PSG COLLEGE OF TECHNOLOGY, COIMBATORE - [illegible] F [illegible]

1	4	2	4½	9	21	44
2	3	5	5	10	23	

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

Mode 0: devices like keyboard, mouse.

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 I/O by all ports (A, B, C)
   - Mode 1 → no interrupt logic
   - Mode 2

In Mode 1,

⇒ Both the input & output are latched

⇒ It uses hand shaking mechanism with the help of Port C [used for strobe signals]

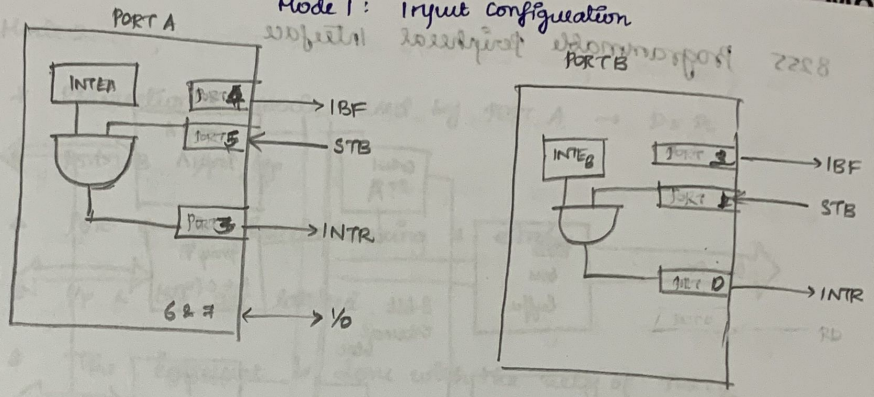
⇒ Port A and B for inputs / outputs

⇒ It accepts interrupts.

PORT A ⇒ D_6 D_5
0 1 ⇒ for mode 1

PORT B ⇒ D_2
1 (for setting to mode 1)

Mode 1 : Input configuration



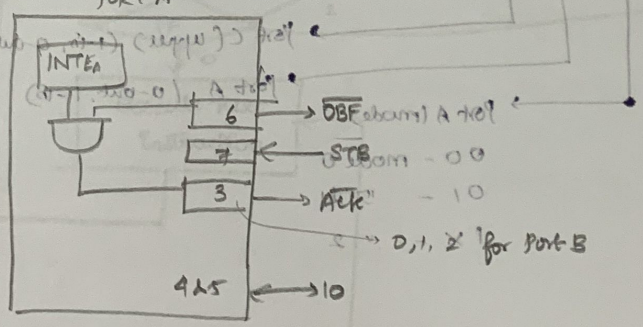
Status word :

I/O	I/O	STB	IBF	INTR	INTE	INTR	IBF

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

Here, D_6 and D_7 are not used by Port A. which are left to connect with I/O.

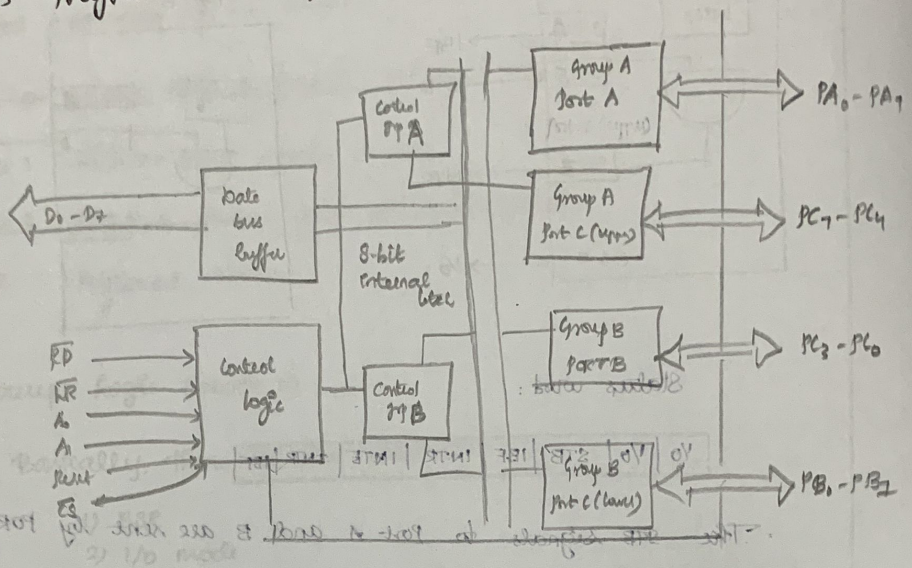
Mode 2 : Output configuration

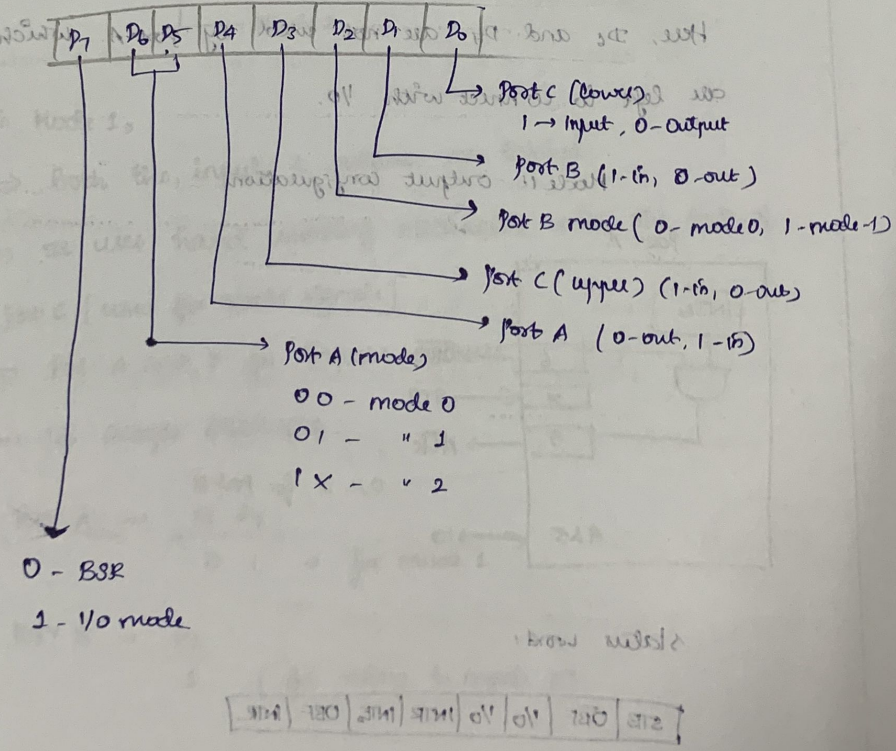


Status word :

STB	OBF	I/O	I/O	INTR	INTE	OBF	INTR

8255 Programmable Peripheral Interface

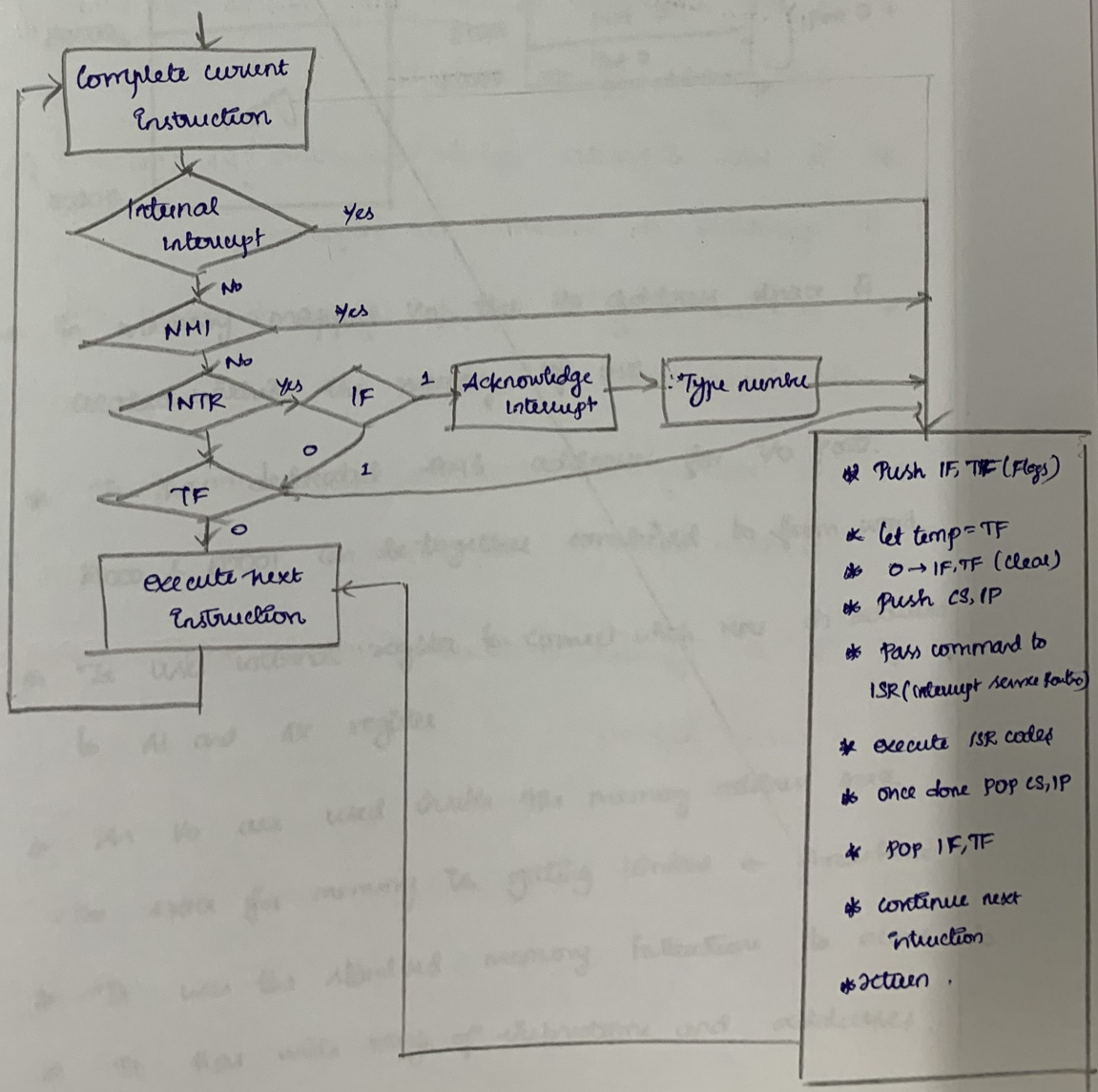




Mode 2:

* Bidirectional mode used by PORT A → D_5 D_6 × 1
* Port B i/p, o/p
* Port C for handshaking & strobes.
* i/p & o/p are latched.
* The interrupt is done with the help of PORT C

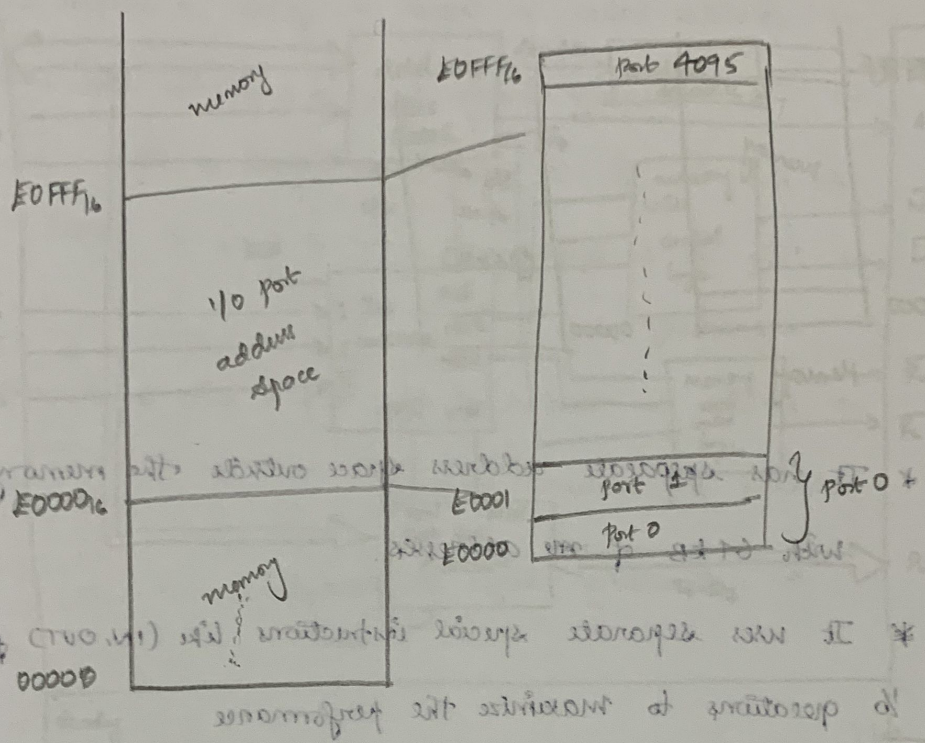
2. c) Interrupt processing sequence:



The flowchart gives the complete illustration of how interrupts are handled based on priority. The sequence of operations performed to execute ISR (interrupt service routine) by pushing the flags, addresses. Then performing ISR & finally storing the pushed ones by popping

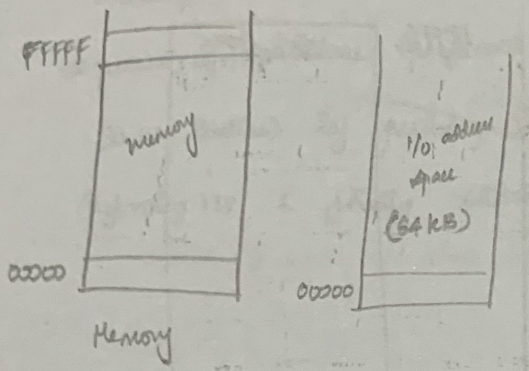
1 b;)

Memory Mapped I/O:



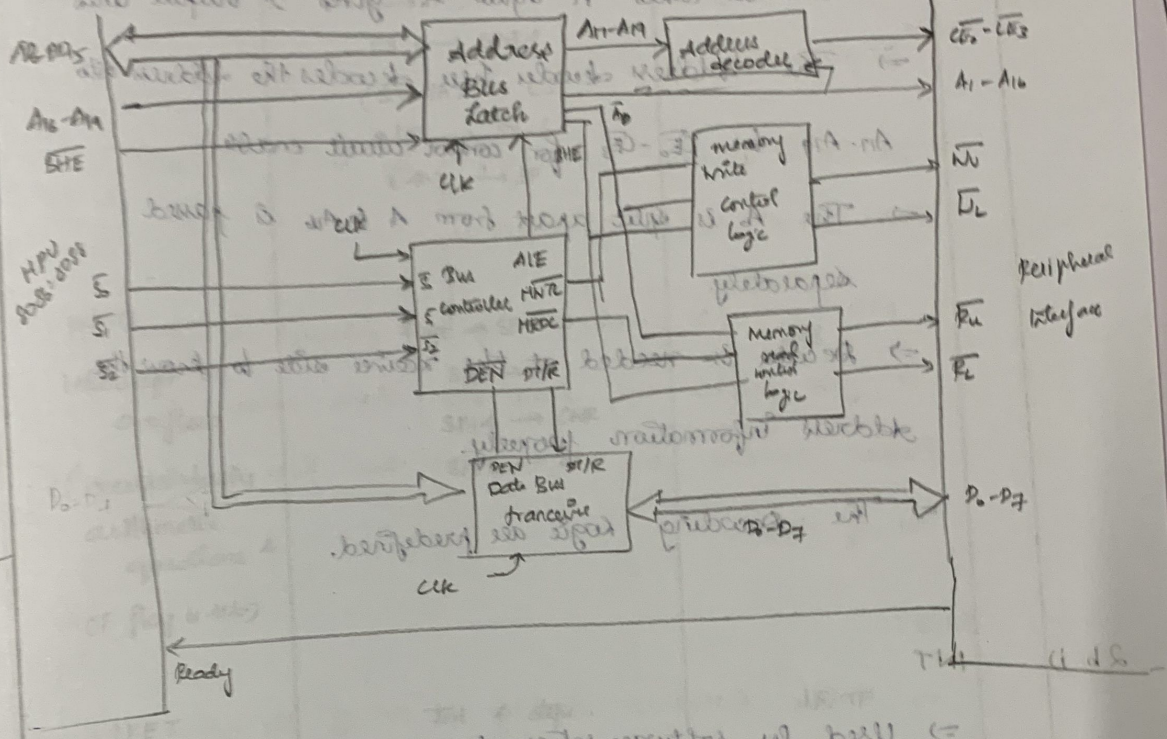
* In memory mapped I/O, the I/O address space is created within the memory of 1MB.
* It has dedicated 4096 addresses for I/O ports.
* E0000 & E0001 can be together combined to form word.
* It uses internal register to connect with MPU in addition to AL and AX register.
* As I/O are used inside the memory 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.



* 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.

3.6 ii) Memory interface circuits:



Address Bus Latch:

=) It latches the addresses provided. The $AD_0 - A_{15}$ of 8086 are multiplexed with data

=) It latches the address separately only then the addresses will not be lost due to some external / internal faults

=) It then passes the address A_0 to memory read & write control logic & $A_1 - A_{16}$ to the interface, $A_{17} - A_{19}$ to Address decoder.

Address decoder.

=> decoder takes n input & gives 2^n output lines

=> The Address decoder then decodes the Address into A_{17}-A_{19} to $\overline{CE_0}$-$\overline{CE_3}$ for control circuit enable.

=> The A_0 is split apart from A to A_{16} is passed separately

=> Decoder is needed at the xceiver side to know the Address information properly

=> The decoding logic are predefined.

2b i) INT

=> Used by software interrupts

=> 0 to 255 (256 software interrupts)

=> predefined interrupts INT 50 ↳ port which leads to the address $(50 \times 4) = 200$ ↳ 4 bytes each.

2b ii)

1100 1000

CS:IP => CA:C8

=> High priority interrupt than NMI & hardware interrupt.

=> INT 0 - divide interrupt.

	Operations	Flags affected
INT n.	Flags → SP+2 0 → IF, TF CS → SP+4 (2+4n) → CS IP → SP+6 (4n) → IP	IF, TF
INTO Interrupt on overflow (occurs during arithmetic operations & OF flag is set)	Flags → SP → CS SP+2 → IP SP+4 → CWR SP+6 → SP	All
IRET (interrupt return)	Int 4 steps	IF, TF

26 ii) NMI: (Non Maskable Interrupts)

⇒ It is a kind of hardware interrupt but different from hardware interrupt in various aspects.

⇒ It cannot be masked by software. Hardware interrupt uses INTR but NMI has separate NMI pin which works when logic 1 i/p.

⇒ It is high priority interrupt than hardware interrupt & it works without setting IF (Interrupt Flag)

=> 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 a masked by software.

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

RESET:

=> Unlike hardware start up on of the systems performed by hardware,

=> This is a (warm-start) by software

=> once this reset is given by software, it then does,

Flags → cleared

IP → 0000H

CS → FFFFH

SS, DS, ES → 0000H

Queue → emptied

AD_0 - AD_{15}, A_{16} - A_{19}, $\overline{BHE}$ ⟶ enters three state

$\overline{RD}$, $\overline{WR}$, $\overline{S_0}$ / IO/$\overline{R}$, S_1 / DT/$\overline{R}$, $\overline{S_2}$ / $\overline{TEST}$, ⟶ goes to 1 then to idle

QS_0, QS_1, $\overline{HOLD}$, ⟶ 0

$\overline{RQ}/GT_0$, $\overline{RQ}/GT_1$ ⟶ 1

ALE ⟶ 0.