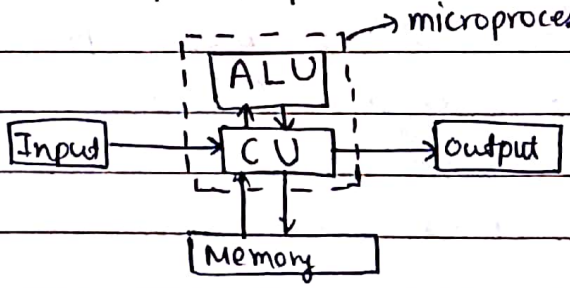
Unit-I

Introduction to Microprocessor

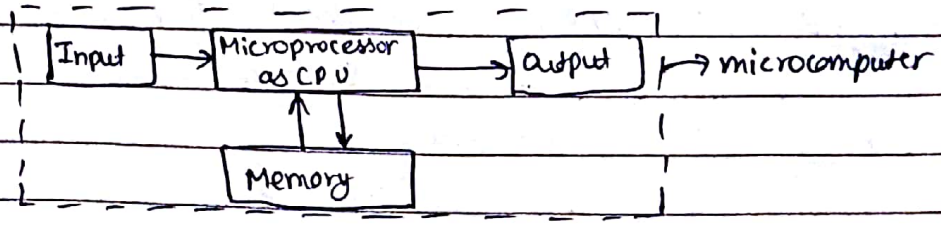
* Microprocessor is a multipurpose programmable logic device that can read binary instructions from a storage device called memory., accepts data as input and processes data according to those instructions and provides result as output.

* Difference between Microprocessor, Microcontroller & Microcomputer

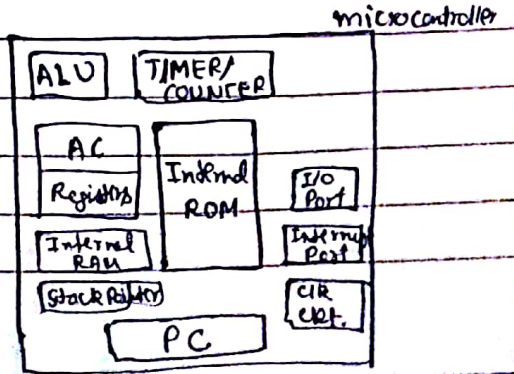
(i) microprocessor
- Programmable device generally called CPU
- Computing & Logical tasks
- Needs ALU, register array with CPU
- It is a multipurpose, programmable, clock drive, register based electronic device that reads binary instruction from memory, accepts binary data as input & process them.



(ii) microcomputer
- microprocessor ~~alongwith~~ along with input device, output device & memory is called a microcomputer.



(iii) Microcontroller
- a device ~~that~~ includes microprocessor, memory & I/O ~~&~~ signal lines on ~~a~~ ~~same~~ single chip, fabricated using VLSI technology



* Evolution of microprocessors

1948 - Transistor
↓
1950 — Fabricated on IC
1960 - Logic gates on IC (SSI techology)
↳ Small scale integration

MSI (Medium Scale Integration) - 100 gates
LSI (Large " ") 1000 gates
VLSI (very " " ") 10000 gates
ULSI (Ultra " " ") 100000 gates

1970 - Major Boon in electronic industry
(On Chip microprocessors)

* → Microprocessor Categories (Based on Size of Data Bus)

ⓐ 4 bit : 1971 — Intel 4004 based on pmos
Intel 4040

ⓑ 8 bit : 1973 - Intel 8008 based on pmos
1973 - Intel 8080 (based on nmos)
1975 - Intel 8085 (+5v)

ⓒ 16 bit : 1978 - Intel 8086
Intel 80186
Intel 8088
Intel 80188
Intel 80286

ⓒ 32 bit : 1985 - Intel 80386 (Computer Graphics & Data processing applications)
Intel 80486

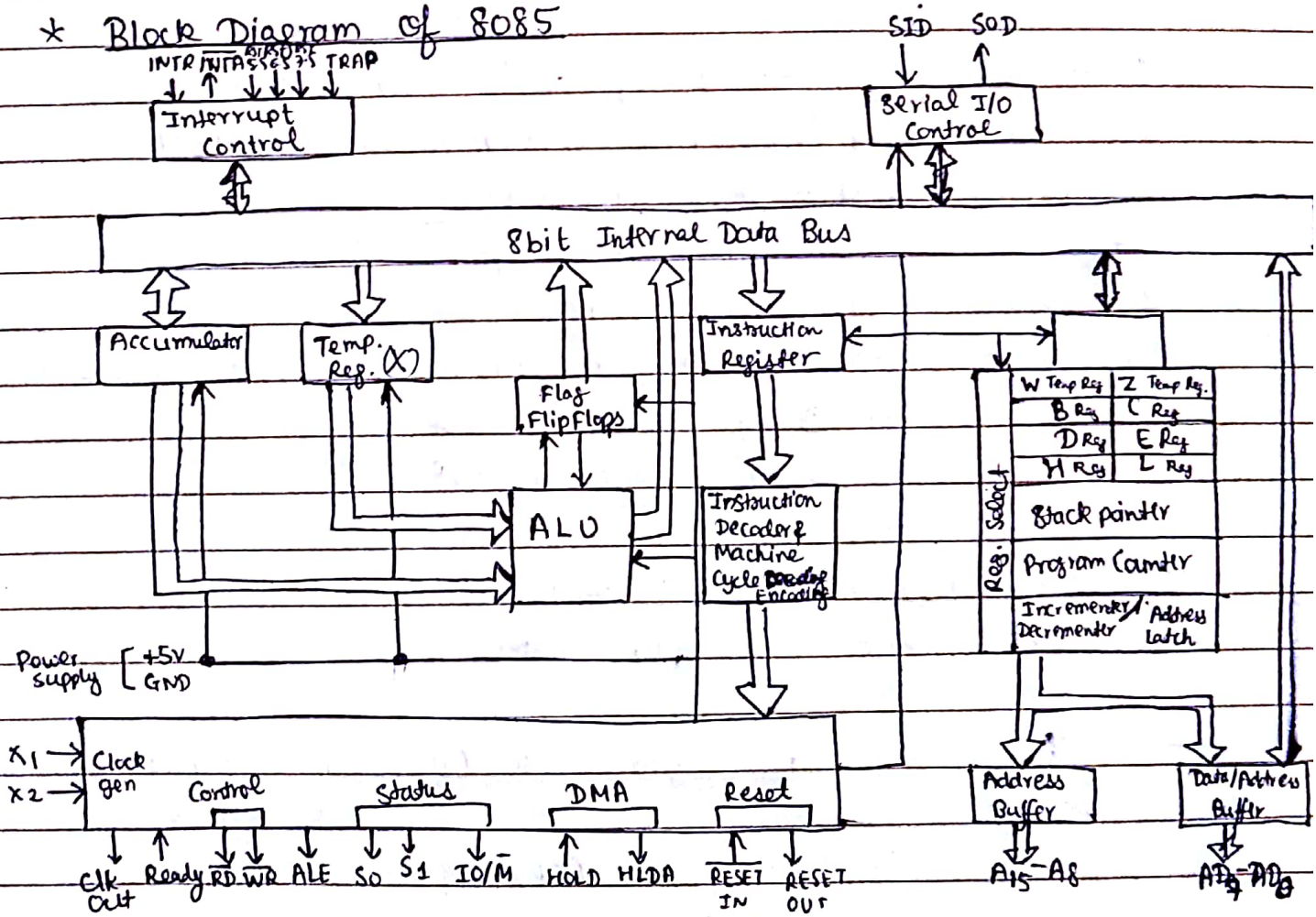
ⓓ 64 bit : 1993 - Intel Pentium
Intel Itanium
SUN'S SPARC
ULTRA SPARC

8085 microprocessor

* Buses are group of lines used for interconnecting various building blocks within a computing system.
* 8085 has 16 address lines. These are unidirectional and carry 16 bit address to identify address in memory location.
* 8085 has 8 data bus line. These are bidirectional and used for data flow.
* In 8085, $AD_0 - AD_7$ are multiplexed buses (address and data bus). These are also known as time shared or time multiplexed address data bus.

Max memory = 2^{16} = 64KB memory

* Block Diagram of 8085



* - General Purpose Registers - B, C, D, E, H, L

B C pair, D E pair, H L pair

Each register → 8 bit of data (1 byte)

Each register pair → 16 bit of data (2 byte)

- Accumulator → 8 bits register which contains 8 flip flops.

- Temporary registers - W, Z (pair, 16 bit) → not used by programmer → 8 bit each

- ALU → Perform binary operation & result of ALU stored in accumulator

- Flag Register or PSW (Program Status Word) - 8 bit register

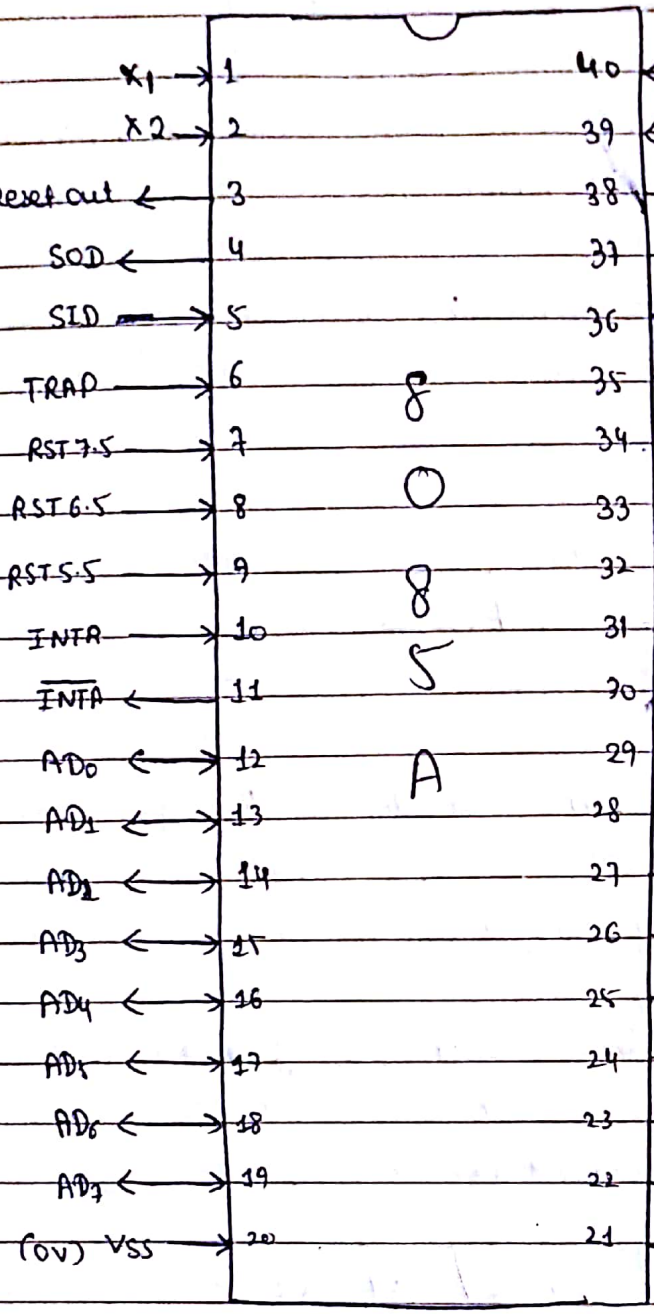
S	Z	X	AC	X	P	X	CF
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

CF → Carry flag → 1 if carry (MSB) for 8 or 16 bit data
AC → Auxilary carry (Half Carry) → Indicate status of 4 LSBs
P → Parity Flag → P=0 if odd parity → No. of 1's
P=1 if even parity
Z → zero Flag → 1 if ALU result are zero, i.e., 00H
0 if ALU result is non-zero, i.e., 01H to FFH
S → Sign Flag → 0 if result is +ve
1 if result is -ve

- Instruction Register (IR) & Instruction Decoder (ID)
for fetching & decoding instructions

- 16 bit registers: (i) Program Counter (PC) → pointer to next instruction to be executed
(ii) Stack pointer (SP) → pointer to stack top.
(iii) Interrupt Control → Enabling & Disabling Interrupts
TRAP, RST 5.5, RST 6.5, RST 7.5, INTR & $\overline{INTA}$
(iv) Serial I/O Control → For serial data transfer

* Pin diagram of 8085 microprocessor:

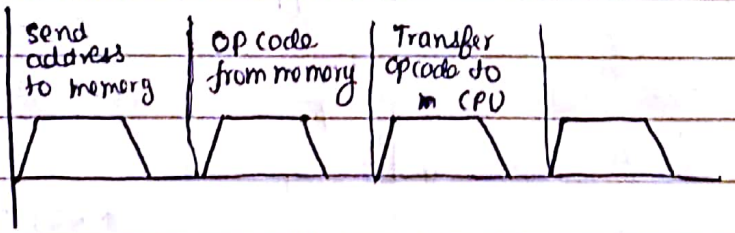


- X_1 & X_2 ⇒ Input pins from frequency oscillator
- SID, SOD ⇒ Serial input and output of data (using SIM)
- Reset Out ⇒ Reset external devices
- $\overline{\text{Reset In}}$ ⇒ Reset Microprocessor
- TRAP ⇒ Highest priority (can't stop) non maskable interrupt (NMI)
  location: 0024H
  Also called RST4.5
- RST 7.5 } 003CH
  RST 6.5 } RST → Restart
  RST 5.5 } 0034H, 002CH
- INTR ⇒ Interrupt request signal
  ⇒ lowest priority
- $\overline{\text{INTA}}$ ⇒ Interrupt acknowledge
- AD_0-AD_7 ⇒ Multiplexed Address Data Bus
- Vss ⇒ Ground
- Vcc ⇒ +5V
- A_8-A_{15} ⇒ Address Bus
- ALE ⇒ (Address Latch Enable) Hold address from multiplexed lines
- Ready ⇒ Check I/O is ready to send or receive data
  1 = ready } for synchronization
  0 = wait
- CLK ⇒ Clock output
- HLD ⇒ Hold operation send DMA signal
- HLDA ⇒ Hold acknowledgement
- So, S1 ⇒ Status Signals
- IO/$\overline{M}$ ⇒ 1 = work with I/O device
  0 = work with memory
- $\overline{WR}$ ⇒ Write Control Signal
- $\overline{RD}$ ⇒ Read Control Signal

IO/$\overline{M}$	S_1	S_0	Operation
0	1	1	Opcode fetch
0	1	0	Memory read
0	0	1	Memory write
1	1	0	I/O read
1	0	1	I/O write
1	1	1	Interrupt Acknowledge
Z	0	0	HALT
Z	X	X	HOLD
Z	X	X	RESET

* Instruction Cycle

- The time taken for the execution of an instruction is called instruction cycle. It consists of a Fetch Cycle and Execute Cycle.
- Machine Cycle contains a no. of clock cycles (T-states), i.e., no of clock cycles to perform fetch, MEMR, MEMW, IOR, IOW.
- Fetch cycle (fetch cycle)
  - → Microprocessor fetch op code from memory if it is slow memory then wait for memory to get opcode (wait cycle)
  - → Consists of 4 T states (always)



- Execute cycle
  - → no. of clock cycle to perform execution of the opcode task specified in instruction opcode.
  - → If operand in general purpose registers then immediate so no of clock cycles used
  - → If operand in memory, then clock cycles for MEMR, MEMW or IOR, IOW. For one operation like IOR, 3T states required.
- Instruction cycle = Fetch cycle + Execute cycle

* 8085 Addressing Modes

- There are five addressing modes:

(i) Immediate Addressing Mode:

→ Data is given with instr^n

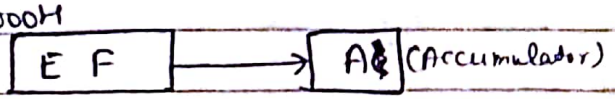
Ex:- 3E0000H MVI A, 35H

35 → A

(ii) Direct Addressing Mode

→ Memory location of data given with instruction

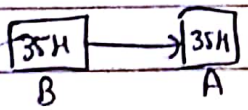
Ex:- LDA 3000H say data at 3000H is EFH



(iii) Register Direct Addressing Mode

→ Registers having data given with instruction

Ex:- MOV A, B



(iv) Register Indirect Addressing Mode

→ Register pair contains address of memory location where data is present. This pair is given with instⁿ

Ex:- LDAX B
→ B-C pair

(v) Implicit Addressing Mode

→ If address of data & destination of result is fixed, then there is no need to give operand with instⁿ, such instⁿ are called implicit addressing mode instⁿ.

Ex:- CMA, STC etc.

* Instruction

- The computer performs given task on specified data in response to a certain instructions.
- op code + operand = instruction
  - ↳ part of instⁿ which specifies task to be performed by microprocessor.
- Instruction set is collection of instructions.
  - → Based on operation
  - → Based on addressing mode (done)
  - → Based on length

- Based on length:

(i) One byte ~~instr~~ / word Instruction Set

→ Instr^n in which no. is not given as operand in instr^n

eg:- MOV A,B etc.

(ii) Two byte / word Instruction Set

→ Instr^n in which first byte will be instruction op code & second byte will be 8 bit nos.

eg:- MVI A, 36H

(iii) Three byte / word Instruction Set

→ Instr^n in which first byte will be 8 bit instruction op code, second byte will be 8 bit LSB of 16 bit no & third byte will be MSB of 16 bit no.

eg:- LXI H, 1536H

(21H, 36H, 15H)

MSB H — L LSB

- Based on operation:

(i) Data Transfer Instr^n

R → Register
M → Memory

				T-states
Move Instructions	MOV Rd, Rs	:	Rs → Rd	4
	MOV Rd, M	:	Rd ← M(HL)	
	MVI Rd, data 8	:	Rd ← data 8 (Immediately move)	7
	MOV M, Rs	:	M(HL) ← Rs	7
	MVI M, data 8	:	M(HL) ← data 8	10
Load & Store Instructions	LDA addr 16	:	(A) ← M(addr 16)	10
	LHLD addr 16	:	(Rp) ← M(addr 16) (load H-L pair) (h) ← M(addr 16+1)	16
	LXI Rp, data 16	:	(Rp) ← data 16	10
	LDAX Rp	:	(A) ← M(Rp) (Register Pair)	7
	STA addr 16	:	M(addr 16) ← A	13
	SHLD addr 16	:	M(addr 16) ← (L) M(addr 16+1) ← (H)	16
	STAX Rp	:	M(Rp) ← (A)	7
Exchange HL pair with DE pair	XCHG	:	[HL] ⇄ [DE]	

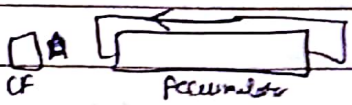
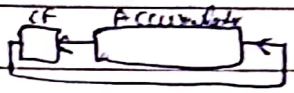
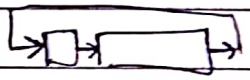
M → Memory
R → Register
Rp → Register pair

(ii) Arithmetic Instⁿ

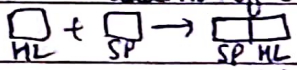
Operation	Instruction		Interpretation	T-states
8 bit add	ADD	R	: A ← A + R	4
	ADI	data8	: A ← A + data8	7
	ADD	M	: A ← A + M(HL)	7
8 bit add with carry	ADC	R	: A ← A + R + CY	4
	ACI	data8	: A ← A + data8 + CY	7
	ADC	M	: A ← A + M(HL) + CY	7
8 bit sub	SUB	R	: A ← A − R	4
	SUI	data8	: A ← A − data8	7
	SUB	M	: A ← A − M(HL)	7
8 bit sub with borrow	SBB	R	: A ← A − R − CY	4
	SBI	data8	: A ← A − data8 − CY	7
	SBB	M	: A ← A − M(HL) − CY	7
16 bit addition	DAD	Rp	: (HL) ← (HL) + Rp	10
Decimal adjust accumulator	DAA		: Convert 8 bit no. in Accumulator into BCD	4
8 bit increment	INR	R	: R ← R+1	4
	INR	M	: M(HL) ← M(HL) + 1	10
16 bit increment	INX	Rp	: (Rp) ← (Rp) + 1	6
8 bit decrement	DCR	R	: R ← R−1	4
	DCR	M	: M(HL) ← M(HL) − 1	10
16 bit decrement	DCX	Rp	: Rp ← Rp − 1	6

(iii) Logical Instⁿ

Operation	Instⁿ		Interpretation	T-states
Boolean AND	ANA	R	: A ← A ∧ R	4
	ANI	data8	: A ← A ∧ data8	7
	ANA	M	: A ← A ∧ M(HL)	7
Boolean OR	ORA	R	: A ← A ∨ R	4
	ORI	data8	: A ← A ∨ data8	7
	ORA	M	: A ← A ∨ M(HL)	7

Operation	Instr^n	Interpretation	T state
Boolean EXOR	XRA R	: A ← A ⊕ R	7
	XRI data8	: A ← A ⊕ data8	
	XRA M	: A ← A ⊕ M(HL)	
Compare	CMP R	: Compare content of A with R	4
	CPI data8	: " " " A with data8	7
	CMP M	: " " " A with M(HL)	7
Complement accumulator	CMA	: A ← A'	4
Complement carry flag	CMC	: CF ← CF'	4
Set carry flag	STC	: CF ← 1	
Rotate Accumulator left without carry	RLC	:  (circular rotate)	4
Rotate Accumulator through carry	RAL	: 	4
Rotate right without carry	RRC	: 	4
Rotate right through carry	RAR	: 	4

(iv) Stack Related Instructions

T-states			
	LXI	SP, 16 bit data	: Load Stack pointer with immediate data
	INX	SP	: Increment Stack pointer
	DCX	SP	: Decrement " "
	DAD	SP	: Double addition of Stack pointer data 
6	SPHL		: Transfer HL pair data into Stack pointer
12	PUSH	Rp	: Push register pair into stack memory
10	POP	Rp	: Pop data into Register pair
16	XTHL		: Exchange Top of stack with HL pair

(v) Branching Instructions

			T-states
	PCHL	: Transfer HL pair data into PC (Program counter)	6
	JMP	addr16 bit : Unconditional jump	10
Conditional jump	JC	addr16 bit : Jump if carry	

Conditional Jump

- JNC addr : Jump if no carry
- JZ addr : Jump if zero
- JNZ addr : Jump if no zero
- JPE : Jump if even parity, P = 1
- JPO addr : Jump if odd parity, P = 0
- JM addr : Jump if minus sign, S = 1
- JP addr : Jump if positive sign S = 0

(vi) Subroutine Instⁿ — T states

- CALL $addr_{16}$: Unconditional call — 18

Conditional Call

- CC $addr_{16}$: Call if carry
- CNC $addr_{16}$: Call if no carry
- CZ $addr_{16}$: Call if zero
- CNZ $addr_{16}$: Call if no zero
- CPE $addr_{16}$: Call if even parity
- CPO $addr_{16}$: Call if odd parity
- CM $addr_{16}$: Call if minus sign
- CP $addr_{16}$: Call if positive sign

- RET : Unconditional Return — 10

Conditional Return

- RC : Return if carry
- RNC : Return if no carry
- RPE : Return if even parity
- RPO : Return if odd parity
- RZ : Return if zero
- RNZ : Return if no zero
- RM : Return if minus sign
- RP : Return if positive sign

(vii) Machine Control Instⁿ — T states

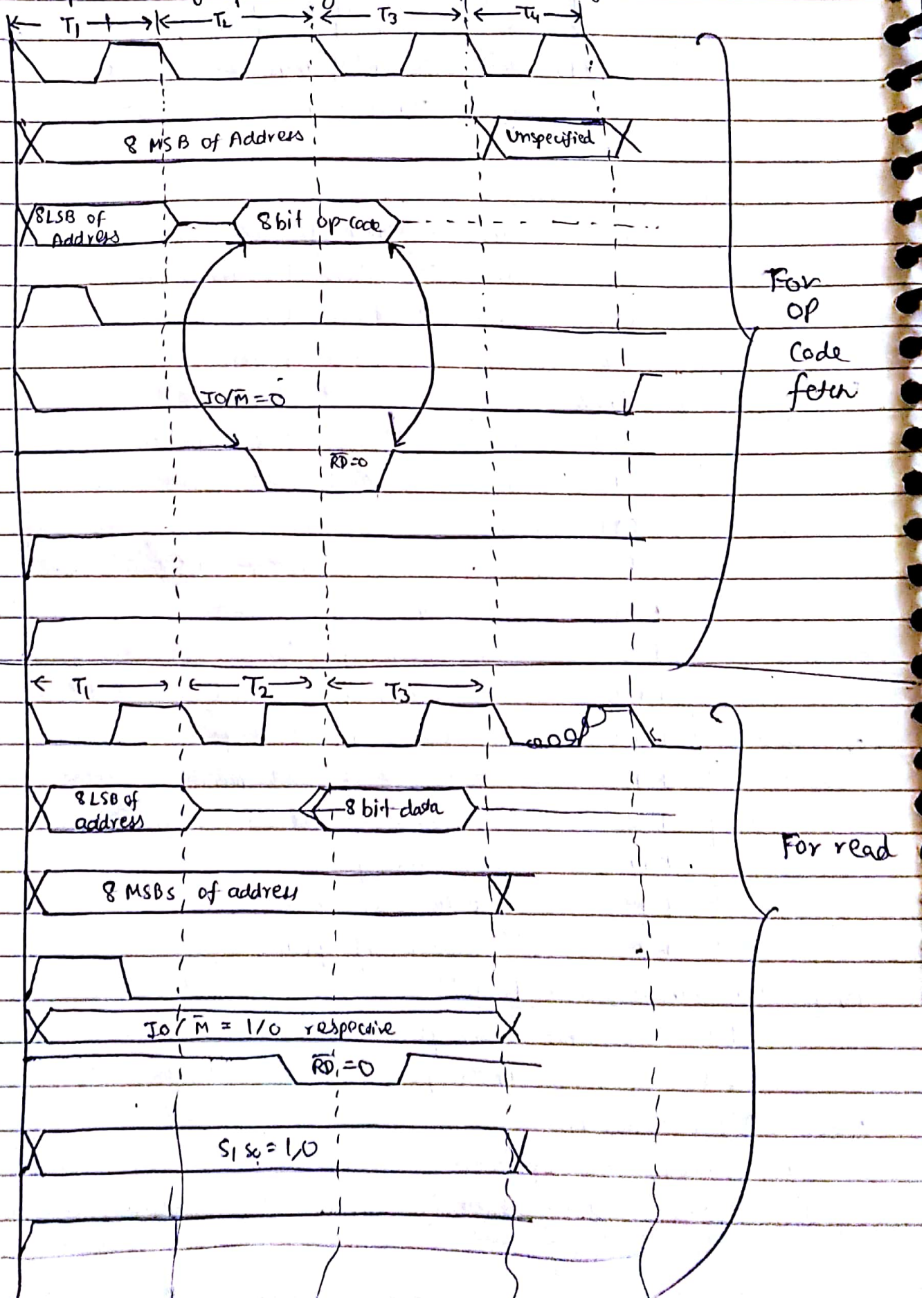
- DI : Disable Interrupt — 4
- EI : Enable Interrupt — 4

(viii) Other Instⁿ — T states

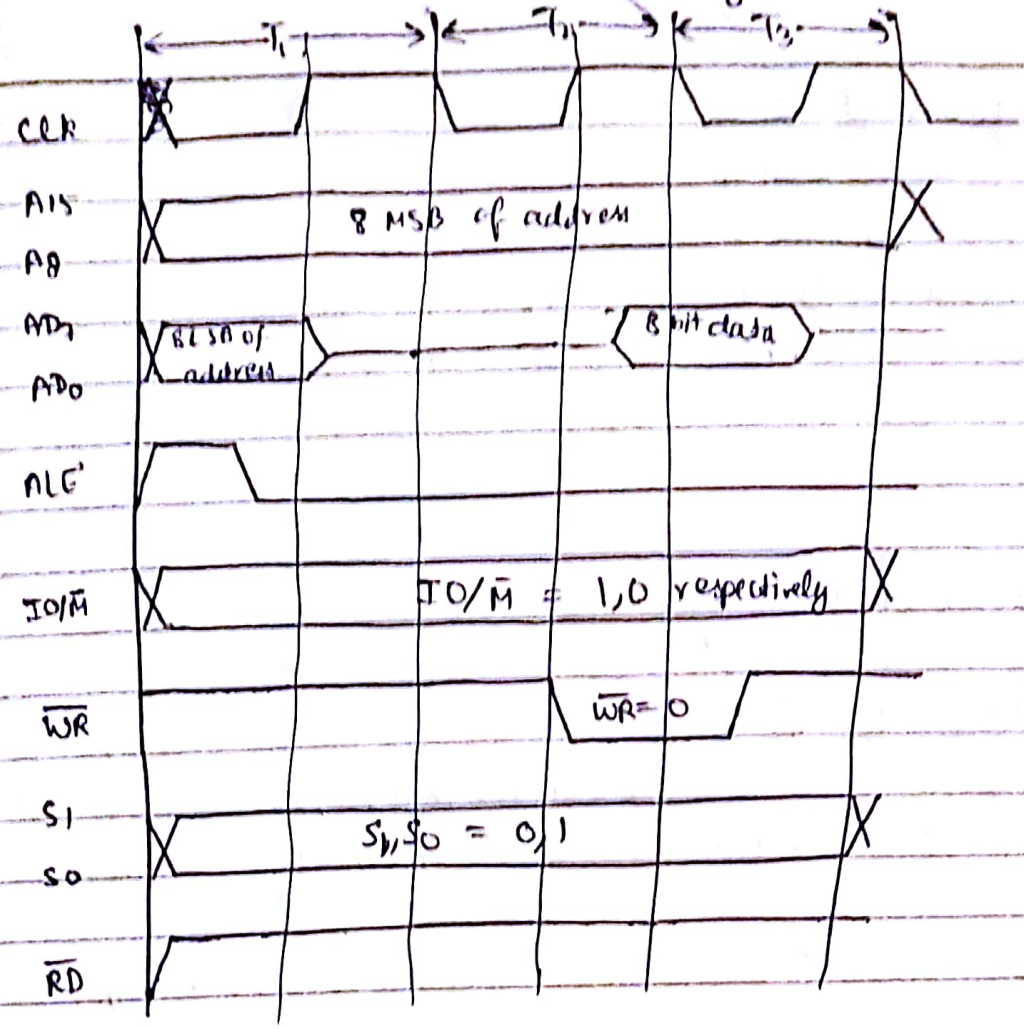
- NOP : No operation — 4
- IN : Input data to accumulator from a 8 bit addr port — 10
- OUT : output data from accumulator to a 8 bit addr port — 10

Timing Diagram

* To represent graphically the machine cycle



For memory / I/O write



* 8085 Interrupts

- Two types of interrupt
  - (i) Software Interrupt : CALL & RSTn (n = 0 to 7)
  - (ii) Hardware Interrupt : TRAP, RST 7.5, RST 6.5, RST 5.5, INTR

- Software Interrupt
  - (i) RSTn → Restart n

		Address
RST 0	-	0000 H
RST 1	-	0008 H
RST 2	-	0010 H
RST 3	-	0018 H
RST 4	-	0020 H
RST 5	-	0028 H
RST 6	-	0030 H
RST 7	-	0038 H

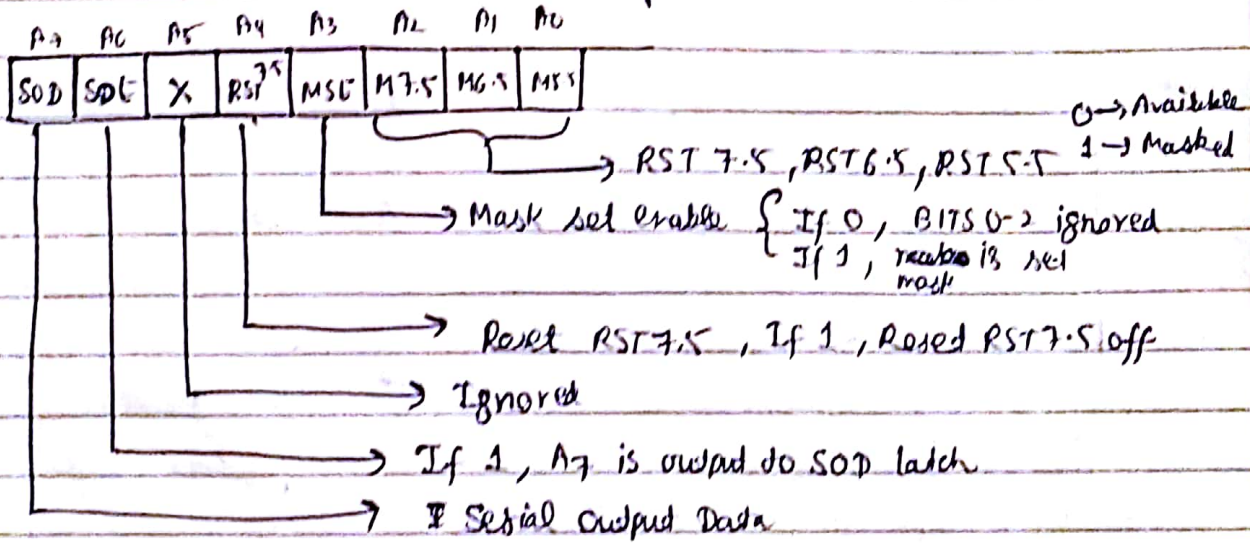
  - (ii) Call (Already Done).

- Hardware Interrupt

Highest ↑	TRAP	– 0024H	+ve edge & level triggered
Priority	RST 7.5	– 0034H	+ve edge triggered
	RST 6.5	– 003CH	level triggered
Lowest	RST 5.5	– 002CH	level triggered
	INTR		

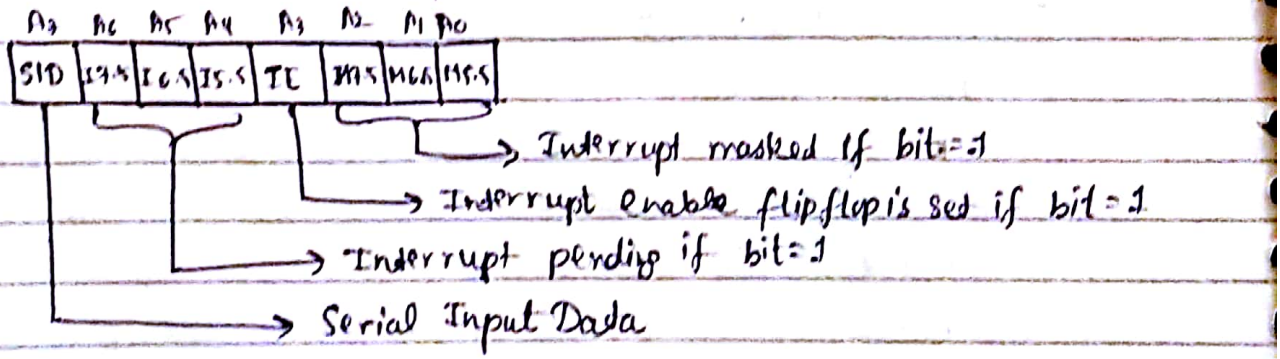
* SIM – SIM Instruction

To block or mask interrupt



- RIM Instruction

To read interrupt mask



* 8085 memory Interfacing

Ex:- Connect 2K byte ROM with 8085 from starting address 8000H

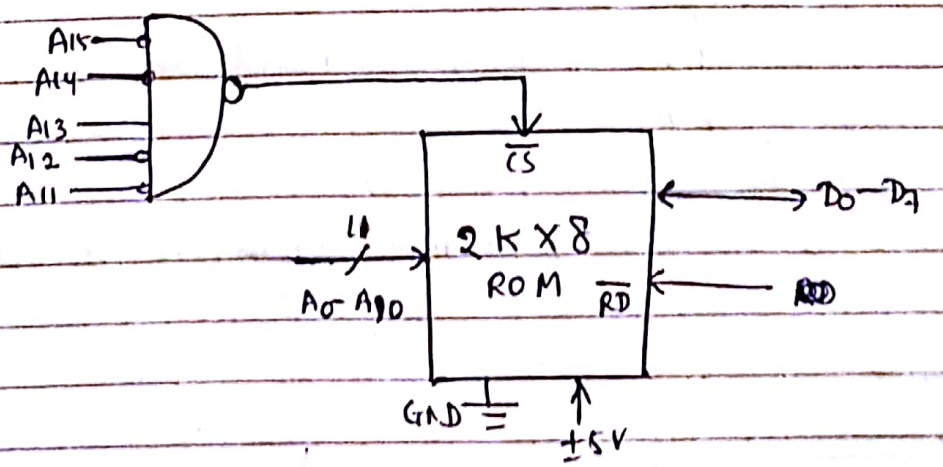
2K = 2^{11} → so 11 address line (A_0 – A_{10})

Complement this side

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6					0			0			0				
0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1
6					7			F			F				

fill with starting address

~~SS~~ = 67FFH so memory from 6000H to 67FFH



NOTE: If starting address not given, use 0000H

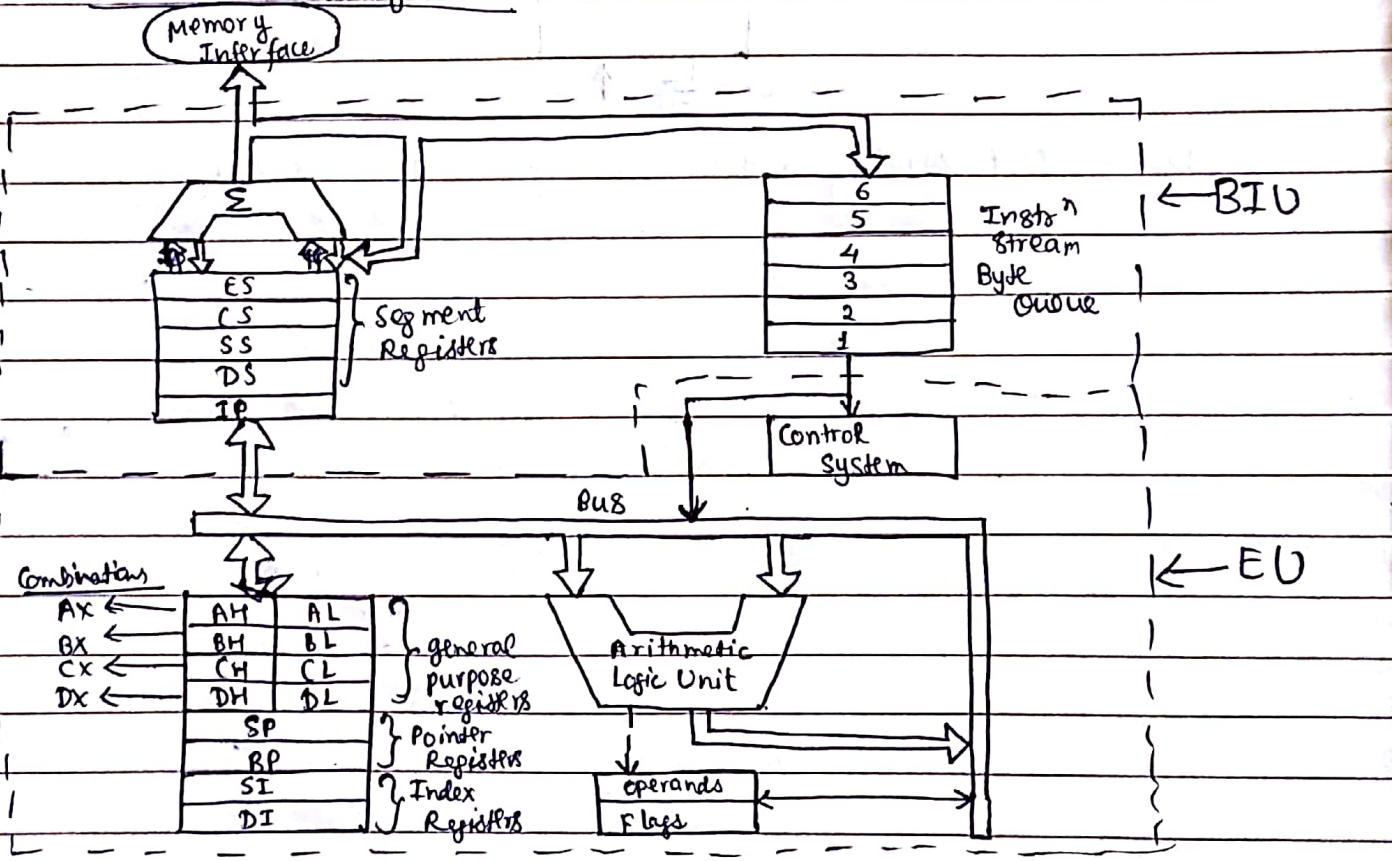
Unit-II

8086 microprocessor

* Address lines = 20 bit ∴ max memory = 2^{20} = 1 MB memory
  Data bus = 16 bit

* **Architecture of 8086:**
  - It is divided into two parts:
    (i) **BIU (Bus Interface Unit)**
    (Send address, fetch instⁿ, data read/write), i.e., handle all addresses & buses
    (ii) **EU (Execution Unit)**
    Tells BIU where to fetch instⁿ or ~~no~~ data, decode & execute instⁿ

  - **Block Diagram of 8086**



Details:

(i) **Address pins:** 20 pins ($A_{19} - A_{16}$) & ($AD_{15} - AD_0$)

(ii) **Data pins:** 16 pins ($AD_{15} - AD_0$)

(iii) **Register set of 8086:**

(a) general purpose registers (shown in diagram)
(b) segment registers
(c) Index registers
(d) Pointer registers

General Purpose Registers

ⓐ AX / AL (Accumulator)

ⓑ BX / BL (Base Register) — If Segment name not present then use DS register

ⓒ CX / CL (Counter)

ⓓ DX (Data Register)

Segment Registers – Holds starting address of segments

ⓐ CS Register – Holds starting address of code segment

ⓑ DS Register – " " " " data "

ⓒ SS Register – " " " " stack "

ⓓ ES Register – " " " " extra "

To get starting address of address it is enough to specify only the MSBs with 4 LSB always taken as 0's.

Index Registers

ⓐ SI → Source Index (EA of string location for source) — EA → Effective address

ⓑ DI → Destination Index (" " " " for destination)

Pointer Registers

ⓐ Stack pointer (SP) → Top of stack

ⓑ Base pointer (BP) → Indicate Address

ⓒ Instruction pointer (IP) → address of next instruction to be executed

(iv) Flag Register / PSW (Program Status Word)

16 bit register — Undefined — same as 8085

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	OF	DF	IF	TF	S	Z	X	AC	X	P	X	CF

Ignored

CF → Carry flag

P → Parity flag

AC → Auxiliary Carry flag

Z → zero flag

S → Sign flag

TF → Single step Trap flag (TRAP)

IF → Interrupt Enable flag (INTR)

DF → Direction flag (string)

OF → Overflow flag

IF, TF, DF → Controlling Flags

CF, P, Z, S, AC, OF → Conditional Flags

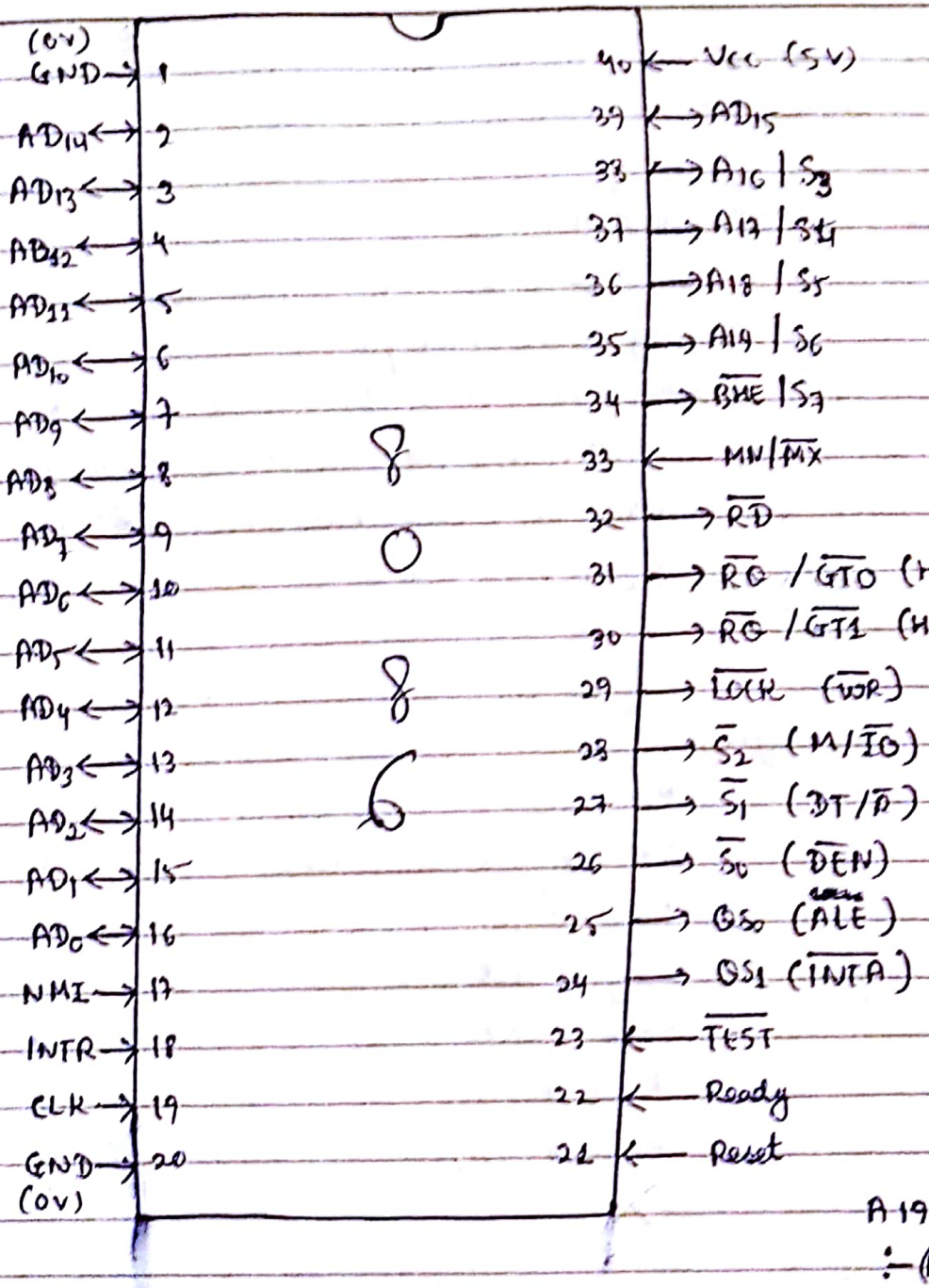
(v) ALU : For arithmetic & logical operations

(vi) Instruction Queue (IQ) :- Contains prefetched instructions
- Helps in pipelining (speeding up processing)
- Pipelining is fetching of instⁿ by BIU & execution of instⁿ by EU simultaneously
- Dump if certain branch instructions

* Segmentation of Memory used with 8086
- The memory connected with 8086 is divided into four memory segments (Code memory segment, Data memory segment, Extra memory segment, Stack memory segment)
- These segments will be maximum of 64K memory location.
- 4 LSBs are always 0's of these segments.
- (i) Code Memory Segment (CMS)
  It is used to store only instruction codes of the program.
  (ii) Data Memory Segment (DMS)
  It is used to store data as well as result.
  (iii) Extra Memory Segment (EMS)
  It is used to store data as well as result.
  (iv) Stack Memory Segment (SMS)
  It is used for storing stack of useful data using PUSH & POP instructions.

* Generation of physical address
- Address of location in memory is called physical address
  $$P.A = B.A. + E.A.$$
- Base address (B.A.) is the starting address of segment
- Effective address (E.A.) or offset or logical address is the displacement from base address.

* 8086 pin diagram :



Multiplexed
$AD_0 - AD_{15}$: Address Data Bus Line

$A_{16} - A_{19}$: Address lines

NMI : Non maskable interrupt (Positive edge triggered)

INTR = Maskable Interrupt (Level triggered)

CLK = Clock

Reset = Reset microprocessor

Ready = 1 - Ready
0 - wait for I/O

TEST = If 0, do not enter wait state
If 1, enter wait state

RD = Read data

Vcc = +5V

GND = Ground (0V)

MN/MX = Minimum/Maximum mode
{ 1 = Minimum
0 = Maximum

BHE / S7 (Bus High Enable / Status line)
→ To enable AD_8 to AD_{15}

A19/S6 — A16/S3
: Address Line / Status Signals

When MN/MX = 1, i.e., minimum mode

INTA : Interrupt Acknowledge

ALE : Address latch Enable

DEN : Data Enable ⇒ States microprocesses ready to transmit & receive data

DT/R : Data Transmit / Receive

M/IO : Memory / (Input/Output)

WR : Write

HOLD : Hold request
HLDA : Hold acknowledgment
} I/O synchronization

When $MN/\overline{MX} = 0$, i.e., maximum mode

QS_0, QS_1 (Queue Status):

QS_1	QS_0	Status
0	0	No operation
0	1	1st byte of (opcode) from queue
1	0	Queue Emptied
1	1	Next byte (opcode) from queue

$\overline{LOCK}$: Bus priority work control. This indicates that other bus masters can't gain control of system bus when $\overline{LOCK} = 0$

Priority:
More — $\overline{RQ}/\overline{GT0}$
Low — $\overline{RQ}, \overline{GT1}$
} Request/grant bus access control

$\overline{S_2}, \overline{S_1}, \overline{S_0}$ status signal

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Bus cycle
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Fetch
1	0	1	Read memory
1	1	0	Write memory
1	1	1	X

* Modes of operation of 8086.

(a) Minimum Mode

In this mode, there is only one microprocessor, & all control signals are given out by the microprocessor chip itself.

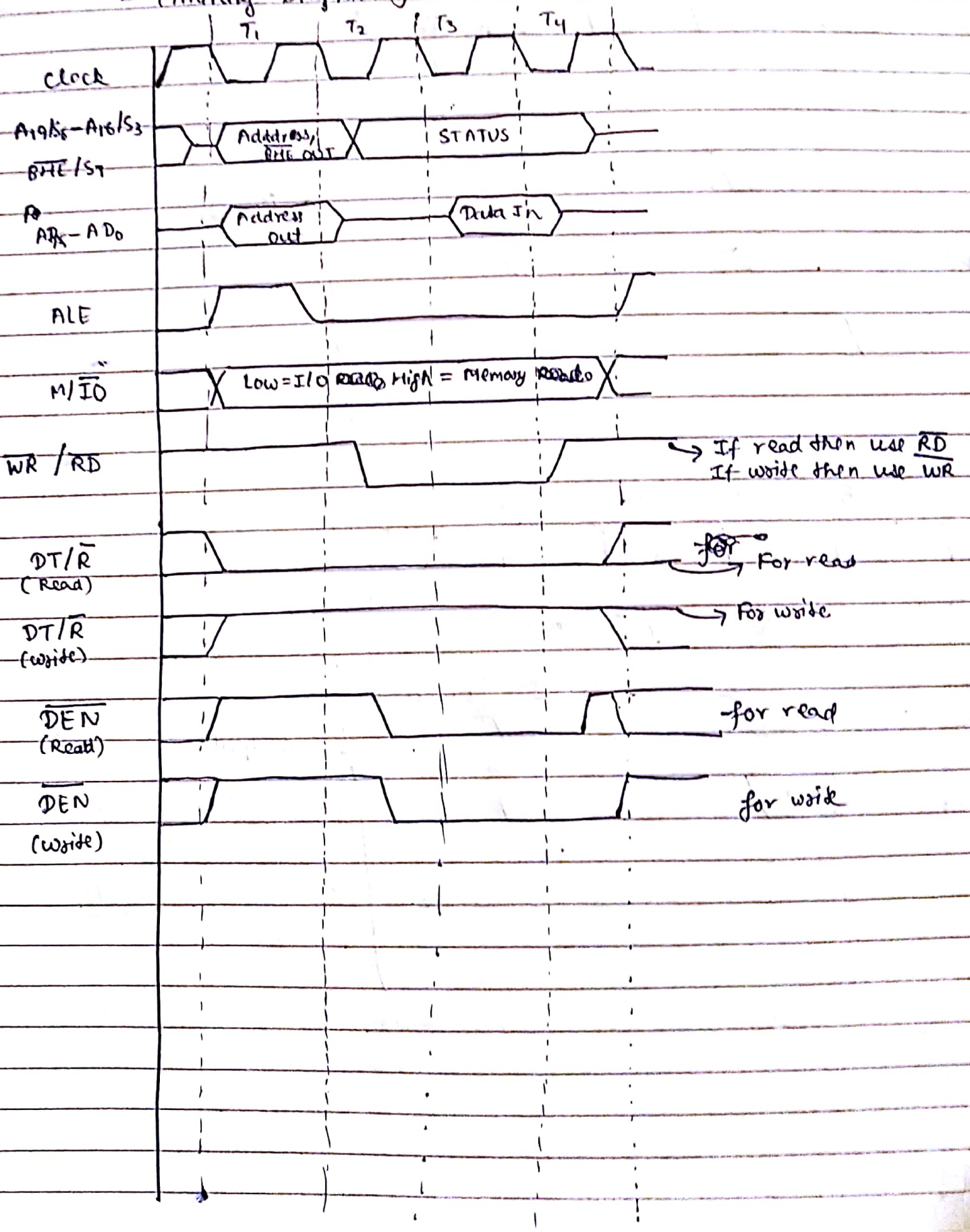
(b) Maximum Mode

In this mode, there may be more than one microprocessor (multiprocessor system) in the system configuration. The other component in the system Bus controller which drives the control signals using the status signals $\overline{S_2}$, $\overline{S_1}$ & $\overline{S_0}$.

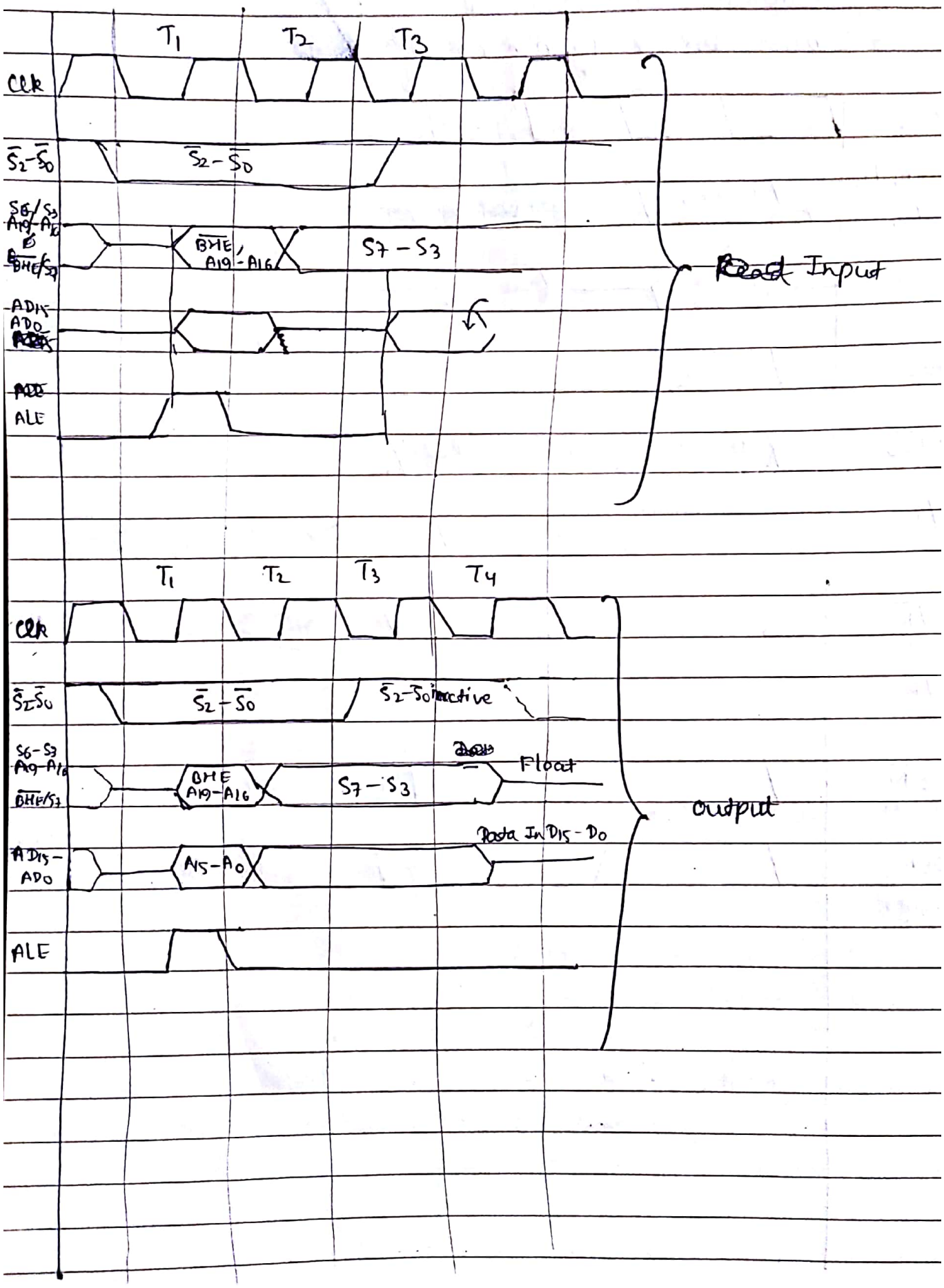
* 8086 Bus Cycle

* - Minimum 4 T states

- Timining Diagram for min mode



- Timing Diagram for ~~80~~ max mode



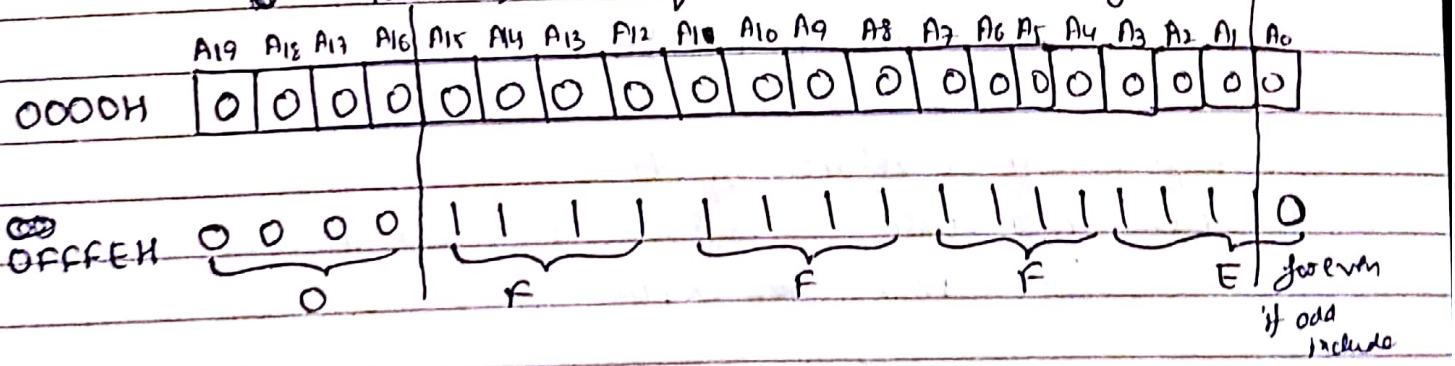
* Memory Interfacing (8086)

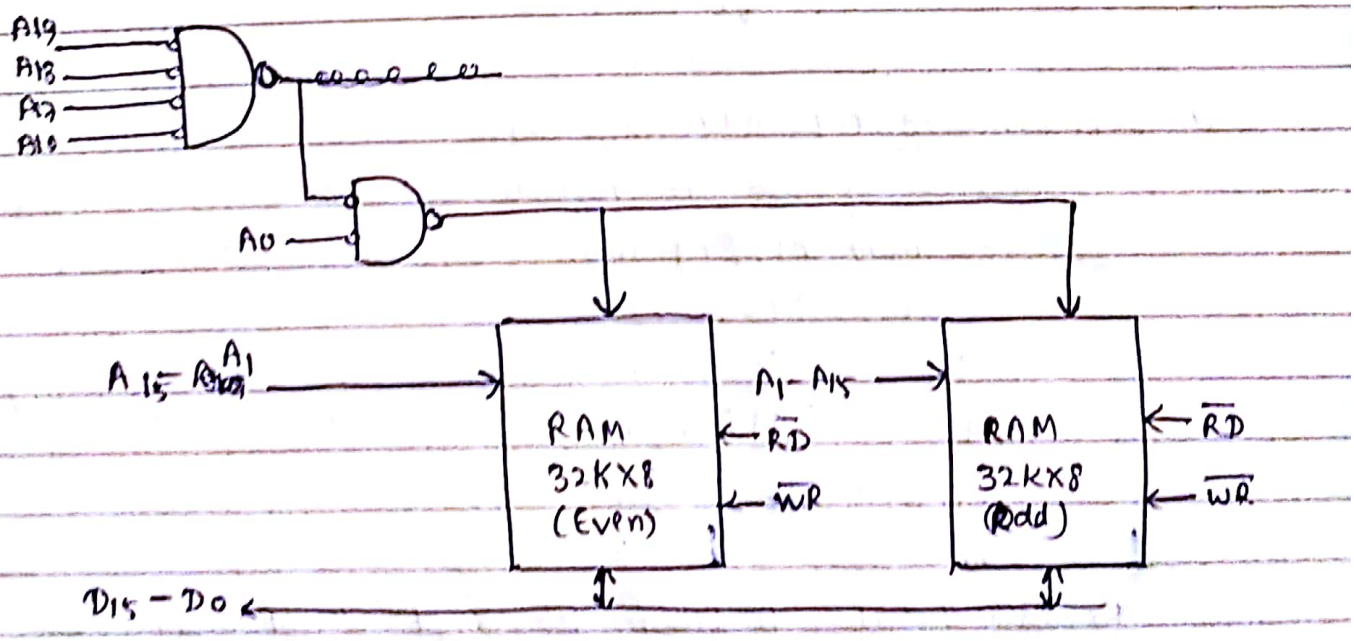
Ex:- Connect 32K word RAM with 8086

↓ means 2 byte

So 32K = 2^{15} words = 64K byte

Note ⇒ requires even & odd memory banks





* Addressing Modes of 8086

(a) For data related Instⁿ

(i) Immediate Addressing Mode

Data along instruction

Ex:- MOV AL, 75H

(ii) Direct Addressing Mode

address of data with instruction

ex:- MOV, AL, [9106 H] if DS contains 1005 H

B.A. = 10050H

EA = 9106H

P.A. : 19156H

19156 [93H] (M) ⟶ [AL]

(iii) Register Direct Addressing Mode

Data in register & register given with instruction

ex:- MOV CX, BX

[1807] (BX) ⟶ [1807] (CX)

(iv) Register Indirect Addressing Mode

address of data in register & register given with instruction

ex:- MOV AL, [SI] with DS contains 1903H

say SI contains 8165H

BA = 19030 H

EA = 8165 H

PA = 21195 H

+8 / 17 = 15+2 ⊖ means ↓

21195 [93] [M] ⟶ [93] AL

(v) Register Relative Addressing Mode or Relative Addressing Mode

Similar to register indirect but with displacement (8 bit/16 bit)

E.A = [BX] ⊕ / [BP] / [SI] / [DI] + 8/16 bit disp

ex:- MOV AL, 15H [SI] or MOV AL, [SI+15]

BA = 19060 H

say DS = 1906 H

SI = 2310 H

EA = 2310 + 15 = 2325 H

BA = 19060

EA = +2325

PA = 1B385 H

1B385H [53] (M) ⟶ [53] AL

(vi) Base Index Addressing Mode (Use Base Index BP or AX Register for Ram)

If data present in memory location then

EA = [BP]/[BX] + [SI]/[DI]

ex:- MOV CX, [BX][SI] or MOV CX, [BX+SI]

Let BX = 1573, SI = A2C2, DS = 1723

EA = 1573 + A2C2 = B735H

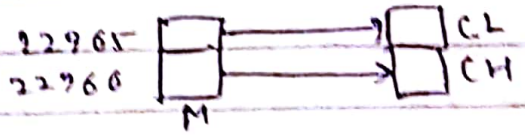
19-15 = 4

15-15 = 0

BA = 17230H

EA = B735H

PA = 22965H



(vii) Relative Base Index Addressing Mode

Similar to Base Index Addressing mode but with 8/16 bit displacement

E.A. = [BX]/[BP] + [SI]/[DI] + 8/16 bit disp

ex:- MOV DX, [BP+SI+95H] or MOV DX, 95H[BP][SI]

Let BP = 1823, SI = 2910 & SS containing 8100 & disp 19-15 = 2

EA = 1823 + 2910 + 95 = 4815 8

BA = 81000H

EA = + 4815 8H

PA = 85815 H

(viii) Implicit Addressing Mode

If address of source & destination are fixed do no operand

ex:- CLD, STD

(b) For Accessing I/O port (I/O Mode)

(i) Direct port

In this mode, the port no. is an 8 bit operand.

= This allows fixed access to port no. 0 - 255

Ex:- OUT 05H, AL

(ii) Indirect port mode

In this mode, port no. is taken forom DX allowing 64K 8 bit ports or 32K 16-bit ports

ex:- IN AL, DX

(c) For Branching Instructions

(i) Intersegment Addressing mode

- Direct (If different segments)
- Indirect (Passed indirectly, i.e., contents of memory block contains 4 bytes)

(ii) Intrasegment Addressing mode

- Direct (If same segments) (Address of same segment)
- Indirect (Displacement in same segment)

* Interrupts of 8086

- Two types:- (i) Software (ii) Hardware

- Hardware Interrupt

(i) NMI (Non maskable Interrupt) : +ve edge triggered & can't be enabled or disabled by software inst^n

(ii) INTR (Interrupt) : Interrupt request from external devices

- Software Interrupts

→ Total no - 256 ($INT_0 - INT_{255}$)

(i) $INT_0 - INT_4$: Dedicated Software Interrupt

(ii) $INT_5 - INT_{31}$: Reserved for particular operation of system

(iii) $INT_{32} - INT_{255}$: Used by user

→ Dedicated Software Interrupts (INT_0 - INT_4)

0000H (a) INT_0 ⇒ Reserved for division operation.
Take new vector starting from 4x0000 = 0000H

0004H (b) INT_1 ⇒ Dedicated for single stepping mode.
(Execution of process step-by-step / debugging)
⇒ Used with trap flag =1, TF = 1 4x0001 = 0004H

0008H (c) INT_2 ⇒ Dedicated for NMI, Executed when NMI pin=1
4x0002H = 0008H

000CH (d) INT_3 ⇒ Dedicated for Breakpoint technique (BPT) (For debugging)
4x0003H = 000CH

0010H (e) INT_4 ⇒ Dedicated for INTO (Interrupt on overflow)
⇒ Used with OF = 1 (overflow flag)
4x0004H = 0010H

- Priorities of Interrupt:

INT_0 > INT_{2-255} > INTO > NMI > INTR > INT_1

* Assembler Directives

- Instruction given to assembler regarding assembly of program are called assembler directives or pseudo instructions.

(a) END : End of program

(b) EQU : (Equate:) Used to assign label with value of symbol

ex:- LABEL EQU 0900H
INBUF EQU ADD

(c) ORG : (Origin:) Directs assembler to start memory allotment for particular segment block.

(d) DB : (Define Byte:) Used to reserve byte or bytes of memory locations in available memory.

ex:- DB 01H, 02H, 03H, 04H
Initialize 4 mem locations with these values

(e) DW: (Define Word:) Similar to DB

(f) DQ: (Define Quadword:) " " "

(g) DT : (Define Ten Bytes:) " " "

(h) ASSUME : Assume logical segment name
(i) PROC : (Procedure:) Mark start of procedure (large subroutine)
(j) MACRO : Mark beginning of macro (small subroutine)
(k) ENDP : End procedure
(l) ENDM : End macro
(m) PTR : (Pointer:) Used to declare type of label, variable or memory operand

Criteria:	Procedure	Macro
Memory Required	less	more
Parameter	Can be passed in registers, stack or memory location	passed as part of statement
Access	by call & ret during program execution	during assembly with name given to macro when defined
Machine Code	only put in memory once	each time generated for instruction

* Constructing machine code for 8086 instruction

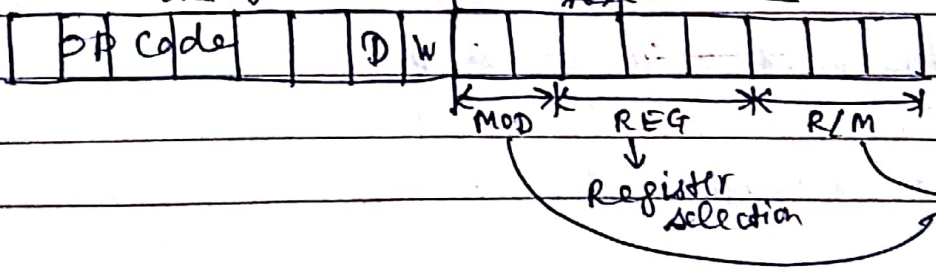
One byte Instr^n

For Direction
0 → from register
1 → to register

for word
0 → byte
1 → word

If Register to register then
MOD = 11. } not for one byte instr^n but other.

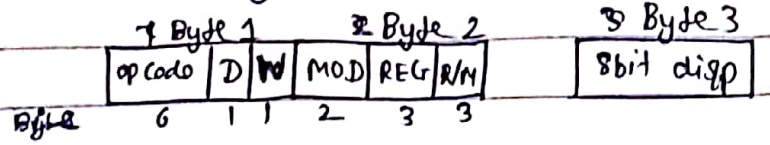
Two byte instr^n



NOTE:
Say given MOS DS: 2345 [BP] DX

Segment override prefix byte (SOP byte)

Three Byte Instⁿ

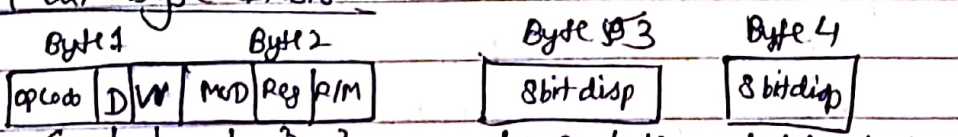


SOP byte = 001 SR 110

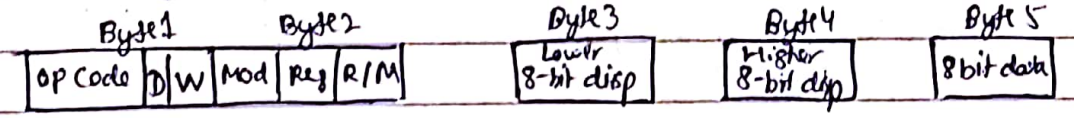
SR	Segment Register
00	ES
01	CS
10	SS
11	DS

So instⁿ code = SOP + machine code

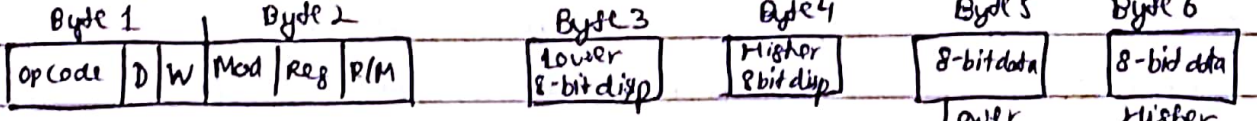
Four Byte Instⁿ



Five Byte Instⁿ



Six Byte Instⁿ



Mod = 11

REG / R/M	MOD=00	MOD=01	MOD=10	W=0	W=1
000	BX + SI	BX+SI+D8	BX+SI+D16	AL	AX
001	BX + DI	BX+DI+D8	BX+DI+D16	CL	CX
010	BP + SI	BP+SI+D8	BP+SI+D16	DL	DX
011	BP + DI	BP+DI+D8	BP+DI+D16	BL	BX
100	SI	SI+D8	SI+D16	AH	SP
101	DI	DI+D8	SI+D16	CH	BP
110	Direct Address	BP + D8	BP+D16	DH	SI
111	BX	BX+D8	BX+D16	CH	DI

D8 → 8-bit Displacement D16 → 16-bit displacement

Ex:- MOV CH, BL ; Two byte instⁿ

opcode → 1000 10

opcode	D	W	MOD	Reg	R/M
1000	1	0	11	011	101

So 8 8DD H

* 8086 Instruction Set

(i) Data Transfer Instructions:

MOV – Move
PUSH – push to top of stack
POP – pop top of stack
PUSHA – Copy all registers to stack
POPA – Copy words from stack to all registers
XCHG – EXchange
XLAT – Translate a byte in AL using look up table in memory
IN – Input from port
OUT – Output to "
LEA – Load Effective Address
LDS – " DS register & other register from memory
LES – " ES " & " " " "
LAHF – " AH with low byte of flag register
SAHF – Store " " " " " " "
PUSHF – Copy flag register to top of stack
POPF – " word at top of stack & to flag register

(ii) Arithmetic Instructions:

Addition:
ADD – Add without carry
ADC – " with "
INC – Increment by 1
AAA – ASCII adjust after addition
DAA – Decimal (BCD) " " "

Subtraction:
SUB – Subtract without borrow
SBB – " with "
DEC – Decrement by 1
NEG – Negate invert
CMP – Compare words
AAS – ASCII adjust after subtraction
DAS – Decimal (BCD) " " "

Multiplication
- MUL — Multiply
- IMUL — " Immediately
- AAM — ASCII adjust after multiplication

Division
- DIV — Divide
- IDIV — " Immediately
- AAD — ASCII adjust after division
- CWD — Convert word. int double word
- CBW — Convert upper byte of word with Copies of sign bit of lower word

(iii) Logical Instructions

NOT — Logical NOT
AND — " AND
OR — " OR
XOR — " XOR
TEST — And operand to flags but don't change operands
SHL/SAL — shift left, put zero in LSB
SHR — Shift right, " " " MSB
SAR — " ", copy old MSB into new MSB
ROL — Rotate left without CF
ROR — " right " CF
RCL — Rotate left with CF
RCR — " right " "

(iv) String Instructions

REP → Repeat until CX=0
REPE/REPZ → Repeat until CX=0 or ZF≠1
REPNE/REPNZ → Repeat until CX=0 or ZF=1
MOVS/MOVSB/MOVSW → Move byte or word from one string to other
CMPS/CMPSB/CMPSW → Compare " " " " " "
INS/INSB/INSW → Input " " " " port
SCAS/SCASB/SCASW → Scan " . Compare " " with AL or AX
LODS/LODSB/LODSW → Load " into AL or word in AX
STOS/STOSB/STOSW → Store byte from AL or word from AX
OUTS/OUTSB/OUTSW → output string byte or word to port

(vi) Branching Instructions

Unconditional:
- CALL — Call Subprogram
- RET — Return
- JMP — Jump

Conditional:
- JC
- JNC
- JZ
- JNZ
- JPE (sign flag → JS/JNS)
- JPO
- JO — Jump if overflow
- JNO — " " no overflow

- INT — Interrupt
- INTO — " overflow if OF = 1
- IRET — Return from interrupt procedure

(vii) Flag Controlling Instⁿ

- STC — set carry flag
- CLC — clear " "
- CMC — complement " "
- STD — Set direction flag
- CLD — clear " "
- STI — Set Interrupt flag
- CLI — Clear " "

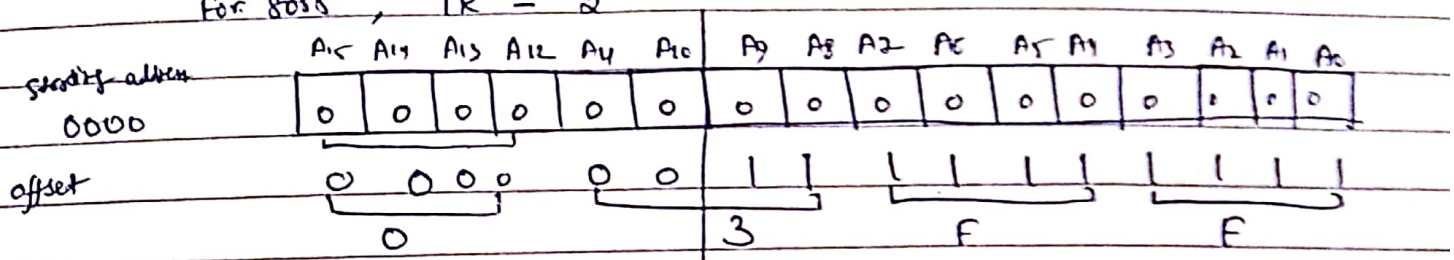
(viii) Other Instructions

- HLT — Halt
- WAIT — Stop until test pin is low
- ESC — Escape
- LOCK — Lock
- NOP — No operation

Q memory address of location of 1K memory chip is FBFFH specify starting address

Ans Assume starting address = 0000H , i.e., 0000 0000 0000 0000

for 8085 , $1K = 2^{10}$



∴ offset = 03FFH for 1K memory with starting address.

ending address = starting address + offset

∴ starting address = FBFF

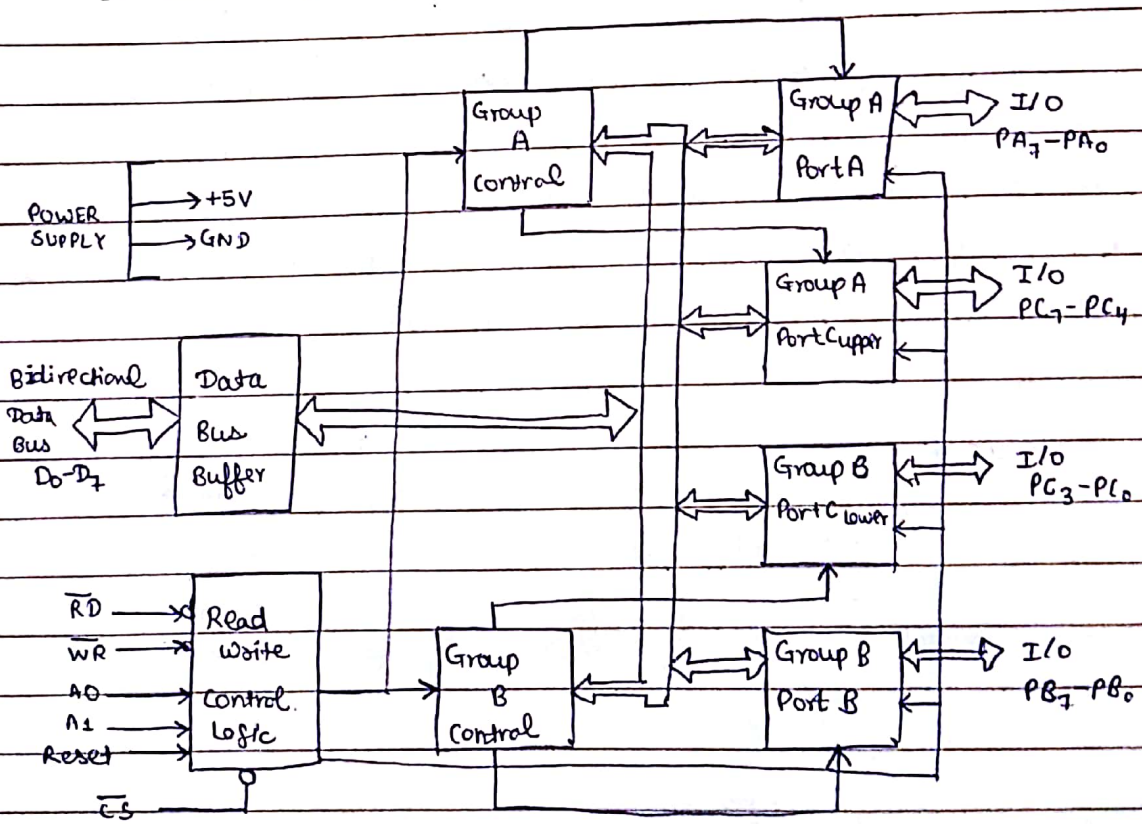
− 03FF

F8 00H

Unit-III

Programmable Peripheral Interface (PPI) – Intel 8255

* It consists of (i) Two 8-bit ports Port A and Port B
  (ii) Two 4-bit ports Port Cupper and Port Clower. These two ports can be used independently or it can be used together as one 8-bit port
  (iii) 8-bit Control word Register (CWR)
* when data transfer is performed using handshake signals then
  (i) for Port A handshake signals generated on Port Cupper pins. Port A & Port Cupper are called Group A
  (ii) for Port B handshake signals generated on Port Clower pins. Port B & Port Clower are called Group B
* Block Diagram of 8255



- Data Bus Buffer ⇒ This 8-bit tristate bidirectional buffer is used to interface 8255 to the system data bus
- Read/Write Control Logic ⇒ It accepts control bus signals as well as inputs from address bus and issues commands to verify individual group control blocks
- Group A control for group A ports & Group B control for group B ports Port C for handshake & status signals

(A_2-A_7 & operate $\overline{CS}$ for 8085 & A_2-A_{15} for 8086)

- $\overline{CS}$ (Chip Select) ⇒ Active Low input which must be enabled for data transfer operation.
- $\overline{WR}$ (write) ⇒ when this pin is made low, CPU can write data on to ports through data bus buffer
- $\overline{RD}$ (Read) ⇒ when this pin is made low, CPU can read the data in ports through data bus buffer
- A_0 & A_1 ⇒ These signals alongwidth $\overline{RD}$ & $\overline{WR}$ inputs control the selection of the control/status word register or one of the three ports.

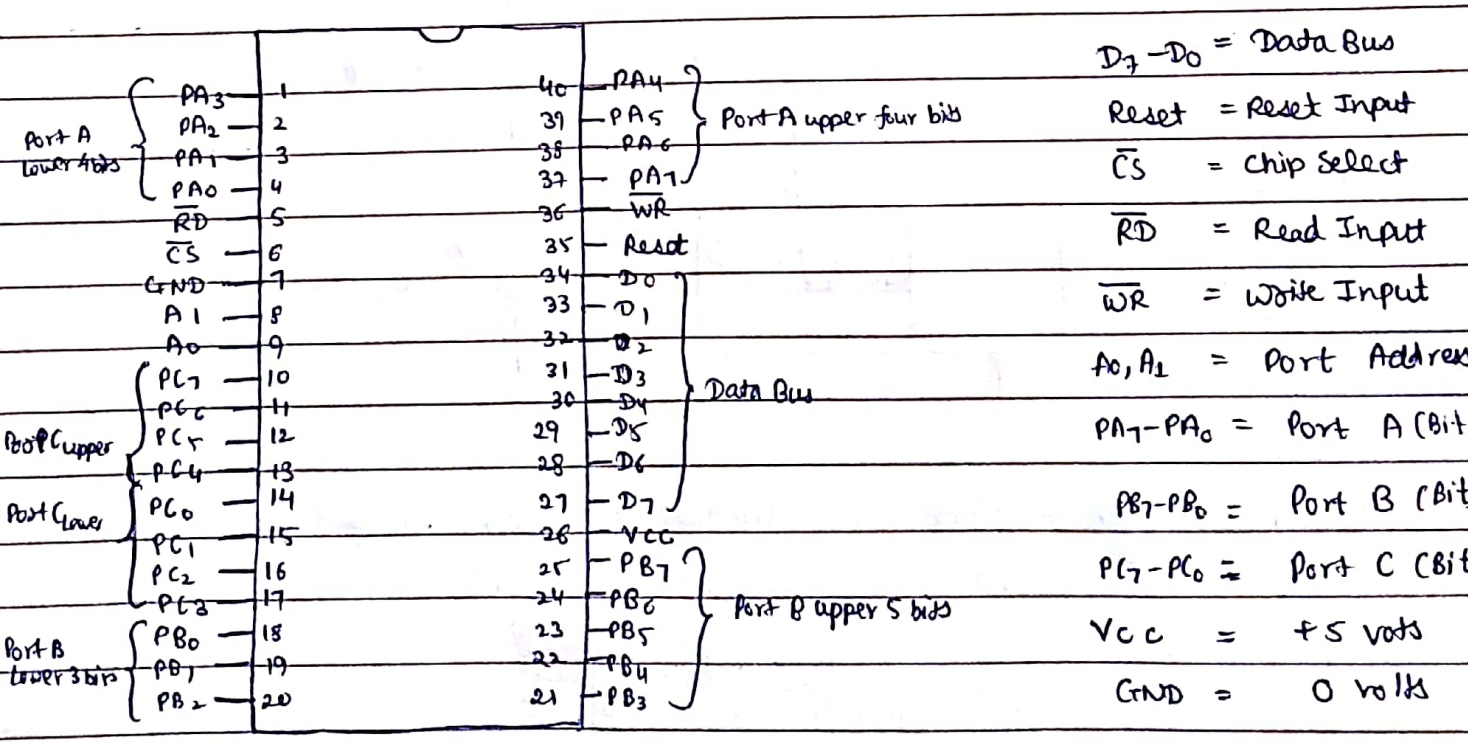
$\overline{CS}$	A_1	A_0	Selected
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control word Register
1	X	X	8255 is not selected

Hex Address (Examples)		
1A	2C	01
1B	2D	02
1C	2E	03
1D	2F	04
X	X	X

- Reset ⇒ Connected to system reset line. when this input is high, the control register is cleared & all ports are set to input mode

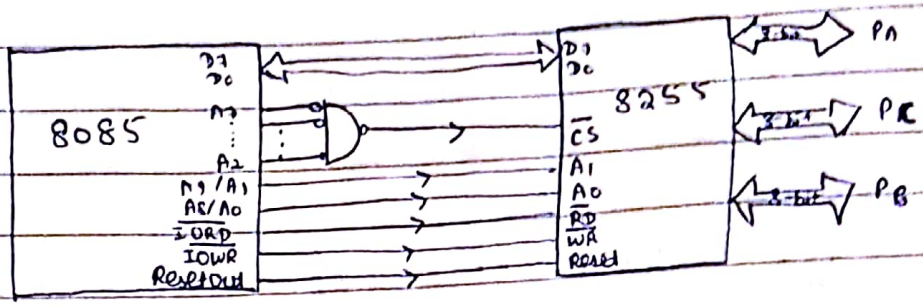
NOTE: The microprocessor sends out a control word to 8255. The control word contain information such as mode, bit-set, bit-reset etc that initializes the functional configuration of 8255.

* Pin Diagram of 8255



D_7-D_0 = Data Bus
Reset = Reset Input
$\overline{CS}$ = Chip Select
$\overline{RD}$ = Read Input
$\overline{WR}$ = Write Input
A_0, A_1 = Port Address
PA_7-PA_0 = Port A (Bit
PB_7-PB_0 = Port B (Bit)
PC_7-PC_0 = Port C (Bit)
Vcc = +5 volts
GND = 0 volts

* Interfacing of 8255 PPI with 8085



* Types of Parall Data Transfer

1) Synchronous (Data transfer speed b/w I/O device matches with microprocessor)

2) Asynchronous (Data transfer speed different b/w IO device & microprocessor)

If data speed of I/O device slow
- (a) using status check or Polled method (Status of the Signal X) 0 → not ready 1 → ready
- (b) using ready pin or wait states
- (c) using interrupt
- (d) using Handshake signal (If data speed of IO device fast or slow)
  Signal x to indicate ready or not IO device & an acknowledge signal Y to indicate ready or not microprocessor.

* Modes of Operation of 8255 PPI:

- The modes can be divided into two parts:
  (i) Parallel I/O mode (b) BSR mode (Bit Set/Reset Mode)

- Parallel I/O modes

⇒ There are three types: (i) Mode 0 (ii) Mode 1 (iii) Mode 2
(Simple I/O mode) (Strobbed I/O mode) (Strobed bidirectional bus I/O mode)

⇒ In mode 1, Port A or port B can be used for a handshake operation.

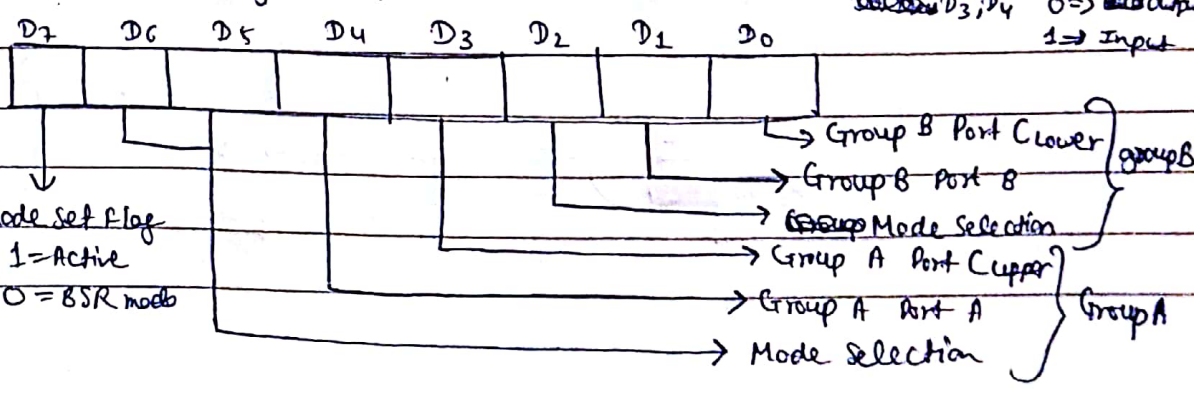
⇒ In mode 2, Port A can be used as bidirectional 8-bit I/O bus.

⇒ Control word Register for I/O mode

D_5, D_6 0 0 ⇒ Mode 0, 0 1 ⇒ Mode 1, 1 0 ⇒ Mode 2

D_2 ⇒ 0 ⇒ Mode 0, 1 ⇒ Mode 1

D_0, D_1, D_3, D_4 0 ⇒ Output, 1 ⇒ Input



- BSR mode

=> only Port C is used in BSR mode. In this mode we can transfer either logic '1' (SET) or logic '0' (RESET) on any one pin of Port C without changing the status of rest of seven pins of port C.

=> Control Word Register for BSR mode

D7	D6	D5	D4	D3	D2	D1	D0
0	X	X	X	A_2	A_1	A_0	S/R

→ Bit Set/Reset 0 → Reset 1 → Set

D6, D5, D4 : Don't Care

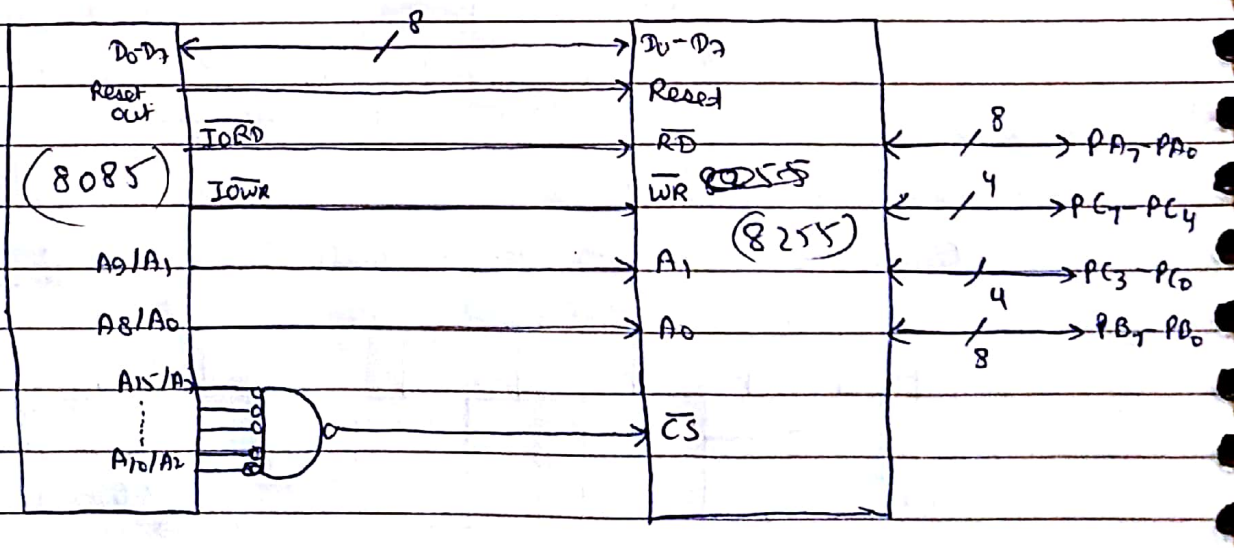
D3, D2, D1 : Indicate corresponding port bit to be set/reset

A_2	A_1	A_0	Port C pin
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

Q Interface 8255 PPI with microprocessor 8085

Ans Let 8 bit address of Port A be 00H

IC	Port	Hex Address	Binary Address: A_7 A_6 A_5 A_4 A_3 | A_2 A_1 A_0 / A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8
	Port A	00H	0 0 0 0 0 0 | 0 0
IC 8255 PPI	Port B	01H	0 0 0 0 0 0 | 0 1
	Port C	02H	0 0 0 0 0 0 | 1 0
	CWR	03H	0 0 0 0 0 0 | 1 1
			To generate $\overline{CS}$ | A_1, A_0 pin



* Interfacing of Seven Segment Display using mode 0.

Look up Table located at starts from 7000H

To translate characters to hex codes for input to display

Q Interface one 7 segment display unit with 8085 & write an ALP to display decimal digits from 0 to 9 for 1 sec each use LOOKUP table.

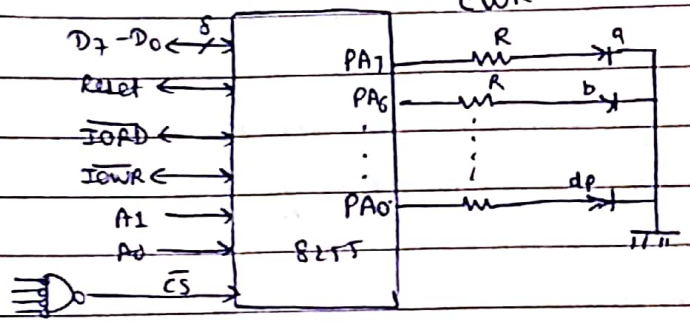
Ans Address Decoding Table:

let address of $P_A \rightarrow 00H$

$P_B \rightarrow 01H$

$P_C \rightarrow 02H$

$CWR \rightarrow 03H$




```
ALP:
LXI   SP, 000H
MVI   B, 0AH   } Counter for 0 to 9
LXI   H, 7000H } Lookup table start
MVI   A, 80H   } Define Port B
OUT   03H
L1: MOV   A, M } Lookup table to Port B
OUT   01, H
CALL  "DELAY" } 1 sec
INX   H
DCB   B
JNZ   L1
XRA   A  } Blank led at last
OUT   01H
HLT
```

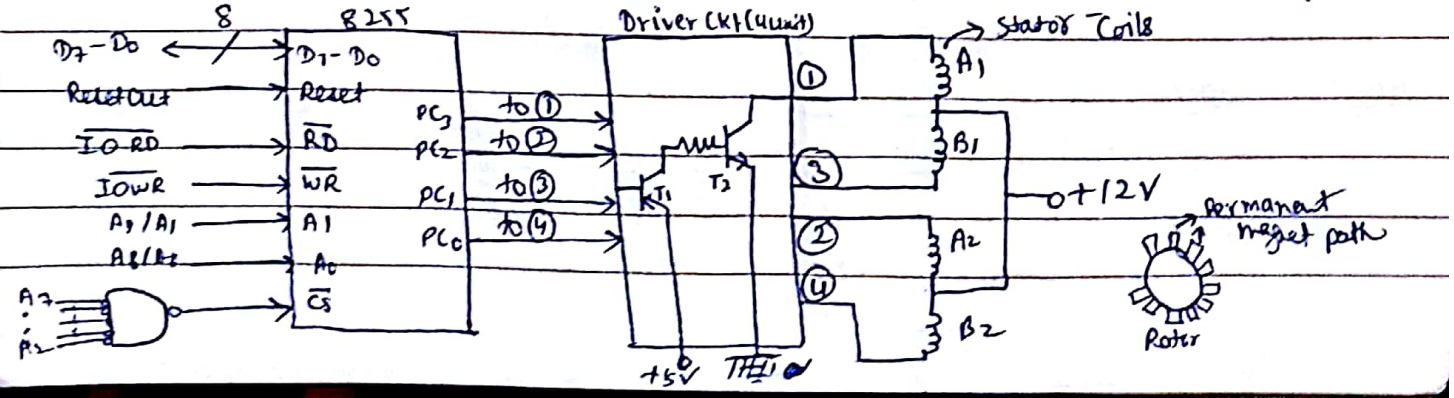
* Stepper Motor Control Using Microprocessor

Q Design the hardware & software to rotate stepper motor in clockwise at a speed of 200 rpm.

Ans For connecting case of switching transister with microprocessor, we need output port like 8255 PPI

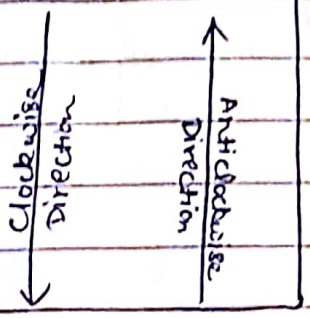
Address Decoding Table for 8255 PPI

IC	Port	Hex address	Binary Address A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 / A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0
8255 PPI	A	20H	0 0 1 0 0 0 0 0
	B	21H	0 0 1 0 0 0 0 1
	C	22H	0 0 1 0 0 0 1 0
	CWR	23H	0 0 1 0 0 0 1 1



- For direction control if initial data is 0011 (03H) then, to rotate clockwise use RRC & for anti-clockwise use RLC

Steps	Sequence of grounding PC3 (A1)	PC2 (A2)	PC1 (B1)	PC0 (B1)	Direction
1	0	0	1	1	
2	1	0	1	1	
3	1	1	0	0	
4	0	1	1	0	
5	0	0	1	1	



- For sped control: If speed N = 200 rounds per minute

Time for one round, $T = \frac{1}{N} = \frac{1}{200}\text{min} = 0.3$ se

If one round = 200 steps

Time for one step = $0.3/200 = 1.5$ msec

So $t_d = 1.5$ msec

- A Software:

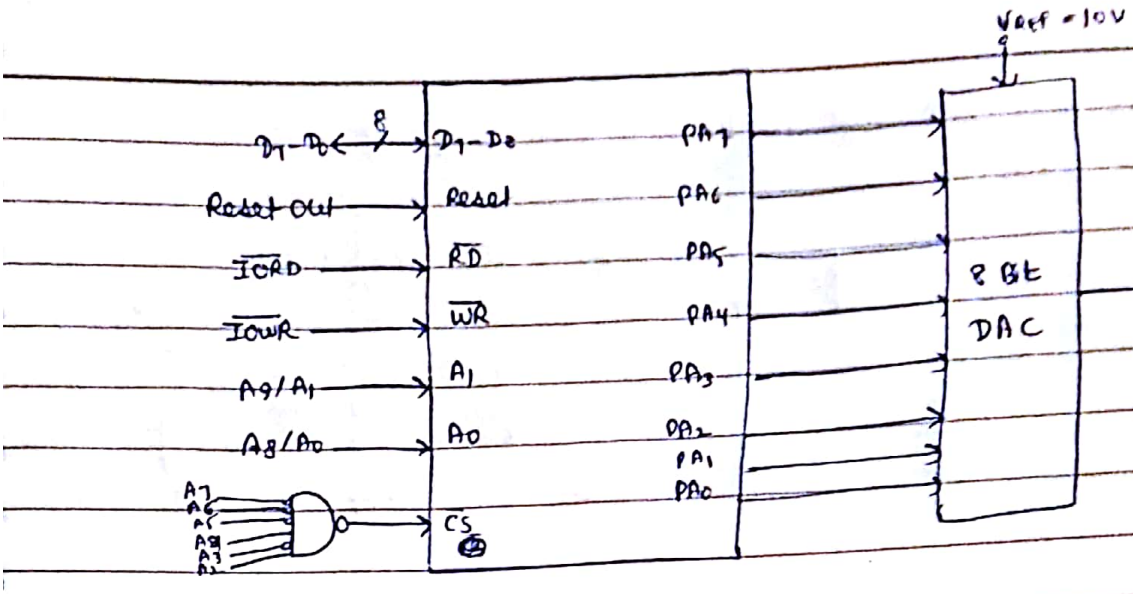
```
LXI   SP, 0000H
MVI   A, 80H      } Port Defined
OUT   23H
MVI   A, 03H      Duplicating 3H data which is initial data for step 1.
L1:  OUT   22H
CALL  DELAY '1.5msec'
RRC
JMP L1
HLT
```

* Interfacing of Digital to Analog Converter (DAC) with 8085 CPU

Q Connect one 8 bit DAC with 8085 CPU & write a software to generate sawtooth waveform on DAC output.

Ans Address Decoding Table for 8255 PPI

IC	Port	Hex address	Binary Address A15 A14 A13 A12 A11 A10 A9 A8 / A7 A6 A5 A4 A3 A2 A1 A0
8255 PPI	A	14 H	0 0 0 1 0 1 | 0 0
	B	15 H	0 0 0 1 0 1 | 0 1
	C	16 H	0 0 0 1 0 1 | 1 0
	CWR	17 H	0 0 0 1 0 1 | 1 1



Software: CWR : 1 0 0 0 0 0 0 0

Program:

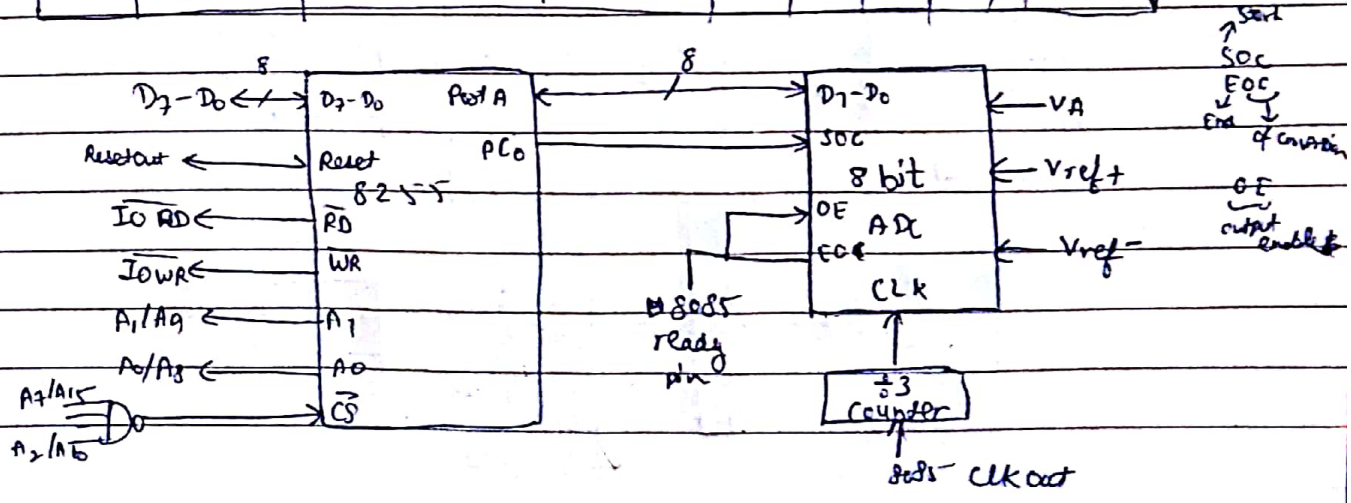
```
MVI  A,80 H
OUT  17 H
XRA  A
L1: OUT  14H
INR  A
JMP  L1
HLT
```

* Interfacing of Analog to Digital Converter (ADC) with 8085

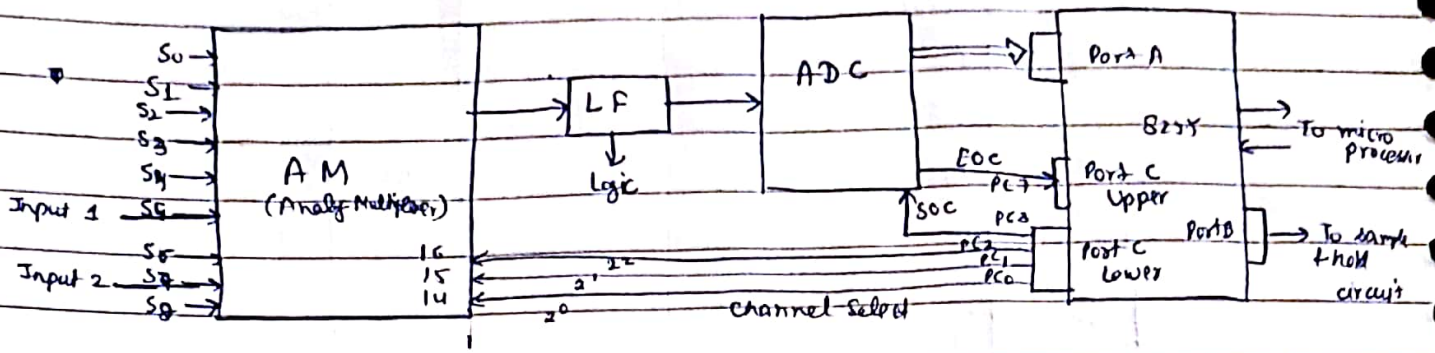
Q Interface 8 bit ADC with microprocessor 8085.

Ans Address Decoding Table

IC	Ports	Hex Address	Binary Address A15 A14 A13 A12 A11 A10	A9 A1	A8 A0
8255	A	7C H	0 1 1 1 1 1	0	0
	B	7D H	0 1 1 1 1 1	0	1
	C	7E H	0 1 1 1 1 1	1	0
	CWR	7F H	0 1 1 1 1 1	1	1

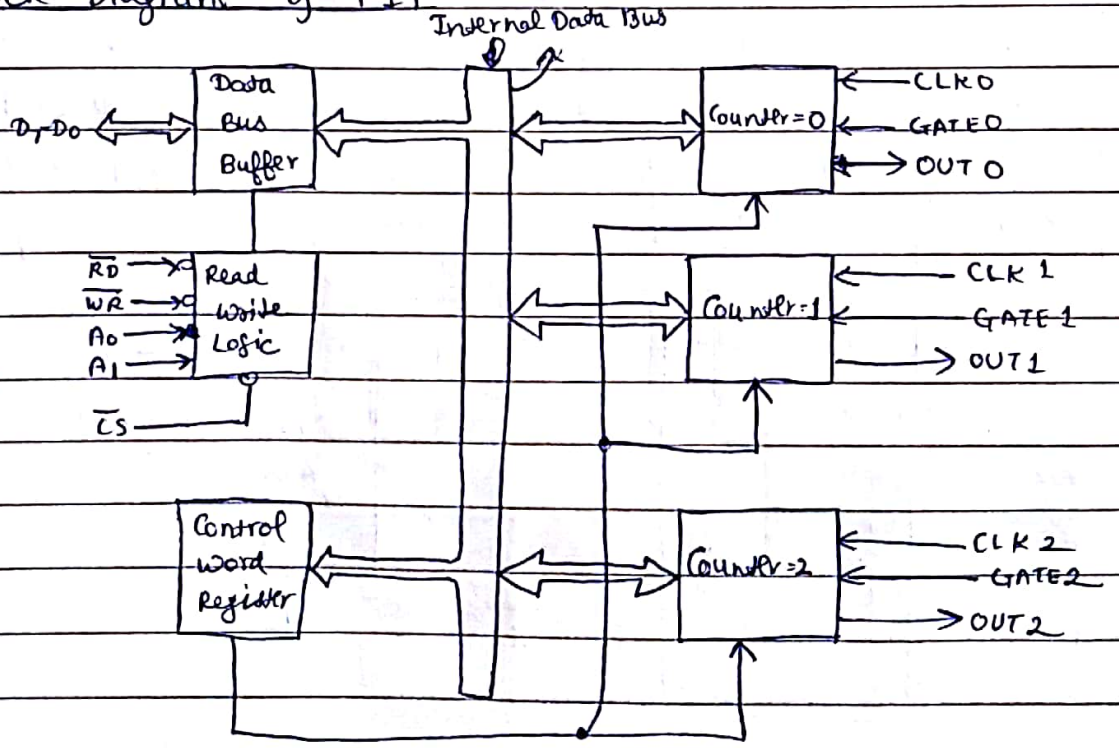


* Interfacing of ADC, Analog Multiplexer & Sample & Hold:



Programmable Interval Timer (PIT) - Intel 8253/8254

* PIT (8253) is a programmable interval timer/counter which can generate accurate time delays and waveforms ranging 0 Hz to 2 MHz using software control. (8254) PIT has higher range (8 MHz).
* 8253 is a 24 pin IC & consists of three identical but independent 16 bit down counters. Each counter can be used either as 16-bit binary counter or as 16 bit BCD counter. For binary counter, max data is FFFFH & for BCD counter, max data is 10000D
* Each counter will decrement by one at each negative edge of clock input & can be used in six different modes, Mode 0 to Mode 5. In different modes different types of waveform are obtained at OUT pin.
* Block Diagram of PIT



- Data Bus Buffer does the following functions:
  ⓐ Programming the modes of 8253 ⓑ Loading counter register
  ⓒ Reading Count value
- Read/Write Logic section have five signals, i.e., $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_0 & A_1

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Write into counter 0
0	1	0	0	1	Write into counter 1
0	1	0	1	0	Write into counter 2
0	1	0	1	1	Write control word
0	0	1	0	0	Read from counter 0
0	0	1	0	1	Read from counter 1
0	0	1	1	0	Read from counter 2
0	0	1	1	1	No-operation
1	x	x	x	x	No-operation
0	1	1	x	x	No-operation

- Control word Register is accessed when $A_0 = 1$ & $A_1 = 1$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

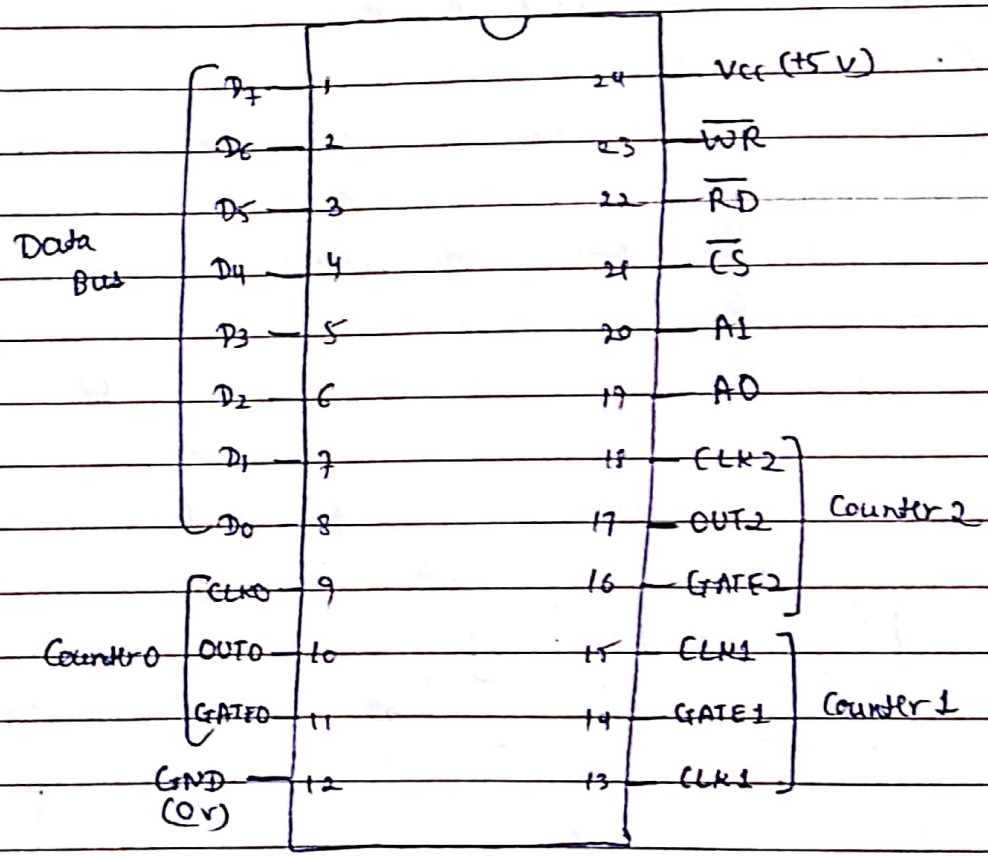
Select counter | Read load / Read write | Mode of operation | → $D_0 = 1$ then BCD Counter, $D_0 = 0$ then Binary Counter

SC1	SC0	Selection
0	0	Counter 0 selected
0	1	Counter 1 selected
1	0	Counter 2 selected
1	1	Invalid for 8253 / Read back for 8254

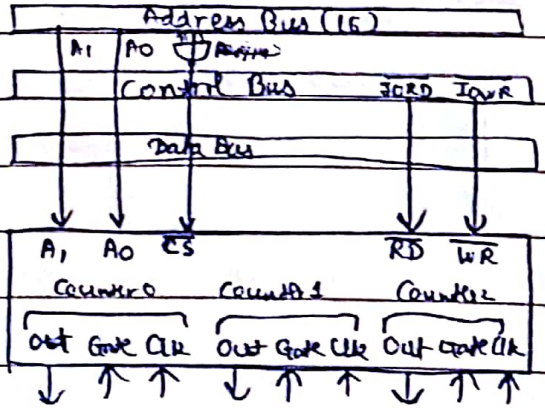
RL1	RL0	LSB / MSB of Counter register selected
0	0	Read on-fly operation (Counter latch Command)
0	1	Read and load only LSB of Counter Register
1	0	Read and load only MSB of Counter Register
1	1	Read and Load LSB first & MSB next

M2	M1	M0	Mode Defines	
0	0	0	Mode 0	Interrupt on Terminal Count
0	0	1	Mode 1	Programmable Retriggerable Monostable
X	1	0	Mode 2	Rate Generator
X	1	1	Mode 3	Square Wave Generator or Divide By N Counter
1	0	0	Mode 4	Software Triggered Strobe
1	0	1	Mode 5	Hardware Triggered Strobe

* Pin diagram of 8253



* Interfacing of 8253/8254 with microprocessor 8085



* Differences between PIT 8253 & 8254.

1) Max input clock frequency of 8253 is 2.6 MHz whereas for 8254 it is 8MHz.

2) 8253 cannot perform read back operation whereas 8254 can.

* There are three possible methods for reading counts:

(a) Simple read operation

(b) Counter latch command ⇒ This command written to CWR when A1A0 = 11.

(c) Read Back Command ⇒ This command written to CWR.

⇒ This command allows user to check count value, programmed mode, & current state of OUT pin & null count flag of selected counter.

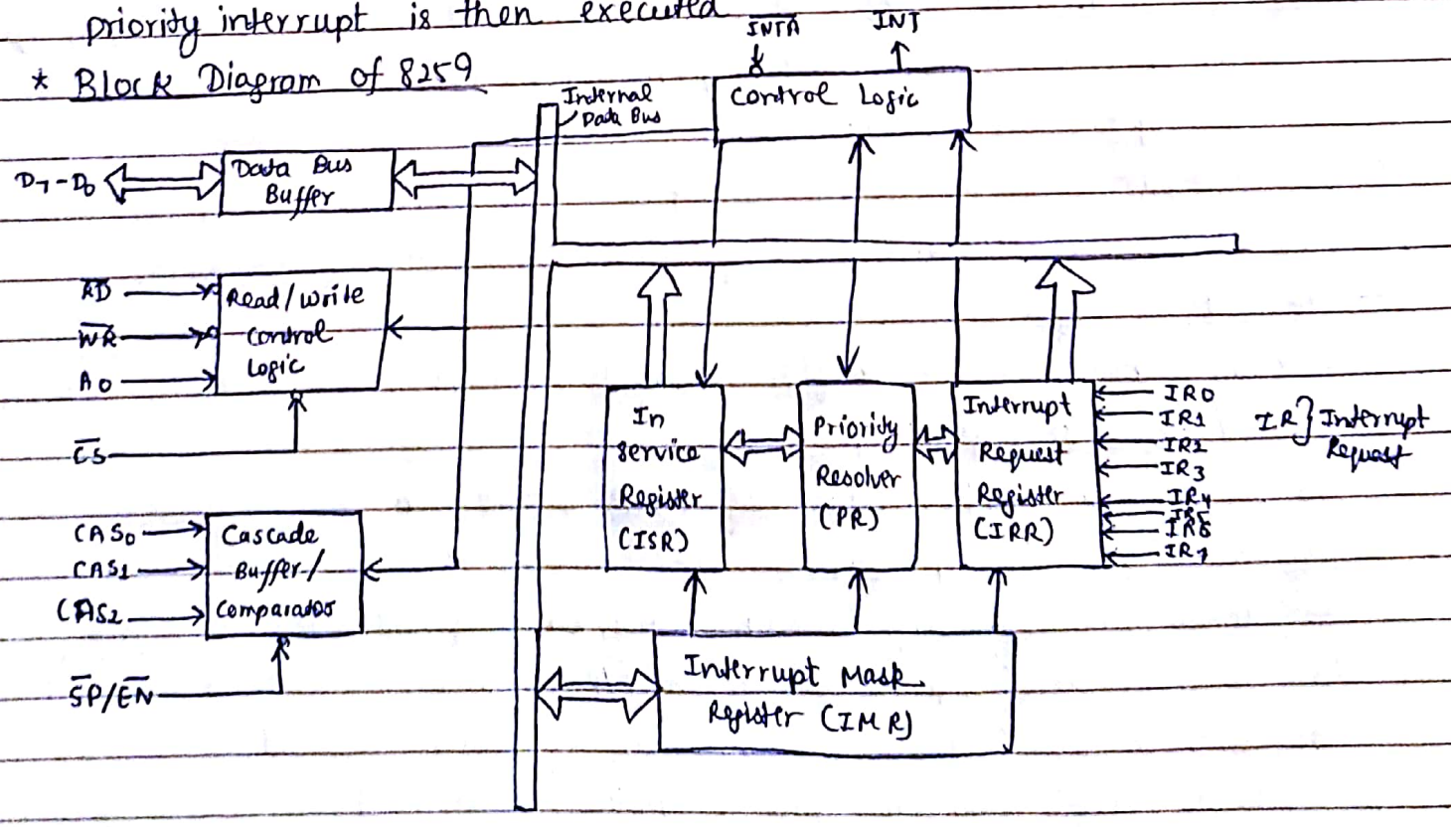
⇒ May be used to latch multiple counter output latches (CLE) (OL).

Priority Interrupt Controller (PIC) - 8259

* PIC is compatible with 8085, 8086, 8088 & can handle 8 separate interrupt requests. PIC is a 28 pin n-mos based IC.

* PIC works as an overall manager in an interrupt driver data transfer scheme. It accepts requests from the peripheral equipments, determine which of the incoming request is of the highest priority. The highest priority interrupt is then executed.

* Block Diagram of 8259



- 8259 can be divided into four main sections:
  1) Interrupt and control Logic section 2) Data Bus Buffer
  3) Read/write control logic section 4) Cascade Buffer/Comparator

- Interrupt and control logic section
  ↳ Has 8 different interrupt request levels (IR_0-IR_7)
  ↳ Has following parts:
  (i) Interrupt Request (IR): Eight different interrupting devices can be connected to these eight IR levels. These IR levels can be defined either as edge triggered or level trigger.
  (ii) Interrupt Request Register (IRR): 8 bit register which contains 8 flipflops D_0-D_7 corresponding to eight IR level IR_0 to IR_7. When any device gives IR signal, the corresponding bit of IRR is set (=1) otherwise remains zero.
  (iii) Interrupt Mask Register (IMR): 8 bit register which contains 8 flip flops corresponding to eight IR pins IR_0 to IR_7. If any IR levels is to be disable (mask) then corresponding bit of IMR should be set.
  (iv) Priority Resolver (PR): Stores priorities of IR_0 to IR_7. By default, IR_0 has highest priority & IR_7 lowest.
  (v) In service Register (ISR): It is 8 bit register which contains 8 flip flops for IR_0 to IR_7. It stores information regarding the interrupts which the microprocessor is servicing.
  (vi) Control Logic: It accept output commands from CPU. It set CWR (Control word Register) according to CPU command.

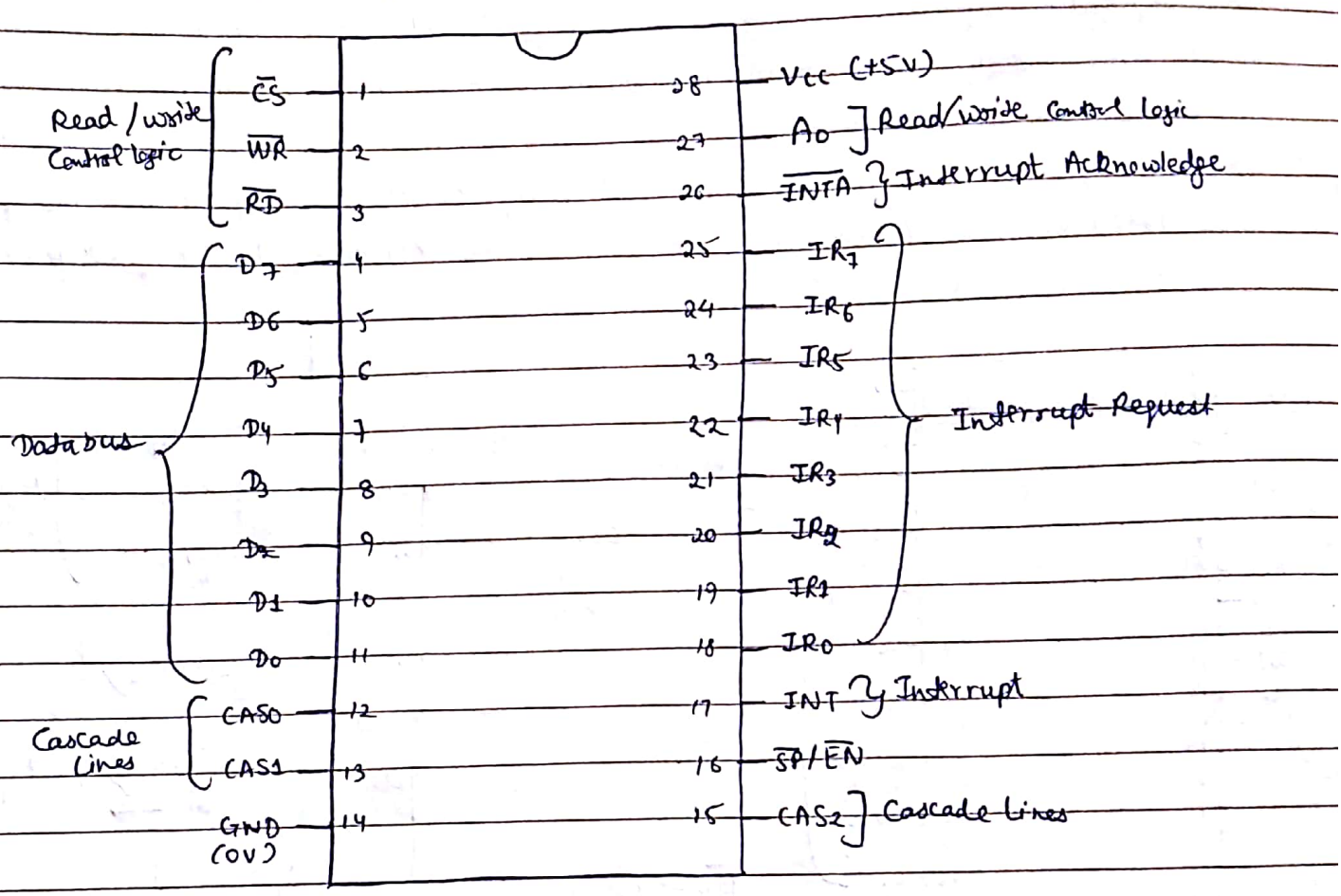
- Data Bus Buffer
  ↳ 8 bit bidirectional buffer used to interface 8259 to system data bus.

- Read/write Control Logic
  ↳ Used for initialization of CWR

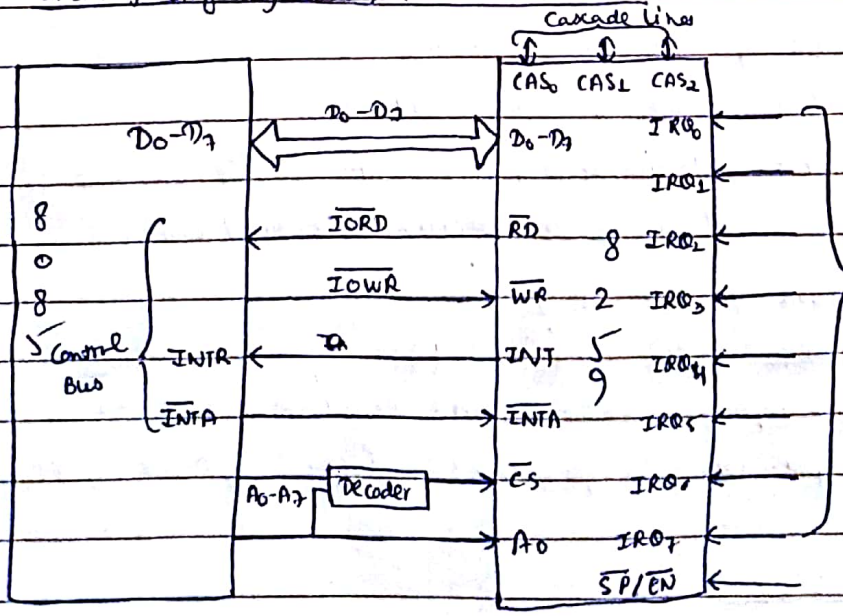
- Cascade Buffer/Comparator
  ↳ Three cascade signals CAS_0, CAS_1, CAS_2 & $\overline{SP}/\overline{EN}$ (slave program/enable buffer) pin.
  ↳ Used to send signals from master O/P to slaves when cascaded devices.
  ↳ Signals lines (CAS_0, CAS_1, CAS_2) indicates slave selected
  ↳ $\overline{SP}/\overline{EN}$ dual function pin. Master if it is 1 & slave if 0 in non-buffered mode

In buffered mode, an output that controls data bus transreceivers. The one connected to 8085 is called master & other as slaves.

* Pin diagram of 8259

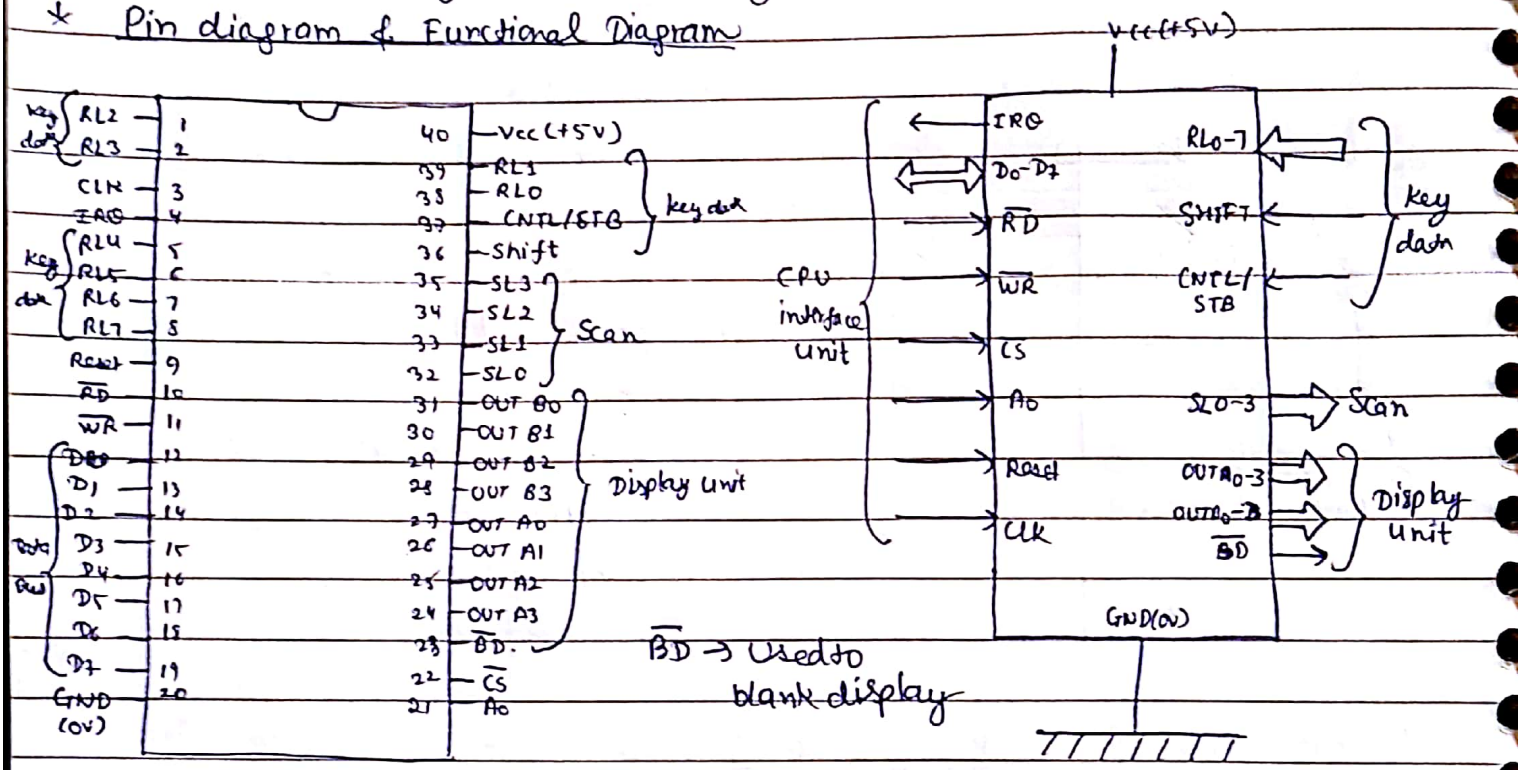


* Interfacing of 8259 (PIC) with 8085



Programmable Keyboard / Display Controller – Intel 8279

* 8279 is a programmable keyboard & display controller. It is a 40 pin IC & has two major segments: keyboard & display.
* The keyboard segment can be connected to a 64-contact key matrix keyboard entries are debounced & stored in internal FIFO memory; an interrupt signal is generated with each entry.
* The display segment can provide a 16-character scanned display interface & has 16x8 R/W memory (RAM). The display can, either left or right entry, be setup in format.
* Pin diagram & Functional Diagram



– 8279 functionally consists of four major sections:

① CPU interface (Microprocessor interface section)

ⓐ Consists of 8 bit data bus D_0-D_7 along CLK, $\overline{CS}$, $\overline{RD}$, $\overline{WR}$ & IRQ lines

ⓑ $A_0 = 1$, signal in/out pertain to status / command
$A_0 = 0$, data

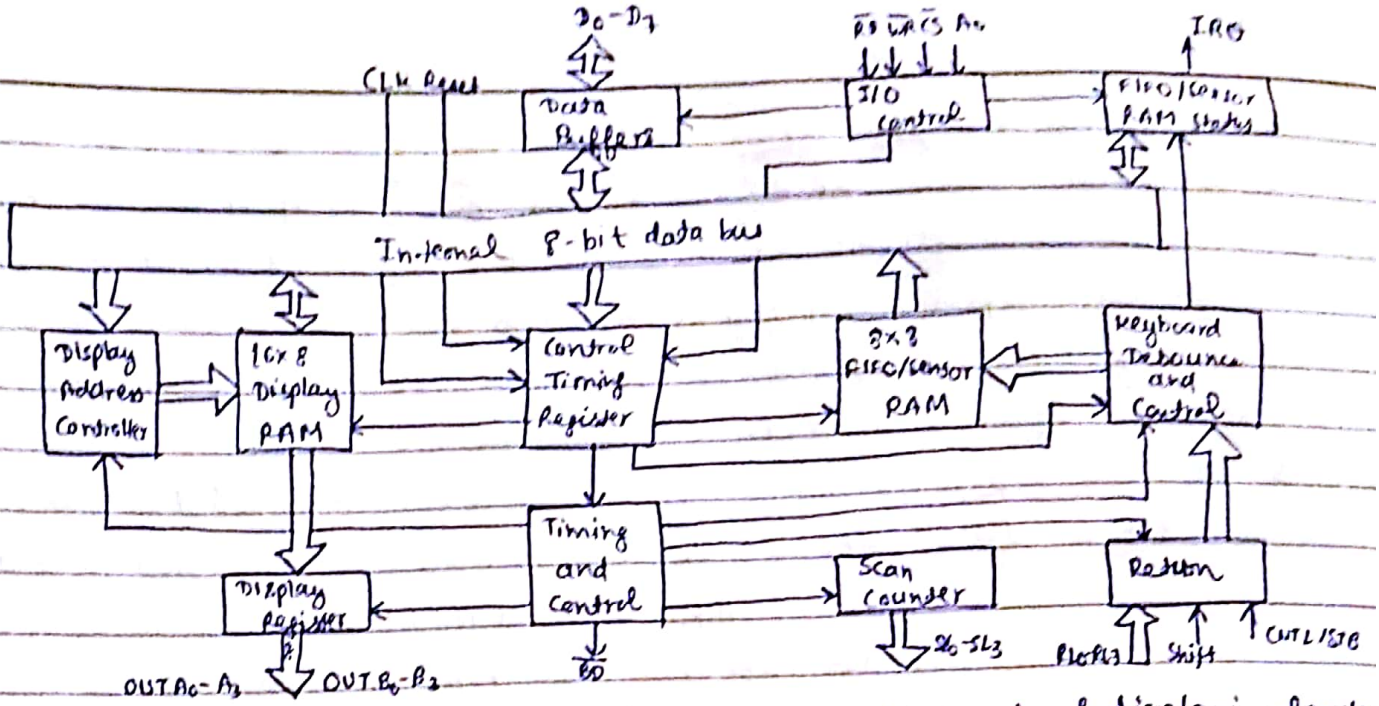
ⓒ IRQ = 1, data in FIFO

② Set of Scan Lines (Scan Section) (SL0-SL3)

③ Input Lines for key display (RL0-7, SHIFT, CNTL/STB)

④ Output lines for Display Unit (Display Unit) ($OUTA_{0-3}$, $OUTB_{0-3}$, $\overline{BD}$)

* Block Diagram of 8279



- 8279 has to perform two operations, keyboard scanning & displaying characters
- Data bus buffer interfaces 8279 to system bus

- I/O section
  - ⇒ I/O section is enabled when CS is low.
  - ⇒ A_0 pin indicates transfer of command or status information when high.
  - ⇒ $\overline{RD}$ & $\overline{WR}$ are read & write signals respectively.
- Control Timing registers & Timing Control
  - ⇒ Reset forces CPU into idle state
  - ⇒ $\overline{BD}$ is used to blank display
  - ⇒ Clk is used to generate clock signal for timing
  - ⇒ Control Timing registers & timing control registers store the keyboard & display modes & other operating conditions programmed by CPU.
- FIFO/Sensor Ram & Status logic
  - ⇒ In keyboard or strobed input mode, acts as 8 byte FIFO RAM.
  - ⇒ IRQ is interrupt output line which goes high when is data in FIFO RAM
- Display Address Registers & Display RAM
  - ⇒ These registers hold address of word currently being read/written by CPU to form display RAM (16 byte)
  - ⇒ OUT A_0-A_3 & OUT B_0-B_3 are O/P ports for two 16×4 internal display refresh registers.
- Scan Mode
  - ⇒ The scan counter has 2 modes to refresh key matrix & refresh display.
  - ⇒ In encoded mode, counter provided binary count that is to be externally decoded to provide the scan line for keyboard & display.
  - ⇒ In decoded mode, counter internally decodes 2 LSBs (bits) & provides a decoded 1 out of 4 scan on SL0-SL3
  - ⇒ Keyboard & display are in same mode at a time.

- Return Buffers & Keyboard debounce & Control

⇒ This section scans for a key closure row wise. If closure detected, keyboard debounce unit debounces the key entry, i.e., wait for 10 ms. After debounce period, if key continues to be pressed, code of key is directly transferred to sensor RAM along with SHIFT & control key status

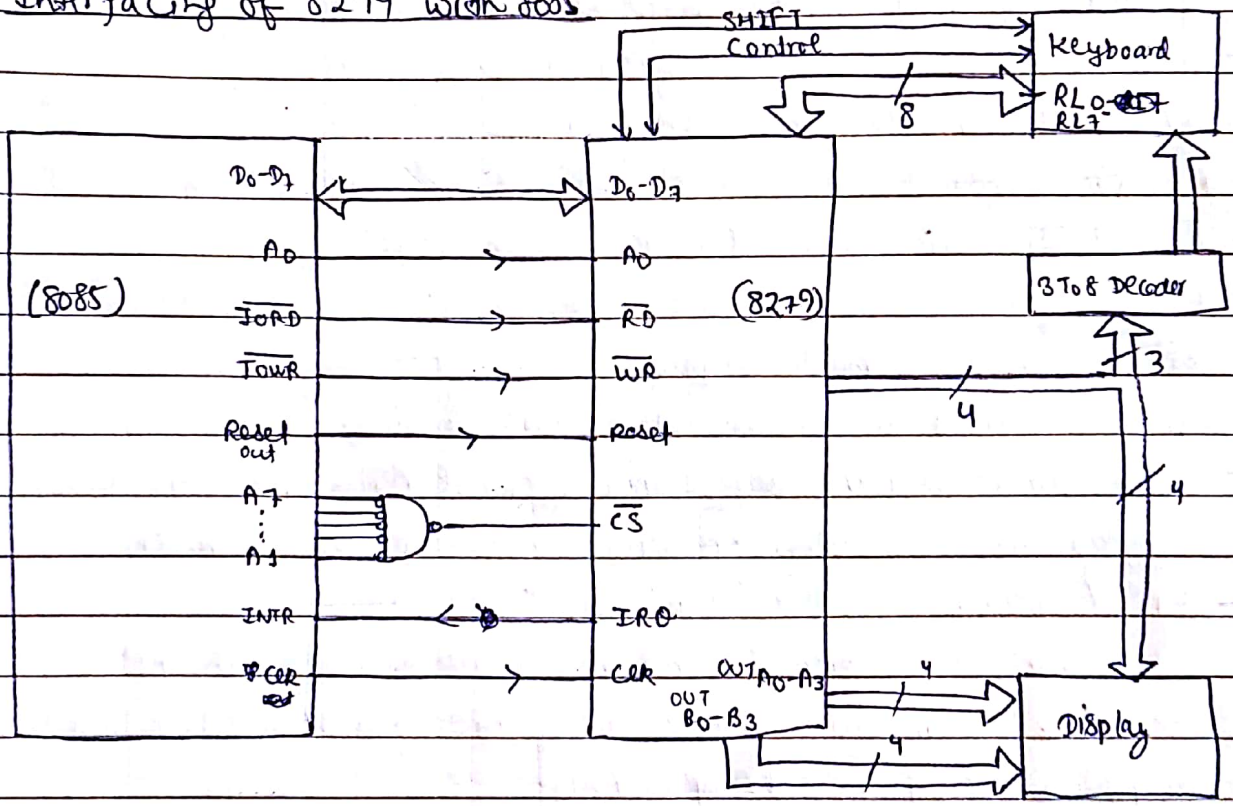
⇒ RL0 - RL7 (Return Lines) → Input lines for keyboard

⇒ SHIFT → Pulled up internally to keep it high until pulled low with closure.

⇒ CNTL/STB-control/Strobbed I/P mode → In keyboard mode, this line used for control input & stored in FIFO in key closure

→ Enters data into FIFO ram in strobed I/P mode

* Interfacing of 8279 with 8085



Universal Synchronous And Asynchronous receiver and Transmitter (USART) - Intel 8251

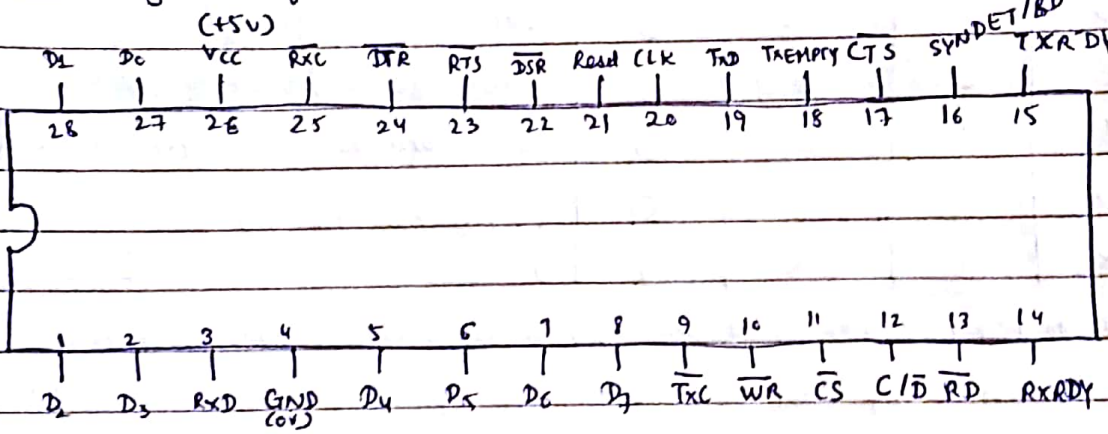
* For data communication, an interface must convert parallel data into serial for transmission & must convert serial data into parallel form after receiving.

* 8251 can transmit as well as receive serial data in synchronous mode as well as asynchronous mode.

* Features of 8251:

① Used in both synchronous as well as asynchronous operations.
② In asynchronous mode, clock rate of 1, 16 or 64 times the baud (bits per second) rate
③ It has automatic break detect & handling facility
④ Three types of error detection is available (parity, overrun & framing)
⑤ It can work in duplex mode (half-duplex & full-duplex).
⑥ It has double buffered data paths with separate I/O registers for control, status, data in data out, which minimizes CPU overhead.

* Pin diagram of 8251 : 28 pin chip



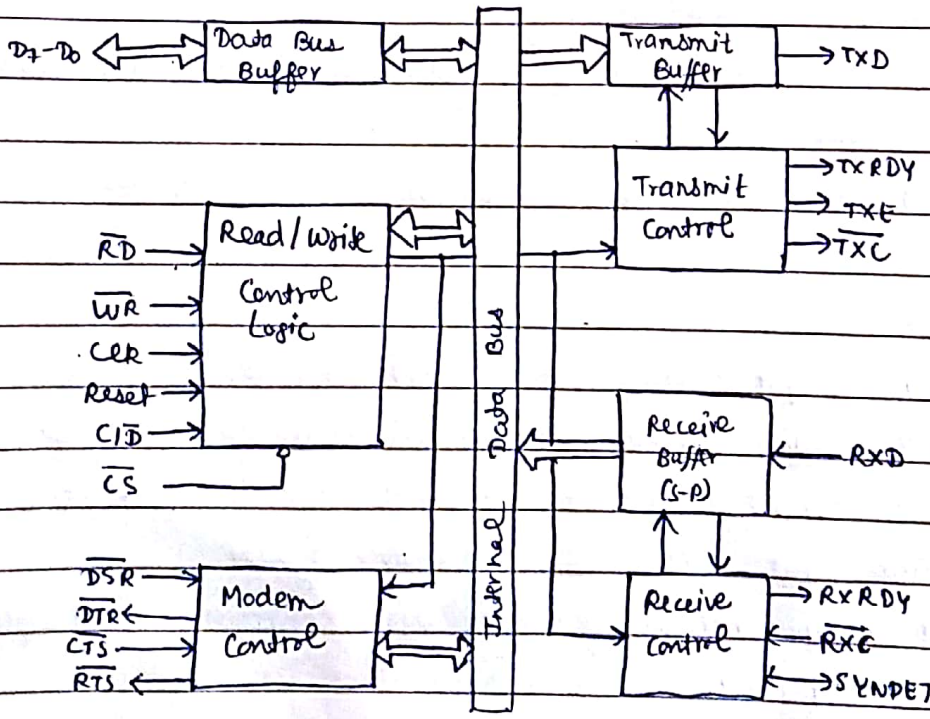
- $D_0 - D_7$ → Data Bus
- $\overline{RD}$ → Inform CPU is reading either data or status information from internal regis
- $\overline{WR}$ → Inform CPU is writing data or control word to 8251
- $\overline{CS}$ (Chip select) → If 1, no read or write operation & databus tristated
- CLK → clock signal
- Reset → forces 8251 into idle state
- $\overline{TXC}$ (Transmitter Clock) → This input controls rate at which character is to be transmitte
- TxD (Transmitter Data) → This output carries serial stream of the transmitted data
- $\overline{RXC}$ (Receiver Clock) → This input controls rate at which character is to be received
- RxD (Receiver Data) → This input receives composite stream of data
- RxRDY (Receiver Ready) → This output indicates 8251 contains a character to be read by CPU
- TxRDY (Transmitter Ready) → This output indicates CPU that 8251 ready to accept character for transmission from CPU
- $\overline{DSR}$ (Data set ready) → Used to check if data set is ready when communicating with modem
- $\overline{RTS}$ (Request to send Data) → Indicate the modem that receiver ready to receive data byte from modem.
- $\overline{CTS}$ (Clear to send) → 8251 enabled to transmit serial data.

- TXE (Transmitter Empty) → Represent that no characters to transmit by transmitter
- SYNDET/BD (Synch Detect / Break Detect) → Used in synchronous mode for detecting synchronous characters & in asynchronous mode for break detect output.
- C/$\overline{D}$ (Control / Data) → Represent word on data bus either Control word ($C/\overline{D} = 1$) or data word ($C/\overline{D} = 0$)

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Operation / Function
0	0	1	0	Data port of 8251 to data bus
0	1	0	0	Data bus to data port of 8251
1	0	1	0	Status word of 8251 to databus
1	1	0	0	Data word to Control register of 8251
x	1	1	0	Data bus tristated
x	x	x	1	Data bus tristated

- $\overline{DTR}$ (Data Terminal Ready) → Used for modem control such as Data Terminal (CPU) ready

* Block Diagram of 8251



ⓐ CPU Interface & Control Logic

- This block accepts $\overline{RD}$, $\overline{WR}$, CLK, C/$\overline{D}$, $\overline{CS}$, D_0-D_7 & Reset Signals

ⓘ Data Bus Buffer

- Tristate buffer used to interface 8251 to system data bus

ⓘⓘ Read/Write Logic Register

- It includes R/W control logic, 6 input signals, 3 buffer registers (data, control & status register). It contains CWR (Control word Register)

(b) Modem Control

- Used to interface to almost any modem using $\overline{DSR}$, $\overline{DTR}$, $\overline{RTS}$, $\overline{CTS}$ signals

(c) Transmitter Section

(i) Transmitter Buffer – Accepts parallel data from data bus buffer & converts it to a serial bit stream

(ii) Transmitter Control – Manages all activities associated with transmission of serial data. It accepts & issues signals both externally & internally to accomplish this function

- Consists of TxRDY, TxE, $\overline{TXC}$

(d) Receiver Section

(i) Receiver Buffer – Accepts serial data, converts this serial input to parallel format & consists of RXD signals

(ii) Receiver Control – Manages all receiver related activities & consists of RxRDY, $\overline{RXC}$, SYNDET/BD signals.

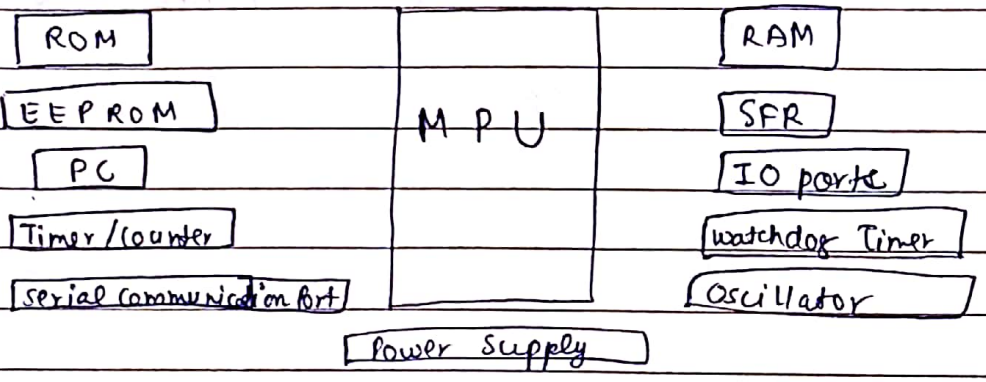
Unit - IV

Microcontroller 8051

* Microcontroller performs all tasks of computer on small scale. It is low cost version of microprocessor.
* on single chip it has MPU (Microprocessor Unit), RAM, ROM, I/O devices

Microprocessor	Microcontroller
(1) Microcomputer is designed around a microprocessor.	* Microcontroller is embedded system based.
(2) Memory, I/O devices are externally connected	* Onchip memory & I/O devices
(3) Slower than microcontroller	* Faster than microprocessor
(4) Higher power consumption	* Lower power consumption
(5) No power saving features	* Power saving features present

* Block Diagram of Microcontroller



- ROM two types
  (i) External ROM ⇒ cheaper & can accomodate large program but less speed & need to connect pins (512 B - 64 KB).
  (ii) onchip ROM
- EEP ROM biggest disadvantage is slow speed
- SFR (Special Function Registers)
  ⇒ Basically part of RAM, predefined special func^n mostly used to control func^n of microcontroller
  ⇒ Ex - PSW, Accumulator, TMOD (Timer Mod Register), TCON (Timer Control Register), DPTR (Data Pointer)
- MPU (Microprocessor Unit)
  ⇒ Monitor & control all processes within the microcontroller

⇒ Consists instruction decoder, ALU, accumulator (like working desk)

- I/O ports

⇒ Connected to I/O devices through pins

- Oscillator

⇒ Provide clock signal to microcontroller

⇒ mostly onchip oscillator

⇒ Basically act as amplifier circuit

- Timer / Counter

- Watchdog Timer

⇒ The microcontrollers consists of another type of timer known as watchdog timer, computer operating properly (COP) timer to make system self reliant

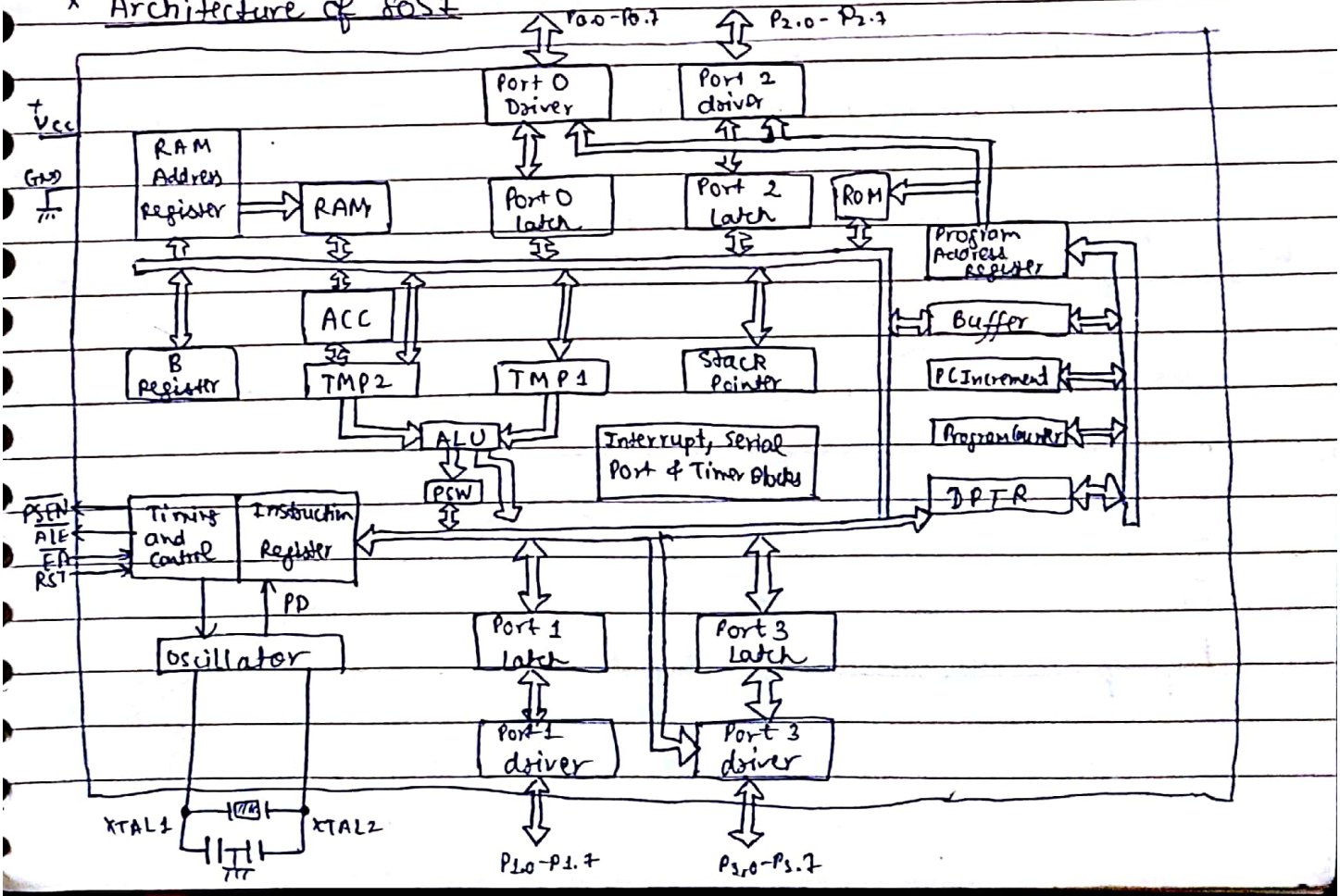
⇒ whenever there is a system failure or software hangs, the watchdog timer automatically detects these anamolies & reset the processor

- Serial Communication Block

⇒ Ideal way of communication for shorter distances (upto several meters)

⇒ Not suitable for long distance communication, parallel communication

* Architecture of 8051



- It consists of:

ⓐ ALU ⓑ Timing & Control Unit ⓒ Instruction Decoder ⓓ 128 byte RAM
ⓔ 22 SFR ⓕ 4K ROM ⓖ Timer 0 ⓗ Timer 1 ⓘ 1 serial port
ⓙ 8bit 4 IO ports ⓚ 1 oscillator

- It can operate at bit level & byte level
- Instruction decoder ⇒ To decode instruction & tell system what operation to be done.
- Timing & Control Unit ⇒ Provides all control & timing signals to perform operation
- Oscillator & Clock ⇒ Generates the clock pulses by which all internal operations are synchronized
- Program Counter ⇒ Points to address of next instruction to be executed
- Data Pointer (DPTR) ⇒ Consists of high byte (DPH) & low byte (DPL) which are used to furnish memory addresses for internal & external code access & external data access.
- Accumulator (A) & B Registers ⇒ Hold results of many instructions
- Flags & PSW (Program Status Word) ⇒ PSW & PCON (Power Control) registers are used to define the flag
- Internal RAM ⇒ Organised in three areas:

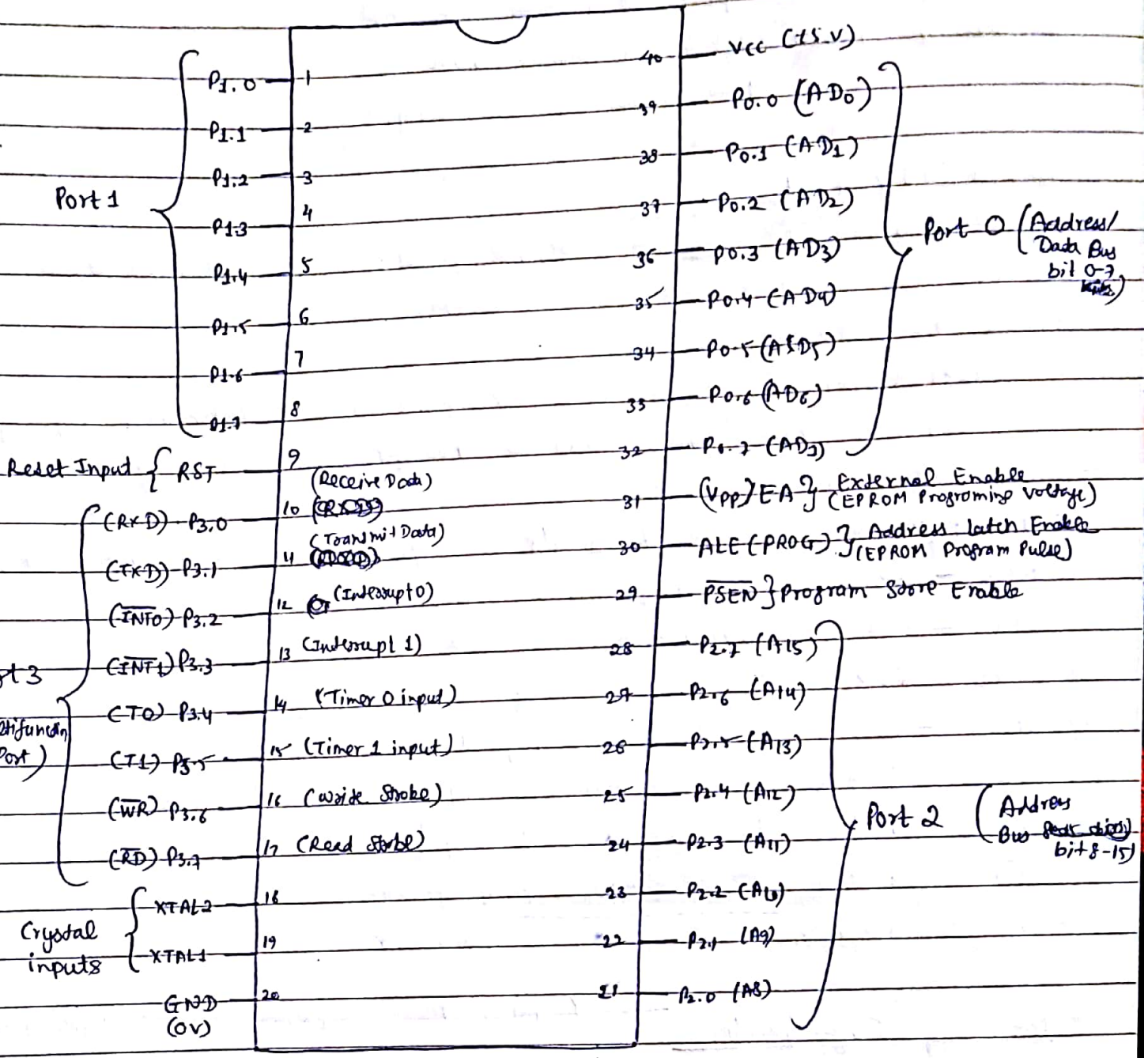
ⓘ Working Registers ⓘⓘ Bit addressable ⓘⓘⓘ General Purpose RAM

- Internal ROM ⇒ Organised in two entities:

① data memory ② Program code memory

$P_{x.y}$ ⇒ Port x bit y

* Pin diagram of 8051 :



- Pin XTAL1 & XTAL2 are provided for connecting a resonant network to form an oscillator

* I/O ports (P0, P1, P2, P3) (Also has explanation of pin diagram pins)

- 8-bit 4 ports that can work in IO config & can be enabled/ disabled
- used to attach extra peripheral
- Bidirectional ports

- Port 0 (P0.1 - P0.7)
=> Multi-functional port & provide multiplexed address data bus
=> Input, output & multi/latch bidirectional mode
1 → Input 0 → output

- Port 1 (P1.0 - P1.7)
=> Bidirectional I/O port, with internal pullups
1 → Input 0 → output

- Port 2 (P2.0 - P2.7)
=> Multi-functional bidirectional I/O port with internal pullups & provides high order address bus for external memory

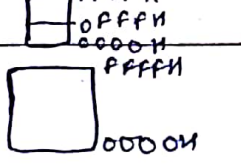
- Port 3 (P3.0 - P3.7)
=> Multi-functional port, I/P & O/P & different functions such as $\overline{WR}$, $\overline{RD}$ etc.

① RXD - P3.0 ⇒ Receive Serial Data
② TXD - P3.1 ⇒ Transmit Serial Data
③ INT0 - P3.2 ⇒ External interrupt 0
④ INT1 - P3.3 ⇒ External interrupt 1
⑤ T0 - P3.4 ⇒ External Timer 0 Input
⑥ T1 - P3.5 ⇒ External Timer 1 Input
⑦ $\overline{WR}$ - P3.6 ⇒ External memory write
⑧ $\overline{RD}$ - P3.7 ⇒ External memory read

* Memory Organisation (Also have explanation of pin diagram pins)

- Two types on chip memory
⇒ 4KB program memory / program code memory (ROM)
⇒ 128 byte data memory (RAM)

- Program Memory
⇒ 4KB ROM is available for store the program
⇒ 16 bit Program Counter (PC) as a default pointer for program memory
⇒ ROM ranges 0000H → 0FFFH
⇒ Can be extended upto 64 KB by two options: $AD_0 - AD_7$, $A8 - A15$ ⇒ 0000H to FFFFH
① 4 KB onchip & & adding extra 60 KB
② 64 KB external chip used instead of 4KB onchip



⇒ $\overline{EA}$ pin - To access external memory
If 0, then discard 4KB onchip ROM (② option for extending)
If 1, then use 4KB onchip ROM (① option for extending)

⇒ ALE pin - Address Latch Enable

⇒ PSEN pin - Used in case of external ROM, read control signal of offchip ROM
If 0, interface with output enable pin of external ROM

- Data Memory

⇒ Store temporary data & intermediate results

⇒ 128 byte on chip RAM
00H → 7FH

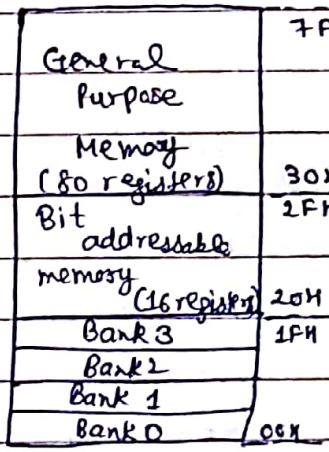
⇒ Divided into different blocks:

(i) 32 working registers organized as four banks of 8 registers each.
Each bank of 8 register called memory bank
To select Bank, RS0 & RS1 pin present in PSW

00-07H } Bank 0 → R0-R7
08-0FH } Bank 1 → R0-R7
10-17H } Bank 2 → R0-R7
18-1FH } Bank 3 → R0-R7
} 00 to 1FH

(ii) Bit addressable memory } 20H to 2FH
Each bit has its own address

(iii) General Purpose RAM } 30H to 7FH }



⇒ To communicate with external pin,
port 1 P1.0 - P1.7 &
port 2 P2.0 - P2.7

* Special Function Registers

- Minimum 21 8bit registers
- Located within internal memory
- 80H → FFH range
- Some bit addressable (INTR, Timer (TCON) Control Reg, Serial (control) SCON register) & other byte addressable registers

NOTE: Program counter (PC) is not a part of SFR & has no internal RAM address.

① Accumulator (A) & B register
- → General purpose data register to store data & intermediate results during a program
- → Accumulator is compulsory register
- → Register B is for multiplication & division otherwise not generally used.
- → Both bit & byte addressable
- → Accumulator → E0 H RAM location
  Register B → F0 H RAM location

② Stack Pointer Register (SP)
- → Temporary data storage
- → Address storage if interrupt or call instruction executed
- → RAM location → - 81 H
- → SP denoted top of stack
- → Generally initialized / reset value 07 H or 00 H.

③ Data pointer register (DPTR)
- → 16 bit register which physically don't exist.
- → DPH & DPL registers combined to form DPTR
- → DPL RAM location - 82 H
  DPH RAM location - 83 H
- → Internal or external memory to access code or data from memory

④ PSW Register (Program Status Word)
- → 8 bit flag register
- → Contains 7 flag bits (CF, AC, OF, PF, RS1, RS0, F0)
- → 1 user flags F0 at bit which may or may not be used.
- → Byte as well as bit addressable
- → RAM location - D0 H
- → Represent data condition of current result

D7	D6	D5	D4	D3	D2	D1	D0
CF	AC	F0	RS1	RS0	OF	X	PF

⑤ I/O port Registers

For port 0 latch ram location 80 H
For port 1 latch ram location 90 H
For port 2 latch ram location A0 H
For port 3 latch ram location B0 H

⑥ Counters & Timers

→ To generate accurate time delays
→ Two 16bit counter registers (T0 & T1)

Timer 0 & Timer 1
= TH0 + TL0 = TH1 + TL1
(TH0 → High byte, TL0 → low byte)

→ RAM location of TL0 is 8A H
RAM location of TL1 is 8B H
RAM location of TH0 is 8C H
RAM location of TH1 is 8D H

→ 4 timer mode:
① Mode 0 - 8 bit timer/counter
② Mode 1 - 16 bit timer/Counter
③ Mode 2 - 8 bit autoreload timer/counter
④ Mode 3 - timer 0 → 8 bit & timer 1 → stop

⑦ TMOD register (Timer/Counter Mode)

→ To select mode of timer
→ RAM location - 89 H

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Gate1	C/T_1	T_1M_1	T_1M_0	Gate0	C/T_0	T_0M_1	T_0M_0
Timer 1				Timer 0			

x → 1/0

Gate$_x$ ⇒ To enable/disable timer 1 & timer 0 through INT1 & INT0 pin

C/T_x ⇒ count no. of pulses for timer 0 & timer 1

T_xM_1, T_xM_0 ⇒ Decide the operating modes of two counters

00 = Mode 0
01 = Mode 1
10 = Mode 2
11 = Mode 3

⑧ TCON Register (Timer/Counter Control)

→ control timer & counter register operations

→ Ram location – 88 H

x → 1/0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
TF_1	TR_1	TF_0	TR_0	IE_1	IT_1	IE_0	IT_0
Timer 1		Timer 0		Timer 1		Timer 0	

TF_x ⇒ (Timer overflow flag) Set to 1 when timer rolls from all 1's to 0.

TR_x ⇒ (Timer Run Control Bit) set to 1 to enable timer to count

IE_x ⇒ (External Interrupt Edge Flag) Set to 1 when a high-low edge signal is received [on port 3.3 (for IE_1) & port 3.2 (for IE_0)]

IT_x ⇒ (External Interrupt Signal type control bit) Set to 1 to enable external interrupt to be triggerred by a falling edge signal, otherwise remain level triggered

⑨ SCON (Serial Control) Register

→ Used to set & control serial mode communication

→ Ram location – 98 H

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SM_0	SM_1	SM_2	REN	TB_8	RB_8	TI	RI

SM_0, SM_1 ⇒ serial mode select

00 = mode 0
01 = mode 1
10 = Mode 2
11 = Mode 3

SM_2 ⇒ Used for enabling microprocessor communication in mode 2 & mode 3 by setting to 1 (Multiprocessors)

RI ⇒ (Receive Interrupt Flag) Automatically set to 1 by hardware after receiving 8 bits

TI ⇒ (Transmit Interrupt Flag) Automatically set to 1 after last bit

TB_8 ⇒ (Transmitter Bit 8) • Used in mode 2 & mode 3 as the 9th bit transmitted

RB_8 ⇒ (Receiver Bit 8) Used in mode 2 & mode 3 & the 9th bit to be received

Stop bit in mode 1

REN (Reception Enable Bit) ⇒ Enables serial reception when set to 1

⑩ Interrupt Enable (IE) Register (Control)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
EA	X	X	ES	ET_1	EX_1	ET_0	EX_0

Reserved for future use

EA → Enable All
If 1, enable interrupt individually
If 0, no interrupt will be acknowledged

ES ⇒ Enable Serial Interrupt
ET_1, ET_0 ⇒ Enable Timer Interrupts
EX_1, EX_0 ⇒ Enable External Interrupts

B → Bit as well as byte addressable
→ Ram location - A8 H

⑪ Interrupt Priority (IP) Register
→ RAM location - B8 H
→ low priority interrupt only interrupted by high priority interrupt
→ higher priority interrupt served if two interrupts of different priority
→ If same priority interrupts simultaneously, then internal polling sequence
Default priority : INT0, IF0 (Timer 0), INT1, IF1 (Timer 1), SI → serial interrupt
High ⟶ low

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
X	X	PT_2	PS	PT_1	PX_1	PT_0	PX_0

} If any one of these bit is 1 then high else low priority

PS → Defines serial port interrupt priority
PT_y → Defines timer y priority (y → 1 / 0)
PX_y → Define external interrupt INTy priority

* 8051 Interrupts

- Provides 5 types of interrupts :
  - 2 external interrupts (INT0 & INT1)
  - 3 internal interrupts (INT0, INT1 & Serial communication)
- There are no software interrupt
- Serial port interrupt handled through SCON
- Timer 1, Timer 0 interrupt handled through TCON
- TCON, SCON, IE & IP control these interrupts
- INT0 & INT1 triggering handled by TCON

	Interrupt	Controlling Bit	Vector location
1)	INT0	IE0, IT0	0003H
2)	Timer 0	TF0, TR0	000BH
3)	INT1	IE1, IT1	0013H
4)	Timer 1	TF1, TR1	001BH
5)	Serial port	RI, TI	0023H

* 8051 Addressing Modes:

- Addressing modes are the various ways of specifying an operand in an instⁿ
- There are six addressing modes in 8051:

① Immediate Addressing mode:

The operand is stated in the instⁿ itself & it comes immediately after opcode used when source operand is constant.

Ex:- ⓐ MOV A, #12H
↳ sign to represent Data
Load 12H data to accumulator

ⓑ MOV R3, #35 } Load decimal data 35 to a R3 (Register 3)

ⓒ MOV DPTR, #2500H } Load 16 bit data 2500H to DPTR reg.

ⓓ MOV P1, #FFH } Load FFH data to port 1

② Register Addressing Mode:

we specify a register in the instⁿ where operand is present

Ex:- ⓐ MOV R3, A } Move contents accumulator to R3

ⓑ ADD A, R6 } Add contents of R6 to accumulator

ⓒ DAA } Decimal adjust accumulator

ⓓ RAR } Rotate accumulator right

③ Direct Addressing Mode:

Generally used memory or I/O, we specify of the memory or an I/O where the operand is present or required

Ex:- ⓐ MOV A, 35H } Copy contents of internal RAM 35H to accumulator

ⓑ MOV A, 80H } Copy data from port 0 to accumulator

ⓒ MOV 40H, R0 } Copy contents of R0 to 40H RAM location

④ Indirect Addressing Mode:

we specify a register which contains address of the memory

location where the operand is present

Ex:- ⓐ MOV A, @R1 → represent indirect addressing
Move content of location whose address is specified by R1 to accumulator

ⓑ MOV @R1, A ; Move content of accumulator to memory location address stored in R1

⑤ Index Addressing Mode

This type of addressing is used for reading look up tables from program memory & for branching instruction. In this mode, only program memory can be accessed & can be read only.

Ex:- MOVC A, @A + DPTR
Move contents of external memory, whose address is generated by adding contents of accumulator & DPTR, to accumulator

ⓒ Register Specific Addressing Mode : Used for call & jump instructions
Ex- CALL

- Relative
- Absolute
- Long

ⓐ Relative addressing mode → Max 8 bit offset
ⓑ Absolute addressing mode → Max 11 bit offset
ⓒ Long addressing mode → Max 16 bit offset

* 8051 Instruction Set

(A) Data Transfer Instruction

(1) MOV dest, source

Ex:- MOV R5, #75H
MOV R1, A
MOV DPTR, #2500H
MOV 89H, R1
TMOD
MOV 65H, #45H
MOV A, @R1

(2) MOVX dest, source ⇒ Used to copy from external ram location into "accumulator" ⇒ Work with Accumulator (A) only

Ex:- MOVX @DPTR, A
MOVX A, @DPTR
MOVX @R0, A
MOVX A, @R0

(3) MOVC dest, src → used to copy data from the code or program memory into accumulator

Ex:- MOV DPTR, #1250H
MOV A, #50H
MOVC A, @A+DPTR

(4) PUSH direct → push contents of specified RAM on top of stack
Ex:- PUSH 55H

(5) POP direct → pop top of stack to specified RAM location
Ex:- POP 45H

(6) XCH A, operand ⇒ Swap contents of accumulator with the content of operand mentioned in instⁿ

Ex:- XCH A, R_N → 0,1,2,3,4,5,6,7
XCH A, @R
XCH A, direct

(7) XCHD A, @R ⇒ Special case of exchange instⁿ in which lower nibbles of contents of accumulator & contents of RAM location specified indirectly in instⁿ are exchanged

Ex:- MOV R1, #45H
XCHD A, @R1

(B) Arithmetic Instructions

① ADD dest, src

Ex:- ADD A, #35H
ADD A, R3
ADD A, 25H
ADD A, @R0

② ADDC dest, src ⇒ add with carry flag

Ex:- ADDC A, #35H : [A] ← [A] + 35H + [CF]
ADDC A, R3 : [A] ← [A] + [R3] + [CF]
ADDC A, 25H
ADDC A, @R0

③ DAA => Decimal Adjust After Addition convert the binary sum of two BCD nos. into a BCD no.

④ SUB dest, src ⇒ Subtract src from dest

Ex:- SUB A, #35H | SUB A, 25H

SUB A, R3 | SUB A, @R0

⑤ SUBB dest, src ⇒ Subtract (src + carry flag) from dest

Ex:- SUBB A, #25D : [A] ← [A] − ((25D + [CF])

SUBB A, R0

SUB B A, 29H

⑥ INC operand ⇒ Increment operand by 1

Ex:- INC R1 | INC A | INC @R0 | INC DPTR

⑦ DEC operand ⇒ Decrement operand by 1

Ex:- DEC R1 | DEC A | DEC @R0 | DEC DPTR

⑧ MUL A, B ⇒ Multiply A & B, i.e., A * B

⑨ DIV A, B ⇒ Divide A & B, i.e., A / B

(C) Logical Instructions

① ANL dest, src ⇒ AND operation b/w dest & src, store result in dest

Ex:- ANL A, #77H | ANL 50H, #50H

ANL A, @R1 | ANL C, 20H

② ORL dest, src ⇒ OR operation b/w dest & src, store result in dest

Ex:- ORL A, #77H | ORL 50H, #25H

ORL A, @R1 | ORL B, 20H

③ XRL dest, src ⇒ XOR operation b/w dest & src, store result in dest

Ex:- XRL A, #78H | XRL 30H, #45H

XRL A, @R2 | XRL C, 80H

④ CLR operand ⇒ Clear contents of operand

Ex:- CLR A | CLR C | CLR 29H

⑤ CPL operand ⇒ Complement contents of operands

Ex:- CPL A | CPL C | CPL 29H

⑥ SETB op ⇒ Set operand bit

⑦ RL A ⇒ Rotate left without carry

⑧ RR A ⇒ Rotate right without carry

⑨ RLC A ⇒ Rotate left with carry

⑩ RRC A ⇒ Rotate right with carry

(11) SWAP A ⇒ Swap two nibbles of the accumulator

$(D_0-D_3) \xleftrightarrow[\text{Exchange}]{} (D_4-D_7)$

(D) Program Branching Instⁿ

① JMP
- Unconditional
- Conditional

② CALL
- Unconditional
- Conditional

③ RETURN
- Unconditional
- Conditional

Unconditional Jump

(i) LJMP (Long Jump) :- Can jump from 0000H to FFFFH

Ex:- LJMP MEM ⇒ Jump to label 'MEM'

LJMP 2500H

LJMP @A+DPTR

(ii) SJMP (Short Jump) :- 8 bit address

(iii) AJMP (Absolute Jump) :- within 2KB memory can jump

AJMP @A+DPTR

Conditional Jump

JC → Jump if carry JNZ → Jump if not zero.

JE → Jump if equal JNE → Jump if not equal etc.

CJNE ⇒ Compare if jump not equal.

DJNZ Rn ⇒ Decrement content of specified reg. by 1 & jump if result is not zero

Unconditional Call

ACALL address (Absolute Call) :- Call subroutine within 2KB memory space

LCALL address (Long Call) :- Call subroutine within 64KB memory space

Conditional CALL → Similar to Conditional jump like CNZ, CC etc.

RET (Return)

RETI (Return from Interrupt Service Routine)