS_1	QS_0	Indication
0	0	No Operation
0	1	First Byte of opcode from the queue
1	0	Empty Queue
1	1	Subsequent Byte from the Queue

8086 QUEUE OPERATION

Block diagram



Read cycle : Minimum



MIN mode:

→ It is used for fixed point representation. Based on memory and i/o

MIN mode performs read and write operation. It is used for single processor

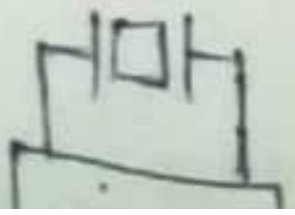
→ MN should be connected to Vcc

→

M/IO →	0	I/O	0	1	I/OW	1	RD
RD →	0		1	0	MR	1	DEN
WR →	1		0	1		0	DT/R

R → reading / Receiving

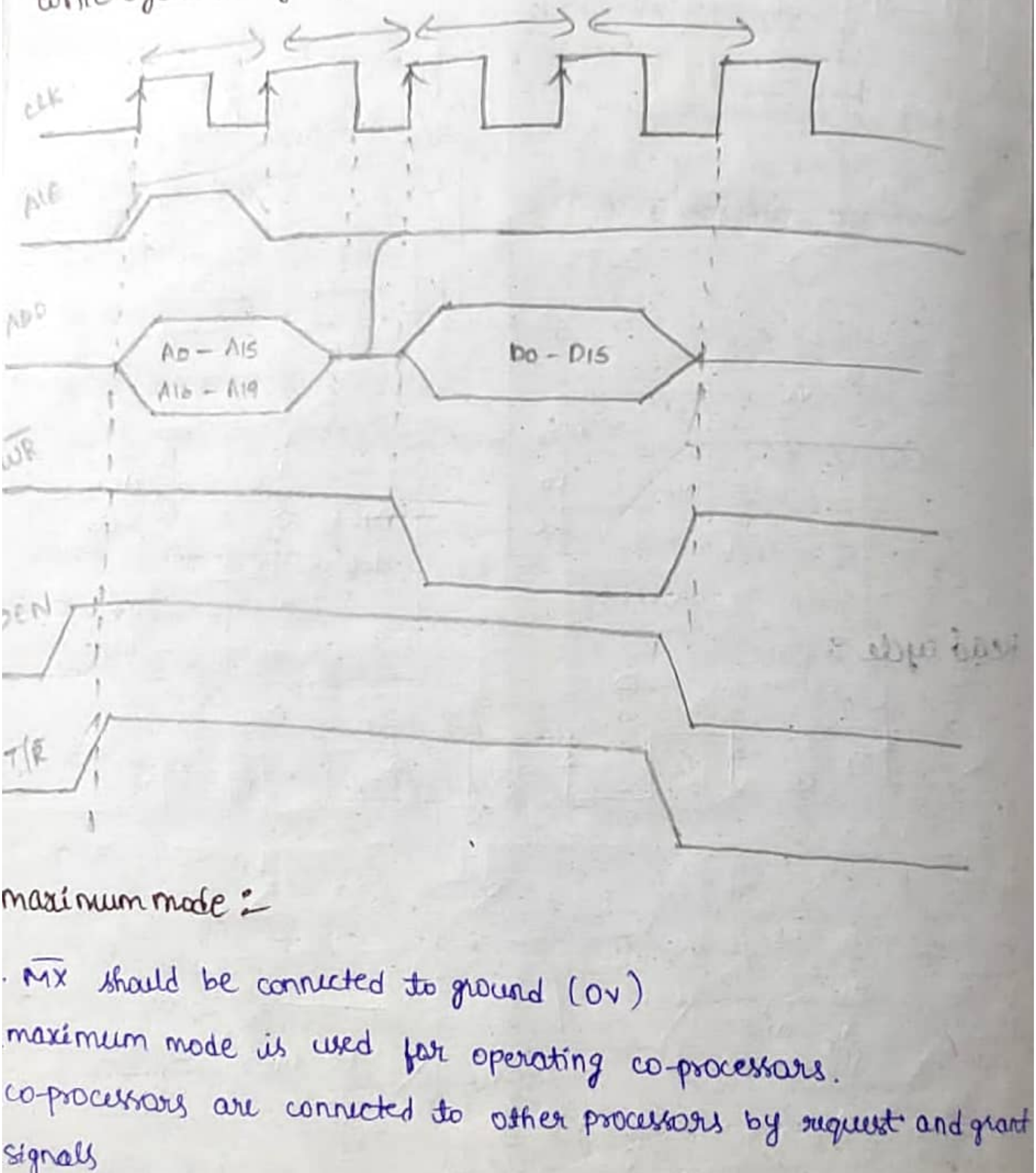
W → writing.



Transceiver

Max mode :- minimum ...

write cycle Timing diag: Direct memory Access (DMA)



maximum mode :-

- $\overline{MX}$ should be connected to ground (0v)
- maximum mode is used for operating co-processors.
- co-processors are connected to other processors by request and grant signals
- Based on the status of Request or grant signal, the synchronizati-on is done memory read & write operations are done (as well as I/O)

clk generator

CLK
Reset
Ready

ALE
BHE
AD0-AD15
A16-A19

8 bit latch

A0-A19

RTO / GTO
RT / GT

8086

Transceiver

D0-D15

INTA
NMI
QS0
QS1
MX

S2
S1
S0

Bus controller

MEMRD
MEMWR
IORD
IOWR

memory
CPU
DMA
I/O

gnd

max mode read cycle

T2 T3 T4

CLK

ALE

BHE / S7

A0 - A15
A16 - A19

A16/S3
A19/S6

D0 - D15

DEN

DT/R

IO/M

Scanned with CamScanner

SS – stack segment
ES – extra segment

→ Instruction Pointer:
contains address of next instruction that is to be executed.

② Execution unit (EU)

→ control unit: executes Instructions
: main component of any Processor.

→ ALU: Mathematical and logical Operations are Performed

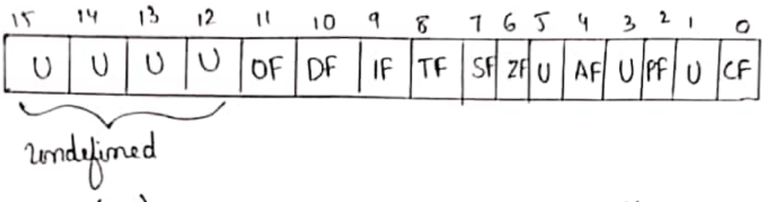
→ General Purpose Registers, Pointers & Index Registers [write from Reg Organ]

→ Operands used with Instructions

Flag Register

A flag is a flip flop that indicates some condition after execution of instruction or controls certain Operations of EU.

* → 16 bit Registers
* Total 9 flag Registers → 6 Status flags
→ 3 Control flags



① Carry flag (CF) → set to '1' when there is an overflow
'0' when no over flow

② Parity flag (PF) → set '1' when there are even number of ones
'0' when odd no of ones in result.

③ Auxilary flag (AF) → set '1' when there is over flow for 4 bits low registers

④ Zero flag (ZF) → set to '1' when result is zero.
'0' when non zero result.

⑤ sign flag (SF) → set to '1' when result is negative
'0' for non negative

⑥ Trap flag (TF) → used for on chip debugging

① Interrupt enable flag (IF) → set to '1' when interrupt came from external devices.

② Direction flag (DF) → Process of data 0 → forward.
1 → back word.

③ overflow flag (OF) → Set to 1 – signed overflow.
0 – no overflow.

without using String manipulation

```
ASSUME CS: CODE, DS: DATA, ES: EXTRA.

DATA SEGMENT

DB 01, 02, 03, 04, 05, 06, 07, 08, 09, 10

COUNT EQU 10

DATA ENDS

EXTRA SEGMENT

EXTRA ENDS.

CODE SEGMENT

ORG 1000H.

START: MOV AX, 2000H
       MOV DS, AX
       MOV AX, 5000H
       MOV ES, AX
       MOV SI, 0
       MOV DI, 0
       MOV CX, COUNT
LOOP1: MOV AL, [SI]
       MOV [DI], AL.
       INC SI
       INC DI
       LOOP LOOP1
       MOV AH, 4CH.
       INT 21H
CODE ENDS
END START.
```

4

with using string manipulation

```
ASSUME CS: CODE, DS: DATA, ES: EXTRA

DATA SEGMENT
DB 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

COUNT EQU 10

DATA ENDS

EXTRA SEGMENT

EXTRA ENDS.

CODE SEGMENT
ORG 1000H
START: MOV AX, 2000h
       MOV DS, AX
       MOV AX, 5000h
       MOV ES, AX
       MOV SI, 0
       MOV DI, 0
       MOV CX, COUNT
       REPE MOVSB.
       MOV AH, 4CH
       INT 21H
CODE ENDS
END START.
```

Adressing Modes of 8086

Several ways of locating data or operand in the memory is called an adressing modes

Flow of Instruction Execution

1. Sequential Instruction → It is further divided into 8 types
2. Control Transfer Instruction → It is of 4 types

Sequential control flow Instructions

1. Immediate Addressing Mode
2. Direct Addressing Mode
3. Register Addressing Mode
4. Register Indirect Addressing Mode
5. Indexed Addressing Mode
6. Register Relative Addressing Mode
7. Based Indexed Addressing Mode
8. Relative Based Indexed Addressing Mode

1) Immediate Addressing Mode.

With in the itself the direct data is available is called Immediate Addressing Mode

Example

MOV AH, 1234H

Note
→ The source and destination must be of same size
→ All registers except segment registers can be used in this mode

2) Direct Addressing Mode

In direct Addressing Mode, a 16 bit offset address [memory] an Io adress is directly specified in the instruction.

Example.

1) MOV AX,[5000] Physical address = DS* 10H + 5000H etc
2) MOV [1234], AL
3) MOV CL, [3456]
4) MOV [8005], AX
5) IN 80H

3) Register Addressing Mode.

The data is available In Register
→ In the Register Addressing Mode, the data is stored in registers
→ All registers, except Ip may be used.

Ex:

MOV Ax, Bx
MOV CX, AX
MOV DS, AX
ADD AL, BL

4) Register Indirect Adressing Mode.

In this Adressing Mode, the offset adress of data is in either Bx (or) SI (or) DI register whereas the default adressing register either DS (or)

Ex:

```
MOV   AX, [BX]
MOV   CX, [DI]
MOV   AX, [SI]
```

5) Indexed Addresing Mode.

In this Adressing Mode, the offset adress are stored in either in SI or DI and the default segment register is DS.

```
MOV   AX, [SI]
MOV   CX, [DI]
MOV   [SI], BX
```

physical adress = DS* 10H + SI e.t.c.

6) Register ~~Ind~~ Relative Adressing Mode.

In this Adressing Mode, the data is available at an affective adress which is formed by 8 bit or 16 bit displacement with the content of any one of the regis [offset ~~regist~~ adress] BX, BP, SI, DI

The default segment register is DS (or) ES

Example

```
MOV   AX, 50H [BX]
MOV   CX, 1234H [SI]
MOV   80H [DI], AX
```

physical adress = DS* 10H + 50 + BX e.t.c.

7) Base Indexed Adressing Mode.

In this Adressing Mode, the displacement adress [16bit] is stored in any one of the base registers BX (or) BP and the offset adress is stored in SI (or) DI and the default segment registers are DS (or) ES

Ex

```
MOV   AX, [BX][SI]
MOV   CX, [BP][DI]
MOV   [BX][SI], AX
```

physical Address = DS* 10H + BX + SI e.t.c.

8) Relative Based Indexed Adressing Mode.

Ex

```
MOV   AX, 50H[BX][SI]
MOV   CX, 1234H[BP][DI]
```

Adressing Modes for Control Transfer Instructions

1. ~~Inst~~ Intrasegment Direct Addressing Mode
2. Intrasegment Indirect Addressing Mode
3. Intrasegment Direct Addressing Mode
4. Intersegment Indirect Addressing Mode.

→ LOOPNZ / LOOPNE (Loop if not zero flag) :- ZF=0

Assembler's directives of 8086

While converting the assembly language program into machine code (or) executable code, the assembler performs certain tasks such as the allocation of memory, assigning the logical names to the segment. for completing these type of task assembler required assembler directives

which helps the assembler to correctly understand the assembly language program.

1. **ASSOME** :- It is used to identify the assembly that which segment is used.

2. **DB** :- (Define Byte) It allocates the space for the array
→ It reserves the byte (8) bites of memory location in the available memory

```
N1      DB      01H,02H, 03H, 04H
LIST    DB      "ADITYA"
RESULT  DB      06H
```

3. DW :- (Define word) word = 2byte, words = 6bytes

```
N1      DW      1234H, 4567H, 045CH.
DATA    DW, 05H
```

4. DQ :- (Define quad word) quard = 4word

It allocate 4 word spaces

5. DT (Define Ten bytes)
It allocated Ten bytes space

6. ENDS :- (End of segment)

7. END (End of program)

8. ENDP (End of procedure) It indicates the end of procedure (subprograms).

9. EQU (Equal)
EQU will assign value to the variable.

10. ORG. (Origin)
It indicates the starting of the memory allotment for the particular segment.

SEGMENT :-

This assembler directive marks the starting of a logical segment

Tabular form for RAM

Starting Address

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	0	0	1	0	0	1	1	0	0	0	0	1	0	0	1	0	1	0
C				9				8				4				A			

Address lines / RAM add.

1	1	0	1	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	
D				F				7				A						3	

Add lines / RAM Add.

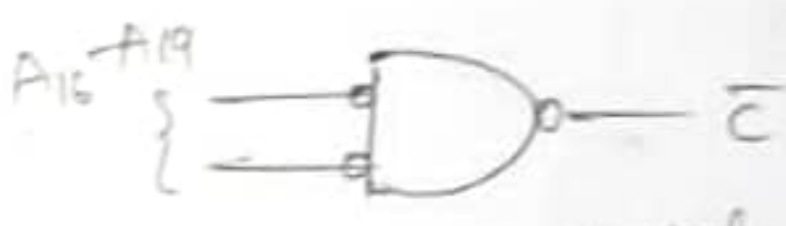
Step 6: Address Range

EPROM - EC3A7H to FDC59H

RAM - C984AH to DF7A3H

Step 7: There will be 4 chip selects $\overline{CS_1}$, $\overline{CS_2}$, $\overline{CS_3}$, $\overline{CS_4}$

EPROM will be selected



RAM will be selected.

Even Address Memory Bank → 0-7

Odd Address Memory Bank → 8-15

Data bus 16-bit

EPROM - Read ($\overline{OE}$)

RAM → Read & Write

⇒ To select odd PROM, $\overline{BHE}$ should be given along with 'C' to inverter NAND Gate

(Vo-17)

Unit-3 3question

Step 1: There are 2 ROM's & 2 RAM's

Step 2: Total Memory in ROM = 32k + 32k = 64k

Total " " RAM = 32k + 32k = 64k

Step 3: No. of Address lines

ROM $\Rightarrow N = 64\,K$

$2^n = 2^6 \times 2^{10}$

$2^n = 2^{16}$

$\Rightarrow \boxed{n = 16}$

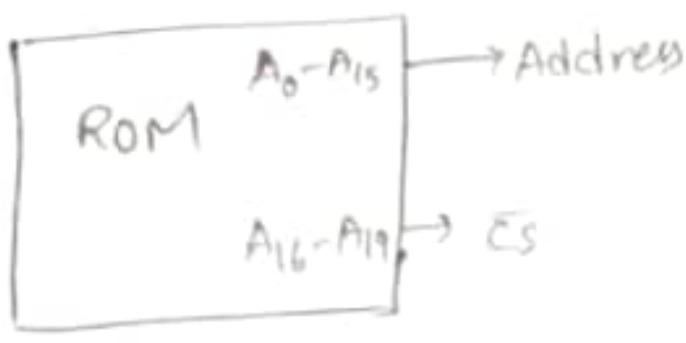
RAM $\Rightarrow N = 64K$

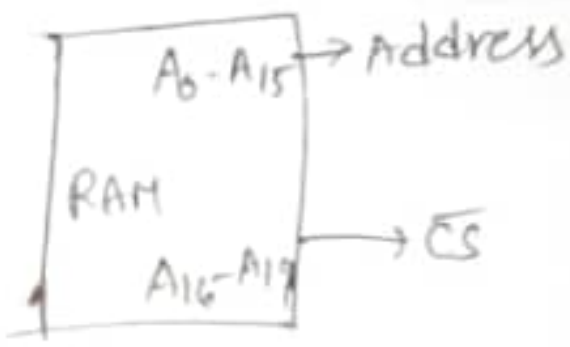
$2^n = 2^6 \times 2^{10}$

$2^n = 2^{16}$

$\Rightarrow \boxed{n = 16}$

Step 4:





Step 5: Tabular form for ROM

Starting Address

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	0	1	1	0	0	0	0	1	1	1	0	1	0	0	1	1	1
E				C				3				A				8		7	

Address lines / ROM Add.

Ending address / effective Address

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	0	1	1	1	1	0	0	0	1	0	1	1	0	0	1
F				D				C				5				9			

CS

Address lines / ROM Add.

Tabular form for RAM

Starting Address

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	0	0	1	0	0	1	1	0	0	0	0	1	0	0	1	0	1	0
C				9				8				4				A			

Address lines / RAM add.

1	1	0	1	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	
D				F				7				A						3	

Add lines / RAM Add.

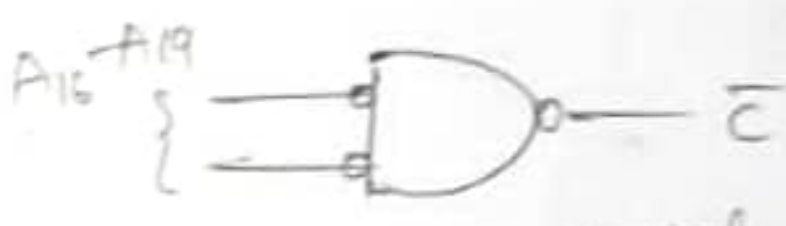
Step 6: Address Range

EPROM - EC3A7H to FDC59H

RAM - C984AH to DF7A3H

Step 7: There will be 4 chip selects $\overline{CS_1}$, $\overline{CS_2}$, $\overline{CS_3}$, $\overline{CS_4}$

EPROM will be selected



RAM will be selected.

Even Address Memory Bank → 0-7

Odd Address Memory Bank → 8-15

Data bus 16-bit

EPROM - Read ($\overline{OE}$)

RAM → Read & Write

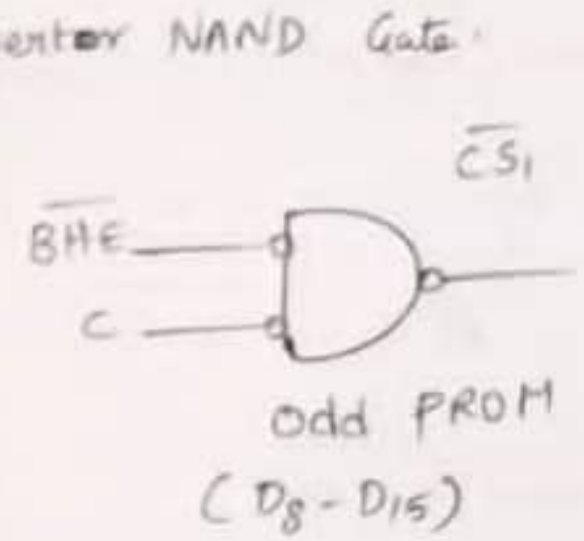
⇒ To select odd PROM, $\overline{BHE}$ should be given along with 'C' to inverter NAND Gate

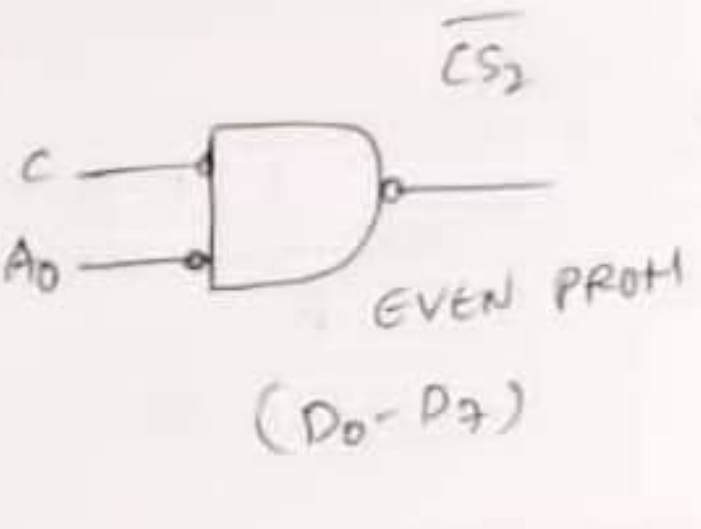
(Vo-17)

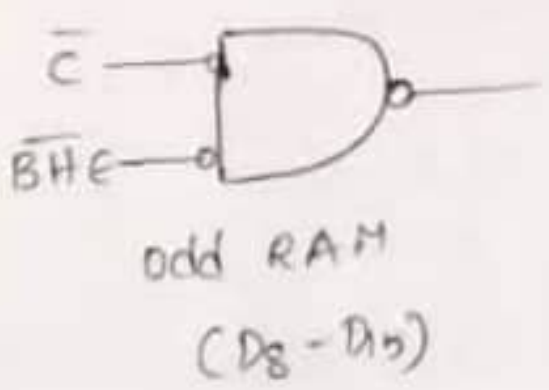
→ To select EVEN PROM A_0 should be given along with $\bar{C}$ to Inverter NAND Gate

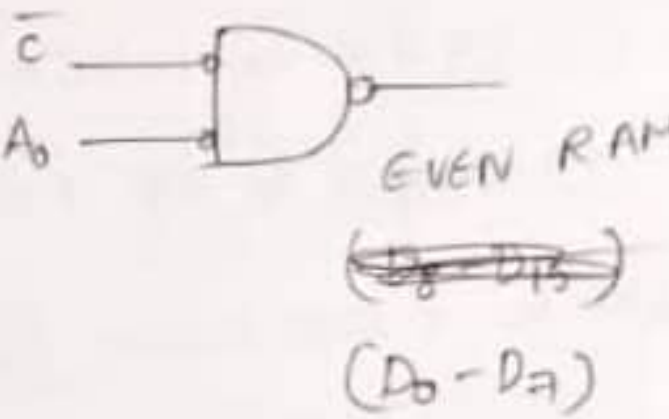
→ To select odd RAM $\overline{BHE}$ should be given along with $\bar{C}$ to inverter NAND gate

→ To select even RAM A_0 along with $\bar{C}$ should be given to Inverter NAND Gate.

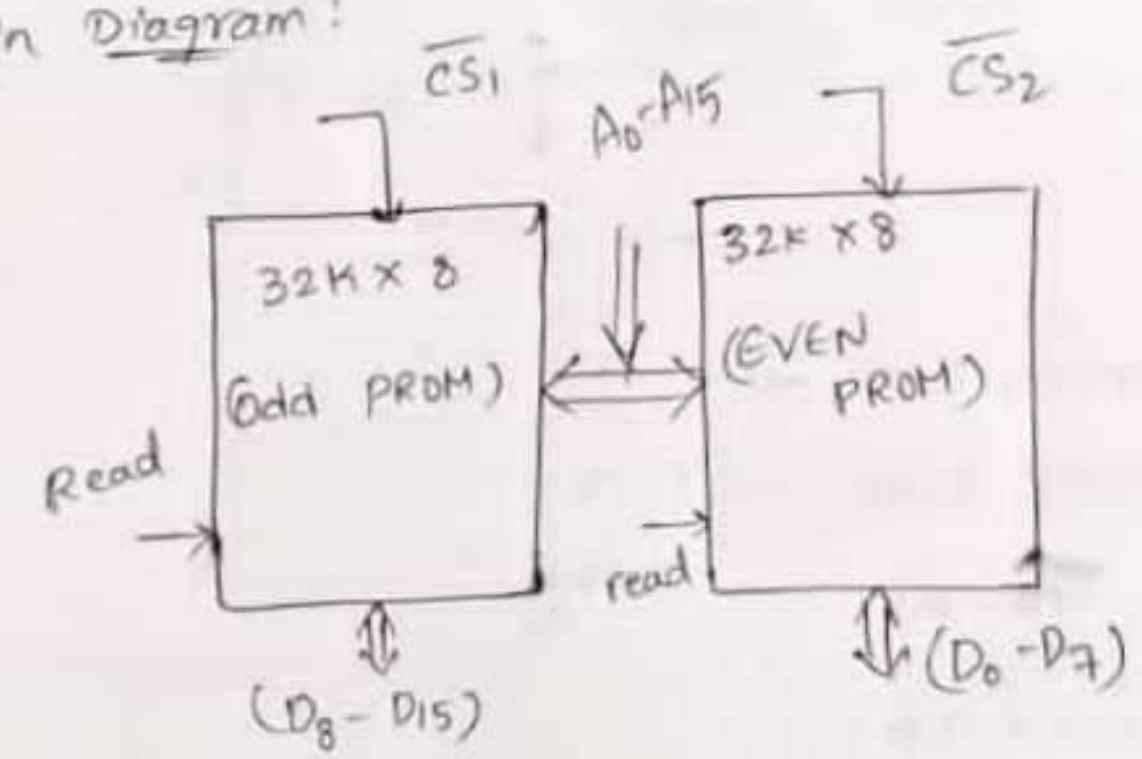


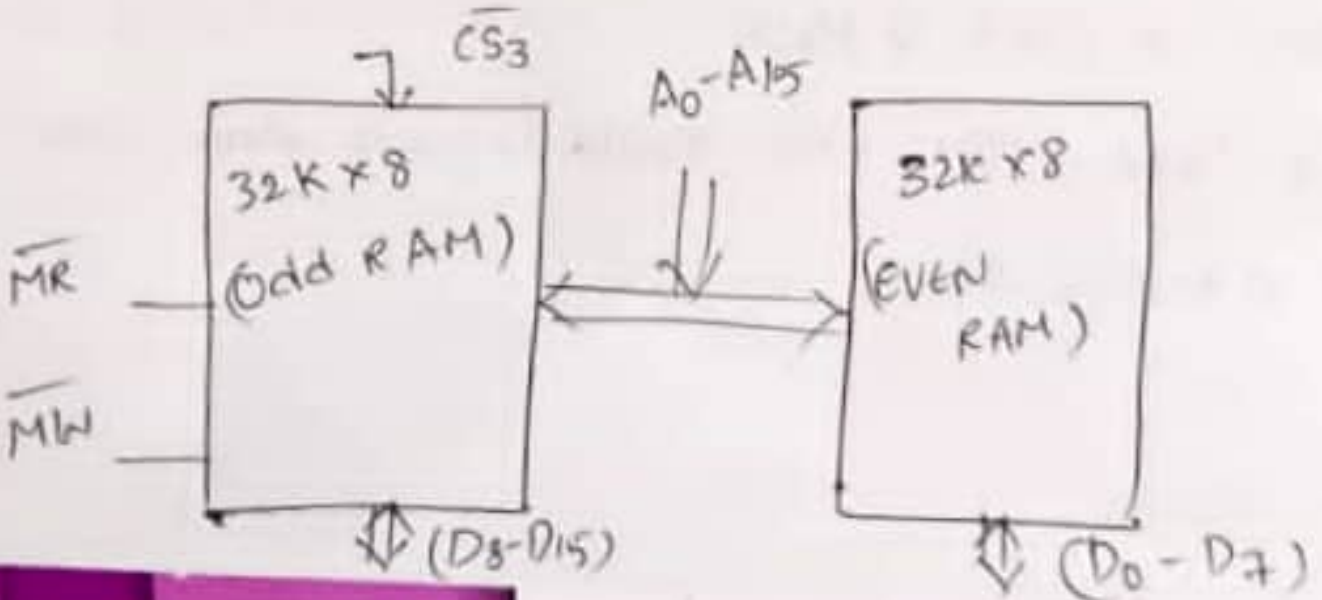




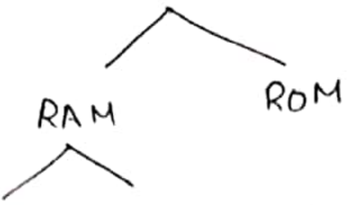


Main Diagram:





Semiconductor Memories:



→ Interfacing with 8086 / 8088

→ memories are stored in arrays (4K×8)
4096 - each location has 8 bits

→ once control is given - all bits are selected by databus 16bit and address by address bus (12 lines)

→ In general N memory locations we require n address line.

$$2^n = N$$
$$n = \log_2 N$$

If P locations are only used then higher order (N-P) are used for decoding all

→ Memory address depends on hardware ckt decoder ($\overline{CS}$)

Procedure

→ Arrange available memory chips 16 bit databus
upper 8-bit bank - odd address bank
lower 8-bit bank - even address bank

→ connect address lines of memory to micro Processor, $\overline{RD}$ & $\overline{WR}$ also.

→ Problem

2 ROM chips < 16K×8, 16K×8 } 32K

2 RAM chips < 32K×8, 32K×8 } 64K

⇒ Problem

2 ROM chips < 16K x 8, 16K x 8 } 32K

2 RAM chips < 32K x 8, 32K x 8 } 64K

① Determine NO of address lines by formula RAM & ROM

ROM = $2^n = N$

$2^n = 32KB$

$n = 15$ address lines

RAM = $2^n = 64KB$

$n = 16$ address lines

② Prepare the address table with address location.

② Sort the address lines for address and remaining for chip select ($\overline{CS}$) signals.

ROM → A0 - A14 — address

ROM → A15 - A19 — $\overline{CS}$ signal

RAM → A0 - A15 — address

RAM → A16 — A19 — $\overline{CS}$ signal.

③ Tabular form

EPROM F8000H

A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	....	A0	
1	1	1	1	1	0	0	0	0	0	0	0		0	start
														FFFFF
1	1	1	1	1	1	1	1	1	1	1	1		1	end

RAM

A11 A10 A9 A8 ...

RAM

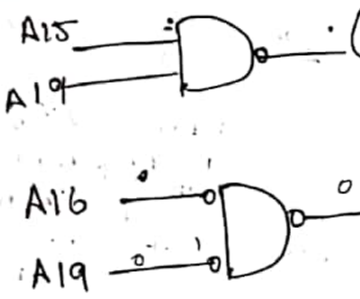
A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8 ... A0	
0	0	0	0	0	0	0	0	0	0	.	. .	00000H Start
0	0	0	0	1	1	1	1	1	1	1	- - -	0FFFFH end.

④ write the Address Range

EPROM — F8000H — FFFFF H

RAM — 00000H — 0FFFF H

⑤ use NAND & Inverter NAND for i/p of CSELECT signals as addres is not full no. need decoder.



along with these $\overline{BHE}$ & A0 use.

combination of ① $\overline{BHE}$ & C ② A0 & C ③ $\overline{BHE}$ & C̄ ④ A0 & C̄

① Draw odd address memory bank, even address mem bank (0-7).

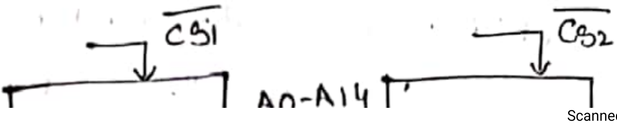
(8-15)

for EPROM & RAM :— blocks & data lines — 16

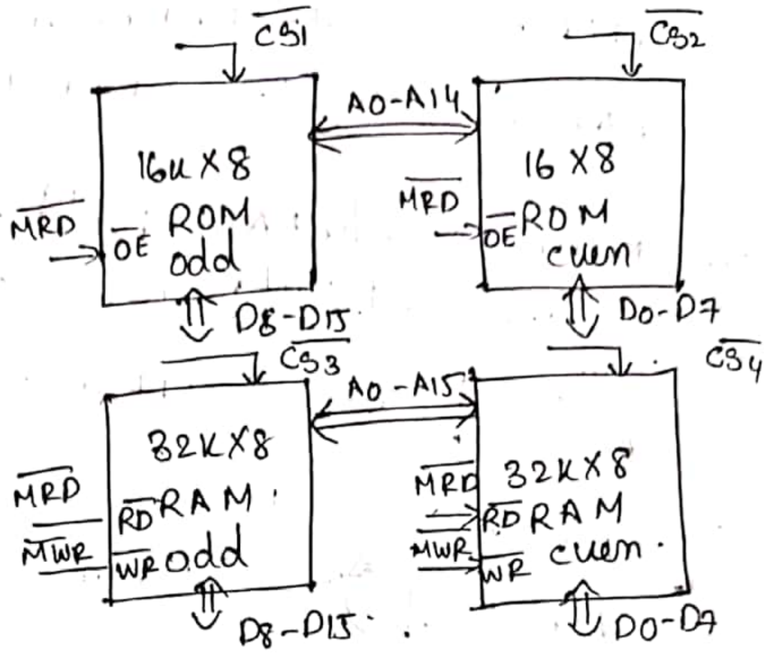
⑤ write the Operations of memory of EPROM & RAM

⑥ Draw the chipselect signals of 4 blocks memory.

Interfacing Diagram



Interfacing Diagram



(c) 0 1, BHE 0 1 → 0 $\overline{CS1}$ (odd ROM) selected, high D8 selected

(c) 0 1, A0 1 → 0 $\overline{CS2}$ (even ROM) selected, low D0 selected

(c) 0 1, A0 1 → 0 $\overline{CS4}$ (even RAM) selected - low D0 selected

(c) 0 1, BHE 0 1 → 0 $\overline{CS3}$ (Odd RAM) selected - high data bus selected

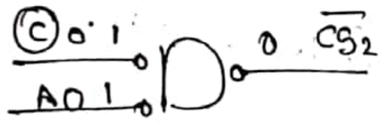
(1) → calculate (c) values from before gates from table.

(2) → $\overline{BHE}$ → odd ROM, odd RAM will be selected

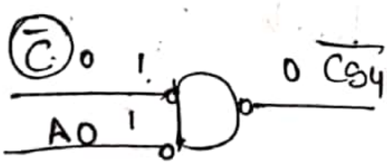
(3) → A0 → even ROM, even RAM will be selected.

→ when, A_2 $\overline{BHE}$ - A0 are 1 0 0 even and odd address in ROM - selected

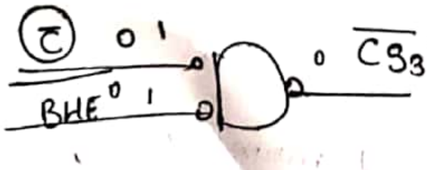
A_2 $\overline{BHE}$ - A0 are 0, 0 0 → even and



(even ROM) selected | low D
selects



(even RAM) Selected - low D
selects



(Odd RAM) Selected - high data bus select

Scanned by CamSca

① → calculate © Values from before gates from table.

② ○ $\overline{BHE}$ → odd ROM, odd RAM will be selected

③ A0 → even ROM, even RAM will be selected.

→ when, A_2 $\overline{BHE}$ - A0 are 10 0 even and odd address in ROM - selected

A_2 $\overline{BHE}$ - A0 are 0, 0 0 → even and odd address in RAM - selected

→ Finally Draw the Diagram