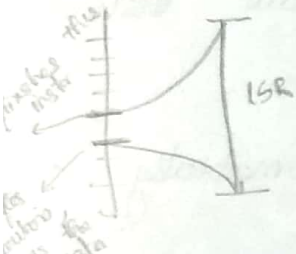
(MM) MICROPROCESSORS AND MICROCONTROLLERS

MODULE - 3

8086 Interrupts

An interrupt is a special condition that arises during the working of a microprocessor. The microprocessor services the interrupts by executing a Interrupt Service Routine (ISR).

There are 3 sources of interrupts in 8086.



1. Hardware Interrupts (External Interrupts):

These interrupts occur as signals on the external pins of the microprocessor. 8086 has two pins to accept hardware interrupts.

(i) NMI (Non-Maskable Interrupt)

(ii) INTR

2. Software Interrupts (Special Instructions):

These interrupts are caused by writing

Scanned by CamScanner

the software interrupt instructions, INTn where 'n' can be any value from 0 to 255 (00H to FFH). Hence, all 256 interrupts can be invoked by software.

3. Internally Generated Interrupts (Conditions produced by the pgm):

8086 is interrupted when some special condition occurs while executing certain instructions in the pgm.

Eg:- An error in division automatically calls the INT0 interrupt.

Interrupt Vector Table (IVT)

IVT stores the address of the ISR (or the interrupt pgm)

The IVT contains ISR address for the 256 interrupts. Each ISR address is stored as CS and IP.

As each ISR address is of 4 bytes (2 - CS (base - higher & lower) and 2 - IP (offset - higher & lower)).

Each ISR address requires 4 locations to be stored.

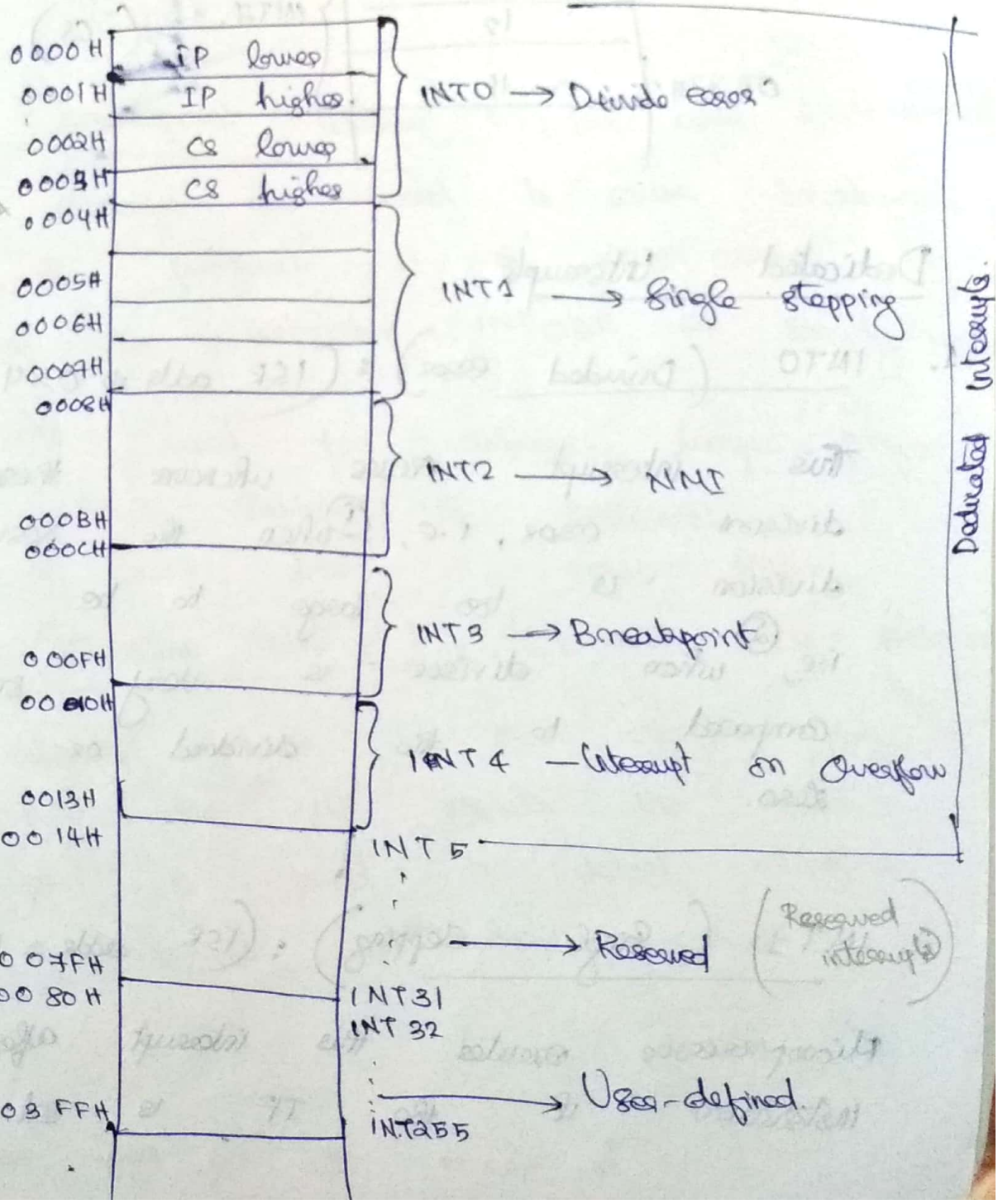
There are 256 interrupts : INT0 → INT255, therefore, total size of the IVT is

Scanned by CamScanner

256 x 4 = 1 KB.

The first 1 KB of the memory address 00000H to 003FFH are reserved for the INT.

Whenever an interrupt INTn occurs, microprocessor does n x 4 to get values of IP and CS from the IVT and hence, performs the ISR.

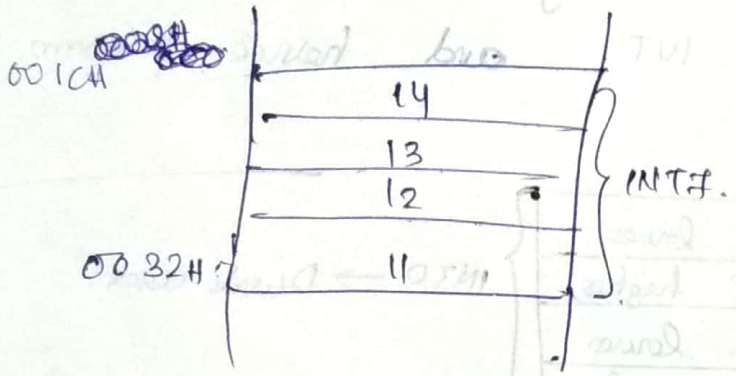


Scanned by CamScanner

Q. The starting address for a type 7 interrupt service procedure is 1112 : 1314. Show where & in what order this address should be placed in the 8086 IVT.

Interrupt 7 ⇒ INT 7

∴ Addr ⇒ 4 × 7 = 28 ⇒ 001CH



1112 : 1314

1112 ↓ base (CS)

1314 ↓ offset (IP)

Dedicated Interrupts

1. INT0 (Divided error) : (ISR addr ⇒ 0 × 4 = 0000H (offset) in IVT)

This interrupt occurs whenever there is division error, i.e, ① when the result of a division is too large to be stored.

② i.e, when divisor is very small as compared to the dividend or divisor is zero.

2. INT 1 (Single stepping) : (ISR addr ⇒ 1 × 4 = 0001H in IVT)

Microprocessor executes this interrupt after every instruction if the TF is set.

to pause after a single instruction

Scanned by CamScanner

It puts microprocessor in single stepping mode; i.e, the microprocessor pauses after executing every instruction.

3. INT 2 (Non Maskable Interrupt) : (ISR addr : 2×4 = 0008H)

The microprocessor executes this ISR in response to an interrupt on the NMI line.

4. INT 3 (Breakpoint Interrupt) : (ISR addr : 3×4 = 000CH)

This interrupt is used to cause breakpoints in the program. It is caused by using the instruction INT 03H or simply INT.

It is used for debugging large pgm whose single stepping is inefficient.
to break from a large pgm.

5. INT 4 (Overflow Interrupt) : (ISR addr : 4×4 = 0010H)

This interrupt occurs if the overflow flag is set and the MP executes the INTO instr. It is used to detect overflow error in signed arithmetic operation.

Scanned by CamScanner

6. INT5 — INT31 (Reserved):

These levels are reserved by INTEL to be used in higher processors like 80386, pentium, etc. They are not available to the users.

Hardware Interrupts

1. NMI (Non-Maskable Interrupt): This is a non-maskable, edge triggered, high priority interrupt. On receiving an interrupt on NMI line, the µp executes INT2. The IF has no effect on this interrupt.

2. INTR: This is a maskable, level triggered, low priority interrupt.

On receiving an interrupt on INTR line, the µp executes 2 $\overline{INTA}$ pulses.

1st $\overline{INTA}$ pulse: the interrupting device acknowledges (prepares to send) the vector number.

2nd $\overline{INTA}$ pulse: the interrupting device sends the vector number 'n' to the µp.

Scanned by CamScanner

Now µp multiplies n x 4 and goes to the corresponding location in the IVT to obtain the ISR address.

INTR is maskable by using IF flag

→ INTR is a maskable interrupt.

A is masked by making IF = 0 by software through CLI instruction. (clear interrupt flag)

It is unmasked by making IF = 1 by software through STI instruction. (set interrupt flag)

Response to Any Interrupt (Interrupt Cycle)

→ the information regarding the currently executing program

1. The µp will PUSH flag register into the stack.

SS : [SP − 1] (FH higher), SS [SP − 2] (FL lower) ← flag.

SP ← SP − 2

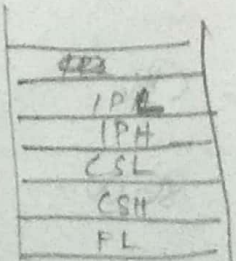
2. Clears IF and TF in the flag register and thus disables INTR interrupt.

IF ← 0 , TF ← 0.

3. PUSH CS into the stack. (stores the base addr of the currently working prgm)

SS : [SP − 1] (CSH), SS : [SP − 2] (CSL) ← CS

SP ← SP − 2



Scanned by CamScanner

4. PUSH IP into the stack. (stores the offset of the currently working pgm)

SS: [SP − 1] (IPH), SS: [SP − 2] (IPL) ← IP

SP ← SP − 2

5. Load new IP from the IVT

IP ← [n × 4], [n × 4 + 1]

(loading the IP & CS of the requested pgm to the CS & IP)

6. Load new CS from the IVT

CS ← [n × 4 + 2], [n × 4 + 3]

Since CS and IP get new values, control shifts to the address of the ISR and continues to execute ISR till the execution of IRET instr and returns to the main pgm in the following steps.

Response to IRET instruction

1. The MP will restore IP from the stack.

IP ← SS: [SP] (IPL), SS: [SP + 1] (IPH)

SP ← SP + 2

(the values in the stack are popped)

2. The MP will restore CS from the stack.

CS ← SS: [SP] (CSL), SS: [SP + 1] (CSH)

Scanned by CamScanner

SP ← SP + 2.

(iii) The MP will restore contents FLAG register from the stack flags.

Flag ← SS: [SP]^{FL} , SS: [SP+1]^{FH}

(ISR)

SP ← SP + 2.

Interrupt Priorities

Interrupt	Priority. Simultaneous Occurrence (If all the interrupts occur simultaneously, the order of priority)	To interrupt another ISR
Divide Error, INTn	1 (Highest)	Can interrupt any ISR.
NMI	2 "	Can ~~interrupt~~ any ISR.
INTR	3	Cannot interrupt an ISR
Single Stepping	4 (Lowest)	Cannot interrupt an ISR.

1. Simultaneous Occurrence : When more than one interrupt occur simultaneously then, all s/w interrupts except Single stepping, get the highest priority

Scanned by CamScanner

2. Ability to interrupt another ISR

Since software interrupts (INTn) are non-masked they can interrupt any ISR.

NMI is also non-maskable hence it can also interrupt another ISR as both are disabled before ip enters an ISR by

IF ← 0 and TF ← 0

// TF → Trap flag.

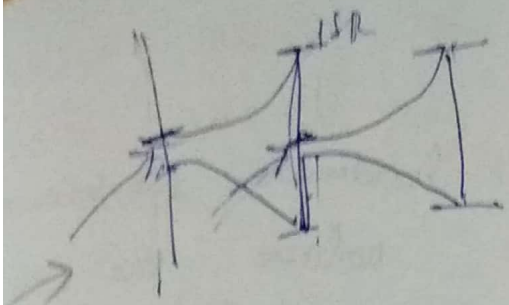
Interrupt Service Routine (ISR)

An interrupt service routine (ISR) is a software that invokes hardware in response to an interrupt. After the execution of ISR, the main program continues its execution further from the point at which it was interrupted.

Interrupt Programming

While programming for any type of interrupt, the programmer must, either externally or

Scanned by CamScanner



through the program, set the interrupt vector table for that type preferably with the CS and IP addresses of the interrupt service routine.

eg:- Execution sequence in case of a software interrupt.

```
ASSUME : CS : CODE , DS : DATA.
DATA SEGMENT
        :
DATA ENDS
CODE SEGMENT
        :
INT 09H
ISR09H  PROC.
        :
ISR09H
ENDP
CODE ENDS
END.
```

the procedure of ISR calling in a prgm.

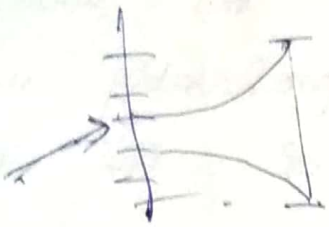
After executing ISR09H

ISR09H → Interrupt service routine for TYPE 09H

Scanned by CamScanner

Interrupt and Polling

Interrupt : Interrupt is a hardware mechanism in which the device notices the CPU that it requires its attention.



Polling : In polling, it is not a hardware mechanism.

It is a protocol in which CPU steadily checks whether the device needs attention.

Difference b/w Interrupt & Polling

Interrupt Method	Polling Method
(i) In interrupt, the device notices the CPU that it requires its attention.	In polling, CPU steadily checks whether the device needs attention.
(ii) An interrupt is not a protocol; it is a hardware mechanism.	It is a protocol & not a hardware mechanism.

Scanned by CamScanner

(iii) In interrupt, the device is serviced by interrupt handler.

In polling, the device is serviced by CPU.

(iv) Interrupt can take place at any time.

CPU steadily ballots the device at regular or proper intervals.

(v) In interrupt, request line is used as indication for indicating that device requires servicing.

In polling, command ready bit is used as an indication for indicating that a device requires servicing.

(vi) processor time is not wasted.

countless processor cycles are wasted.

8259

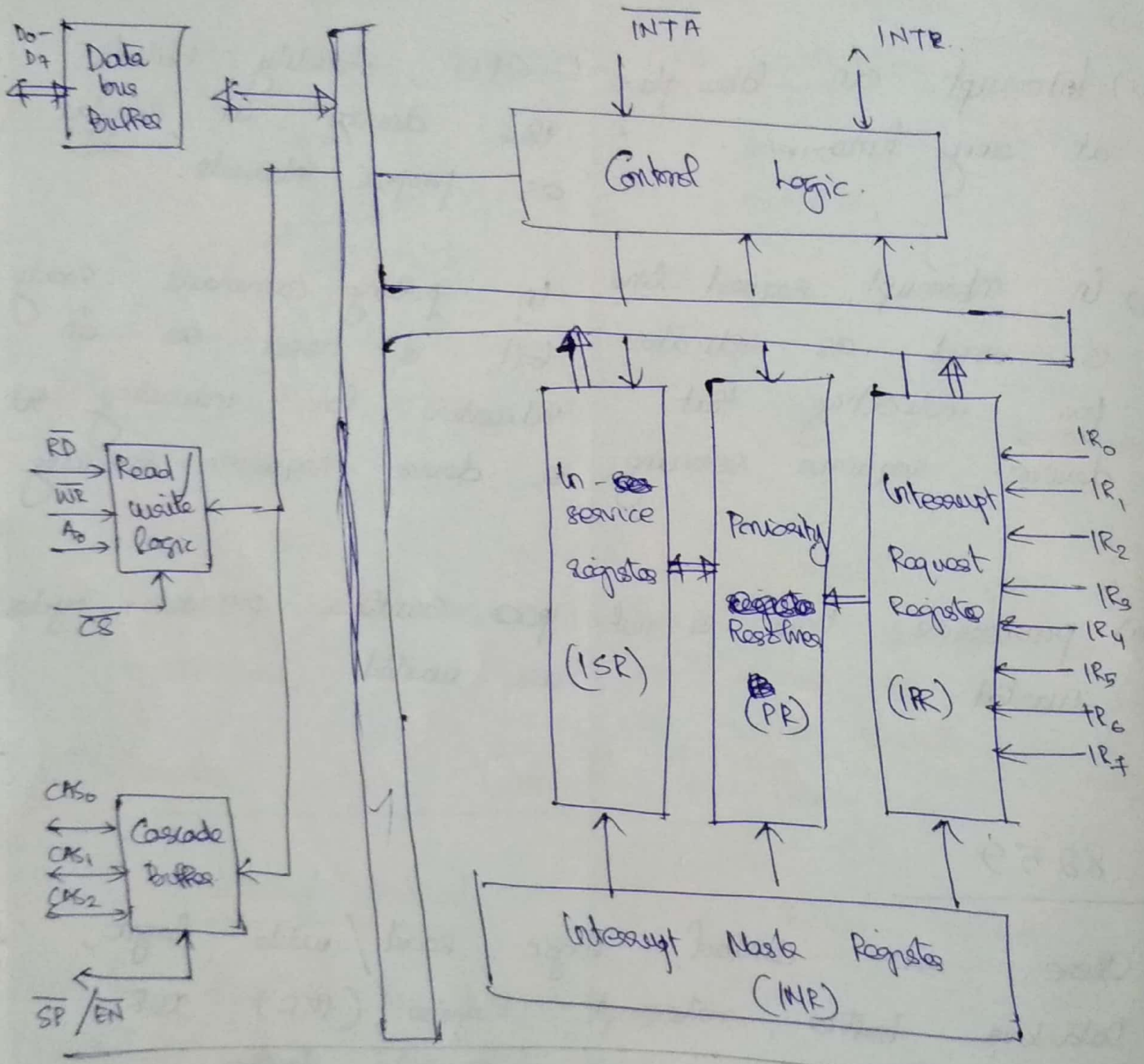
These are control logic, read/write logic, Data bus buffer, interrupt Register (IRR), ISR, IMR and PR & cascade buffer.

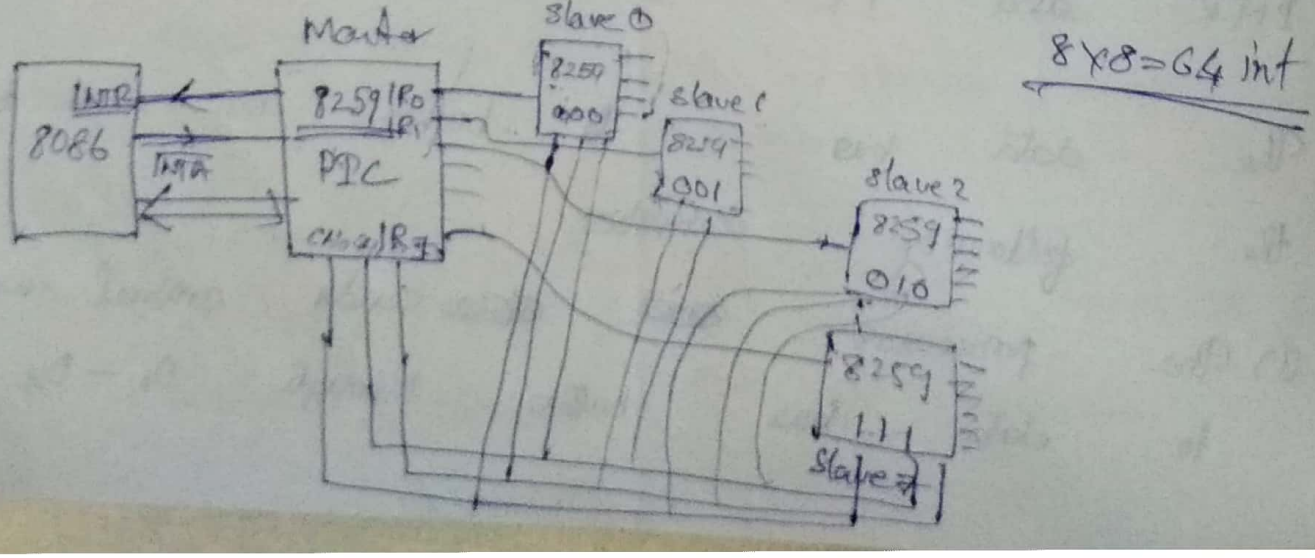
The data bus & its buffer are used in the following activities:

(i) The processor sends control word to data bus buffer through $D_0 - D_7$.

Scanned by CamScanner

Block diagram of 8259 Programmable Interrupt Controller - PIC





8 x 8 = 64 int

Scanned by CamScanner

8259 is a chip that helps in the proper management of interrupts to 8086. Multiple 8259 are cascaded to select multiple interrupts. 8259 determines the interrupt from the devices and passes it to 8086 for processing.

It has cascading lines to select the various 8259 connected.

Since there are only 8 interrupt pins, 3 bits are required to represent each slave 8259. (000 to 111)

(ii) The processor read status word from data bus buffer through $D_0 - D_7$.

(iii) From the data bus buffer, the 8259 send type no. or the opcode & address through $D_0 - D_7$ to the processor.

Interrupt Request Register (IRR):

The interrupts at the IR i/p lines are handled by interrupt request register internally. It stores all the interrupt request in it & arranges to serve them 1 by 1 on to the priority basis.

Priority Resolver (PR):

The unit determine the priorities of the interrupt request. In default case, the IR_0

Scanned by CamScanner

the highest priority while the IR_7 has the lowest.

In-Service Register (ISR):

It keeps a track of the request being served. That means this stores all the interrupt request those are being served.

Interrupt Mask Register (IMR):

The IMR stores the bits required to mask the interrupt lines to be masked.

Interrupt Control logic:

This block manage the interrupt & the interrupt acknowledgement signal is send to the CPU for serving 1 of the 8 interrupt request

Data Bus Buffer:

The buffer interface internal 8259 A bus to the µp data bus.

Read / Write Control logic:

This circuit accept & decode commands from the CPU. This block allow the status of the 8259A to be transferred

Scanned by CamScanner

or to the data bus. This block stores & compare the IDs of all 8259 A used in the system.

Cascade Buffer :

This helps to select the cascaded 8259 chips using the CAS lines.

Q. Write the conditions which cause 8086 to perform Type 1, Type 2 & Type 3 interrupts.
↓ single stepping ↓ NMI ↓ Breakpoint.

Q. Discuss 8086 interrupt acknowledgement cycle.

Q. Draw the architectural blk diagram of 8259 and explain each functional part.

Q. Give description of (i) Non-maskable (ii) maskable interrupts.

Q. Brief description of ISR.

Q. Disadvantages of polling over interrupt scheme.

Q. Explain NT with diagram.

Q. Explain dedicated interrupts of 8086.

Scanned by CamScanner