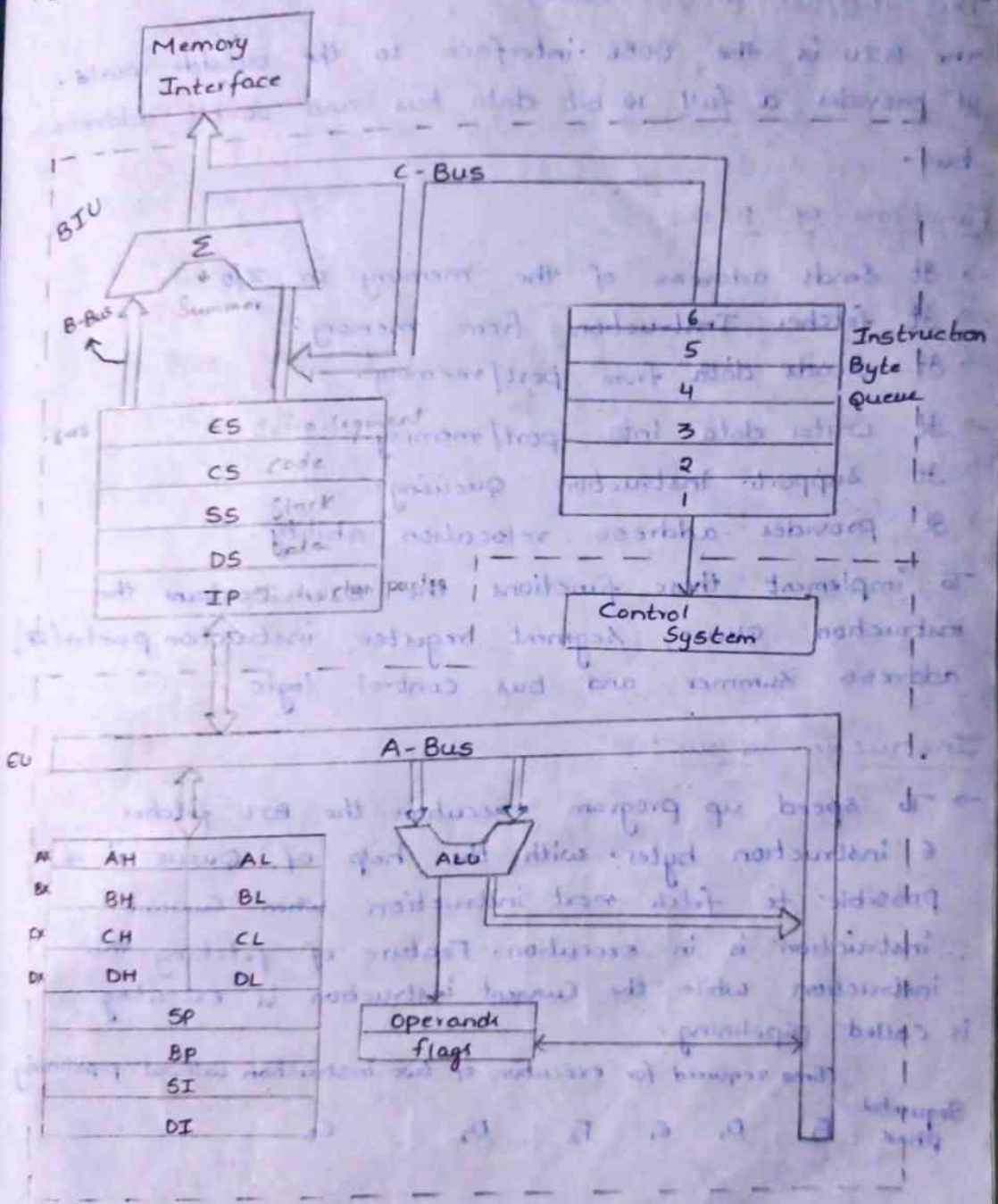
# III unit:

# 8086 Microprocessor

## Architecture of 8086 µp




1978 invented by intel

16-bit µp [$D_0$ - $D_{15}$]

20-bit [address bus] $2^{20}$ memory

14 - registers [length of each is 16]

Clock frequency - 5MHz

Support pipelining ⇒ [fetch, decode, execute ... at a time]

Multiprogramming

Architecture internally divided into 2 parts Separate Functional parts. These are (i) Bus interface unit & (ii) Execution unit. These two functional units can work simultaneously to increase system speed

Bus interface unit :- (BIU)

The BIU is the 8086 interface to the outside world. It provides a full 16-bit data bus and 20-bit address bus.

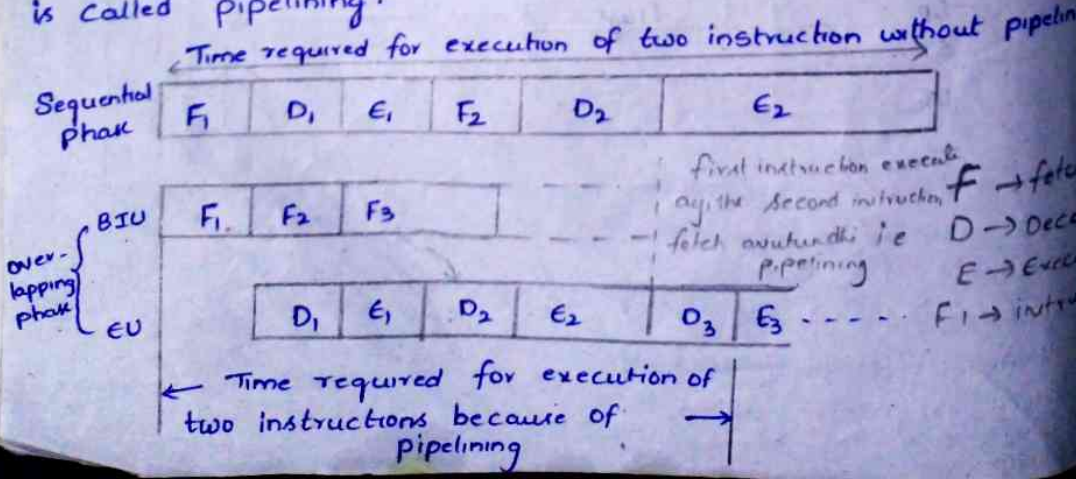
Functions of BIU :-

→ It sends address of the memory or I/O
→ It fetches Instructions from memory.
→ It reads data from port/memory.
→ It writes data into port/memory.
→ It supports instruction queueing.
→ It provides address relocation ability.

To implement these functions the BIU contains the instruction queue, segment register, instruction pointer, address summer and bus control logic.

Instruction queue :-

→ To speed up program execution the BIU fetches 6 instruction bytes. with the help of Queue it is possible to fetch next instruction when Current instruction is in execution. Feature of fetching the instruction while the Current instruction is executing is called pipelining.



Execution unit :- (EU)

The EU of 8086 tells the BIU from where to fetch instruction or data, decodes and executes instructions to perform these functions execution unit Contains ALU, flag register, general-purpose registers, pointers and index registers.

ALU (Arithmetic & Logic unit)

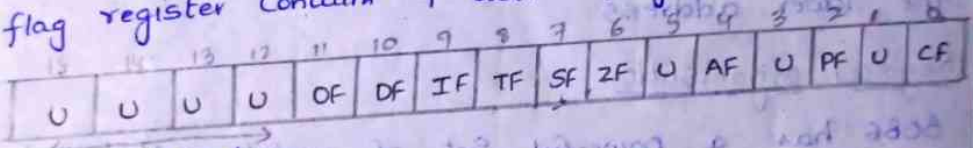
The ALU is 16-bit it performs addition, Subtraction, multiplication, division, Inc, dec, X-OR, AND, OR, NOT Compliment etc.

Control System :-

Control Circuitry directs Circuitary directs and Controls the internal operation

Flag register

The flag register Contains 9 active flags




Carry flag :- It is set if there is a carry for addition

parity flag :- It is set if result of operation contains even number of 1's otherwise it is reset. [error checking]

Auxiliary carry flag :- It is set if there is Carry from lower nibble (4-bit).

Zero flag :- It is set if the result is zero otherwise reset.

Sigen flag :-

It is set if the result is negative ($D_7$ or $D_{15}$) otherwise it is reset.

Overflow flag :- (OF)

It is set if the result is out of range

Trap flag (TF) :-

One way to debug a program is to run the program one instruction at a time and see the Content of

used registers and memory variables after execution of every instruction. This process is called "Single stepping" through the program. Trap flag is used for single stepping. If it is set a trap is executed after execution of each instruction.

Interrupt flag (IF):-

→ It is set a certain type of interrupt can be recognised by the 8086

Direction flag (DF) :-

It is used with string address. If DF=0 the string is processored from it begining with the first element having the lowest address otherwise the string is processored from high address to the lower address

Register organisation :-

8086 has a powerful set of registers. It includes general purpose, special purpose, pointers and index registers

General purpose registers :-

| | 15 ... 8 | 7 ... 0 |
|---|---|---|
| AX | AH | AL |
| BX | BH | BL |
| CX | CH | CL |
| DX | DH | DL |

8086 has 4 16-bit registers labelled as AX, BX, CX, DX Each 16-bit general purpose register can be split into 2-8bit registers. The letters L & H specify the lower & higher bytes of a particular register and X specify the complete 16-bit registers.

→ These are either used for holding data, variables and intermediate results temporarily.

→ The register AX is used as 16-bit accumulator and it perform arithmetic and logical operations.

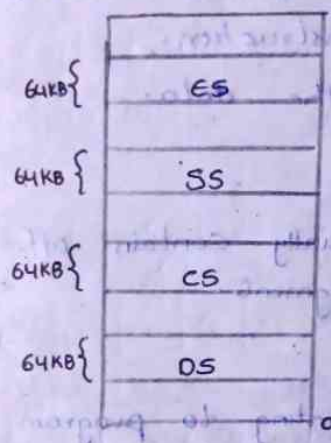
→ BX is used as off-set storage for generating physical addresses incase of certain addressing modes.

→ CX is used as default Counter incase of string & loop instructions.

→ DX can be used as a port number in I/o operations

Segment registers

→ IMB memory is divided into segment with a maximum size of 64KB. 8086 allows 4 Segment registers at a time




4- Segment registers are code segment (CS), data segment (DS), stack segment (SS), and extra segment (ES).

Code-Segment :-

→ It is 16-bit register containing the address of 64KB Segment with processor instruction.

→ The processor uses code segment for all accesses to instructions referred by IP register.

→ It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

Data-Segment :-

→ It is 16-bit register containing the address of 64KB Segment with program data.

→ AX, BX, CX, DX and index registers SI, DI data is located in data segment.

Stack Segment :-

→ It is 16-bit register containing address of 64KB Segment with program stack.

→ Stack pointer (SP) data and base pointer (BP) data located in the stack segment.

→ It is used for addressing stack segment of memory

Extra Segment

→ It is 16-bit register containing address of 64KB with program data. By default the processor assume that the DI register references the ES segment in string manipulation instruction.

→ ES Segment Contain the data.

pointers & Index registers :-

The pointers IP, BP, SP usually contain off-set within code, data and stack Segments.

Stack pont :-

It is a 16-bit address pointing to program stack in Stack Segment.

Base pointer :-

It is 16-bit register. It is used for off-set address

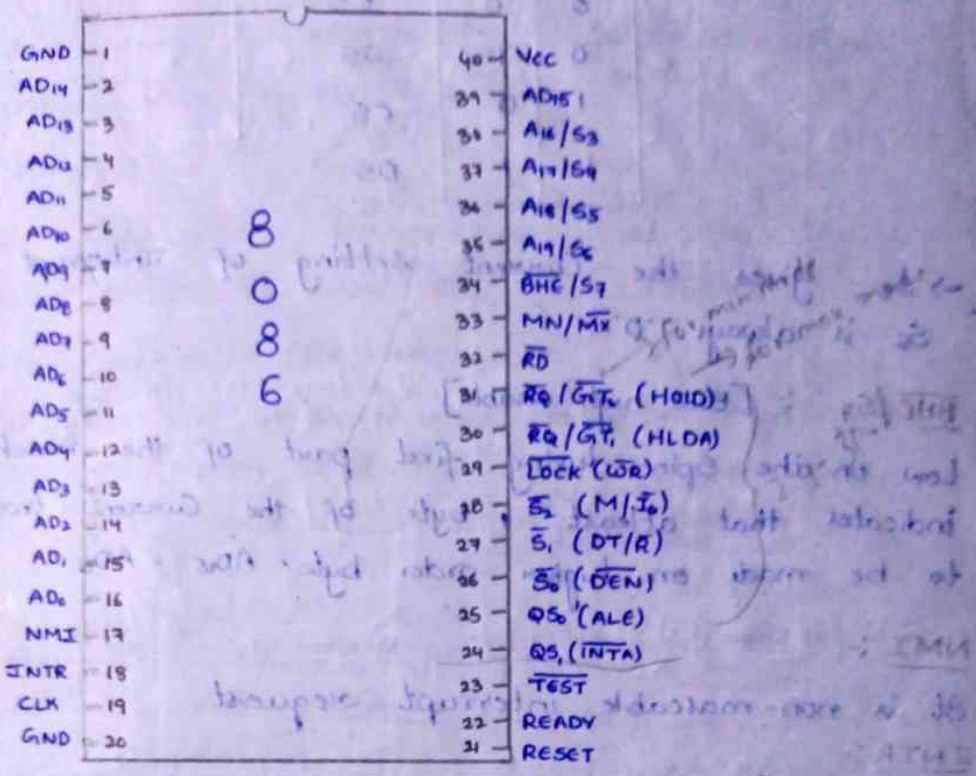
Source Index :- (SI)

SI used for Indexed, based indexed and register indirect addressing as well as Source data addresses in String manipulation instructions.

Destination Index (DI) :-

It is a 16-bit register. DI is used for Indexed, based-indexed and register indirect addressing as well as destination data address in string manipulation instructions.

pin diagram of 8086 :-




8086 Signals can be categorised into 3-groups

1) Signals having common functions in both minimum and maximum mode.
2) Signals having special functions for minimum mode
3) Signals having special functions for maximum mode.

Signals having common function in both mode

AD0 - AD15 :- Act as address bus during 1st part of machine cycle and data bus for remaining part of the machine cycle

A19/S6 - A16/S3 :- During the first part of the machine cycle these are used to output upper four bits of address. During remaining part of machine cycle these are used o/p status which indicates type of operation to be performed in that cycle. S5 & S4 indicate the segment register used as follows

| $S_4$ | $S_3$ | Register |
|---|---|---|
| 0 | 0 | ES |
| 0 | 1 | SS |
| 1 | 0 | CS |
| 1 | 1 | DS |

→ $S_5$ gives the current setting of Interrupt flag and $S_6$ is always '0'.

$\overline{BHE}/S_7$ : [Bus high enable]
Low on the Spin during first part of the machine cycle indicates that atleast 1 byte of the current transfer is to be made on higher order byte. $AD_{15} - AD_8$.

NMI :-
It is non-mascable interrupt request

INTR :-
It is level trigger mascable interrupt request.

CLK :- Clock
8086 requires clock signal from some external crystal control req generator to synchronize internal operation

RESET :-
It clears PSW, instruction pointer, Data segment, Extra segment, stack segment and the instruction queue. It then sets CS to FFFF H

READY :-
If this signal is low the 8086 enters into wait state.

$\overline{TEST}$ :
This signal is only used by the wait instruction. 8086 enters into wait state until a low signal on the $\overline{TEST}$ pin.

$\overline{RD}$ :- $\overline{RD}$ is a low whenever the 8086 is reading the data from memory or an I/o device

MN/$\overline{MX}$ :- The 8086 can be configured in either minimum mode or maximum mode using this pin.

MN = 0 then Max mode perform
MN = 1 then min mode perform

Signal definitions (24 to 31) for min mode

INTA :- this indicates recognisation of an interrupt request.

ALE :- This signal is provided by 8086 to demultiplex the $AD_0 - AD_{15}$ into $A_0 - A_{15}$ & $D_0 - D_{15}$

$\overline{DEN}$ (Data enable) :-
this signal informs the transreceiver that the CPU is ready to send or receive the data.

DT/$\overline{R}$ :- (Data transmitter / Receiver)
Signal is used to control data flow direction high on this pin indicates that 8086 is transmitting the data and low indicates that 8086 is receiving the data.

M/$\overline{IO}$ :-
If it is high indicates memory data transfer. If it is low indicate IO data transfer.

$\overline{WR}$ :-
$\overline{WR}$ is low whenever the 8086 is writing data into memory or IO device.

HOLD, HLDA :-
A high on HOLD pin indicates that another master is requesting to take over the system bus. On receiving the HOLD signal the processor output HLDA signal high as an acknowledgement. At the same time processor tri-states the system bus. A low on HOLD gives the system bus control back to the processor. processor then outputs low signal on HLDA.

## Signal definition for max mode

$QS_1$, $QS_0$ :- These two output signals reflects the status of instruction queue

| $QS_1$ | $QS_0$ | Status |
|---|---|---|
| 0 | 0 | NOP |
| 0 | 1 | first byte of an op-code |
| 1 | 0 | Queue is empty |
| 1 | 1 | Subsequent byte of an op-code |

$\overline{S_2}, \overline{S_1}, \overline{S_0}$ :-

These three status signal indicate the type of transfer to be take place during the current bus cycle.

| $\overline{S_2}$ | $\overline{S_1}$ | $\overline{S_0}$ | Machine Cycle |
|---|---|---|---|
| 0 | 0 | 0 | Int ack (acknowledge) |
| 0 | 0 | 1 | I/o read |
| 0 | 1 | 0 | I/o write |
| 0 | 1 | 1 | halt |
| 1 | 0 | 0 | Inst ifetch |
| 1 | 0 | 1 | memory read |
| 1 | 1 | 0 | memory write |
| 1 | 1 | 1 | inactive (doesn't perform any operation) |

$\overline{LOCK}$ :-

This signals indicates that the bus is not to be used by another processor.

$\overline{RQ}/\overline{GT_1}$ & $\overline{RQ}/\overline{GT_0}$ :-

The maximum mode, HOLD & HLDA pins are replaced by $\overline{RQ}/\overline{GT_1}$ & $\overline{RQ}/\overline{GT_0}$. By using bus request signal another master can request for the system bus and the processor communicate that the request is granted to the requesting master by using bus grant signal

## Addressing modes :-

8086 has 8 addressing modes.

1) Immediate addressing mode
2) Register addressing mode
3) Direct addressing mode
4) Register Indirect addressing mode
5) Based addressing mode
6) Indexed addressing mode
7) Based Index addressing mode
8) Based Index with displacement

### Immediate addressing mode :-

The addressing mode in which the data operand is part of the instruction itself.

Eg :- MOV CX, 4929H (16 bit register → CX, 4929 → 16 bit Number)
MOV AL, FFH (AL → 8 bit register)
ADD AX, 2387H ;

### Register addressing mode

In this the register is the source of an operand in the instruction

Eg :- MOV AX, CX [Register]

### Direct addressing mode

The addressing mode in which the effective address of memory location is written directly in the instruction

Eg :- MOV AX, [1592H] [ ] → direct address

### Register indirect Addressing mode

This addressing mode allows data to be address at any memory through an offset address hold in any of the following base pointer (BP), Destination Index (DI), SI, BX

SI → Source Index
BX → Base register

Eg :- MOV AX, [BX]

Suppose the register BX Contain 3895H then the Content of 4895H move to AX

Based addressing mode :-

In this addressing mode the off-set address of operand is given by Sum of Content of BX/BP, register and 8/16 bit displacement. This displacement is added to Content of a base reguter [BX + 8/16]

Indexed addressing mode

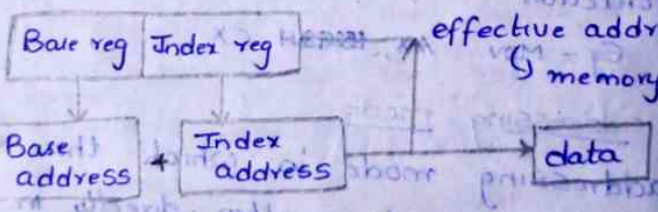
In this addressing mode the operand off-set address is found by adding Content of SI (Source Index Register) or DI Reguter and 8-bit / 16-bit displacement

Eg:- MOV BX, [SI + 16]

Based Index addressing mode

In this A.M the off-set address of an operand is Computed by Summary of base register to the Content of Index register

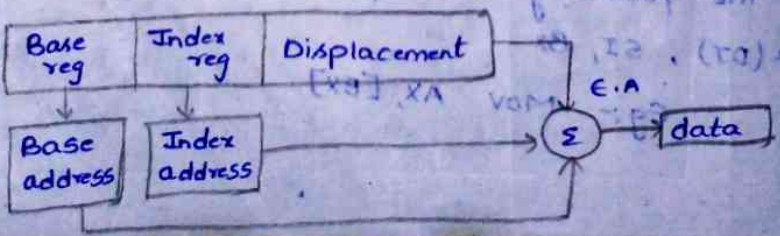
Eg:- ADD CX, [AX + SI]




Based Indexed With displacement :-

In this addressing mode the operand off-set addressing is Computed by adding the base reguter and Index register and 8/16 bit displacement.

Eg:- MOV AX, [BX + DI + 8]
MOV CX, [BX + SI + 16]



Eg :- MOV AX, [BX+DI+8]
MOV CX, [BX+SI+16]

Minimum mode of 8086 :-

The 8086 µp operates in minimum mode when MN/$\overline{MX}$ = 1.

→ In minimum mode, 8086 is only the processor in the system which provides all the control signal which are needed for memory operations & I/O interfacing

→ Here the circuit is simple but it does not supports multi processing.

→ The other components which are transreciever, latches, 8284 clock generator, 74138 decoder, memory and I/O devices are also present in the system.

→ The address bus of 8086 is 20-bit long by this we can access $2^{20}$ byte memory i.e 1MB. Out of 20 bit bytes, 16 bytes x bit $A_0$ to $A_{15}$ are multiplexed with a data bus.

→ By multiplexing, it means they will act as address line during the first T-state of the machine cycle and in the rest, they act as data line, $A_{16}$ - $A_{19}$ are multiplexed $S_3$ - $S_6$

8282 Latch :-

The latches are buffer D-flip flops. They are used to separate the valid address from the multiplexed address/data bus. By using the control signal ALE which is connected to strobe (STB) of 8282. The ALE is active high signal, here 3 latches are required because the address is 20 bit.

8286 transreceiver :-

They are bidirectional buffers and also known as data amplifiers. They are used to separate valid data from multiplexed address and data bus

Two transreceivers are needed because the data bus is 16-bit. 8286 is connected to DT/$\overline{R}$ and $\overline{DEN}$ (data enable) signals. They are enabled through the DEN signal. The direction of data on the data bus is controlled by the DT/$\overline{R}$ signal. DT/$\overline{R}$ is connected to T and $\overline{DEN}$ is connected to $\overline{OE}$.

| $\overline{DEN}$ | DT/$\overline{R}$ | Action |
|---|---|---|
| 1 | X | Transreceiver is disabled |
| 0 | 0 | Receive data |
| 0 | 1 | Transmit data |

Direction of data flow

→ At the time of data transfer to enable output of transreceiver its $\overline{OE}$ (output enable) should be low. This is accomplished by connection of DEN signal of 8086 to OE pin of 8286. Since DEN signal goes low then when CPU is ready to send or receive the data

→ M/$\overline{IO}$ = 1 then memory transfer is performed over the bus and M/$\overline{IO}$ = 0 then IO operation is performed.

→ $\overline{RD}$ indicates that 8086 is performing read data or instruction fetch process is occuring. During read operations one other control signal is also used. Which is DEN and it indicates the external devices when they should put data on the bus.

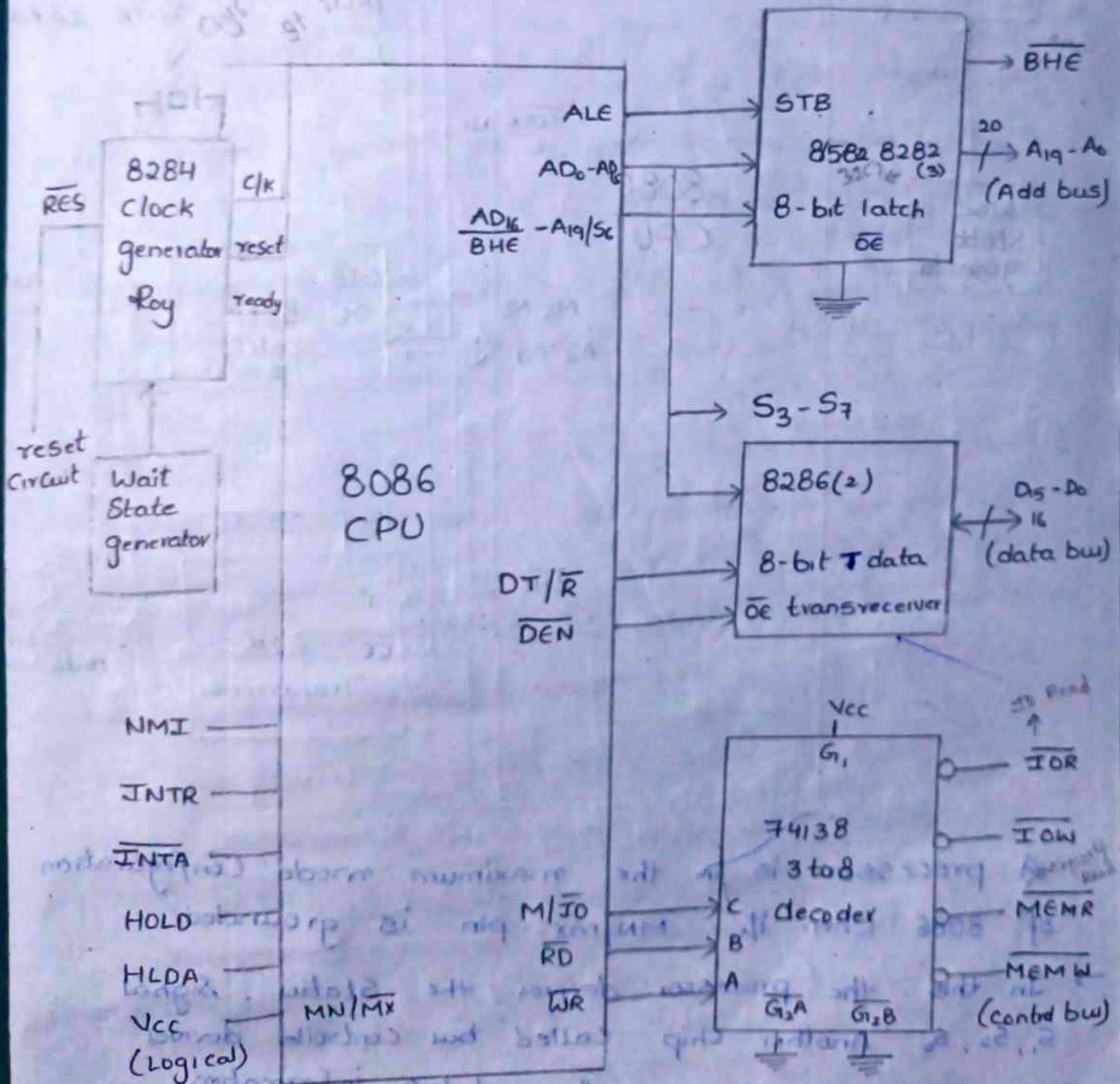
→ Control signals for all operations are generated by decoding M/$\overline{IO}$, $\overline{RD}$, $\overline{WR}$. These are decoded by using 74138 3:8 decoder.

| M/$\overline{IO}$ | $\overline{RD}$ | $\overline{WR}$ | action |
|---|---|---|---|
| 1 | 0 | 1 | Memory read |
| 1 | 1 | 0 | Memory write |
| 0 | 0 | 1 | I/O read |
| 0 | 1 | 0 | I/O write |

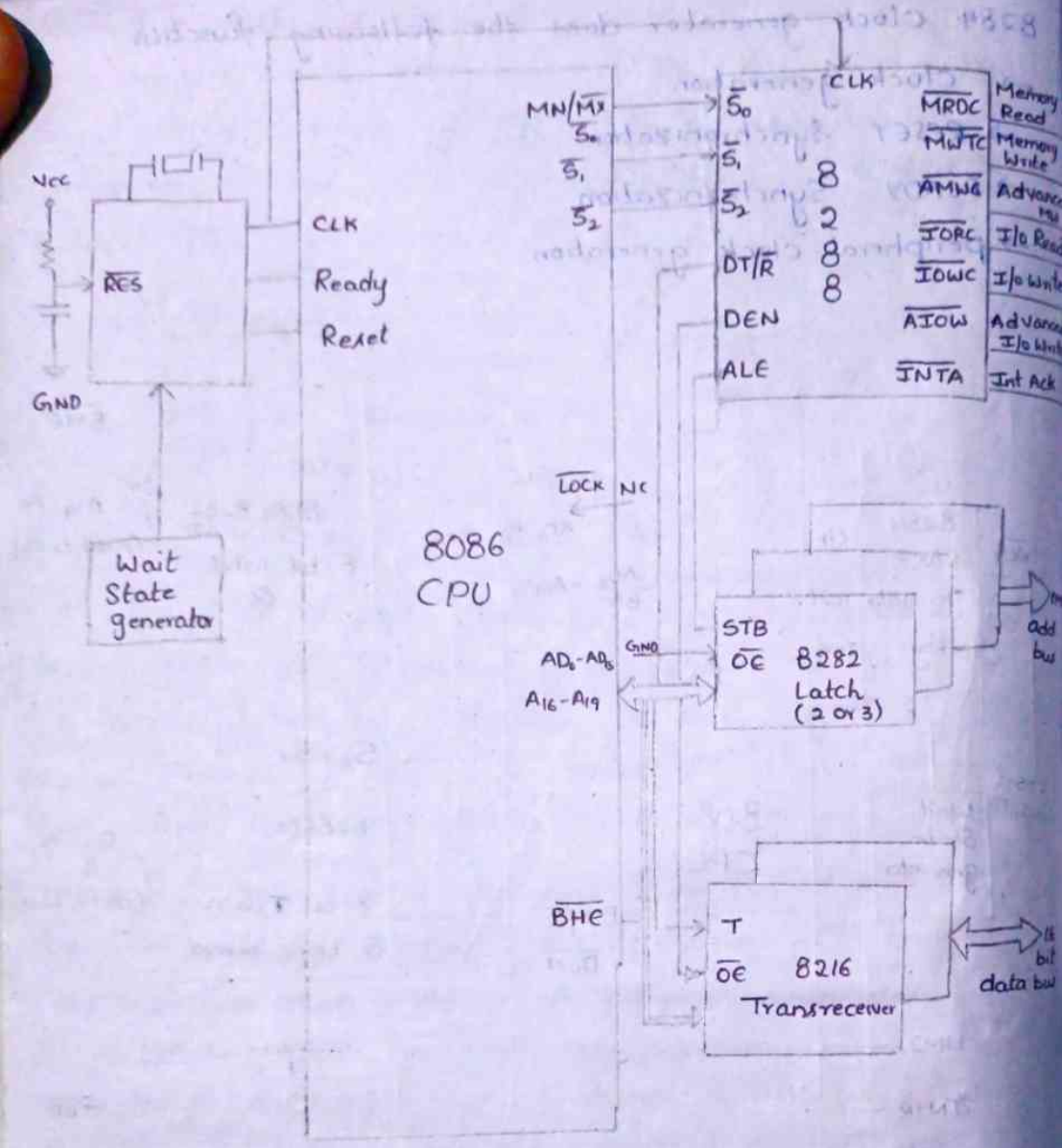
# 8284 Clock generator :-

8284 clock generator does the following function

→ clock generation
→ RESET Synchronization
→ READY Synchronization
→ peripheral clock generation



# Maximum mode of 8086 :-




→ A processor is in the maximum mode Configuration of 8086 when its MN/MX pin is grounded

→ In this the processor derives the status signal $S_1, S_2, S_0$ Another Chip called bus controller derives the Control Signal using this status information.

→ In the maximum mode there may be more than one microprocessor in the system Configuration

→ The Components in the System are same as in the minimum mode system.

→ The basic function of the bus Controller chip IC 8288, is to derive Control Signals like RD and WR DEN, DT/$\overline{R}$, ALE etc. using the information by the processor on the Status lines.

→ The bus controller chip has input line $S_2$, $S_1$, $S_0$ & C/K. These inputs to 8288 are driven by CPU.

→ It derives the outputs ALE, DEN, DT/$\overline{R}$, MRDC, MWTC, AMWC, IORC, IOWC and AIOWC

→ INTA pin used to issue two acknowledge pulses to the interrupt Controller or to an interrupting device. IORC, IOWC are I/o read and I/o write Command Signals. These Signals enable an IO interface to read or write data from or to the address bus.

→ The MRDC, MWTC are memory read Command and memory write Command used as memory read and write Signal.

→ For both these write Command Signals the advanced Signals namely AIOWC and AMWTC are available

| Status inputs | | | CPU Cycle | 8288 Command |
|---|---|---|---|---|
| $\overline{S_2}$ | $\overline{S_1}$ | $\overline{S_0}$ | | |
| 0 | 0 | 0 | Int Ack | $\overline{INTA}$ |
| 0 | 0 | 1 | Read I/o port | $\overline{IORC}$ |
| 0 | 1 | 0 | Write I/o port | $\overline{IOWC}$, $\overline{AIOWC}$ |
| 0 | 1 | 1 | Halt | None |
| 1 | 0 | 0 | Inst fetch | $\overline{MRDC}$ |
| 1 | 0 | 1 | Read memory | $\overline{MRDC}$ |
| 1 | 1 | 0 | Write memory | $\overline{MWTC}$, $\overline{AMWC}$ |
| 1 | 1 | 1 | inactive | None |

→ Here the only difference b/w min & max mode is the status signals used and available Control and advanced Command Signals.

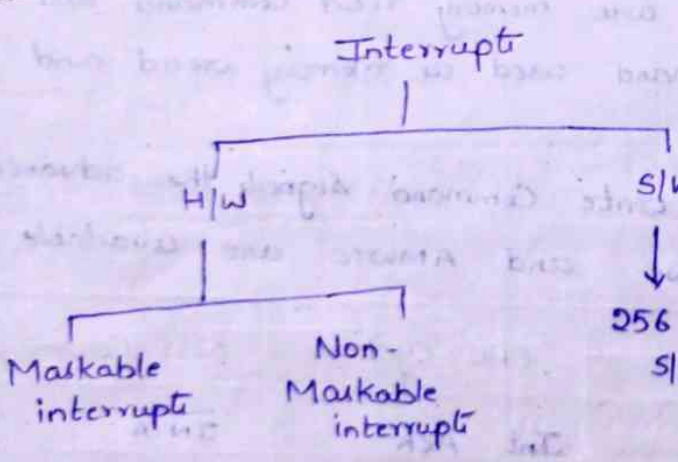
## Interrupts in 8086 :-

→ While the microprocessor is executing a program an interrupt breaks the normal sequention of execution of instruction, divert it execution to some other program called ISR. ISR is a special program to instruct the up on how to handle the interrupt. after executing control returns to back again to the main program.

→ The processor can be interrupt in the following ways.

(i) By an external signal generated by a peripheral
(ii) By an internal signal generated by a special instruction in the program.
(iii) By an internal signal generated due to an exceptional condition which occurs while executing an instruction.

## Types of Interrupt :-




### Hardware interrupt :-

Hardware interrupt are caused by any peripheral device by sending a signal through a specified pin to the microprocessor. They are 2 interrupt in 8086

(i) NMI (Non-maskable interrupt)
(ii) INTR (Interrupt Request)

NMI :- It is a single pin non-maskable interrupt which can not be disable

→ It is the highest priority interrupt in hardware interrupt

→ It is type-2 interrupt

INT R :-

INT R provides the single request. It is activated by IO port.

→ This can be marked or delayed that's why this is called maskable interrupt.

Software interrupt :-

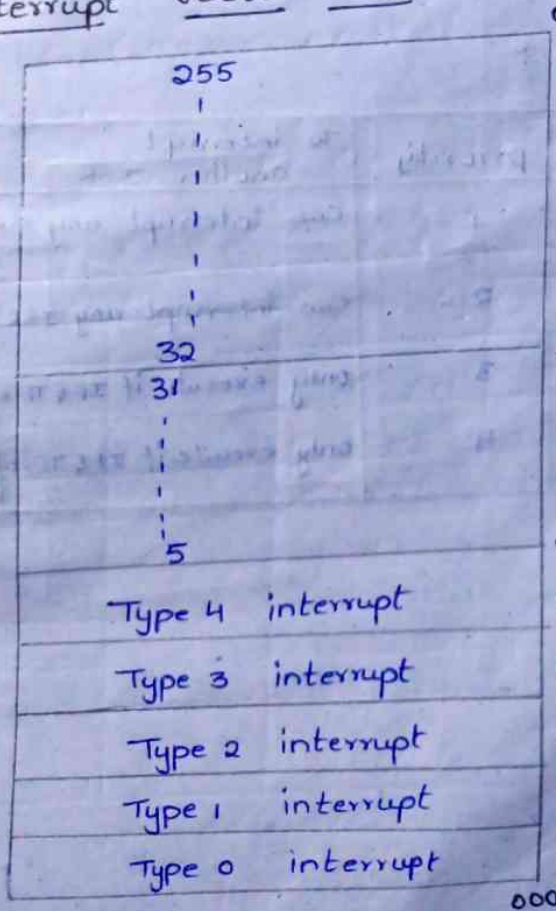
The S/W interrupts can be generated by inserting the instruction INT within the program.

→ They are 256 types of S/W interrupts in 8086

→ In 8086 format INT Type ranges from 00H - FFH

→ Starting address ranges from 00000H - 003FFH

8086 interrupt Vector table




purpose of debugging

power failure cond

Type 0 interrupt :-

It corresponds to divided by '0'

→ When the Quotient from division instruction is too large 8086 will automatically execute type 0 interrupt

Type 1 interrupt :-

Single step execution for de-bugging the program.

→ In this instruction the processor will execute one instruction and wait for further instruction

Type 2 interrupt :-

→ It represents NMI and it is used in power failure condition

Type 3 interrupt :-

→ It is used for de-bugging the program

Type 4 interrupt :-

→ It is overflow interrupt. It is used for check overflow condition after any signed arthimetic operation in the system

Interrupt priority :-

| Interrupt | priority | To interrupt another ISR |
|---|---|---|
| INT n | 1 | Can interrupt any ISR |
| NMI | 2 | Can interrupt any ISR |
| INTA | 3 | Only execute if IF & TF=0 |
| Single step | 4 | only execute if IF & TF=0 |