SEMICONDUCTOR MEMORY INTERFACING

→ Semiconductor memories are two types. They are RAM and ROM.

→ Semiconductor RAM's is of broadly two types.

* Static RAM
* Dynamic RAM

→ For example 4KB of memory is there, for addressing 4KB of memory 12 address lines are required.

→ In general to address a memory location out of 'N' memory locations, we will require atleast 'n' bits of address i.e n address lines where $n = \log_2 N$.

→ If out of 'N' locations only 'p' memory locations are to be interfaced, then the least significant P address lines out of available 'n' address lines can be directly connected from µp to memory chip & remaining (n-p) higher order address lines may be used for address decoding. The o/p of decoding ckt is connected with the $\overline{cs}$ pin of memory chip.

The General procedure of static memory interfacing with 8086:-

1) Arrange the available memory chips so as to obtain 16 bit data bus width. The upper 8 bit bank is called "odd addressing memory

bank" & the lower 8 bit bank is called "even address memory bank"

2) Connect the available memory address lines of memory chips with microprocessor & also connect the memory $\overline{RD}$, $\overline{WR}$ i/ps to the corresponding processor control signals. Connect the 16 bit data bus of memory bank with that of the microprocessor 8086.

3) The remaining address lines of the microprocessor, $\overline{BHE}$ and A_0 are used for decoding the required chip select signals for the odd & even memory banks. The $\overline{CS}$ of the memory is derived from the o/p of decoding circuit.

Interface two 4KX8 EPROMS & two 4KX8 RAM chips with 8086. Select suitable maps.

Total 8K bytes of EPROM need 13 address lines A_0-A_{12} & the address lines A_{13}-A_{19} are used for decoding to generate the chip. select $\overline{BHE}$ goes low when a transfer is at odd address (or) higher byte of data is to be accessed

Let us assume that the latched address $\overline{BHE}$ & demultiplexed data lines are readily available for interfacing.

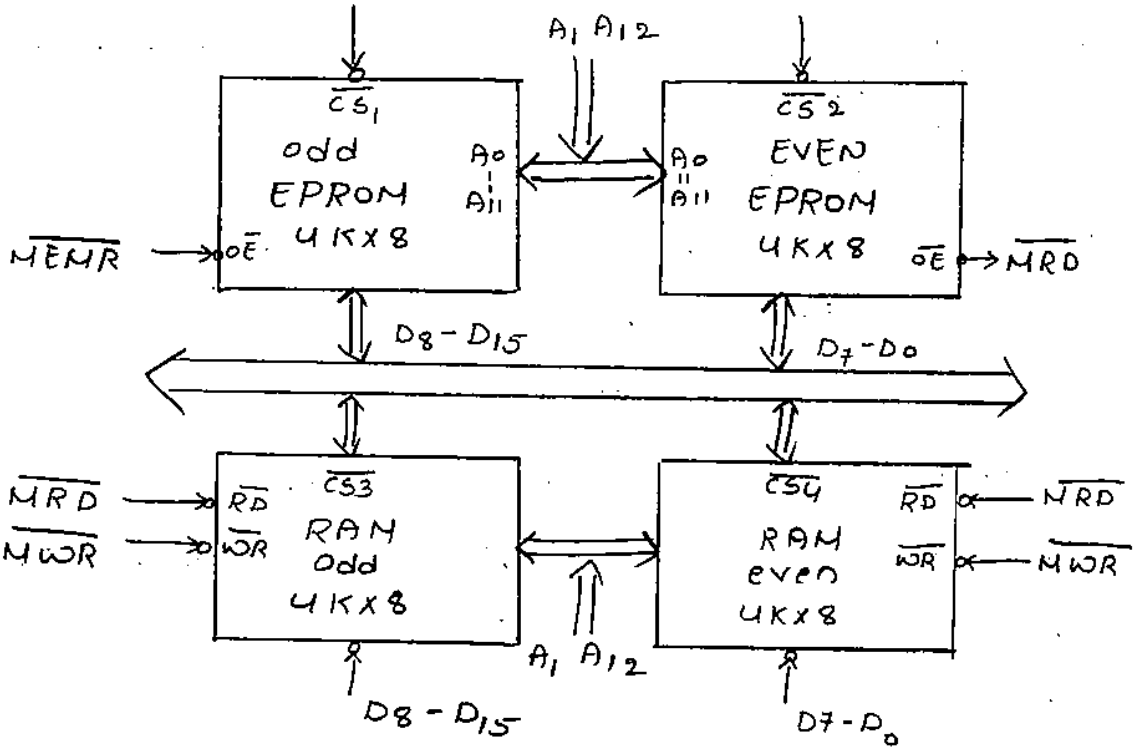
Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
						EPROM						8K								
FE000H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
FDFFFH	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
						RAM						8K								
FC000H	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

→ The memory system in this example contains in total four 4Kx8 memory chips. The two 4Kx8 chips of RAM & ROM are arranged in parallel to obtain 16 bit data bus.

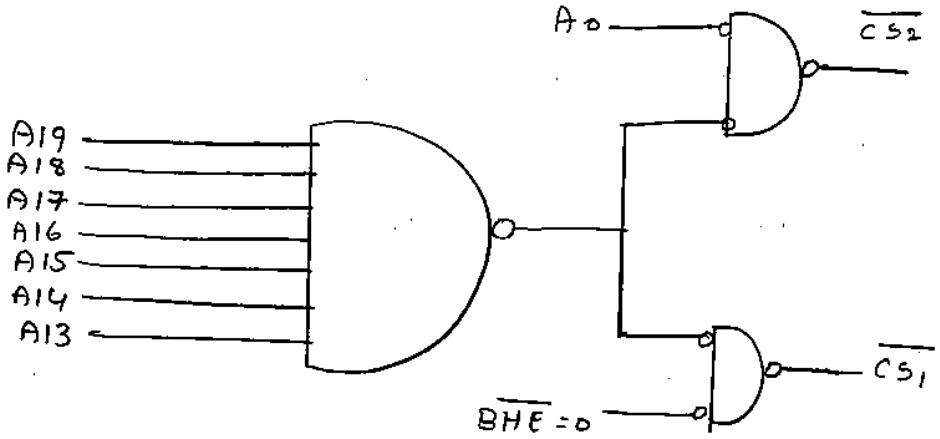
→ If A_0 is 0 the address is even & is in RAM, then the lower RAM chip is selected indicating 8 bit transfer at an even address.

→ If A_0 is 1, the address is odd & is in RAM the $\overline{BHE}$ goes low, the upper RAM chip is selected, indicating 8 bit transfer at an odd address.

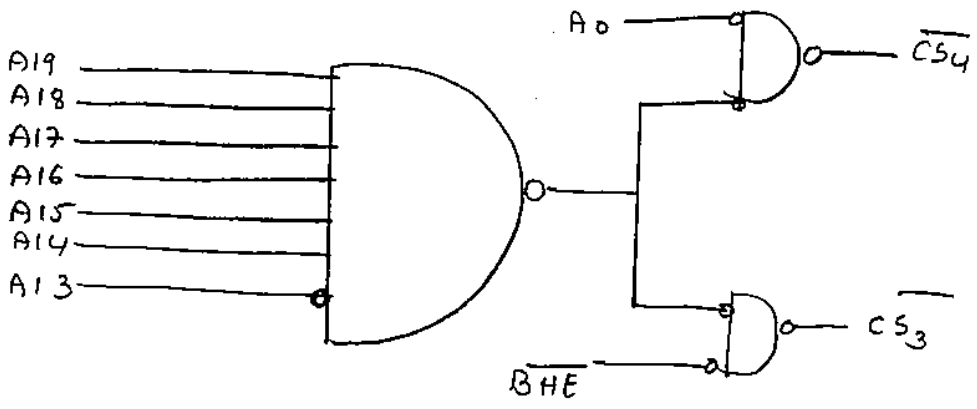
→ If selected addresses are RAM, the respective ROM chips are selected. If at a time A_0 & $\overline{BHE}$ are '0', both RAM or ROM chips are selected i.e the data transfer is of 16 bits.



For EPROM:-



For RAM:-



DMA

→ In the applications where the CPU is to transfer bulk data, it may waste a lot of time in just transferring the data from source to destination using program controlled data transfer

→ The alternate way of transferring the bulk data is the direct memory access (DMA) technique in which data is transferred under the control of DMA controller, after it is properly initialised by the CPU.

→ A DMA controller is designed to complete the bulk data transfer much faster than the CPU.

DMA controller 8257:-

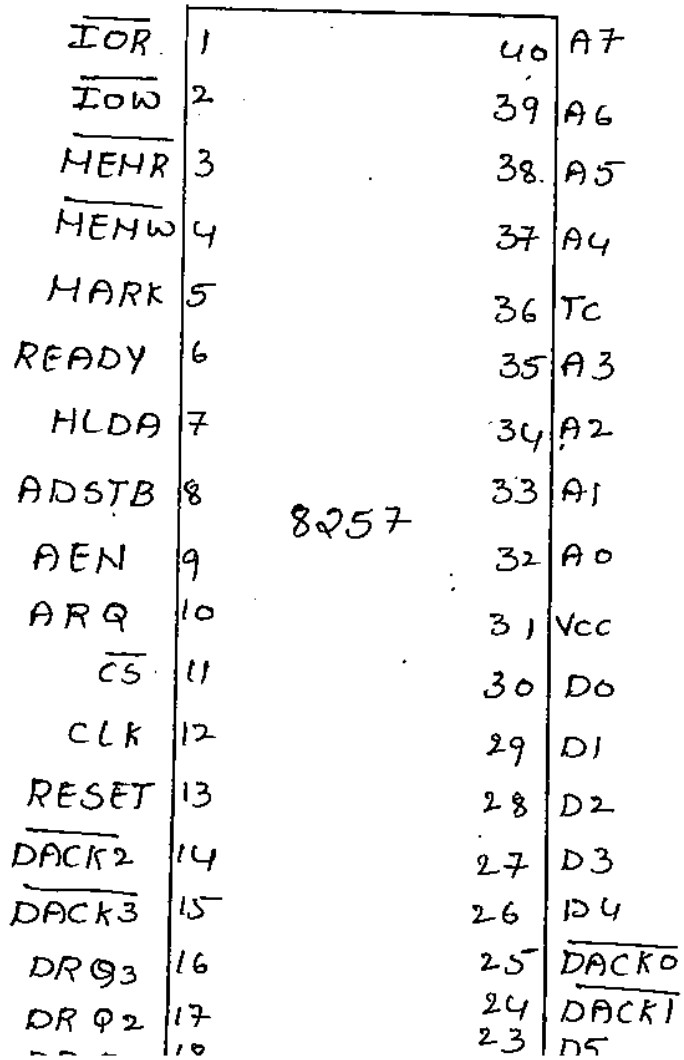
→ DMA mode of data transfer is the fastest among all the modes of data transfer. In this mode, the device may transfer data directly to/from memory without any interference from CPU. The device requests the CPU (through DMA controller) to hold its data, address and control bus. so that the device may transfer data directly to/from memory. The DMA data transfer is initiated only after receiving HLDA signal from the CPU. For facilitating DMA type of data transfer b/w several devices a DMA controller may be used.

→ Intel 8257 is a four channel DMA controller designed to be interfaced with their

→ The 8257 on the behalf of the devices a requests the CPU for bus access using ping HOLD in minimum mode. In min max mode of microprocessor $\overline{RQ}/\overline{GT}$ pin is used to request the bus access.

→ On receiving the HLDA signal (in min mode) or $\overline{RQ}/\overline{GT}$ signal (in max mode) from CPU, the requesting device gets the access of bus & after completing of data transfer, then over the control of the bus back to the CPU.

Pin diagram of 8257:-



$DRQ_0 - DRQ_3$:

These are the four individual channel DMA request i/ps, by the peripheral devices for requesting DMA services. The DRQ_0 is the highest priority while DRQ_3 has the lowest one, if the fixed priority mode is selected.

$\overline{DACK_0} - \overline{DACK_3}$:

These are the active low DMA acknowledge o/p lines which inform the requesting peripheral that the request has been accepted & bus is relinquished (give up) by the CPU. These lines may act as stroble lines for the requesting devices.

$D_0 - D_7$:

These are bidirectional data lines used to interface the system bus with the internal data bus of 8257. These lines carry command words to 8257 & status words from 8257 slave mode i.e under control of CPU.

When 8257 is the bus master (master mode i.e not under CPU control) it uses $D_0 - D_7$ lines to send higher byte of generate address to the latch. The address is transferred over $D_0 - D_7$ during the first clock cycle of DMA cycle. During the rest of period data is available on the data bus.

IOR:

This is an active low bidirectional tristate i/p line in slave mode. In slave mode, this i/p signal is used by the CPU to read the internal register of 8257.

This line acts as o/p in master mode. In master mode this signal is used to read the data from peripheral during a memory write cycle

IOW:

This is an active low bidirectional tristate i/p lines in slave mode, used to load the contents of data bus to the 8 bit mode register (or) upper/lower byte of 16 bit DMA address register (or) terminal count register

In master mode, it is a control output that loads the data to a peripheral during DMA memory read cycle.

CLK:

This is a clock frequency i/p required to derive the basic system timings for internal operation of 8257.

Reset:

This active high asynchronous i/p disables all the DMA channels by clearing the mode register & tristates all the control lines.

A_0-A_3: There are four least significant address lines. In slave mode they act as i/p which select one of the register to be read or written.

In master mode, they are four least significant memory address o/p lines generated by 8257.

A_4~A_7: This is the higher nibble of the lower byte address generated by 8257 during master mode of DMA operation.

$\overline{CS}$: This is an active-low chip select line that enables the read/write operations from 8257 in slave mode.

In master mode; it is automatically disabled to prevent the chip from getting selected while performing the DMA operation.

Ready: This is used in interfacing slower peripherals. This ask signal from slow peripherals. If it is high, indicates that peripherals are ready for data transfer.

HRQ: The hold request o/p, request access of system bus.

HLDA: The CPU derives this i/p to DMA controller, while granting the bus to the device. This i/p if high indicates to DMA controller that bus

MEMR: This active low memory read o/p is used to read data from addressed memory locations during DMA read cycles.

MEMW: This active low three state o/p is used to write the data to address memory location during DMA write operation.

ADSTB: This o/p line from 8257 strobes higher byte of memory address generated by DMA controller into latches.

AEN: This o/p is used to disable system data bus & control bus driven by CPU. This is also may be used to transfer higher byte of generated address over the data bus

If the 8257 is I/o mapped, this should be used to disable the other I/o devices when DMA controller address is on address bus.

TC: Terminal count o/p indicates to the currently selected peripheral that the present DMA cycle is the last for the previously programmed data block. If the Tc stop but in mode set register is set, the selected channel will be disabled at the end of DMA cycle.

MARK: The modulo 128 mark o/p indicates to

cycle is the 128th cycle since the previous MARK o/p. The mark will be activated after each 128 cycles (or) integral multiples

Vcc: This is a +5v supply pin acquired for the operation of circuits.

GND: This is a return line for p supply

Internal Architecture of 8257

The chip supports four DMA channels i.e four peripheral devices can independently request for DMA data transfer through these channels at a time.

The DMA controller have 8 bit internal data buffer, a read/write unit, a control unit, priority resolving unit along with set of registers.

DRegister organization of 8257: The 8257 performs the DMA operation over four independent DMA channels. Each of four channels of 8257 has a pair of two 16 bit registers. These are DMA address registers terminal count register. Also there are two common registers for all channels namely mode set register & status register. The CPU selects one these ten registers using address lines $A_0 - A_3$.

DMA address Registers: Each DMA channel has one DMA address register. The function of this register is to store the address of starting memory location, which will be accessed by DMA channel. Thus the starting address of memory block which will be accessed by device is first loaded in DMA register of channel.

Terminal count register: This register is used to cease (or) stops the data transfer through DMA channel after required no. of DMA cycles.

The low order 14 bits of terminal count register are initialized with binary equivalent of no. of required DMA cycles mines one. After each DMA cycle, the TC register count will be decremented by one & finally it becomes zero after the required no. of DMA cycles are over

The bits 14 & 15 of this register indicates the type of DMA operation. If the device wants to write data into memory, the DMA operation is called DMA write operation. Bit 14 of register in this are is set to one & bit 15 will be set to zero.

Mode set Register:

The function of the mode set register

is to enable the DMA channels individually & also to set the various modes of operations. DMA channel should not enabled till the DMA address register of Tc register contain valid information.

Mode set register is normally programmed by the CPU after initializing the DMA address register & Tc register.

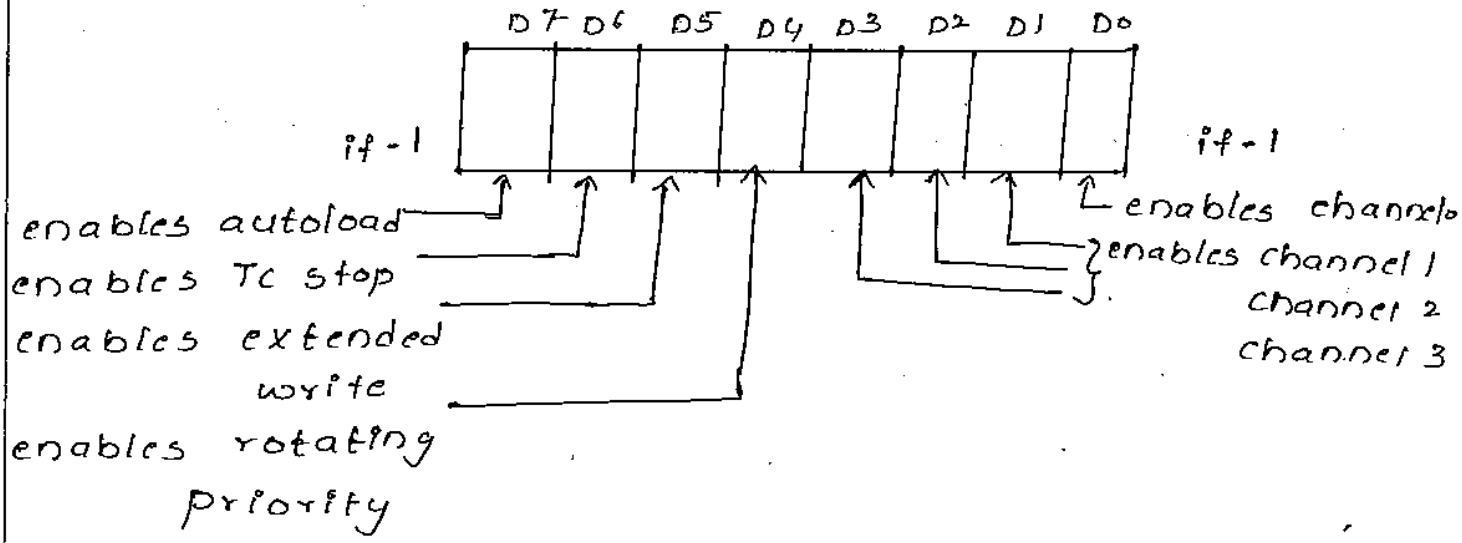
These bits D_0-D_3 enable one of the four DMA channels of 8257. For ex: if D_0 is 1 channel '0' is enabled. If D_4 is set rotating priority is enabled, otherwise it is normal i.e fixed priority is enabled.

If Tc stop bit is set, the selected channel is disabled after Tc condition is reached & it furthur prevents any DMA cycle on channel. To enable channel again, this must be reprogrammed. If Tc stop bit is zero, the channel is not disabled even after the count reg reaches zero & furthur requests are allowed on same channel.

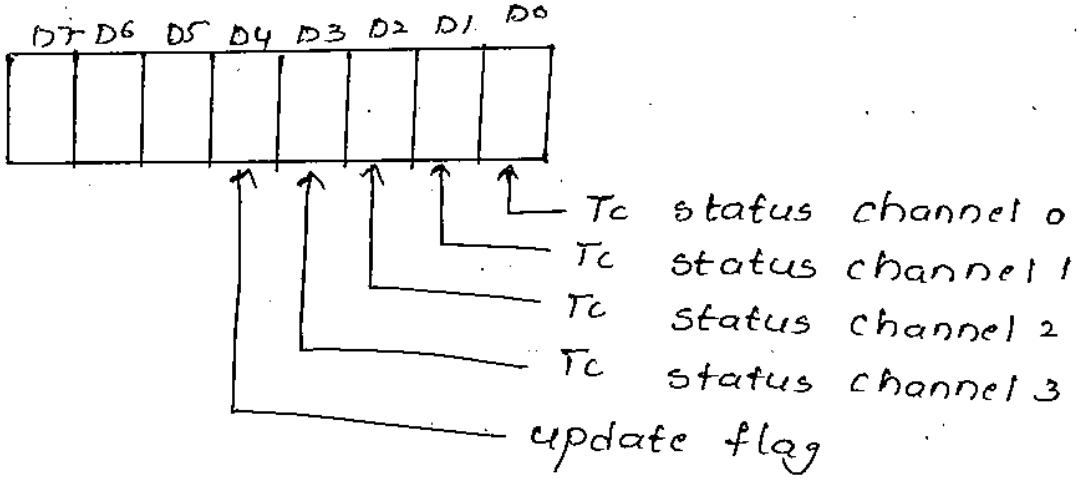
The auto load bit, if set, enables channel 2 for block chaining operations, without immediate software intervension b/w two

using DMA, channel 2 registers are reloades with corresponding channel 3 registers for the next block transfer, if update flag is set..

The extended write bit is set, extends duration of $\overline{MEMW}$ & $\overline{IOW}$ signals by activating them earlier. This is useful in interfacing peripherals with different access times



Status Register:



The lower order 4 bit of this register contains Tc status for the four individual

set, it indicates that the specific channel has reached the Tc condition. These bits remains set till either the status is need by CPU or the 8257 is reset.

If update flag is set, the contents of channel 3 reg are reloaded to corresponding reg. of channel 2, whenever channel 2 reaches the Tc condition, after transferring the one block & next block is to be transferred using autoload feature of 8257

Databus buffer, Read/write logic, control unit & priority resolver:

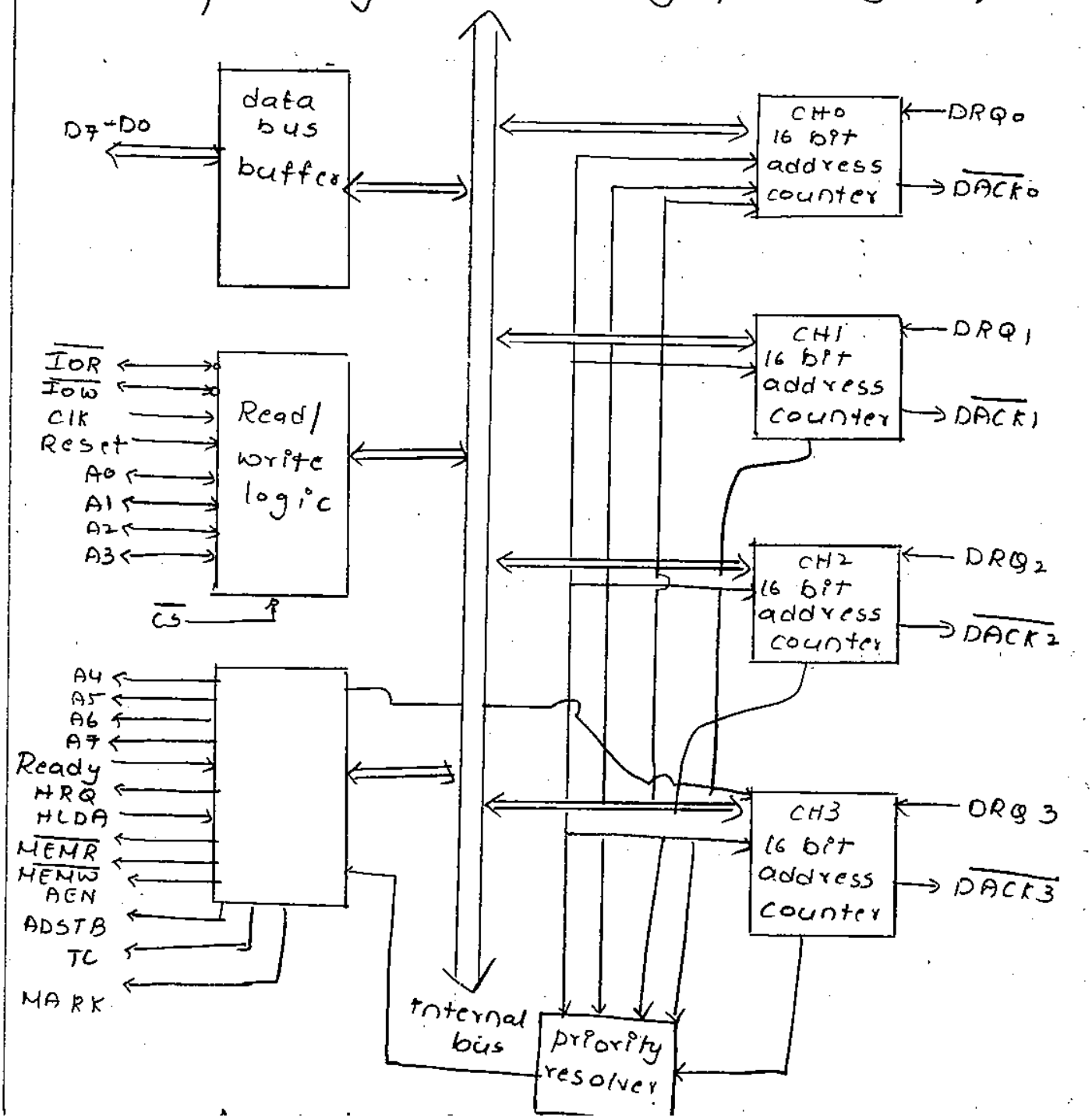
The 8-bit bidirectional data bus buffer interfaces the internal bus of 8257 with the external system bus under control of control signa

In slave mode, the read/write logic accepts the Ilo Read & Ilo write signals, decodes the A_0-A_3 lines. IO read signals is used to read the content of selected reg. & Io write signal is used to load contents on data bus to the addressed internal reg. In master mode, the read/ write logic generates, $\overline{IOR}$ & $\overline{IOW}$ signals used to read selected peripheral & load into selected

Peripheral respectively

The control logic controls the sequence of operations & generates required control signals like AEN, ADSTB, $\overline{MEMR}$, $\overline{MEMW}$, TC, MARK along with address lines & A_0-A_7 in master mode.

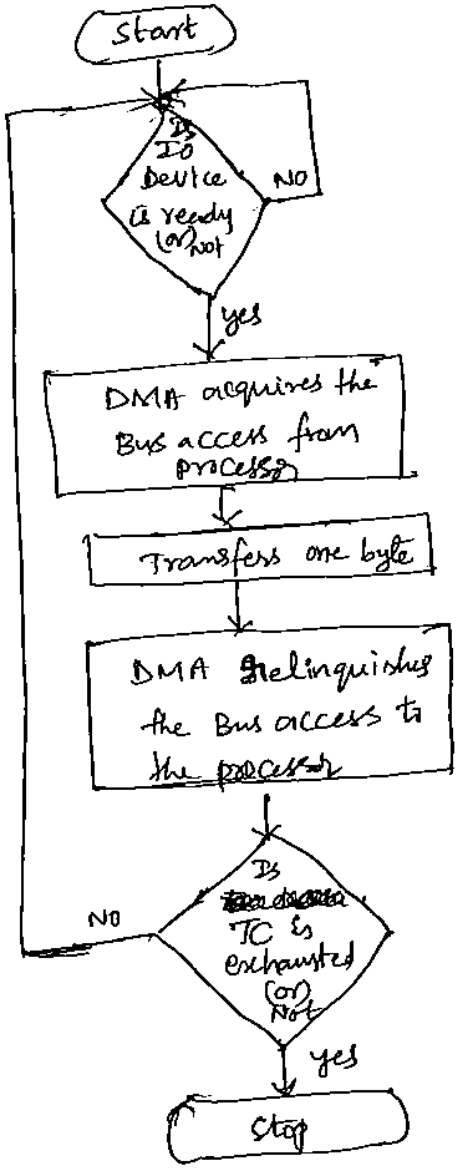
The priority resolver, resolves priority of four DMA channels depending upon whether normal priority or rotating priority is programm



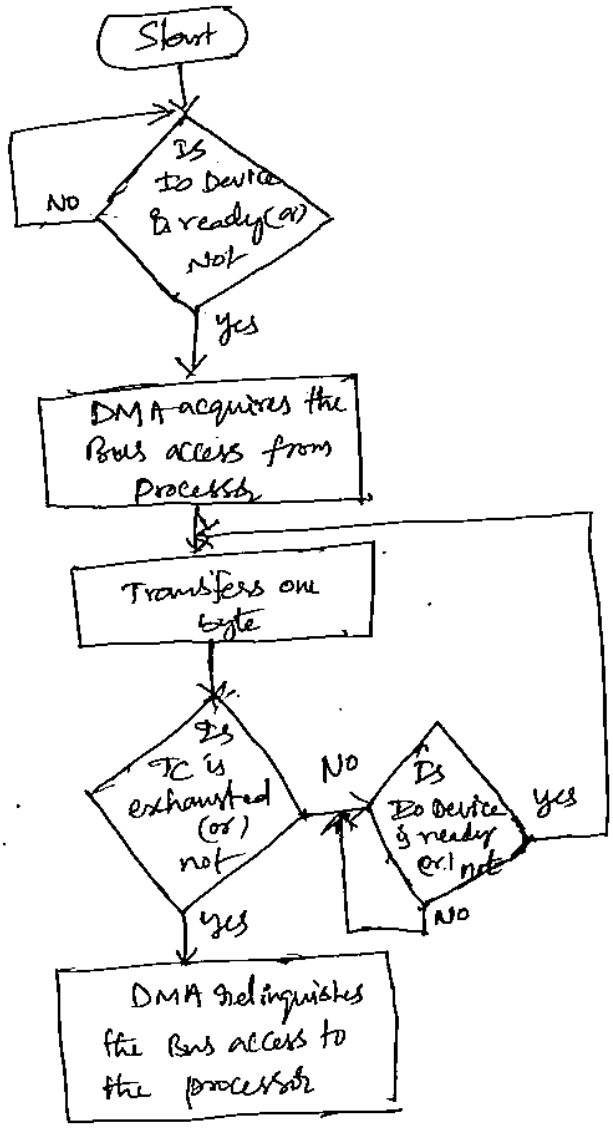
Data Transfer methods of DMA

1. Single byte transfer method.
2. Block data transfer method.
3. Demand (or) Burst transfer method.

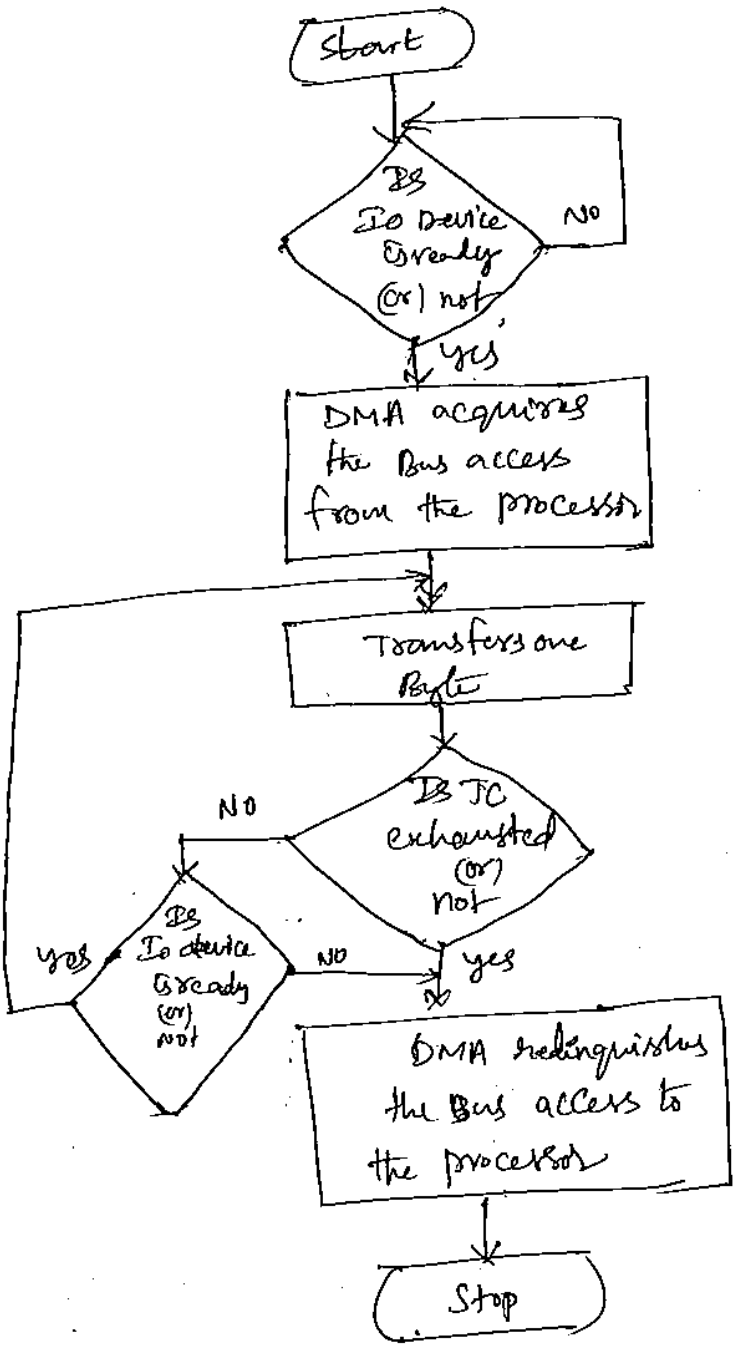
(1) Single byte transfer method:



(2) Block Data transfer



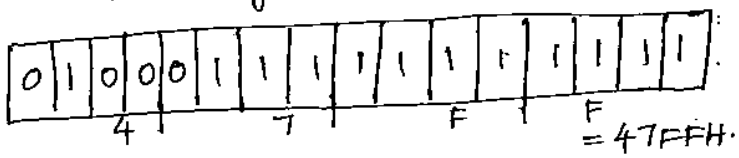
Demand Data transfer



(P) Interfacing of 8257 with 8086, so that CH0 DMA address register as an IO add. 80H and modeset Register as Io add. 88H. Initialize the 8257 with the normal priority with TC Stop and non extended write, Auto load is not required. The transfer is to take place channel zero. Write a ALP to move 2KB of data from pheripheral to memory address 2000:5000H with the above initialization.

Sol. 1. DMA add register contents are 5000H. (starting memory location)

2. TC register contents : 47FFH.



→ 2KB data is to be transferd so that 2K DMA cycles are required. Hence TC first 14 bits are filled with the Binary equivalent of 2K-1. & last two bits are DMA type bits. Here it is write operation. Hence 14th & 15th bits are 01.

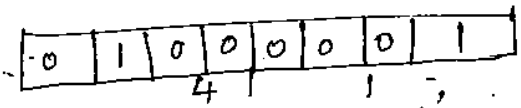
Binary equivalent of 2K-1 =

$2^{11}-1$ = 11111111111

2^{11} = 100000000000

$2^{11}-1$ = 11111111111

3. Mode set register contents = 41 H.



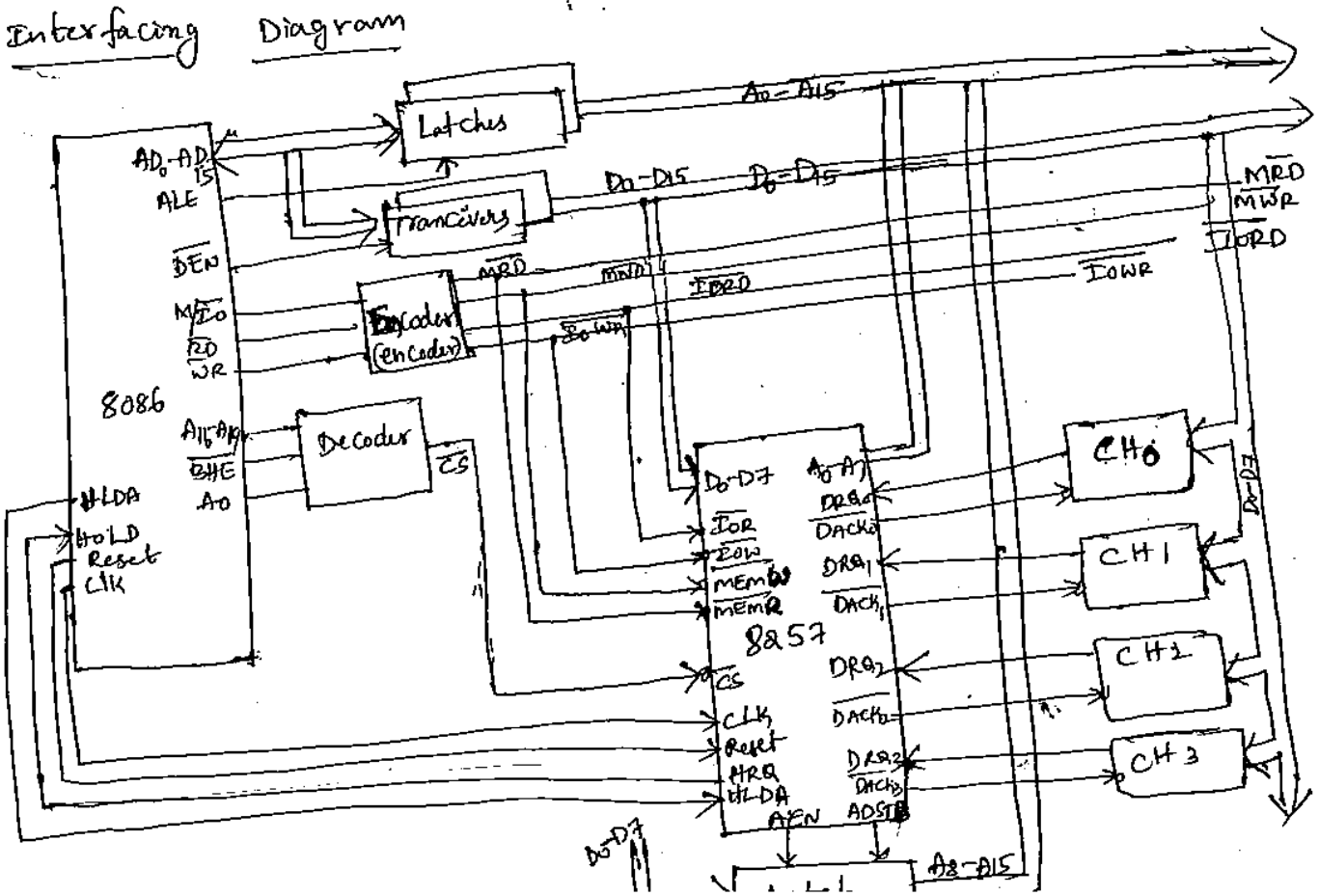
Then Io addresses of

DMA address Registers = 80H
TC register = 81 H
Mode set register = 88H.

$A_3A_2A_1A_0$ of CH0
DMA add. reg = 0000
TC reg = 0001
Mode set reg = 1000.

```
Data segment
DMAL equ 00H
DMAH equ 50H
TCL  equ FFH
TCH  equ 47H
MSR  equ 41H
Data ends
Assume CS: code, DS: Data
Code segment
Start: MOV AX, data
       mov DS, AX
       MOV AL, DMAL
       OUT 80H, AL
       mov AL, DMAH
       OUT 80H, AL
       MOV AL, TCL
       OUT 81H, AL
       MOV AL, TCH
       OUT 81H, AL
       MOV AL, MSR
       OUT 88H, AL
       INT 3H
Code ends
end start.
```

Interfacing Diagram



Interrupt cycle of 8086:-

→ There are two types of interrupts. These are external interrupts and internal interrupts.

→ In external interrupt, an external device or a single interrupts the processor from outside or in other words the interrupt is generated outside the processor for ex: Keyboard interrupt.

→ The internal interrupt, it is generated internally by the processor or by the execution of an interrupt instruction.

The example of this interrupt are divide by zero interrupt, overflow interrupt, interrupts due to INT instructions.

→ Suppose an external device interrupts the CPU at the interrupt pin, other NMI or INTR of 8086, while the CPU is executing an instruction of a program. The CPU first completes the execution of the current instruction. The IP is then incremented to point to the next instruction. The CPU then ask the requesting device on its $\overline{INTA}$ pin immediately if it is NMI, TRAP or divide by zero interrupt If it is INTR, the CPU checks the IF flag. If flag is set, the interrupt request is acknowledged using the $\overline{INTA}$ pin. If the IF is not set, the interrupt request is ignored.

→ After interrupt is acknowledged, CPU computes the vector address from the type of interrupt, that may be passed to the interrupt structure of CPU internally or externally.

→ The contents of IP & CS are pushed

to the stack. The contents of IP & CS now point to the address of the next instruction of the main program from which the execution is to be continued after executing the ISR.

→ The control is transferred to interrupt service routine for serving the interrupting device. The new address of ISR is found from the interrupt vector table. The execution of ISR starts. If further interrupts are to be responded during the time the first interrupt is being serviced, the IF should again be set to 1.

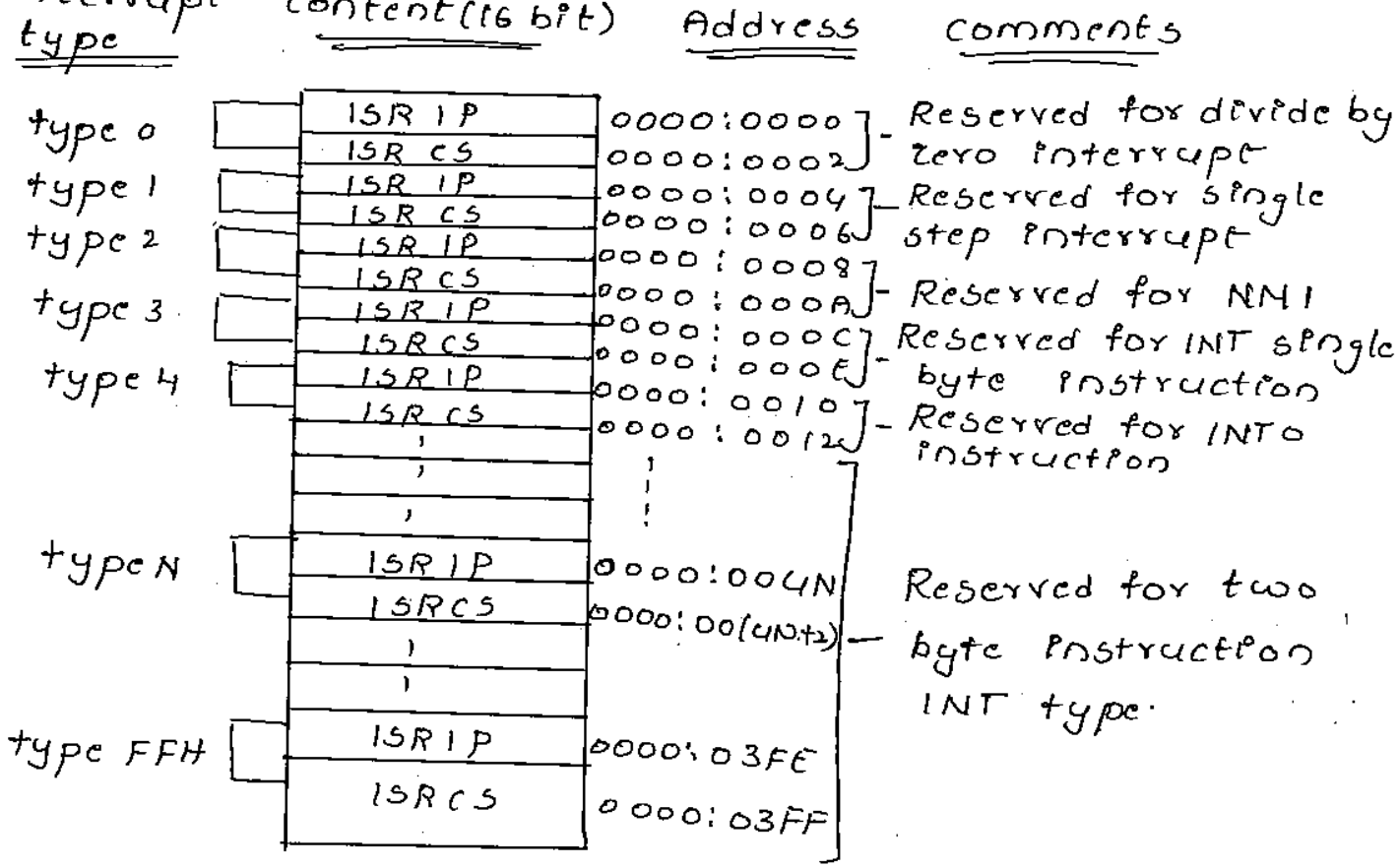
→ If interrupt flag is not set to 1, the subsequent interrupt signals will not be acknowledged by processor, till the current one is completed. The programmable interrupt controller is used for managing such multiple interrupts based on their priorities.

→ At the end of the ISR the last instruction should be IRET. when the CPU executes IRET, contents of flags, IP & CS which were saved previously are now retrieved to the respective registers. The execution continues from this address (received by IP & CS) onwards.

→ Every external & internal interrupt is assigned with a type (N), that is either implicit (in case of NMI, TRAP, divide by zero) or specified in the instruction INT N (in case of internal interruots). In case of external interrupts, the

type is passed to the processor by an external hardware like PIC.

→ In the zeroth segment of physical address space i.e CS:0000, INTEL has reserved 1024 locations for storing the interrupt vector table. The 8086 supports a total of 256 types of interrupts i.e from 00 to FFH. Each interrupts requires 4 bytes, [2 bytes by IP & 2 for CS of its ISR). Thus a total of 1024 bytes are required for 256 interrupts types. Hence interrupt vector table starts at location 0000:0000 & ends at 0000:03FFH



Non maskable interrupt:-

The processor has a NMI i/p pin, that has the highest priority among external interrupts

TRAP is an external interrupt having

except devide by zero exception.

When NMI is activated, the current instruction being executed is completed & then the NMI is served. In case of string type instructions, thus interrupt will be served only after complete string has been manipulated.

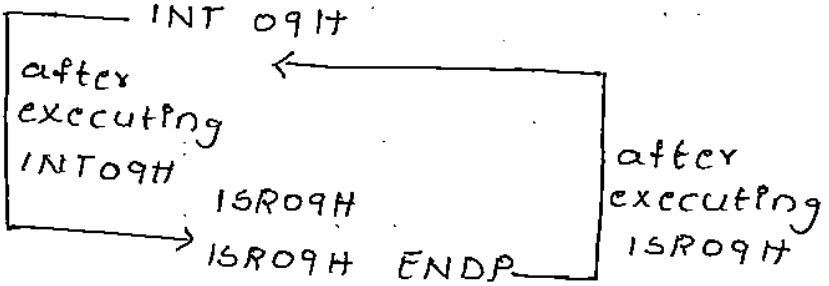
Maskable Interrupt (INTR):-

The processor also provides pin INTR, that has lowest priority as compared to NMI. Furthar priorites, with in the INTR types are decided by the type of INTR signal, that is to be passed to the processor via data bus by some external device like PIC. The INTR signal is level triggered & can be masked by resetting the interrupt flag.

If IF is reset, the processor will not serve any interrupt appearing at the pin. If the IF is set, processor is ready to respond to any INTR interrupt. Once the processor responds to an INTR signal, the IF automatically reset.

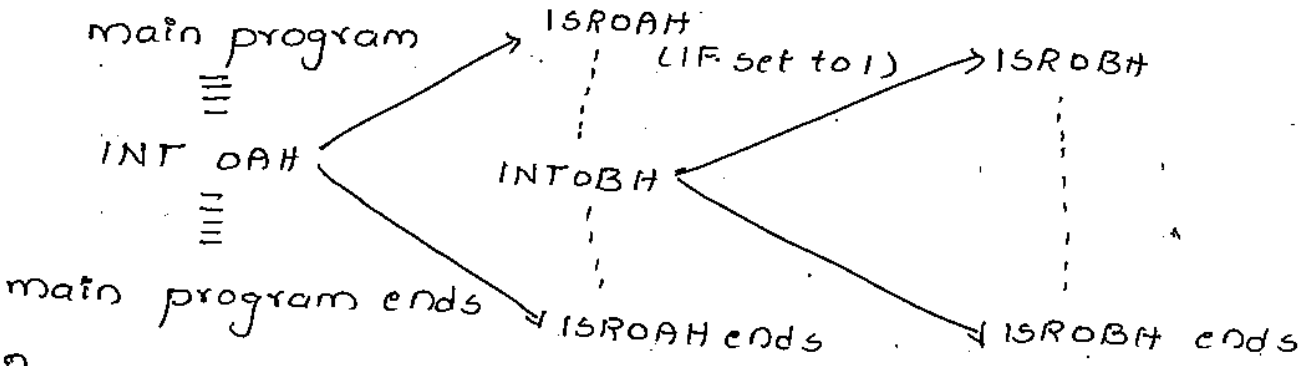
Transfer of control during exception of an interrupt service routine:-

```
Assume cs:code, DS:data
data segment
  ;
data ends
code segment
  ;
```



ISR09 → interrupt service routine for type 09.

Transfer a control for nested interrupt:-

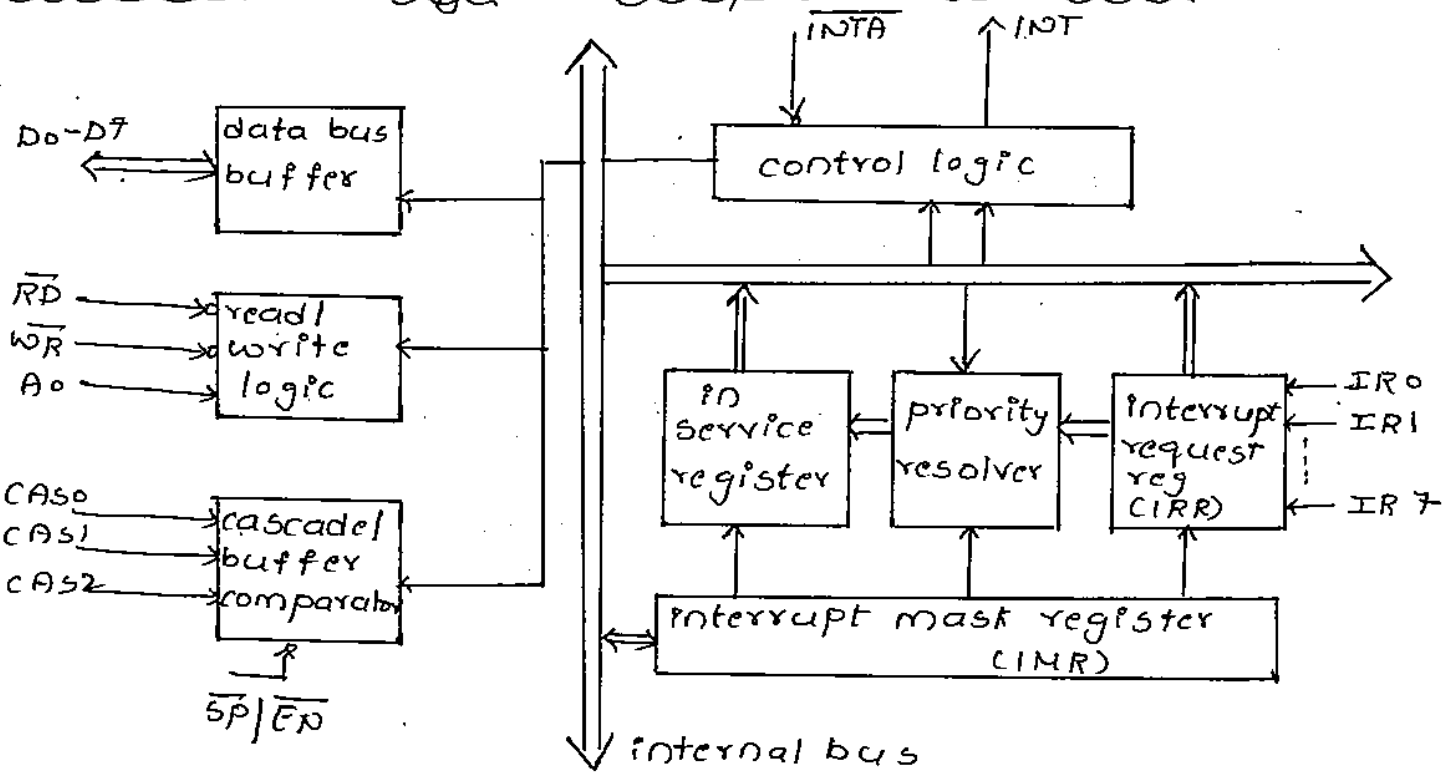


Programmable interrupt controller (8259A):-

In multiple interrupt systems, processor will have to take care of priorities for the interrupts, simultaneously occuring at the interrupt request pin. To overcome this difficulty, we require a PIC which is able to handle the multiple interrupts at a time. This controller takes care of number of simultaneously appearing interrupt requests along with their types & priorities. This will retrieve the processor from all these tasks.

8259 was designed to operate only with the 8 bit processor like 8085. A modified version 8259A was later introduced that is used to operate with 8 bit as well as 16 bit processors.

Architecture & signal descriptions of 8059A:-



Interrupt request register (IRR):-

IRR stores all the interrupts requests in it in order to serve them one by one on the priority basis

In Service Register (ISR):-

This stores all the interrupt requests those are being served i.e ISR keeps a track of requests being served.

Priority resolver:- This unit determines priorities of interrupt requests appearing simultaneously. The highest priority is selected & stored into corresponding bit of ISR during $\overline{INTA}$ pulse.

The IR0 has highest priority while IR7 has lowest one, normally in fixed priority mode. The priorites however may be altered by programmin

Interrupt mask register:- This stores the bits required to mask the interrupt inputs. IMR operates on IRR at direction of priority resolver.

Control logic:- This block manages the interrupt & interrupt ack signals to be sent to the CPU for serving one of the eight interrupt requests.

This also accepts the interrupt ack signal from CPU that causes 8259A releases the vector address on data bus.

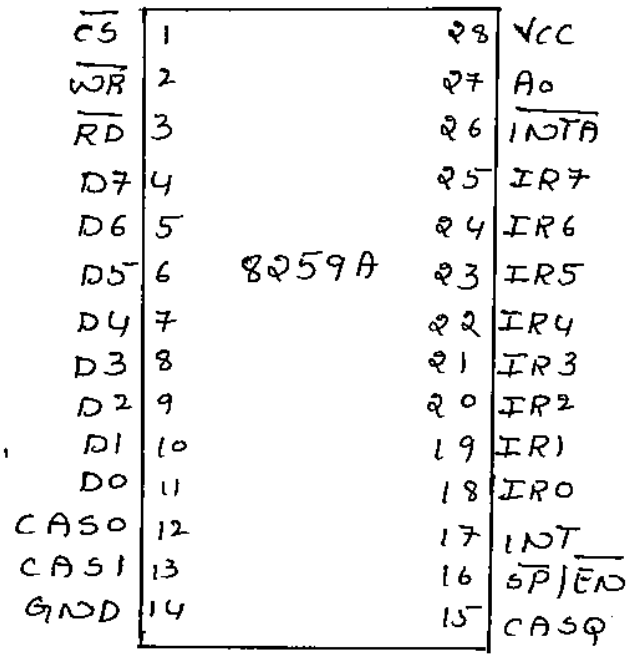
Data bus buffer:- This buffer interfaces internal 8259A bus to the microprocessor system bus. Control words, status & vector information pass through data buffer during read/write operations.

Read/write control logic:- This ckt accepts & decodes commands from CPU & allows transferring of status of 8259A on to the data bus.

Cascade buffer/comparator:- This block stores and compares ID's of all the 8259A's used in system. The three pins CAS0-2 are o/ps when the 8259A used as master & pins acts as i/p's. when the 8259A is in slave mode. The 8259A sends the ID of interruption slave device on data bus in master mode.

In slave mode, it will send its pre-programmed vector address on the data bus.

Pin diagram of 8259A:-



Interrupt sequence in an 8086 system:-

1) One or more IR lines are raised high that set corresponding IRR bits.

2) 8259A resolves priority & sends an INT signal to CPU.

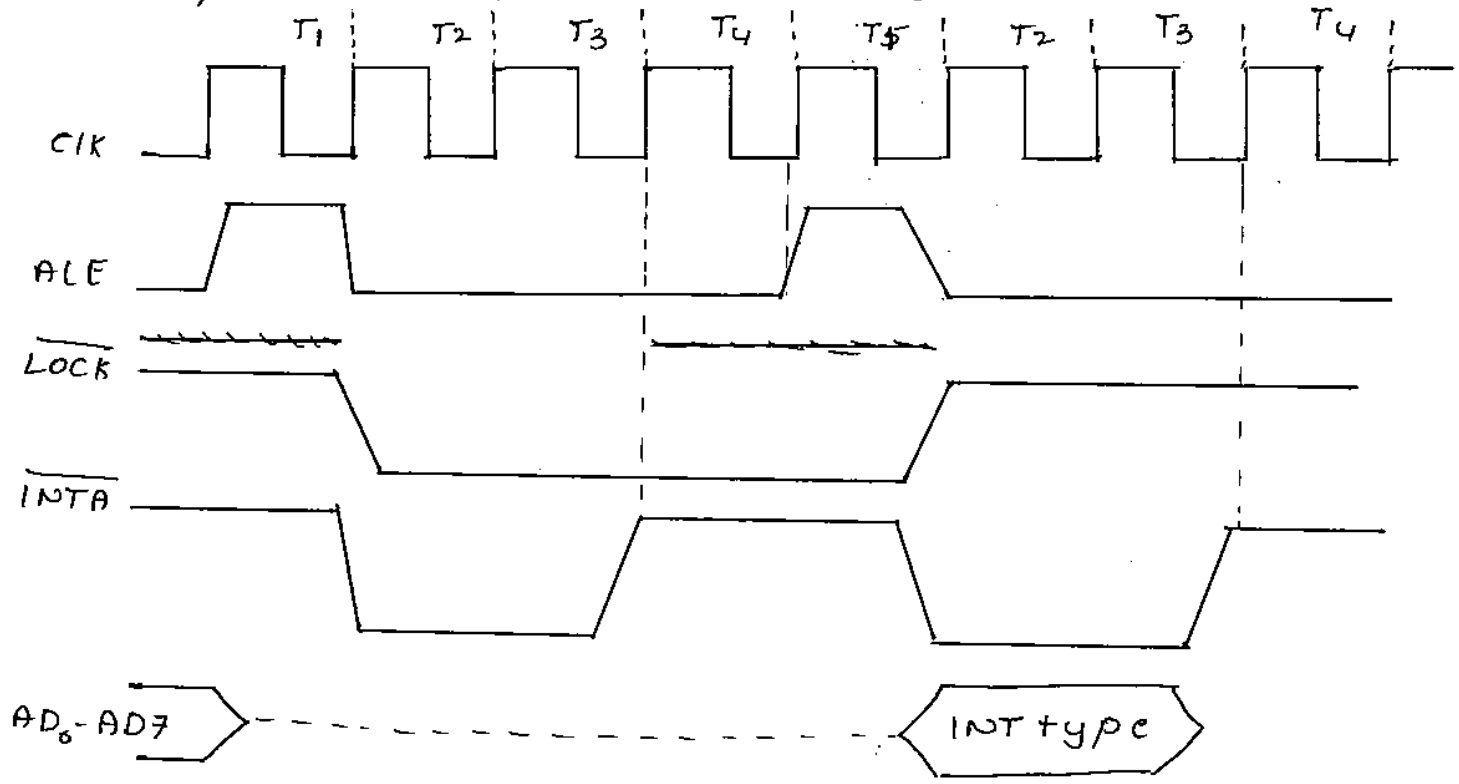
3) The CPU ack with $\overline{INTA}$ pulse.

4) Then highest priority ISR bit is set & the corresponding IRR bit is reset. The 8259A does not drive data bus during this period.

5) 8086 will initiate second $\overline{INTA}$ pulse. During this period of 8259A releases.

6) This completes interrupt cycle. The ISR bit is reset at the end of second $\overline{INTA}$ pulse if automatic end of interrupt mode is programmed [EOI]. Otherwise ISR bit remains set until an appropriate EOI command is issued at the end of interrupt sub routine.

Interrupt ack sequence of 8086:-



Command words of 8259A:-

Command words are classified into two groups.

(1) initialization commond word (icws) &

2) operation command words (ocws)

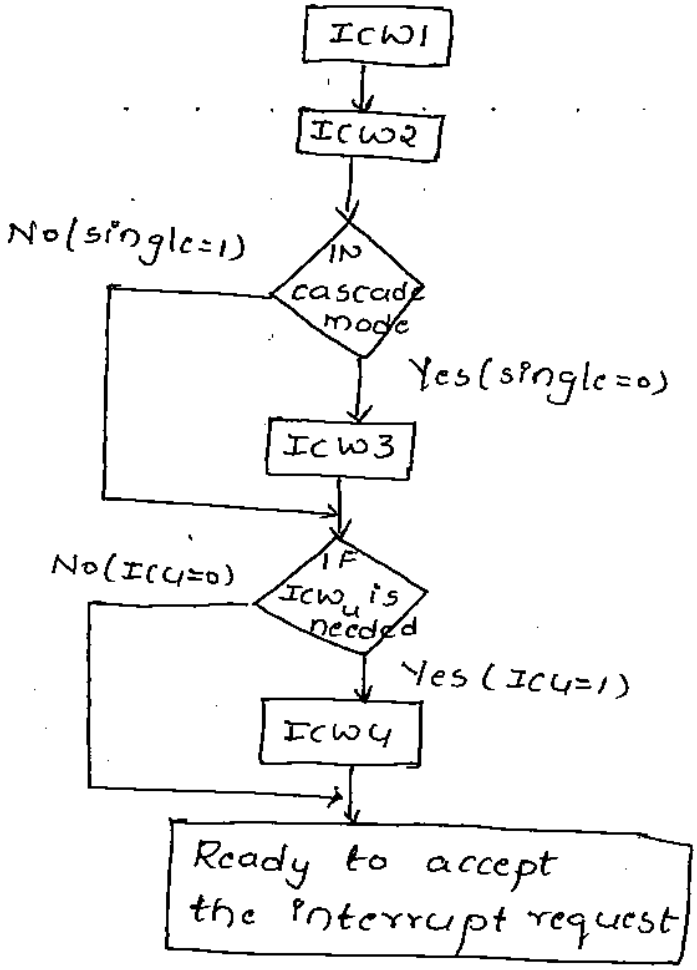
1. Initialization command words (icws):-

Before it starts functioning, 8259A must be initialized by writing 2 to 4 command words into respective command word registers. These are called as initialization command words (icws).

If $A_0=0$ & $D_4=1$, control word is recognized as ICW1. It contains the control bits for edge/level triggered mode, single/cascade mode, call address interval & whether ICW4 is required or not etc.

If $A_0=1$, the control word is recognized as ICW2. The ICW2 stores the details regarding interrupt vector address.

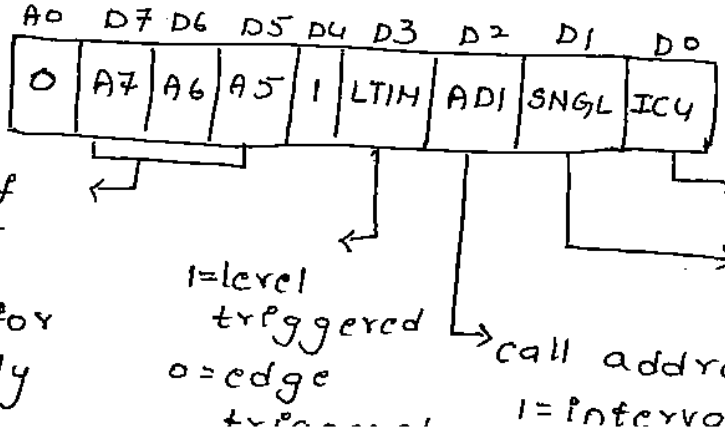
The initialization sequence of 8259A described



Once ICW1 is loaded, following initialization procedure is carried out internally.

1) The edge sense circuit is reset.
2) IMR is cleared.
3) IR7 i/p is assigned to lowest priority
4) Slave mode address is set to 7.
5) Special mask mode is cleared
6) If IC4=0, all functions of ICW4 are set to '0'

ICW1



ICW2:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	T7	T6	T5	T4	T3	A10	A9	A8

T7-T3 are A4-A0 of interrupt vector address
A10-A8 selected a/c to interrupt request level.

Note:- ICW1 & ICW2 are compulsory command word in initialization sequence of 8259A while ICW3 & ICW4 are optional.

When cascading is used, ICW3 is used. The single bit in ICW1 indicates whether 8259A is in cascade mode or not.

In master mode the 8 bit slave register will be set bitwise to '1' for each slave in the system.

Master:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	S7	S6	S5	S4	S3	S2	S1	S0

S_n=1, IR_n i/p has a slave
=0, IR_n i/p doesnot have a slave

Slave:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	ID2	ID1	ID0

In slave mode bits D2 to D1, identify the slave i.e 000 to 111 for slave 1 to slave 8

$D_2 D_1 D_0$ = 000-111 for slave 1 to slave 8

ICW4:- The use of command word depends on IC4 bit of ICW1. If IC4=1, ICW_4 is used, otherwise it is neglected.

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	SFNM	BUF	M/S	AEOI	uP

SFNM → special fully nested mode is selected if SFNM=1

BUF:- =1 the buffered mode is selected, In the

M/S:- If M/S=1 8259A is master, if M/S=0 8259A is slave. If BUF=0, M/S bit is to be neglected.

AEOI:- If AEOI=1, automatic end of interrupt mode is selected

uPM:- If the uPM=0, the MCS-85 system operation is selected & if uPM=1, 8086/88 operation is selected

Operational command words:- Once 8259A is initialized, it is ready for its normal function i.e for accepting the interrupts but 8259A has its own ways of handling the received interrupts called as modes of operation.

These modes of operation can be selected by programming i.e writing three internal reg called as operation command word registers. The data written into them is called as operation command words. These ocw are OCW1, OCW2, OCW3

OCW1 is used to mask the unwanted interrupts. If mask bit is 1 corresponding interrupt request is masked & if it is '0' the request is enabled.

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

1-mask set
0-mask reset

OCW2:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	R	SL	EOI	0	0	L2	L1	L0

EOI → 0 0 1 non specific EOI command
0 1 1 specific EOI command

Automatic rotation
1 0 1 rotate on non specific EOI command
1 0 0 rotate on sp automatic EOI command(sa
0 0 0 " " " " /clear

Specific rotation
1 1 1 rotate on specific EOI command
1 1 0 set priority command (Here L_2-L_0 are used)
0 1 0 No operation.

Thus three bits L_2, L_1, L_0 in OCW2 determine the interrupt level to be selected for operation if SL bit is active

L_2	L_1	L_0	IR no
0	0	0	0
0	0	1	1
	⋮		
1	1	1	7

In OCW3, if the ESMM bit (enable special mask mode) is set to 1, SMM bit is enabled to select or mask the special mask mode. When ESMM bit is 0, the SMM is neglected.

If ESMM=1 & SMM=0 the 8259A will return to normal mask mode

OCW3:-

A0								
0	0	ESMM	SMM	0	1	1	RR	RIS

1 = poll command
0 = no poll command.

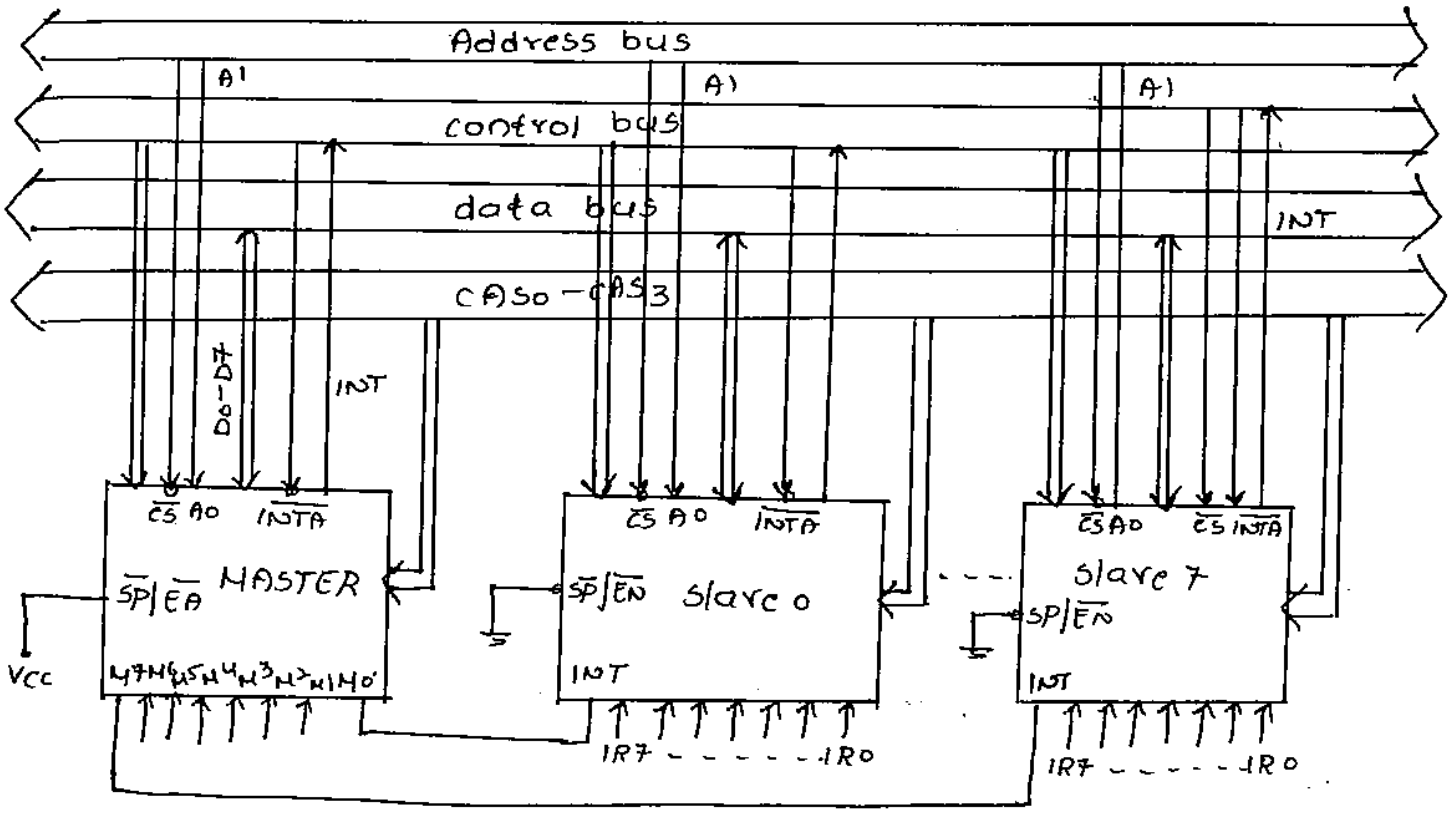
Poll command:-

In polled mode of operation, the INT o/p of 8259A is neglected, though it normally functions, by not connecting INT o/p or by masking INT i/p of up

Note:- Using $L_2L_1L_0$ by the bottom priority level can be selected. The selected bottom

the other priorities. If IR5 is selected as bottom priority, then IR5 will have lowest priority & IR4 will have the next highest priority. Thus IR6 has highest priority

8259A in cascade mode:-

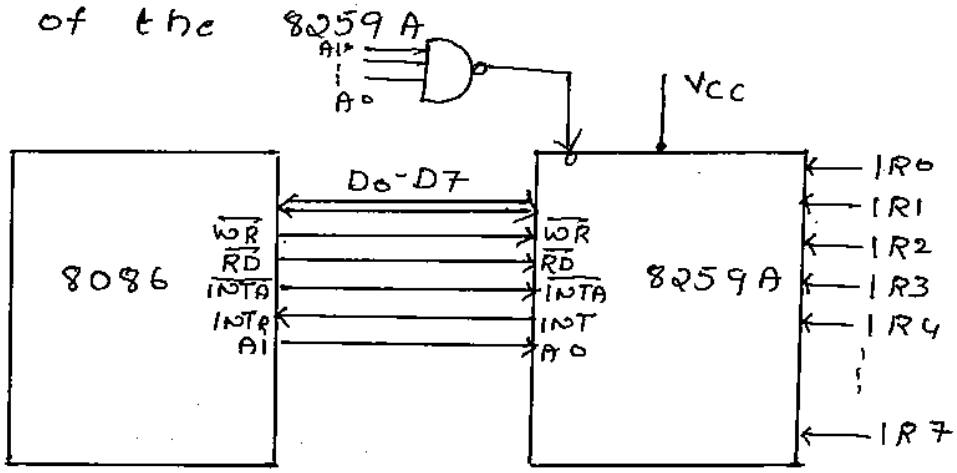


Interfacing & programming 8259:-

8) ⟶ Show 8259A interfacing connections with 8086 at the address 074X. Write an ALP to initialize the 8259A in single level triggered mode, with call address interval of 4, non buffered, no special fully nested mode. Then set the 8259A to operate with IR6 masked, IR4 as bottom priority level with rotate on specific EOI mode. Set special mask mode of 8259A. Read IRR & ISR into registers BH & BL respectively.

Sol) Let the starting vector address is 0000:0010. The inter connections of 8259A with 8086 are shown in fig.

8259 is interfaced with lower byte of 8086. Data bus, hence A0 line of microprocessor system is abondened & A1 of microprocessor system is connected with A0 of the 8259A



ICW1:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	1	1	1	1

= 1F0H

↓ dont care for 8086

ICW2:- vector address 0000:0010 for IR3

T7	T6	T5	T4	T3	A10	A9	A8
1	0	0	0	0	0	1	1

= 83H

There is no slave hence ICW3 is given below

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0	0

= 00H

ICW4:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	0	1

= 01 H

OCW1:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	0	0	0	0

= 40 H

OCW2:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
	1	1	1	0	0	1	0	0

= E4H

↓ Specific EOI command with rotating priority

bottom priority level set at IR4

OCW3:-

for reading IRR:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	0	1	0	1	0

Read IRR

OCW_3 = 6AH

for reading ISR:-

A0	D7	D6	D5	D4	D3	D2	D1	D0
	0	1	1	0	1	0	1	1

Read ISR

```
code segment
assume cs:code
start: mov  AL,1FH      } set 8259A is single, level
       mov  DX,0740H    } triggered mode with call
       out  DX,AL       } address of interval of 4
       mov  DX,0720H    select the vector address
       mov  AL,83H      0010H for IR3H
       out  DX,AL
       mov  AL,01H ; ICW4 for 8086 system,normal
       out  DX,AL   EOI, non buffered, SFNM masked
       mov  AL,40
       out  DX,AL
       mov  AL,0EUH
       mov  DX,0742H
       out  DX,AL
       mov  AL,6AH
       out  DX,AL   write OCW3 for reading IRR &
       IN   AL,DX   store in BH
       mov  AL,6BH
       out  DX,AL   write OCW3 to read IBR & store
       IN   AL,DX                    in AL
       mov  BL,AL
       INT  21H
code ends
end start
```

Serial Communication

Explain Serial data Transfer Schemes.

The data with in the microcomputer is transferred in parallel, as it is the fastest way to transfer. In parallel data transfer 'n' lines required to transfer 'n' bits at a time. While in the case of long distance Communication, parallel data transfer is not suitable. It becomes a costliest way. So Serial data transfer is preferred.

In Serial data Communication only one bit can be sent at a time. Hence it needs only one transmission path. As serial data transfer sends only one bit at a time, it is some what slower but cheaper in the case of long distance Communication.

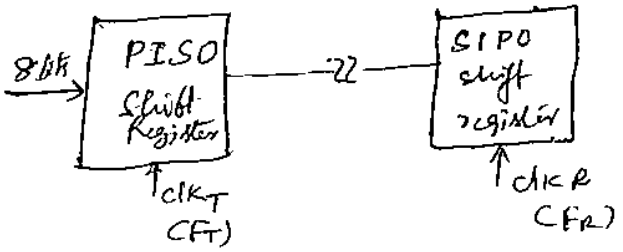
Another advantage of serial data transfer is, it imposes less overhead compared to parallel data transfer.

Types of Serial data transfer

There are two types of serial data transfer.

(1) Synchronous serial data transfer (2) Asynchronous Serial data transfer

(1) Synchronous Serial data transfer:



If the operating frequency of transmitter & receiver is exactly same ($F_T = F_R$), then it is called Synchronous serial data transfer.

If there is slight difference in frequencies then there will be error in data transmission.

Ex: Data transfer b/w processor & CRT terminal, Keyboard, printer.

(2) Asynchronous Serial data transfer:

In this type of data transmission there is slight difference in the frequency of transmitter & receiver which can be adjusted by using Stop bits after each Char. is transferred.

Ex: Data transfer b/w processor and modem, audio cassette.

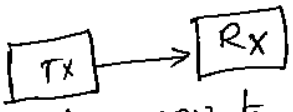
Depending upon the direction of the data transfer serial data transfer is divided into three types.

(1) Simplex Data transfer (2) Half Duplex data transfer

(3) Full Duplex Data transfer.

(1) Simplex data transfer

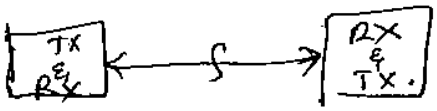
If the data can be transferred only in one direction from transmitter to receiver is called simplex data transfer



Ex: Data transfer from cpu to printer.

(2) Half Duplex Data transfer

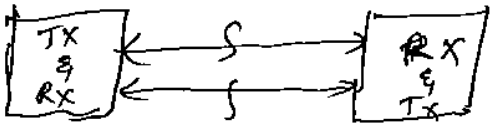
If the data can be transferred in both the directions but there is only one Transmission line. Such type of data transfer is called half duplex data transfer



Ex: wireless system (or) Walkie talkie

(3) Full Duplex Data transfer

If the data can be transferred in both the directions simultaneously, so there are two different Transmission lines.



Ex : Telephone.

Serial data transfer

Serial data communication is usually opted in digital data communication because the cabling required for transmission & reception is less.

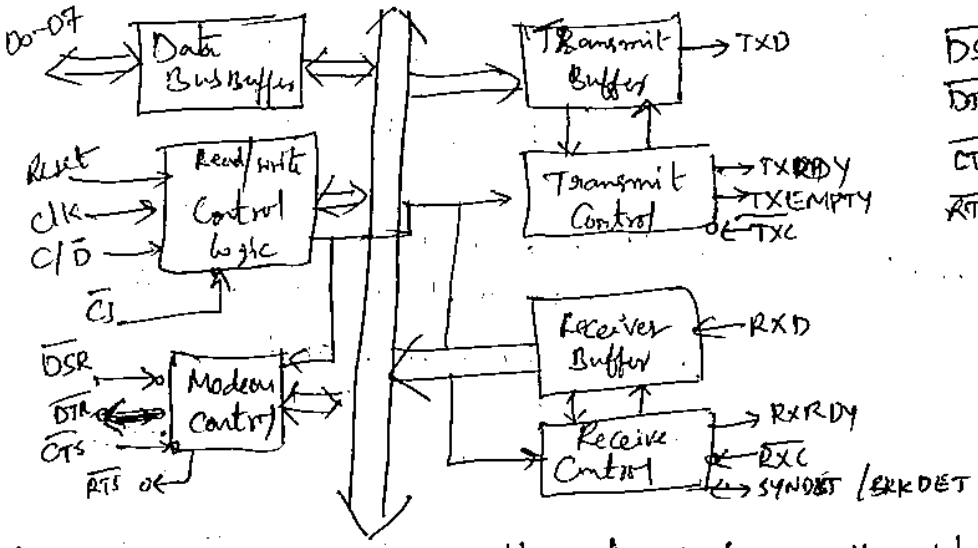
In parallel communication a word (or) byte is transferred simultaneously so number of transfers requires no. of parallel wires.

- Cost for serial communication is less compared to parallel communication
- Serial communication is usually used for long communication. & parallel communication is for short distance communication.
- Transmission problems are less in serial communication.
- parallel communication is used only with in CPU. But Serial Communication is used by most of the transmitters such as UART, USART etc.

→ Diff. b/w synchronous & Asynchronous serial data transmission

Synchronous	Asynchronous
(1) If there is a slight difference in freq then there will be error in data transmission	(1) In this type of data transfer if there is a slight difference in freq of transmitter & receiver it can adjusted by using the Stop bits after each char. is transferred
(2) This type of data transfer is used when the processor & IO devices match speed	(2) This type of data transfer is used when the processor & IO device doesn't match in speed.
(3) In this, whether the IO devices is ready for data transfer or not is not checked	(3) Here it is checked by the CPU before transferring data
(4) Here IO devices are All ways ready	(4) Here IO devices are not always ready
(5) It is faster	(5) It is slower
(6) Here detects the errors in received data.	(6) There is no detection.
(7) Ex: Data transfer b/w processor &	(7) Ex: Data transfer b/w processor & modem, audio cassets.

Architecture of 8251



$\overline{DSR}$ – Data Set Ready
$\overline{DTR}$ – Data terminal ready
$\overline{CTS}$ – Clear to Send
$\overline{RTS}$ – Request to Send.

→ The Data Bus Buffer Interfaces the internal bus of the ckt with System Data bus.

→ The read write Control logic unit Controls the Operation of the peripheral depending upon the operations initiated by the CPU.

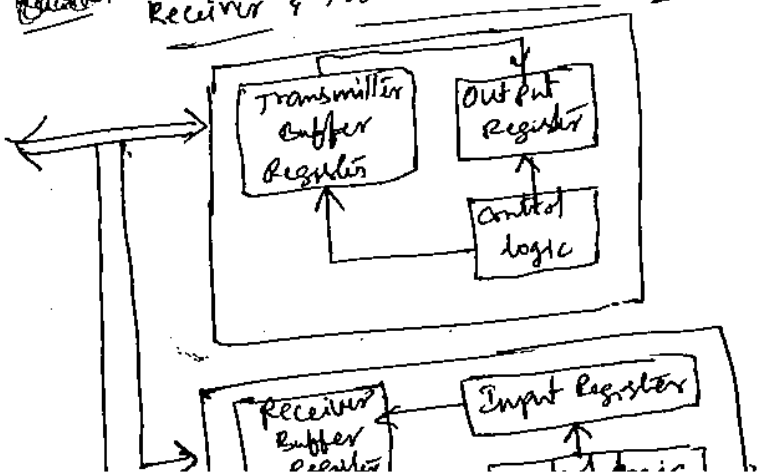
→ The modem Control unit handles the modem handshake signals to Coordinate the Communication b/w the modem & the USART.

→ The transmit Control unit transmits the data byte received by the data buffer from the CPU for further Serial Communication. This decides the transmission rate which is controlled by the $\overline{TXC}$ input frequency. This unit also derives two transmitter status signals namely TXRDY, TXEMPTY.

The transmit Buffer is parallel to serial Converter that receives a parallel byte for conversion into a serial signal.

→ The receive Control unit decides the receiver freq as Controlled by the $\overline{RXC}$ i/p frequency. This unit generates receiver ready (RXRDY) signal that may be used by the CPU for handshaking.

Receiver & Transmitter Block diagram



Serial data Transmission standards & their specifications

Serial data transmission standards are

(1) RS232C (2) RS 422 (3) RS-485

RS 232-C

Used in serial data communication, Its application is peripheral Connectivity for pcs.

It is used to transfer the data in a range upto 50 feets.

Maximum data rate 20 Kbps.

No. of Drivers = 1

Receivers = 1

Max. loaded driver O/p voltage levels = ±5V.

Driver load impedance = 3 to 7K

Receiver i/p impedance = 3 to 7K.

RS 422

It is specially designed to overcome the distance & speed limitations of RS232. & it accomodates multiple receivers.

Max. cable length = 4000 feet.

Drivers = 1

Receivers = 10

Max. data rate = 10mbps down to 100kbps

o/p voltage level = ±2V.

Driver load impedance = 100Ω

Receiver i/p impedance = 4K.

RS-485

It has same distance coverage & speed as RS 422.

But it allows multiple drivers on the same bus.

Max. cable length = 4000 feet

Drivers = 32

Receivers = 32

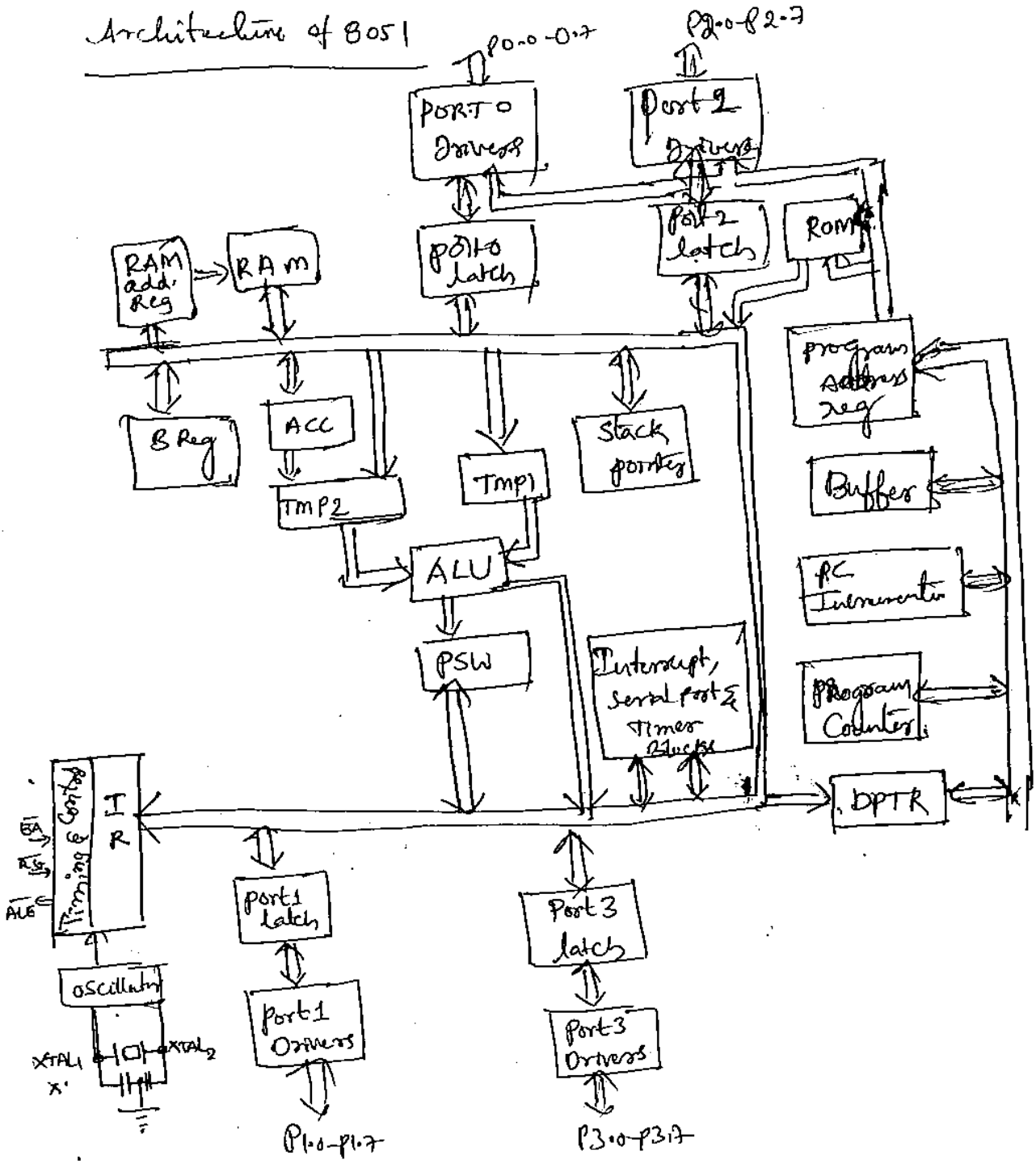
max. data rate = 10Mb/s down to 100kbps

O/p voltage levels = ±1.5V

Driver load impedance = 54Ω

Receiver i/p impedance = 12K.

Architecture of 8051



Accumulator (ACC): The ACC (or) 'A' acts as an operand register. This is either be implicit or specified in the instruction.

The ACC Register address is allotted in the on-chip special function register bank.

B Register: It is used to store one of the operands for multiply & Divide ~~operat~~ Instructions.