Chapter - ③

Instruction cycle. → It is defined as the time require to Complete the execution of an Instruction.

An Instruction is a Command which is Given to Computer to perform a particular task. To perform a particular task, a programmer has to write a sequence of instruction, Called program.

program & data are stored in memory the CPU fetches one instruction from memory at a time

The Necessary steps that a CPU Carries out to fetch an instruction & Necessary data from memory & to execute it Called instruction cycle.

Hence an Instruction cycle Consist of a fetch cycle & execute cycle.

In fetch cycle, ~~the~~ the, CPU fetches op-code from the memory.

The Necessary steps which are Carried out to get data from memory & to perform the specified operation Called an execute cycle.

Hence the total time required to execute an Instruction is–

$$\boxed{IC = EC + FC}$$

Fetch operation → The 1st byte of an Instruction is op-code. , The rest of the bytes are data or operand address.

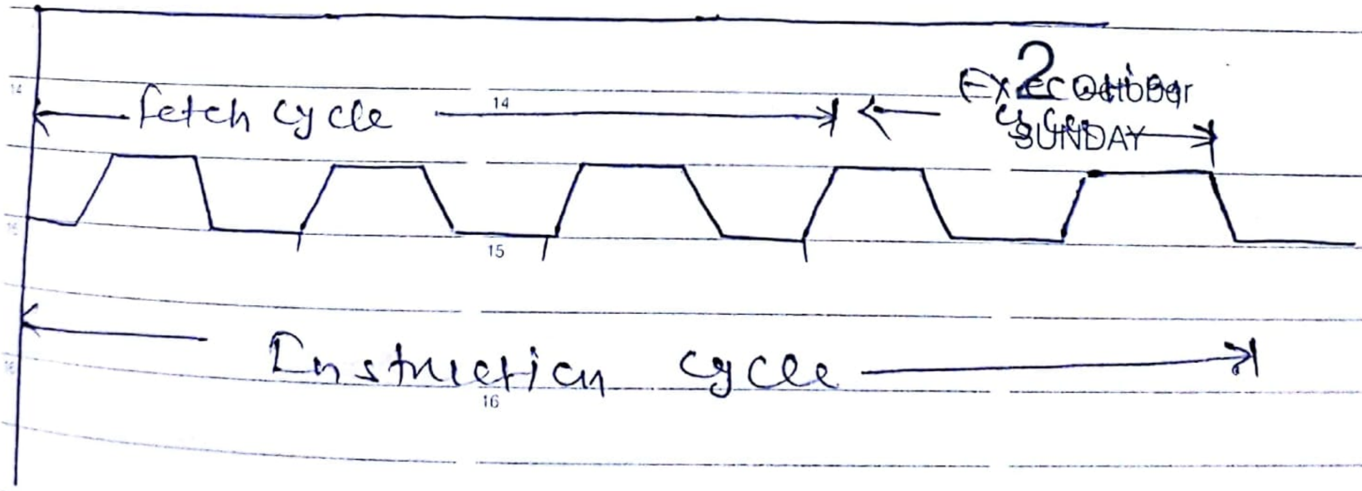
In the begeining of the fetch cycle the content of the program counter, which is the address of location where the Opcode is stored is sent to memory.

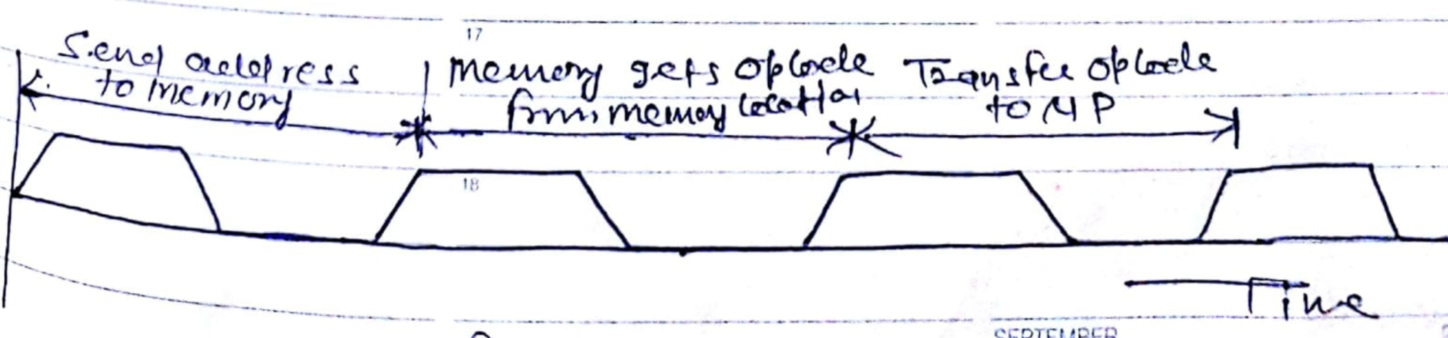
The memory places the op-code on the data Bus so as to transfer it to CPU. The entire operation of fetching takes 3 ~~m/c cycle~~ clock pulses.

Execute cycle → The op-Code fetched from the memory Goes to Data Register & then to Instruction Register & then to decoder circuitary. After the Instruction is decoded, the execution begains.

If the operand is in the General purpo Register, then Execution is immediately performed. If an operand contains data or operand address which is in memory, then CPU has to perform some read operation to get data. After receiving the data it performs execution.

* A read cycle is simillar to fetch cyc In case of read cycle, the quantity received from the memory is data or opera address while in fetch cycle it is opcode.





fetch cycle

Ex:

Draw the timing Diagram for MVI A, 2FH.

Soln The No. of bytes require to store the instruction - 2 bytes.

Hence Instruction will contain 2 M/C cycle one for Opcode fetch & one for memory read operation

Since there is No operation related to memory or I/O. Hence No additional execution cycle require.

∴ Instruction cycle = Read cycle + Execution cycle

↓

(Opcode fetch + memory read) + None

↓

4T + 3T

= 7T

Let the instruction is stored in memory as shown.

Memory Address	Hex Code	Instruction	
		Opcode	Operand
7000H	3E	MVI	A, 2FH
7001H	2F		2F

10 October
MONDAY

Priorities

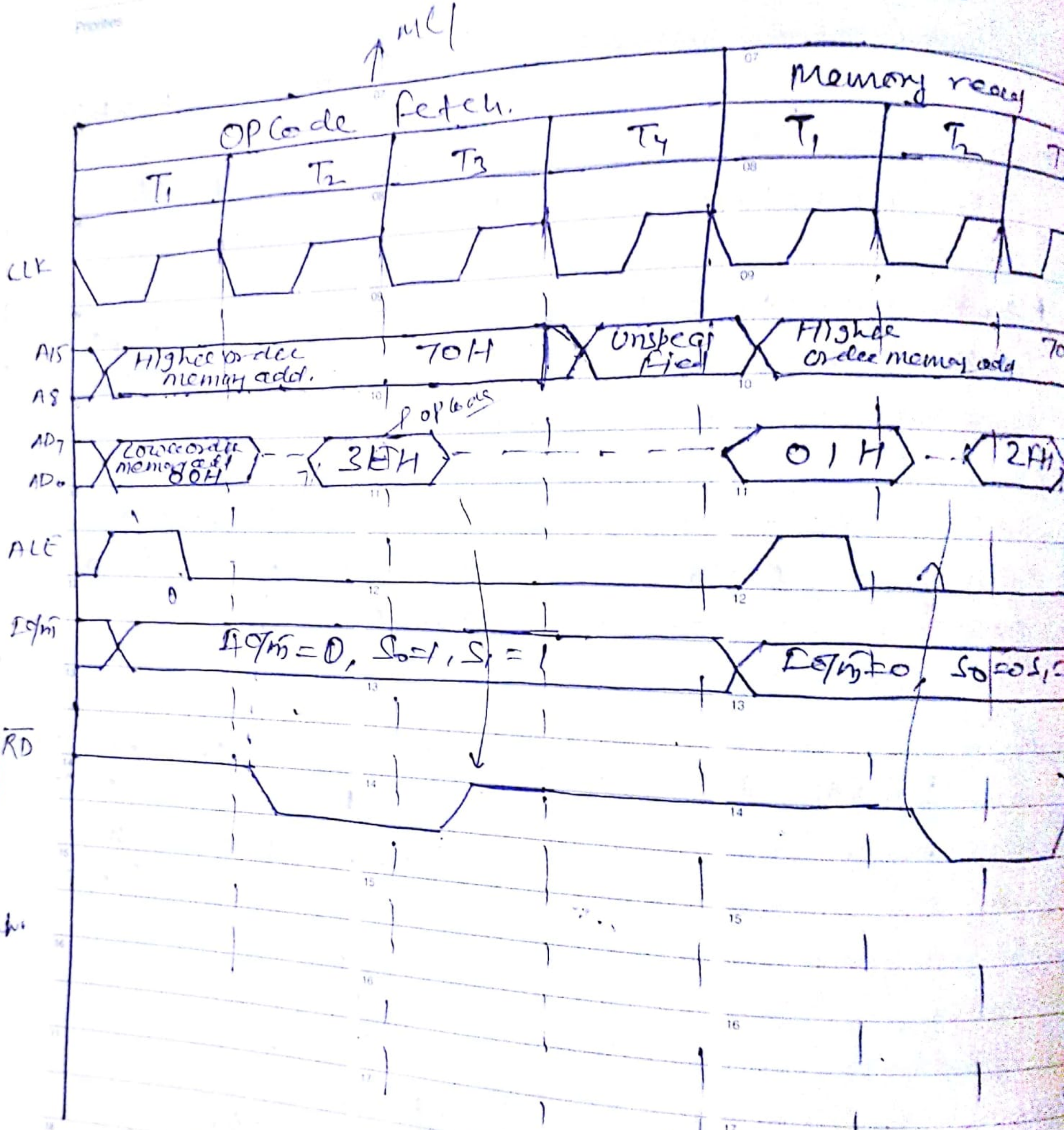
Week 41
283-82

11 October
TUESDAY

284-81

12 October
WEDNESDAY

MCl



machine cycle →
machine cycle is the time required to Complete one operation. It may be memory read. , memory write, I/o Read, I/o write.

in 8085 micro processor there are 9 diff. machine cycle.

machine cycle	Status signal		
	IO/$\overline{M}$	S_1	S_0
1 Opcode fetch.	0	1	1
2 Memory read	0	1	0
3 Memory write	0	0	1
4 I/o Read	1	1	0
5 I/o write	1	0	1
6 Interrupt 8 Acknowledge	1	1	1
7 Halt	Z	0	0
8 Hold	Z	X	X
9 Reset	Z	X	X

T-State → T-State in defined as one subdivision of operation performed in one clock period.

17 October
MONDAY

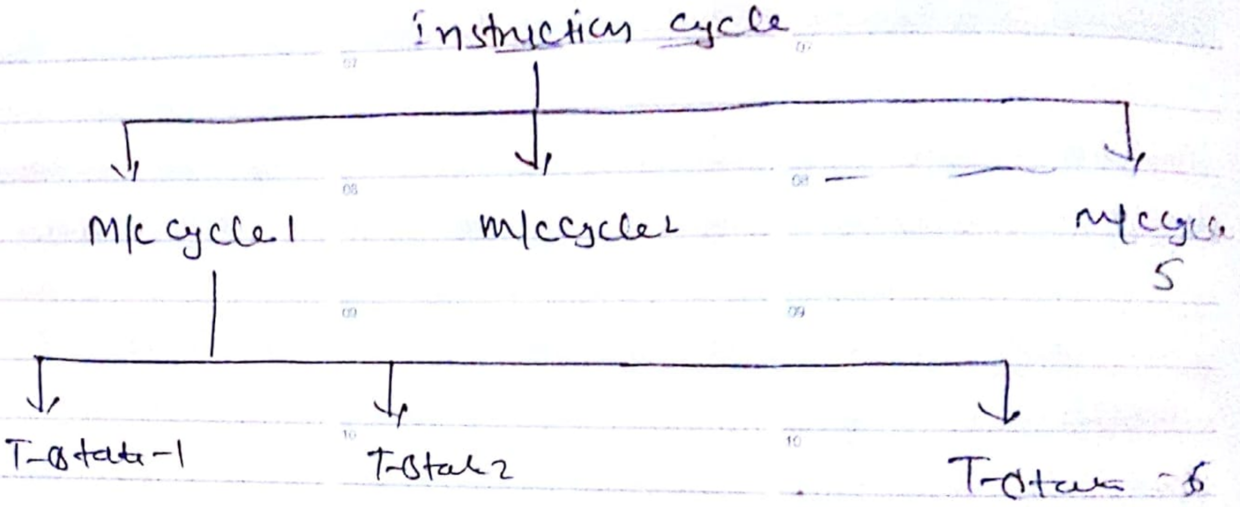
Week 42
290·75

18 October
TUESDAY

291·74

19 October
WEDNESDAY

Priorities



EVENING

INTERRUPT → The microprocessor In General executes tee instruction in a sequencial manner but Some times It is needed to break the sequence of program.

Hence "Interrupt is defined as a signal which is Generated by peripheral devices to break tee sequence of main program & program Jumps to the pre-specified memory location".

A subroutine is written at that memory location to Complete tee task provided by tee peripheral. This subroutine is called "Interrupt service routine".

23 October SUNDAY

If more than one peripherals are Connected to tee microprocessor, ~~then~~ then any one of them may demand for tee ~~f~~ service.

There are two method for providing tee service

① Polling check → In this method, micro processor Continuously check all the peripherals one by one. If any peripheral wants the service it will provide other wise it will check the next peripheral

(2) Interrupt Check → In this method the peripheral which wants the service Generates the signal to break the sequence of main program. In response to this signal microprocessor provide the service to that peripheral.

Interrupt Analogy →

Assuming that You are reading a novel on Your desk. where there is a Telephone for You to receive a Response to the Tele phone following steps should occur-

(i) The telephone system should enabled.

(2) You should Glance at the light of Telephone at certain interval to varify that wheather some one is calling

(3) If You see a blinking light, You should

pick the receiver, Say Hello, & wait for response. once you pick the phone, the line is busy, & No more Calls Can be received Untill you replace the receiver

(4) Asuming that Caller is your roommate, & the request to shut all the windows of His room.

(5) You insert a book mark on ~~book~~ the page you are reading.

(6) You replace the receiver on the hook.

(7) You shut your roommate's window.

(8) You Go back to your book, find your mark & start reading again

→

(1) The microprocessor is ~~bsy~~ busy to executing the main program & Interrupt ~~Enable~~ flip-flop (EI) is enable.. Hence all peripherals are allowed to request the service

(2) After executing each Machine Instruction cycle, & The micro processor Check the INTR PIN.

(3) peripherals Generates the Interrupt signal & send to the INTR pin of microprocessor when it wants to service the microprocessor.

(4) The micro processor Complete the main Current instruction, save the address of next instruction of main program & send ACKnowledge signal ($\overline{INTA}$) on the $\overline{INTA}$ pin.

(5) After $\overline{INTA}$ signal, No other Interrupt requested is accepted. The microprocessor provide the service to the peripheral.

(6) while providing the service to the ~~Interrupt~~ peripheral, The Micro processor may or may not ~~be~~ respond to other Interrupt request from other peripherals, but there is one Interrupt (TRAP), to whom micro processor has to respond.

After providing the service to the peripharal micro processer enables all interrupts using software Instruction.

(7) The micro processor returns to the main programs ~~after~~ which in previously stored.

~~Typ~~ Classification of Interrupts.

(i) Hard ware Interrupts
(ii) soft ware Interrupts

~~(i) Soft ware Interrupt :~~

(i) Hard ware Interrupt. —

The types of Interrupts, when micro processor ~~receive~~ pins are used to receive Interrupts are called Hard ware Interrupts.

The micro processor has 5 pins for receiving external interrupt signal. These pins are TRAP, RST 7.5, RST 6.5, RST 5.5, INTR.

NOVEMBER 2011

W	M	T	W	T	F	S	S
44		1	2	3	4	5	6
45	7	8	9	10	11	12	13
46	14	15	16	17	18	19	20
47	21	22	23	24	25	26	27

Hard ware Interrupts — Maskable & Non-maskable
— Vectored & Non-vectored

(i) vectored & Non-vectored Interrupts →

→ when a peripharal Interrupts the micro processor, the Micro processor break the sequence of main program & program Jump to the specified memory address. This specified memory address is the starting address of ISR. These Interrupts are Called vectored Interrupts.

→ Thes interrupts for which the vectored location in Not pre defined or It is ~~defined~~ determined by some additional external means are Called Non-vectored Interrupts.

~~Interrupt~~ Type ~~of vector~~ Vector Location

Interrupt	Type of vector	Vector location	Subroutine location.
TRAP	vectored	0024H	–
RST 5.5	"	002CH	–
RST 6.5	"	0034H	–
RST 7.5	"	003CH	–
INTR	Non-vectored.	—	0000H 0008H 0010H 0018H 0020H 0028H 0030H 0038H

The subroutine location for INTR in determined by externally connected Hard ware & this can be any one of eight location

→ Maskable & Non-maskable Interrupts →

Software Interrupts → The interrupt Caused by special interrupt instruction is called software interrupt.

After executing these instruction, the microprocessor Complete the execution of instruction it is currenty executing & then Transfer the Controll to subroutine program.

In 8085 there are 8 RST (Restart) instructions. when ever RST instruction is executed, The Address stored in PC is Now stored In the STACK before the program Controll is Transfered to the subroutine.

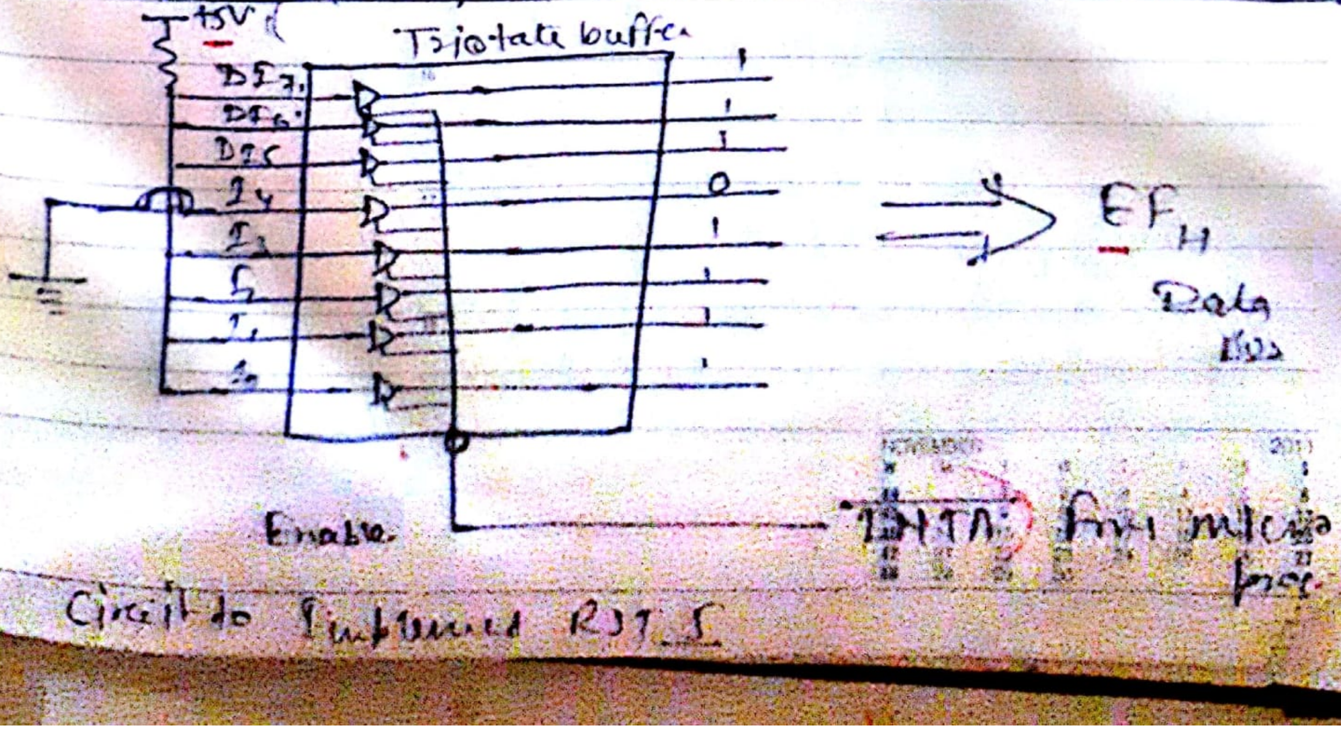
Now after executing the RET instruction the program Controll return back to the main program.

The Difference bet^n CALL & RST instruction —

(i) ~~Be~~ for CALL instruction, we have to specify the address. whereas for RST instruction the location are pre defined.

(2) The execution of CALL Instruction is specified by the programmer where execution of RST Instruction are decided by peripharal by inserting the hex code into the data BUS.

Instruction	Binary Code								Hex Code	Call location
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
RST 0	1	1	0	0	0	1	1	1	C7	0000H
RST 1	1	1	0	0	1	1	1	1	CF	0008H
RST 2	1	1	0	1	0	1	1	1	D7	0010H
RST 3	1	1	0	1	1	1	1	1	DF	0018H
RST 4	1	1	1	0	0	1	[illegible]	1	E7	0020H
RST 5	1	1	1	0	1	1	1	1	EF	0028H
RST 6	1	1	1	1	0	1	1	1	F7	0030H
RST 7	1	1	1	1	1	1	1	1	FF	0038H



Circuit to Implement RST 5

The address of the program Counter is stored in the stack before the program Controll is Transfer to the RST Call location when the processor returns to the Return encounters Instruction in the Subroutine associated with the RST Instruction, the program returns to the address that was stored in the stack.

The Signal INTR is used to insert any 8 Restart Instruction

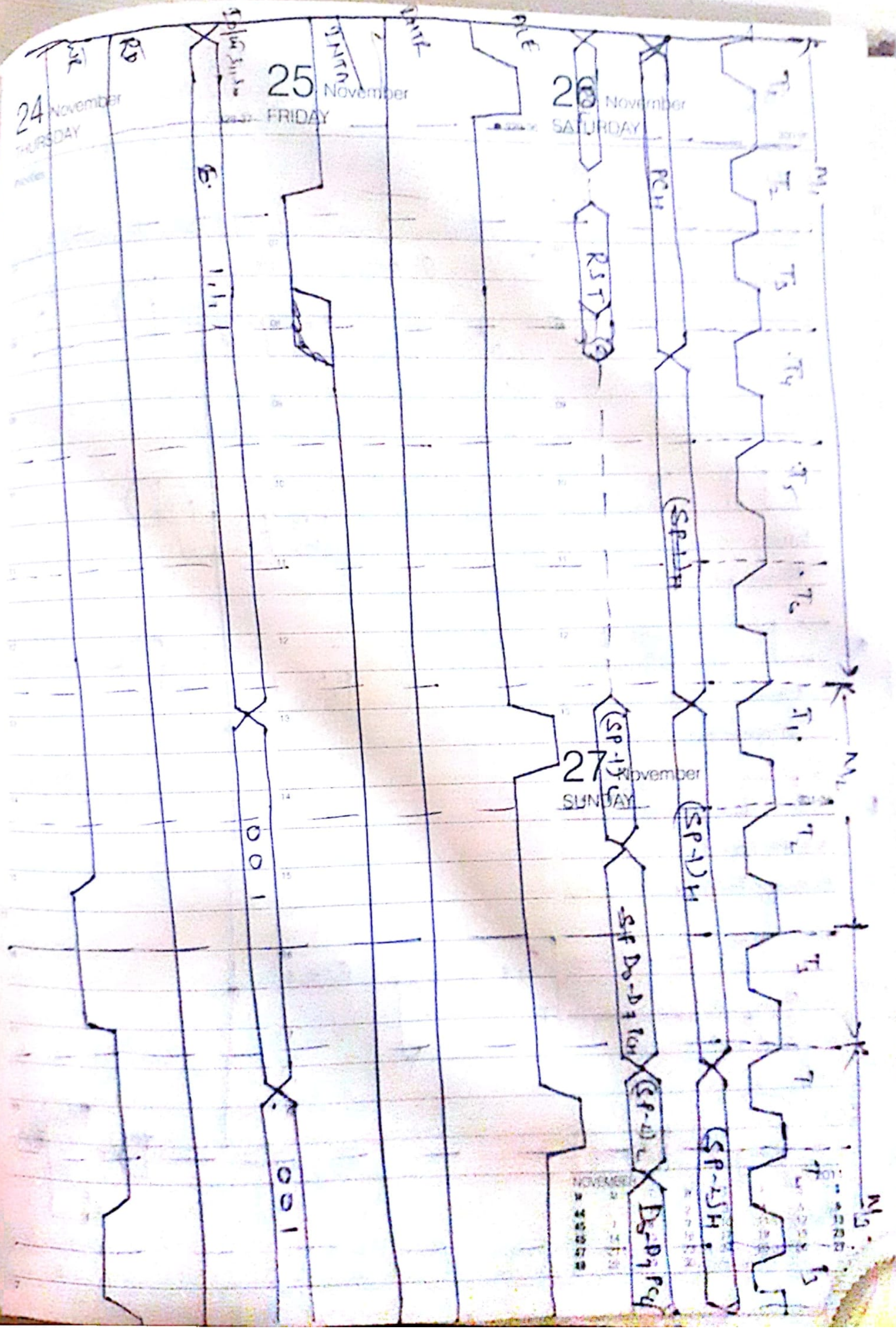
for ex. RST 5 is built using Tristate buffer. In Response to the INTR signal, the 8085 sends $\overline{INTA}$ low signal which is used to enable the buffer & RST instruction is placed on the data BUS.

24 November
THURSDAY

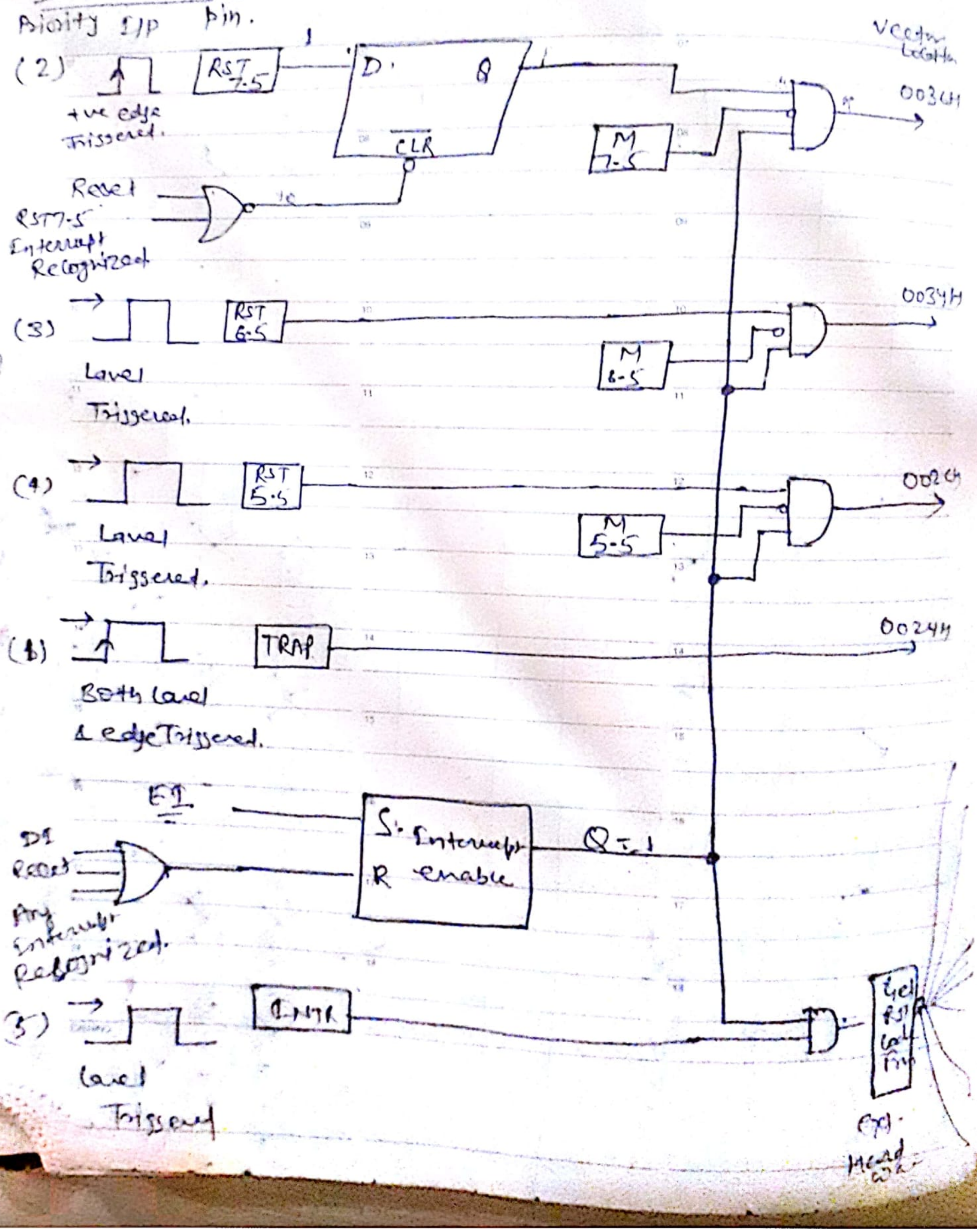
25 November
FRIDAY

26 November
SATURDAY

27 November
SUNDAY



INTERRUPT Structure



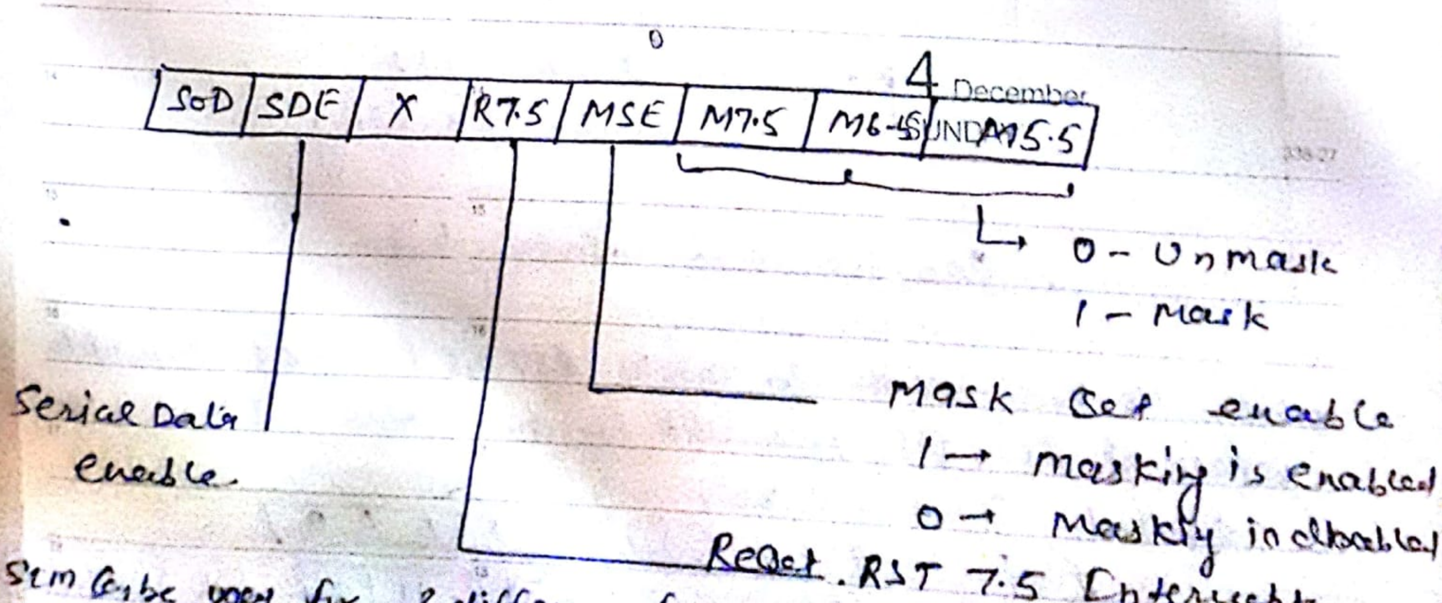
Priorities

Interrupt Related Instructions.

① EI → Enable Interrupt
1- byte Instruction.
No flags affected.

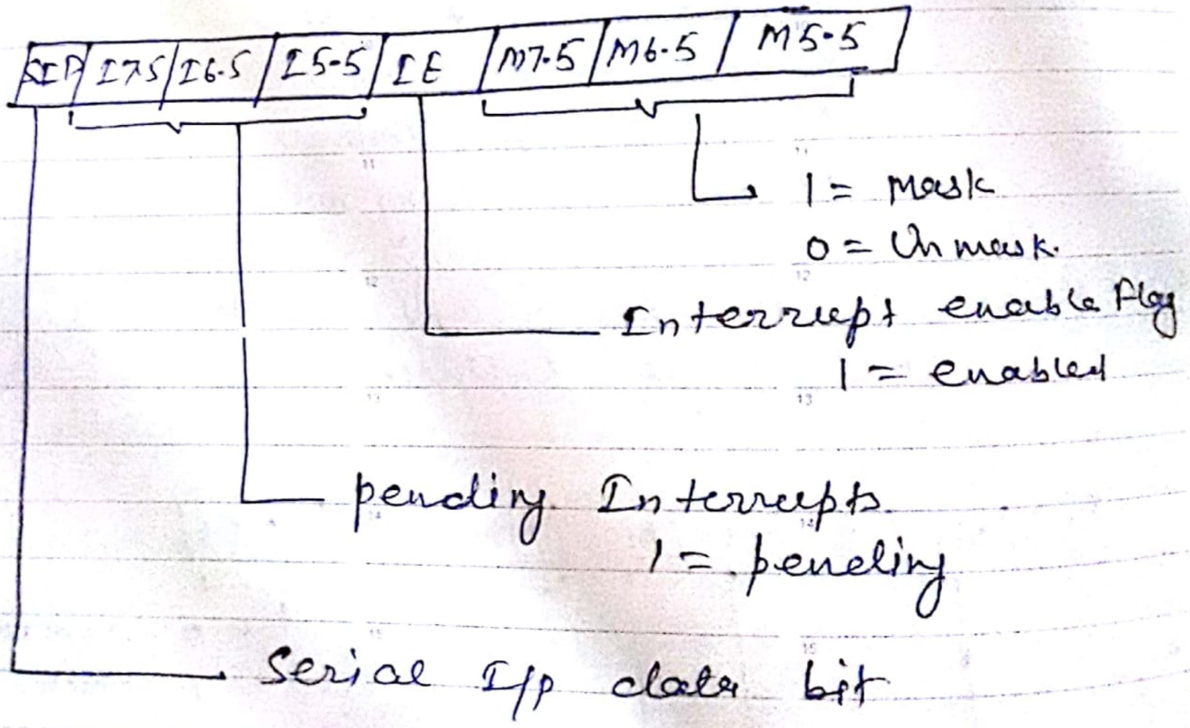
② DI → Disable the interrupt.
1- byte Instructions
No flags affected.
None Addressing mode.

③ SIM → Set Interrupt mask.
→ 1-byte Instruction.



Sim Can be used for 2 different functions-
1- To set mask for RST 7.5, 6.5, 5.5 . This Instruction reads the Contents of Accumulator & disable & Enable the Interupt according to Content of Accumulator
2 → To Reset 7.5 RST7.5 FF.
3-) Serial O/P data

RIM (Read Interrupt Mask)
This indicate the status of Interrupts. Hence status of Interrupt can be checked with the help by examining the content of Accumulator



Ex. write the Instructions To mask RST 7.5 & RST 6.5 Interrupts where as RST 5.5 Interrupt is available - Disable the serial Communication

Ans

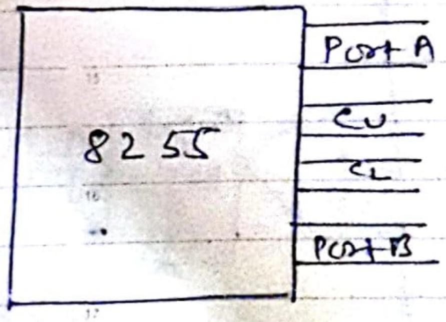
0	0	x	0	1	1	1	0

~~programmable Interrupt Controller~~ — ~~8259A~~

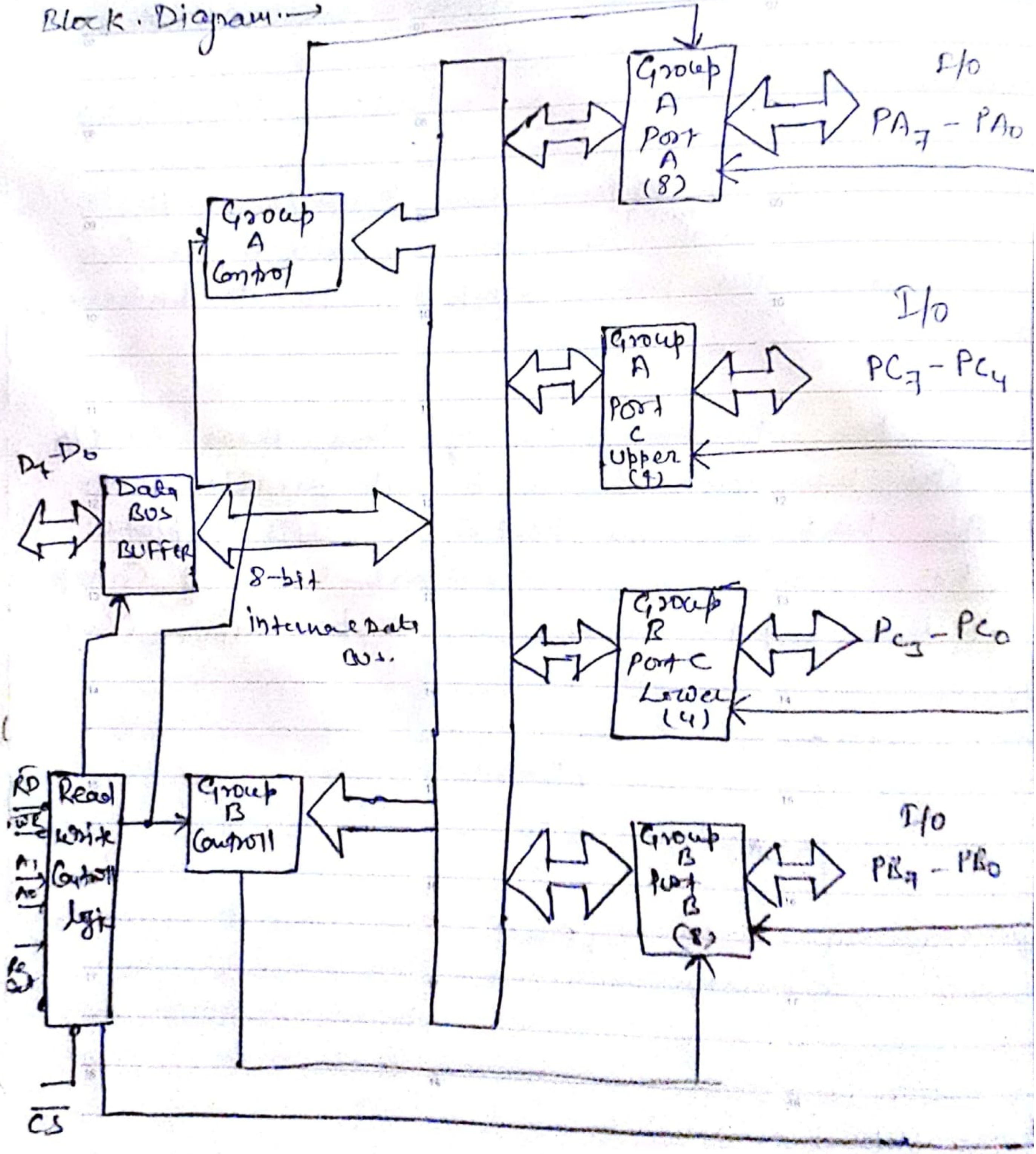
programmable peripheral Interface (PPI) 8255

8255 is a programmable, parallel I/o device it can be ~~used~~ programmed to transfer data under various condition. from simple I/o to Interrupt I/o.

8255 has 40 pins. out of ~~thes~~ these 24 I/O pins can be grouped in 8-bit parallel ports port A, port B. & port C. bits of port C can be used as individual bits of Group of 4 bits. port Cupper (Cu) & port Clower (CL)



Block. Diagram →



port A → Contains 8 - pin for sending / Receiving Data.

port B - 8 bit parallel ports used for sendig & receiving data.

port C → 8-bit parallel port is divided in to two 4-bit ports.

Control_ logic →

$\overline{RD}$ → This is active low signal, when the signal is Low, the MPU reads the data from the selected I/O port of 8255.

$\overline{WR}$ → This is active Low signal, when the signal is low, the MPU write in to selected I/O port of ~~8~~ 8255.

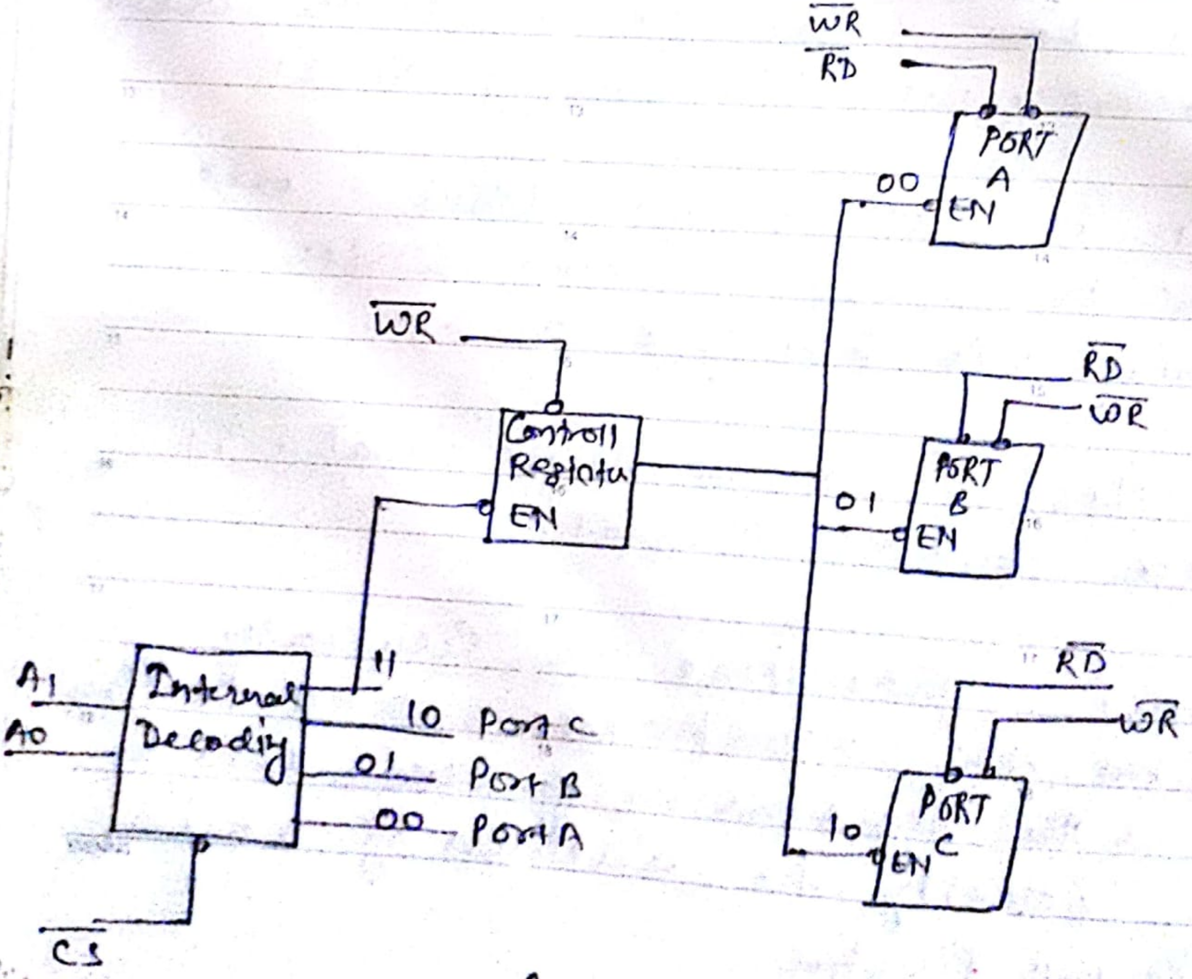
Reset → This is active High signal. It Clears the Controll Register.

Address Logic → These pins are Generally Connected to the micro processor address lines A_0 & A_1. The Combination of these two is used to idetify the address of various ports & Controll Register.

$\overline{CS}$ → It is Master Chip select signal
It is active Low signal.

Address Decoding for 8255

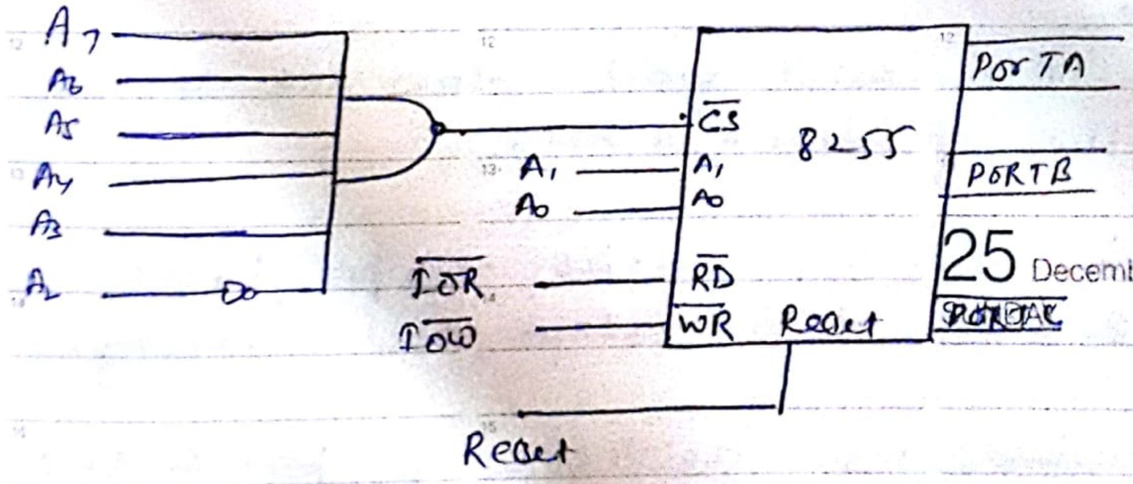
$\overline{CS}$	A_1	A_0	port selected
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Controll Register
1	x	x	8255 Not selected



Controll Logic Diagram of 8255

Controll Register → This Register in internal to the 8255 chip. The Contents of this Register is called Controll word. Controll word specify I/O function & Modes of various ports of 8255.

Ex. Find the address of various ports & Controll Register for the circuit shown below.



Sol)

$\overline{CS}$ Logic						Port select		Hex address	Port Name
A7	A6	A5	A4	A3	A2	A1	A0		
1	1	1	1	1	0	0	0	F8 H	A
1	1	1	1	1	0	0	1	F9 H	B
1	1	1	1	1	0	1	0	FA H	C
1	1	1	1	1	0	1	1	FB H	Controll Rgz

MODES of 8255

(i) BSR (bit set/Reset mode)
(ii) I/O mode.

Bit D_7 of Control Register specify either I/O functions or bit set/Reset functions

if Bit $D_7 = 1$ then Bit $D_7 - D_0$ determine the I/O functions in various mode.

If Bit $D_7 = 0$, port C operates in BSR mode. ~~B~~

the BSR Control word does not affect the function of port A & port B.

To Communicate with the peripheral through 8255A, three steps are necessary

(i) Determine the address of port A, B, C, & of the Controll Register according to the chip select logic & Address lines A_1 & A_0

(ii) Write a Control word in Controll Register

(iii) Write I/O instruction to Communicate with peripherals through ports A, B, C

Designed & Printed by : Eagle Press, 'Diary House', Chennai, India.

BSR Controll word → This Controll word. wh written in the Controll Register, set or reset one bit at a time

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	X	X	X	Bit	select		S/R

D_7 ↑ BSR mode

D_6–D_4: Not used

D_0 → Set = 1, Reset =

D_3 D_2 D_1	
0 0 0	= Bit 0
0 0 1	= Bit 1
0 1 0	= Bit 2
0 1 1	= Bit 3
1 0 0	= Bit 4
1 0 1	= Bit 5
1 φ 0	= Bit 6
1 1 1	= Bit 7

Input/output mode → I/P, O/P mode is used to Tran's fer the data to & from the peripharals.

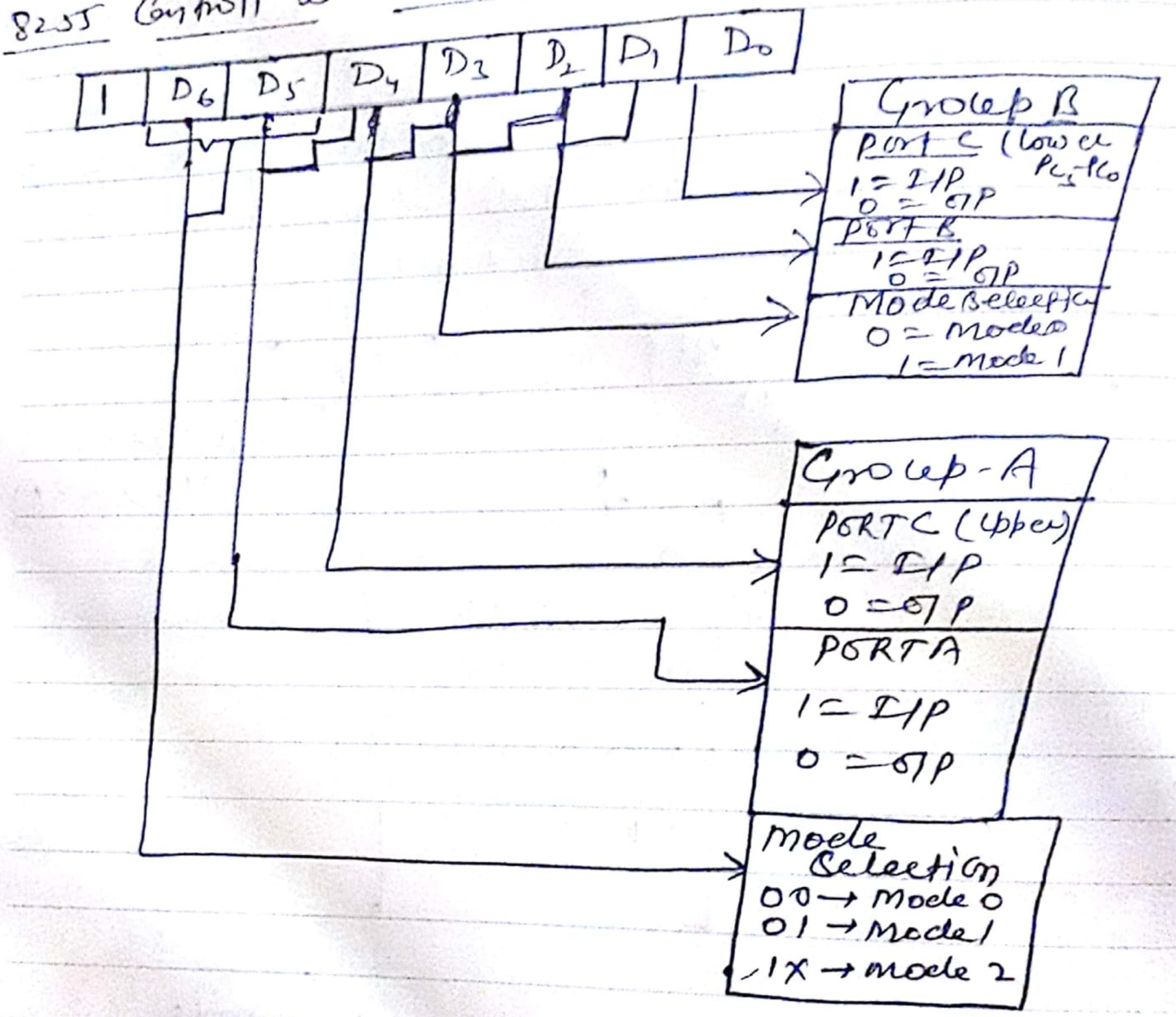
(i) Mode-0 → Simple I/P, O/P mode

(ii) ~~I/P, O/P~~ → mode-1 I/P, O/P with hand shake signals.

(iii) Bidirectional mode.

Notes

8255 Controll word format →



Q. Mode - 0 In this mode, ~~A~~ ports A & ports B are used as 2 8 bit Input/output ports & port C as two 4 bit ports. each port can be programmed to function as simply I/P or output ports mode-1 I/P, O/P Mode with handshake → In this mode I/P, or output data Transfer is Controlled by handshake signal.

Handshake signals are used to Synchronise the operation. This Signal is used to when the speed of peripheral & speed of data Transfer of microprocessor are not same.

The handshake signal is used to inform the microprocessor that peripheral device is ready to Communicate with the peripheral microprocessor or NOT.

Before The data Transfer exc The sh handshake signal are exchanged b/w the microprocessor & peripherals.

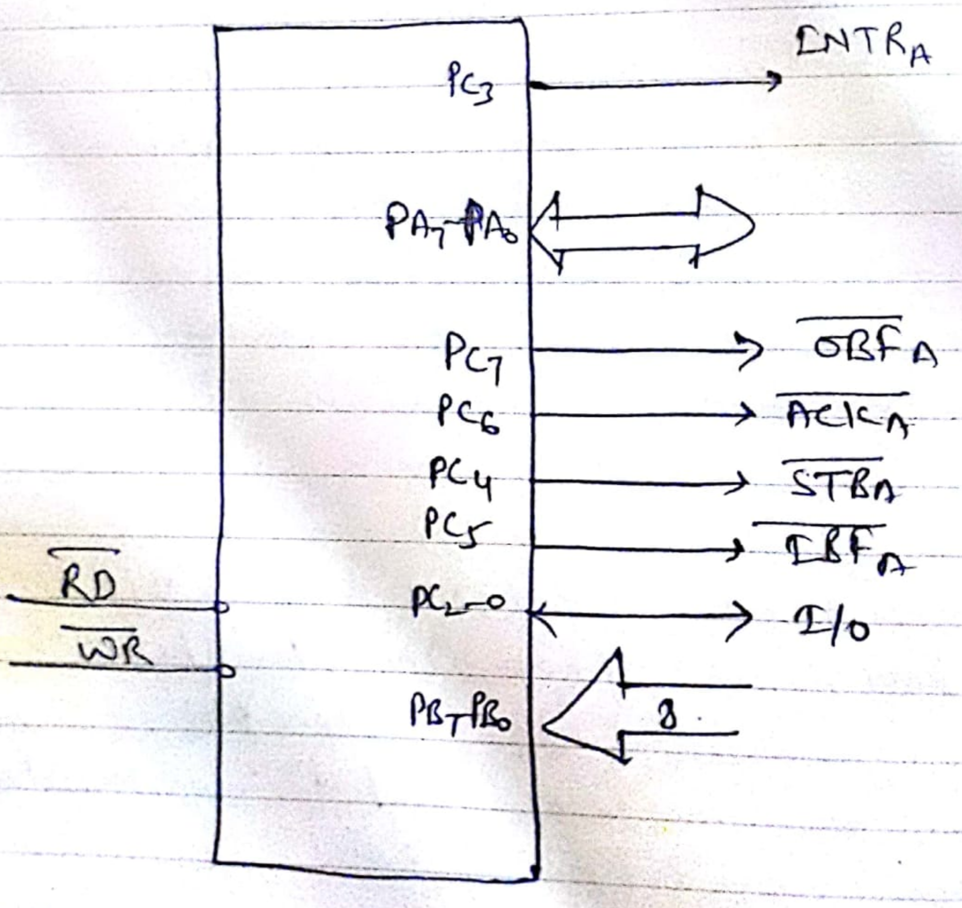
In this mode
(i) port A & port B work as 8 bit I/P, O/P ports.
(ii) Each port uses three lines for handshaking from port C
& Remaining two lines of port C used for simple Input / output functions.

mode 2 → Bi directional data Transfer → This mode allows bidirectional data Transfer over a single 8-bit data BUS only in Group A with port A as 8-bit bidirectional data BUS & lines Pc_3- PC_7 are used

Notes

for Hand shaking purpose , Bo Port B can be used in mode 0 or in mode 1.

Mode 2 is used in application such as data Transfer betn two computer & floppy disk controller ~~PAtt~~ Interface



8255 mode 2 with mode 0 (I/P)

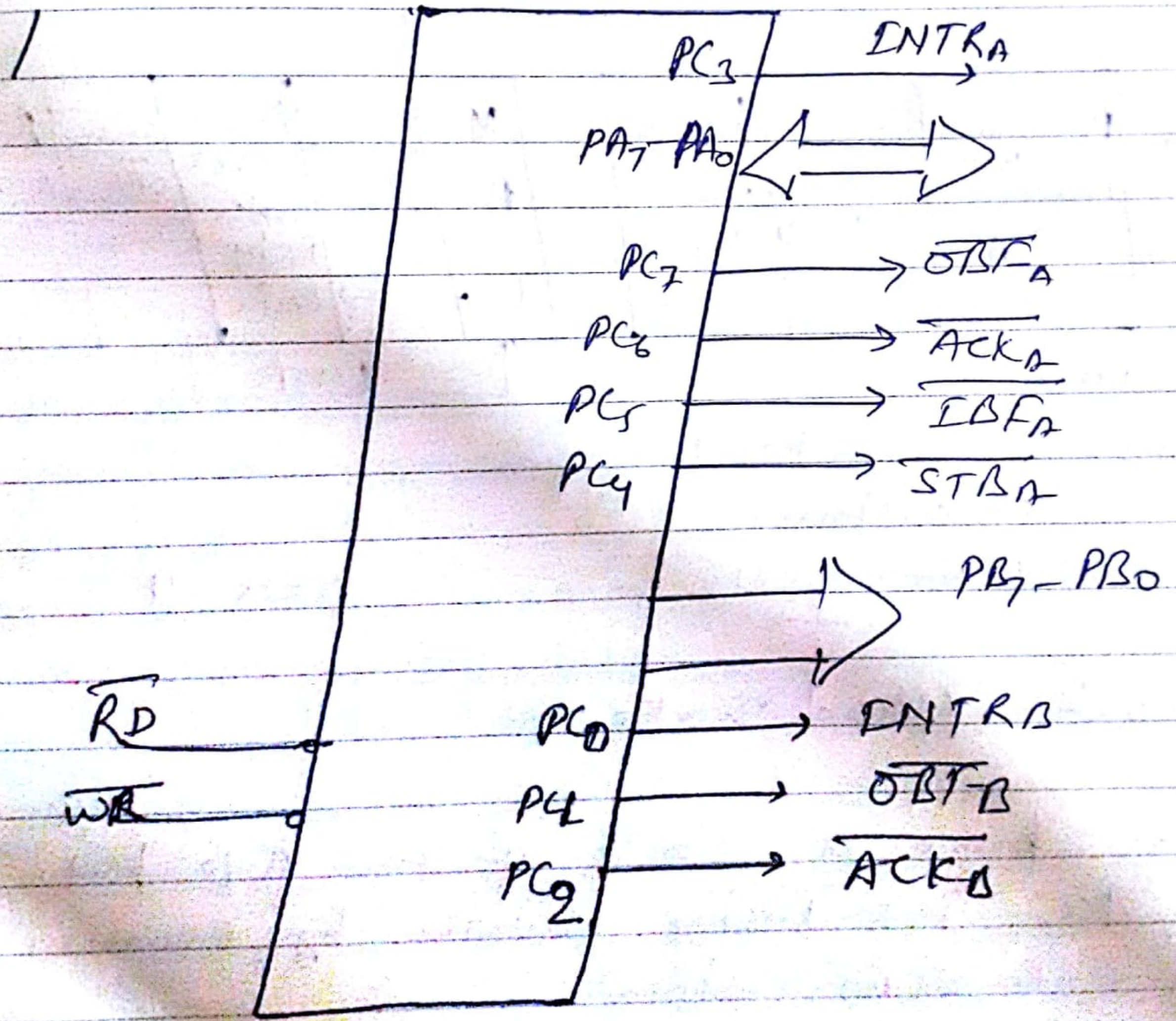
1	1	x	x	x	0	1	1/0

port A in

port C lower (PC_2-PC_0)
1= I/P
0= O/P

In mode 2 The port B Can be Configured In mode 0 or mode1

If port B is Configured in mode 0, three pins (PC0, PC1, DC2) used for Input/output purpose . . & If Port B in used in mode 1 then three pins (PC0, PC1, PC2) also used in hand shaking signals for Port B.



Controlword

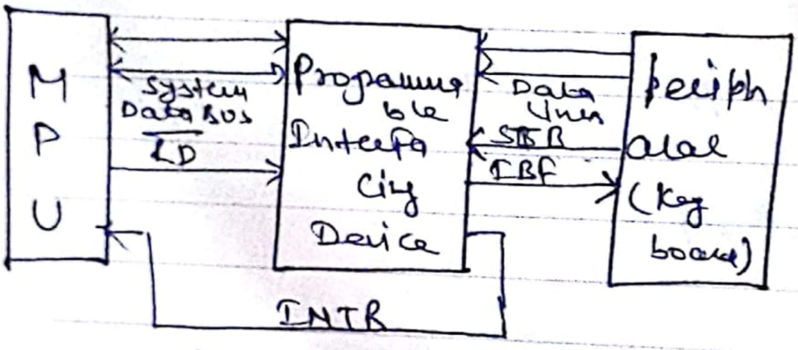
1	1	x	x	x	0	0	x

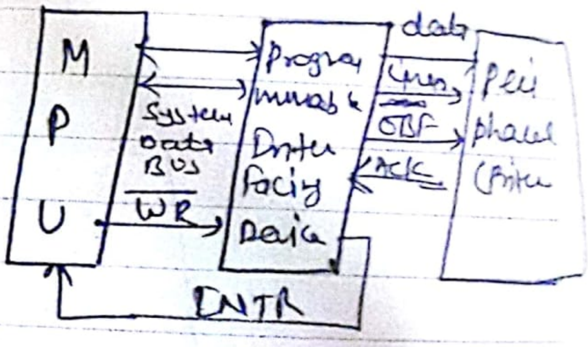
Hand shaking

Notes

programmable Device with hand shake signals. Since MPU & peripharals are operate at a Different speeds. Hence some signals are Transmittal ~~pro~~ prior to data Transfer bet the fast responding MPU & slow responding peripharals. like Priter, key board.

these signals are called Hand shake Signals. These signals are provided by programmable Device.





Interfacing Device with hand shake signals for data I/P.

data O/P.

Data I/P with Handshake. →

1) A peripharal place a data in the I/p port & inform the interfacing Device by sending strobe (~~SB~~ STB) signal

2) The Device inform the peripharal that it's I/p port is full & do not send the next byte untill this byte has been read. This message is conveyed ~~by~~ to the peripharal by sending signal IBF

Scanned by CamScanner

③ The MPU keeps checking the status untill a byte is available or The device Inform the MPU by sending an Interrupt.

④ The MPU Read the byte by sending the Controll signal $\overline{RD}$.

Data O/P with hand shake →

① The MPU writes a bytes in the O/P port of the programmable Device by sending the Controll signal $\overline{WR}$

② THE Device inform the peripheral by sending hand shake signal OBF that a byte is on the way

③ The peripheral acknowledge the the byte by sending the sending the signal ACK.

④ The Device Interrupts the MPU for the next byte programmable Interrupt Controller-(PC) PIC→ 8259

8259 is a programmable Interrupt Controller & it is used to provide a additional Interrupts

The 8255 Can -

(1) Manage 8 interrupts according to the instruction written in the Controll Register. This is equal to providing 8 interrupt pin on the processor in place of one INTR pin.

(2) mask each interrupt request individually

(3) Read the status of pending interrupt, in service interrupts, & masked interrupts.

(4) set up to work with either the 8085 micro processor mode or 8086/ 8088 micro processor mode

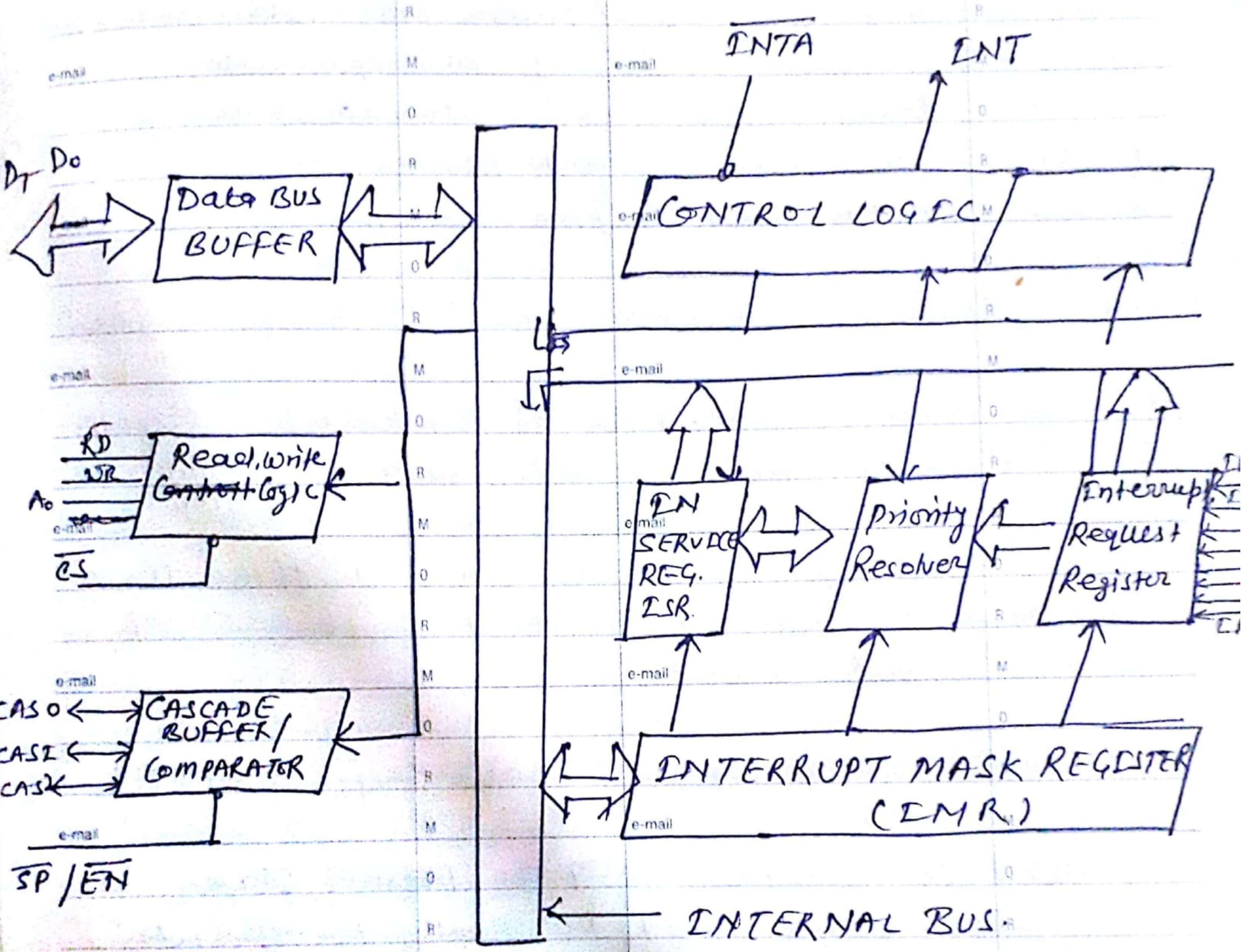
(5) provides eight level of interrupt priority in a variety of modes..

Block Diagram →

(1) Read / write logic → It has the following signal.

A_0 → when The Address Line A_0 is at logic 0, then Controller is selected to write a Command or read the status.

The chip Select & A_0 determine the the port address of Controller.



(ii) Controll Logic → It has two pins

INT —(as an output) → It is Connected to the interrupt pin of the MPU. when ever a valid interrupt is asserted, this signal Goes High & $\overline{INTA}$ (as input) is the Interrupt acknowledge signal from MPU.

Data BUS Buffer → It has bidirectional Data pins (D7-D0). these pins are Connected to the data BUS of microprocessor.

The Data BUS buffer ~~is used to~~ allows the 8085 to send Controll words to 8259 & read status word from 8259.

CAScade BUffer/ Comparator → This block is used to expend the Number of Interrupt levels by Cascading two or more 8259.

CAS0 → CAS2 (Cascade Lines) → these lines are used to Connect More than one 8259 in Cascade mode

$\overline{SP}/\overline{EN}$ (Slave program / Enable buffer) → It is used to Define 8259 as Master/ slave.

when single 8259 is used. It operates as master, & $\overline{SP}/\overline{EN}$ pin is Connected to +5V (logic 1) & when more than one 8259 are used then only one 8259 it can work as Master & Remaining 8259 can work as slave, & $\overline{SP}/\overline{EN}$ Connected to Ground (logic 0)

Interrupt Register & priority Resolver. →

The interrupt Request Register has eight I/P lines (IR0-IR7) for interrupts.

when these lines goes High, the request are stored in Register

The In service Register stores the levels that are currently being serviced.

Interrupt Mask Register stored the masking bits of interrupt lines to be masked.

The priority Resolver examines the three Register & Determine wheather INT should be sent to MPU OR NOT.

Interrupt operation →

To implement the interrupt. The interrupt enable flip-flop in the microprocessor should be enable & 8259 should be initialized by writting command words in the Controll Register.

there are two types of command words one initiallization command words (ICW) & second is operational command words (OCW)

The ICW is used to set up the proper condition & specify the RST vector address

The OCW are used to perform functions such as masking interrupts.

once the 8259 is initiallized then following sequence occur—

(i) The IRR is stores the interrupts requests

(ii) The priority Resolver Check the three register - the IRR for interrupt requests, IMR for masking bits & ISR for interrupt request being served.

Then It resolve the priority & set the INT signal High.

(iii) The MPU acknowledge the signal High by sending $\overline{\text{INTA}}$.

(iv) After Receiving the $\overline{\text{INTA}}$ signal, the appropriate priority bit in the ISR is set to indicate which interrupt level is being served & the corresponding bit in the IRR is Reset to indicate that the request is accepted. Then the opcode for CALL instruction

(v) (CD H) is placed on the Data BUS

(V) MPU Decodes the CALL Instruction & sends Two more $\overline{\text{INTA}}$ signals on the Data BUS

(VI) When 8259 Receives the second $\overline{\text{INTA}}$ It place the lower order byte of CALL address on the data BUS.

At the third $\overline{\text{INTA}}$, It places the Higher order byte of the address Data BUS.

(VII) During the third $\overline{\text{INTA}}$ pulse, The ISR

Scanned by CamScanner

bit is Reset either by automatically or by a Command word that is issued at the end of Comma service routine

(VIII) The program sequence is Transferred to The memory location specified by CALL Instruction.

priority modes →

(i) Fully Nested modes. → This is a General purpose mode in which all Interrupt requests are arranged from Highest to Lowest

IR0	IR1	IR2	IR3	IR4	IR5	IR6	IR7
4	5	6	7	0	1	2	3

↑ Lowest priority

↑ Highest priority

when an Interrupt is acknowledged, the Highest priority request is determined & it's vector location is place on Data Bus.

(ii) Automatic Rotation mode → In ~~this~~ mode some applications there can be equal priority assigned to No. of Interrupting Device. Hence with the help of this mode, The request after being served received the lowest priority

IR0	IR1	IR2	IR3	IR4	IR5	IR6	IR7
4	5	6	7	0	1	2	3

Lowest

Assuming that IR4 has just being served.

(iii) specific Rotation mode → this is Similar to Automatic Rotation mode except that the programmer can select any IR for the lowest priority.

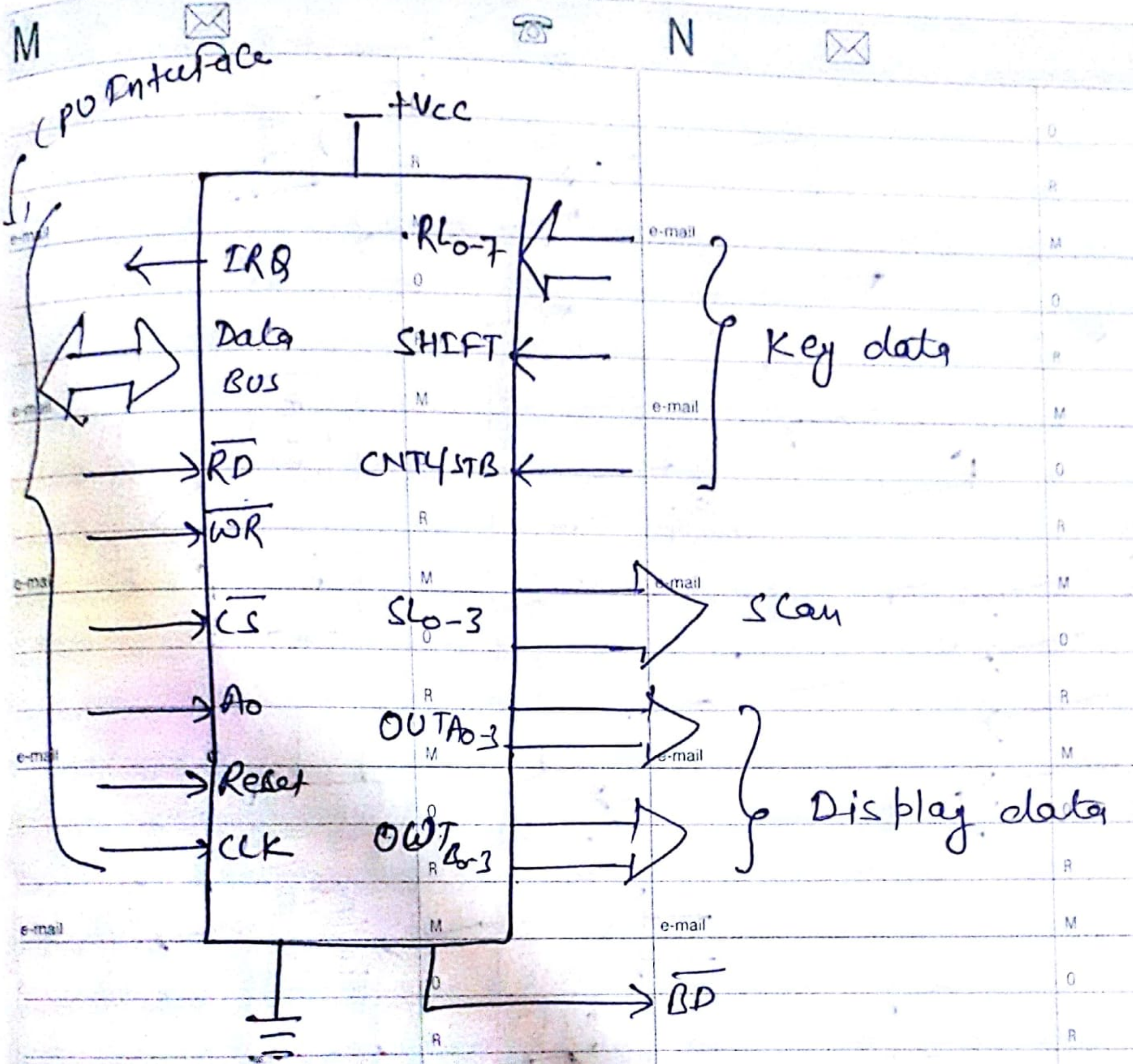
ey Board / Display Controller 8279.

8279 is a hard ware oft approch to interface a Key board & Display unit. The disadvantage of soft ware approch is that micro processor is occupied for a Considerable amount of time in checking the keyboard & refreshing the display

The 8279 is a 40 pin device with two segments key board & display.
The key board segment is connected to key metrix. Key board entries are stored in internal FIFO memory. an interrupt signal is Generated with each entry
The display segment can provide display interface with devices like LED's. this segment has 16X8 R/W memory (RAM) which can be used to read/write Information for display purpose

R (1) Key board section —

(1) RL0 – RL7 (Return Lines) → These Input Lins



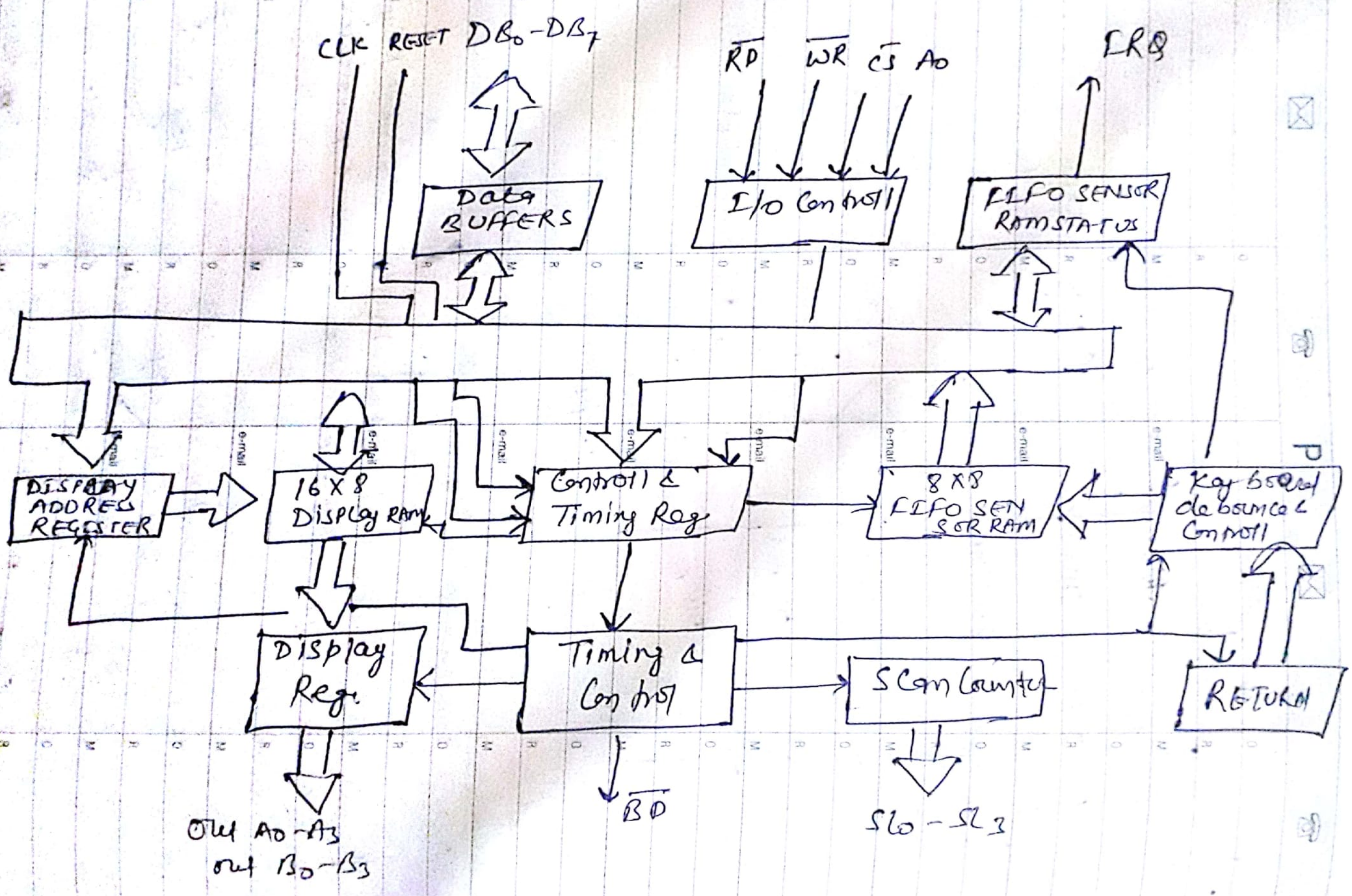
are Connected to Key board.

(ii) SHIFT → It is used in the scanned keyboard when key is closed the shift I/P status is stored.

(iii) CNTL/STB (Control/Strobe) → The status of SHIFT key & Control key is can stored along with key closure. The Key board can operate in two mode—

Display Address Register

(i) Two key lock out mode— In this


CLK
RESET
$DB_0 - DB_7$
$\overline{RD}$
$\overline{WR}$
$\overline{CS}$
A_0
IRQ
Data BUFFERS
I/O Controll
FIFO SENSOR RAM STATUS
DISPLAY ADDRES REGESTER
16 X 8 Display RAM
Controll & Timing Reg
8 X 8 FIFO SENSOR RAM
Key board debounce & Controll
Display Reg.
Timing & Control
Scan Counter
RETURN
Out $A_0 - A_3$
out $B_0 - B_3$
$\overline{BD}$
$SL_0 - SL_3$

Then only First key is recognized.

N-Key roll over mode → In this mode Simultane keys are recognized & their codes are stored in internal buffer.

<u>Scan Section</u> → This section has a scan counter & four scan lines. These four lines can be decoded to Generate 16 scan lines. These lines can be connected to the rows of matrix key board.

<u>Display Section</u> → It contains eight Input lines divided in to two Groups $A_0 - A_3$ & $B_0 - B_3$. These lines can be used as either as a group of Eight lines or two group of four. The display can be blanked by usin $\overline{BD}$ line.

<u>Processor Section</u> →

(i) <u>$DB_0 - DB_7$</u> → Bidirection Data Bus used to Transmit the command.

(ii) <u>IRQ</u> → Active High signal. This pin Goes High whenever entries are stored in FIFO RAM. This signal is used to interrupt the MPU to indicate the availibility of data.

~~iii~~