



PSG COLLEGE OF TECHNOLOGY, COIMBATORE - 641 002 F 88788

Form No.	Course of Instruction	Department/Section	Semester
222251	Polymer Physics	B.Tech	3
Branch & Semester	BB CE 63	3 Sem	III SEMESTER
Constitutes	192552	100	Mechanics and Metallurgy
Faculty Name	Designation	Section Date	Comments
1	1 3 4 9 21	44	✓
2	5 6 10 23	44	✓

### 1. c) Practical examples of I/O devices.

Mode 0 : devices like keyboard, mouse, etc.

Mode 1 : printer and related devices

Mode 2 : Network devices and advanced printers

BSR : Keyboard, mouse.

### Interrupt logic (Mode 1) :

Basically, there are two interrupts

1) BSR

2) I/O mode.

- └ Mode 0 - simple ready of all ports (A, B, C)
- └ Mode 1 → no interrupt logic
- └ Mode 2

(2 above) & when a stray signal is R.B.A to ready

In Mode 1,

⇒ Both the input & output are latched

⇒ It uses hand shaking mechanism with the help of

Port C [used for strobe signals]

⇒ Port A and B for inputs/outputs

⇒ It accepts interrupts

PART A → D<sub>6</sub> D<sub>5</sub>

0 1 → for mode 1

PART B → D<sub>2</sub>

1 (for setting to mode 1)

$$HFO : HSO = 91 : 83$$

# Motion Detection and Alerts

LPC2148

ROPROCESSORS A  
LABORATORY

ABINAYA B - 222  
AKSHARA P - 222  
GAYATHRI K S - 222  
INIYAN N - 222  
SNESHA B - 222

BACHELOR OF E

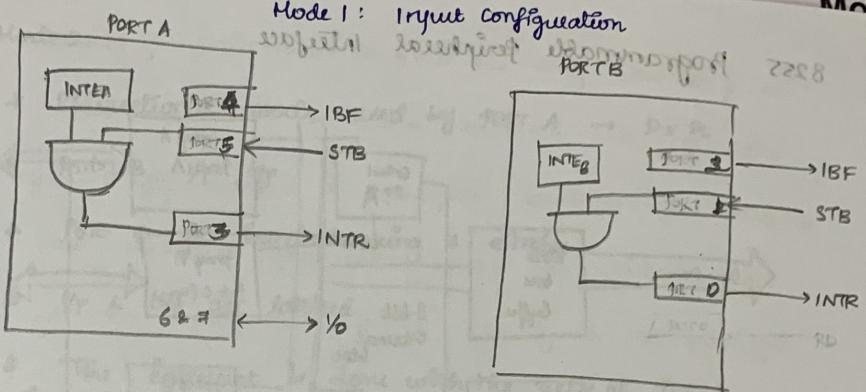


Date

MENT OF COMP  
PSG COLLEGE  
(Autono)

COIMB

## Mode 1: Input configuration



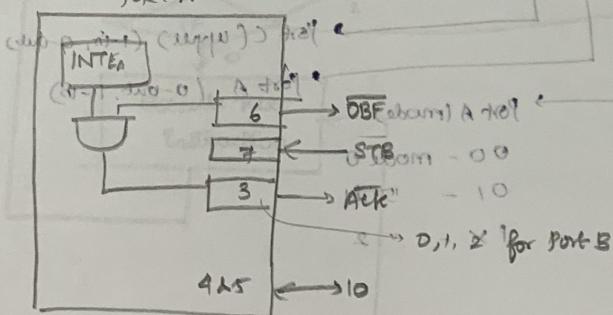
Status word:

Y0	Y0	STB	IBF	INTR	INTE	TMR	IBF
----	----	-----	-----	------	------	-----	-----

The STB signals to Port A and B are sent by PORT C as per interrupt logic.

Here, D6 and D7 are not used by Port A, which are left to connect with Y0.

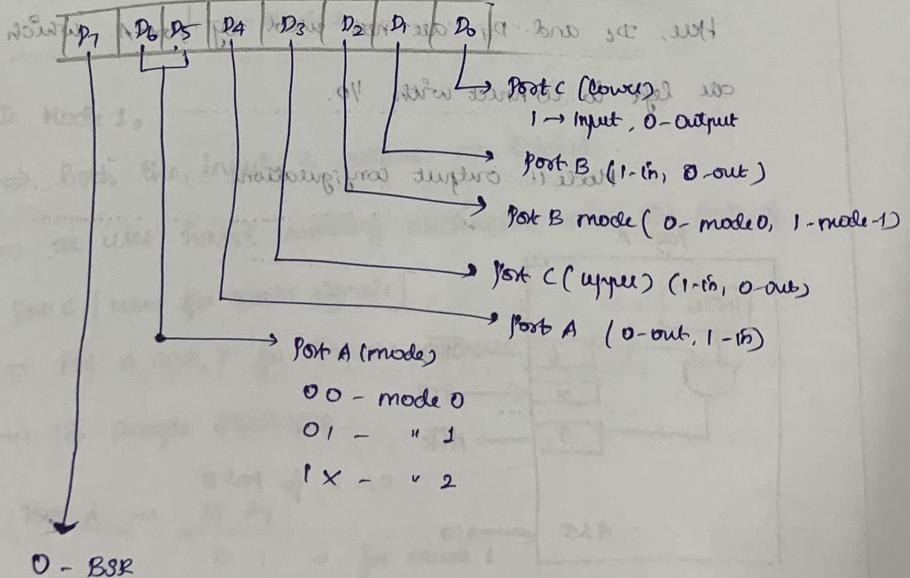
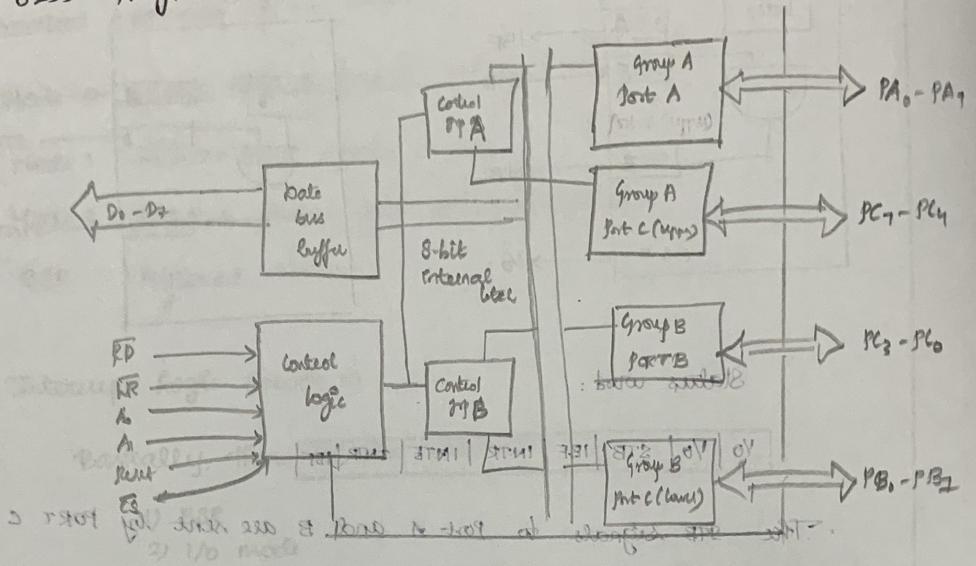
## Mode 2: Output configuration



Status word:

STB	OBF	Y0	Y0	INTR	INTE	OBF	INTR
-----	-----	----	----	------	------	-----	------

## 8255 Programmable Peripheral Interface

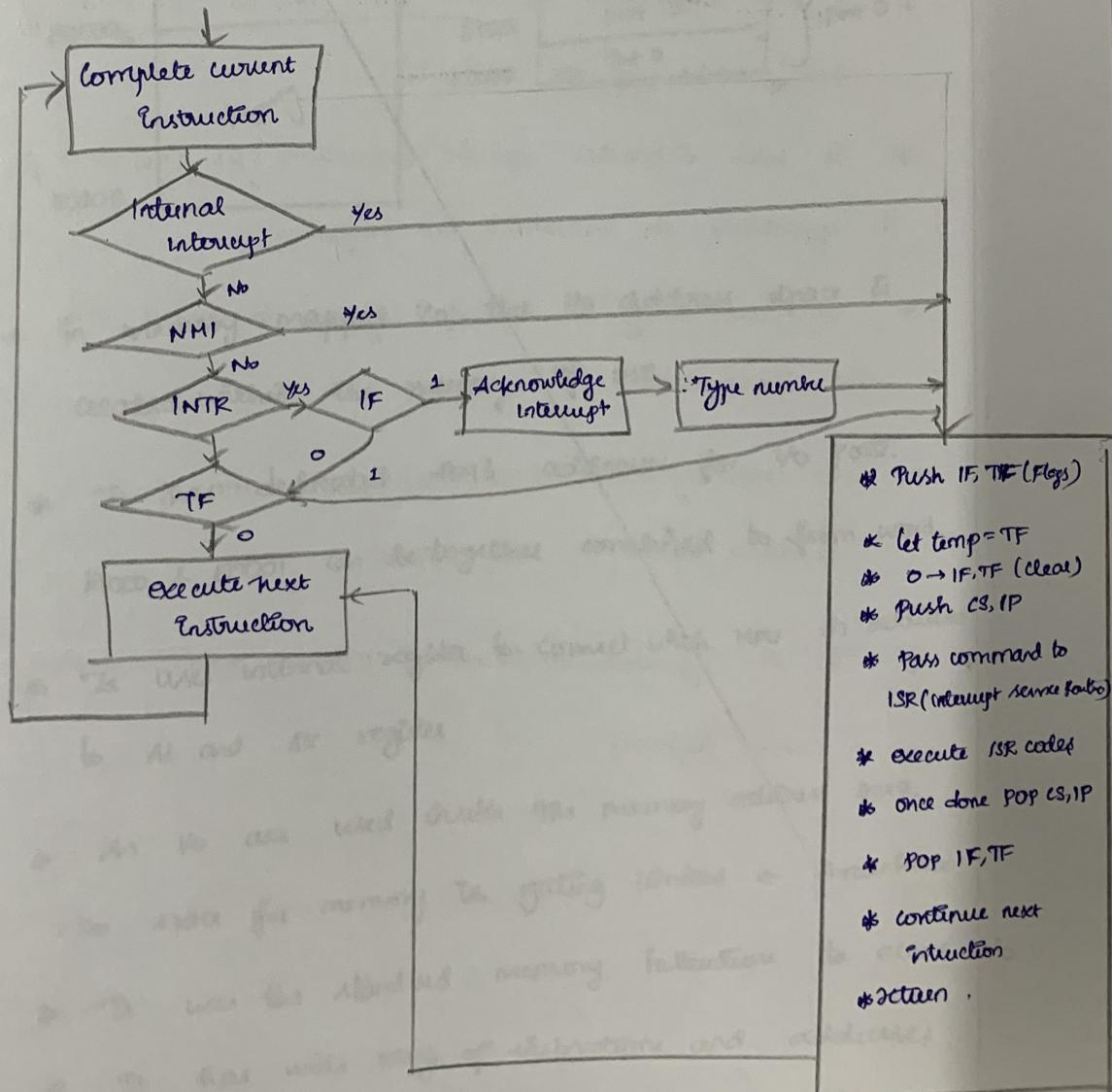


1 - BSR  
0 - I/O mode

Mode 2: for multi-tasking ext wrig, handshaking ext

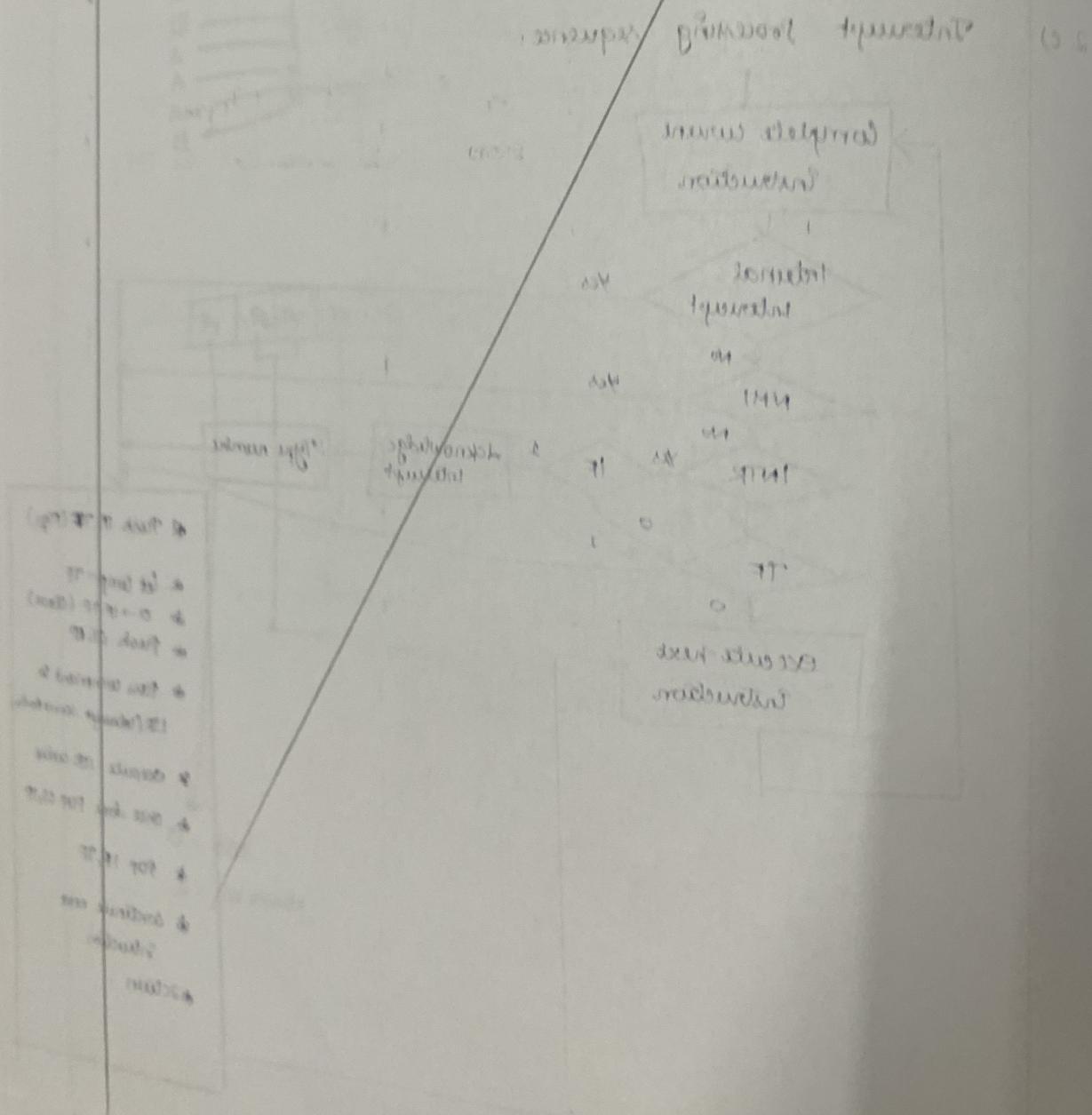
- \* Bidirectional mode used by PORT A  $\rightarrow$  D<sub>5</sub> D<sub>6</sub> x, y, z
- \* Port B for I/P, O/P & handshaking protocols to peripherals
- \* Port C for handshaking & strobes.
- \* Y<sub>P</sub> & O<sub>P</sub> are latched
- \* The interrupt is done with the help of PORT C

## 2. c) Interrupt processing sequence:



2 b D

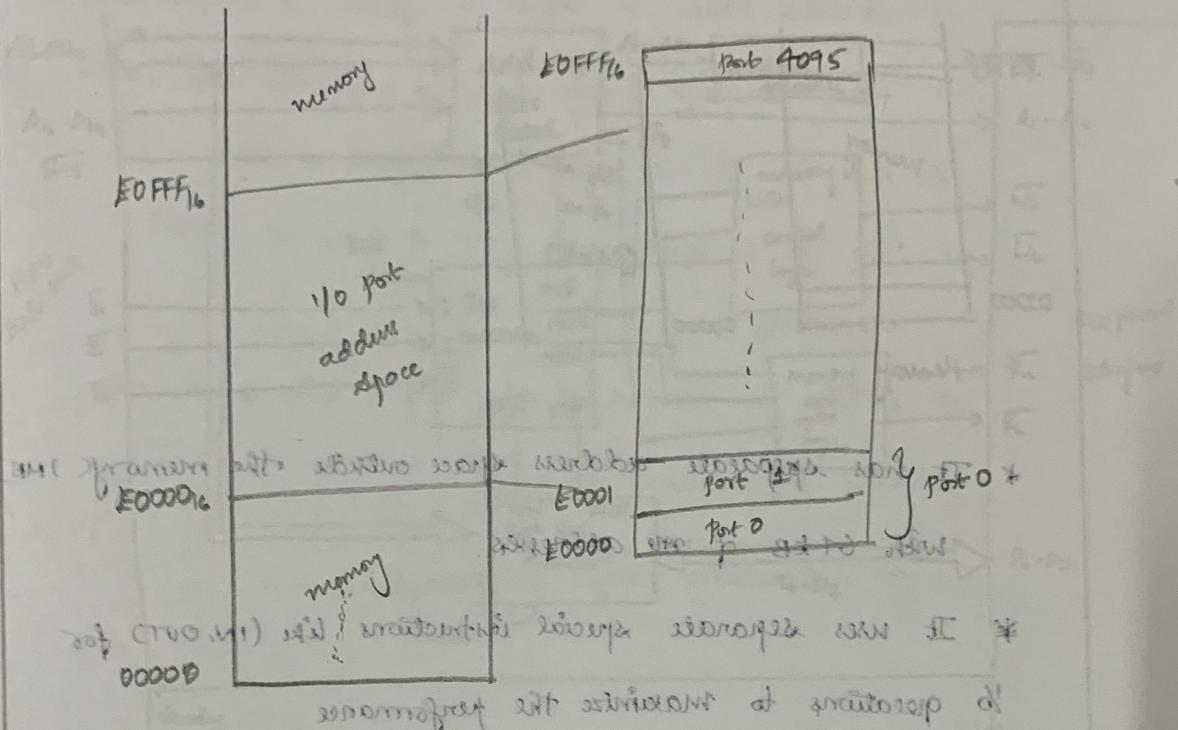
The flowchart gives the complete illustration of how  
interrupts are handled based on priority. The  
sequence of operations performed to execute one C interrupt  
service routine by pushing the flags, addresses, then to  
Memory ISR & finally restoring the pushed ones by popping  
them to get back into state of execution.



1 b)

## Memory Mapped I/O:

AV below



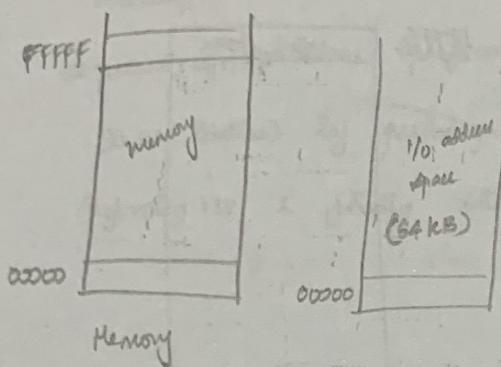
- \* In memory mapped I/O, the I/O address space is created within the memory of 1MB.
- \* It has dedicated 1096 addresses for I/O ports.
- \* ~~It uses internal register to connect with I/O~~ In addition to AL and AX register.
- \* As I/O are used inside the memory address space, the space for memory is getting limited ← drawback.
- \* It uses the standard memory instructions to access I/O.
- \* It has wide range of instructions and addresses.

Isolated I/O.

of separate format

Qd L

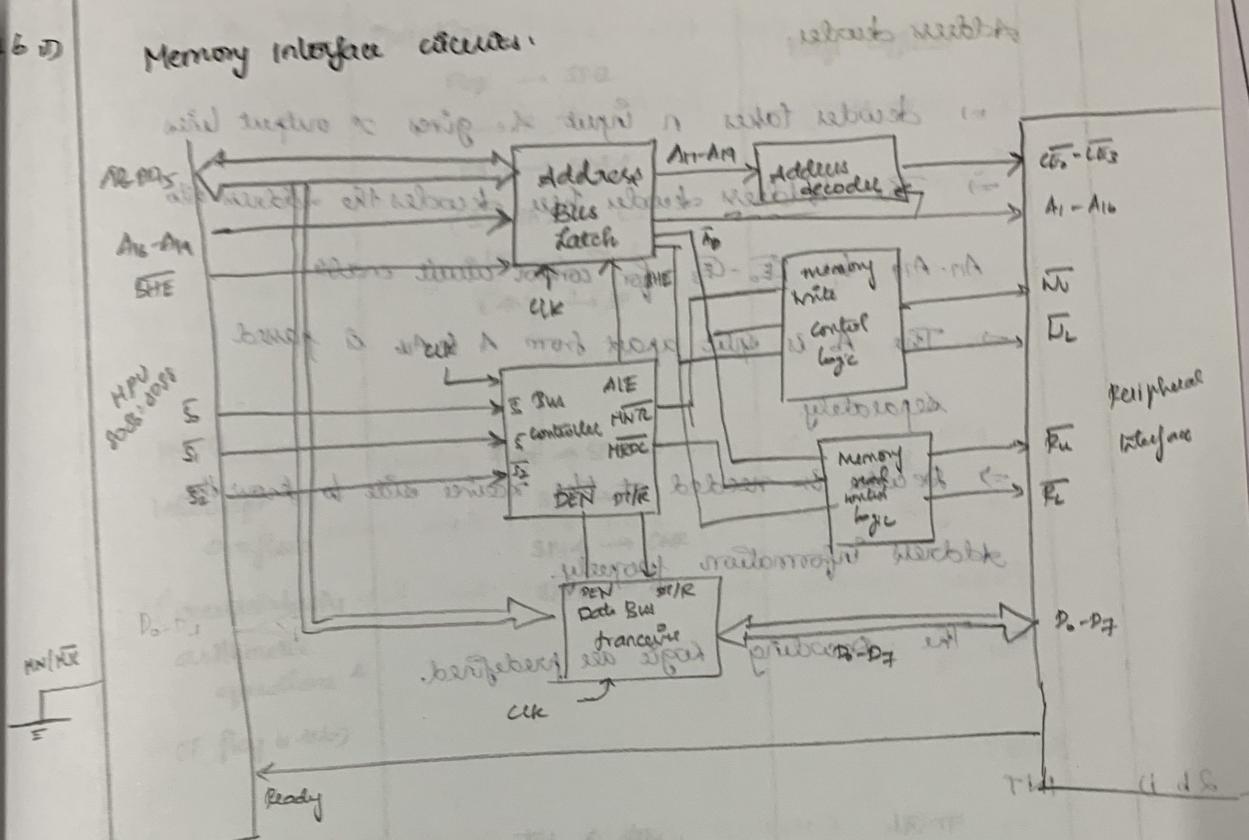
26 ID



- \* It has separate address space outside the memory (1MB) with 64 KB of addresses.
- \* It uses separate special instructions like (IN, OUT) for I/O operations to maximize the performance.
- \* It uses only AL and AX registers to connect with the microprocessor.
- \* The memory can utilize the whole 1MB of address space in this context.
- \* It is efficient in terms of memory address accessing & control signals.

36 (d)

### Memory Interface circuit



Address Bus Latch (with words size 8 or 16 bits)

=> It latches the addresses provided between AD<sub>0</sub>-AD<sub>15</sub> of 8086

are multiplexed with data

=> It latches the address separately only then the

addresses will not be lost due to some external / internal

factors.

=> It then passes the address A<sub>0</sub> to memory read & write

control logic & A<sub>1</sub>-A<sub>16</sub> to the interface, A<sub>1</sub>-A<sub>9</sub> to Address

decoder.

## Address decoder.

- ⇒ Decoder takes  $n$  input & gives  $2^n$  output lines
- ⇒ The Address decoder here decodes the address into  $A_1-A_{19}$  to  $\overline{CE}_0-\overline{CE}_3$  for control circuit enable.
- ⇒ The  $A_0$  is split apart from  $A_0$  to  $A_{16}$  is passed separately.
- ⇒ Decoder is needed at the receiver side to know the address information properly.
- ⇒ The decoding logic are predefined.

2b i)

## INT

- ⇒ Used by software interrupt
- ⇒ 0 to 255 (256 software interrupt) and NMI

⇒ predefined interrupt INT 50, port which leads

to the address  $(50 \times 4) \rightarrow 200$  word  
↳ 4 bytes each.

CS:IP  $\Rightarrow$  CA: C8  
Registers errors of sub field of our new memory

⇒ High priority interrupt than NMI & hardware.

Shows how to implement. the memory will be mapped next

⇒ INT 0 - divide interrupt -

Registers

2b ii)

## - Protection and Alert System Using

<b>INT n.</b> <i>(interrupts during program execution)</i> <b>Operations:</b> Flags → SP+2 $O \rightarrow IF, TF$ $CS \rightarrow SP+4$ $(2+4n) \rightarrow CS$ $IP \rightarrow SP+6$ $(4n) \rightarrow IP$ <i>(all registers are pushed onto stack)</i> <i>new IP = old IP + 2</i>	<b>Flags affected:</b> <i>All</i> $SP+2 \rightarrow IP$ $SP+4 \rightarrow CR$ <i>(stack pointer is popped back to IP)</i> <i>IP = new IP</i> <i>IF, TF = 0</i>
<b>INTO</b> <i>(interrupts during program execution)</i> <b>Interrupt on overflow</b> <i>(occurs during most arithmetic operations &amp; OF flag set)</i>	<i>Stack → SP → CS</i> <i>SP+2 → IP</i> <i>SP+4 → CR</i> <i>IP = new IP</i> <i>IF, TF = 0</i> <i>CR = 0</i> <i>IP = new IP</i> <i>IF, TF = 0</i>

EERING

- 2b) **NMI: (Non Maskable Interrupt)**
- ⇒ It is a kind of hardware interrupt but different from hardware interrupt in various aspects.
  - ⇒ It cannot be masked by software. Hardware interrupt uses INT# but NMI has separate NMI pin which works when logic 1 is up.
  - ⇒ It is high priority interrupt than hardware interrupt & it works without setting IF [interrupt flag]

## Motion Detection and Alert System Using 8051

- => It is used less than hardware interrupts as it  
is used only in emergency situation of urgent interrupt  
requirement.
- => It can be initiated at anytime by which it will  
not be masked by software.

=> NMI → 1 (to enable) If this interrupt is enabled then  
it first pushes all flags onto stack, clears interrupt  
pushes address and performs ISR then pops all the contents  
out of stack

### RESET:

- => Unlike hardware startup, most of the system is performed  
by hardware.  
This is a (warm-start) by software.
- => Once this reset is given by software, it then does

Flags → cleared (Programmed Selection not) : NMI (initially interrupt is asserted)  
CS → FFFFH (Initial value of program counter)  
SS, DS, ES → 0000H (Initial values of segment registers)  
Queue → emptied (Initial state of interrupt queue)  
Special registers not initialized (Initial state of special registers)

# Ion Detection and Alert System Using 1 PC2148

$AD_0 - AD_{15}, A_6 - A_9, BYE \rightarrow$  enters three state

$\overline{RD}_1, \overline{WR}, \overline{S_1 DIR}, \overline{S_1 DTIR}, \overline{S_2 TEST} \rightarrow$  goes to 1 then to idle

$Q_S, Q_D, HDA \rightarrow 0$

$\overline{RQ1GT}, \overline{RQ2GT} \rightarrow 1$

ALE  $\rightarrow 0$ .

