

MICROPROCESSOR :

It is a semiconductor device which consists of logic gates manufactured by using LSI or VLSI Technology capable of doing arithmetic or logical operation.

or

It is a multipurpose programmable logic device that reads binary instructions from a storage device called memory, accepts binary data as I/P and process the data as per the instruction & provide the result as O/P.

APPLICATIONS :

1. Traffic light control system
2. Temperature control in blast furnace.
3. Speed control in stepper motor
4. Rolling display
5. Water level indicator
6. Electronic appliances
7. Used in DSP.
8. Used in mobile phones.
9. Used in nanotechnology.
10. Used in security system.
11. Used in automobiles.

EVOLUTION :

1. Intel 4004 - 4 bit
2. Intel 4040 - 4 bit
3. Intel 8008 - 8 bit
4. Intel 8080 - 8 bit
5. Intel 8085 - 8 bit
6. Intel 8086 - 16 bit
7. Intel 80186 - 16 bit
8. Intel 80286 - 32 bit 16 bit (no. of address line is 24)
9. Intel 80386 - 32 bit
10. Intel 80486 - 32 bit
11. Pentium - 64 bit

Features of 8085 : It was developed by N-MOS technology.

1. It is a 8-bit processor.
2. Operating freq. is 3-5 MHz
3. It requires 5V DC supply.
4. Has 16 address bus, hence the memory size is 2^{16} i.e. 64KB.
5. It has 80 different instructions.
6. It has 246 opcode / machine code / Hex-code

opcode - operational code

instructⁿ = opcode + operand
(compulsary) (optional)

9. The clock cycle period is 320 nanosec.

[clock cycle / T-state / T-cycle]

7. Here all the datas are in hexa-decimal form.

8. It is a 40-pin DIP IC-packaging. (DIP = Dual In Packaging)

Architecture :

The whole architecture consists of 3 section

1. ALU (Arithmetic Logic Unit)
2. Timing & Control Unit
3. Registers (Imp)

1. ALU :

This unit is capable of doing all logical & arithmetical operations like addⁿ, subtracⁿ and logic and, logic OR, XOR, increment, decrement, shift right, shift left etc.

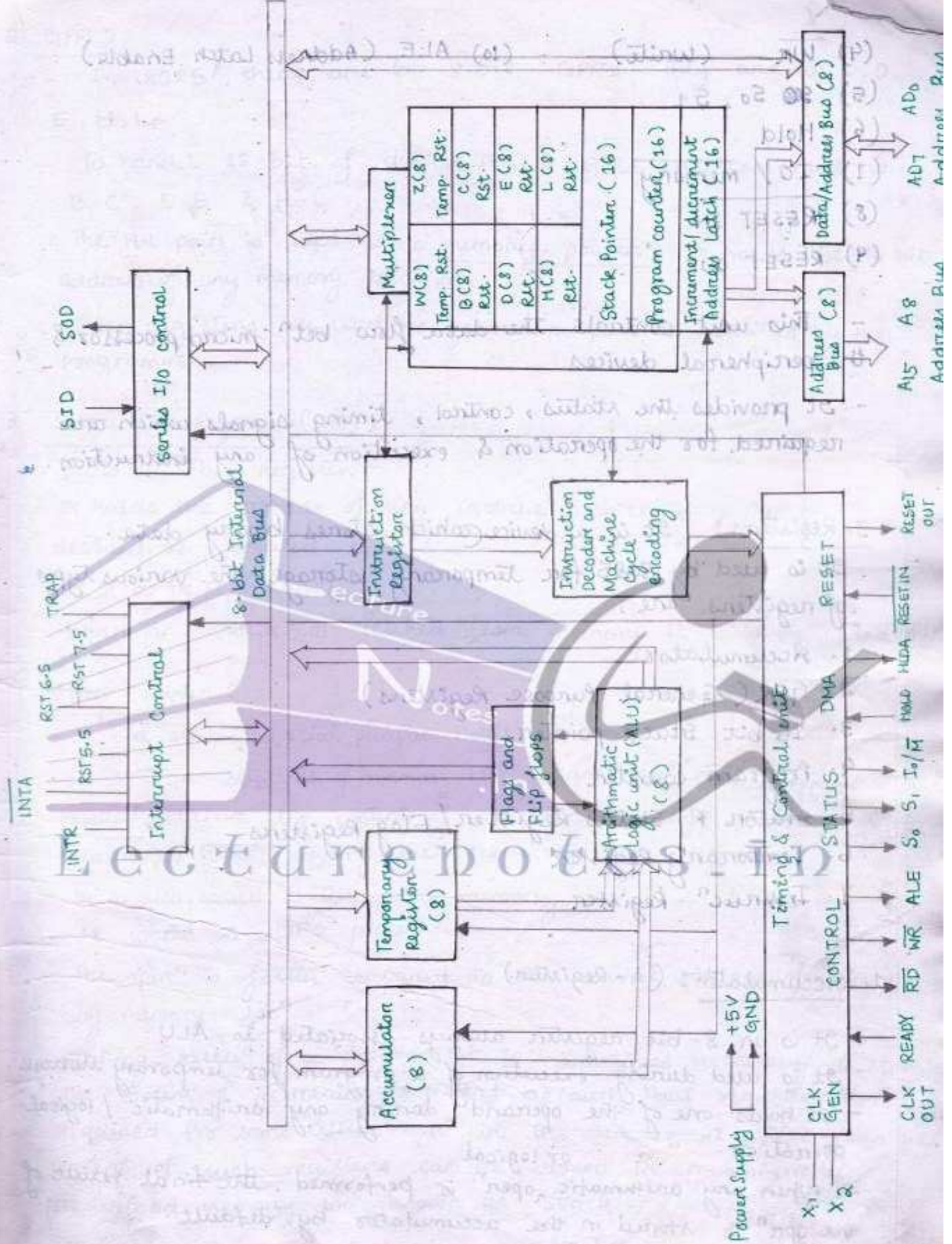
8085 is not capable to multiply or divide directly since it doesn't have any specific instructⁿ.

2. Timing & Control Unit :

It generates the timing & control signals which are necessary for the execution of any instructions.

The various signals are

- (1) Clock (CLK)
- (2) Ready
- (3) Rd~~low~~ (Readbar) It is an active low signal.



[8085 MICROPROCESSOR FUNCTIONAL BLOCK DIAGRAM]

(4) \overline{WR} (Write)

(10) ALE (Address Latch Enable)

(5) ~~So~~ S_0, S_1

(6) Hold

(7) $\overline{IO}/\overline{memory}$

(8) RESET IN

(9) RESET OUT

- This unit controls the data flow betⁿ micro-processor & the peripheral devices

- It provides the status, control, timing signals which are required for the operation & execution of any instruction.

3. Registers: It is a device which stores binary data.

It is used by 8085 for temporary storage. The various types of registers are,

1. Accumulator

2. GPR (General Purpose Registers)

3. 16 bit Stack pointer

4. Program counter

5. Status & Status Register / Flag Registers

6. Temporary Register

7. Instrucⁿ Register

1. Accumulator: (A-Register)

- It is an 8-bit register always associated to ALU.

- It is used during execution of a program for temporary storage.

- It holds one of the operand during any arithmetic / logical operation.

- When any arithmetic operⁿ is performed, the final result of the operⁿ is stored in the accumulator by default.

- It is always act as an I/p to ALU.

2. GPR :

- In 8085, there are 6 8-bit GPRs. They are B, C, D, E, H, L.
- To handle 16-bit of data, the 6 GPRs are combined as B-C, D-E & H-L.
- The H-L pair is used as a memory pointer. It holds the 16 bit address of any memory location.
- These registers are used for general purpose as defined by the programmer.

3. Instruction Register (IR) :

- It is a 8-bit register.
- It holds the opcode of any instrucⁿ which is going to be decoded or executed.
- It is a part of ALU.
- When an instrucⁿ is fetched from memory it will be loaded in IR.

4. Stack Pointer :

- It is a 16-bit special purpose register.
- Stack is a sequence of memory locⁿ assigned by the programmer to store/retrieve the content of accumulator, flags, PC (program counter), GPRs during the execuⁿ of any program.
- It is also called extra area memory locⁿ.
- It works on LIFO process.
- The operⁿ is faster compared to the normal storage of data in any memory locⁿ.
- During execuⁿ of a program it is sometimes necessary to store the content of certain registers because that register is required for some other operⁿ in the subsequent steps. Then the content of such registers can be stored in an assigned/predefined memory locⁿ known as stack.
- The top of the stack is known as stack top.
- Stack pointer is a 16-bit register which holds the address of the stack top.

5. PROGRAM COUNTER: (PC).

- It is a 16 bit register
- It holds the address of the next instrucⁿ which is going to be executed.
- Microprocessor fetches an instrucⁿ from memory, executes it & increment the content of PC each time.

6. FLAG REGISTER / STATUS REGISTER:

- It is a 8-bit register. i.e. 8 flipflops are there.
- Out of these 8 FFs, 5 FFs are used to show the various status of the accumulator after any arithmetic or logical operation performed.

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
Sign	Zero	X	AC	X	P (Parity)	X	CY (Carry)

Various Status Flags:

D₀ → Carry flag . D₂ → Parity flag

D₆ → Zero flag

D₇ → Sign flag

D₄ → Auxiliary carry flag

D₅, D₃, D₁ → Inactive

1. Carry Flag:

The carry flag / carry bit is set to 1 indicates a carry bit is generating after any arithmetic / logical operⁿ. Else this bit is set to zero.

2. Parity Flag:

The parity flag P is set to 1 if the result of any arithmetic or logical operⁿ is having even no. of 1. Else it is set to zero.

3. Auxilliary Carry: This flag is set to 1 if any carry is propagating from 3rd bit to 4th bit during any arithmetic / logical oprⁿ.
(D3) (D4)

eg:

	D ₇	D ₆	D ₅	D ₄		D ₃	D ₂	D ₁	D ₀
	0	0	0	0		0	1	0	1
+	0	0	0	0		0	1	1	1
<hr/>									
	0	0	0	0		1	1	0	0

Hence no carry from D_3 to D_4 , Hence $AC = 0$.

4. Zero flag: This flag is set to 1 if the content of accumulator is 00H after any arithmetic / logical opⁿ is performed.

eg:

0	0	0	0	0	1	0	1
0	0	0	0	0	1	1	1
<hr/>							
0	0	0	0	1	1	0	0

Here accumulator content is not zero. Hence zero-flag content is zero.

5. Sign flag:

- This flag set to one if the result of any arithmetic or logical operation is negative else it is zero.



7. TEMPORARY REGISTER :

- This is a 8-bit register associated to ALU and it holds the data during any arithmetic or logical operation temporarily.
- It is associated to processor not to user.

BASIC OPERATION OF 8085 :

The basic 5 operations are ,

1. Fetch : Fetching of an Instruction.
2. Memory Read : Taking a data from memory i.e. Read.
3. Memory write : Write data into memory.
4. I/O read : Reading data from I/O devices.
5. I/O write : Write a data into I/P & O/P devices.

INSTRUCTIONS OF 8085 :

- Instruction is a command to a processor to perform certain operations.
- Each instruction consists of 2 parts ,
 - a) Opcode
 - b) operand

OPCODE : It is the 1st part of an instruction which specifies the task performed by the processor.

OPERAND : The 2nd part of an instruction is the operand.

The operand may be a 8-bit data , 8-bit address , 16-bit data , 16-bit address.

The I/O devices have 8-bit address but Memory address is 16bit

TYPES OF INSTRUCTIONS ACCORDING TO WORD SIZE :

According to the word size 8085 instructions are categorised in 3 groups.

1. 1 byte instructions.
2. 2 byte
3. 3 byte

If
(a) 1-byte Instructions: The opcode and operand of any instruction take 1-byte of memory space, then it is called 1-byte instruction.

eg: $\begin{array}{c} \text{MOV} \quad A, B \\ \text{opcode} \quad \text{operand} \end{array}$
 $\begin{array}{c} \text{ADD} \quad B \\ \text{opcode} \quad \text{operand} \end{array}$ (content of B is added to content of accumulator)
 $\text{ADD} \quad M$

(b) 2-byte Instructions: In a 2-byte instructions the 1st byte specifies the opcode & 2nd byte specifies the operand which may be a data or address.

eg: $\text{MVI} \quad B, 78H$

Here $\text{MVI} \quad B \rightarrow \text{opcode}$
 $78H \rightarrow \text{operand}$

Fetch: Reading of opcode (eg: $\text{MVI} \quad B$) (Here B is the RST)

Memory Read: Reading of operand (eg: $78H$)

(c) 3-byte Instruction: The 1st byte specifies the opcode & the next 2-bytes specifies a 16-bit data/Address.

Lecture notes in

BASED ON OPERATION:

Basing on operation there are 5 types of instructions,

1. Data Transfer group
2. Arithmetic & Logic group
3. Logical group.
4. Branch Control group
5. I/O machine control group

1. Data Transfer Group:

(1) **MOV R₁, R₂** :

[R₂] → R₁

eg: MOV A, B

* Register addressing mode

The content of R₂ goes to R₁.

Here operation is only fetch.

It is a one byte instruction.

No. of T-states is 4.

(2) **MOV R, M** :

Content of memory locⁿ moved to register.

eg: MOV B, M

* Indirect addressing mode

It is a one byte instruction.

Here operation is fetch and memory read. (ie. machine cycle is 2)

No. of T-states = 7 (4+3)

(3) **MOV M, R** :

Content of register goes to the memory locⁿ which is already predefined through the LXI statement.

Machine cycle is 2.

Operation is fetch and memory write.

T-states = 7

* Indirect addressing mode

1-byte instruction.

(4) **MVI R, data** :

8-bit

move immediately the data to the R₁.

eg: MVI B, 55H

* Immediate addressing mode.

Word size 2-byte.

Machine cycle - 2

operation - fetch, memory read

No. of T-states = 4+3 = 7

(5) MVI M ; data:

move immediately the 8-bit data to any memory locⁿ.

machine cycle = 3

operⁿ = fetch, memory read, memory write (Fetch → MR → MW)

word size = 2 byte

No. of T-states = 10 (4+3+3)

(6) LXI Rp, address (16 bit) data:

Load the ~~HL~~ pair register pair immediately with the 16 bit data.

eg: LXI H 1000H → address (H means HL pair)

LXI B 1000H → data (B means BC pair)

LXI D 2000H (D means DE pair)

* Immediate

- Word size - 3 bytes

- No. of machine cycles - 3

- operation - Fetch, memory read, memory read.

eg: LXI H, 4300H

8
0
8
5

Memory	
2000	opcode
2001	00H (LSB)
2002	43H (MSB)

The LSB part of the 16 bit data i.e. 00H will be stored at register 'L' and the MSB part i.e. 43H will be stored at register 'H'.

(7) LDA, address:

Load the content of memory locⁿ directly on the accumulator:

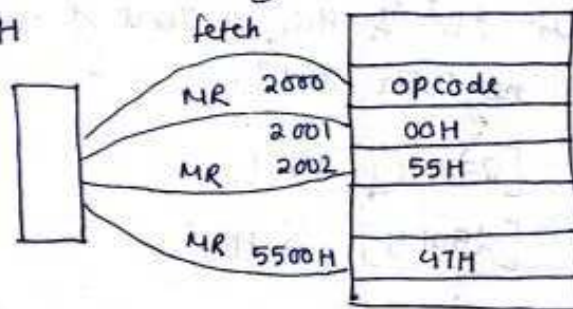
eg: LDA, 5500H

operation = fetch + MR + MR + MR

Machine cycle = 4

Word size = 3 bytes

T-states = 13



- 1st operation is fetch.

- Then 2 MR operⁿs to read the address i.e. 5500H

- Then one more MR operⁿ to read the data i.e. 47H present in the memory locⁿ 5500H.

* Direct addressing mode.

(8) **STA, address**:

Store the content of accumulator in the specified memory locⁿ given in the instrucⁿ.

Machine cycle = 4

operation = fetch, MR, MR, MW

Word size = 3 byte

T-state = 13

* Direct

(9) **XCHG**:

Exchange the content of HL pair with DE pair.

Word size 1 byte

Machine cycle = 1

Operation = Fetch

T-state = 4

* Implicit

(10) **LHLD address**:

Load the HL-pair directly

eg: LHLD 2500H

It indicates the content of 2500H will be stored in 'L' and the content of 2501H will be stored in 'H' by default.

It indicates the content of memory locⁿ 2500H will go to register 'L' & the content of next memory locⁿ i.e. 2501H will go to register 'H'.

[2500H] → L

[2501H] → H

Wordsize = 3 bytes

Machine cycle = 5

operation = fetch, MR, MR, MR, MR

T-States = 16

2000	LHLD
2001	06H
2002	25H
2500	X
2501	Y

X → L

Y → H

11. **SHLD address**:

Store the HL pair directly.

eg: SHLD 4500H

It will store the content of 'L' to the 1st memory locⁿ given in the instrucⁿ itself & the content of 'H' will be stored in the next memory locⁿ.

Word size = 3 bytes

Machine cycle = 5

operⁿ = fetch + MR + MR + MW + MW

T-States = 16

[L] → 4500H

[H] → 4501H

12. **LDAX Rp**:

It is an indirect addressing mode.

It loads the content of memory locⁿ whose address is in the register pair into the accumulator.

eg: [B] = 45H

eg: LDAX BC

[C] = 73H

Then it will load the content of 4573H into accumulator

Wordsize = 1 byte

Machine cycle = 2

operation = fetch + MR

T-states = 7

13. **STAX rp** :

The content of accumulator whose address is will go to the memory locⁿ whose address is given in the register pair.

eg: STAX, DE

wordsize - 1 bytes

Machine cycle - 2

operⁿ - Fetch + MW

T-states = 7

a. Arithmetic Group :

14. **ADD R** : Add the content of any register to accumulator and store the result in accumulator.

eg: ADD B

Word size = 1 byte

Machine cycle = 1

operⁿ = fetch

T-state = 4

15. **ADD M** : Add the content of memory locⁿ with the accumulator and the result will be stored in accumulator.

word size = 1 byte

Machine cycle = 2

operⁿ = fetch + MR

T-states = 7

16. **ADC R** : Add the content of register with carry to the accumulator & result of the operⁿ is stored in accumulator.

word size = 1 byte

M.C = 1

operⁿ = fetch

T-states = 4

17. **ADC M**: Add the content of memory locⁿ with carry to the accumulator & store the result in accumulator.

Word size = 1 byte

Machine cycle = 2

Operation = fetch + MR

T-states = 7

18. **ADI data**: (8bit) Add immediately the data with accumulator.

Word size = 2 bytes

M-cycle = 2

Operⁿ = fetch + MR

T-states = 7

eg: ADI 35H

19. **ACI data** (8bit): Add immediately the data with carry with accumulator.

Word size = 2 bytes

M-cycle = 2

Operⁿ = fetch + MR

T-states = 7

20. **DAD Rp**: Add the content of register pair with HL pair;

M-cycle = 3

No. of T-states = 10

Operⁿ = Fetch + MR + MR

Word size = 1 byte

21. **SUB R**: Subtract the content of any register from accumulator and stores the result in accumulator.

eg: SUB B

Word size = 1 byte

Machine Cycle = 1

Operation = Fetch

T-states = 4

22. **SUB M**: Subtract the content of memory location from the accumulator and the result will be stored in accumulator.

Word size = 1 byte

Machine cycle = 2

Operation = fetch + MR

T-states = 7

23: **SUI data** : Subtract immediately the data from accumulator.

Word size = 2 bytes

Machine cycle = 2

Operation = fetch + MR

T-states = 7

24: **SBB R** : Subtract the content of register with borrow from the accumulator and stores the result in accumulator.

Word size = 1 byte

Machine cycle = 1

Operation = Fetch

T-states = 4

25: **SBB M** : Subtract the content of memory location with borrow from the accumulator and stores the result in accumulator.

Word size = 1 byte

Machine cycle = 2

Operation = fetch + MR

T-states = 7

26: **SBI data** : Subtract immediately the data with borrow from accumulator.

Word size = 2 bytes

Machine cycle = 2

Operation = fetch + MR

T-states = 7

27: **INR R** : Increment the content of register ~~para~~ by 1.

Word size = 1

Machine cycle = 1

Operⁿ = Fetch

T-states = 4

28. **INR M**: Increment the content of memory locⁿ by 1.

Word size = 1

Machine cycle = 3

operⁿ = fetch + MR + MW

T-states = 10

29. **DCR R**: Decrement the content of register by 1.

word size = 1

machine cycle = 1

operation = fetch

T-states = 4

30. **DCR M**: Decrement the content of memory locⁿ by 1.

Word size = 1

Machine cycle = 3

Operation = fetch + MR + MW

T-states = 10

31. **INX rp**: Increment the content of register pair by 1

Word size = 1 byte

Machine cycle = 1

operation = fetch

T-states = 6 (exceptional case)

eg: INX • H

H	L	→	H	L
45	00		45	01

32. **DCX rp**: Decrement the content of register pair by 1

Word size = 1 byte

Machine cycle = 1

operⁿ = fetch

T-states = 6 (exceptional case)

3. Logical Group:

2nd Feb '10

33. **ANA R**: The content of register will have an 'and' operⁿ with the content of accumulator.

Word size = 1 byte

Machine cycle = 1

Operation = Fetch

T-states = 4

34. **ANA M**: The content of memory locⁿ will have an 'and' operⁿ with accumulator.

Word size = 1 byte

Machine cycle = 2

Operation = Fetch + MR

T-states = 7

35. **ANI 75H**: Immediate 'and' operⁿ will be performed of the given 8 bit data with accumulator.

Word size = 2 bytes

Machine cycle = 2

Operation = F + MR

T-states = 7

36. **ORA R**: The contents of the accumulator are logically ORed with the contents of register.

Word size = 1 byte

T-states = 4

Machine cycle = 1

Operation = Fetch

37. **ORA M**: The contents of the accumulator are logically ORed with the contents of memory.

Word size = 1 byte

Machine cycle = 2

Operation = F + MR

T-states = 7

38. **ORI data**: The contents of the accumulator are logically ORed with the 8-bit data in the operand & the result is stored in accumulator.

Word size = 2 bytes
Machine cycle = 2

Operation = F + MR
T-states = 7

39. **XRA R**: The contents of the register are XORed with accumulator

Word size = 1
Machine cycle = 1
Operation = Fetch
T-states = 4

40. **XRA M**: The contents of the memory are XORed with accumulator

Word size = 1
Machine cycle = 2
Operation = Fetch + MR
T-states = 7

41. **XRI data**: The 8-bit data is XORed with the content of accumulator

Word size = 2 bytes
Machine cycle = 2
Operation = Fetch + MR
T-states = 7

42. **CMA**: Complement the accumulator content.

eg: CMA

[A] = 75 → 8A

Word size = 1

Machine cycle = 1

No. of T-states = 4

Operation = Fetch

43. **CMC**: Complement the carry status.

The carry flag will get complemented while the other flags remain constant/unchanged.

Word size = 1

Machine cycle = 1

T-states = 4

Operation = Fetch

44. **STC**: Set the carry status flag.

This is set to '1' after the instruction is executed.

Word size = 1 byte

Machine cycle = 1

T-states = 4

Operation = Fetch

45. **CMP R**: Compare the register content with accumulator.

Word size = 1

Machine Cycle = 1

Operation = Fetch

T-states = 4

46. **CMP M**: Compare the content of the memory locⁿ with accumulator.

Word size = 1

machine cycle = 2

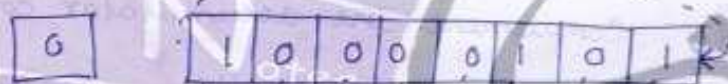
Operation = Fetch + MR

T-states = 7

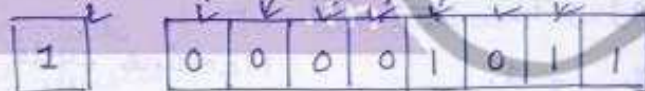
47. **RLC**: Rotate accumulator left.

The content of accumulator is rotated left by 1 bit. After RLC instruction executed, the 7th bit of the accumulator move to the carry bit as well as to the zeroth bit.

eg:



After RLC



eg:

1 00110001

0 01100010

Word size = 1

Machine Cycle = 1

Operation = Fetch

T-states = 4

48. **RRC**: Rotate accumulator right.

After this instruction executed, the 0th bit of 'A' move to the carry bit, as well as to the 7th bit.

eg:

0 10000101

1 11000010

C A

Word size = 1

Machine cycle = 1

Operation = Fetch

T-states = 4

49. **RAL** : Rotate accumulator left through carry.

After this instrucⁿ executed, the 7th bit of the accumulator moves to carry bit & the carry bit moves to the zeroth bit.

eg:

1 10000101
1 00001011
0 B

0 00110001
0 01100010
6 2

Word size = 1 byte

Machine Cycle = 1

Operation = Fetch

T-states = 4

50. **RAR** : Rotate the content of accumulator by 1 bit towards right through carry. The 0th bit will move to the carry bit & carry bit will move to the 7th bit.

1 10000101
1 11000010
C 2

Word size = 1

Machine cycle = 1

Operation = Fetch

T-states = 4

51. **CPI data** : Compare immediate with accumulator

Word size = 2 bytes

Machine cycle = 2

T-states = 7

Operation = F + MR

Branch Control Group:

The instructions of this group change the normal sequence of operⁿ of the program. There are 2 types of branch instrucⁿ.

i) Conditional

ii) Unconditional

1- Conditional: This instrucⁿ changes the normal sequence of operⁿ when the specified condⁿ given in the instrucⁿ is satisfied.

2- Unconditional: This instrucⁿ transfer the program to the given level unconditionally. The various branch control instrucⁿ are:-

xxxp Jump (JMP), call (CALL), return (RET)

Unconditional Jump

JMP address level: It is a 3 byte instrucⁿ.

$$M-C = 3$$

eg: JMP 4300H

$$T\text{-states} = 10$$

$$\text{oper}^n = F + MR + MR$$

The 2nd & 3rd byte of the instrucⁿ give the address of the level where the program jumps.

• Conditional jump: After execution of the conditional jump instrucⁿ, the program jumps to the instrucⁿ specified in the address level, if the specified condⁿ is satisfied.

The conditional jump will take 3 machine cycles if the condⁿ is True (10-T) and it takes 2 machine cycles (7-T) if the condⁿ is false.

$$3(MC) = F + MR + MR$$

$$2(MC) = F + MR$$

The various conditional jumps are:

JZ address level : Jump if result is zero.

JNZ address level : Jump if result is not zero.

JC address level : Jump if result has a carry

JNC address level : Jump if result has no carry

JP address level : Jump if the result is positive

JM address level : Jump if the result is negative

JPE address level : Jump if result has even parity.

JPO address level : Jump if result has odd parity.

Unconditional call:

Machine-cycle = 5

Openⁿ = F + MW + MW + MR + MR ?

T-states = 6 + 3 + 3 + 3 + 3 = 18

A subroutine always ends with a return where as the main program ends with a ~~halts~~ halt.

Conditional Call:

If condⁿ is true 5 M.C., T = 18

condⁿ is false 2 M.C., T = 9

CC : call on carry

CNC : call with no carry

CZ : call on zero

CNZ : call on no zero

CP : call on positive

CM : call on minus

CPE : call on parity even

CPO : call on parity odd

Unconditional Return : The program sequence is transferred from the subroutine to the calling program. This instruction is used in conjunction with CALL or conditional CALL instructions.

Word size = 1 byte

Machine Cycles = 3

Operation = Fetch + $\frac{1}{2}$ MR + MR

T-states = 10

Conditional Return :

If the condition is true, Machine cycle = 1

operation = Fetch

T-states = 6

If the condition is false, Machine cycle = 3

operation = Fetch + $\frac{1}{2}$ MR + MR

T-states = 12

OPCODE

DESCRIPTION

RC :	Return on carry
RNC :	Return with no carry
RZ :	Return on zero
RNZ :	Return on no zero
RP :	Return on positive
RM :	Return on minus
RPE :	Return on parity even
RPO :	Return on parity odd

I/O and Machine Control Group:

IN PORT address (8-bit) :

eg: IN 07H, it indicates, the I/O to the accumulator is from a I/O port whose address is given. The data available on the port address will move to the accumulator.

Word size = 2 byte

Machine cycle = 3

Operation = F + $\frac{1}{2}$ MR + I/O read

T-states = 10

OUT port address (8-bit): It indicates data from the accumulator will move to the port address given in instrucⁿ.

eg: OUT 75H

Word size = 2 byte

Machine cycle = 3

Operation = Fetch + $\frac{M}{2}R$ + I/O W

T-states = 10

(T-states = $3N+1$, $N = m.c.$)

PUSH: Push the content of register, accumulator, program counter to stack. eg: PUSH rp. The content of register pairs move to stack.

Word size = 1 byte

M.C = 3

Operation = Fetch + MW + MW

T-states = $6+3+3 = 12$

(Stack is a locⁿ present in memory but stack pointer is in processor)

POP: Move back the contents of stack to the register or register pairs which is already earlier stored in stack.

Word size = 1 byte

MC = 3

OP = F + MR + MR

T-states = 12.

HALT: Stop the execution of the processor.

Word size = 1 byte

MC = 1

OP = Fetch

T-states = 4

XTHL: Exchange the stack-Top^{data} with HL pair content

Word size = 1 byte

MC = 5

OP = Fetch + MR + MR + MW + MW

T-states: 16

[SP] \rightarrow L

[SP+1] \rightarrow H

SPHL: Move the content of HL pair to stack pointer.

Word size = 1

MC = 1

OP = F

T-states = 6

NOP: No operation. It is used to create some time delay

Word size = 1

MC = 1

OP = F

T-states = 4

EI: Enable the interrupt

Word size = 1

MC = 1

OP = F

T-states = 4

DI: Disable the interrupt.

Word size = 1

MC = 1

OP = F

T-states = 4

RIM: Read interrupt mark.

Word size = 1

MC = 1

OP = F

T-states = 4

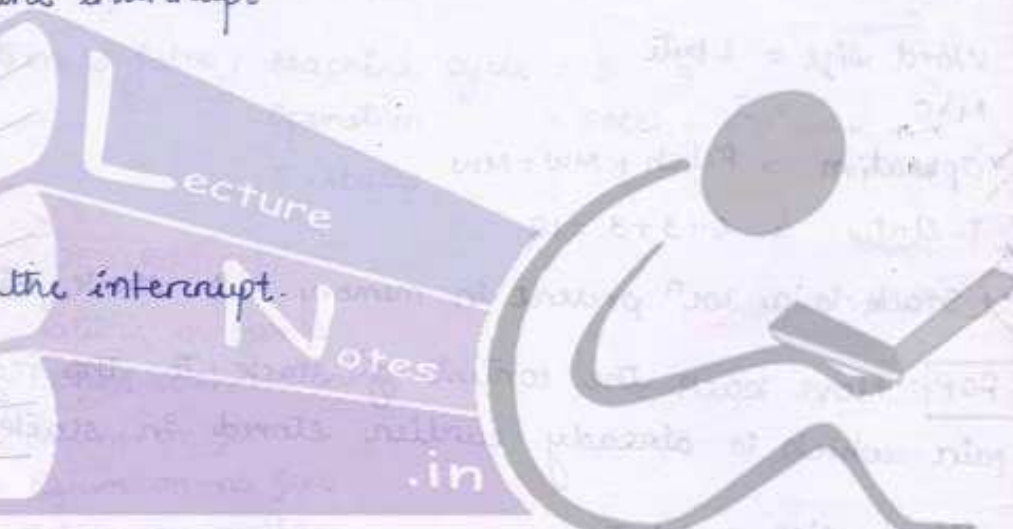
SIM: Set interrupt mark.

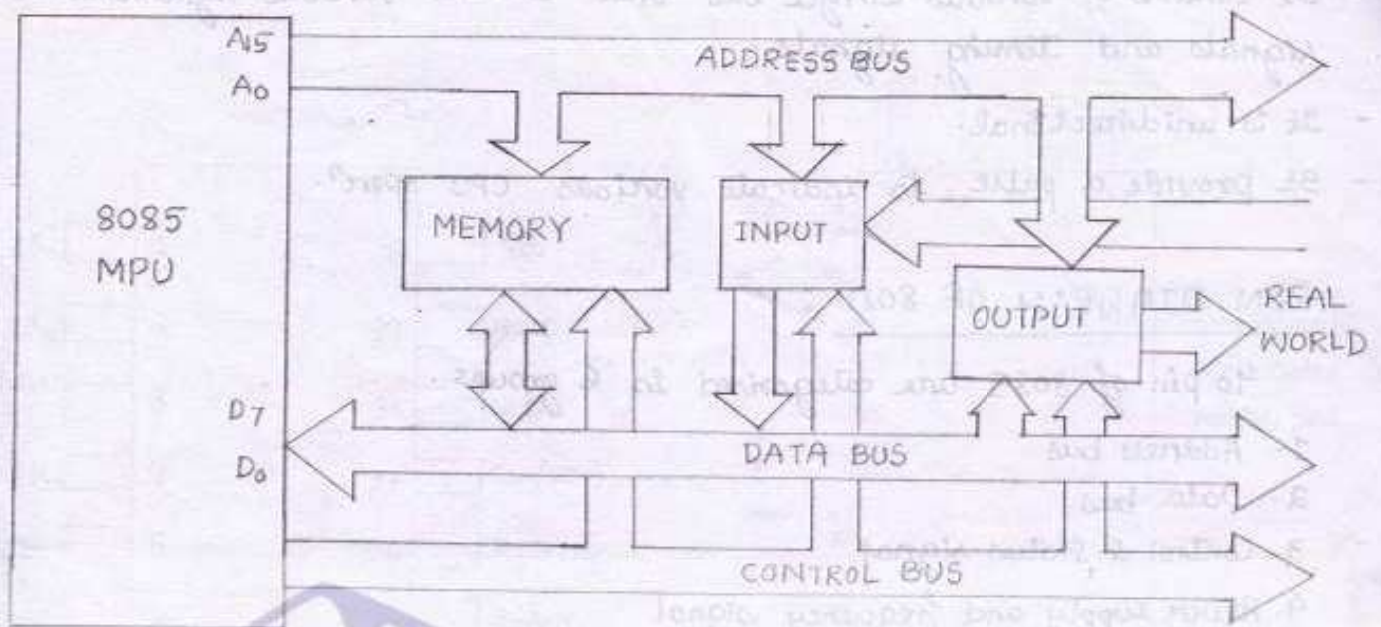
Word size = 1

MC = 1

OP = F

T-states = 4





(The 8085 Bus Structure)

8085 has 3 different Type of buses.

1. Data Bus
2. Address Bus
3. Control Bus

1. **ADDRESS BUS :** It is a group of 16 lines.

- It is always unidirectional.
- The bits in the address bus flow from CPU to peripheral devices or to any memory locⁿ.
- μP uses these buses for 2 basic operations.
 - Identification of a memory locⁿ
 - Identification of a peripheral locⁿ (i.e. I/O)
- 8085 can address upto $2^{16} = 64 \text{ KB}$ of memory space. with a starting address 0000H to FFFFH.

2. **DATA BUS :**

- It is a group of 8-lines, used to transfer data betⁿ processor and peripheral or memory devices in both the dirⁿ.
- It is bi-directional
- CPU uses data bus for transferring data.
- The data range is from 00H to FFH.

3. CONTROL BUS :

- It consists of various single bus that carries various synchronisation signals and timing signals.
- It is unidirectional.
- It provide a pulse to indicate various CPU operⁿ.

PIN DIAGRAM OF 8085 :

40 pin of 8085 are categorised in 6 groups.

1. Address bus
2. Data bus
3. Control & Status signal
4. Power supply and frequency signal
5. Externally initiated signal
6. Serial I/P and O/P port

1. Address Bus:

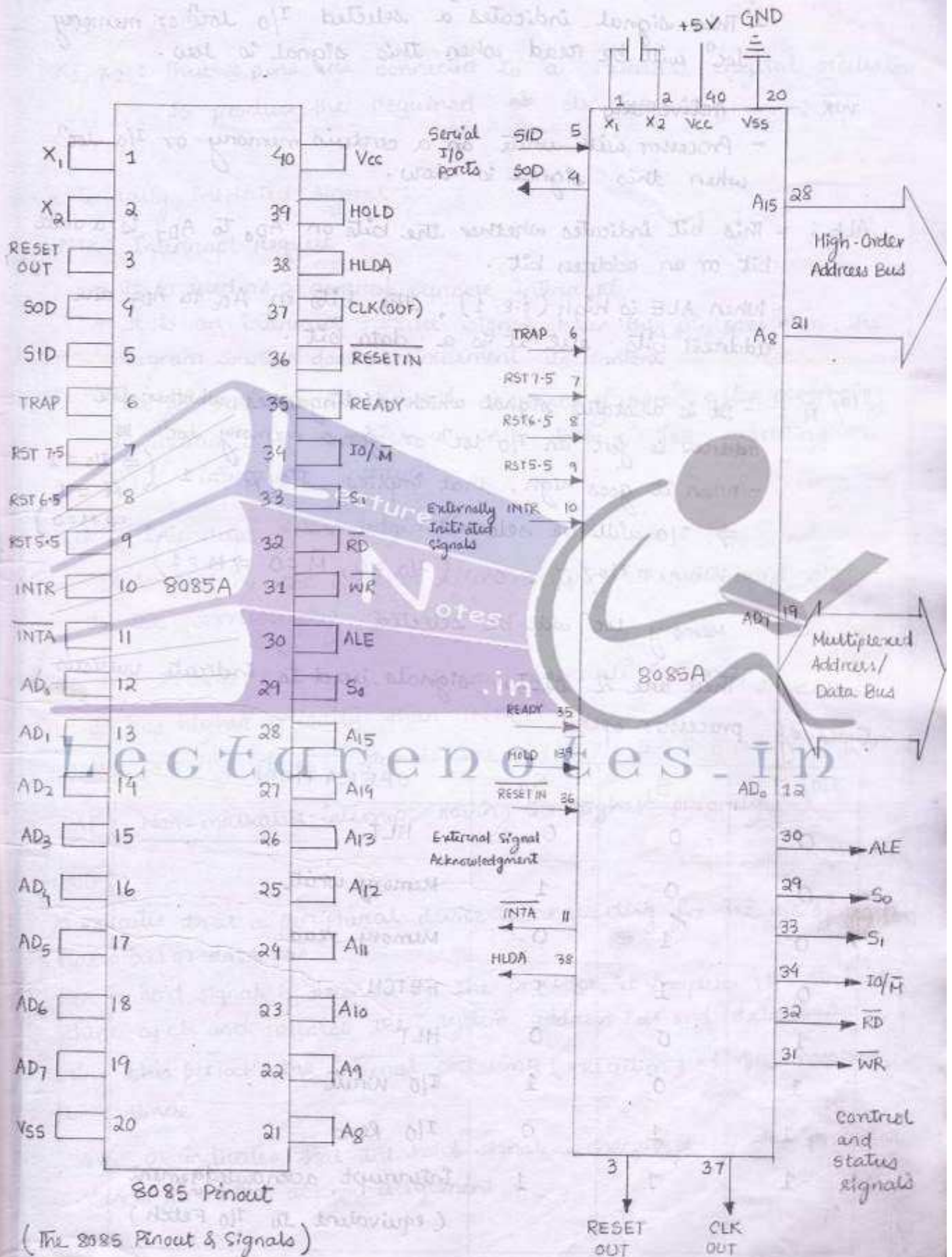
- It has 16 bit address bus.
- The pins are AD_0 to AD_7 , A_8 to A_{15} .
- Out of these 16 pins, the 8 MSB bits i.e. A_8 to A_{15} are only for carrying address whereas the 8 LSB bits i.e. AD_0 to AD_7 may carry address or data.

2. Data Bus:

- Also known as multiplexed address bus.
- The data pins are AD_0 to AD_7 .
- These pins are bi-directional.
- In executing an instruction during the 1st clock cycle (i.e. T_1 stage) the 8 LSB bits act as an address bit and for the remaining clk cycle it acts as a data bit.

3. Control and status signal:

- Under this section, there are 2 control signals and ⁴ status signals.
- The control signals are \overline{RD} (read) & \overline{WR} (write)
- It is an active low signal.



- The status signals are $S_0, S_1, I/O/\bar{M}$, ALE (address latch enable)

\overline{RD} : - It is an active low signal.

- This signal indicates a selected I/O locⁿ or memory locⁿ will be read when this signal is low.

\overline{WR} : - Active low

- Processor will write on a certain memory or I/O locⁿ when this signal is low.

ALE : - This bit indicates whether the bits on AD_0 to AD_7 is a data bit or an address bit.

- When ALE is high (i.e. 1), the bits on AD_0 to AD_7 are address bits else it is a data bit.

$I/O/\bar{M}$: - It is a status signal which distinguishes, whether the address is for an I/O locⁿ or for a memory locⁿ.

- When it goes high, that implies $I/O/\bar{M} = 1$ ($\Rightarrow I/O = 1$
 $\bar{M} = 1$
 $\Rightarrow M = 0$)
 $\Rightarrow I/O$ will be selected.

- When $I/O/\bar{M} = 0$ ($I/O = 0, \bar{M} = 0 \Rightarrow M = 1$)
 Memory locⁿ will be selected.

S_0, S_1 : There are 2 status signals, used to indicate various types of processor operⁿ.

$I/O/\bar{M}$	S_1	S_0	OPERATION
0	0	0	HLT
0	0	1	Memory write
0	1	0	Memory read
0	1	1	FETCH
1	0	0	HLT
1	0	1	I/O write
1	1	0	I/O Read
1	1	1	Interrupt acknowledgement (equivalent to I/O Fetch)

→ Power Supply and frequency signal:

$$V_{CC} = +5V$$

$$V_{SS} = GND$$

X_1, X_2 : These 2 pins are connected to an external crystal oscillator to produce the required ~~to~~ clk freq.

5 Externally Initiated signal:

INTR: Interrupt request.

- It is used as a general purpose interrupt.
- It is an interrupt request signal. When this pin goes high, the program counter doesn't increment its content.
- The μP performs the normal sequence of opⁿ, after performing all instrucⁿs it executes a CALL instrucⁿ for executing the INTR interrupt instrucⁿ.

INTA: Interrupt Acknowledgement

- It is sent by the processor after receiving the interrupt signal to the corresponding device.

RST 7.5, RST 6.5, RST 5.5: These are maskable interrupt.

- It has higher priority than INTR.
- The priority of RST signals are $7.5 > 6.5 > 5.5$.

TRAP: Non-maskable interrupt having the highest priority.

HOLD:

It indicates that a peripheral device is requesting for the use of system address bus or data bus.

When a hold signal is received by the processor, it completes its current machine cycle and releases the system address bus and data bus.

During this period, the internal processing (execution) of the processor will continue.

HLD A: It indicates that the hold signal is received by the processor. It stands for hold acknowledgement.

READY :

It is used by the μp to check whether, the peripheral is ready to transfer data or not.

If the ready pin is high \Rightarrow Peripheral is ready to transmit or receive data.

If it is low \Rightarrow Then the processor has to wait till the ready pin goes high.

STD : Serial I/P data

- It is a dataline for serial I/P. The data on these line is loaded on the 7th bit of the accumulator by getting an RIM instrucⁿ.

SOD : Serial O/P data

- It is a dataline for serial O/P. The 7th bit of the accumulator will transfer to the SOD line by getting a SIM instrucⁿ.

CLK : It is a triggering pulse. Any triggering pulse can be applied through this pin.

RESET IN : It resets the program counter to zero.

The CPU is held in reset condition as long as reset is applied.

RESET OUT : It indicates that the CPU is in being reset condⁿ.

Lecture notes in

VARIOUS ADDRESSING MODES: ~~AD~~

1. Direct addressing mode
2. Register addressing mode
3. Register direct addressing mode
4. Register indirect mode
5. Immediate addressing mode
6. Implicit addressing mode

1. Direct Addressing Mode :

When data is in the memory, that must have a 16 bit address, if that address is given directly in the instruction then that is called direct addressing mode.

** The different technique used to specify the operand in the instruction is called addressing mode.

eg: LDA 6000H
STA 5000H
LHLD 6000H

2. Register addressing mode :

In this mode the operand is specified through register.

eg: MOV A, B
ADD B
ANA C
XRA L

3. Register indirect addressing mode :

In this mode the address of the data is indirectly specified.

eg: LDAX B
STAX D
MOV A, M
ADD M

4. Implicit addressing mode :

In this mode the content of accumulator is assumed as the operand.

eg: CMA, RAL, RLC, RAR.

5. Immediate addressing mode :

The operand is immediately supplied to the instruction.

eg: MVI A, 60H
ADI FFH
ANI 6AH

Instruction Cycle:

- Instruction is a command to CPU to perform a given opⁿ (specified on the given data).

Program: The sets of instrucⁿ given by the programmer to perform a specific task is known as a program.

- The necessary steps that the CPU carries out to fetch an instrucⁿ and take the necessary data from the memory and execute it constitute an instrucⁿ cycle.
- The instrucⁿ cycle consists of a fetch cycle and an execuⁿ cycle.

$$IC = FC + EC$$

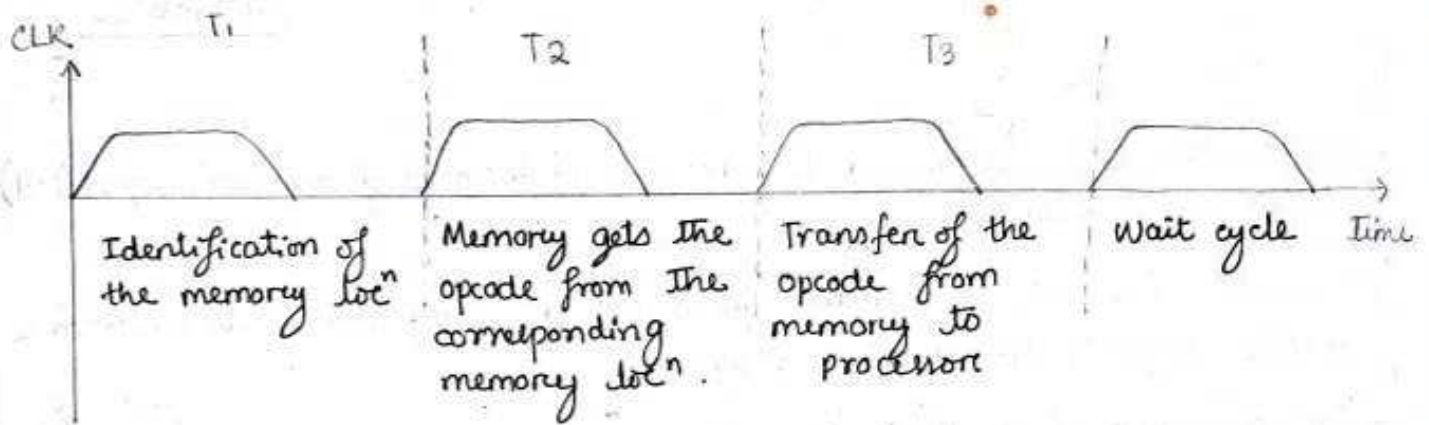
IC = Instrucⁿ cycle

FC = Fetch cycle

EC = Execution cycle.

Fetch: The necessary steps which are carried out to fetch an instrucⁿ from memory constitute a fetch cycle.

- Generally the 1st byte of any instrucⁿ is the opcode and the remaining bytes may be a 8-bit data/address or 16 bit data/address (operand).
- In the beginning of the fetch cycle, the content of the program counter, which is the address of the memory locⁿ where opcode is available is sent to the processor.
- The memory places the opcode on the data bus and send it to the processor.
- The processor stores this opcode in the IRs from where it is further fed for decoding and execution.
- The entire process is known as fetching i.e. reading of opcode (not data) opcode (hexcode or machine code)
- The fetching opⁿ takes 3 clk-pulses or 3 T-states but one more extra T-state is available if the processor or the memory is slower by nature. This extra cycle is known as wait cycle.



Execution: After the opcode is being fetched from the memory, it goes to the IRs.

- From the IRs it goes to the decoding unit. After the instructⁿ being decoded, execuⁿ takes place.

- During the time of execuⁿ, if the operand is available in the general purpose register (GPR) then the execuⁿ takes place immediately.

- The time taken to decode and execute is 1 clk cycle or 1 T-state if the operand is in GPR.

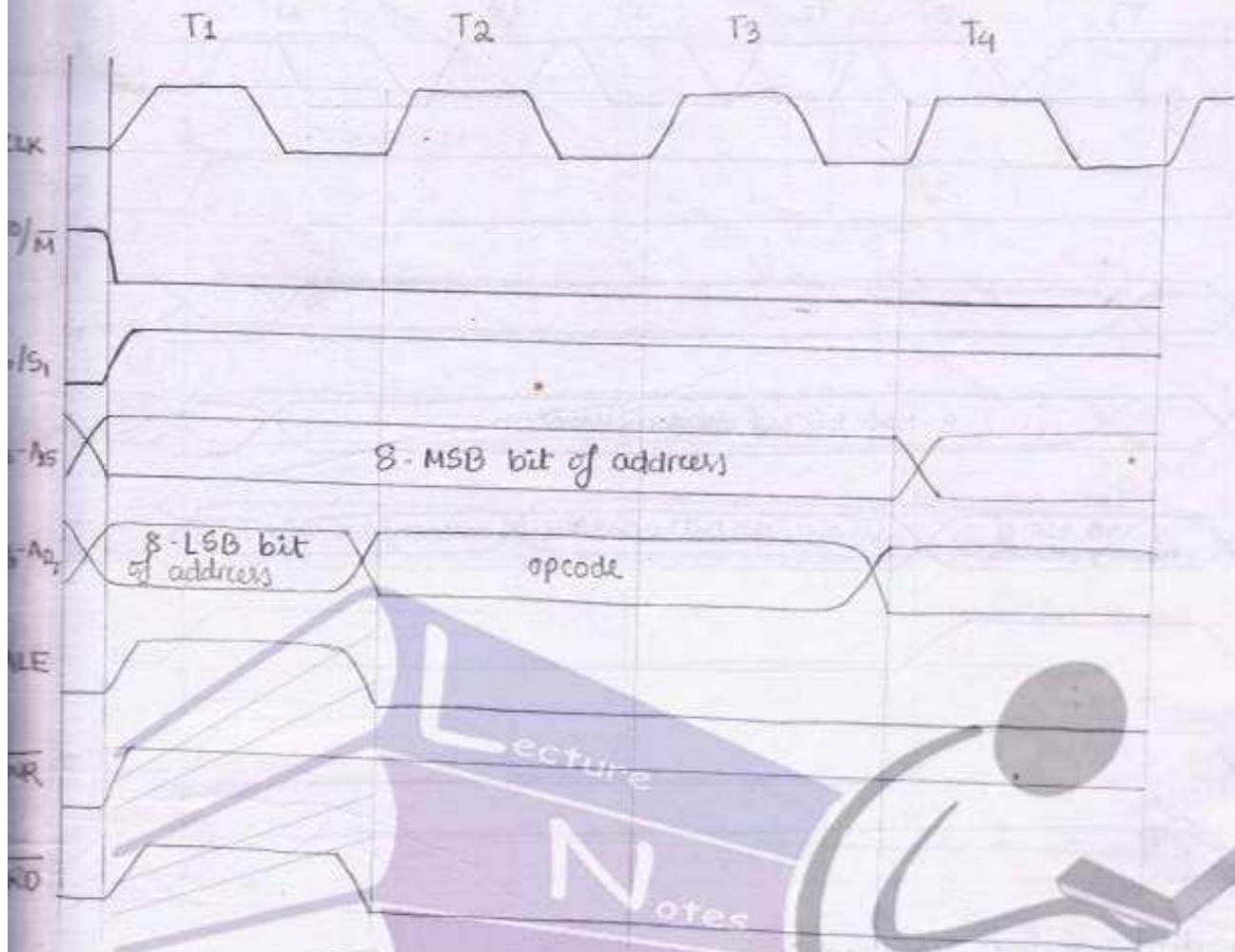
- If the operand is in certain memory / I/O locⁿ then the processor has to perform certain oprⁿ like MR / MW / I/O R / I/O W.

TIMING DIAGRAM: It is a graphical ^{representation} of certain pins, whose status get changed for various processor oprⁿ during each T-states.

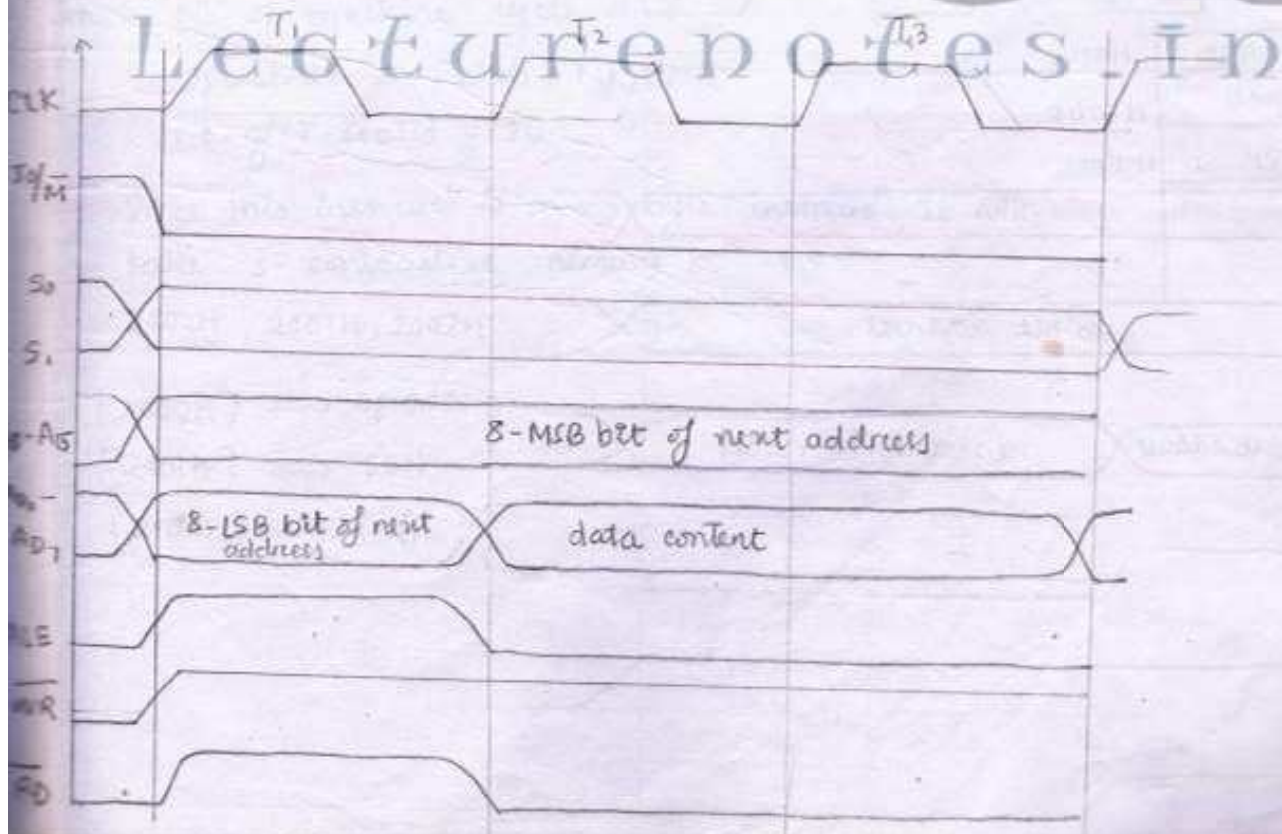
The pins are,

1. I/O / \overline{M}
2. CLK
3. S₀, S₁
4. Address bus A₈ - A₁₅
5. Data bus A_{D0} - A_{D7}
6. \overline{RD}
7. \overline{WR}
8. ALE

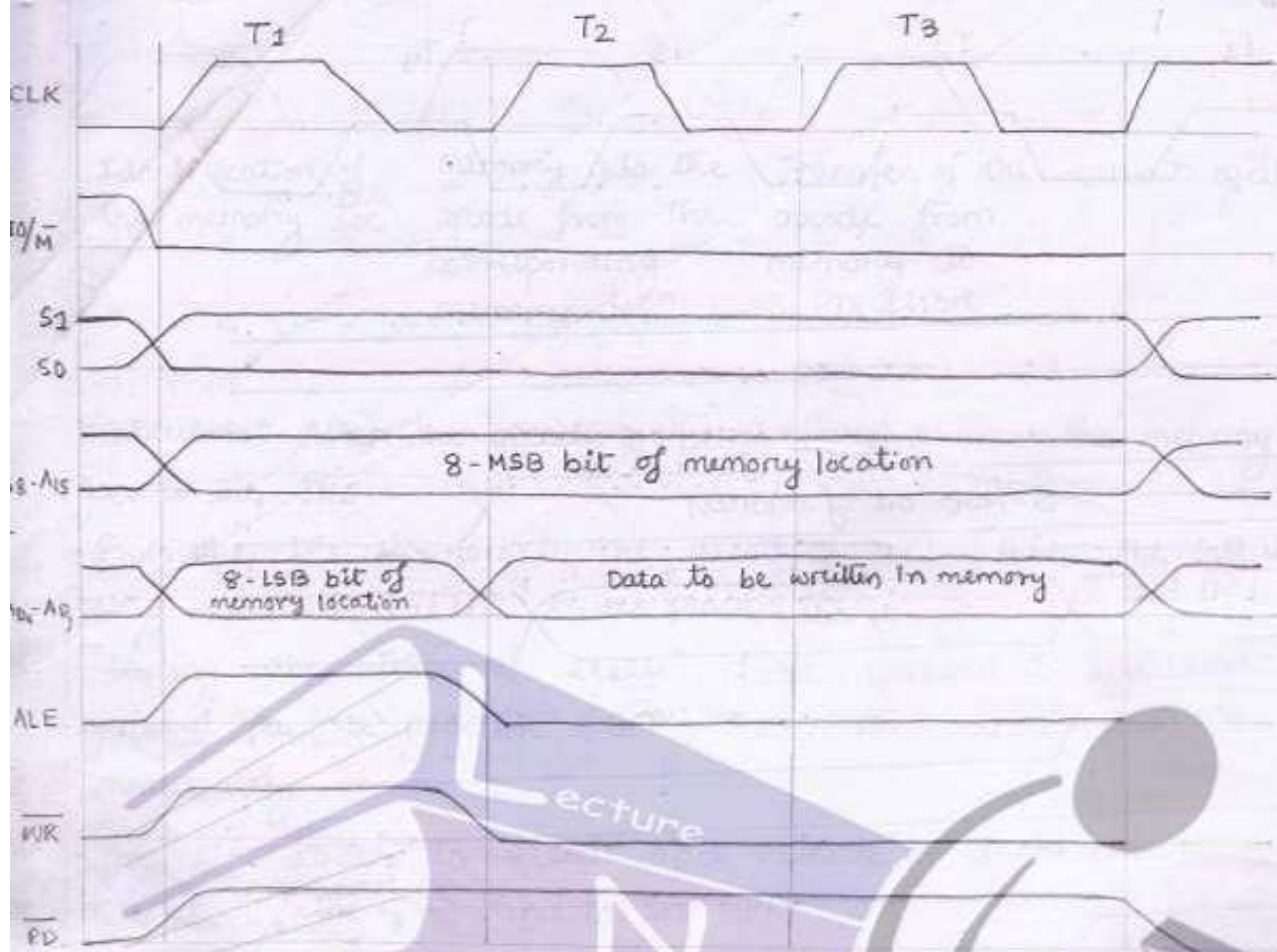
Timing Diagram for fetch:



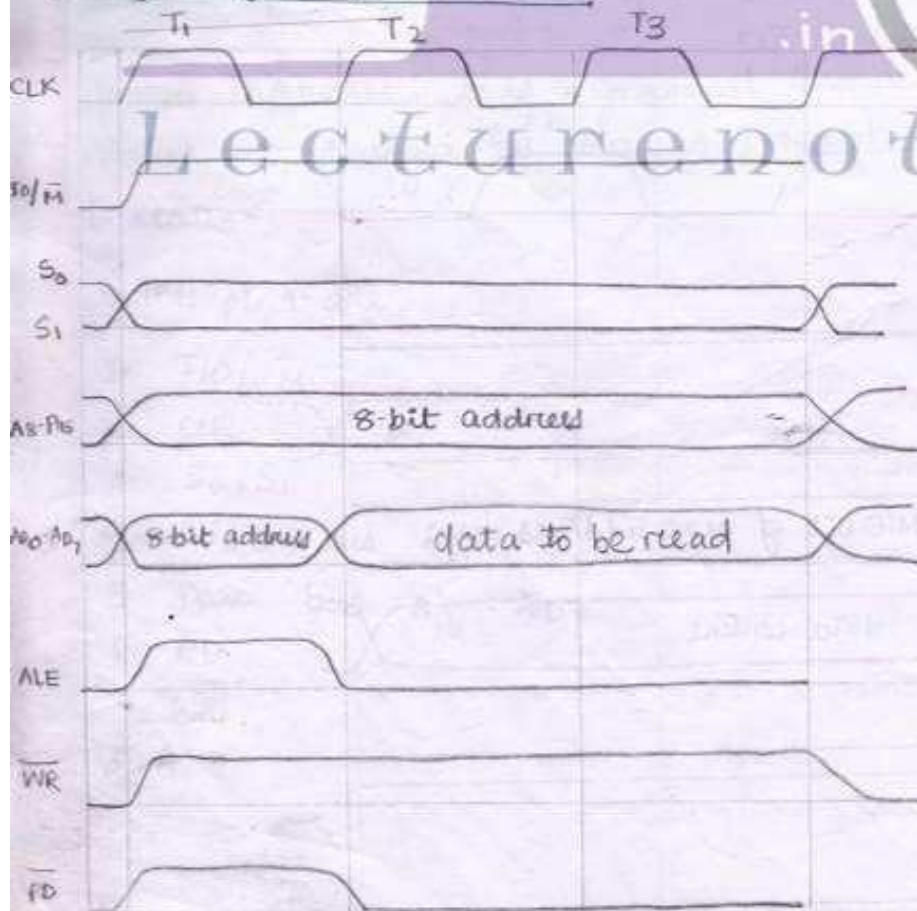
Timing Diagram for MR:



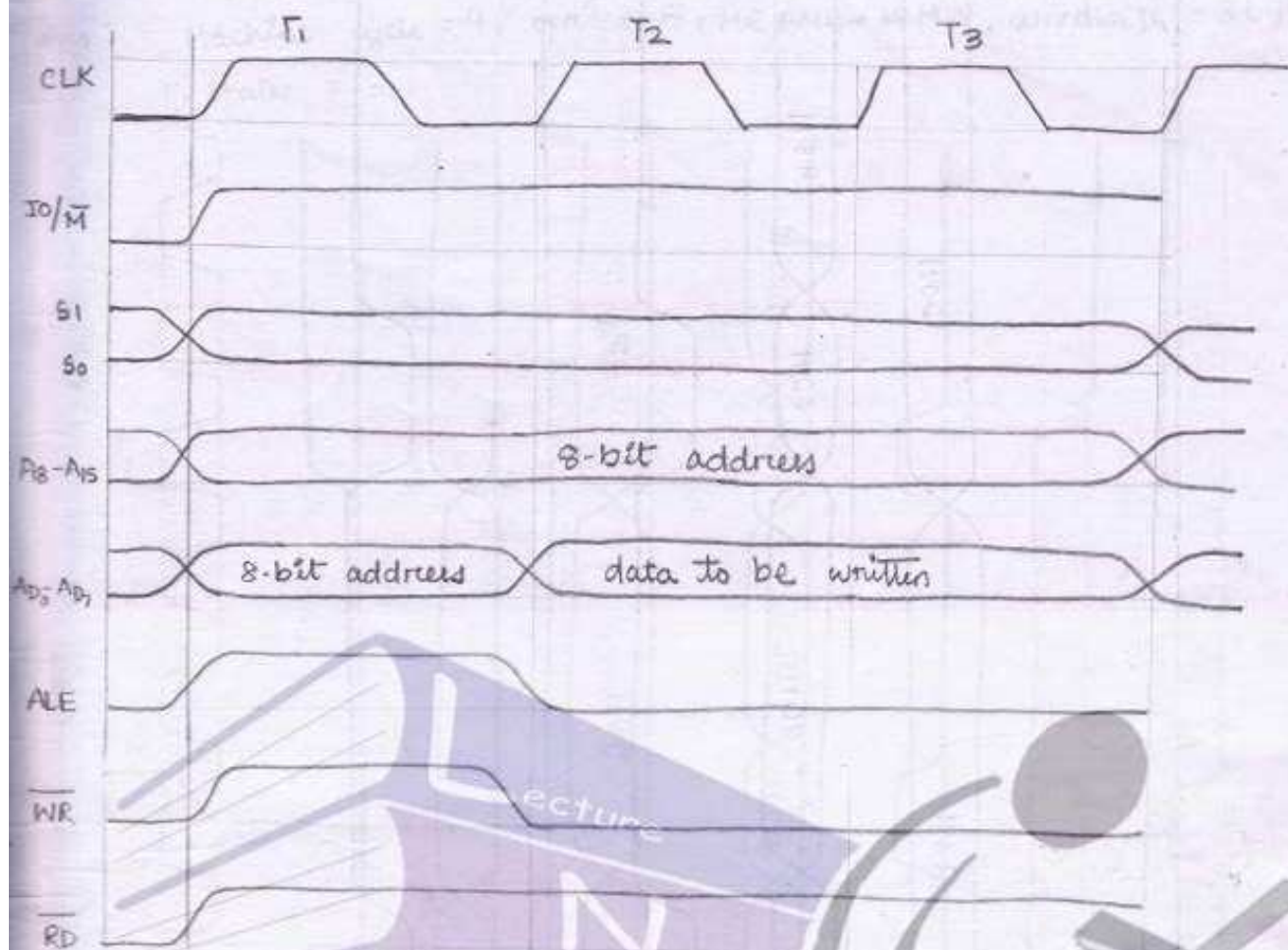
Timing diagram for MW:



Timing diagram for I/O Read:



Timing diagram for I/O write:



eg: 2000H LXI 4500H. Draw the timing diagram.

13th Feb'10

Ans: No. of machine cycle = 3
 operation = Fetch + MR + MR
 No. of T-states = 10

Since this instrucⁿ is a 3-byte instrucⁿ, it will take 3 consecutive memory locⁿ i.e.

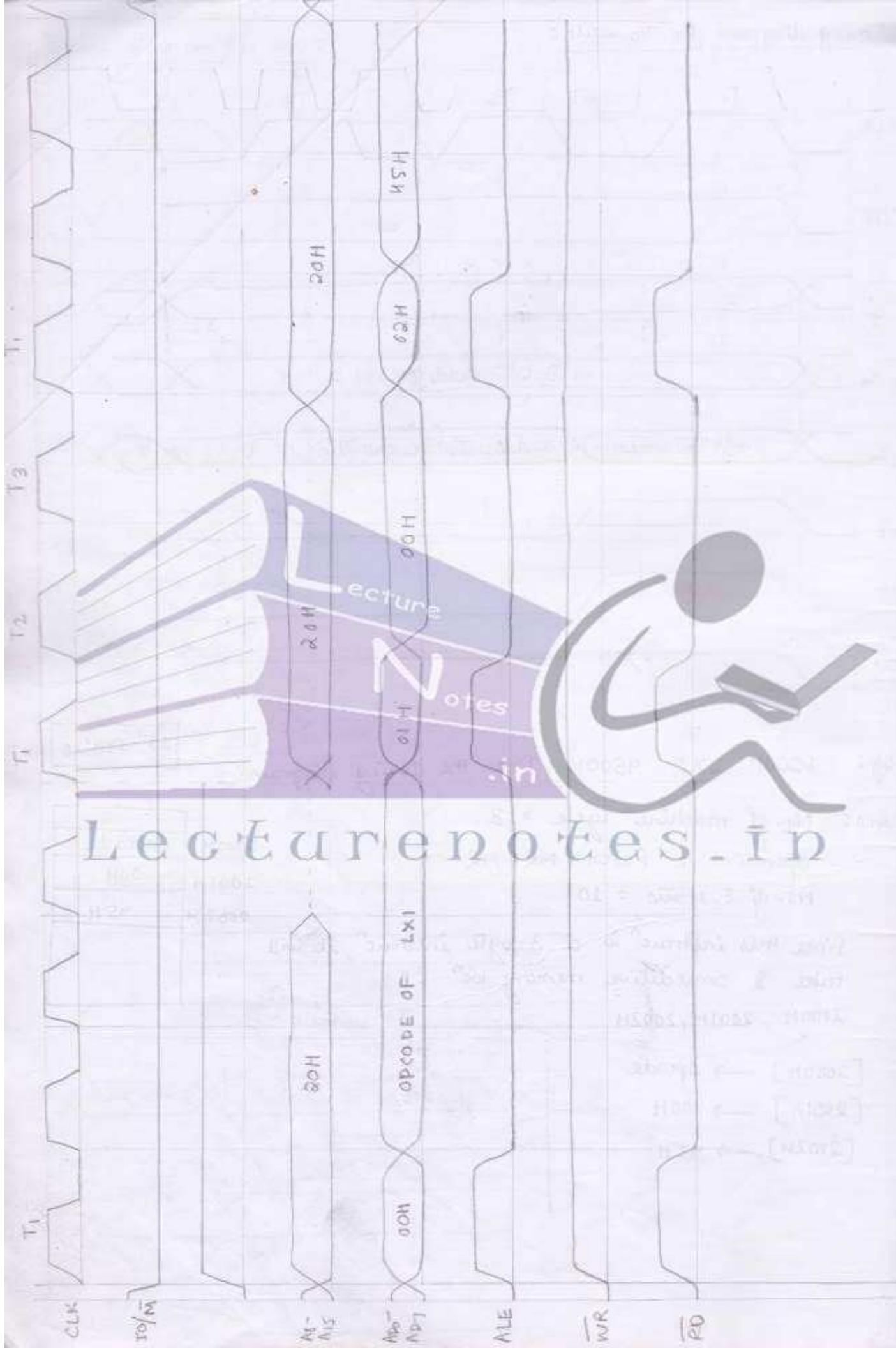
2000H, 2001H, 2002H

[2000H] → opcode

[2001H] → 00H

[2002H] → 45H

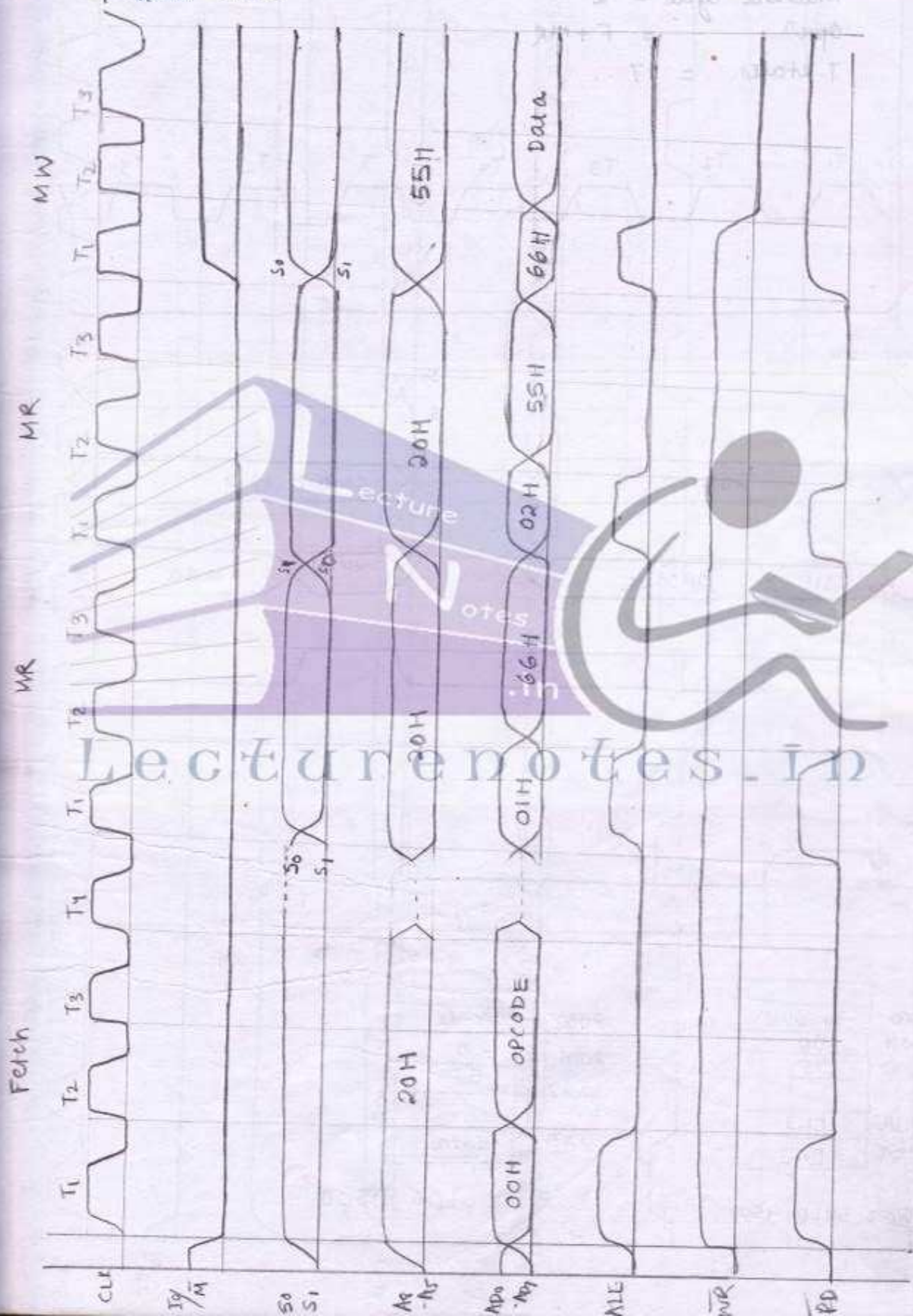
2000H	opcode
2001H	00H
2002H	45H



Q. ~~2000H~~ 2000H STA 5566H

7 M, A VDM -8

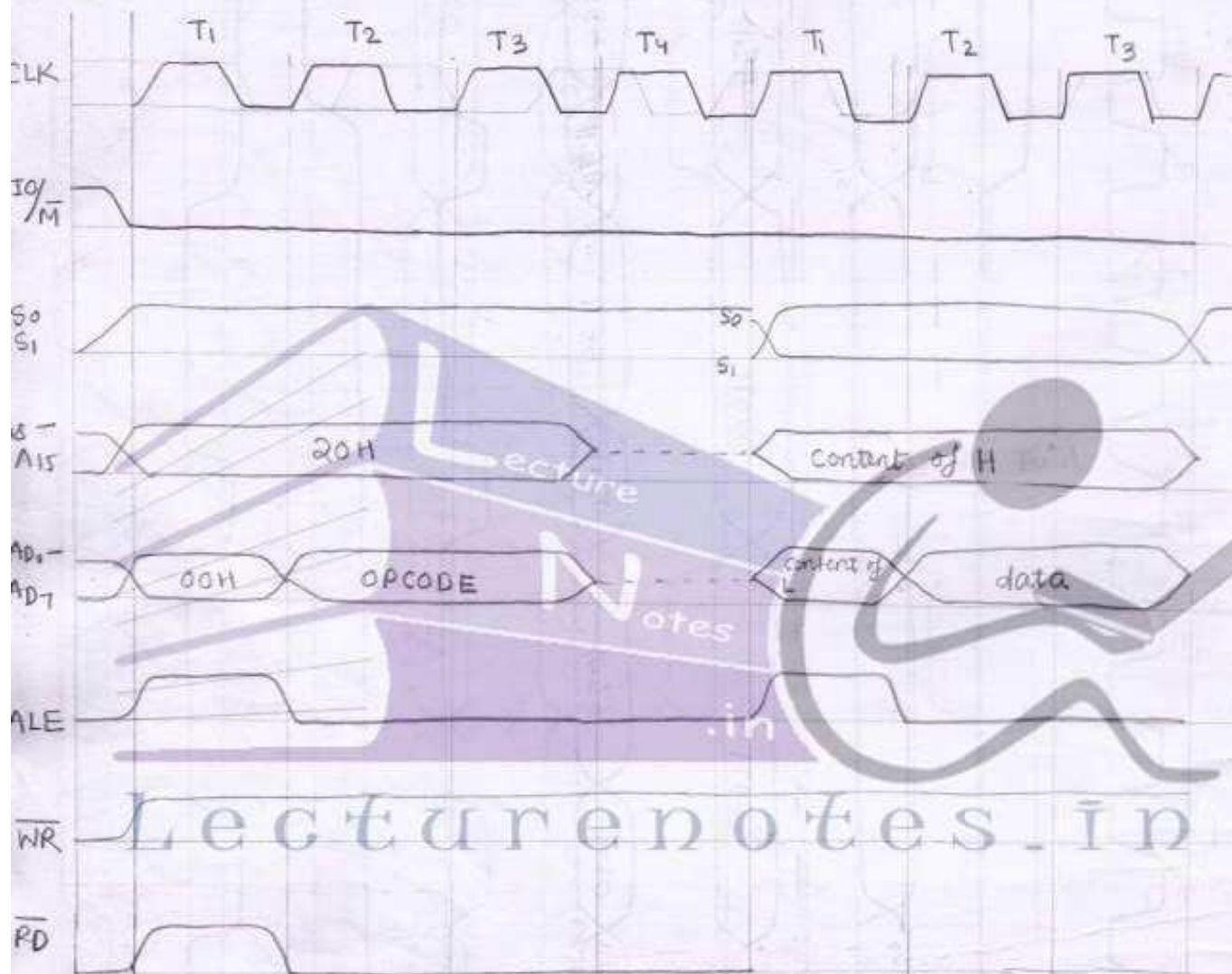
Ans: Machine cycle = 4, $opn^1 = F + MR + MR + MW$, wordsize = 3 byte
T-states = 13.



Q- MOV A, M :

Ans : 2000H MOV A, M 4302H

Machine cycle = 2
 Operⁿ = F + MR
 T-states = 7



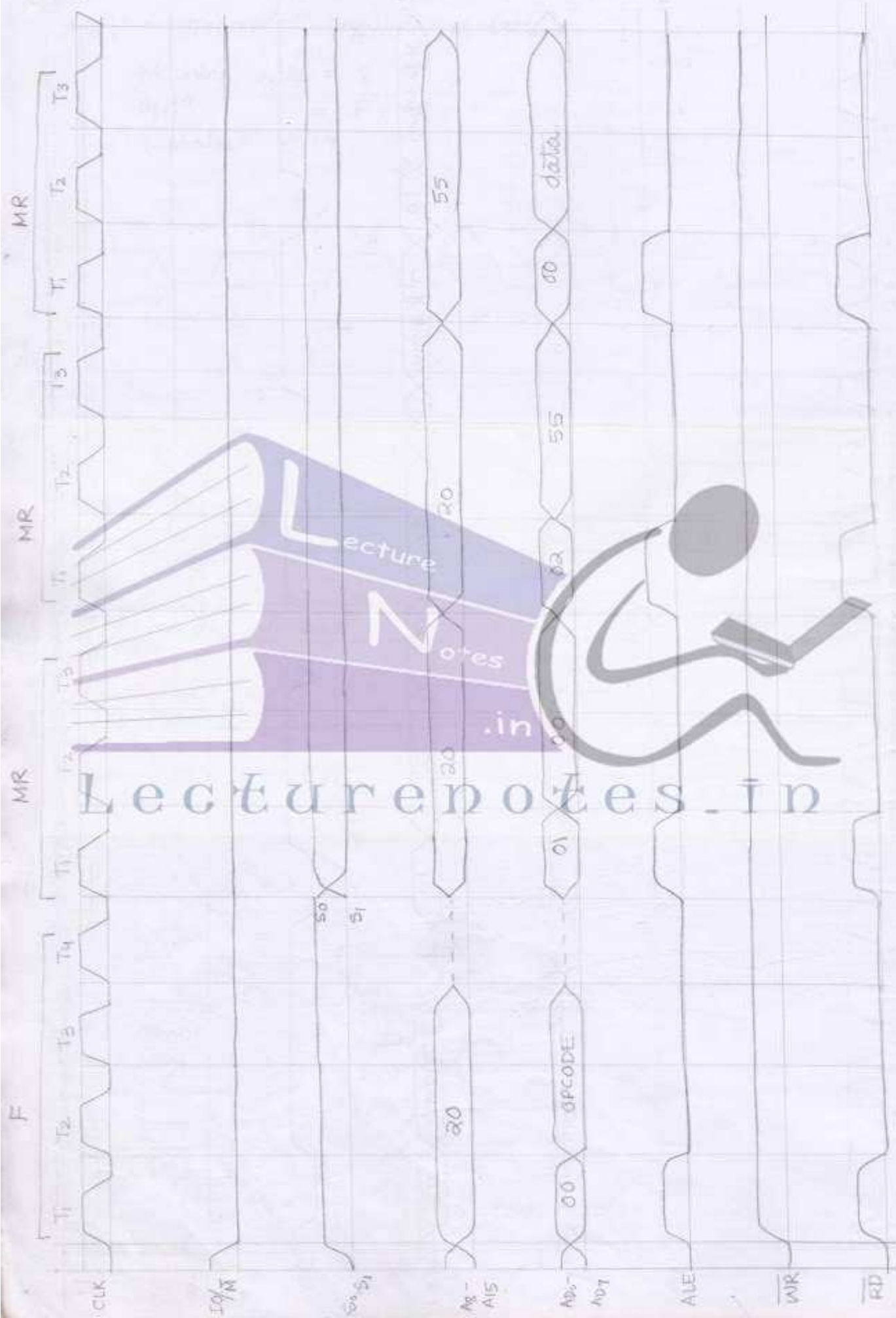
2000	OPCODE
2001	00
2002	45
4500	[L]
4501	[H]

2000 SHLD 4500

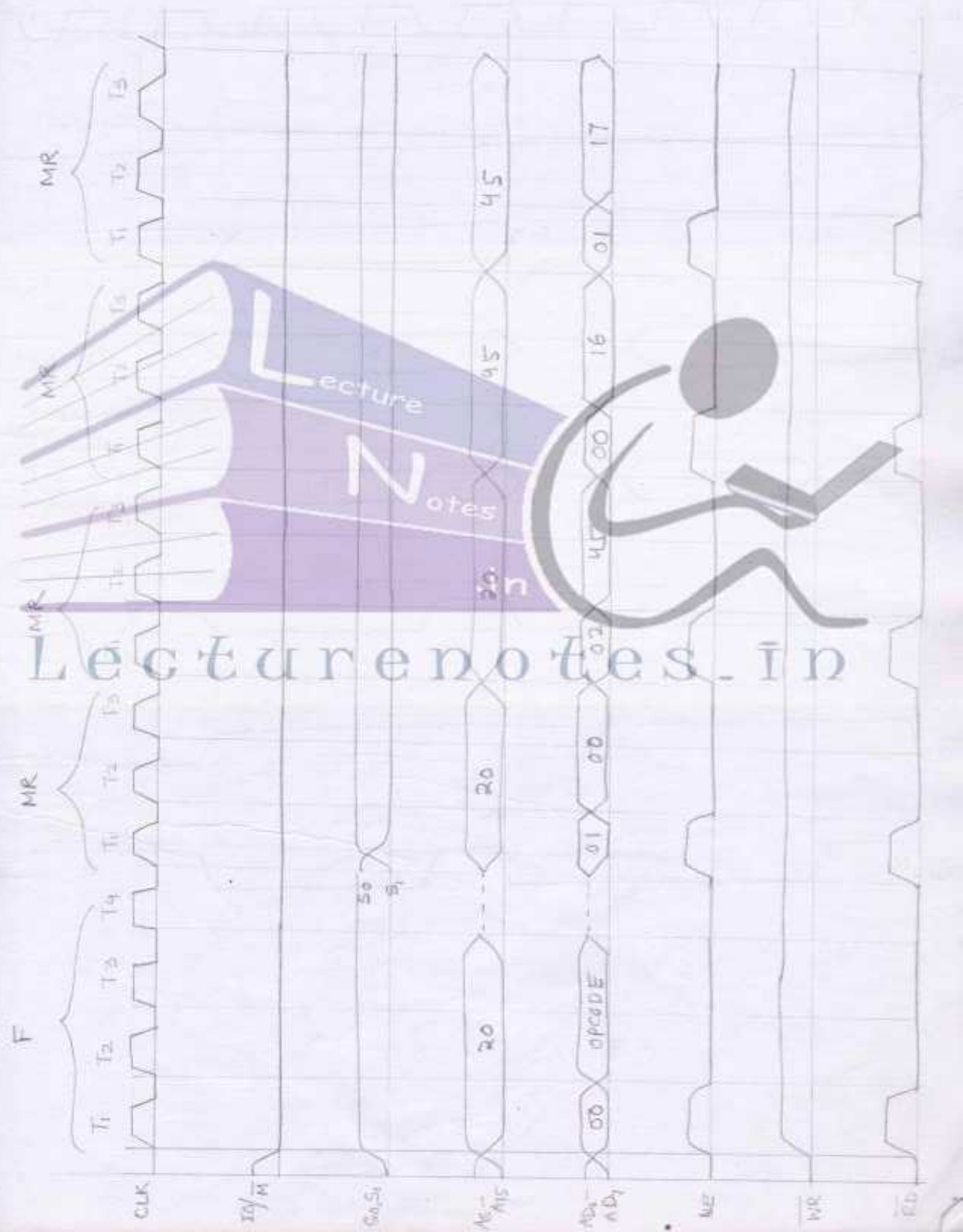
2000	opcode
2001	00
2002	55
5500	data

2000, LDA 5500

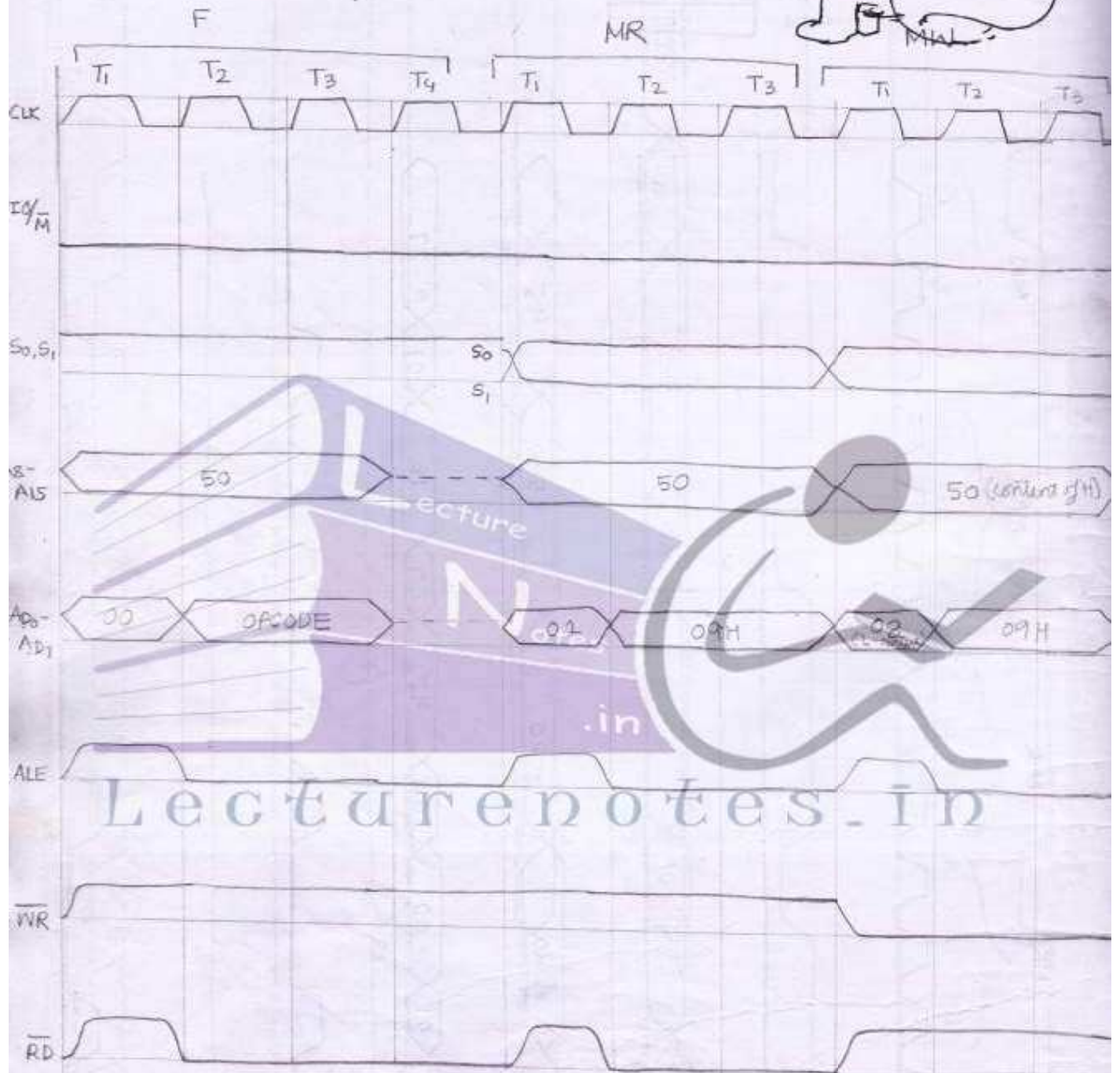
4. LDA



2000	OPCODE
2001	00
2002	45
4500	16
4501	17

$$16 \rightarrow [L]$$
$$17 \rightarrow [H]$$


6- 5000 MVIH,09H :



T. XCHG :

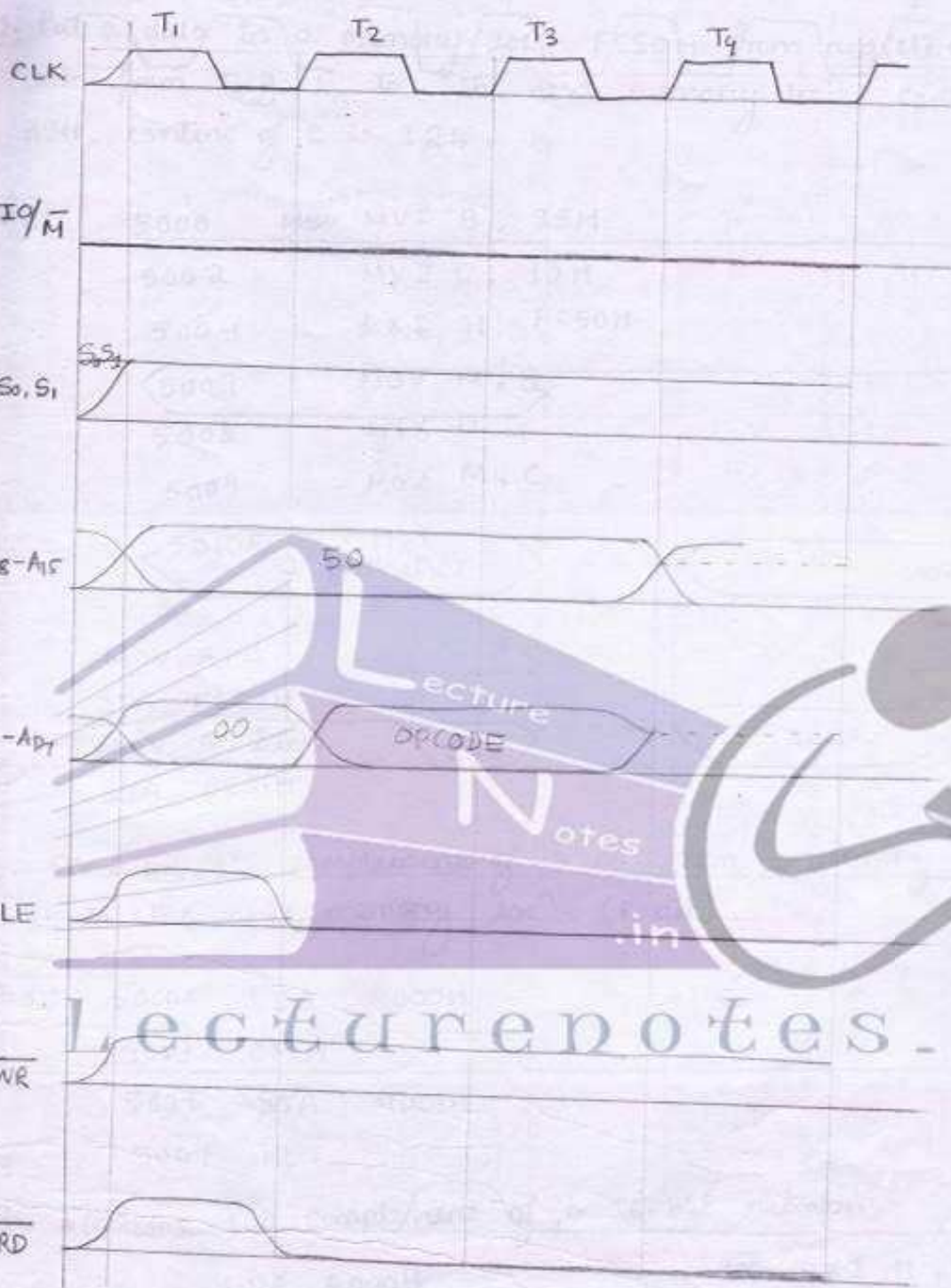
5000 XCHG

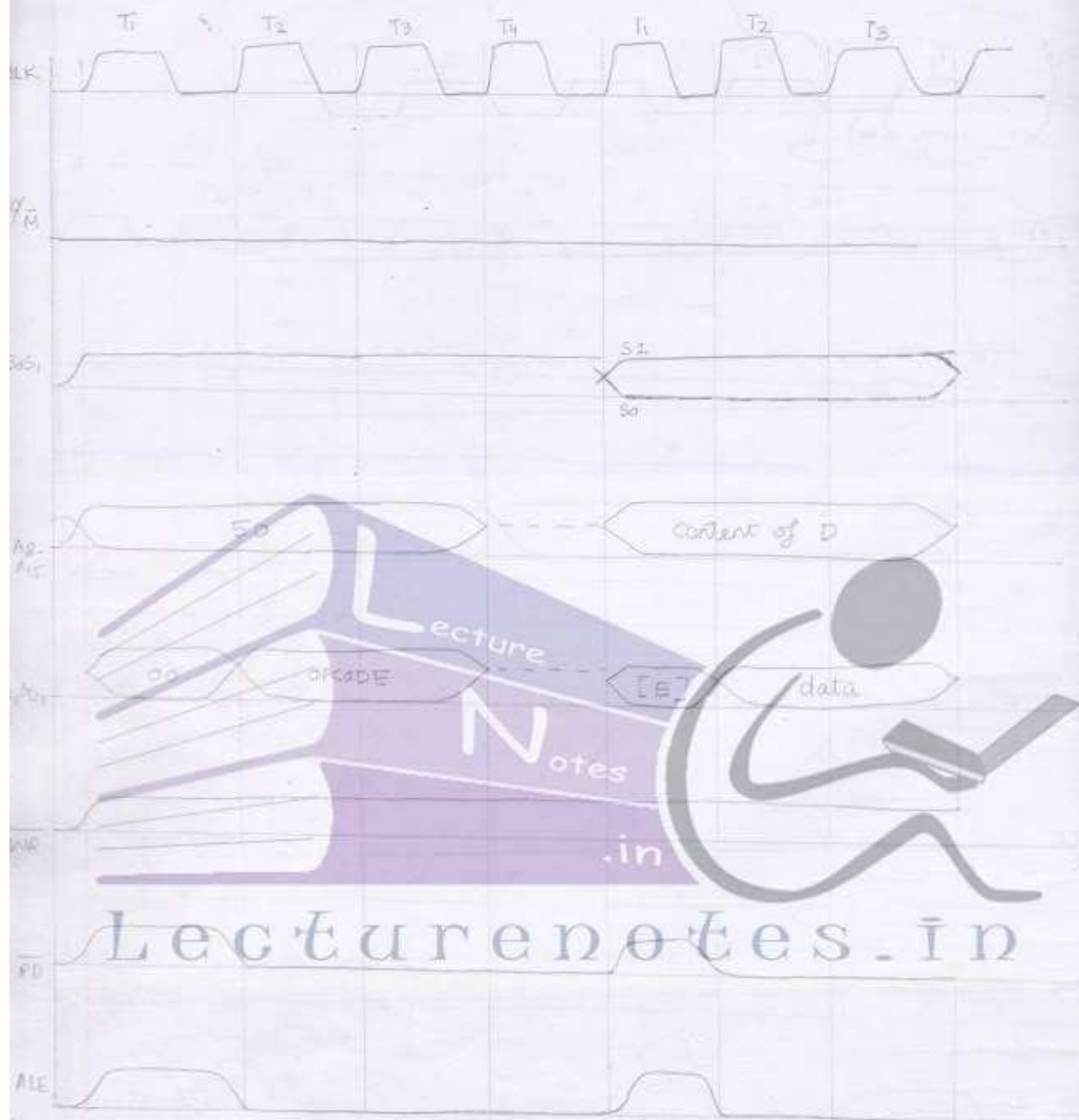
DE :

XCHG

5000

5000





8085 PROGRAMMING

15th Feb '10

1. Put a data in a memory locⁿ FC50H from register B. Put another data from reg. C to the next memory locⁿ. Content of B is 35H, content of C is 12H.

Ans:

5000	MOV	MVI B, 35H
5002		MVI C, 12H
5004		LXI H, FC50H
5007		MOV M, B
5008		INX H
5009		MOV M, C
5010A		HLT

or

	MOV	A, B
	STA	FC50H
	MOV	A, C
	STA	FC51H

2. Find the 1's complement of a no. from a memory locⁿ and store it in the next memory locⁿ. (8-bit)

Ans:

5000	LDA	4000H
5003	CMA	
5004	STA	4001H
5007	HLT	

3. Find the 1's complement of a 16-bit number.

LDA 4000H

CMA

STA 4002H

LDA 4001H

CMA

STA 4003H

HLT

or

LXI H, 4000H

MOV A, M

CMA

STA 4002H

INX H

MOV A, M

CMA

STA 4003H

HLT

3. Find the 2's complement of any number.

```
LXI H, 4000H
MOV A, M
CMA
ADD INR A
STA 4002H
INX H
MOV A, M
CMA
INR A
STA 4003H
HLT
```

```
LXI H, 4000H
MOV A, M
CMA
INR A
JNE STA 5000H
INX H
JC *
INX H
MOV A, M
CMA
INR A
STA 5001H
HLT
```

eg: $0000\ 0000 \rightarrow 1111\ 1111 + 1 \rightarrow 1,0000\ 0000$
 1's

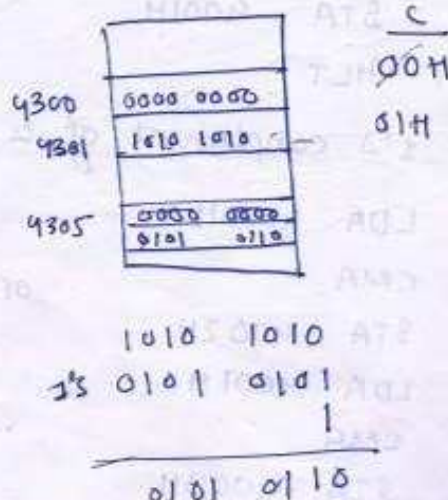
$0000\ 0000$
 2's

for 8-bit :

```
LDA 4500H
CMA
INR A
STA 4501H
HLT
```

Lecture notes in

```
LXI H, 4300H
MVI C, 06H
MOV A, M
CMA
ADD A, 01H INR A
JNC **
INRC
** STA 4305H
INX H
MOV A, M
CMA
ADD A, C
STA 4306H
HLT
```



4. Write a program to shift a data (8-bit) by 1 bit left.

```
LDA 4000H
ADD A
STA 4001H
```

5. Write a program to find the gray code of a 8-bit number.

6. Write a program for addⁿ of a series of numbers; (num = 16 bit).

7. Find the largest no. from a group of data array.

8. Arrange data in ascending order.

9. Multiplication, division.

PROGRAMS:

1. BINARY TO GRAY:

```
LXI H, 4200H
MOV A, M
MOV B, A
RAR
XRA B
STA 4300H
HLT
```

2. LARGEST NUMBER:

```
4100 LXI H, 4200H
4103 MOV C, M
4104 MVI A, 00H
4106 INX H
4107 CMP M
4108 JNC 410C
410B MOV A, M
410C INX H
410D DCR C
410E JNZ 4107
4111 STA 4300H
4114 HLT
```

3. MULTIPLICATION:

```
LXI H, 4200H
MOV B, M
MVI C, 00H
MVI A, 00H
INX H
** ADD M
JNC *
INR C
* DCR B
JNZ **
STA 4202H
MOV A, C
STA 4203H
HLT
```

4. DIVISION:

```
MVI D, 00H
LDA 4200H
MOV B, A
LDA 4201H
** SUB B
JC *
INR D
JMP **
* ADD B
STA 4300H
MOV A, D
STA 4301H
HLT
```

Handwritten division examples:

4200H / 4201H = 1

4200H / 4201H = 1

4200H / 4201H = 1

4200H / 4201H = 1

4200H / 4201H = 1

ADDITION OF A SERIES OF NUMBERS

```

LXI H, 4200H
MOV C, M
MVI A, 00H
MVI B, 00H

INX H
** ADD M
JNC *
INR B
* INX H
DCR C
JNZ **
STA 4300H
HLT
    
```

ARRANGE DATAS IN ASCENDING ORDER:

```

LXI H, 4200H
MOV C, M
MVI A, 00H
MVI B, 00H

INX H
MOV A, M
** CMP M
JNC *
INR B
STA 4250H
INR B
    
```

* INX H

DCR C

JNZ **

STA 4300H

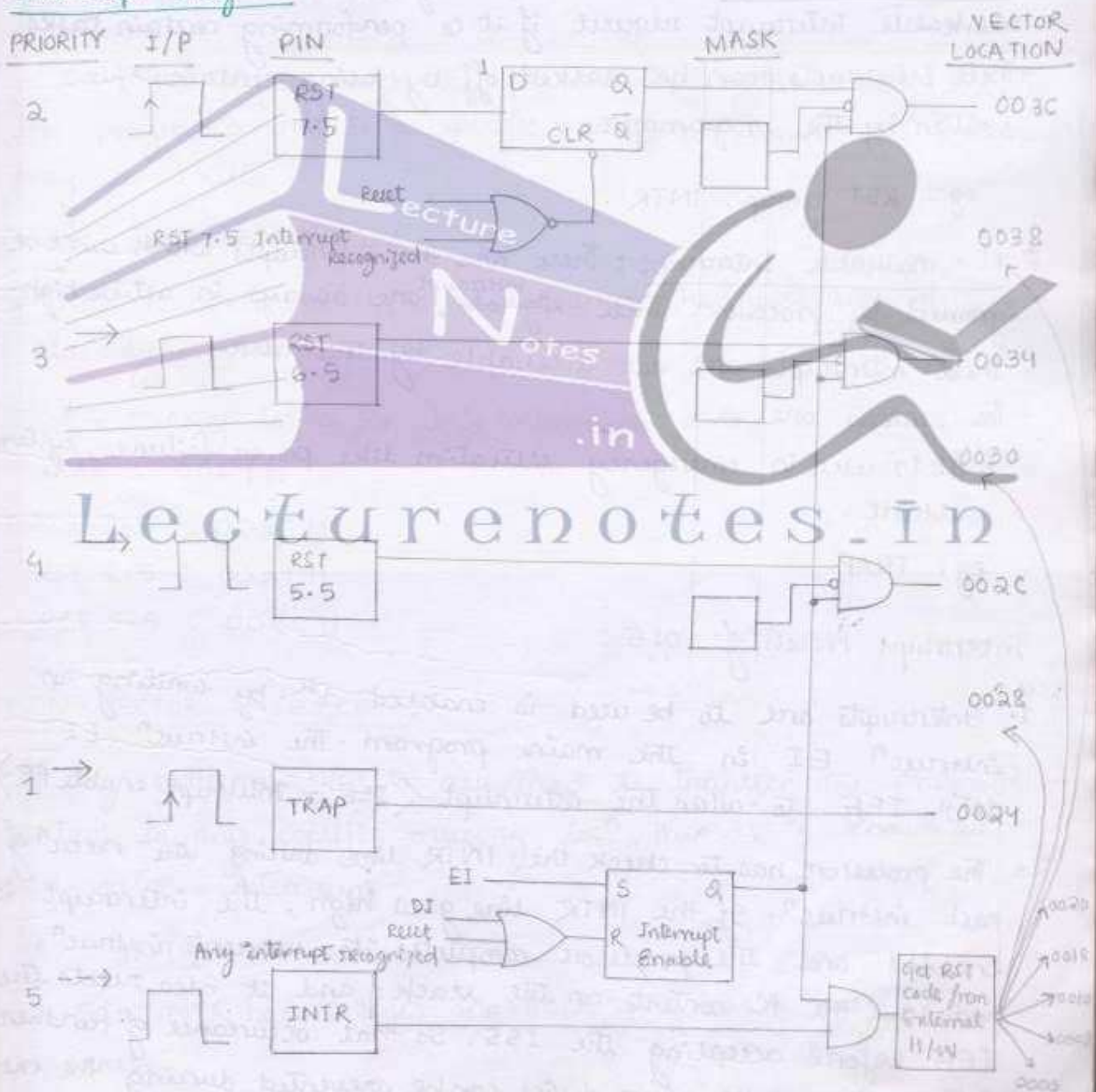
	H	M	A	B	D	C
		5	5	9	-	1
4200		5	5	9	-	1
4201	2	5	5	9	2	8
4200	2	5	5	9	4	7
4201	5	5	5	9	4	7
4202	1	5	5	9	4	7
4201	4	5	5	9	4	7
4202		5	5	9	4	7

INTERRUPT :

Interrupts are the signals that cause the CPU to sustain what it is doing and transfer a special program called interrupt service subroutine (ISS) or Interrupt handler.

It is responsible for

- cause of interrupt
- service " "
- transfer back the control to the main program.

Interrupt Diagram :

Classification of Interrupt :

1. Hardware Interrupt : Interrupt caused by the I/O device.
2. Software Interrupt : Interrupts caused by the abnormal internal condⁿ which is because of any special instrucⁿ is called a s/w interrupt.

Based on priority interrupts are of 2 types.

- (i) Maskable (ii) Non-maskable.

1. Maskable Interrupt : Processor can ignore or make delay to a maskable interrupt request if it is performing certain tasks.

- These interrupts can be masked off by using certain s/w written by the programmer.

eg: RST group, INTR.

2. Non-maskable Interrupt : These are the interrupts which can't be ignored by processor. These ^{interrupt} signals are always in active high.

- These interrupts are not accessible by the user.

- The reasons are :-

• It is used in emergency situation like power failure, system restart.

eg: TRAP.

Interrupt Process of 8085 :

1. Interrupts are to be used is enabled 1st. by writing an instrucⁿ EI in the main program. The instrucⁿ EI sets IEF to allow the interrupt <IEF - Interrupt enable FF>
2. The processor has to check the INTR line during the execuⁿ of each instrucⁿ. If the INTR line goes high, the interrupt is enabled. and the processor completes its current instrucⁿ. and save the PC content on the stack. and it also resets the IEF before accepting the ISS. So that occurrence of further interrupts can by other device can be prevented during the execuⁿ of any ISS.

3. It also sends a signal \overline{INTA} i.e. interrupt acknowledgment. A call instructⁿ is executed so that the PC will get the address where ISS is present.

4. The ISS task is performed. The ISS is a subroutine present at a specified memory locⁿ.

5. In the ISS we have to put an EI instructⁿ again to enable the interrupt and a RET so that the control will come back to its initial posⁿ.

VECTOR INTERRUPT:

18th Feb'10

- An interrupt for which the internal h/w automatically transfer the program control to a specific memory locⁿ where the interrupt program exists.

eg: TRAP, RST groups.

- Hence no external h/w is required. The necessary h/w is already provided by 8085.

- The memory locⁿs for this interrupt are,

TRAP : 0024 H

RST 7.5 : 003CH

RST 6.5 : 0034 H

RST 5.5 : 002CH

NON-VECTOR INTERRUPT:

- If any external h/w is required to transfer the program control to any specific memory locⁿ, then it is known as non-vector interrupt.

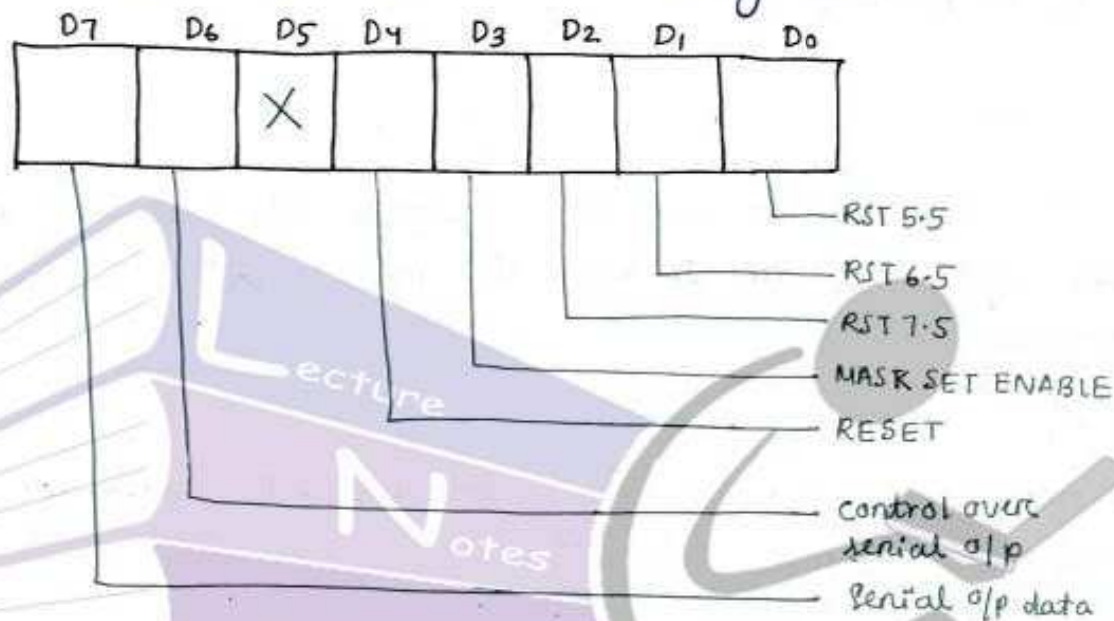
eg: INTR

This INTR has 8-lines and each line has a specific address.

SIM: Set Interrupt Mask

- 1 byte instrucⁿ
- Can be used for 3 different funcⁿ,

- (1) The execuⁿ of SIM enables or disables the interrupts on the basis of the bit content of accumulator.
- (2) It is used to set 'mask' for RST 5.5, 6.5, 7.5 interrupt
- (3) It creates a path for serial o/p of data.



(a) D_0, D_1, D_2 : These bits are assigned for RST 5.5, 6.5, 7.5 respectively.

- If any of these bits is zero the corresponding interrupt is enable.

- If any of these bits is set to 1, the corresponding interrupt is disabled.

(b) D_3 :

- It is used to mask off the b or mask on the D_0, D_1, D_2 bits
- If $D_3 = 1$, all interrupts are effective.
- $D_3 = 0$, all interrupts are disabled.

(c) D_4 :

- It has some additional control over RST 7.5.
- If $D_4 = 0$, RST 7.5 is enabled.
- $D_4 = 1$, RST 7.5 is disabled.

D₅: Don't care bit.

D₆, D₇: It is used for serial O/P of data.

- If the SIM instructⁿ is executed, the content of 7th bit of accumulator is o/p on a SOD line of the processor provided D₆ is enabled.

If D₆ = 1, then D₇ bit can be placed on SOD line.

Q. MVI A, 18

SIM

HLT

What will happen to all the RST interrupts if these 3 lines are executed.

Ans:

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
0	0	0	1	1	0	0	0

D₀ = 0, RST 5.5 is enabled

D₁ = 0, RST 6.5 is enabled

D₂ = 0, but D₄ = 1 \Rightarrow RST 7.5 is disabled.

D₆ = 0 so SOD line disable

22nd Feb '10

RIM: Read Interrupt Mask

- This is a 1-byte instruction.

- This interrupt indicates the current status of interrupt by reading the content of accumulator.

- This instructⁿ identifies the pending interrupt bits.

- It receives serial I/P of data.

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
SID	RST 7.5	RST 6.5	RST 5.5	IE	RST 7.5	RST 6.5	RST 5.5

- a) D₀, D₁, D₂ : It is used for mask the interrupts RST 7.5, 6.5, 5.5.
 It can be masked on when the bits are 1.
- b) D₃ : It is used for enable the interrupt, when this bit is 1 all the interrupts are enabled.
- c) D₄, D₅, D₆ : These bits are used for pending interrupts.
 - If one of the bit is 1, the corresponding RST interrupt is in pending state. i.e. if 15.5 is '1' \Rightarrow RST 5.5 is pending.
- d) D₇ : This bit shows serial I/p data bits i.e. any peripheral device ^{can} sends a serial data bit if D₇ bit = 1.

Delay Programming :

		how many times the instruction is executing	T-states
(1)	MVI C, 10H	1	7
	* DCR C	16	64
	JNZ *	16	15 \times 10 + 1 \times 7
	RET	1	10
			<hr/> 238

Find out the total no. of T-states, for the above subroutine.

Lecture notes in

(2)	LXI D, FFFF	1	10
	* DCX D	65535	262140 393210
	MOV A, D	65535	262140
	ORA E	65535	262140
	JNZ *	65535	65534 \times 10 + 7 = 655347
	RET	1	10
			<hr/> 1572857

Time period / T-states = 10 msec

Time = ~~15.728~~ sec.

Time = 4.369 hr.

$$10 + N(6 + 4 + 4) + (N-1) \times 10 + 1 \times 7 + 10$$

- Memory of microprocessor stores binary instructions for and data for the processor.

- There are 2 types of memory.

(i) Main memory (RAM/ROM)

(ii) Secondary memory.

Memory is made up of certain registers which is also known as memory location. Each register consists of certain FFs which is known as memory cell. and each FF is capable of storing a single bit of a data.

66

Memory interfacing is a technique which synchronizes the speed of memory with the μP so that there will not be any loss of data.

- Memory interfacing is of 2 types

(a) Memory interfacing

(b) I/O interfacing

(a) MEMORY INTERFACING:

The primary funcⁿ of memory interfacing is that the processor should be able to read from or write into a given register of a memory chip.

Other funcⁿs are,

→ The μP should be able to select the chip first.

→ It identifies the particular register / memory locⁿ.

→ It assigns the starting address and ending address of each memory chip.

Generally the μP is having 16 address lines. Out of these 16 address lines, some are used to select the memory chip and the remaining address lines are used to identify a particular register.

1. Determine the amount of memory space to be interfaced.
2. Find out the no. of address lines i.e. required to identify a particular memory locⁿ. If some address lines are unused because of small size of the memory chip, then those address lines are decoded to select a particular memory chip.
3. Once the chip is selected, the control signal \overline{RD} and \overline{WR} is generated along with the $I/O/\overline{M}$ signal to enable the appropriate memory locⁿ for reading & writing opⁿ.

Q. How many chips are required to address a memory size 32×1024 bytes if the address line of each chip is 6.

Ans: Memory size = $32 \text{ KB} = 2^{15}$ size, chip memory = $2^6 = 64$ bytes size

$$\frac{32 \times 1024}{64} = 512 = \text{no. of chips reqd.}$$

MEMORY & I/O ADDRESSING:

Intel 8085 uses 16 bit address bus for addressing memory and I/O devices. There are 2 schemes used for addressing for I/O and memory.

- (1) Memory mapped I/O scheme
- (2) I/O mapped I/O scheme.

1. Memory mapped I/O scheme:

In this scheme there is only one address space. Out of which the total address space some addresses are assigned to memory and some are assigned to I/O. The address for I/O device is different from the address which is assigned for memory. In this scheme all the data transfer instrucⁿ of μP can be used both for memory and I/O device. This scheme is suitable for small system. In this mode $I/O/\overline{M}$ is not used to distinguish M and I/O.

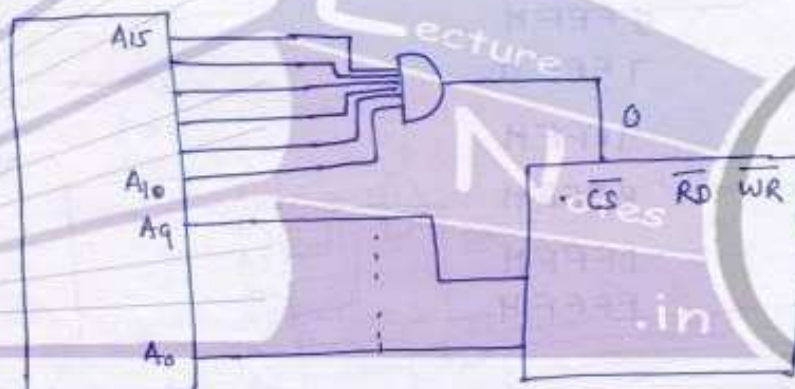
- In this scheme the I/O device is interfaced in the same manner as that of memory device.

2. I/O mapped I/O scheme:

- In this scheme addresses assigned to memory locⁿ can also be assigned to a I/O device.
- In this mode I/O/ \overline{M} signal is used to distinguish whether it is a I/O operⁿ or Memory operⁿ.
- This scheme is used for large system.
- In this scheme IN and OUT instrucⁿs are used.

INTERFACING DESIGN:

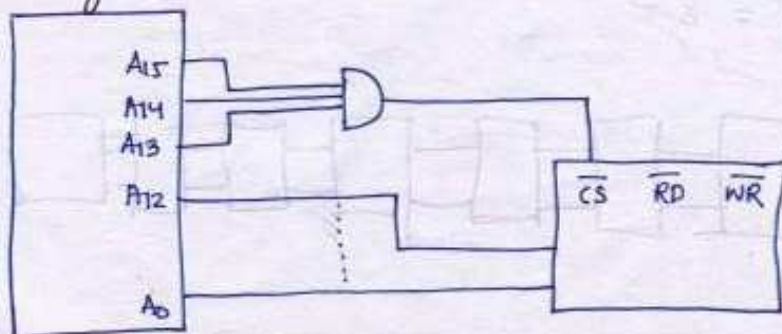
Interface a 1KB RAM to μP .



chip select address						memory location address							
A15	-	-	-	-	A0	A9	-	-	-	-	-	-	A0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	1	1	1

i.e. memory space is from 0000H to 03FFFH.

Q. Interface a 8KB RAM to μP .



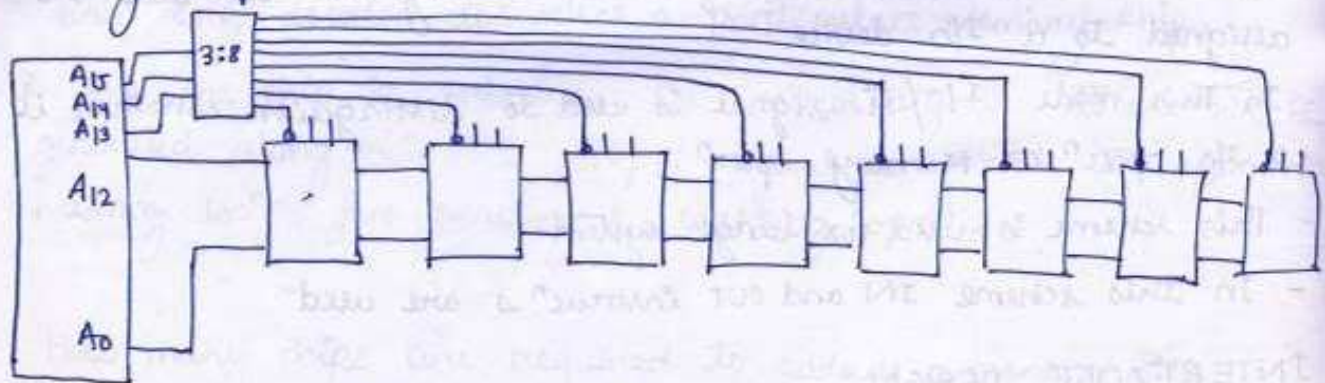
memory space is from
0000H to 1FFFFH

3. Interface a 64 KB memory using 8 KB chips.

Memory size = 64 KB

chip size = 8 KB

No. of chips = 8



	<u>SA</u>	<u>EA</u>
chip 1 :	0000H	1FFFH
chip 2 :	2000H	3FFFH
chip 3 :	4000H	5FFFH
chip 4 :	6000H	7FFFH
chip 5 :	8000H	9FFFH
chip 6 :	A000H	BFFFH
chip 7 :	C000H	DFFFH
chip 8 :	E000H	FFFFH

4. Interface a 2KB of memory space using 256 byte of memory chip.

5. Designing of RAM & ROM.

6. Design a 2764 EPROM (8KB size)

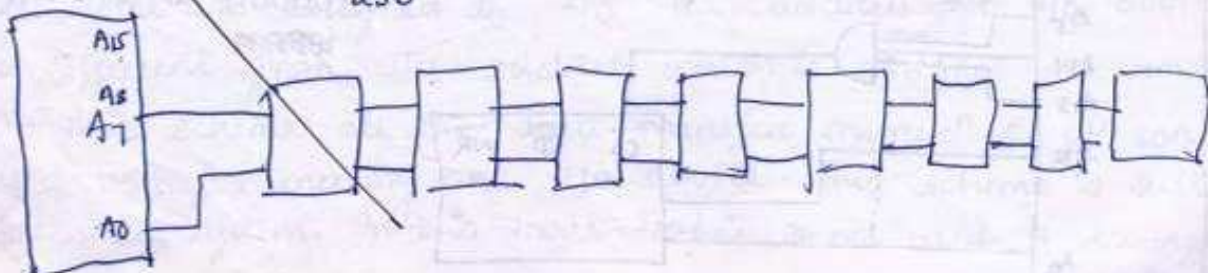
7. Design a 6264 RAM. (")

ANSWERS:

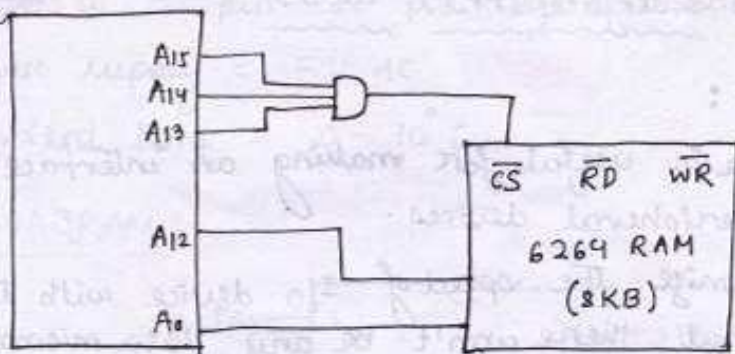
4. Memory size = 2KB = 2×1024 byte

chip size = 256 byte

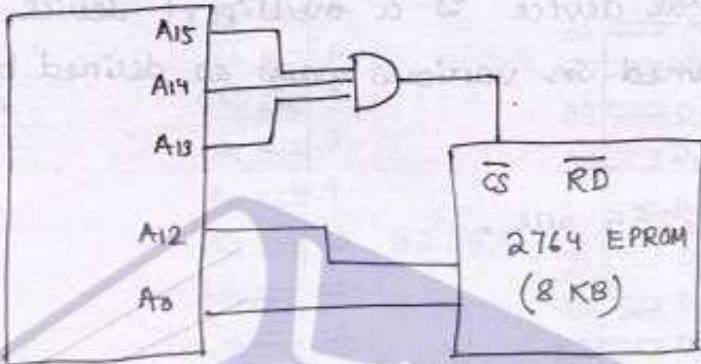
No. of chips = $\frac{2 \times 1024}{256} = 8$



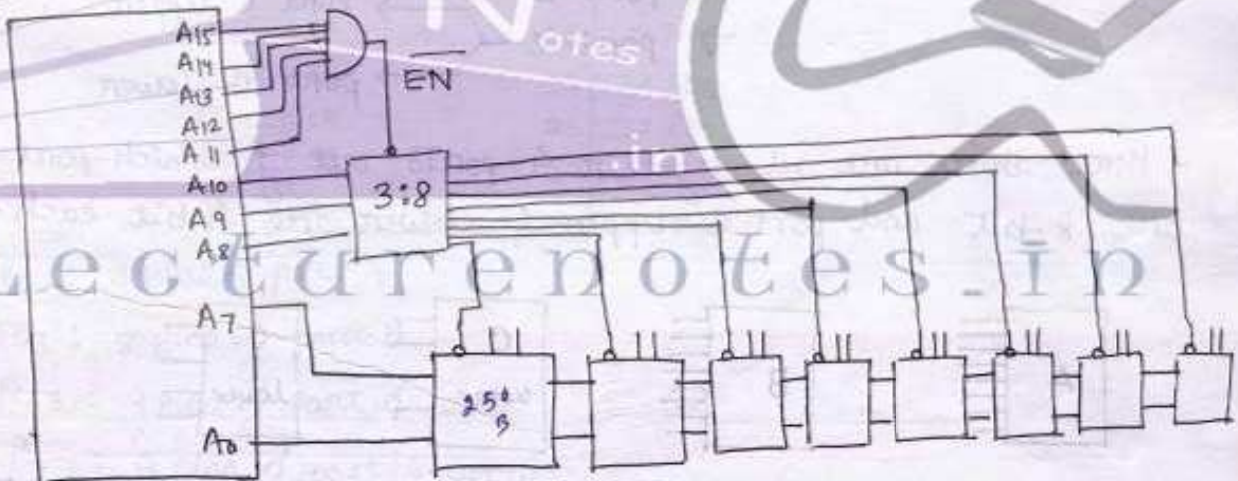
Q.7.



Q.6.



Q.



Programmable Peripheral Interface (PPI) 8255 :

2nd Mar '10

* Introduction and Features :

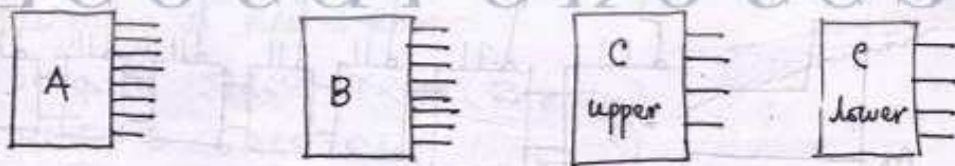
- It is a PPI. This device is useful for making an interface betⁿ a processor and the peripheral devices.
- This device will synchronize the speed of I/O device with the speed of processor so that there won't be any data mismatch during the data transfer.
- A programmable peripheral device is a multiport device where the ports may be programmed in various ways as desired by the programmer.

- The various version of 8255 are,

- (a) 8255 A
- (b) 8255A-5

- 8255 has 3 ports. → port A
→ port B
→ port C → Port C upper
→ port C lower

- Hence there are all together 4 ports out of which port A and B are 8-bit and port C upper & lower are 4 bit each.



- Hence each port can be programmed as an I/p port or as an O/p port.
- 8255 operates in 2 modes.

(1) I/O mode

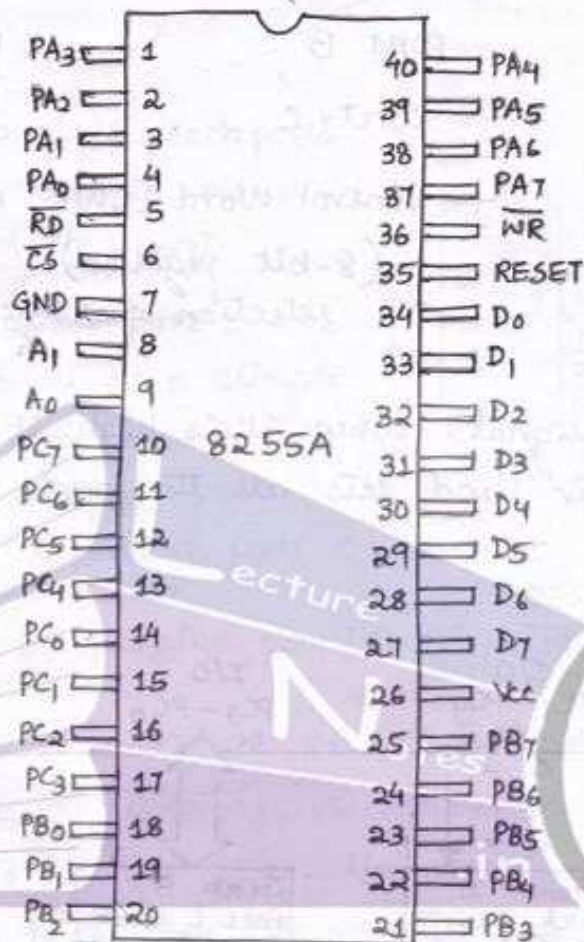
(2) BSR (Bit set reset) mode.

Again I/O mode operates in 3 various sub mode.

- (a) mode 0
- (b) mode 1
- (c) mode 2

- It is a 40 pin IC package, designed with H-MOS Technology.
- Power supply = 5V dc
- Ambient Temp = 0-70°C

PIN DIAGRAM:



PA₀ - PA₇ : 8 pins of port A

PB₀ - PB₇ : 8 pins of port B

PC_{L0} - PC_{L3} : 4 pins of port C lower

PC_{U4} - PC_{U7} : 4 pins of port C upper

D₀ - D₇ : Bidirectional data bus for data transfer

Vcc : Power supply

GND : ground

\overline{CS} : chip select signal, active low signal. If it is enabled, it creates an interface betⁿ 8085 and peripheral devices (8255)

\overline{RD} : Active low signal, used for read operⁿ. When this signal goes low, CPU reads data from the selected port.

\overline{WR} : Active low signal, used for write operⁿ. When this pin goes low, CPU writes data of the selected port of 8255.

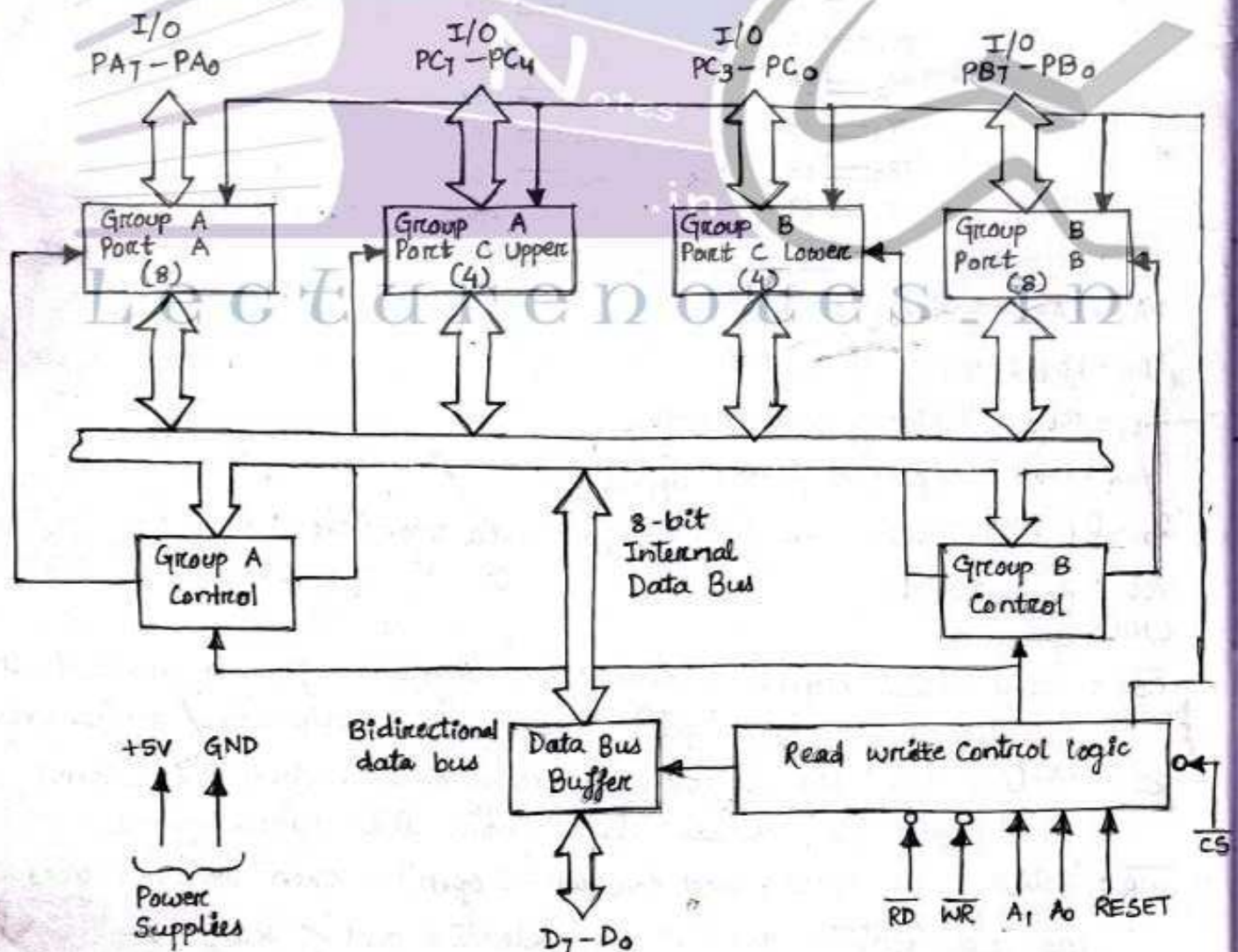
a_2, a_1 : These 2 pins are used for selecting various ports and to select the control word register.

Imp

\overline{CS}	A_1	A_0	
0	0	0	- Port A will be selected
0	0	1	- port B
0	1	0	- port C
0	1	1	- Control Word (CWR) Register (8-bit register)
1	X	X	- No selection of 8255

RESET : Active high signal. When it is high, it clears the control register and sets all the ports as I/p ports.

BLOCK DIAGRAM OF 8255 :



MODES OF OPERATION OF 8255

The I/O mode of operation of 8255 has 3 different modes.

- mode 0
- mode 1
- mode 2

MODE 0: In this mode, each port (A, B, C) can operate as a simple I/P and O/P port.

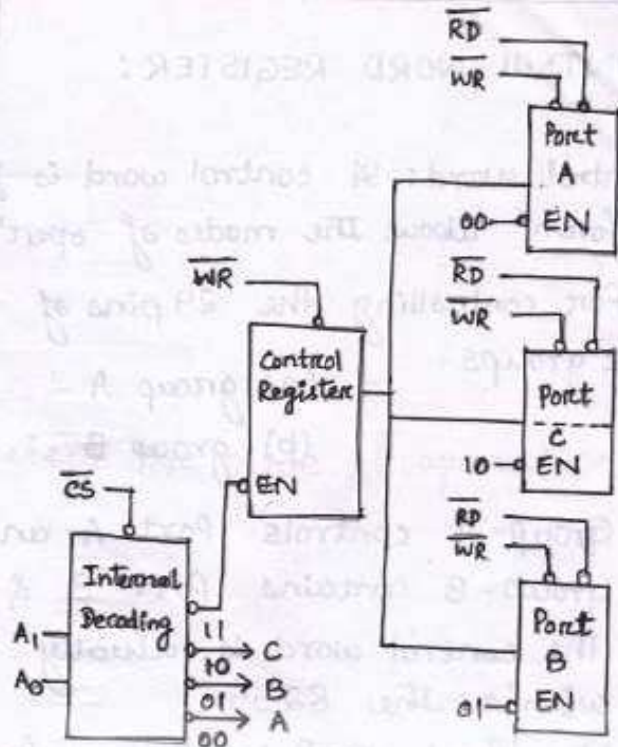
MODE 1: Here port A and B are designed to operate as a simple I/P, O/P port and by the

same time 6 pins of port C ^{are} used to control port A & B and the remaining 2 pins of port C can be used as I/O port.

- When port B is operation as I/P & O/P port, $PC_0 - PC_2$ will control it.
- When port A is operating as I/P & O/P port, $PC_3 - PC_5$ will control.
- When port A is operating as O/P port, PC_3, PC_6 & PC_7 will control.
- When port B is acting as I/P port, port A as I/P port the pins $PC_0 - PC_2$ will control port B and $PC_3 - PC_5$ will control port A and PC_6, PC_7 will act as simple I/O port at that time.
- When port B is acting as O/P port, and port A as O/P port, $PC_0 - PC_2$ will control port B, PC_3, PC_6 and PC_7 will control port A. $PC_4 - PC_5$ will act as I/P or O/P port. (Ghosh Sreidhar)

MODE - 2: It is a bidirectional mode.

- In this mode port A can be programmed as a bidirectional I/O port.
- Mode-2 operⁿ is only for port A.
- When port A is programmed to act in mode 2 by the same time port B can be used to operate either in mode '0' or '1'.
- For mode 2 operation $PC_3 - PC_7$ are used for controlling port A. Remaining 3 pins are unused at that time.



CONTROL WORD REGISTER:

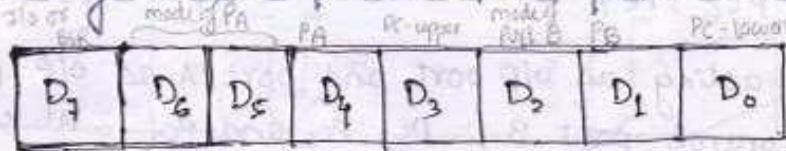
Control word: A control word is formed which contains the various informⁿ about the modes of operⁿ and funcⁿ of various ports.

- For controlling the 24 pins of I/O port, as it is divided into 2 groups.

(a) group A

(b) group B

- Group-A controls Port A and port c upper.
- Group-B contains Port B & port c lower.
- The control word is usually written on a CWR which is within the 8255.
- No read operⁿ is allowed for a CWR.
- A control register ^{is} an 8-bit register. The control word bit corresponding to a particular bit position will identify the mode of operation as well as the port status (i/p or o/p).
- If a port is made to act as an i/p port, then the bit corresponding to that particular port is set to 1.
- If a port is made to act as an o/p port, then the bit corresponding to that particular port is set to 0.



D₀ : It is for port c lower.

To make port c lower as i/p port. D₀ is set to 1.
else for o/p port, it is set to 0.

D₁ : Port for B.

To make as i/p port, D₁ = 1.

To make as o/p port, D₁ = 0.

D₂ : for the selection of mode for port B.

if P_B has to be operate in mode 0, this bit is set to '0'.

if P_B operates in mode 1, it is set to '1'.

D_3 : for port c upper.

Pc as i/p port, $D_3 = 1$.

PA as o/p port, $D_3 = 0$.

D_4 : for port A.

PA as i/p port, $D_4 = 1$

PA as o/p port, $D_4 = 0$.

D_5 & D_6 : These are used to select the mode of operation of port A.

D_6	D_5	MODE
0	0	Mode 0
0	1	Mode 1
1	0	Mode 2
1	1	Mode 2

D_7 : D_7 will identify whether 8255 is operating in i/o mode or BSR mode.

If $D_7 = 0$, BSR mode.

$D_7 = 1$, i/o mode.

* A control word 95H is assigned to 8255. Explain the status of various ports.

→ 1 0 0 1 0 1 0 1

$D_7 = 1$, i/o mode

$D_6 D_5 = 00$, PA is in mode 0.

$D_4 = 1$, PA i/p || $D_3 = 0$, Pc upper o/p || $D_2 = 1$, PB mode 1

$D_1 = 0$, PB o/p || $D_0 = 1$, PC i/p

* What will be the control word for operating 8255 in mode 0, PA - i/p, PB - i/p, PC - i/p, PB in 0 mode.

→ 1 0 0 1 1 0 1 1

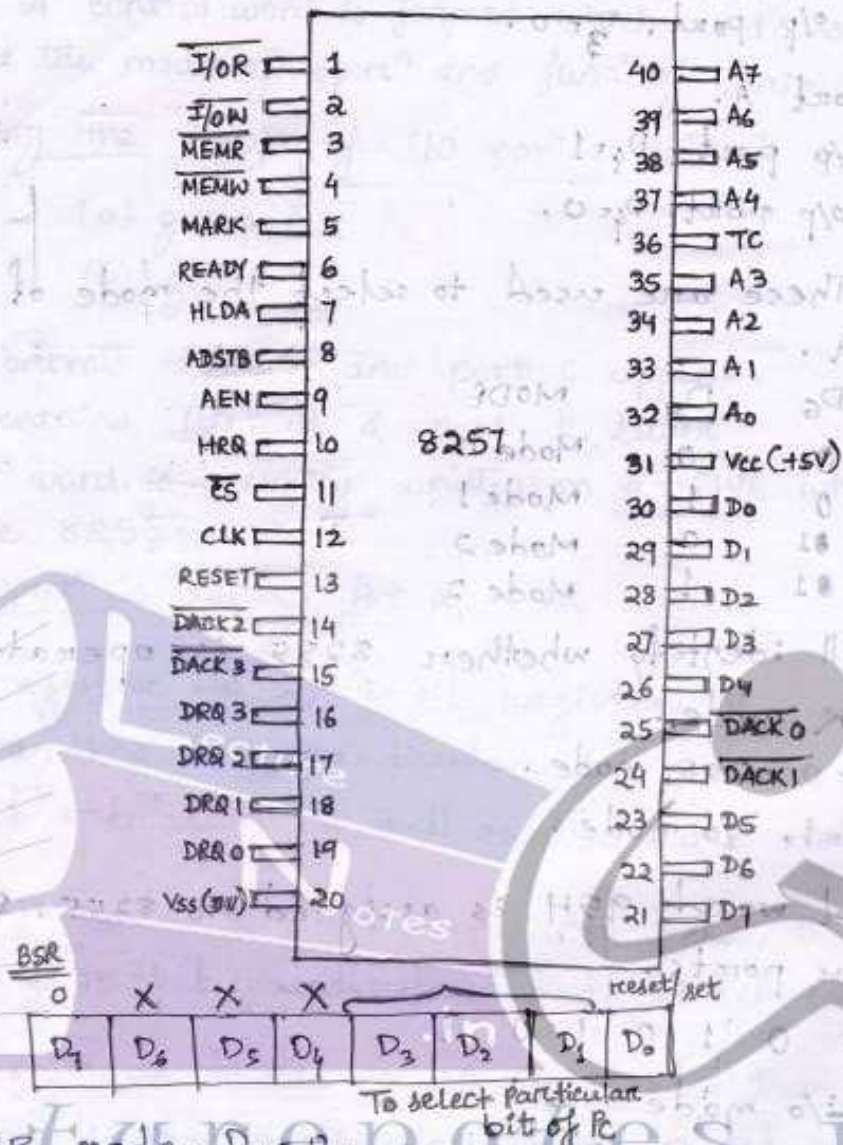
BSR MODE:

- This mode is concerned only about the 8-pins of Pc.

- 8255 will operate in BSR mode when $D_7 = 0$.

- This mode will make set or reset to each bit of port c.

cont...



D_7 : for BSR mode, $D_7 = 0$.

D_0 : It will make set & reset on/off to a particular bit of Pc.

$D_0 = 0$, reset/off || $D_0 = 1$, set/on

D_1, D_2, D_3 : These 3-bits are used to select a particular bit of Pc.

D_3	D_2	D_1	Bit
0	0	0	Pc_0
0	0	1	Pc_1
0	1	0	Pc_2
0	1	1	Pc_3
1	0	0	Pc_4
1	0	1	Pc_5
1	1	0	Pc_6
1	1	1	Pc_7

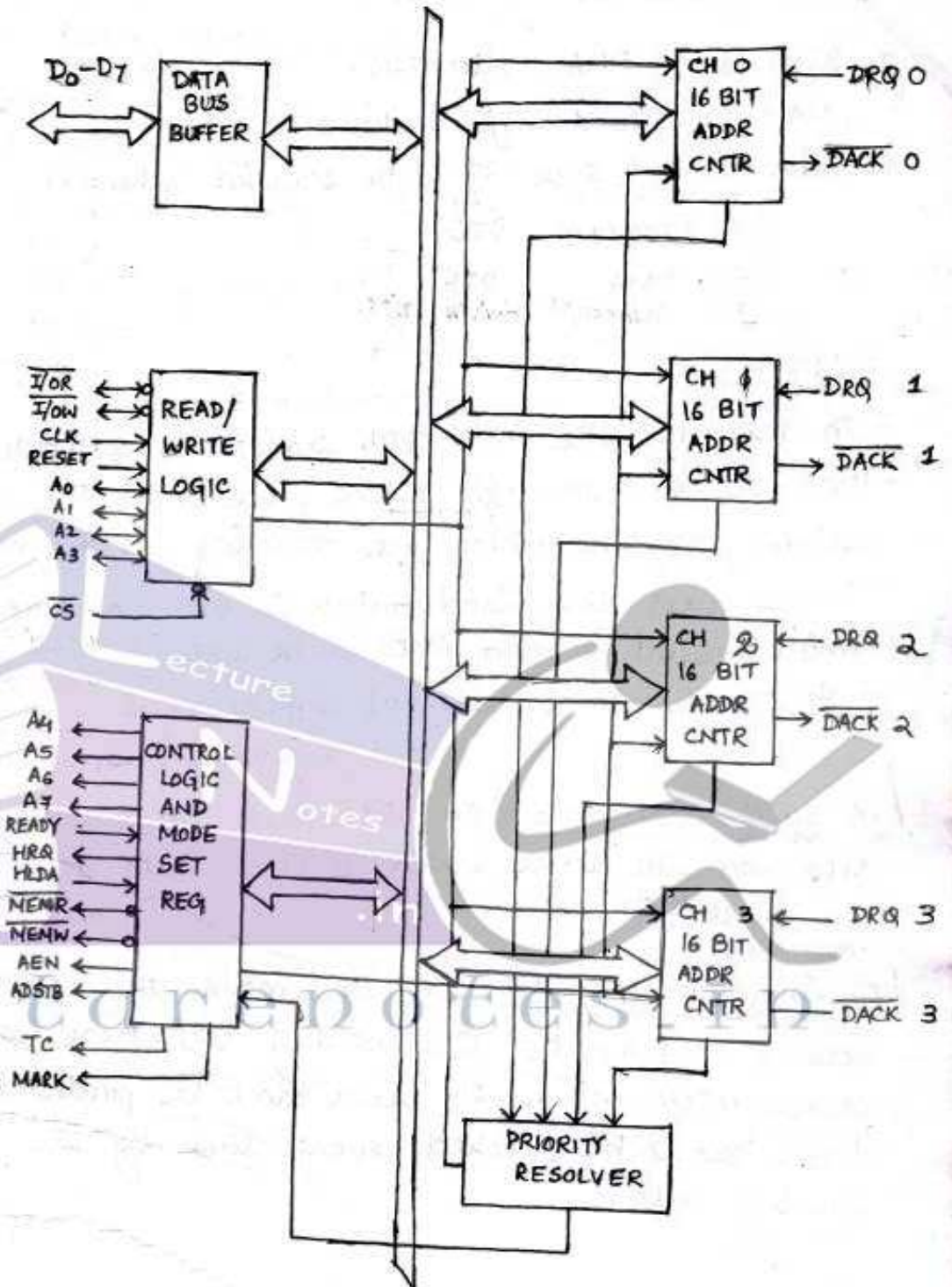
|| D_4, D_5, D_6 are not used in BSR mode. Usually they are set to '0'.

eg: To select 7th bit of Pc, $CW = 0FH$

(Memory)

8257

(I/O)



[FUNCTIONAL BLOCK DIAGRAM OF 8257]

8257 (DMA Controller) :

8th March

DMA: Direct Memory Access.

- 8257 is a DMA controller.
- DMA is a data transfer scheme.
- There are 3 types of data transfer schemes.
 - (1) Program DTS
 - (2) DMA DTS
 - 3 Interrupt driven DTS.

Program DTS:

- In this case the whole data transfer is controlled by the CPU.
- Here the data transfer taking place under the control of the certain programs which are residing in the main memory.
- In this case the disadvantage is we can transfer only a small amount of data. Data can be lost.
- It operates in 2 modes. (a) Synchronous
(b) Asynchronous

(a) Synchronous mode: In this mode, the device which sends data and the device which receives data are synchronized in the same clk pulse.

(b) Asynchronous mode: In this mode, the speed of I/O device doesn't match with the speed of CPU. Here the timing characteristics of an I/O device can't be predicted. So, the I/O device has to be checked every time by the CPU before sending data.

DMA DTS: In this DTS, data is directly transferred from memory to I/O or I/O to memory without the help of CPU.

- It is an I/O technique used for high speed data transfer.
- The controller 8257 is used to create an interface betⁿ I/O and the RAM for which high speed & high amount of data transfer can take place.

- This controller is known as DMA controller.
- 8257 is a programmable DMA controller used with 8085 & other processors for direct memory access.
- It has 4 channels, channel 0, ch 1, ch 2 & ch 3. Hence 4 I/O devices can be interfaced to the memory through 8257 simultaneously.

- Each channel has 2 registers, ~~one~~ byte count register (BCR),
 ② DMA address register.

or ① Terminal Count Register

- It operates in 3 different modes, ① There are two common registers for all the channels

(a) DMA ~~flag~~ write

(b) " read

(c) " verify.

① mode set Register
 ② status "

- 8257 can operate in another 2 modes depending on the speed of memory & I/O.

(a) cycle stealing mode

(b) Burst mode

(1) Cycle Stealing Mode:

- In this mode, the memory and I/O device is very slow in speed.
- Here 8257 repeats a single byte transfer sequence for a longer time as long as the peripheral maintains its request high.
- This type of opⁿ is called cycle stealing mode.
- This mode of opⁿ is preferable for transferring small amount of data.

(2) Burst Mode:

- If the memory & I/O device is fast enough to send / receive a bulk of data then it is called Burst Mode.
- It is preferable for transferring a bulk amount of data.

DMA Register:

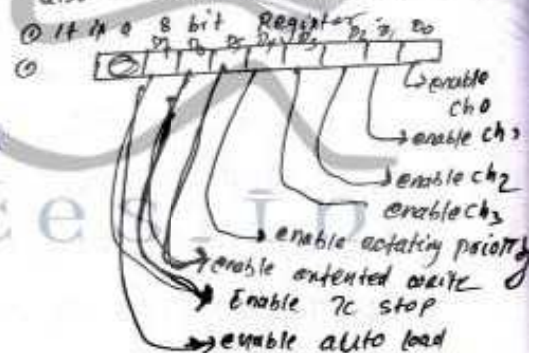
- ① Each channel has a DMA Register.
- ② The starting address of the memory location which will be accessed by the device is 1st loaded on the DMA address register of the channel.
- It is a 16 bit register.
- This register is loaded by the CPU with the address of 1st memory location i.e. to be accessed during DMA transfer.
- During DMA operⁿ (each) it stores the next memory locⁿ to be accessed in the next DMA cycle.

Byte Count Register: (Terminal Count Reg. / TCR)

- 16 bit register
- Used to keep track on the no. of bytes that is to be transferred for any channel.
- Also used to set the mode of operⁿ for various channel.
- The lower 14 bits i.e. $A_0 - A_{13}$ are loaded with a value i.e. to be transferred and the next 2 MSB A_{14} & A_{15} is used to set the mode of operⁿ of DMA.

A_{15}	A_{14}	
0	0	Verification
0	1	DMA write
1	0	DMA read
1	1	No operation

mode set Register: - The mode set register is used for programming the 8257 as per the requirements of the system. The bit of the mode set Register is to enable the DMA channel individually & also to set the various modes of operⁿ.



BLOCK & PIN DESCRIPTION:

DRQ 0 - DRQ 3: (DMA Request)

- Each channel accepts a DMA request from its ~~per~~ associated peripheral through this pin.
- Priority $\rightarrow DRQ\ 0 > 1 > 2 > 3$
- The peripheral raise this pin high in order to obtain the DMA cycle/ control.
- Hence the peripheral with lower priority will have to maintain its DRQ high until it gets an acknowledgment from 8257.

DACK 0 - DACK 3 :

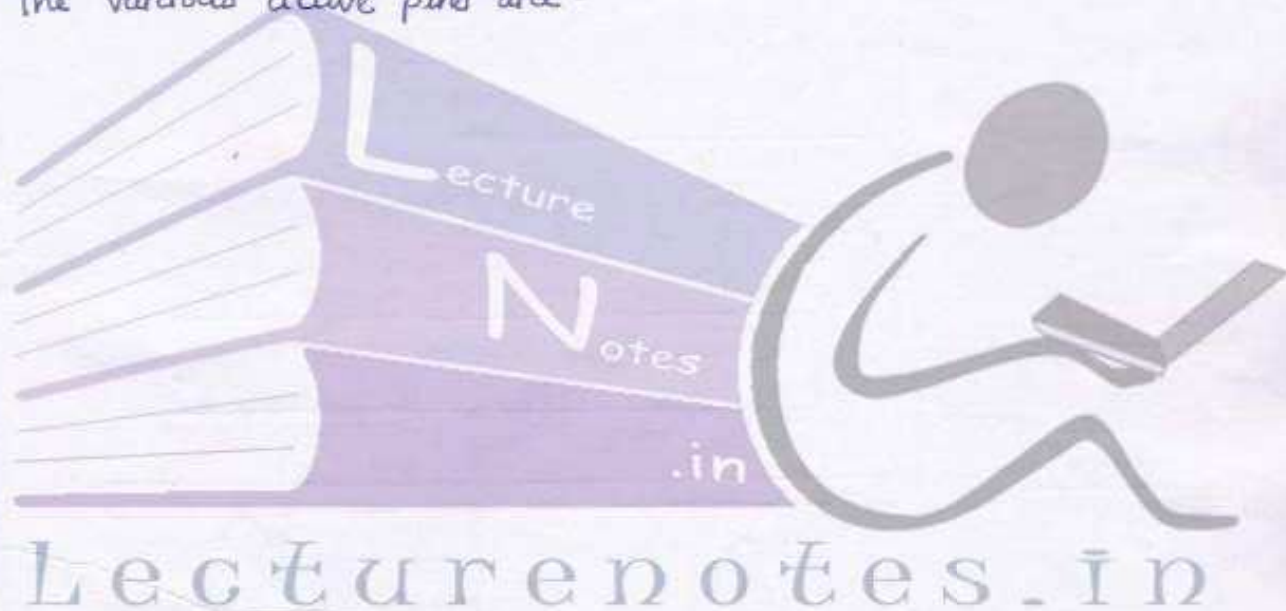
- A peripheral device requesting service through DRQ I/O to a channel is issued with an acknowledgment signal through DACK signal.
- Any DRQ request given by the I/O device will be acknowledged back by the DMA controller with a DACK signal.

D₀ - D₇ :

- Bidirectional data bus buffer used to interface 8257 with the system data bus.

READ/ WRITE CONTROL LOGIC :

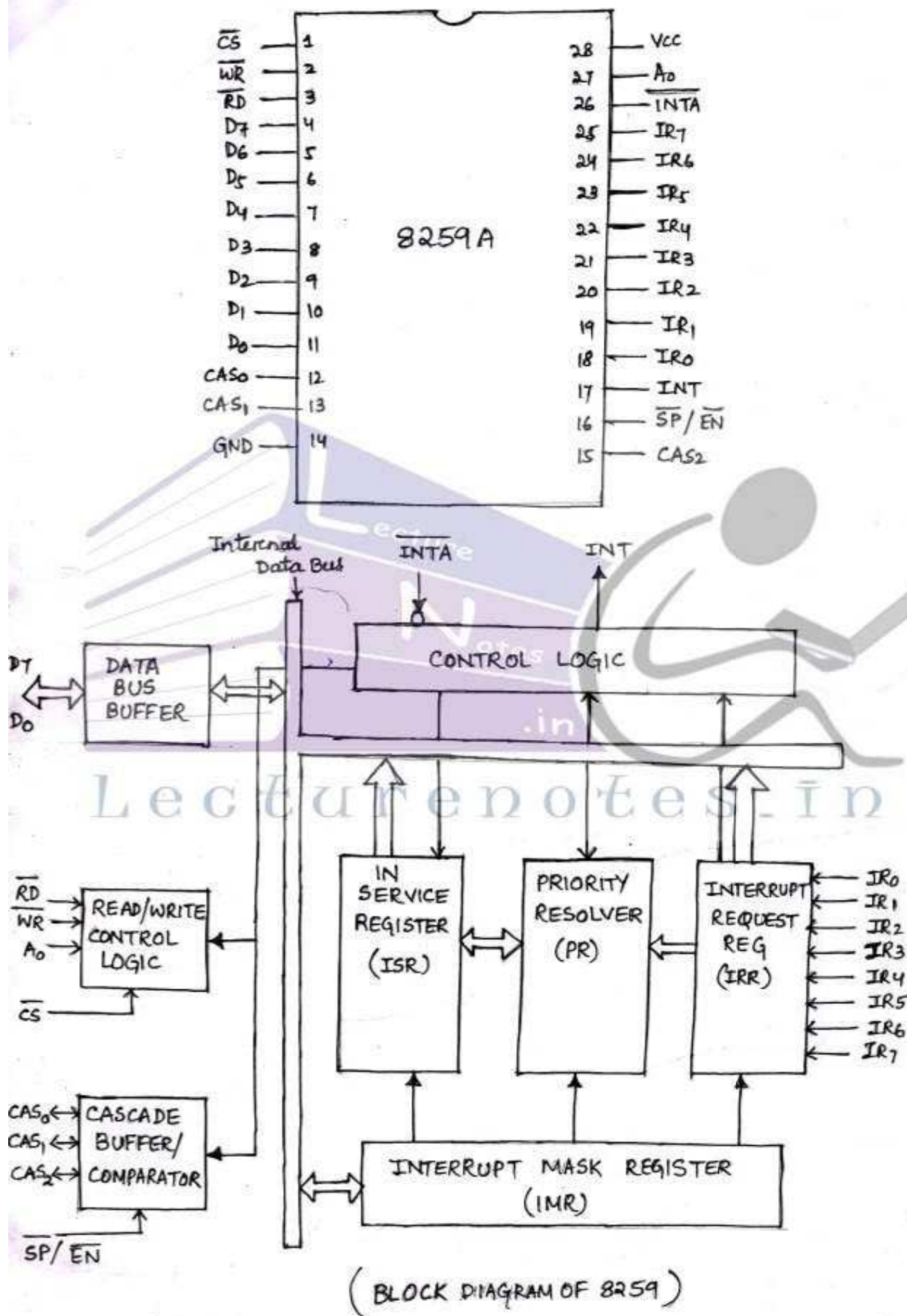
The various active pins are -



Status Register is 8 bit Register. The lower order 4 bits of this register contain the terminal count status for the four individual channels.

11th Mar

8259 PIC (Priority Interrupt Controller)



- Its main funcⁿ is to control the whole interrupt system.
- It accepts interrupt request from I/O devices.
- At a time if one 8259 is connected to a processor, then it can take request from 8 different I/O devices.
- If 8, 8259 are connected cascadedly, then it can take request from 64 different I/O devices.
- So, in order to accept more interrupts we can connect a series of 8259 in cascade.

FUNCTION: It determines which interrupt has highest priority at a given time. and sends an interrupt request signal to the CPU by activating the INTR pin of CPU.

- After getting the \overline{INTA} from CPU, 8259 places a CALL instrucⁿ along with the associated address of the current interrupt on the bus.
- In addⁿ to that 8259 can be programmed for a variety mode of oprⁿ. The various oprⁿs are,
 - (1) Priority check
 - (2) Masking of interrupt
 - (3) Setting of interrupt
 - (4) Gives complete informⁿ about the interrupt
- In order to initialize 8259 as an I/O device, CPU executes a set of initialisation command word i.e. (ICW). Similarly some OCWS are used to make 8259 operate at various

↓
(operational command word)

modes which includes masking of a particular interrupt.

BLOCK & PIN DESCRIPTION :

The whole block diagram of 8259 consists of 4 sections.

- (1) Interrupt & control logic system
- (2) Data bus buffer
- (3) RD and WR control logic block
- (4) Cascade buffer/Comparator section

Imp

(1) Interrupt & Control logic System:

It consists of various registers like IRR, IMR, ISR, PR.

IRR (Interrupt Request Register)

PR (Priority Resolver/Register)

IMR (Interrupt Mask Register)

ISR (" service ")

1. IRR

- It is used to store the information about the interrupt I/Os requesting for service.
- IRR has 8 pins starting from IR_0 to IR_7 , which allows 8 interrupts from 8 different I/O devices can come at a time.

2. ISR

- It holds the inform about the interrupt which is currently being serviced.

3. PR

- It determines the priority of interrupt requests.
- It decides the priority of any interrupt as decided by the priority mode set by OCWS.

4. IMR: This register can be programmed by an OCW to store the bits, which marks a particular interrupt (interrupt is idle)
- Any interrupt is being masked off by a specific s/w, which will not be serviced even if it sets the corresponding bit of IRR.
 - IMR operates on IRR.

CONTROL AND LOGIC BLOCK :

- This block has an i/p line and an o/p line i.e. \overline{INTA} and INT .
- After determining the priority, 8259 puts an interrupt request to the CPU through INT line.
- This INT line is directly connected to the $INTR$ line of CPU.
- Once INT gets set, CPU responds the signal by sending an acknowledgement signal \overline{INTA} .
- Once \overline{INTA} is received by 8259, the device places the address of the interrupt on a data bus by executing a call instruction.

Data Bus Buffer: There are 8 bit bidirectional data bus buffers, used to interface 8259 with system data bus and address bus.

Read and Write Control Logic Block :

- This section consists of ICW register and OCW registers, which are programmed by the CPU to set up 8259 and to make it operate in various modes.
- Under this section, the active pins are \overline{CS} , \overline{WR} , \overline{RD} .

\overline{CS} : This is a active low signal used to select the device.

\overline{WR} : It is used to write the ICWs and OCWs on 8259.

\overline{RD} : It is an active low signal used to read the status of various registers like IRR , IMR , \overline{ICWR} , \overline{OCWR} , PR .

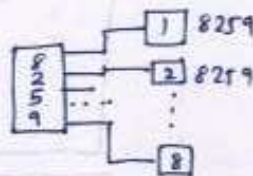
Cascade buffer and comparator :

- In order to communicate with more no. of I/O devices, the 8259 can be connect in cascade.

- This type of connecⁿ improves the interrupt handling capability.

- Here, from the various 8259, one 8259 will act as master and others are known as slaves.

- Here The 1st 8259 \rightarrow master
rest 8 8259 \rightarrow slaves

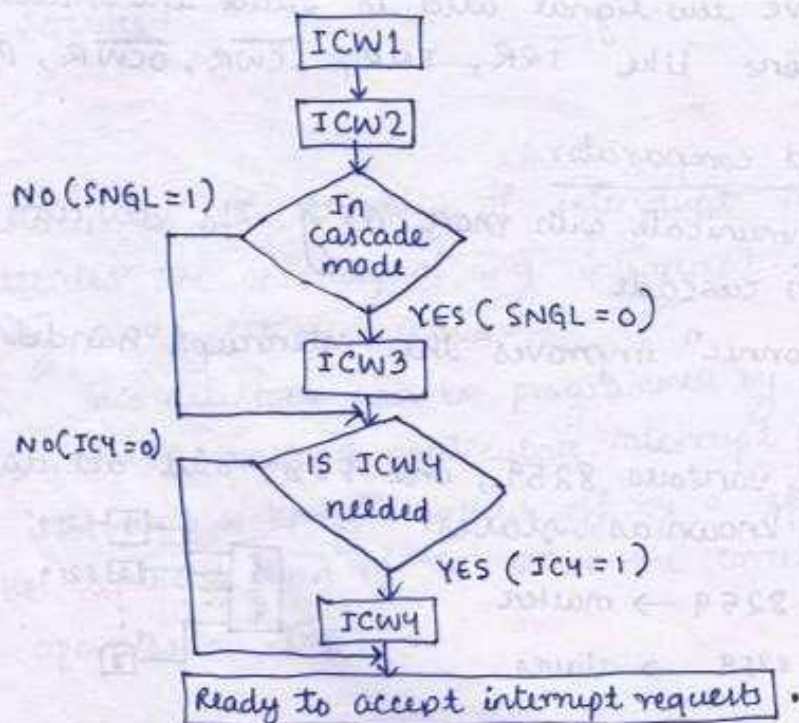


- Imp - 8259 can be set to work in master mode or in slave mode by the pin $\overline{SP}/\overline{EN}$. (Slave program / enable buffer)
- The 8259 can operate both in buffer mode as well as in non-buffer mode.
 - In non-buffer mode the pin $\overline{SP}/\overline{EN}$ is used to specify whether 8259 is operating as master or slave.
 - If a 0 volt is applied to this pin, it will act as slave. If 5 volt is applied, then it will act as master.
 - In buffer mode this pin is used as an o/p to enable the databus buffers of the system.

8259 PROGRAMMING :

- 8259 is programmed by the CPU by loading a set of ICW commands and OCW commands.
- Each 8259 is needed to be initialised by ICWs.
- The various ICWs are
 - ICW 1
 - ICW 2
 - ICW 3
 - ICW 4

ICW FLOWCHART :



ICW 1 :

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	A_7	A_6	A_5	1	LTIM	ADI	SNGL	ICW4

- An initialization command word issued to 8259 with $A_0 = 0$ and $D_4 = 1$ is treated as ICW 1.
- If $ICW 4 = 1$ (i.e. $D_0 = 1$) then ICW 4 is needed. If it is 0 then ICW 4 is not needed.
- If $D_1 = 1$, then 8259 is single.
If $D_1 = 0$, then 8259 is cascaded.
- ADI : Call address interval
If $D_2 = 0$, interval is 8
If $D_2 = 1$, interval is 4
- LTIM : There are 2 types of triggering modes used in 8259.
 - level triggering
 - edge "

LTIM = 0, edge triggering
LTIM = 1, level triggering
- A_5 to A_7 : It gives the vector address of interrupt.
Once ICW 1 is executed, the following task will be performed.

1. IMR get cleared
2. IR 7 I/P is assigned to lowest priority
3. Slave mode address set to 7.

8051:-

There are 3 types of branch control instructions:-
JMP, CALL, RETURN

1) JMP - It can be categorized in 3 groups-

- Long jump (LJMP) - ^{range} 64 KB
- Short jump (SJMP)
- Absolute jump (AJMP)

LJMP:- using this jump processor can jump anywhere within the memory (64KB)

SJMP - The short jump transfers or makes jump the program control within a 256 byte range.

AJMP:- The range of this jump is 2KB.

2) CALL:-

- LCALL (Long CALL)
- ~~SJMP~~ AJMP (Absolute CALL)

ACALL:- The range of the Subroutine is 2KB.

LCALL - The range is 64KB.

RETURN
3) RET, RETI:-

RET - Return from Subroutine

RETI - Return from Interrupt

MOV P1, # 80H → 80H load in port 1.

(port 1 to port 7)

STB P1.0 →



MICROCONTROLLER

MICROPROCESSOR

1. It is a general purpose device which can be programmed to perform different types of operation as per the requirement.
2. In case of μP the memory & the I/O devices are external. Hence interfacing ckt is required.
3. μP built systems are bulky, more expensive.
4. Power consumpⁿ more.
5. Slow in speed.
6. More flexible.
7. It has a single memory chip where both data & ports are lying.

MICRO CONTROLLER

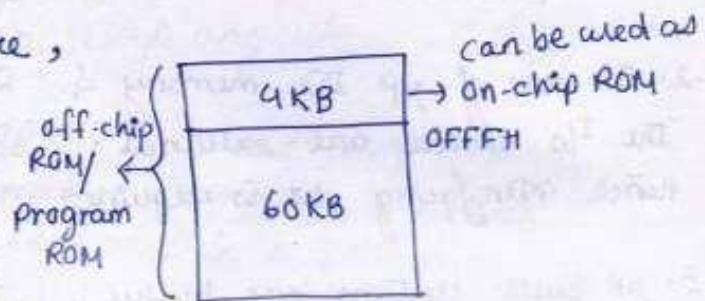
1. It is a specific purpose device. which once programmed, can't be modified or changed later on. Hence applied for a specific purpose.
2. μC has on-chip ROM, RAM, I/O ports and timing ckt everything.
3. Less expensive and less bulky than μP .
4. Less power consumpⁿ.
5. Fast.
6. Less flexible.
7. μC has sep^r separate data & ports.

LECTURENOTES IN FEATURES OF 8051 :

- 8-bit μ controller developed by H-MOS technology.
 - Operating freq = 12MHz.
 - Here multiplication & division instrucⁿs are directly available.
 - It has a boolean processor that supports bit wise operation.
 - It has 4KB onchip ROM memory.
 - It has 256 byte of onchip RAM.
 - 8051 can have a max^m of 64KB ROM. (extended^d program memory)
 - It is having a 64KB of ^{external} data memory i.e. RAM.
 - It has 32 bidirectional i/p/o lines. which are arranged for 4, 8-bit ports namely, Port-0, port-1, port-2, port-3.
- 8096 - 16 bit μC 8031/8052 - μC

- It has 2, 16-bit Timer and counter register.
- It has 4 register banks namely bank-0, bank-1, bank-2 & bank-3.
- Each bank has 8 registers.
- It has 16-bit address bus, multiplexed with port-0 & 2.
- one serial I/O port on the chip.
- Other version of 8051 are,

Intel MCS 51
PIC / micro chip
~~Atmel~~ Atmel



64KB \rightarrow 4KB + 60KB

- It has 6 interrupt sources

CPU	RAM	ROM
I/O ports	Timer	Serial port

Architecture of 8051:

The whole architecture of 8051 consists of

1. CPU
2. 2 types of memory (RAM, ROM)
3. I/O devices
4. Registers
5. Various ports.

\rightarrow GPRs

\rightarrow Stack Pointer

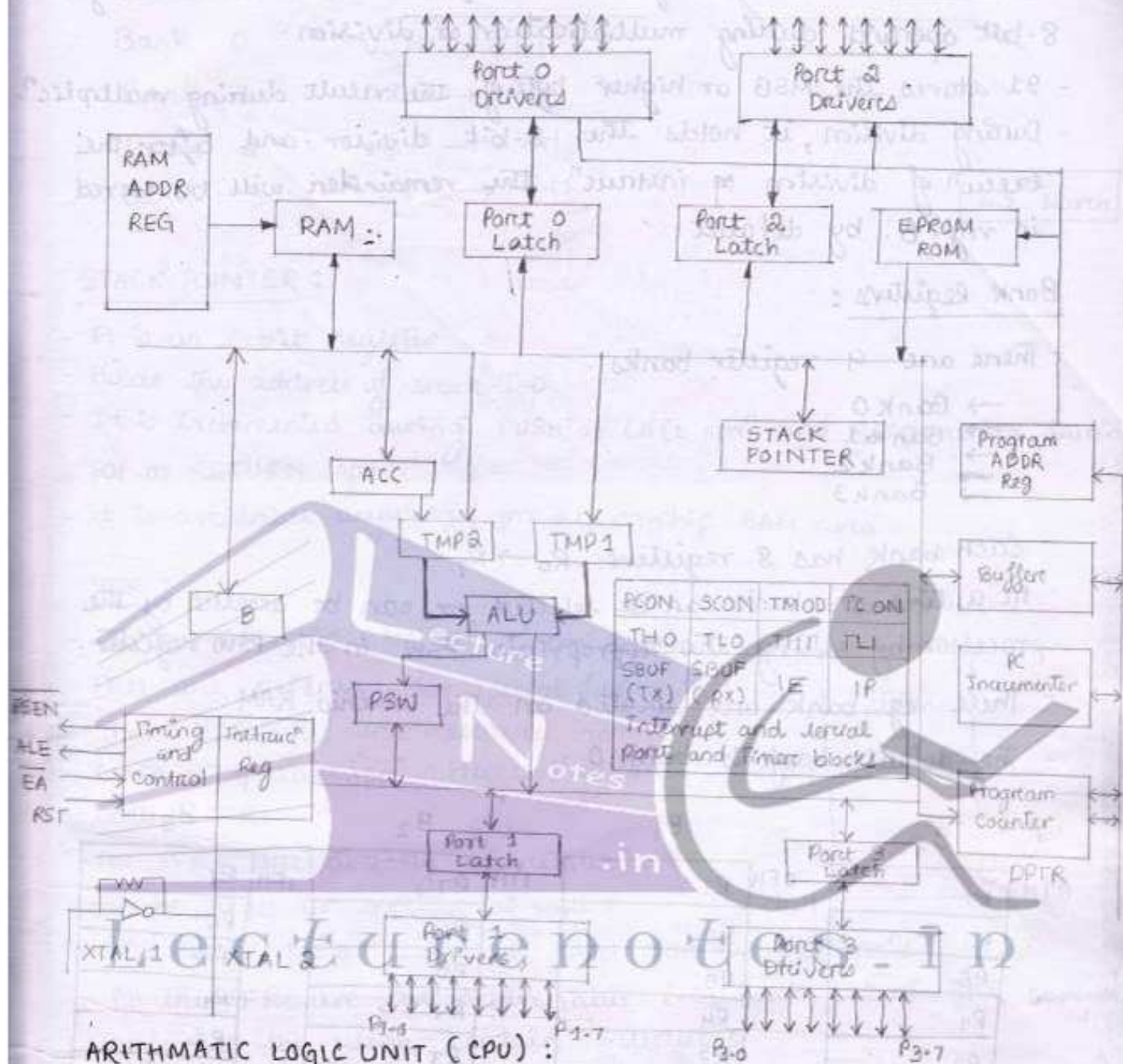
\rightarrow program counter

\rightarrow SFR (special funcⁿ register)

\rightarrow Timing & Control reg.

\rightarrow PSW (Program Status Word) reg.

\rightarrow DPTR (Data Pointer)



ARITHMETIC LOGIC UNIT (CPU) :

- This unit is doing all kind of arithmetic & logical operations on a 8-bit data bitwise.
- The additional funcⁿs are, multiplication & division.

REGISTERS:

(1) GPRs: There are 34 GPRs. They are,

→ Accumulator

Imp → Reg. B

→ 4 Banks having 8 registers each.

Register B: 8 bit register. Its funcⁿ is to store one of the 8-bit operand during multiplication or division.

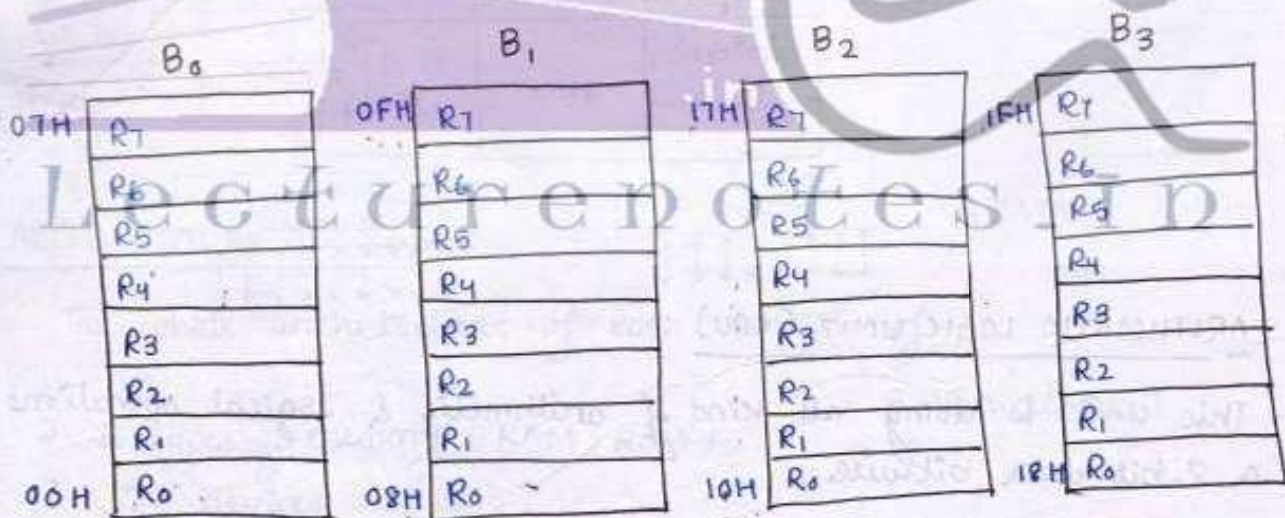
- It stores the MSB or higher byte of the result during multiplicⁿ.
- During division, it holds the 8-bit divisor and after the execuⁿ of division of instrucⁿ the remainder will be stored in reg. B. by default.

Bank Registers:

- There are 4 register banks.

→ Bank 0
→ Bank 1
→ Bank 2
→ Bank 3

- Each bank has 8 registers $R_0 \rightarrow R_7$.
- At a time one bank can be selected or can be accessed by the processor by setting the appropriate PSW in the PSW register.
- These reg. bank are located on the on-chip RAM.
- The default bank is bank 0.



- If we want to store a data in R₄, it will be stored in the memory locⁿ 04H by default of bank 0.
- If the programmer select a particular bank. (let it bank 3) then the data will be stored in 1CH.

Address range of each bank,

Bank 0	00H → 07H
Bank 1	08H → 0FH
Bank 2	10H → 17H
Bank 3	18H → 1FH

23rd March

STACK POINTER :

- It is an 8-bit register.
- Holds the address of stack-Top.
- It is incremented during PUSH or CALL opⁿ and decremented during POP or RETURN opⁿ.
- It is available anywhere on the on-chip RAM area.

SFR :

- On the ON-chip RAM locⁿ, from 80H to FFH are reserved for special funcⁿs.
- These 128 bytes are reserved for certain special opⁿ. Hence they are called SFRs.
- All SFRs are directly addressable & can be read or written as well.



- So, these locⁿs can't be used for normal opⁿs.

- All the SFRs are bit addressable i.e. each bit of SFR can be set/reset by using certain instrucⁿs.

eg: The various SFRs are,

SFR Symbol	Register Name	Address
ACC*	Accumulator	EOH
B*	Reg. B	FOH
P ₀ *	Port 0	80H
P ₁ *	Port 1	90H
P ₂ *	Port 2	A0H
P ₃ *	Port 3	B0H

SFR SYMBOL	REGISTER NAME	ADDRESS
------------	---------------	---------

DPTR	Data Pointer	80H
→ DPH	Data Pointer (higher byte)	83H
→ DPL	" " (lower byte)	82H
PCON	Power Control	97H
SCON*	Serial control	98H
TCON*	Timer 0 / Counter control	88H
TMOD	Timer/counter Mode control	89H
SBUF	Serial data buffer	99H
PSW*	Prog. Status Word	D0H
IP*	Interrupt Priority control	8BH
IE*	" enable "	A8H

PSW

- 8 bit register consists of carry flag, auxiliary carry, parity, overflow & some of the bits ^{used to} select a particular bank.

PSW7	PSW6	PSW5	PSW4	PSW3	PSW2	PSW1	PSW0
C	AC	FO	RS1	RS0	OV	-	P

PSW0: PSW0 \Rightarrow 1, acc. has ~~an~~ ^{odd} no. of one.
 \Rightarrow 0, acc. has ~~an~~ ^{even} no. of '1's.

PSW1: Undefined.

PSW2: It is the generation of carry in the 6th bit not in the 7th or vice versa (i.e. when PSW2 = 1)

(i.e. carry is generated in D6 not in D7 or in D7 not in D6)

PSW3 & PSW4: These 2 bits are used to select a particular bank from the bank reg.

PSW ₄	PSW ₃	Bank Selected
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

TH0	Timer/counter 0 (high)	8CH
TL0	0 (low)	8AH
TH1	1 (high)	8DH
TL1	1 (low)	8BH

ADDRESSING MODES:

PSW₅: F0 is available to user as a general purpose flag.
This flag can be set/reset by using certain s/w.

PSW₆: If carry is generated from P₃ → D₄.
Then PSW₆ = 1, otherwise PSW₆ = 0.

PSW₇: If carry is generated at D₇, PSW₇ = 1, otherwise PSW₇ = 0.

DATA POINTER:

- 16 bit reg.
- higher byte of reg. is known as DPH & lower byte is known as DP_L.
- DPTR is used for addressing off-chip data ^{or prog.} memory & i/p, o/p port with a command MOV X and MOV C.
- With this 16-bit pointer, a max^m of 64 KB of data memory and a max^m of 64 KB of prog. memory can be addressed.
- Used as a GPR.
- There is an instructⁿ INC DPTR which is used to increment the content of DPTR but there is no such instructⁿ which will decrement the content of DPTR.

TIMER REG.:

There are 3 16-bit timer & counter reg. which are in the form of pair as TL0 & TH0 and TL1, TH1.

- The oprⁿ of this register is to give timing & counting.
- There are various modes in which the time can be configured & controlled. For this purpose to select the timing mode, a reg. 'Tmod' is used & to control the time 'Tcon' is used.

I/O Ports: There are 4 I/O ports P_0, P_1, P_2, P_3 .

- Each port has 8-lines & each line is bit-addressable.

ADDRESSING MODES:

The various addressing modes are,

- 1) Register addressing mode
- 2) Direct addressing mode
- 3) Register indirect addressing mode
- 4) Immediate addressing mode
- 5) Base register plus index register addressing mode.

(1) Register addressing mode:

In register addressing, registers R_0 through R_7 from the selected register bank, accumulator, B-register, carry bit and DPTR are used. An MCS-51 instruction using this addressing mode refers the registers R_0 through R_7 in the opcode itself. The least significant bits of the opcode indicate which register is to be used.
eg: $\text{MOV } A, R_n$

(2) Direct Addressing mode:

In direct addressing mode, the direct addressing address of the operand is specified in the instruction itself. Direct addressing mode uses the lower 128 bytes of internal RAM and the special funcⁿ register. eg: $\text{MOV } A, \text{direct}$ i.e. $\text{MOV } A, 54H$.

(3) Register Indirect Addressing:

Register indirect addressing uses any one of the registers R_0 or R_1 from the selected register bank, as a pointer to the locations in the 256 bytes of data memory block. The SFRs are not addressed by this mode also the external memory beyond the lower 256 bytes is not addressed.

eg: $\text{MOV } A, @R_i$

(4) Base register

(4) Immediate Addressing mode:

Immediate addressing allows using immediate data (constants) as a part of the instruction. eg: `MOV A, #45H`.

(5) Base register plus index register addressing mode:

This mode allows a byte to be accessed from the prog. memory, whose address is calculated as the sum of a base register (DPTR or PC) and index register, accumulator.

eg: `MOVC A, @A+DPTR`

This "instruc" will fetch a byte from the prog. memory, whose address is calculated by adding the original 8-bit unsigned contents of the accumulator and the 16-bit contents of the DPTR.

If the DPTR contains `0FFFOH` and the accumulator contains `05H`, then the byte stored in `FFFFH` will be copied into the accumulator. This method facilitates the look-up table access.

8051 INSTRUCTION SET:

1. DATA TRANSFER INSTRUCTION:

1. `MOV A, Rn`: The content of register (R_n) will move to A.
The default bank is bank 0 if any other bank is not selected.

- eg: `MOV A, R1`.
- 1 byte instruction.

2. `MOV A, direct` (8-bit address): eg - `MOV A, 34H`

Content of `34H` (on-chip RAM) is moved to A. (But on-chip ROM can't be used as it is a 16-bit address memory)

Move the content of on-chip memory locⁿ whose address is given in the "instruc" to the accumulator. 2-byte instrucⁿ.

3. `MOV Rn, A`: The content of A is moved to the register.

(by default bank 0). eg: `MOV R2, A`.

- 1 byte instrucⁿ.

4. MOV A, #data : Move immediate data to A.

eg: MOV A, #FFH

- FFH is transferred to A directly.
- Can access 8-bit data.
- 2 byte instruction.

5. MOV A, @R_i : Move the content of the on-chip memory locⁿ whose address is in the R_i. It is a 1-byte instruction.

eg: MOV A, @R₇

The data present in R₇ is the address of an on-chip memory and the content of that memory locⁿ is moved to A.

[R₇] = 35H

[35H] = 09H

Then 09H will be moved to A



6. MOV R_n, direct : Move the content of on-chip memory location whose address is given in the instruction to the given register (by default bank 0).

eg: MOV R₂, 34H

Content of 34H is moved to the R₂ of bank 0 (by default).

- 2 byte instruction.

7. MOV R_n, #data : The data given in the instruction is directly moved to the register. eg: MOV R₁, #34H

- 2 byte instruction.

8. MOV direct, A : Move the content of on-chip A to the memory locⁿ given in the instruction. (2 byte instruction)

9. MOV direct, R_n : Move the content of register to the memory locⁿ given in the instruction. (2 byte instruction)

10. MOV direct, direct : Move the content of one memory locⁿ to other memory locⁿ given in the instruction.

eg: MOV 35H, 63H.

- 3 byte instruction.

11. MOV direct, #data: Move the data given in the instrucⁿ directly to the memory locⁿ given in the instruction.
eg: MOV 34H, #03H.
12. MOV direct, @Ri: The data present in the on-chip memory location whose address is present in the Ri is moved to the memory location which is given in the instrucⁿ.
eg: MOV 45H, @R3
- 2 byte instrucⁿ.
13. MOV @Ri, direct: The data present in the memory locⁿ whose address is given in the instrucⁿ is moved to the memory locⁿ whose address is present in Ri.
eg. MOV @R2, 25H. (2 byte)
14. MOV @Ri, A: The content of A is moved to the memory locⁿ whose address is present in Ri.
eg. MOV @R2, A (1 byte instrucⁿ)
15. MOV DPTR, #data (16-bit): Load the data pointer with the 16-bit data. 3-byte instrucⁿ.
eg: MOV DPTR, #4500H
16. MOV A, @A+DPTR: Move code byte relative to DPTR to A.
- 1 byte.
17. MOV A, @A+PC: Move code byte relative to PC to A.
- 1 byte.
18. MOV @Ri, #data: Move immediate data to the indirect RAM.
- 2 byte instrucⁿ.
19. PUSH direct: Push the direct byte onto stack. (2-byte)
20. POP direct: Pop the direct byte from stack. (2-byte)

21. MOVX A, @Ri : Move the external RAM (8-bit address) to A.
The content of register which refers to the external RAM & the content of the off-chip RAM memory locⁿ goes to A. (1 byte instrucⁿ)
It is of base register + index register addressing mode type.
22. MOVX A, @DPTR : Move the external RAM (16-bit address) to accumulator. 1 byte instruction.
23. MOVX @Ri, A : Move the accumulator content to the external RAM (8-bit address). 1 byte instrucⁿ.
24. MOVX @DPTR, A : Move the content of A to external RAM (16-bit address). 1 byte instrucⁿ.
25. XCH A, Rn : Exchange the content of reg. with [A].
1-byte instrucⁿ.
26. XCH A, direct : Exchange the content of the memory locⁿ given in the instrucⁿ with [A]. 2-byte instrucⁿ.
27. XCH A, @Ri : Exchange indirect RAM with accumulator.
1-byte instrucⁿ.
28. XCHD A, @Ri : Exchange low order digit indirect RAM with A.
1-byte instruction.

2. ARITHMETIC INSTRUCTIONS :

<u>MNEMONICS</u>	<u>DESCRIPTION</u>	<u>BYTE</u>
1. ADD A, Rn	Add register to A	1
2. ADD A, direct	Add direct byte to A	2
3. ADD A, @Ri	Add indirect RAM to A	1
4. ADD A, # data	Add immediate data to A	2
5. ADDC A, Rn	Add register to A with carry.	1
6. ADDC A, direct	Add direct byte to A with carry	2

7. ADDC A, @Ri	Add indirect RAM to A with carry	1
8. ADDC A, #data	Add immediate data to A with carry	2
9. SUB A, Rn	Subtract register from accumulator	1
10. SUB A, direct	Subtract direct byte from A	2
11. SUB A, @Ri	Subtract indirect RAM from A	1
12. SUB A, #data	Subtract immediate data from A	2
13. SUBB A, Rn	Subtract register from A with borrow	1
14. SUBB A, direct	Subtract direct byte from A with borrow	2
15. SUBB A, @Ri	Subtract indirect RAM from A with borrow	1
16. SUBB A, #data	Subtract immediate data from A with borrow	2
17. INC A	Increment accumulator	1
18. INC Rn	Increment register	1
19. INC direct	Increment direct byte	2
20. INC @Ri	Increment indirect RAM	1
21. DEC A	Decrement accumulator	1
22. DEC Rn	Decrement register	1
23. DEC direct	Decrement direct byte	2
24. DEC @Ri	Decrement indirect RAM	1
25. INC DPTR	Increment data pointer	1
26. MUL AB	Multiply A & B (16 bit product in B:A register)	1
27. DIV AB	Divide A by B (quotient in A, remainder in B)	1
28. DA A	Decimal adjust accumulator	1

3 LOGICAL INSTRUCTIONS :

<u>MNEMONICS</u>	<u>DESCRIPTION</u>	<u>BYTE</u>
1. ANL A, Rn	AND register to accumulator	1
2. ANL A, direct	AND direct byte to A	2
3. ANL A, @Ri	AND indirect RAM to A	1
4. ANL A, #data	AND immediate data to A	2
5. ANL direct, A	AND accumulator to direct byte	2
6. ANL direct, #data	AND immediate data to direct byte	3
7. ORL A, Rn	OR register to A	1
8. ORL A, direct	OR direct byte to A	2
9. ORL A, @Ri	OR indirect RAM to A	1
10. ORL A, #data	OR immediate data to A	2
11. ORL direct, A	OR accumulator to direct byte	2
12. ORL direct, #data	OR immediate data to direct byte	3
13. XRL A, Rn	XOR register to A	1
14. XRL A, direct	XOR direct byte to A	2
15. XRL A, @Ri	XOR indirect RAM to A	1
16. XRL A, #data	XOR immediate data to A	2
17. XRL direct, A	XOR accumulator to direct byte	2
18. XRL direct, #data	XOR immediate data to direct byte	3
19. CLR A	Clear accumulator	1
20. CPL A	Complement accumulator	1
21. RL A	Rotate accumulator left	1
22. RLC A	Rotate A left through carry.	1
23. RR A	Rotate accumulator right	1
24. RRC A	Rotate A right through carry	1
25. SWAP A	Swap nibbles within the A	1

eg: SWAP A

before [A] → 35H

after [A] → 53H

4] BOOLEAN VARIABLE MANIPULATION INSTRUCTION :

<u>MNEMONICS</u>	<u>DESCRIPTION</u>	<u>BYTE</u>
1. CLR C	Clear Carry	1
2. CLR bit	Clear direct bit	2
3. SETB C	Set carry	1
4. SETB bit	Set direct bit	2
5. CPL C	Complement carry	1
6. CPL bit	Complement direct bit	2
7. ANL C, bit	AND direct bit to carry	2
8. ANL C, /bit	AND complement of direct bit to carry	2
9. ORL C, bit	OR direct bit to carry	2
10. ORL C, /bit	OR complement of direct bit to carry	2
11. MOV C, bit	Move direct bit to carry	2
12. MOV bit, C	Move carry to direct bit	2
13. JC rel	Jump if carry is set	2
14. JNC rel	Jump if carry is not set	2
15. JB bit, rel	Jump if direct bit is set	3
16. JNB bit, rel	Jump if direct bit is not set	3
17. JBC bit, rel	Jump if direct bit is set & clear bit.	3

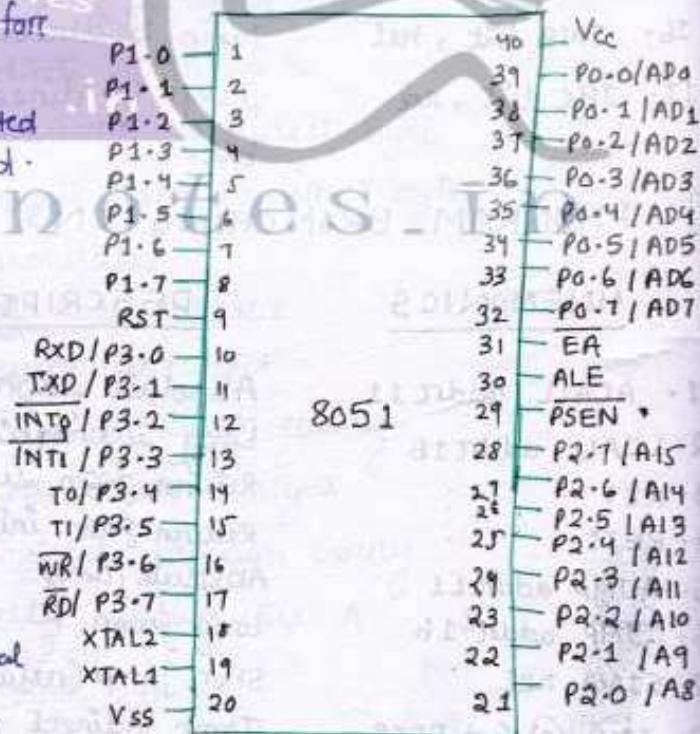
5] PROGRAM BRANCHING INSTRUCTIONS - IN

<u>MNEMONICS</u>	<u>DESCRIPTION</u>	<u>BYTE</u>
1. ACALL addr 11	Absolute subroutine call	2
2. LCALL addr 16	Long subroutine call	3
3. RET	Return from subroutine	1
4. RETI	Return from interrupt	1
5. AJMP addr 11	Absolute jump	2
6. LJMP addr 16	Long jump	3
7. SJMP rel	Short jump (relative address)	2
8. JMP @ A + DPTR	Jump indirect relative to DPTR	1
9. JZ rel	Jump if accumulator is zero	2
10. JNZ rel	Jump if accumulator is not zero	2

- | | | |
|--------------------------|---|---|
| 11. CJNE A, direct, rel | Compare direct byte to accumulator and jump if not equal. | 3 |
| 12. CJNE A, #data, rel | Compare immediate data to A and jump if not equal. | 3 |
| 13. CJNE Rn, #data, rel | Compare immediate to register and jump if not equal | 3 |
| 14. CJNE @Ri, #data, rel | Compare immediate to indirect byte and jump if not equal. | 3 |
| 15. DJNZ Rn, rel | Decrement register and jump if not zero | 3 |
| 16. DJNZ direct, rel | Decrement direct byte and jump if not zero | 3 |
| 17. NOP | No operation | 1 |

PIN DESCRIPTION OF 8051 :

- XTAL2 & XTAL1 : These pins are for crystal connection. In case of external clock, it must be connected to XTAL2 & XTAL1 is grounded.
- Vcc : It is connected to +5V power supply.
- VSS : It is grounded.
- Port 0 : 8-bit ^{real} bi-directional open drain I/O. Low order address and data bus is also multiplexed with port 0.
- Port 1 : 8-bit quasi bi-directional I/O. It is to be configured as either i/p or o/p. Writing a '1' to the port latch causes it to act as i/p.



NOTE : Quasi → Port pins are pulled high internally with fixed pull up resistors.

- RST: For resetting the device.

INTERRUPTS OF 8051 :

An interrupt is an event that interrupts normal prog. execution. They give us a mechanism to put-on-hold the normal prog. flow, execute a sub-routine and resume normal program.

This subroutine is called as an interrupt handler and is only executed when a certain event occurs.

Need of Interrupt :

When certain events occur, the ability to interrupt normal prog. execution makes it easier and much more efficient to handle certain condition.

MCS-51 supports 5 vectored interrupts sources. These are external interrupt 0, external interrupt 1, timer/counter 0 interrupt, timer/counter 1 interrupt & serial port interrupts.

When an interrupt is generated, the PC is pushed onto stack. Vectored address is loaded in the PC. As the vectoring takes place, that particular interrupt flag corresponding to the interrupt source is cleared by the h/w. In MCS-51, these flags are bits IEO, IE1, TFO, TF1, RI & TI.

The program now starts executing from the vectored locⁿ. This subroutine is called as ISS. The ISS ends with RETI instrucⁿ. The interrupt vector locations in 8051 are spaced out at every 8 bytes.

Lecture notes in

Interrupts :

<u>Interrupt</u>	<u>Flag affected</u>	<u>Vector</u>	<u>Cause of interrupt (if enabled)</u>
External interrupt 0 <u>INT0</u> pin	IE0	0003H	A high to low transition on pin <u>INT0</u> .
Timer/counter 0 interrupt	TFO	000BH	Overflow of timer/counter 0
External interrupt 1 <u>INT1</u> pin	IE1	0013H	A high to low transition on pin <u>INT1</u>
Timer/counter 1 interrupt	TF1	001BH	Overflow of timer/counter 1.
Serial port	RI + TI	0023H	When either TI or RI flag is set.

Initializing 8051 interrupts:

The interrupt enable (IE) register allows the programmer to enable interrupts as needed. This register IE is bit addressable.

EA	-	-	ES	ET1	EX1	ET0	EX0
----	---	---	----	-----	-----	-----	-----

EA : 9th is for enable all.

if EA = 0, disable all interrupts

EA = 1, allows each of individual interrupt to be enabled.

ES: Enable/disable serial port interrupt

if ES = 0, disable
ES = 1, enable (provided EA = 1)

ET1: Enable/disable timer interrupt 1

if ET1 = 0, disable
ET1 = 1, enable.

EX1: Enable/disable external interrupt 1

if EX1 = 0, disable
EX1 = 1, enable

ET0: Enable/disable timer interrupt 0

if ET0 = 0, disable
ET0 = 1, enable

EX0: Enable/disable external interrupt 0

if EX0 = 0, disable
EX0 = 1, enable.

Interrupt priorities:

The interrupt priority register gives the informⁿ about the interrupt priority.

-	-	PT2	PS	PT1	PX1	PT0	PX0
---	---	-----	----	-----	-----	-----	-----

Timer interrupt 2
(for 8032/8052 only)

serial i/p
interrupt

External interrupt 0

Timer interrupt 0

External interrupt 1

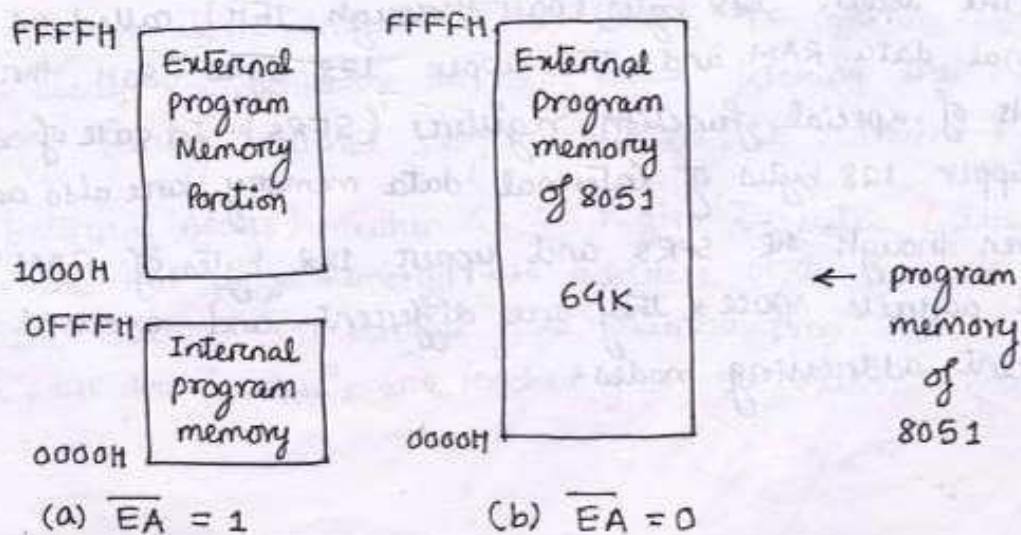
Timer interrupt 1

- When more than one interrupts are enabled, we can program the interrupt priority levels by setting or clearing the bits in SFR called interrupt priority.
- IP register is also bit addressable. If the bit is set, the particular interrupt will have high priority.
- A high priority interrupt can interrupt ~~can~~ the low priority interrupt, but a high priority interrupt will not be interrupted by the low priority interrupt.
- If the request of interrupts of 2 different priority levels occur simultaneously, naturally the interrupt having the high priority will be served.
- If the same priority level interrupts request simultaneously, then within each priority level there is a polling structure due to the inherent priority in the order shown in the figure.

$PT_2 > PS > PT_1 > PX_1 > PT_0 > PX_0$

MEMORY ORGANIZATION:

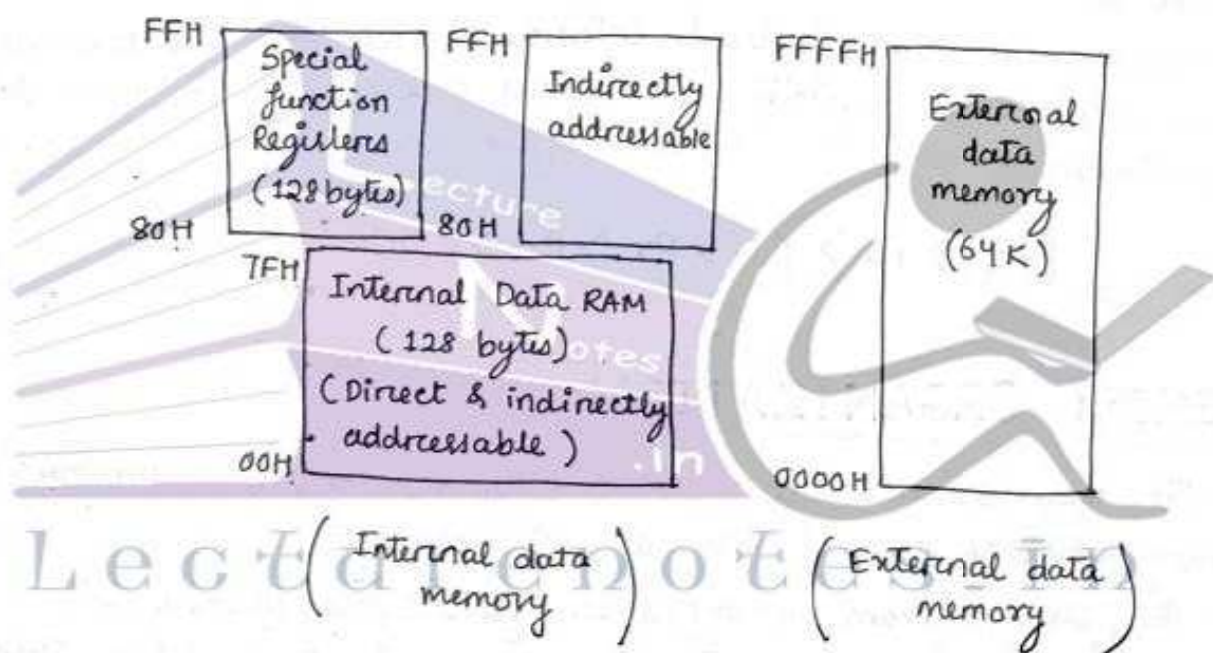
- The MCS-51 has 64K external data memory, 64K program memory and 256 bytes of internal data memory.
- The 64K program memory space of 8051 is divided into internal and external memory. If the \overline{EA} pin is high, then 8051 executes from the internal program memory until the address exceeds 0FFFH.



Status of \overline{EA} pin	Program execution from 0000H through 0FFFH	Program execution from 1000H through FFFFH
High (1)	Internal program memory	External prog. memory
Low (0)	External prog. memory	External prog. memory

- After that, locations 1000H through FFFFH are executed from the external memory portion.

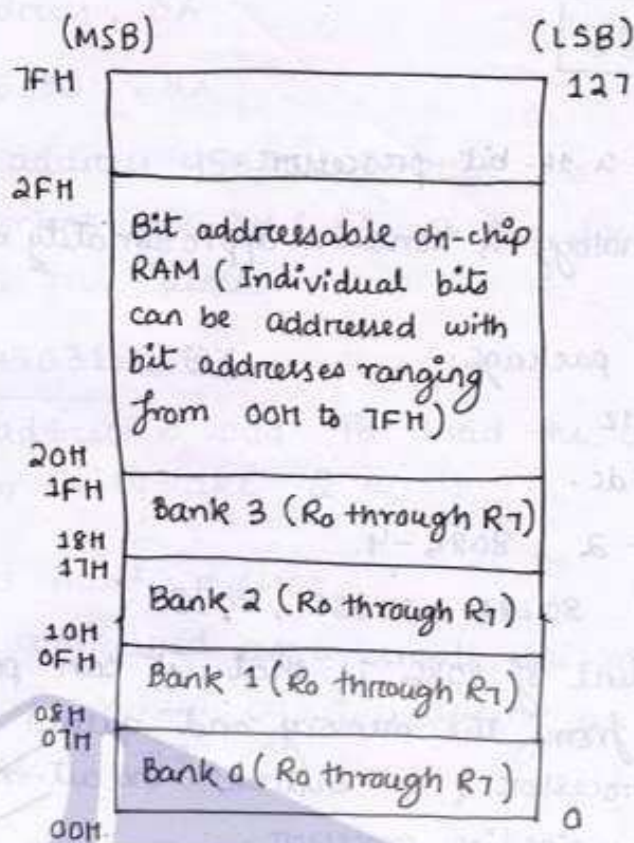
- If \overline{EA} pin is held low, then 8051 executes instructions from external memory only. The external 64K of data memory can be accessed using MOVX instruction.



- The internal data memory of 8051 is 256 bytes, which is divided into two parts again.

- The lower 128 bytes (00H through 7FH) called as the internal data RAM and the upper 128 bytes (80H through FFH) consists of special function registers (SFRs). In case of 8032/52, the upper 128 bytes of internal data memory are also addressable.

- Even though the SFRs and upper 128 bytes of RAM have the same address space, they are different and accessed through different addressing modes.



On-chip RAM (Lower 128 bytes)

7. Port 2 : 8-bit quasi bi-directional I/O port. It is multiplexed with the highest order address bus.
8. Port 3 : 8-bit quasi bi-directional I/O port. Alternate funcⁿ of Port 3 pins.
- | | |
|--------------------------------|---|
| P3.0 → RXD serial I/p | P3.4 → To Timer/Counter 0 external i/p |
| P3.1 → TXD serial o/p | P3.5 → T1 " " " |
| P3.2 → INT0 external interrupt | P3.6 → \overline{WR} external data memory write |
| P3.3 → INT1 external interrupt | P3.7 → \overline{RD} " " " read. |
9. ALE : Used for latching the low address byte during external memory access.
10. PSEN : Program store enable (\overline{PSEN}) is the O/P control signal, activated every six oscillator periods, while fetching the external prog. memory. During internal prog. execution, it remains high.
11. \overline{EA} : (External Access), when it is high, executes instrucⁿs from the internal prog. memory till address 0FFFH; beyond this address, the instrucⁿs are fetched from external prog. memory. If it is low, all the instrucⁿs are fetched from the external memory.

FEATURES: It is a 16-bit processor.

- It uses H-MOS technology & contains approximately around 29,000 transistors.
- It is a 40-pin IC package.
- Operating freq = 5MHz
- Power supply = 5V dc.
- Versions = 8086-2, 8086-4
- Advanced version = 80286, 80386, 80387.
- The one imp. feature of 8086 is that, it can pre-fetch upto 6 instrucⁿ bytes from the memory and queues them in order to speed up the processor & or instrucⁿ execution.
- It is also known as pipeline processor.
- It has 20 address lines, so the memory size of 8086 is $2^{20} = 1\text{MB}$.
- The address range is 00000H to FFFFFH.
- An IC - 8284 is used in 8086 to generate the required clock freq.
- 8086 uses segmented memory concept to handle the large memory space effectively.
- In 8086 the opcode fetching & the execution of instrucⁿ take place simultaneously. Hence, it is known as a pipeline processor.
- A 16-bit word in 8086 use 2 consecutive memory locⁿ. The LSB of the word will be stored at lower address & MSB of the word will be stored at higher address.
- 8086 is having a powerful instrucⁿ set which includes multiplication, division & various arithmetic oprⁿ of decimal no.
- Imp - 8086 can read a 16-bit word in one oprⁿ if the 1st byte of the data is at an even address.
- It takes 2 consecutive memory oprⁿs for a 16-bit word if the 1st byte of the data is at an odd address.

eg: MOV address, BX

MOV 45030H, BX

Here the address 45030H is even. So, the move instruction loads the content of BH & BL to the locⁿ 45031H & 45030H respectively in one access.

eg: MOV 45031H, BX

Here the address is odd. To load the content of BL & BH the processor will take 2 access.

8086 supports multiprocessor.

8086 can be configured as a small uni-processor system or as a multiprocessor system by introducing a controlling pin or signal pin "MN/ $\overline{\text{MAX}}$ ".

- If $\text{MN}/\overline{\text{MX}} = 1$, processor is operating in min^m mode i.e. as an uniprocessor.

if $\text{MN}/\overline{\text{MX}} = 0$, operating in max^m mode as a multiprocessor.

BLOCK ARCHITECTURE OF 8086:

The whole architecture of 8086 consists of 2 sections.

- 1) Execution unit (EU)
- 2) Bus interfacing unit (BIU)

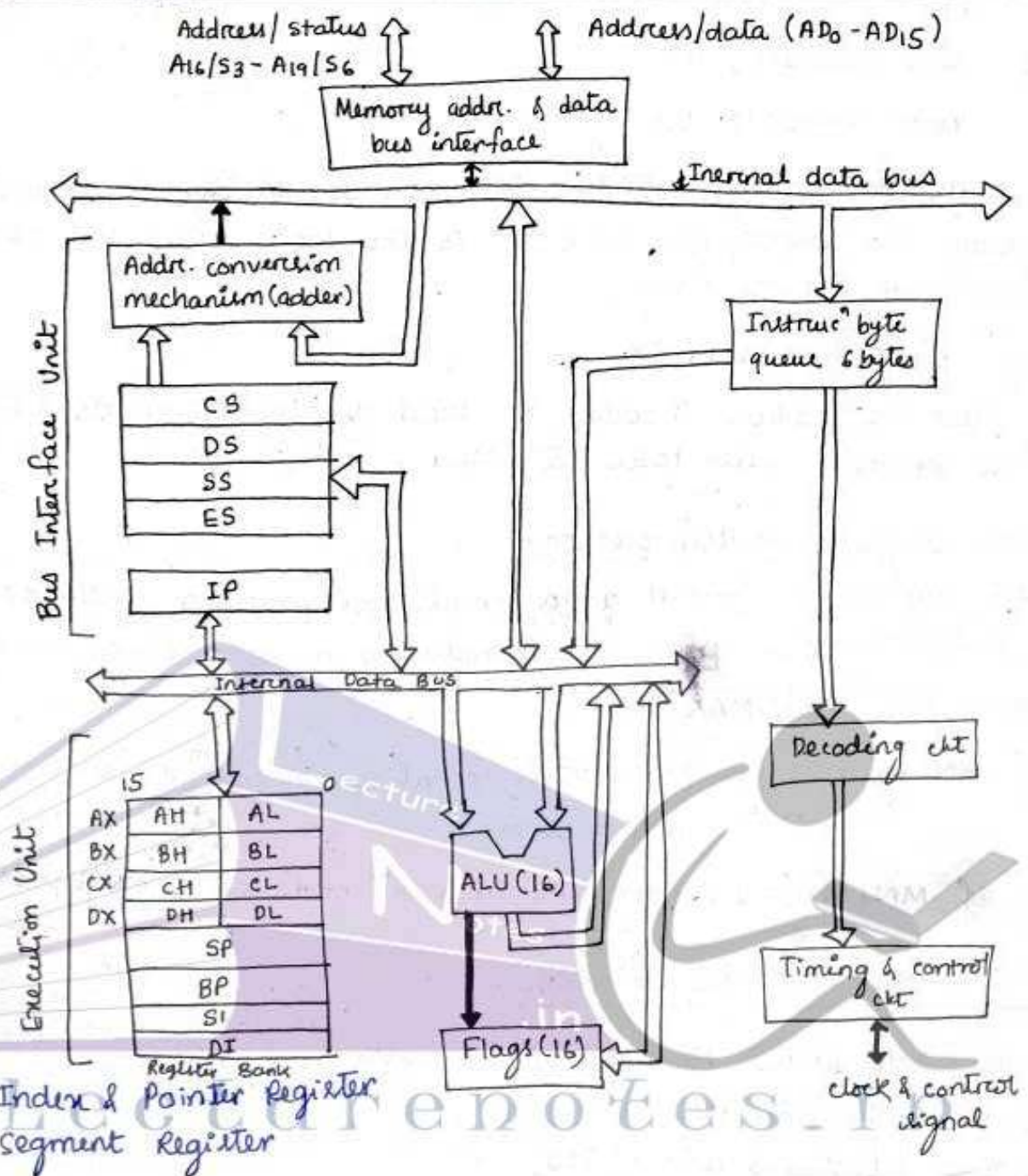
EXECUTION UNIT:

The execution unit of 8086 handles all controlling funcⁿs, such as -

→ Telling the BIU, where to fetch instruction, where to read a data & also performs the instruction decoding & execution.

→ The EU is further subdivided into.

1. Instruction decoder & control subsystem (IDCS)
2. ALU
3. Flag Register
4. GPR



5. Index & Pointer Register
6. Segment Register

IDCS:

- The instrucⁿ decoder translates the received instrucⁿ from BIU & performs the necessary oprⁿ.
- The IDCS performs the proper sequencing & proper synchronization of signals & generates the necessary controlling signals for the smooth execuⁿ of an instrucⁿ.

ALU: It can work as an 8-bit or as a 16-bit processor.

Imp

GPRS: 8086 has 4 GPRS namely AX, BX, CX, DX.

- Each reg. is a 16-bit register and can be subdivided into AH & AL where AH \rightarrow higher order bytes of AX
AL \rightarrow lower " " " "

- General name of AX is accumulator.

BX: Called as Base register.

- Also called as memory pointer

CX: Counter register

DX: May be used as an implicit operand or destination in some instructions.

- It is used to hold the higher 16-bit result in a 16×16 operation.
- It also stores the remainder in a division operation & dividend before division.

01.04.10

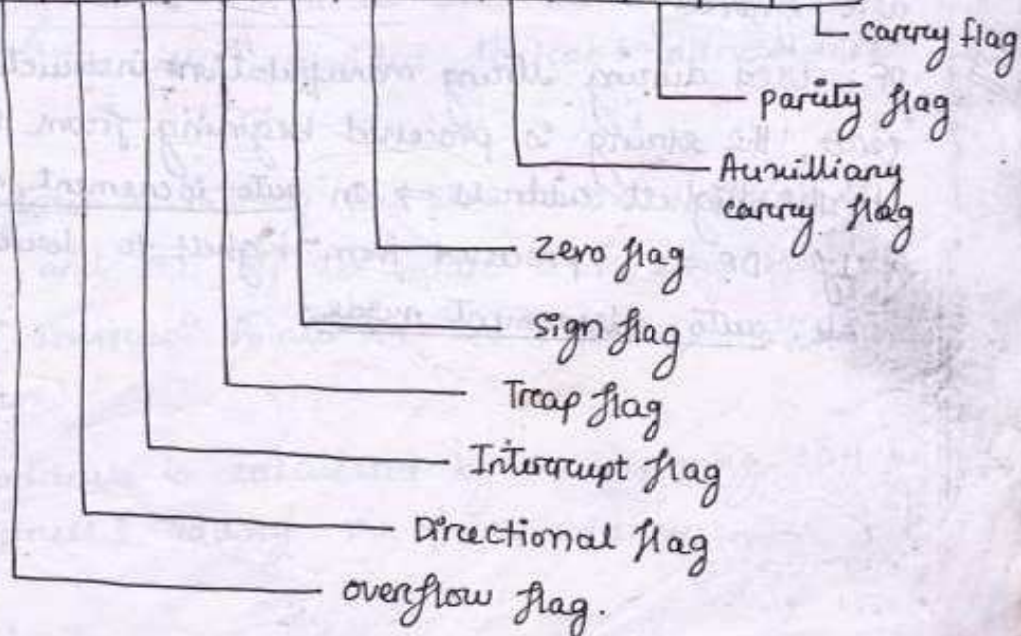
FLAG REGISTER:

- 16 bit register which contains 9 active flags.
- This 9 active flags are divided into 2 groups.

(i) conditional flag

(ii) control flag.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF



conditional flag

CF

PF

ACF

ZF

SF

control flag

TF

IF

DF

OF

ACF: If any carry is generating from 3rd to 4th bit or 7th to 8th bit during addⁿ or any borrow is required from 4th to 3rd bit or 8th to 7th bit, $AC \rightarrow 1$. otherwise $AC \rightarrow 0$.

OF: If the result of any oprⁿ is not large enough to accomodate the result then $OF \rightarrow 1$. (if carry generated at 15th bit). This is applicable for signed binary numbers because here one additional bit is provided with the number to represent signed binary numbers.

TF: (Trap/Trace)

If this flag is set, the processor enters into the single step execution mode. In other words a Trap interrupt is generated after execution of each instrucⁿ.

IF: If this is set, the maskable interrupts are recognized by CPU. else when $IF = 0$, all the maskable interrupts can be ignored.

DF: Used during string manipulation instructions. If this flag is zero, the string is processed beginning from the lowest address to the highest address \Rightarrow In auto increment mode.

If $DF = 1$, processed from highest to lowest address i.e. auto decrement mode.

POINTER & INDEX REGISTER:

- 8086 has 2, 16-bit index reg. & 2, 16-bit pointer reg.
- The index reg. are SI (Source index)
DI (Destination index)
- Pointer reg → Stack pointer
→ Base pointer
- All are used to hold the offset address.

SEGMENT REGISTER:

8086 has 4 segment registers (but 16 segments)

- Code segment (CS)
- Data segment (DS)
- Stack segment (SS)
- Extra segment (ES)
- The total memory of 8086 (1MB) is divided into 16 segments each of 64 KB.
- The address line of 8086 is of 20-bits. So, searching for a memory locⁿ from 1MB of memory space will require a longer time.
- Hence a concept, segmented memory management is used to make the access faster.
- The bus interfacing unit calculate the 20-bit physical address internally using the user provided logical address and off-set address.
- The logical address is nothing but the 16 bit content of CS, DS, ES, SS. and the off-set addr. is nothing but the content of IP (Instruction Pointer), SI (source index), DI (Destination index).
- The physical address is calculated by multiplying 10H with the logical address & adding the off-set address.

eg: $[CS] = 1234H$

$[IP] = 003CH$

Then physical address = $1234H \times 10H + 003CH$

or $= 12340H + 003CH$

effective address = $1237CH$

If ~~the~~ DSR content is the logical addr, then SI/DI content \rightarrow off. set addr.
BUS INTERFACING UNIT:

This section consists of following sub section

1. Instrucⁿ queue
2. " pointer
3. ^{Bus} Control & address generator

1. Instruction Queue:

- 8086 uses a method called pipe line method to speed up the opriⁿ.
- It has a 6 byte FIFO register. When execution unit is decoding or executing any instrucⁿ, at that time it doesn't require its buses. Hence the buses are free during execution. But in case of 8086, the BIU unit can funcⁿ independently.
- During the time of execution & decoding, BI unit takes the control over system buses & ~~fetch~~ pre-fetches the instrucⁿ bytes from the memory for execution unit.
- This pre-fetched instrucⁿs are stored in a 6-byte reg. known as instruction queue register.
- When execution unit is free, it can directly access the IQR.

Imp.

2. Instruction Pointer:

It holds the off. set address (16-bit) for calculation of 20-bit physical address.

3. Bus Control & Address generator :

- Used by BIU to control the external buses.
- It also generates the necessary timing & control signals and also synchronization signal for the smooth opⁿ of instrucⁿ.

PIN DESCRIPTION OF 8086 :



Lecture notes in

The whole pins of 8086 are categorised in 3 groups.

1. PINS common to both min^m & max^m mode.

2. Special PINS for min^m mode.

3. Special PINS for max^m mode.

1. COMMON PINS :

1. $AD_0 - AD_{15}$: Bidirectional data bus. Carries address during the 1st clk cycle^(T₁) and carries data for remaining clk cycles.

2. $A_{16}, A_{17}, A_{18}, A_{19}$:

2. $A_{16}/S_3, A_{17}/S_4, A_{18}/S_5, A_{19}/S_6$: These are time multiplexed address & status lines. During T₁, there are the MSB bits for of the address for any memory operⁿ. If these bits are low, then it is for I/O operⁿ.

[In T₁ if 1 → Memory operⁿ
0 → I/O operⁿ]

- The status informⁿ is available from those lines during T₂, T₃, clk periods onwards, Since these pins are time multiplexed.

- The additional funcⁿ of S₃ & S₄ is to indicate a segment register for any memory access.

S₄

S₃

0

0

extra segment / alternate data

0

1

Stack segment

1

0

Code segment

1

1

Data segment

3. **BHE / S7** : BHE \rightarrow Bus high enable.

- This signal is used to indicate the transfer of data over the higher order data bus. (i.e. D₈ to D₁₅)
- It is an active low signal & responsible for data transfer of higher order bytes.
- S7 is a status signal which indicates for certain acknowledgement signal. (for interrupts)

4. **\overline{RD}** : Active low signal and shows the states for T₂, T₃, T₄ of any read cycle. The signal remains tristated during the 'hold acknowledge'.

5. **READY** : This is the acknowledgement from the slow devices / memory that they have completed the data transfer.

6. **INTR** : Level triggered signal, internally synchronized. This is sampled during the last clock cycle of each instrucⁿ to determine the availability of the request.

7. **TEST** : This i/p is examined by a WAIT instrucⁿ.

If $\overline{TEST} = 0$, execution will continue

$\overline{TEST} = 1$, processor will be in idle state.

8. **NMI** (Non-maskable interrupt) : This is an edge-triggered i/p which causes a Type 2 interrupt. A transition from low to high initiates the interrupt response at the end of the current instrucⁿ. It is internally synchronized.

9. **RESET** : It causes the processor to terminate the current activity and start execution from FFFFH. It must be active for at least 4 clock cycles. It restarts execuⁿ when RESET = 0.

10. **CLK** : It provides the basic timing for processor opⁿ & bus control activity. $\delta = 33\%$ (δ = duty cycle). Freq. range 5MHz - 10MHz.

11. **VCC** : +5V power supply for the opⁿ of internal ckt.

12. **GROUND**: Ground for internal ckt.
13. **MN/ \overline{MX}** : ~~For~~ The logic levels at this pin decides whether the processor operates in min^m mode (single processor) or max^m mode (multiprocessor).

2. PINS FOR MINIMUM MODE :

1. **M/ \overline{IO}** : If it is 1, memory will be selected.
0, I/O " " "
2. **\overline{INTA}** : when $\overline{INTA} = 0$, the processor has accepted the interrupt. This signal is used as a read strobe for interrupt acknowledge cycles.
3. **ALE**: It indicates the availability of the valid address on the address/data lines, and is connected to latch enable i/p of latches.
4. **DT/ \overline{R}** : Data transmit/receive
This O/p is to decide the dirⁿ of data flow through the transceivers (bidirectional buffer). When processor sends data, this signal is high. When processor receives data it is low.
5. **HOLD**: ~~at~~ when it is high, it indicates to the processor that another master is requesting the bus access. It is an asynchronous i/p & it should be synchronized externally.
6. **HLDA**: After receiving HOLD, the μP issues the acknowledge signal on HLDA pin. when $HOLD = 0$, HLDA is made 0.
7. **\overline{WR}** : Active low signal, Performs write opⁿ.

8. **DEN** : Data enable

- This signal indicates the availability of valid data over address bus & data bus.
- This is used to enable the transceiver to separate the data from the multiplexed address/data signal.

3. **PINS FOR MAXIMUM MODE :**

1. **$\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$** : There are 3 status lines used to indicate various opⁿs carried out by processor.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Operation
0	0	0	Interrupt acknowledgement
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Hlt
1	0	0	Code access
1	0	1	Read Memory (MR)
1	1	0	Write memory (MW)
1	1	1	Passive/Inactive

2. **LOCK** : It indicates that other system bus masters will be prevented from gaining the system bus while the **LOCK** = 0.

3. **QS_0 , QS_1** : There are queue status pins.

- These lines give the informⁿ about the status of the code prefetched queue (Instrucⁿ Queue Reg.)

QS_1	QS_0	Operation
0	0	NOP
0	1	1st byte of the opcode from the queue reg.
1	0	Empty the queue
1	1	Subsequent byte of the QR

4. $\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$: These are for request/grant.

- These pins are used by other local bus masters in max-mode to force the processor to release the local bus after the processor completed its current bus cycle.
- These are bi-directional pins.
- $\overline{RQ}/\overline{GT_0}$ has higher priority than $\overline{RQ}/\overline{GT_1}$

H.W

What is memory segmentation and physical memory organization of 8086?

The

MEMORY SEGMENTATION:

The memory in an 8086/88 based system is organized as segmented memory. In this scheme, the complete physically available memory may be divided into a number of logical segments. Each segment is 64K bytes in size and is addressed by one of the segment registers. The 16-bit contents of the segment register actually point to the starting location of a particular segment. To address a specific memory location within a segment, we need an off-set address. The off-set address is also 16-bit long so that the max^m offset value can be FFFFH. and the max^m size of any segment is thus 64K locations.

The CPU 8086 is able to address 1MB of physical memory. The complete 1MB memory can be divided into 16 segments, each of 64KB size. The addresses of the segments may be assigned as 0000H to F000H respectively. The offset address values are from 0000H to FFFFH so that the physical address ranges from 00000H to FFFFFH. The main advantages of the segmented memory scheme are:

1. Allows the memory chip capacity to be 1MB although the actual addresses to be handled are of 16-bit size.
2. Allows the placing of code, data and stack portions of the same program in different parts (segments) of memory, for data and code protection.

3. Permits a program and/or its data to be put into different areas of memory each time the program is executed. i.e. provision for relocation is done.

PHYSICAL MEMORY ORGANIZATION:

- In 8086, the 1MB memory is physically organised as an odd bank and an even bank each of 512 KB, addressed in parallel by the processor.

- Byte data with an even address is transferred on D₇-D₀, while the byte data with an odd address is transferred on D₁₅-D₈ bus lines.

- The processor provides 2 enable signals, BHE and A₀ for selection of either even or odd or both the banks. The instrⁿ's stream is fetched from memory as words and is addressed internally by the processor as necessary. In other words, if the processor fetches a word (consecutive 2 bytes) from memory, there are different possibilities, like:

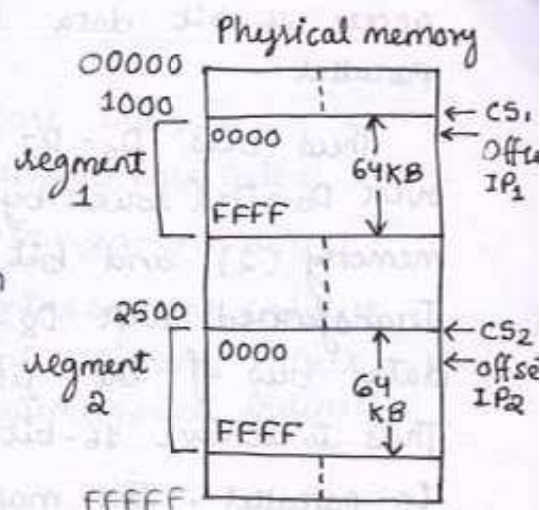
1. Both the bytes may be data operands.
2. Both the bytes may contain opcode bits.
3. One of the bytes may be opcode while the other may be data.

- The opcodes and operands are identified by the internal decoder ct which further derives the signals. These act as i/p to the timing and control unit. The timing and control unit then derives all the signals required for execution of the instrⁿ.

- To read or write a complete word from/to memory, if it is located at an even address, only one read or write cycle is required. If the word is located at an odd address, the first read or write cycle is required for accessing the lower byte while the 2nd one is required for accessing the upper byte, i.e. 2 bus cycles are required.

8086 is a 16-bit microprocessor and hence can access 2 bytes of data in one MR/MW/IOR/IOW opⁿ. But memory chips are of 1 byte size. So, 2 consecutive memory locⁿs are used.

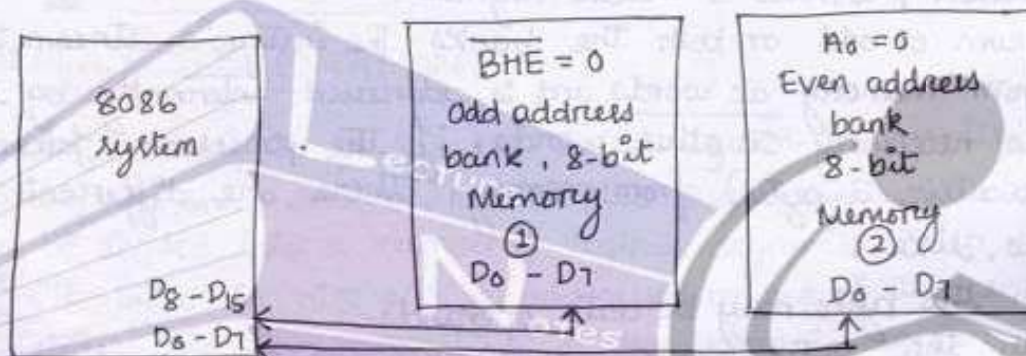
(Lower byte is stored in 1st memory locⁿ, higher byte is stored in the next memory locⁿ)



A map of an 8086 memory system starts at 00000H and ends at FFFFFH. 8086 being a 16-bit processor is expected to access 16-bit data to/from 8-bit ~~accessed~~ memory chips in parallel.

Thus bits $D_0 - D_7$ of a 16-bit data will be transferred over $D_0 - D_7$ (lower byte) of 16-bit data bus to/from 8-bit memory (2) and bit $D_8 - D_{15}$ of the 16-bit data will be transferred over $D_8 - D_{15}$ (higher byte) of the 16-bit data bus of the μP , to/from 8-bit memory (1).

Thus to achieve 16-bit data transfer using 8-bit memories, in parallel, the map of the complete system byte memory addresses is divided into 2 memory banks.



<Physical memory organization>

Lecture notes in

ADDRESSING MODES OF 8086:

The various addressing modes are,

1. Immediate
2. Direct
3. Register
4. Register indirect
5. Index
6. Register relative
7. Base Index
8. Relative base index
9. Intra segment direct
10. Intra segment indirect
11. Inter segment direct
12. Inter segment indirect

extra:

13. I/O
14. Port
15. Implied
16. String

1. IMMEDIATE ADDRESSING MODE:

Here data is present in the "instruct" itself. Data size may be 8-bit/16 bit. eg: `MOV AX, 0050H`
`MOV AH, 34H`
`MOV AL, 52H`

2. DIRECT ADDRESSING MODE:

In this case, a 16-bit memory address (off-set address) will be provided in the "instruct" itself.

eg: `MOV AX, [0050H]`

Here the default segment is DS. The actual physical address $PA = 10H \times DS + \text{off-set address}$.

3. REGISTER ADDRESSING MODE:

In this addressing mode, data is stored transfer takes place betⁿ 2 reg. eg: `MOV AX, BX`.

* Except IP. (It can't be used in this mode)

4. REGISTER INDIRECT ADDRESSING MODE:

The off-set addr. of the data is present in the reg.

eg: `MOV AX, [BX]`

The content of BX = off-set addr.

Here the default segment register is either DS or ES.

Here SI & DI can be used to store the off-set address.

$$PA = 10H \times DS + [BX]$$

5. INDEX ADDRESSING MODE:

Here the off-set addr. is stored in one of the reg. index reg. Default segment is DS.

eg: `MOV AX, [SI]`

$$PA = 10H \times DS + [SI] \quad \text{or} \quad PA = 10H \times DS + [DI]$$

6. REGISTER RELATIVE

In this addressing mode, the data is available at an effective address formed by adding an 8-bit/16-bit displacement with the content of one of the reg. like BX, BP, SI, DI with the default segment DS or ES.

eg: `MOV AX, 50H [BX]`

$$PA = 10H \times DS + 50H + [BX]$$

7. BASED INDEX

Here the effective address of the data is formed by adding the content of a base register (BX or BP) with the content of index reg. (SI/DI) along with the default segment (DS/ES).

eg: `MOV AX, [BX][SI]`

$$PA = 10H \times DS + [BX] + [SI]$$

8. RELATIVE BASED INDEX ADDRESSING MODE :

If we put a displacement of 8-bit/16-bit to based index addr. mode, then it will be known as relative based index a. mode

eg: `MOV AX, 50H [BX][SI]`

$$PA = 10H \times DS + 50H + [BX] + [SI]$$

9. IMPLIED ADDRESSING MODE :

No operand in the instrucⁿ.

eg: `CLR, HLT`

10. IO/PORT ADDRESSING MODE :

There are 2 types of port addressing mode.

1. Fixed/Direct port mode (DPM)
2. Variable/Indirect port mode (IPM)

(a) DPM: In this case 1-byte port addr. will be provided in the instrucⁿ. eg: `OUT 06H, AL ; IN AL, 06H`

(b) IPM: The port address will not be provided in the instrucⁿ. The addr. of the port will be taken from DX.

eg: `OUT DX, AL`
`IN AL, DX`

Q. The contents of different registers are given below. Form effective addresses for different addressing modes.

offset (displacement) = 5000H

`[AX] - 1000H, [BX] - 2000H, [SI] - 3000H, [DI] - 4000H, [BP] - 5000H.`

`[SP] - 6000H, [CS] - 0000H, [DS] - 1000H, [SS] - 2000H, [IP] - 7000H.`

Shifting a number 4 times is equivalent to multiplying it by 16 or 10H.

(i) Direct addressing mode

MOV AX, [5000H]

DS : OFFSET \Rightarrow 1000H : 5000H

10H * DS \Rightarrow 10000

offset \Rightarrow +5000

15000H - Effective address.

(ii) Register indirect : MOV AX, [BX]

DS : BX \Rightarrow 1000H : 2000H

10H * DS \Rightarrow 10000

[BX] \Rightarrow +2000

12000H - Effective addr.

(iii) Register Relative:

MOV AX, 5000[BX]

DS : [5000 + BX] \Rightarrow 1000 : [5000 + 2000]

10H * DS \Rightarrow 10000

offset \Rightarrow +5000

[BX] \Rightarrow +2000

17000H - Effective address.

(iv) Based indexed:

MOV AX, [BX][SI]

DS : [BX + SI]

10H * DS \Rightarrow 10000

[BX] \Rightarrow +2000

[SI] \Rightarrow +3000

15000H - Effective addr.

(v) Relative Based indexed:

MOV AX, 5000[BX][SI]

DS : [BX + SI + 5000]

10H * DS \Rightarrow 10000

[BX] \Rightarrow +2000

[SI] \Rightarrow +3000

offset \Rightarrow +5000

1A000 - Effective address.

FEATURES:

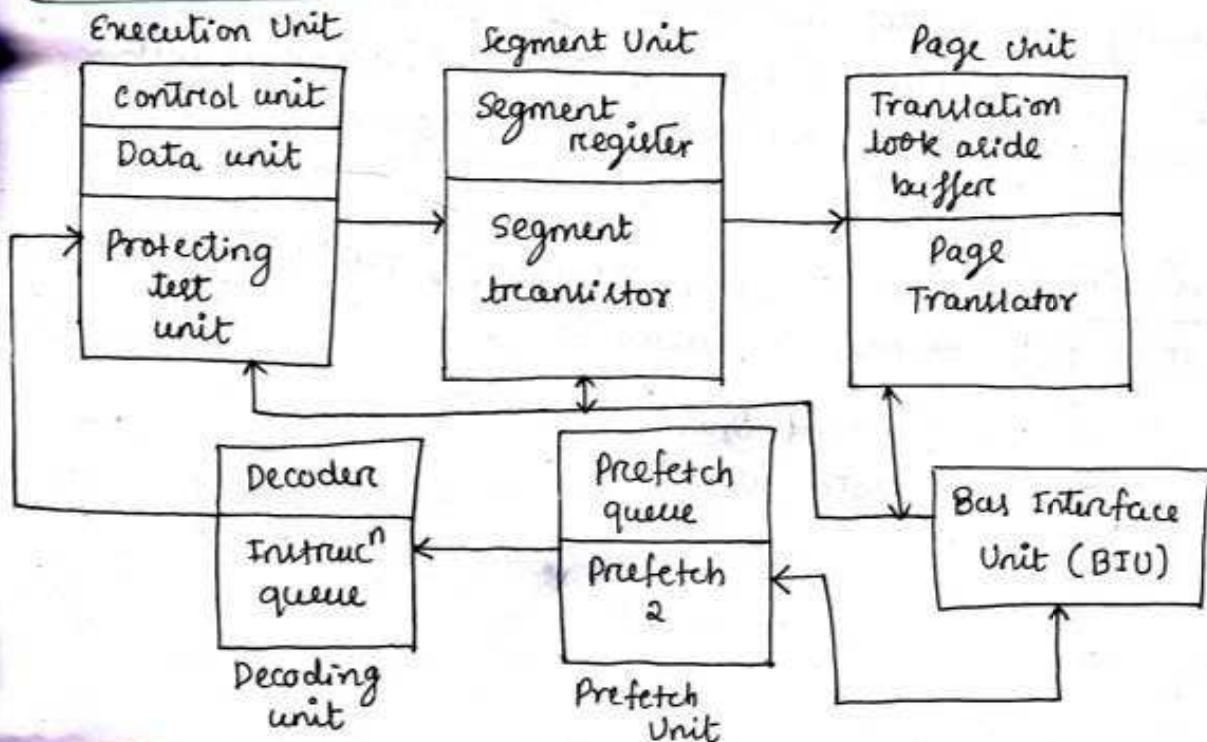
- It is a 32-bit processor.
- It uses HMOS-3 technology.
- It is having a very high operating freq. i.e. 20 - 33 MHz.
- It responds 8-bit / 16-bit / 32-bit.
- The no. of addr. line is 32. So, it can address upto $2^{32} = \underline{4GB}$.
- It can access upto 64 TB of virtual memory.

$$1 \text{ TB} = 2^{40}$$

$$64 \text{ TB} = 2^{46}$$

(Real memory)

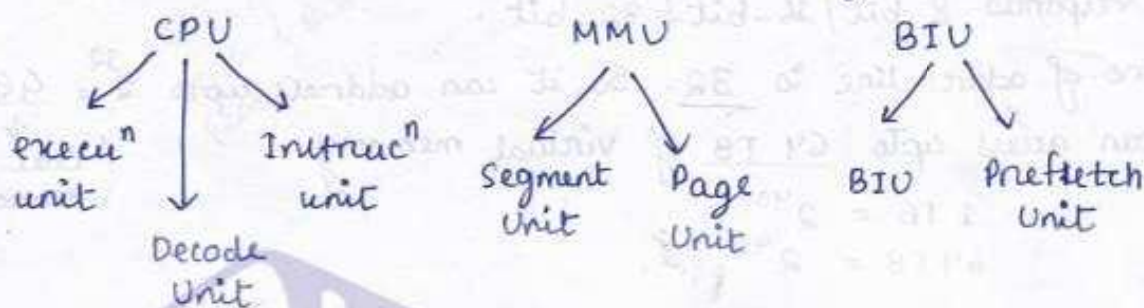
- It provides multitasking supports, memory management, pipeline architecture and a high speed interface on a memory chip.
- It is known as pipeline processor as fetching, decoding, execuⁿ, memory mgmt, bus access for several instrucⁿ can operate at a time.
- It is having 129 instrucⁿs.
- Its other versions are, 80386 SX, 80386 DX
- It has 132 pin IC package. (QIP - Quadra inline packaging)

ARCHITECTURE:

- The whole architecture is divided into 3 section.
- Those 3 sections are

- ✓ 1. CPU
- ✓ 2. Memory management
- ✓ 3. Bus Interfacing

Again each secⁿ is divided into no. of sub-secⁿ.

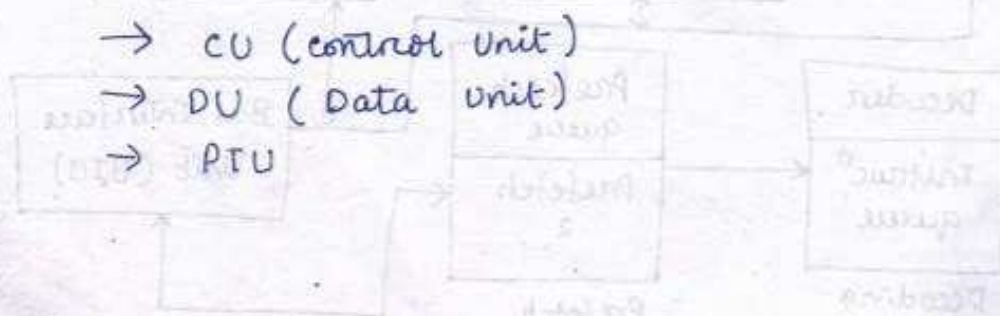


BIU: This interfaces 80386 with the memory and I/O devices. Based on the internal request for fetching instrucⁿ, the port prefetch unit of 80386 generates addr, data and control signals for the whole bus cycles.

This unit has a priority checker to resolve the priority of various bus requests.

Port prefetch Unit: It prefetches instrucⁿ when BIU not executing bus cycles. It stores the instrucⁿ in a 16-byte IQR for execution by the instrucⁿ decoder unit. It ~~operation~~ operates in FIFO technique.

Execⁿ Unit: This unit processes the instrucⁿ's which are residing in the IQR. It has 3 subunits.



Control Unit: It contains micro codes and various parallel h/w's for faster arithmetic opⁿ like. Multiplicaⁿ, divisⁿ & effective addr. calculⁿ.

Data Unit: This includes ALU, GPRs for performing various logical opⁿ

PU: This unit check for protecⁿ violation under the control of certain microcodes.

Segment Unit: In order to manage the huge memory of 80386 the total memory is divided into certain segments. Since 80386 has a virtual memory of 64 TB, to handel this memory the min^m size of each segment is 4GB.

Page Unit: It uses certain techniques for calculating the physical address. Paging unit works under the control of segment unit. Each segment is further sub-divided into numbers of pages.

Decode Unit: It is used to decode the instrucⁿ before going for execution.

12th April

Lecture notes in REGISTER ORGANIZATION OF 80386:

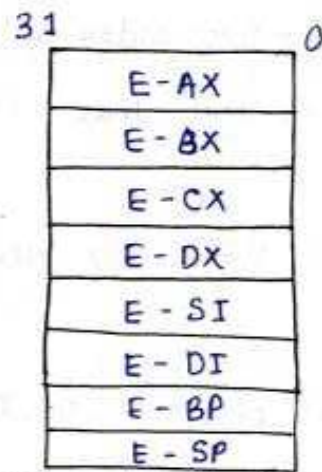
80386 has 16 different types of registers out of which there are 8, 32-bit registers namely,

E-AX	E-SI
E-BX	E-DI
E-CX	E-BP
E-DX	E-SP

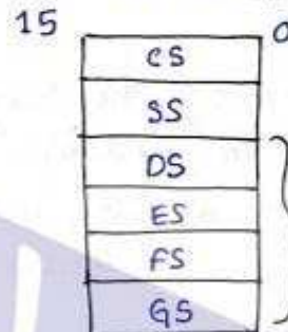
(E → extra)

80386 has six 16-bit registers.

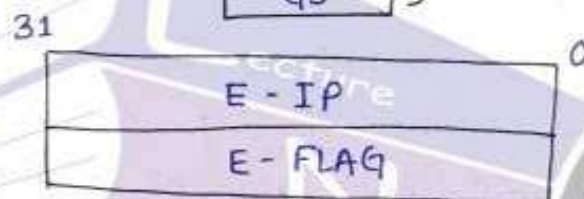
CS, SS, DS, ES, FS, GS



General data and address register

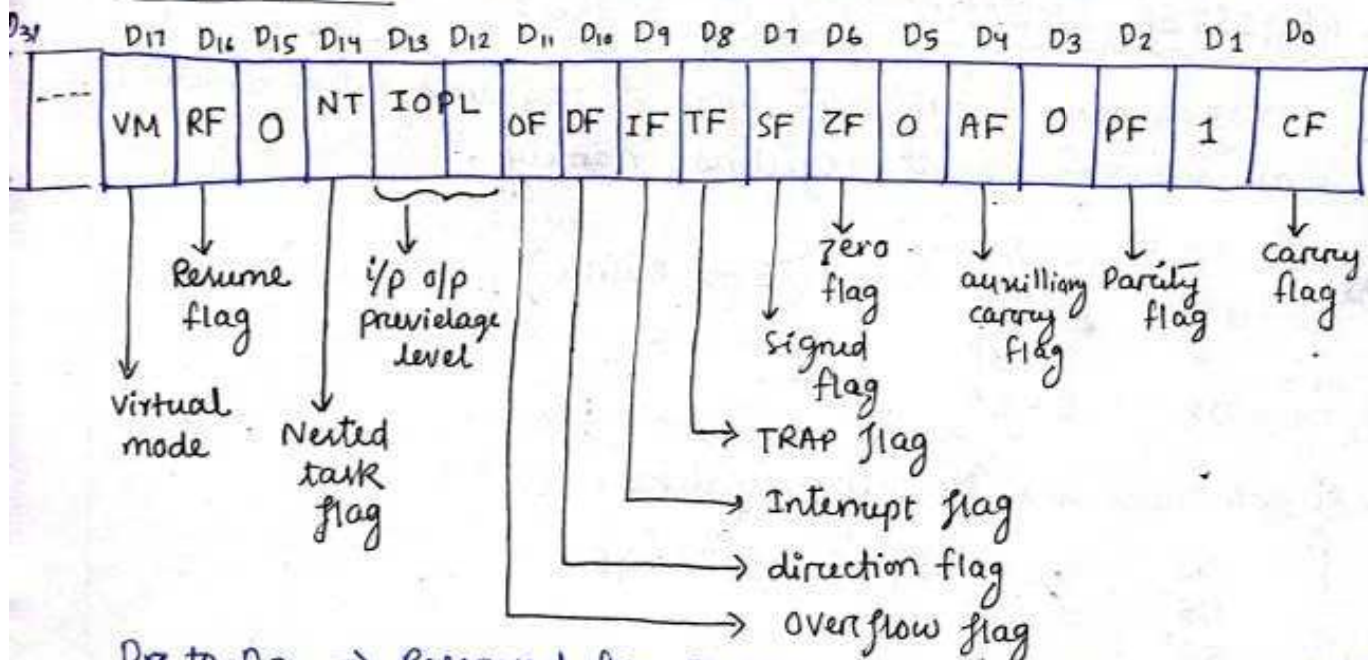


data segment



There are another two 32-bit reg. present in 80386 namely E-IP, E-FLAG.

FLAG REGISTER:



D8 to D31 → Reserved for INTEL
so, we can put 0.

* wherever 0 is present
i.e. reserved for INTEL.

- Reserved bits are D_3, D_5, D_{15} and D_{17} to D_{31} .
These are reserved for INTEL. Hence are always set to 0.

RF: If $RF = 1$, 80386 ignores all debugging faults.
 $RF = 0$, processor has to give service to the debugging

VM: If $VM = 1$, 80386 executes 8086 operation.
 $VM = 0$, 80386 operates 80386 opⁿ (protected)

NT: Related to the return operation.

if $NT = 0$, normal RET
 $NT = 1$, indirect RET

IOPL: It indicates the privilege level of the ^{current} I/O opⁿ.

- It is a 2-bit field that supports 80386 protection features for I/O opⁿ.
- This field defines the privilege level needed to execute any I/O instrucⁿ.

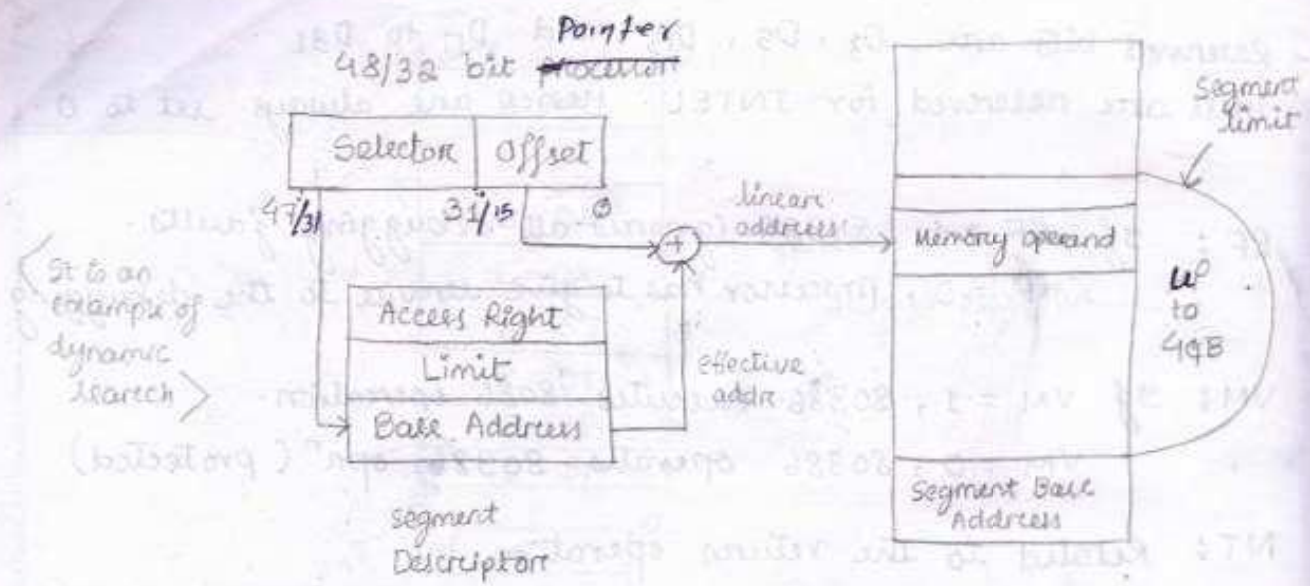
MODES OF OPERATION OF 80386:

80386 operates in 3 different modes.

1. Real address mode
2. Protected
3. Virtual

PROTECTED ADDRESS MODE:

- It is a normal 32-bit operation mode of 80386. All the instrucⁿs, features of 80386 are available in this mode.
- Here 80386 can address 4GB of physical/real memory and 64 TB of virtual memory.
- In this mode the content of segment reg. used as selectors to address descriptors which contains segment limit, base address.

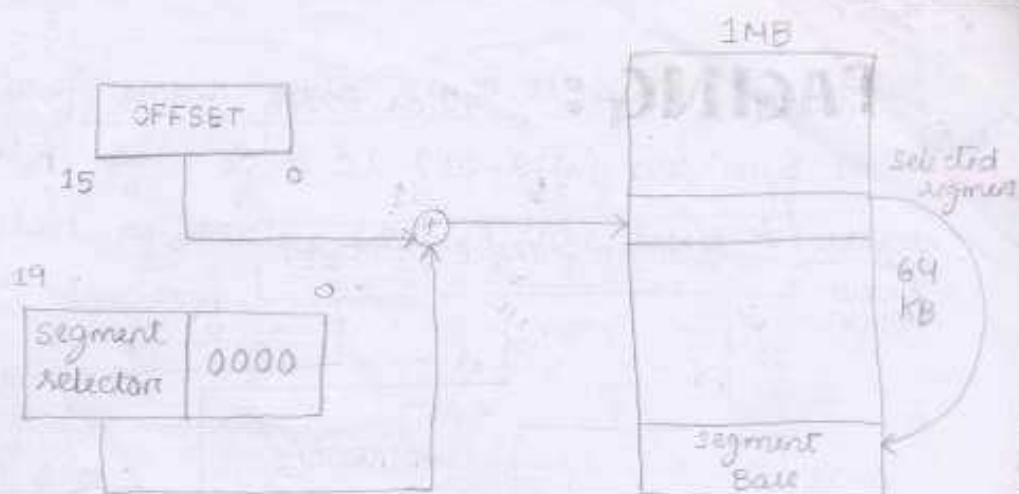


(Protected mode addressing without paging)

- The effective addr. is added with the segment base addr. to calculate linear address. This linear addr. is further used as physical address if paging unit is disabled. Otherwise this paging unit converts the linear addr. into physical address.

REAL ADDRESSING MODE:

- Also known as real mode.
- In this mode of opⁿ, the μP depends upon a 16-bit unit.
- This mode appears to the programmer as a simple 8086 with a few new instrucⁿs.
- In real mode, 80386 can address up to 1MB of physical memory with the address line A_0 to A_{19} .
- Here the paging unit is disabled. Hence the real address is same as that of the physical address.
- To form the physical address, appropriate segment reg. content (16-bit) are shifted left by 4 bits & then added with 16-bit offset address.



VIRTUAL ADDRESSING MODE :

15th April

This is known as V86 mode.

It is the mode in which 80386 can go back & forth repeatedly betⁿ V86 mode & protected mode in a faster speed.

In its protected mode of opⁿ, 80386 provides a virtual 8086 env^o operating environment to execute 8086 prog.

In real mode, we can execute the 8086 prog. but once 80386 enters into protected mode from real mode it can't return back to real mode without "resetting" the processor.

V86 mode of opⁿ of 80386 offers an advantage of executing the 8086 prog. even if it is in protected mode.

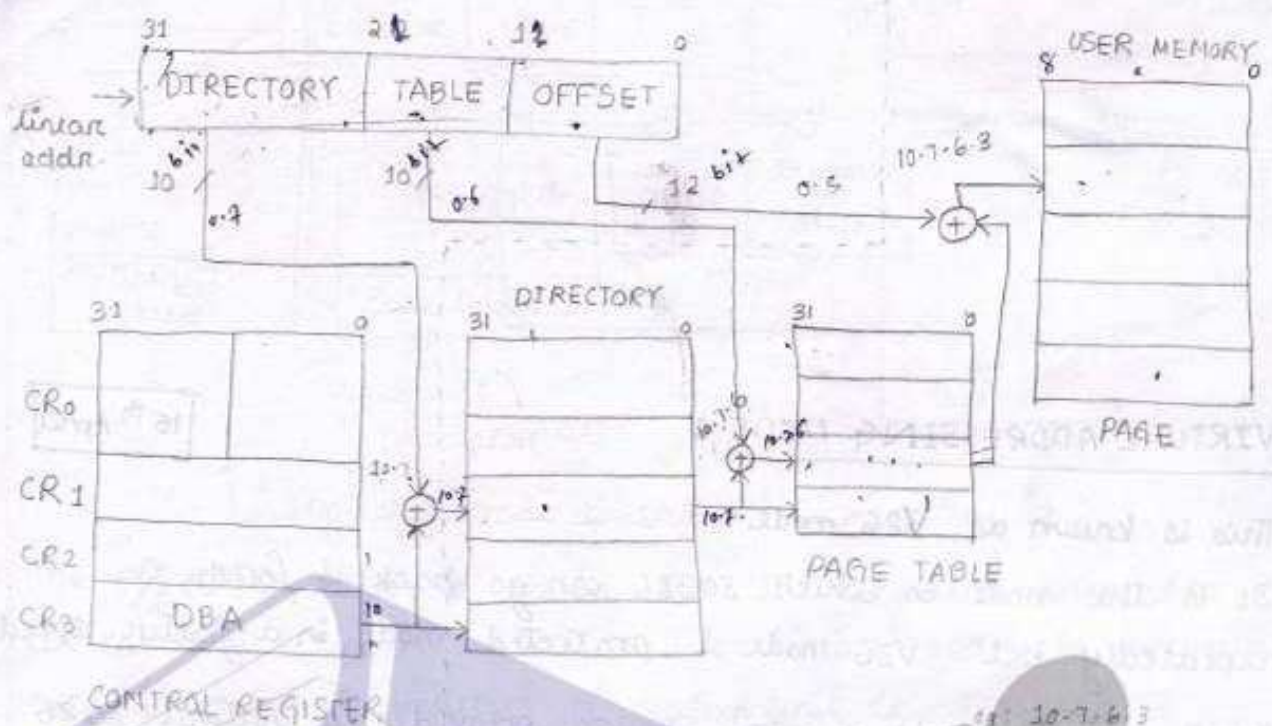
The addressing mechanism in V86 mode is same with that of the 80386 real mode.

In virtual mode 8086 can address 1MB of physical memory that may be anywhere in the 4GB space of address in protected mode of 80386.

PAGING :

Inside 80386

Inside Memory



- Paging is one of the memory management technique used for virtual memory multi-tasking operating system.
- This technique is useful to access a particular memory location quickly.
- The advantage of paging scheme is that the complete segment of the task need to be in the physical memory at any time & only few pages of the segment which are currently required will be available in the ~~ph~~ memory.
- Paging mechanism provides an effective technique to manage the physical memory for any multi-tasking system.
- A paging unit consists of a ^{page} page table, page directory, a page descriptor & base/control reg.

Paging Unit :

The paging unit converts the linear address which is provided by the segment unit into physical address. It converts the complete map of a task into pages. & The size of each pg. is 4KB.

- From the 32-bit of linear addr., 0 to 11 are used to select a particular pg. 12 to 21 (10-bits) are used to select a particular pg. table. {22-31 are used to select a particular pg. directory.

Page Directory:

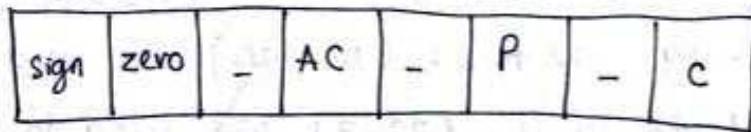
- 4KB (max^m) in size.
- Each directory entry is of 4 byte. So, The Total no. of entries allowed to a directory is 1024.

Page Table:

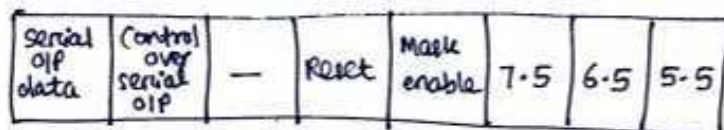
- Each P.T. is 4KB in size.
- contains max^m 1024 entries.
- The pg. table entries contain the starting addr. of the page & the statistical informⁿ about the page.

Lecture notes in

8085



(8085 flag Register)

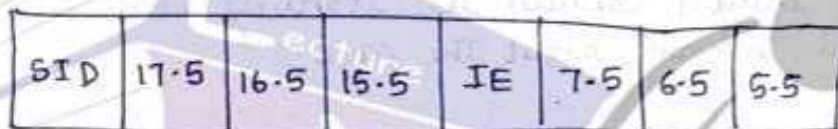


(SIM)

0 - enable
1 - disable

0 - all interrupts disabled
1 - " " enabled

0 - RST 7-5 enabled
1 - " " disabled



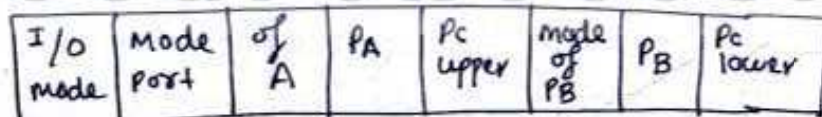
1 \Rightarrow corresponding
RST is
pending

1 = masked
0 = not masked

1 = all interrupts enabled
0 = " " disabled

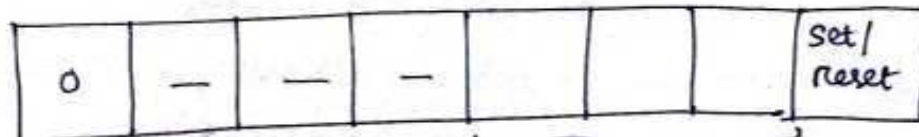
(RIM)

8255



(1)

(CWR for I/O mode)



(CWR for
BSR mode)

To select the
particular
bit of port C

Mode set Reg:

7	6	5	4	3	2	1	0
Autoloed	TC Stop	Extended write	Rotating Priority	DMA CH-3	DMA CH-2	DMA CH-1	DMA channel 0

7	6	5	4	3	2	1	0
0	0	0	update flag	TC Status CH 3	TC Status CH 2	TC Status CH 1	TC Status for CH 0

(Status Register)

Update flag = 1, when TC o/p is high for the respective channel.
 0, TC bits are cleared when, → status word is read
 → 8257 receives a reset.

Lecture notes in

8259

ICW1:

A ₀	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
⑤	A ₇	A ₆	A ₅	①	LTIM	ADI	SNG	ICW ₄

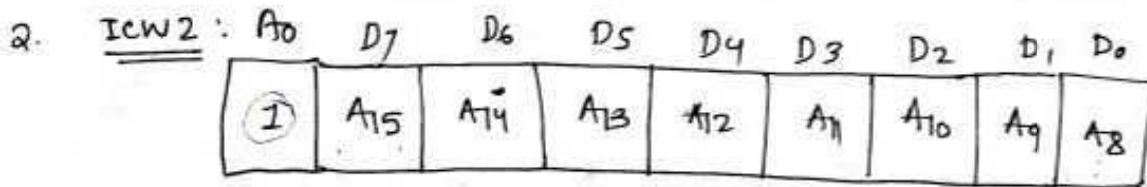
D₀ = 0, ICW₄ not reqd.
 1, ICW₄ "

D₁ = 0, 8259 is single
 1, " " cascaded

D₂ = 0, call add. interval is 8
 1, " " " " 4

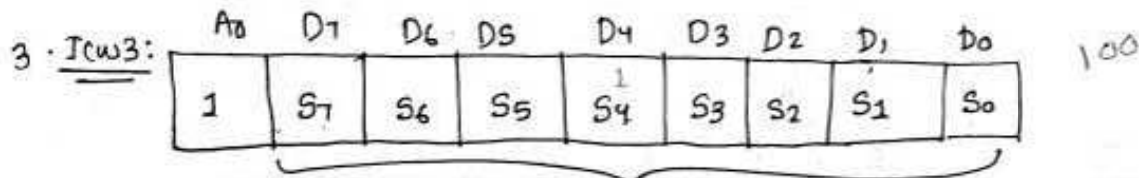
D₃ = 0, edge triggering
 1, level triggering

A₅ to A₇ give the
 vector address of
 interrupt.



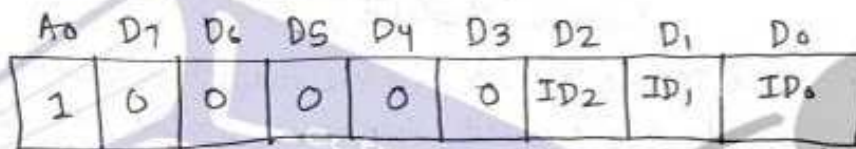
A₈ - A₁₅ → interrupt vector addr.

This is used to load the high-order byte of the interrupt vector address of all the interrupts.



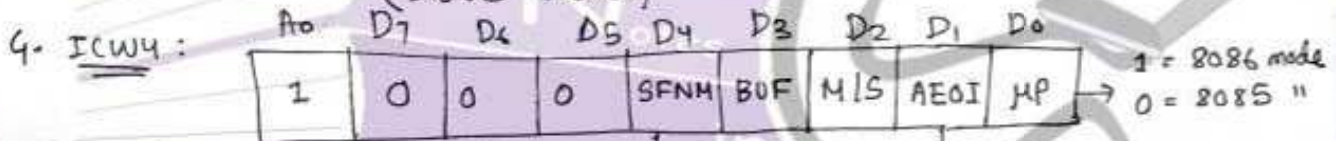
1 → IR i/p has a slave
0 → IR i/p doesn't have a slave.

(master device)



Slave ID

(Slave device)

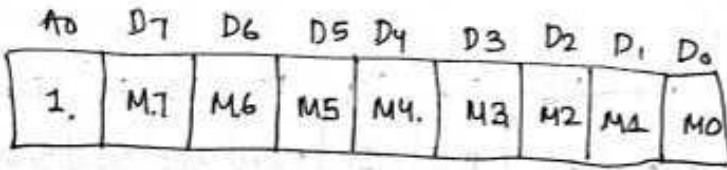


1 = Special fully nested mode
0 = Not SFNM

1 = auto EOI
0 = Normal EOI

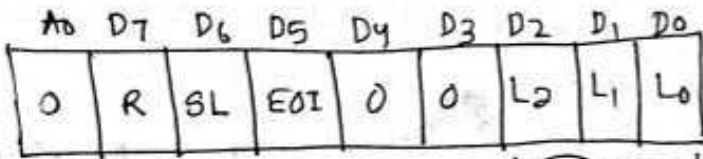
1 = 8086 mode
0 = 8085 "

OCW1:



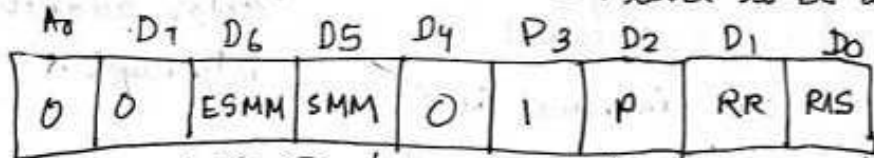
1 → to be masked
0 → to be unmasked

OCW2:



IR level to be acted upon:

OCW3:

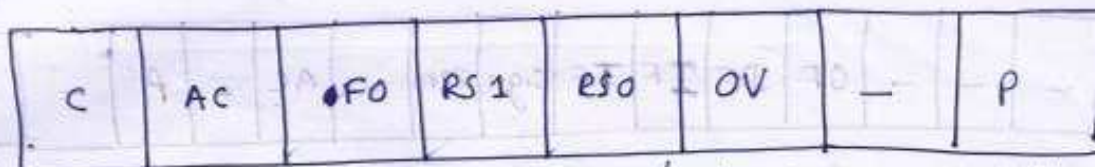


Special mask mode.

00, 01 → No action
10 → Reset special mask
11 → Set special mask.

1 = Poll command
0 = No poll command

Read reg. command
00, 01 → NoP
10, 11 → Read 15 reg.



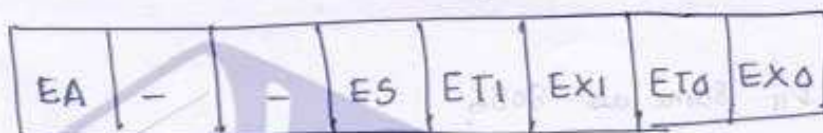
↓
general
purpose
flag

Bank
select

↓
1 if carry is generated
in D₆ not in D₇ or vice versa

odd - 1
even - 0

(PSW Register of 8051)



1 → enable

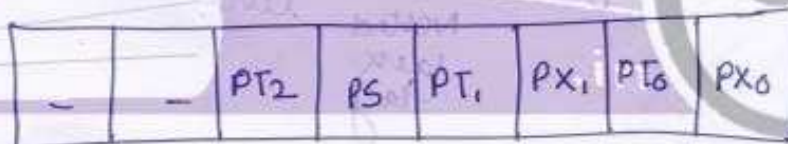
0 → disable for all

S → serial port interrupt

T₀, T₁ → Timer/counter

X₀, X₁ → External interrupt

(Interrupt enable (IE) reg. of 8051)



↓
for
8032/
8052
only

(Interrupt priority reg. of 8051)



TF → Trap flag (if TF = 1, processor enters into single step execution mode)

IF → Interrupt flag (1 = maskable interrupts are recognized by CPU else all are ignored)

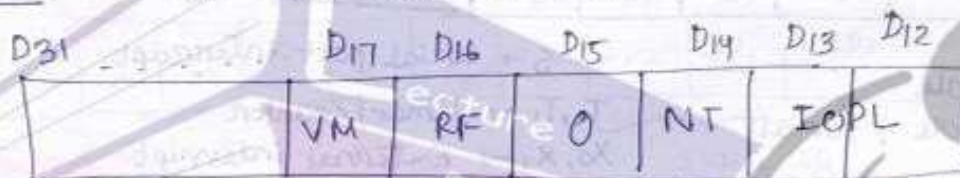
DF → Directional flag (0 = auto decrement mode, 1 = auto increment mode)

OF → Overflow flag (1 = carry generated at 15th bit)

(8086 flag reg.)

80386

D0 → D11 same as 8086



↓
Virtual mode

↓
Resume flag

↓
Nested task flag

↓
I/O Privilege level

VM : 1, 80386 executes 8086 mode
0, " " 80386 "

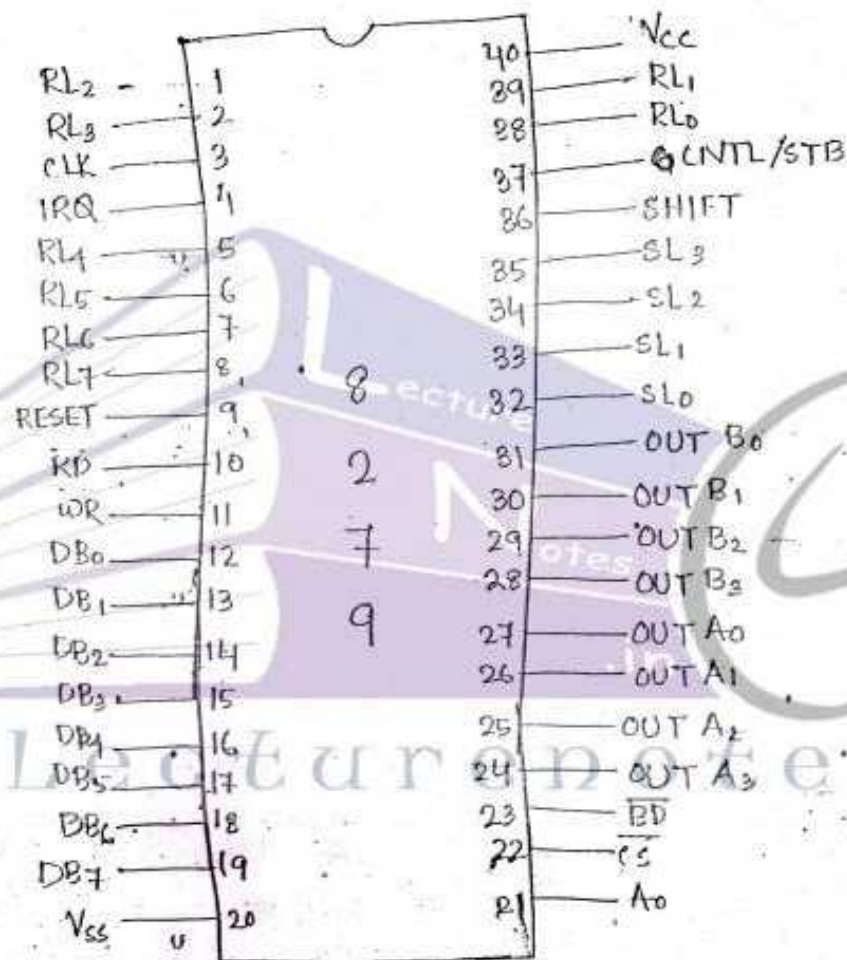
RF : 1, 80386 ignores all debugging faults
0, has to give service to debugging

NT : 1, Indirect Return
0, Normal Return

IOPL : Indicates the privilege level of current I/O operation.

→ 8279 Programmable Keyboard/Display Interface :- 3

- The 8279 is a hardware approach to interface a matrix keyboard and a multiplexed display with the CPU
- It is a 40-pin device with two major segments,
 - keyboard
 - and display.



DB0 - DB7 → Bidirectional databus

CLK → clock input

RESET → Reset input

\overline{CS} → chip select pin

\overline{RD} → Read operation

\overline{WR} → write operation

$\overline{A0}$ → Buffer Address

IRQ \rightarrow Interrupt request output

SL₀ - SL₃ \rightarrow Scan lines

RL₀ - RL₄ \rightarrow Return lines

SHIFT \rightarrow shift input

ENTL/STB \rightarrow control/strobe input

OUT A₀ - OUT A₃ \rightarrow Display (A) outputs

OUT B₀ - OUT B₃ \rightarrow Display (B) outputs

\overline{BD} \rightarrow Blank display output

\rightarrow The keyboard segment can be connected to a 64-contact key matrix. Keyboard entries are debounced and stored in the internal FIFO (First-in-First-out) memory, an interrupt signal is produced with each entry.

\rightarrow The display segment can give a 16-character scanned display interface with each device as LEDs.

\rightarrow This segment has 16 x 8 R/W memory (RAM) which can be used to read/write information for display purpose.

\rightarrow The display can be set up in either right entry or left entry format.

\rightarrow The ~~6279~~ internal block of 8279 consists of 4 major sections like, keyboard, scan, display and MPU interface.

Keyboard section :-

\rightarrow This section involves lines (RL₀ - RL₄) that can be connected to 8 columns of a keyboard plus two additional lines shift and ENTL/STB.

\rightarrow The status of the SHIFT key and the control key can be stored along with a key closure.

→ The keys are automatically debounced and the keyboard can operate in two modes like two key lockout or N-key rollover.

→ In the two key lockout mode, if two keys are pressed almost simultaneously, only the first key is identified.

→ In the N-key rollover mode, simultaneous keys are identified and their codes are stored in the internal buffer, it can also be set up so that no key is identified until only one key remains fixed.

Scan section

→ The scan section contains a scan counter and four scanlines ($SL_0 - SL_3$). These four scanlines can be decoded using a 4 to 16 decoder to produce 16 lines for scanning.

→ These lines can be connected to the rows of a matrix keyboard and the digit drivers of a multiplexed display.

Display section

→ The display section has 8 output lines divided into two groups $A_0 - A_3$ and $B_0 - B_3$. These lines can be used either as a group of eight lines or as two groups of four, in addition with the scan lines for a multiplexed display.

→ The display can be blanked by using the BD line.

→ This section involves 16×8 display RAM. The MPU can read from or write into any of these registers.

• Interface section : —

→ This section involves 8 bidirectional data lines ($DB_0 - DB_7$), one interrupt request line (IRQ) and six lines for interfacing, including the buffer address line (A_0).

→ When

$A_0 = \text{high}(1)$, signals are interpreted as control words
 $= \text{low}(0)$, signals are interpreted as data.

→ IRQ line goes high whenever data entries are kept in the register.

Programming 8279 —

→ 8279 is a complex device that can accept 8 different commands to execute various functions.

→ The initialization commands are

1. left or right entry and key rollover.

2. clock frequency prescaler.

3. starting address and incrementing mode of RAM.

4. RAM address to read and write data and

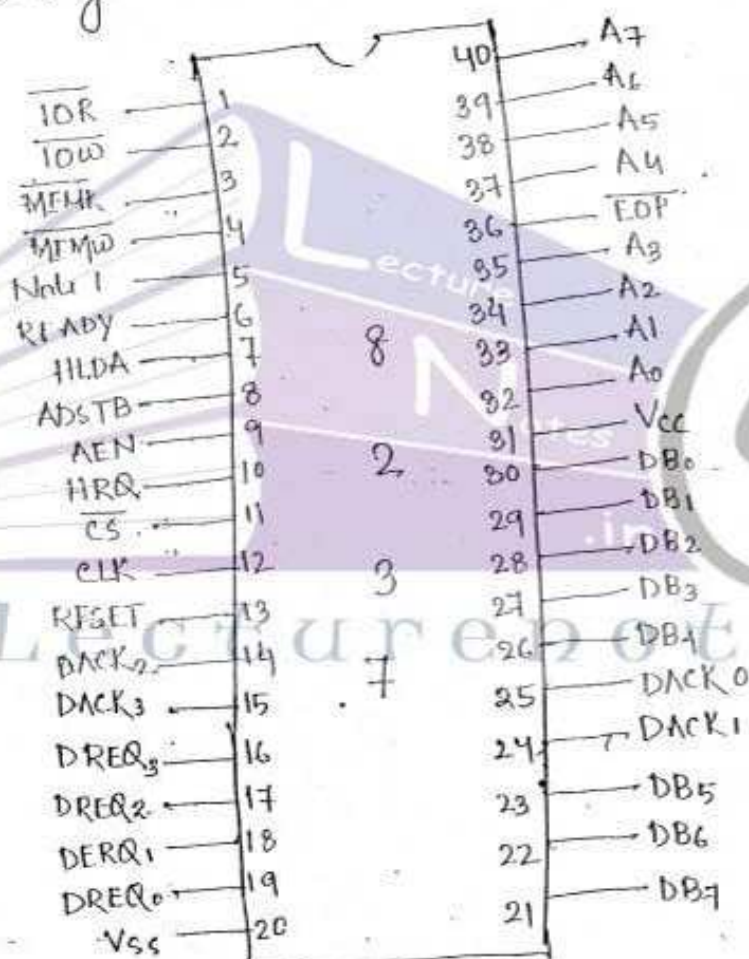
incrementing mode.

5. blanking format.

8237 - Programmable DMA Controller :-

→ It is a LSI controller IC that is used to utilize the DMA function in microprocessor systems :

→ DMA capability permits device such as peripherals to execute high speed data transfers between either two sections of memory or between memory and an I/O device .



→ In a microprocessor system, the 8237 behaves as a peripheral controller device and its operation must be initialized through software. This is performed by reading from or writing into the bits of its internal registers .

→ Whenever the 8237 is not in use by a peripheral device for DMA operation, it is in an idle state .

→ In this state, the microprocessor can issue commands to the DMA controller and read from or write to its internal registers.

→ Databus lines DB0 through DB7 form the path over which these data transfers occur. The register which is accessed is determined by a 4-bit register address that is applied to address inputs A0 to A3.

→ ~~8237~~ 8237 consists of 4 independent DMA channels i.e. channel 0 through channel 3. Each of these channels is dedicated to a specific peripheral device.

→ When a peripheral device wants to execute DMA operations, it makes a request for service at its DREQ input by switching it to logic 1. On receiving the request signal, the DMA controller passes acknowledge the request through DMA-acknowledge (DACK) signal.

→ During DMA cycles, the DMA controller, not the MPU drives the system bus. It produces the address and all control signals required to perform the memory or I/O data transfers.

→ At the beginning of all DMA bus cycles, a 16-bit address is output on lines A0 through A7 and DB0 through DB7. The upper 8-bits of the address available on the databus lines, ~~and~~ when the ADSTA signal is active.

→ The AEN (Address Enable) output signal is active during the complete DMA bus cycle and.

can be used to both enable the address latch and disable other devices connected to the bus.

→ The READY input is used to accommodate slow memory or I/O devices. READY must go active (logic 1) before the 8237 will complete a memory or I/O bus cycle. As long as READY is at logic 0, wait states are inserted to extend the duration of the current bus cycle.

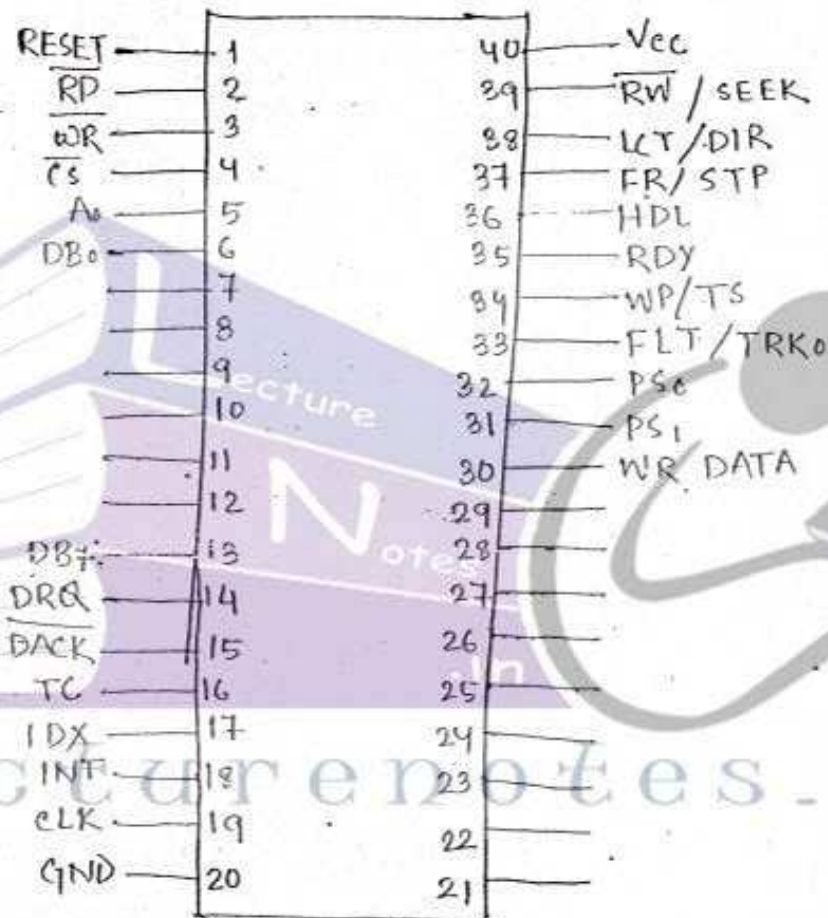
→ For I/O device to memory data transfer, \overline{IOR} and \overline{MEMW} signals are used, and from memory to an I/O device data transfer \overline{MEMR} and \overline{IOW} signals are used.

→ For memory-to-memory data transfer, \overline{MEMR} and \overline{MEMW} signals are used.

Lecture notes in
Lecture notes in

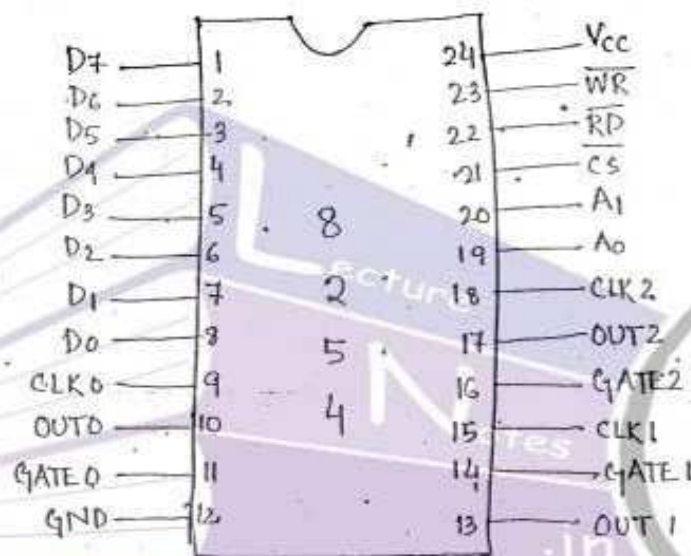
8272 - Floppy Disk controller :-

- 8272 is a 40 pin IC package that is used in the circuit and control functions required to interface a floppy disk drive with a processor.
- 8272 is able to interface four floppy disk drives simultaneously with a single host processor.



8254.- Programmable Interval timer :-

→ The 8254 Programmable interval timer/counter is functionally similar to the software designed counters and timer. It produces accurate time delays and can be used for applications such as a real-time clock, an event counter, a digital one-shot, a square-wave generator and a complex waveform generator.



D0 - D7 → 8-bit data bus

CLK0, CLK1, CLK2 → Counter clock inputs

GATE0, GATE1, GATE2 → Counter gate inputs

OUT0, OUT1, OUT2 → Counter outputs

\overline{RD} - Read control signal

\overline{WR} - Write command

\overline{CS} - chip select

A0, A1 - counter select

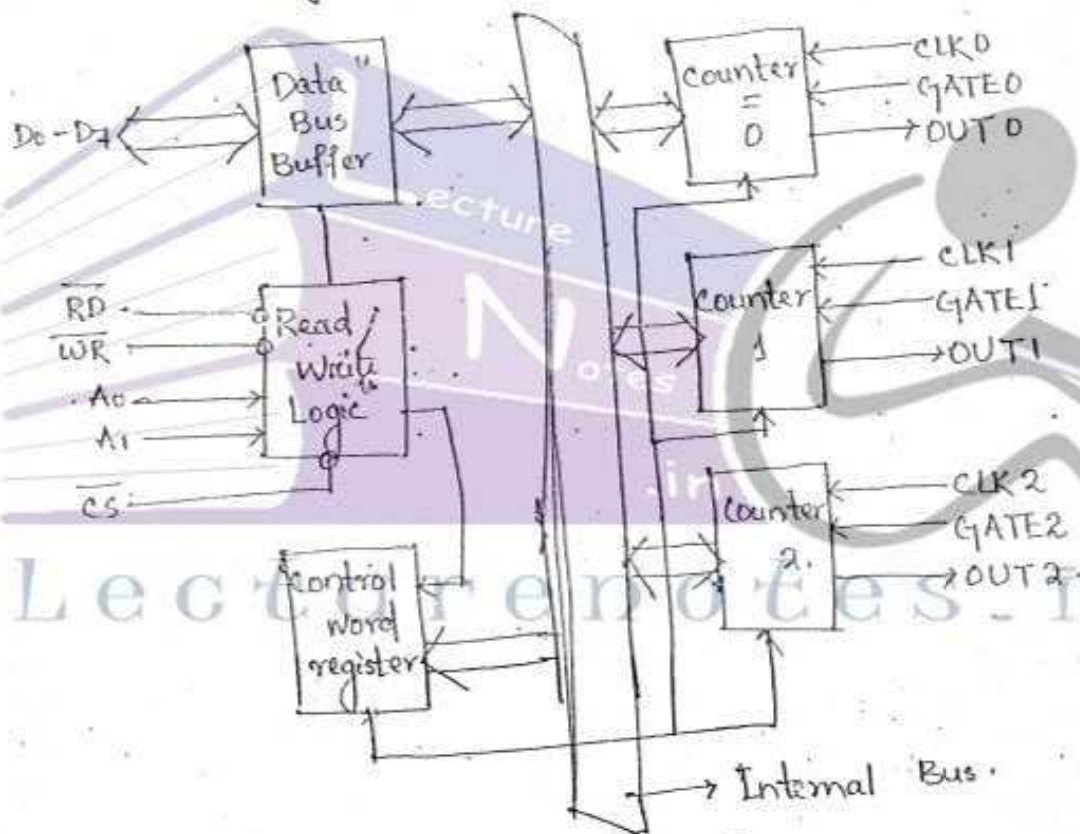
Vcc - +5V

GND - Ground connection.

→ The 8254 timer consists of 3 identical 16-bit down counters that can operate independently in any one of the 6 modes.

→ It is a 24-pin DIP and needs a 4.5V powersupply.

→ To operate a counter, a 16-bit count is placed in its register and on command begins to decrement the count until it reaches 0. At the end of the count it produces a pulse that can be used to interrupt the MPU. The counter can count either in binary or BCD.



(Block Diagram)

→ 8254 contains 3 counters - counter 0
counter 1
counter 2

and a data bus buffer, Read/Write control logic, and a control register.

→ Each counter has two input signals clock (CLK) and GATE and one signal OUT.

→ The Data Bus Buffer is a tri-state 8-bit bidirectional buffer that is connected to the data bus of the MPU.

→ Control Logic :-

→ The control section has 5 signals, \overline{RD} , \overline{WR} , \overline{CS} , A_0 & A_1 .

→ Address lines A_0 and A_1 of the MPU are usually joined to lines A_0 and A_1 of 8254.

A_1	A_0	Selection
0	0	→ Counter 0
0	1	→ Counter 1
1	0	→ Counter 2
1	1	→ Counter 3

Modes of operation :-

→ The 8254 can operate in six different modes.

1) Mode 0 : Interrupt on Terminal Count :-

→ In this mode the OUT signal is low.

→ Once a count is placed in the register the counter is decremented every cycle and when the count reaches zero, the OUT goes high.

→ It can be used as an interrupt.

2) Mode 1 : Hardware - Retriggerable One shot :-

→ In this mode the OUT is initially high.

→ When the Gate is triggered, OUT goes low and at the end of the count, the OUT goes high again, thus producing a one-shot pulse.

Mode 2 : Rate Generator :

- This mode is used to generate a pulse equal to the clock period at a given interval.
- When a count is placed, the OUT stays high until the count reaches 1, and then the OUT goes low for one clock period.
- The count is reloaded automatically, and the pulse is generated continuously.

Mode 3 : Square-wave generator :

- In this mode, when a count is placed, the OUT is high. The count is decremented by two at every clock cycle and when it reaches zero the OUT goes low and the count is reloaded again.
- This process is repeated continuously, thus producing a continuous squarewave with period equal to the period of the count is produced.

Mode -4 : Software Triggered strobe :

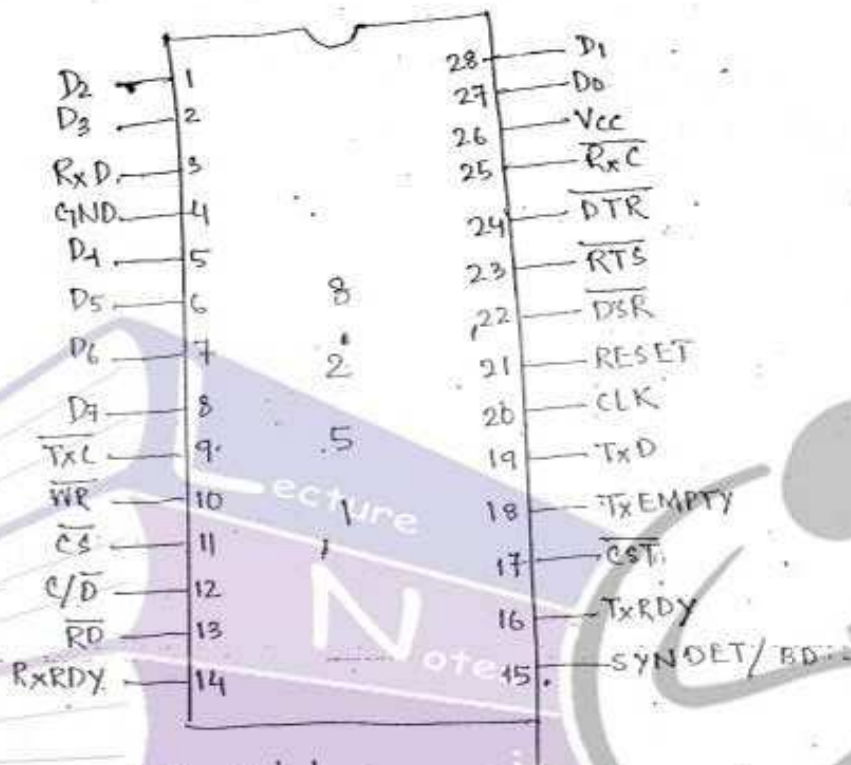
- In this mode, the OUT is initially high, it goes low for one clock period at the end of the count.
- The count must be reloaded for subsequent outputs.

Mode -5 : Hardware Triggered strobe :

- This is similar to Mode-4, except that it is triggered by the rising pulse at the gate.
- Initially, the ~~out is to~~ OUT is low, and when the gate pulse is triggered from low to high the count begins.
- At the end of the count, the OUT goes low for one clock period.

8251 - USART :

→ 8251 is a programmable chip designed for synchronous and asynchronous serial data communication, packaged in a 28-pin DIP.



D₀ - D₇ → 8-bit databus

C/D → control or data

RD → Read data command

WR → Write data command

CS → chip select signal

CLK → clock pulse

RESET → Reset signal

TxC → Transmitter clock

TxD → Transmitter data

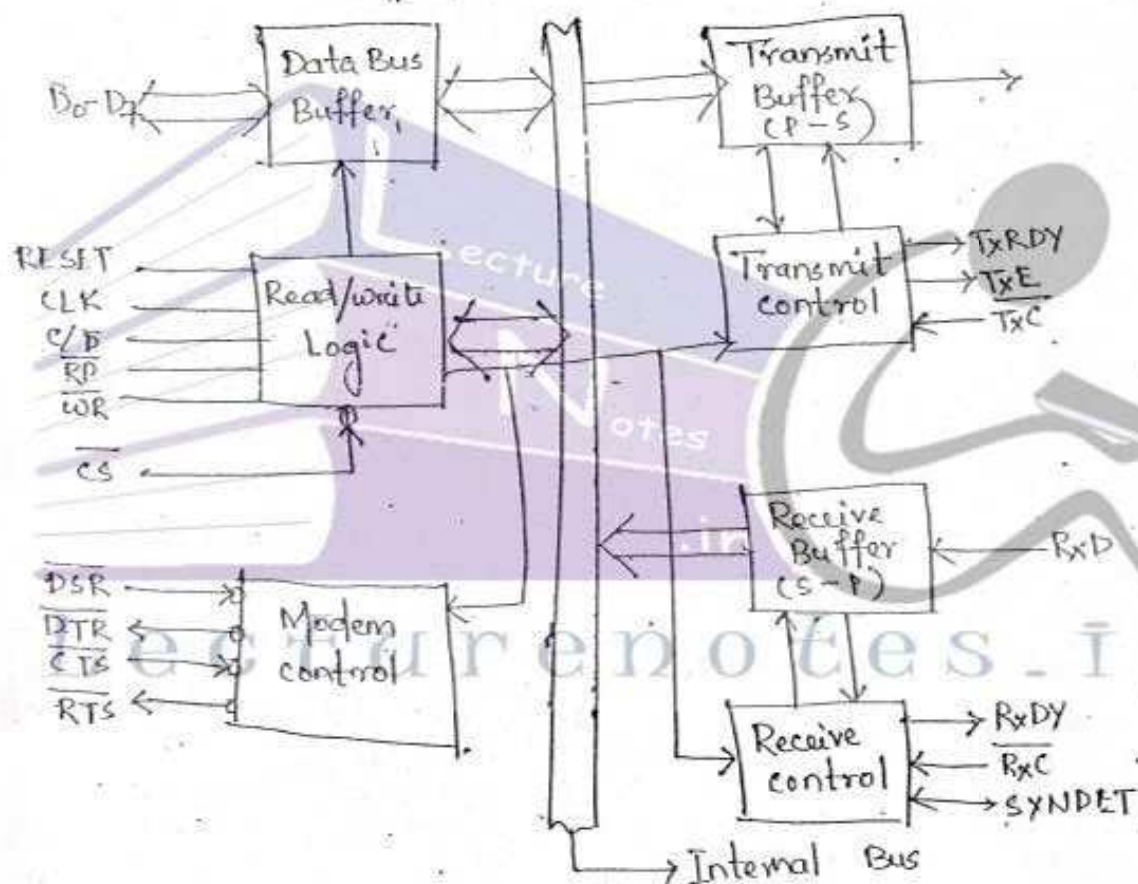
RxC → Receiver clock

RxD → Receiver data

RxRDY → Receiver Ready

TxRDY → Transmitter Ready

- $\overline{DSR} \rightarrow$ Data set Ready
- $\overline{DTR} \rightarrow$ Data terminal ready
- $\overline{SYNDET}/BD \rightarrow$ Sync Detect / Break Detect
- $\overline{RTS} \rightarrow$ Request to send data
- $\overline{CTS} \rightarrow$ Clear to send data
- $TxE \rightarrow$ Transmitter Empty
- $V_{cc} \rightarrow +5\text{V dc}$
- $GND \rightarrow$ Ground connection.



(Block Diagram)

→ The block diagram of 8251-USART consists of 5 main sections.

→ Read/write Control Logic

→ Transmitter

→ Receiver

→ Data Bus Buffer

→ Modem control

→ The control logic interfaces the chip with the microprocessor and determines the functions of the chip according to the control word in its register and monitors the dataflow.

→ The transmitter section converts a parallel word received from the microprocessor into serial bits and transmits them over the Tx/D line to a peripheral.

→ The receiver section receives serial bits from a peripheral; converts them into a parallel word and transfers the word to the microprocessor unit.

→ The modem control is used to develop data communication through modems over telephone lines.

Transmitter section:

→ The Transmitter accepts parallel data from the microprocessor and converts ~~from~~ them into serial data.

→ It contains two registers.

① Buffer register to hold 8-bits and

② Output register to convert 8-bits into a stream of serial bits.

→ The μp writes a byte in the buffer register, whenever the output register is empty, the contents of the buffer register are transferred to the output register.

- Three output signals and one input signal are associated with the transmitter section.

TxD → Transmit Data: Serial bits are transmitted on this line.

TxC → Transmitter clock: This input signal controls the rate at which bits are transmitted by the USART.

TxRDY → Transmitter Ready: When this signal is high, it means that the buffer register is empty and the USART is ready to accept a byte.

TxE → When this signal is high, it specifies that the output register is empty. This signal is reset when a byte is transferred from the buffer to the output registers.

Receiver section:

- The receiver accepts serial data on the RxD line from a peripheral and converts them into parallel data.

→ It has two registers: → input register
buffer register.

- When the RxD line goes low, the control logic assumes it is a start bit, waits for half a bit time and samples the line again.

- The associated signals with this unit are two input signals and one output signal.

They are,

RxD → Receive Data: Bits are received serially on this line and converted into a parallel byte in the receiver input register.