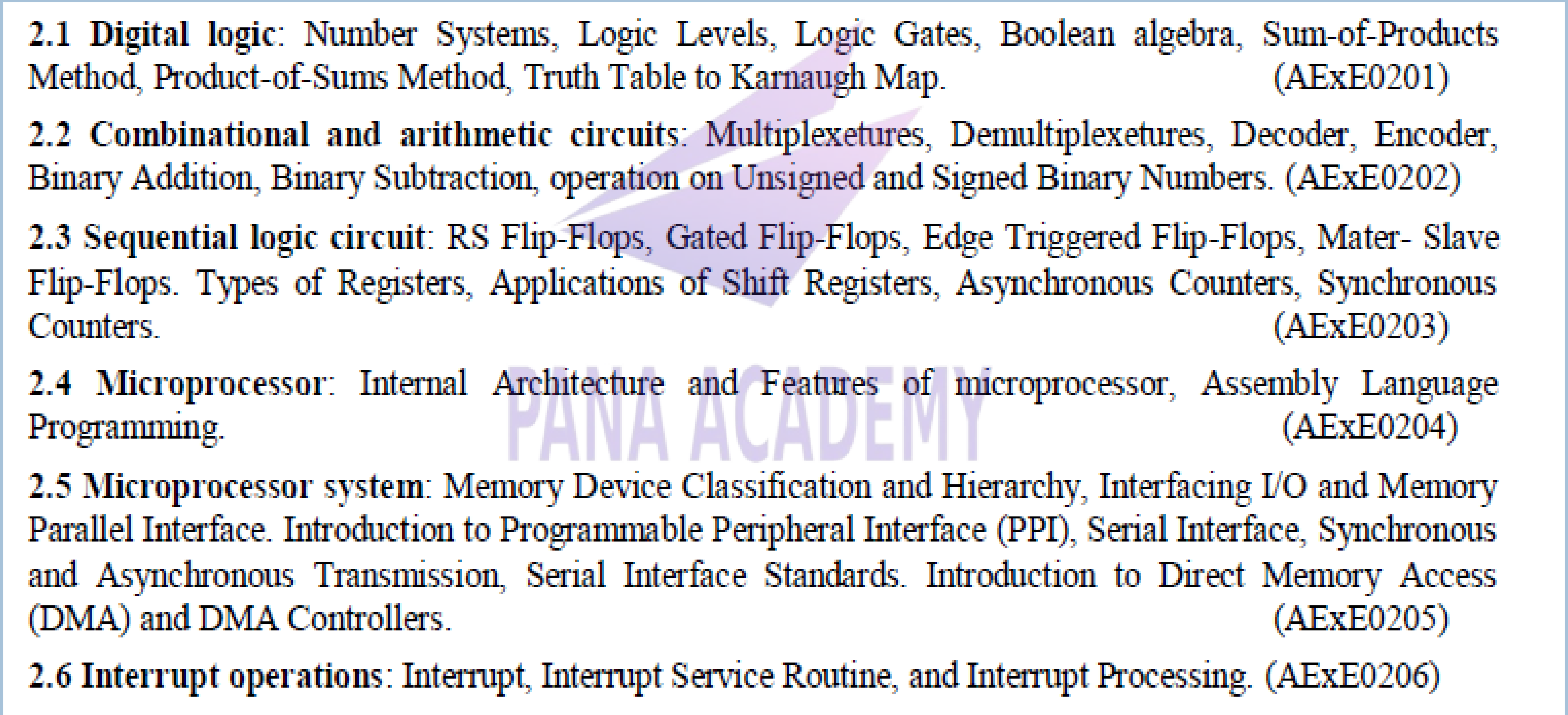
2. Digital Logic and Microprocessor

Er. Pralhad Chapagain

Syllabus



2.1 Digital logic: Number Systems, Logic Levels, Logic Gates, Boolean algebra, Sum-of-Products Method, Product-of-Sums Method, Truth Table to Karnaugh Map. (AExE0201)

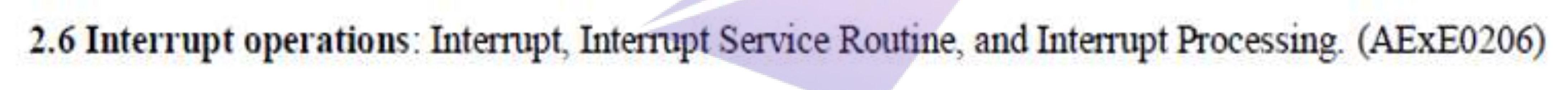
2.2 Combinational and arithmetic circuits: Multiplexetures, Demultiplexetures, Decoder, Encoder, Binary Addition, Binary Subtraction, operation on Unsigned and Signed Binary Numbers. (AExE0202)

2.3 Sequential logic circuit: RS Flip-Flops, Gated Flip-Flops, Edge Triggered Flip-Flops, Mater- Slave Flip-Flops. Types of Registers, Applications of Shift Registers, Asynchronous Counters, Synchronous Counters. (AExE0203)

2.4 Microprocessor: Internal Architecture and Features of microprocessor, Assembly Language Programming. (AExE0204)

2.5 Microprocessor system: Memory Device Classification and Hierarchy, Interfacing I/O and Memory Parallel Interface. Introduction to Programmable Peripheral Interface (PPI), Serial Interface, Synchronous and Asynchronous Transmission, Serial Interface Standards. Introduction to Direct Memory Access (DMA) and DMA Controllers. (AExE0205)

2.6 Interrupt operations: Interrupt, Interrupt Service Routine, and Interrupt Processing. (AExE0206)



2.6 Interrupt operations: Interrupt, Interrupt Service Routine, and Interrupt Processing. (AExE0206)

INTERRUPTS

- Interrupt is signals send by an external device to the processor, to request the processor to perform a particular task or work.
- Mainly in the microprocessor based system the interrupts are used for data transfer between the peripheral and the microprocessor.
- The processor will check the interrupts always at the 2nd T-state of last machine cycle.
- If there is any interrupt it accept the interrupt and send the INTA (active low) signal to the peripheral.
- The vectored address of particular interrupt is stored in program counter.
- The processor executes an interrupt service routine (ISR) addressed in program counter.
- It returned to main program by IRET instruction.

INTERRUPTS

- **Need for Interrupt:**
  - Interrupts are particularly useful when interfacing I/O devices that provide or require data at relatively low data transfer rate.

INTERRUPT OPERATIONS

- The transfer of data between the microprocessor and input /output devices takes place using various modes of operations like programmed I/O, interrupt I/O and direct memory access.
- In programmed I/O, the processor has to wait for a long time until I/O module is ready for operation.
- So the performance of entire system degraded.
- To remove this problem CPU can issue an I/O command to the I/O module and then go to do some useful works. The I/O device will then interrupt the CPU to request service when it is ready to exchange data with CPU.
- In response to an interrupt, the microprocessor stops executing its current program and calls a procedure which services the interrupt.

INTERRUPT OPERATIONS

- The interrupt is a process of data transfer whereby an external device or a peripheral can inform the processor that it is ready for communication and it requests attention.
- The response to an interrupt request is directed or controlled by the microprocessor.

INTERRUPT OPERATIONS

- **Process of interrupt Operation**
- **From the point of view of I/O unit**
  - I/O device receives command from CPU
  - The I/O device then processes the operation
  - The I/O device signals an interrupt to the CPU over a control line.
  - The I/O device waits until the request from CPU.
- **From the point of view of processor**
  - The CPU issues command and then goes off to do its work.
  - When the interrupt from I/O device occurs, the processor saves its program counter & registers of the current program and processes the interrupt.
  - After completion for interrupt, processor requires its initial task.

POLLING VERSUS INTERRUPT

- Each time the device is given a command, for example ``move the read head to sector 42 of the floppy disk" the device driver has a choice as to how it finds out that the command has completed. The device drivers can either poll the device or they can use interrupts.
- Polling the device usually means reading its status register every so often until the device's status changes to indicate that it has completed the request.
- Polling means the CPU keeps checking a flag to indicate if something happens.
- An interrupt driven device driver is one where the hardware device being controlled will cause a hardware interrupt to occur whenever it needs to be serviced.

POLLING VERSUS INTERRUPT

- With interrupt, CPU is free to do other things, and when something happens, an interrupt is generated to notify the CPU. So it means the CPU does not need to check the flag.
- Polling is like picking up your phone every few seconds to see if you have a call. Interrupts are like waiting for the phone to ring.
- Interrupts win if processor has other work to do and event response time is not critical.
- Polling can be better if processor has to respond to an event ASAP; may be used in device controller that contains dedicated secondary processor.

POLLING VERSUS INTERRUPT

- **Advantages of interrupt over Polling**

- Interrupts are used when you need the fastest response to an event. For example, you need to generate a series of pulses using a timer. The timer generates an interrupt when it overflows and within 1 or 2 sec, the interrupt service routine is called to generate the pulse. If polling were used, the delay would depend on how often the polling is done and could delay response to several msecs. This is thousands times slower.

- Interrupts are used to save power consumption. In many battery powered applications, the microcontroller is put to sleep by stopping all the clocks and reducing power consumption to a few micro amps. Interrupts will awaken the controller from sleep to consume power only when needed. Applications of this are hand held devices such as TV/VCR remote controllers.

- Interrupts can be a far more efficient way to code. Interrupts are used for program debugging.

INTERRUPT STRUCTURES

- A processor is usually provided with one or more interrupt pins on the chip.
- Therefore a special mechanism is necessary to handle interrupts from several devices that share one of these interrupt lines. There are mainly two ways of servicing multiple interrupts which are polled interrupts and daisy chain (vectored) interrupts.

POLLED INTERRUPT

- Polled interrupts are handled by using software which is slower than hardware interrupts.
- Here the processor has the general (common) interrupt service routine (ISR) for all devices.
- The priority of the devices is determined by the order in which the routine polls each device.
- The processor checks the starting with the highest priority device.
- Once it determines the source of the interrupt, it branches to the service routine for that device.

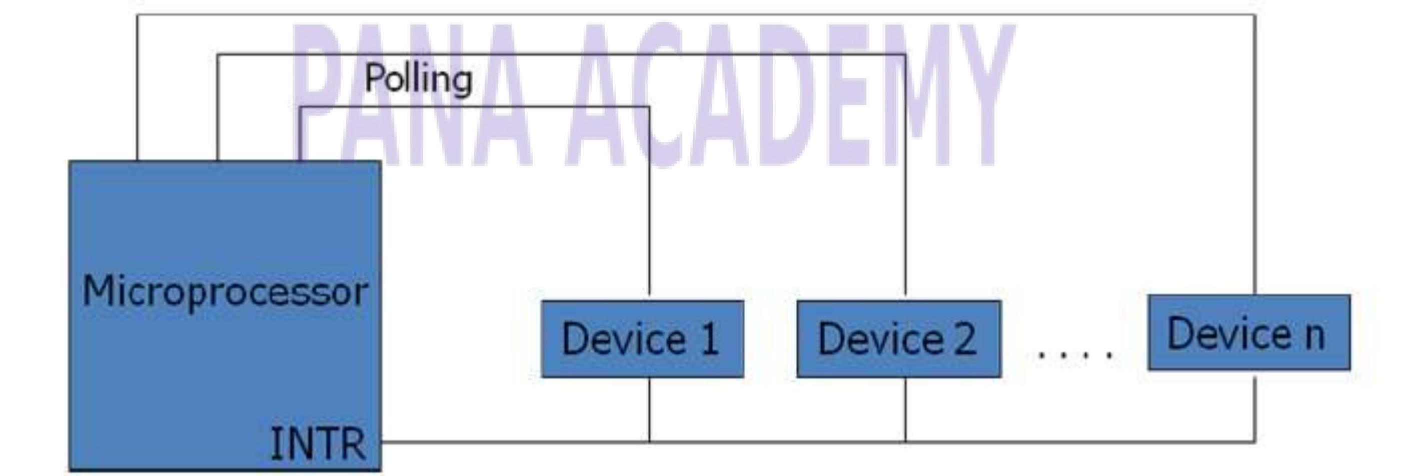


Fig: Polled Interrupt

POLLED INTERRUPT

- Here several eternal devices are connected to a single interrupt line (INTR) of the microprocessor.
- When INTR signal goes up, the processor saves the contents of PC and other registers and then branches to an address defined by the manufactures of the processor.
- The user can write a program at this address to find the source of the interrupt by starting the polled from highest priority device.

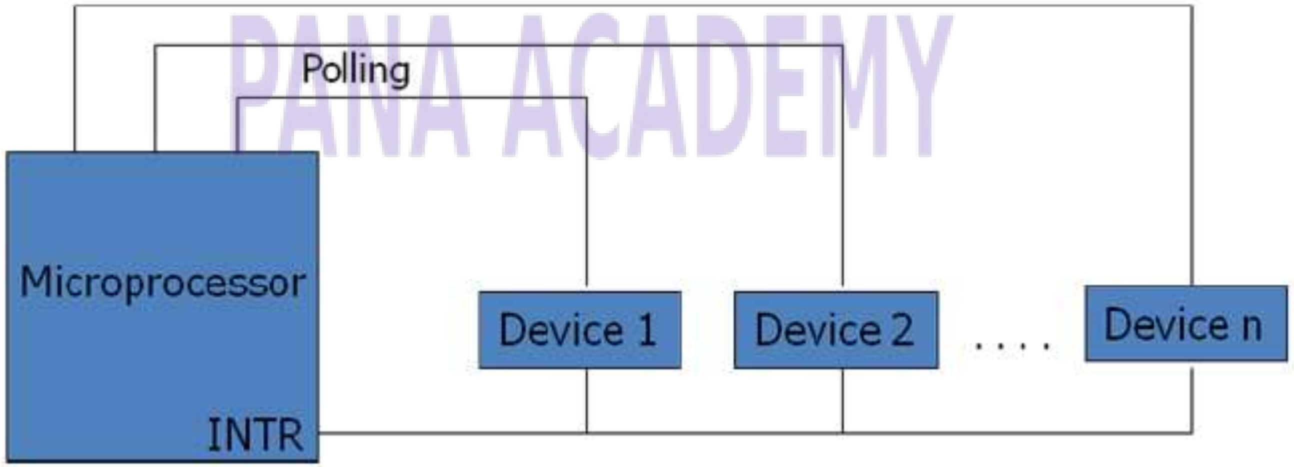


Fig: Polled Interrupt

DAISY CHAIN (VECTORED) INTERRUPT

- In polled interrupt, the time required to poll each device may exceed the time to service the device through software.
- To improve this, the faster mechanism called vectored or daisy chain interrupt is used.
- Here the devices are connected in chain fashion.
- When INTR pin goes up, the processor saves its current status and then generates INTA signal to the highest priority device.
- If this device has generated the interrupt, it will accept the INTA; otherwise it will push INTA to the next priority device until the INTA is accepted by the interrupting device

DAISY CHAIN (VECTORED) INTERRUPT

- When INTA is accepted, the device provides a means to the processor for findings the interrupt address vector using external hardware.
- The accepted device responds by placing a word on the data lines which becomes the vector address with the help of any hardware through which the processor points to appropriate device service routine.
- Here no general interrupt service routine need first that means appropriate ISR of the device will be called.

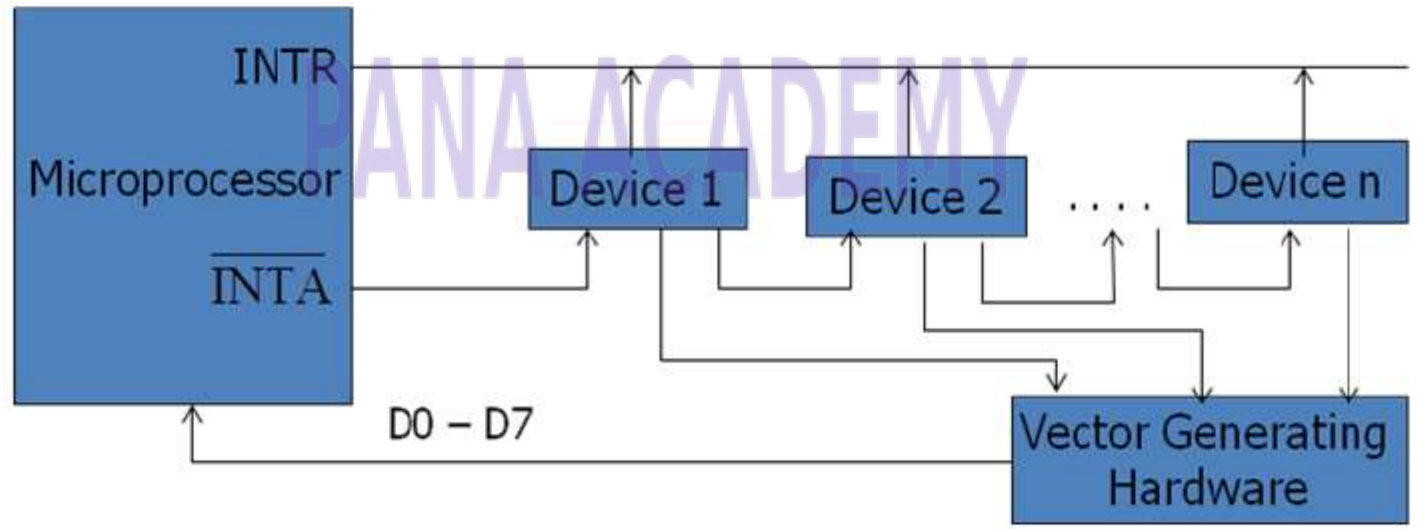


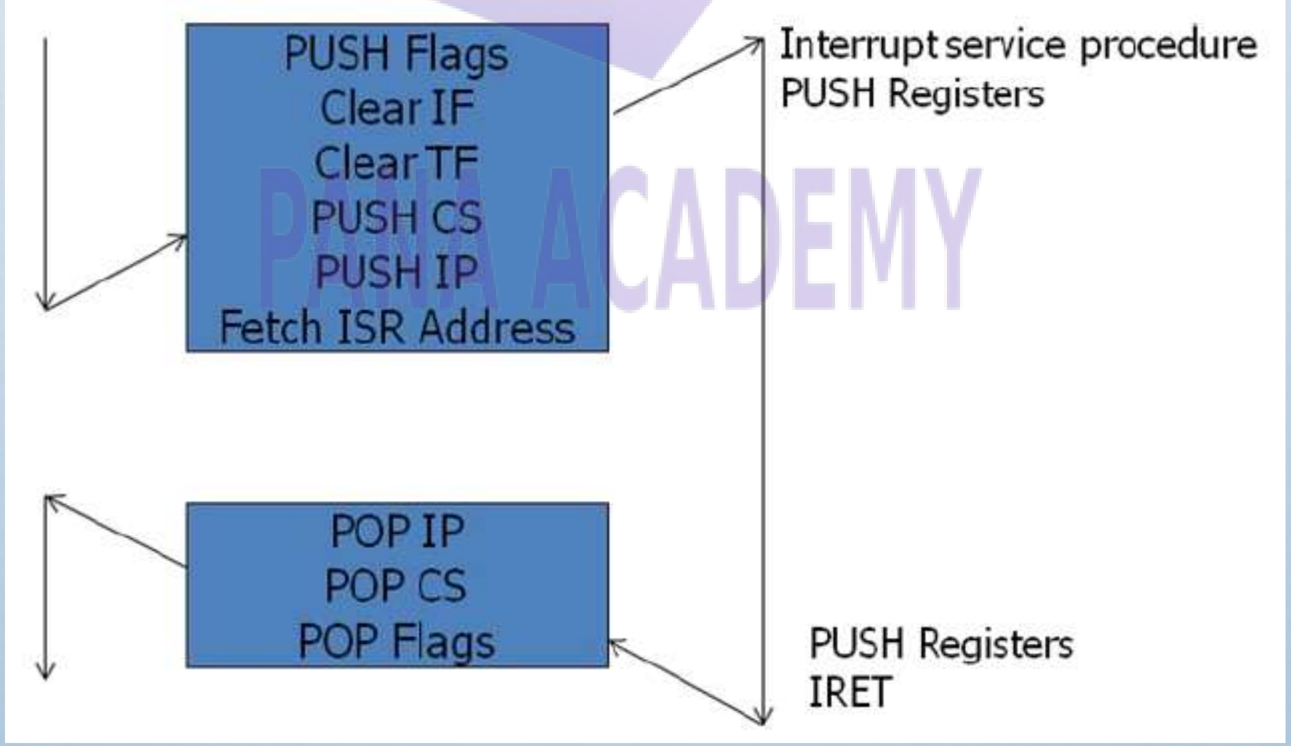
Fig: Vectored (Daisy Chain) Interrupt

INTERRUPT PROCESSING SEQUENCE

- The occurrence of interrupt triggers a number of events, both in processor hardware and in software. The interrupt driven I/O operation takes the following steps.
  - The I/O unit issues an interrupt signal to the processor for exchange of data between them.
  - The processor finishes execution of the current instruction before responding to the interrupt.
  - The processor sends an acknowledgement signal to the device that it issued the interrupt.
  - The processor transfers its control to the requested routine called “Interrupt Service Routine (ISR)” by saving the contents of program status word (PSW) and program counter (PC).
  - The processor now loads the PC with the location of interrupt service routine and the fetches the instructions. The result is transferred to the interrupt handler program.
  - When interrupt processing is completed, the saved register’s value are retrieved from the stack and restored to the register.
  - Finally it restores the PSW and PC values from the stack.

INTERRUPT PROCESSING SEQUENCE

- The figure summarizes these steps.
- The processor pushes the flag register on the stack, disables the INTR input and does essentially an indirect call to the interrupt service procedure.
- An IRET function at the end of interrupt service procedure returns execution to the main program.



INTERRUPT - TYPES

1. External interrupts:

- These interrupts are initiated by external devices such as A/D converters and classified on following types.
- Maskable interrupt :
  - It can be enabled or disabled by executing instructions such as EI and DI. In 8085, EI sets the interrupt enable flip flop and enables the interrupt process. DI resets the interrupt enable flip flop and disables the interrupt.
- Non-maskable interrupt:
  - It has higher priority over maskable interrupt and cannot be enabled or disabled by the instructions.

INTERRUPT - TYPES

2. Internal interrupts:

- These are indicated internally by exceptional conditions such as overflow, divide by zero, and execution of illegal op-code.
- The user usually writes a service routine to take correction measures and to provide an indication in order to inform the user that exceptional condition has occurred.
- There can also be activated by execution of TRAP instruction. This interrupt means TRAP is useful for operating the microprocessor in single step mode and hence important in debugging.
- These interrupts are used by using software to call the function of an operating system. Software interrupts are shorter than subroutine calls and they do not need the calling program to know the operating system’s address in memory.

MULTIPLE INTERRUPTS DEALING APPROACHES:

- If the processor gets multiple interrupts, then we need to deal these interrupts one at a time and the dealing approaches are:

Sequential processing of interrupts

- When user program is executing and an interrupt occurs interrupts are disabled immediately. After the interrupt service routine completes, interrupts are enabled before resuming the user program and the processor checks to see if additional interrupts have occurred.

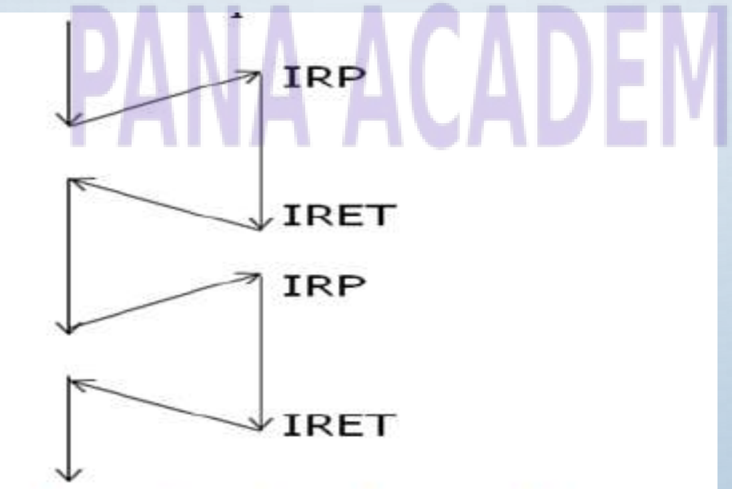


Fig: Sequential Interrupt Service

MULTIPLE INTERRUPTS DEALING APPROACHES:

Priority wise processing of interrupts:

- The drawback of sequential processing is that it does not take account of relative priority or time critical needs.
- The alternative form of this is to define priorities for interrupts and to allow an interrupt of higher priority to cause a lower priority interrupts pause until high priority interrupt completes its function.

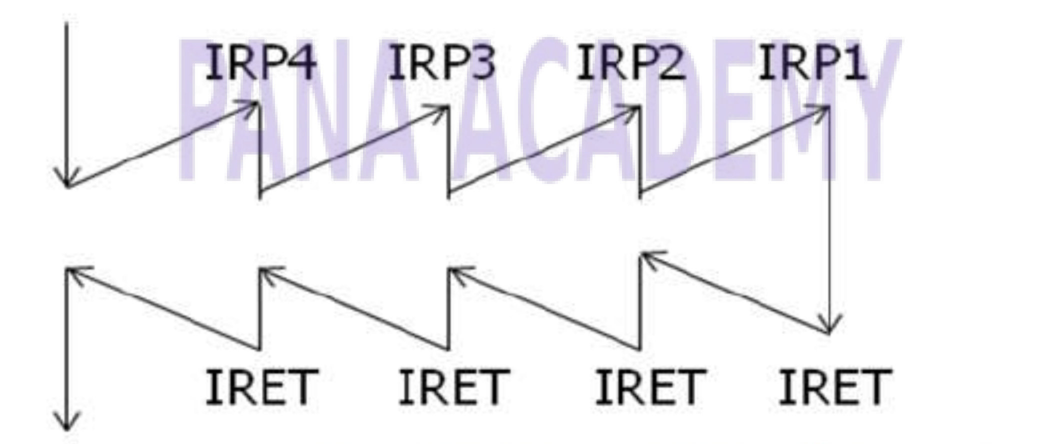


Fig: Priority wise Interrupt service

INTERRUPT SERVICE ROUTINE

- An interrupt service routine (ISR) is a software routine that hardware invokes in response to an interrupt.
- ISRs examine an interrupt and determine how to handle it.
- ISRs handle the interrupt, and then return a logical interrupt value.
- Its central purpose is to process the interrupt and then return control to the main program.
- An ISR must perform very fast to avoid slowing down the operation of the device and the operation of all lower priority ISRs.
- As in procedures, the last instruction in an ISR should be ret.

INTERRUPT SERVICE ROUTINE

ISR is responsible for doing the following things:

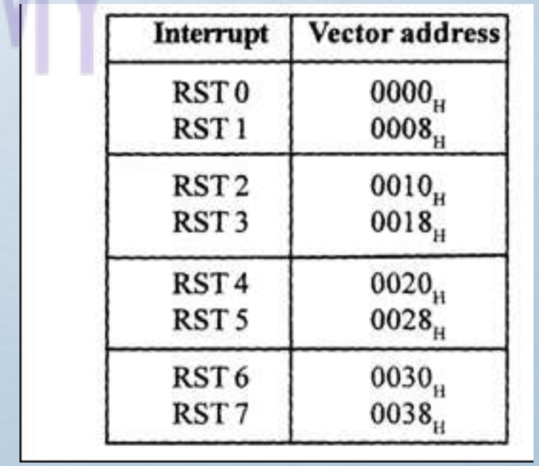
- **Saving the processor context**
  - Because the ISR and main program use the same processor registers, it is the responsibility of the ISR to save the processor's registers before beginning any processing of the interrupt. The processor context consists of the instruction pointer, registers, and any flags. Some processors perform this step automatically.
- **Acknowledging the interrupt**
  - The ISR must clear the existing interrupt, which is done either in the peripheral that generated the interrupt, in the interrupt controller, or both.
- **Restoring the processor context**
  - After interrupt processing, in order to resume the main program, the values that were saved prior to the ISR execution must be restored. Some processors perform this step automatically.

INTERRUPT PROCESSING IN 8085

- Interrupt is signals send by an external device to the processor, to request the processor to perform a particular task or work.
- Mainly in the microprocessor based system the interrupts are used for data transfer between the peripheral and the microprocessor.
- The processor will check the interrupts always at the 2nd T-state of last machine cycle.
- If there is any interrupt it accept the interrupt saves the value of PSW and PC into the stack and send the INTA (active low) signal to the peripheral.
- The vectored address of particular interrupt is stored in program counter.
- The processor executes an interrupt service routine (ISR) addressed in program counter
- It returned to main program by RET instruction.

TYPES OF INTERRUPT

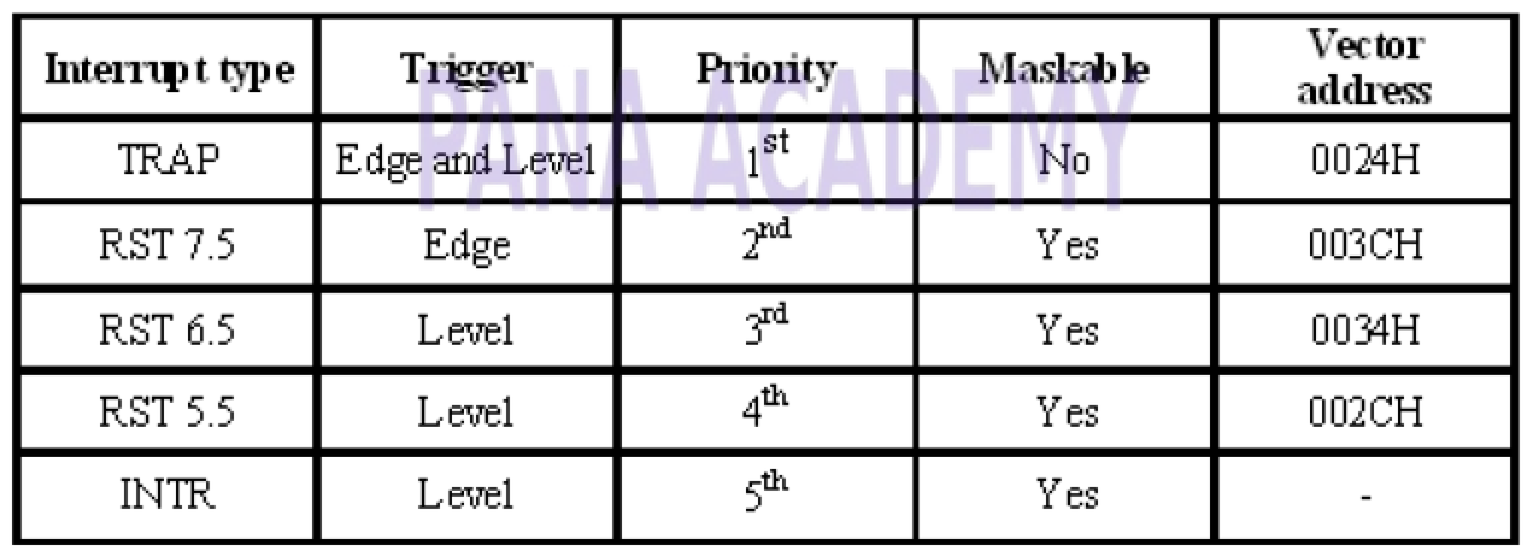
- It supports two types of interrupts.
  - Hardware
  - Software
- **Software interrupts:**
  - The software interrupts are program instructions. These instructions are inserted at desired locations in a program.
  - The 8085 has eight software interrupts from RST 0 to RST 7. The vector address for these interrupts can be calculated as follows.
  - Interrupt number * 8 = vector address
  - For RST 5; 5 * 8 = 40 = 28H
  - Vector address for interrupt RST 5 is 0028H
  - The Table shows the vector addresses of all interrupts



Interrupt	Vector address
RST 0 RST 1	0000_H 0008_H
RST 2 RST 3	0010_H 0018_H
RST 4 RST 5	0020_H 0028_H
RST 6 RST 7	0030_H 0038_H

TYPES OF INTERRUPT

- **Hardware interrupts (Interrupt Pins and Priorities)**
  - An external device initiates the hardware interrupts and placing an appropriate signal at the interrupt pin of the processor.
  - If the interrupt is accepted then the processor executes an interrupt service routine.
  - The 8085 has five hardware interrupts
  - (1) TRAP (2) RST 7.5 (3) RST 6.5 (4) RST 5.5 (5) INTR



Interrupt type	Trigger	Priority	Maskable	Vector address
TRAP	Edge and Level	1st	No	0024H
RST 7.5	Edge	2nd	Yes	003CH
RST 6.5	Level	3rd	Yes	0034H
RST 5.5	Level	4th	Yes	002CH
INTR	Level	5th	Yes	-

TYPES OF INTERRUPT

- **TRAP:**
  - This interrupt is a non-maskable interrupt. It is unaffected by any mask or interrupt enable.
  - TRAP bas the highest priority and vectored interrupt.
  - TRAP interrupt is edge and level triggered. This means hat the TRAP must go high and remain high until it is acknowledged.
  - In sudden power failure, it executes a ISR and send the data from main memory to backup memory.
  - The signal, which overrides the TRAP, is HOLD signal. (i.e., If the processor receives HOLD and TRAP at the same time then HOLD is recognized first and then TRAP is recognized).
  - There are two ways to clear TRAP interrupt.
    - By resetting microprocessor (External signal)
    - By giving a high TRAP ACKNOWLEDGE (Internal signal)

TYPES OF INTERRUPT

- **RST 7.5:**
  - The RST 7.5 interrupt is a maskable interrupt.
  - It has the second highest priority.
  - It is edge sensitive. i.e. Input goes to high and no need to maintain high state until it recognized.
  - Maskable interrupt. It is disabled by,
    - 1. DI instruction
    - 2. System or processor reset.
    - 3. After reorganization of interrupt.
  - Enabled by EI instruction**.**

TYPES OF INTERRUPT

- **RST 6.5 and 5.5:**
  - The RST 6.5 and RST 5.5 both are level triggered. . ie. Input goes to high and stay high until it recognized.
  - Maskable interrupt. It is disabled by,
    - DI, SIM instruction
    - System or processor reset.
    - After reorganization of interrupt.
  - Enabled by EI instruction.
  - The RST 6.5 has the third priority whereas RST 5.5 has the fourth priority.

TYPES OF INTERRUPT

- **INTR:**
  - INTR is a maskable interrupt.
  - It is disabled by,
    - 1. DI, SIM instruction
    - 2. System or processor reset.
    - 3. After reorganization of interrupt.
  - Enabled by EI instruction.
  - Non- vectored interrupt. After receiving INTA (active low) signal, it has to supply the address of ISR.
  - It has lowest priority.
  - It is a level sensitive interrupts. ie. Input goes to high and it is necessary to maintain high state until it recognized.

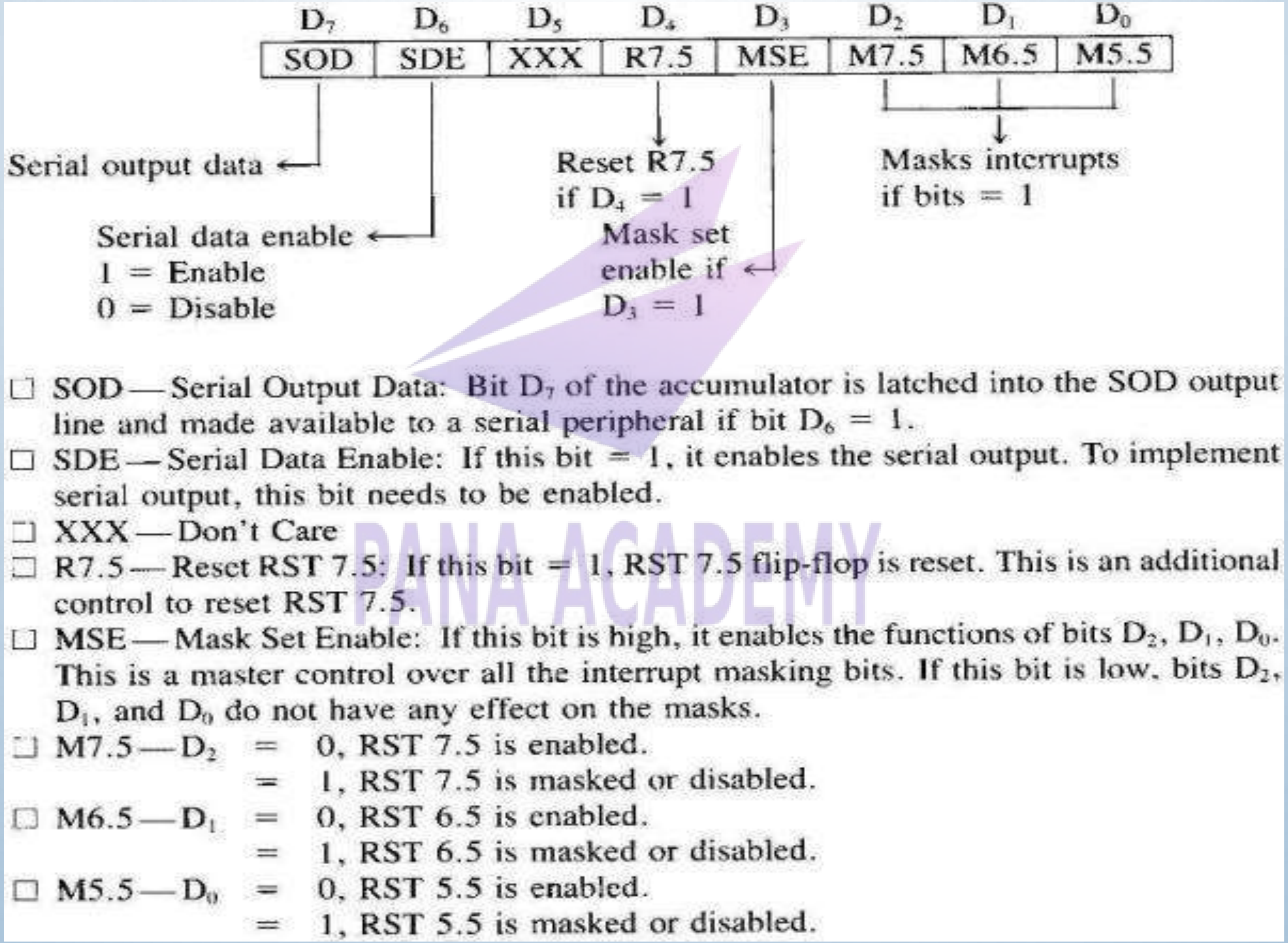
TYPES OF INTERRUPT

- **The following sequence of events occurs when INTR signal goes high.**
  - **1**. The 8085 checks the status of INTR signal during execution of each instruction.
  - 2. If INTR signal is high, then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled.
  - 3. In response to the acknowledge signal, external logic places an instruction OPCODE on the data bus. In the case of multibyte instruction, additional interrupt acknowledge machine cycles are generated by the 8085 to transfer the additional bytes into the microprocessor.
  - 4. On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction**.**

INTERRUPT INSTRUCTIONS

- **SIM (Set Interrupt Mask) instruction:**
  - The 8085 provide additional masking facility for RST 7.5, RST 6.5 and RST 5.5 using SIM instruction.
  - This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
  - The masking or unmasking of RST 7.5, RST 6.5 and RST 5.5 interrupts can be performed by moving an 8-bit data to accumulator and then executing SIM instruction.
  - The format of the 8-bit data is shown below.

INTERRUPT INSTRUCTIONS



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SOD	SDE	XXX	R7.5	MSE	M7.5	M6.5	M5.5

Serial output data

Serial data enable
1 = Enable
0 = Disable

Reset R7.5
if $D_4 = 1$

Mask set
enable if
$D_3 = 1$

Masks interrupts
if bits = 1

□ SOD — Serial Output Data: Bit D_7 of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit $D_6 = 1$.

□ SDE — Serial Data Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.

□ XXX — Don't Care

□ R7.5 — Reset RST 7.5: If this bit = 1, RST 7.5 flip-flop is reset. This is an additional control to reset RST 7.5.

□ MSE — Mask Set Enable: If this bit is high, it enables the functions of bits D_2, D_1, D_0. This is a master control over all the interrupt masking bits. If this bit is low, bits D_2, D_1, and D_0 do not have any effect on the masks.

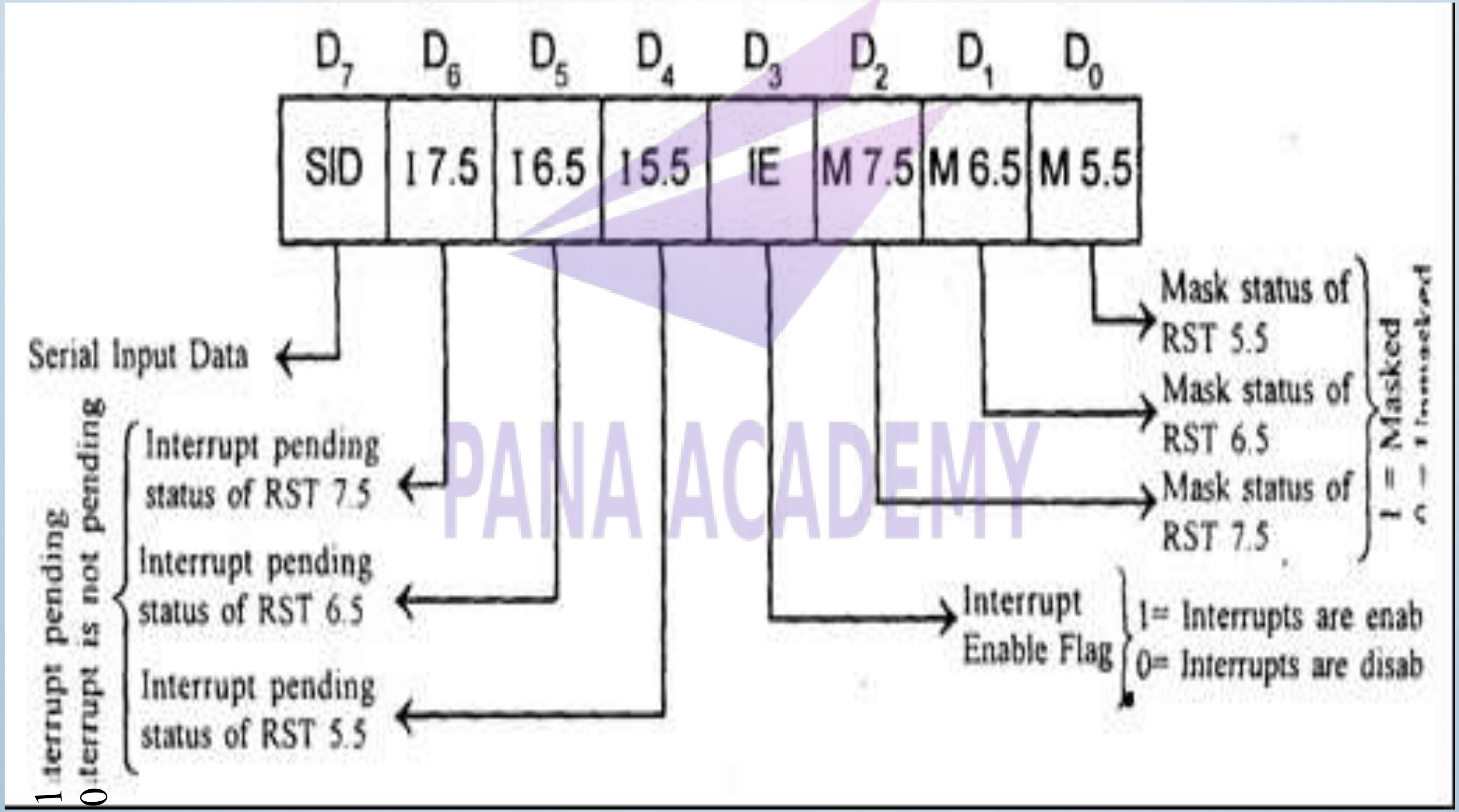
□ M7.5 — D_2 = 0, RST 7.5 is enabled.
= 1, RST 7.5 is masked or disabled.

□ M6.5 — D_1 = 0, RST 6.5 is enabled.
= 1, RST 6.5 is masked or disabled.

□ M5.5 — D_0 = 0, RST 5.5 is enabled.
= 1, RST 5.5 is masked or disabled.

INTERRUPT INSTRUCTIONS

- **RIM (Read Interrupt Mask) instruction:**



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SID	I 7.5	I 6.5	I 5.5	IE	M 7.5	M 6.5	M 5.5

INTERRUPT INSTRUCTIONS

- The status of pending interrupts can be read from accumulator after executing RIM instruction.
- This is a multipurpose instruction used to read the status of RST 7.5, 6.5, 5.5 and read serial data input bit.
- When RIM instruction is executed an 8-bit data is loaded in accumulator, which can be interpreted as shown in above fig.
- **Bits 0-2** show the current setting of the mask for each of RST 7.5, RST 6.5 and RST 5.5. They return the contents of the three masks flip flops. They can be used by a program to read the mask settings in order to modify only the right mask.
- **Bit 3** shows whether the maskable interrupt process is enabled or not. It returns the contents of the Interrupt Enable Flip Flop. It can be used by a program to determine whether or not interrupts are enabled.
- **Bits 4-6** show whether or not there are pending interrupts on RST 7.5, RST 6.5, and RST 5.5. Bits 4 and 5 return the current value of the RST5.5 and RST6.5 pins. Bit 6 returns the current value of the RST7.5 memory flip flop.
- **Bit 7** is used for Serial Data Input. The RIM instruction reads the value of the SID pin on the microprocessor and returns it in this bit.

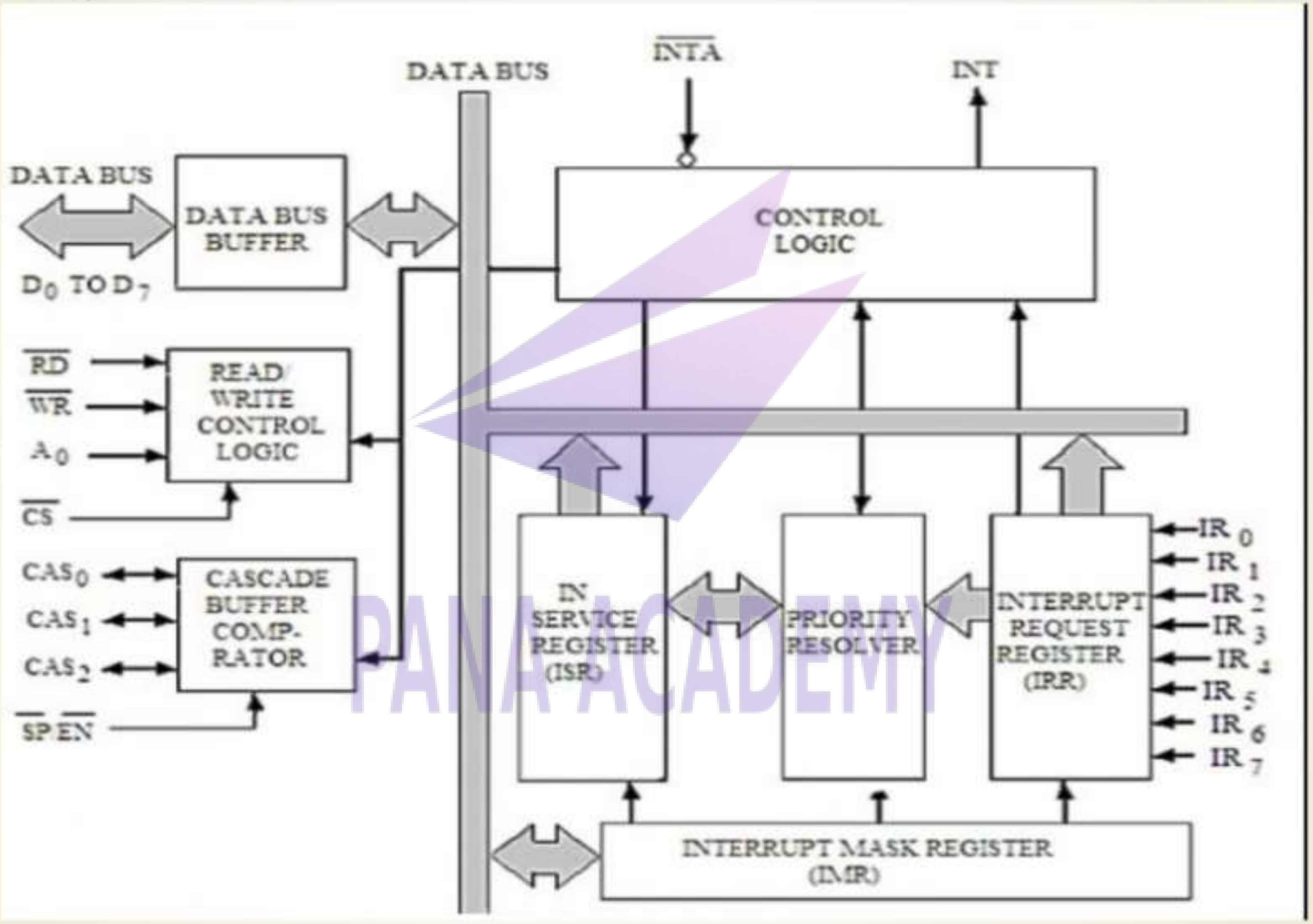
INTERRUPT INSTRUCTIONS

- **DI**
  - Disable interrupts
  - The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.
  - 1 byte instruction
  - Example: DI
- **EI**
  - Enable interrupts
  - The interrupt enable flip-flop is set and all interrupts are enabled.
  - No flags are affected.
  - After a system reset or the acknowledgement of an interrupt, the interrupt enable flip flop is reset, thus disabling the interrupts.
  - This instruction is necessary to enable the interrupts (except TRAP).
  - 1 byte instruction
  - Example: EI

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (1)

- The 8259 is known as the Programmable Interrupt Controller (PIC) microprocessor.
- In 8085 and 8086 there are five hardware interrupts and two hardware interrupts respectively.
- But adding 8259, we can increase the interrupt handling capability.
- This chip combines the multi-interrupt input source to single interrupt output.
- This provides 8-interrupts from IR0 to IR7.
- Let us see some features of this microprocessor.
  - This chip is designed for 8085 and 8086.
  - It can be programmed either in edge triggered, or in level triggered mode
  - We can mask individual bits of Interrupt Request Register.
  - By cascading 8259 chips, we can increase interrupts up to 64 interrupt lines
  - Clock cycle is not needed

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (2)



8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (3)

Data Bus Buffer

- This block is used to communicate between 8259 and 8085/8086 by acting as buffer. It takes the control word from 8085/8086 and send it to the 8259. It transfers the opcode of the selected interrupts and address of ISR to the other connected microprocessor. It can send maximum 8-bit at a time.

R/W Control Logic

- This block works when the value of pin CS is 0. This block is used to flow the data depending upon the inputs of RD and WR. These are active low pins for read and write.

Control Logic

- It controls the functionality of each block. It has pin called INTR. This is connected to other microprocessors for taking the interrupt request. The INT pin is used to give the output. If 8259 is enabled, and also the interrupt flags of other microprocessors are high then this causes the value of the output INT pin high, and in this way this chip can responds requests made by other microprocessors.

Interrupt Request Register

- It stores all interrupt level that are requesting for interrupt service.

Interrupt Service Register

- It stores interrupt level that are currently being execute..

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (4)

Interrupt Mask Register

- It stores interrupt level that will be masked, by storing the masking bits of interrupt level.

Priority Resolver

- It checks all three registers, and set the priority of the interrupts. Interrupt with the highest priority is set in the ISR register. It also reset the interrupt level which is already been serviced in the IRR.

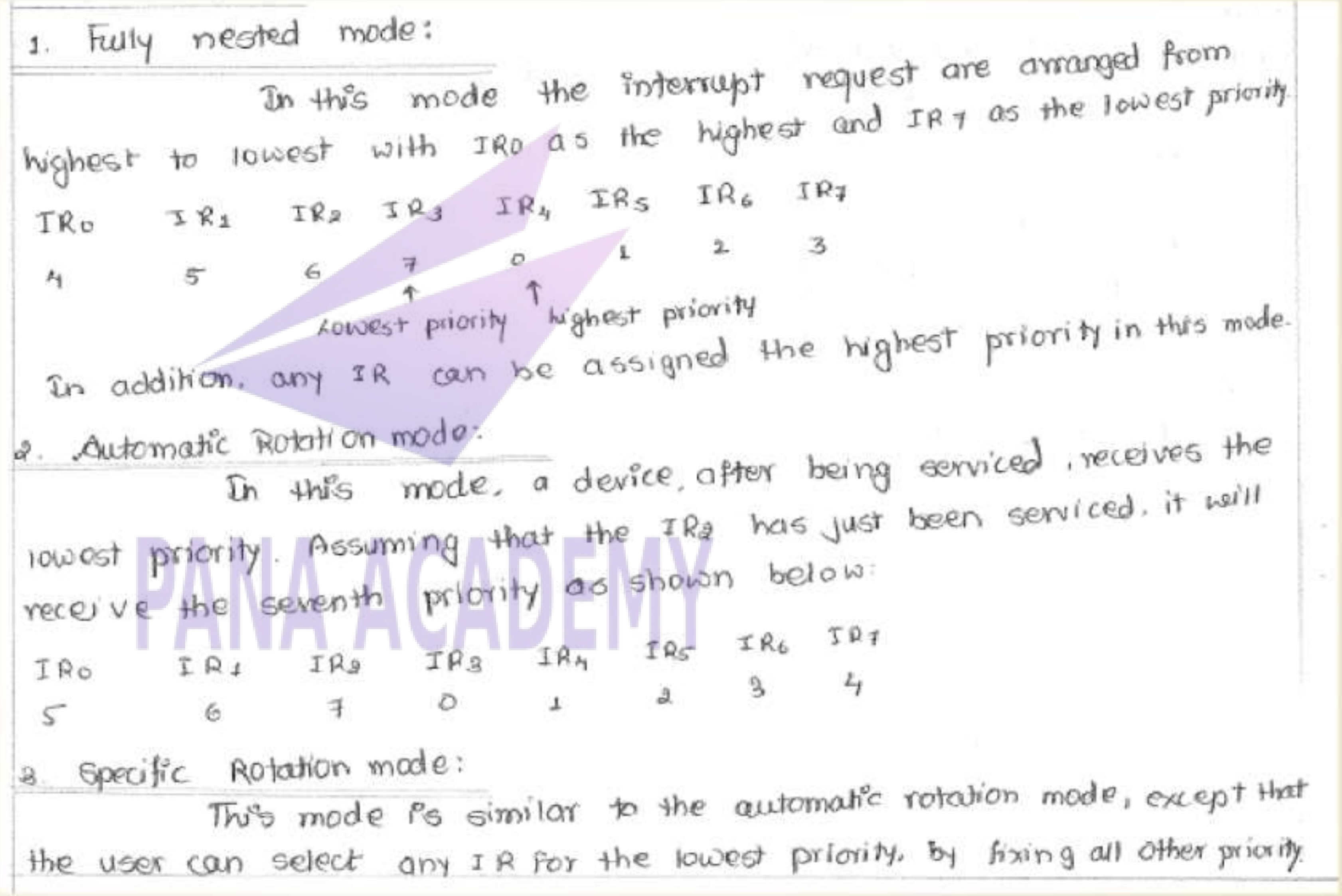
Cascade Buffer

- To increase number of interrupt pin, we can cascade more number of pins, by using cascade buffer. When we are going to increase the interrupt capability, CSA lines are used to control multiple interrupts.

- SP/EN (Slave program/Enable buffer) pin is when set to high, works in master mode else in slave mode. In Non Buffered mode, SP/EN pin is used to specify whether 8259 work as master or slave and in Buffered mode, SP/EN pin is used as an output to enable data bus.

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (4)

Priority Modes:



1. Fully nested mode:

In this mode the interrupt request are arranged from highest to lowest with IR0 as the highest and IR7 as the lowest priority.

IR_0	IR_1	IR_2	IR_3	IR_4	IR_5	IR_6	IR_7
4	5	6	7	0	1	2	3
			↑ lowest priority	↑ highest priority			

In addition, any IR can be assigned the highest priority in this mode.

2. Automatic Rotation mode:

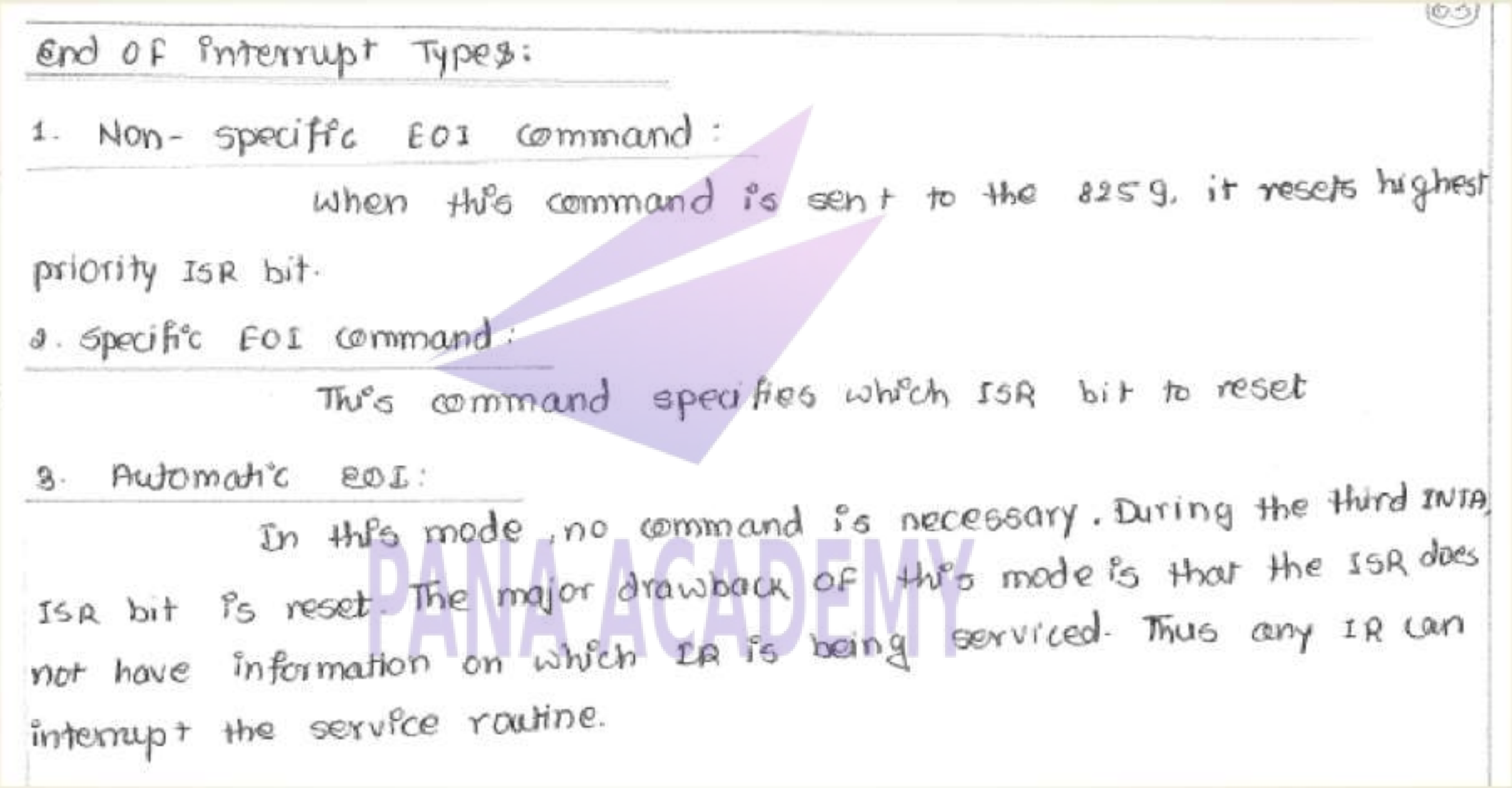
In this mode, a device, after being serviced, receives the lowest priority. Assuming that the IR2 has just been serviced, it will receive the seventh priority as shown below:

IR_0	IR_1	IR_2	IR_3	IR_4	IR_5	IR_6	IR_7
5	6	7	0	1	2	3	4

3. Specific Rotation mode:

This mode is similar to the automatic rotation mode, except that the user can select any IR for the lowest priority, by fixing all other priority

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (5)



End of Interrupt Types:

1. Non- specific EOI command:

When this command is sent to the 8259, it resets highest priority ISR bit.

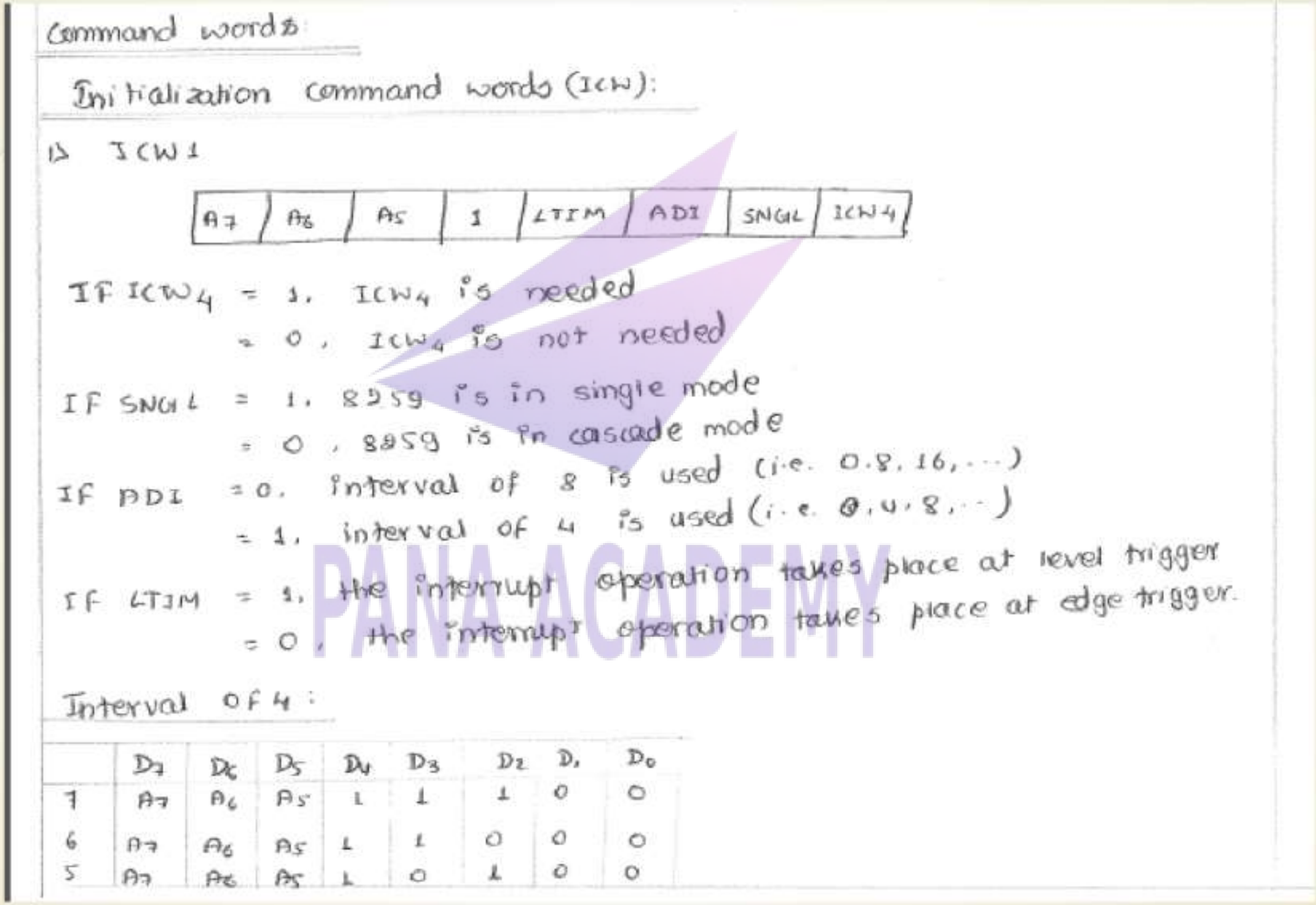
2. Specific EOI command:

This command specifies which ISR bit to reset

3. Automatic EOI:

In this mode, no command is necessary. During the third INTA, ISR bit is reset. The major drawback of this mode is that the ISR does not have information on which IR is being serviced. Thus any IR can interrupt the service routine.

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (6)



Command words:

Initialization command words (ICW):

1) ICW1

A7	A6	A5	1	LTIM	ADI	SNGL	ICW4

IF ICW4 = 1. ICW4 is needed
= 0, ICW4 is not needed

IF SNGL = 1. 8259 is in single mode
= 0, 8259 is in cascade mode

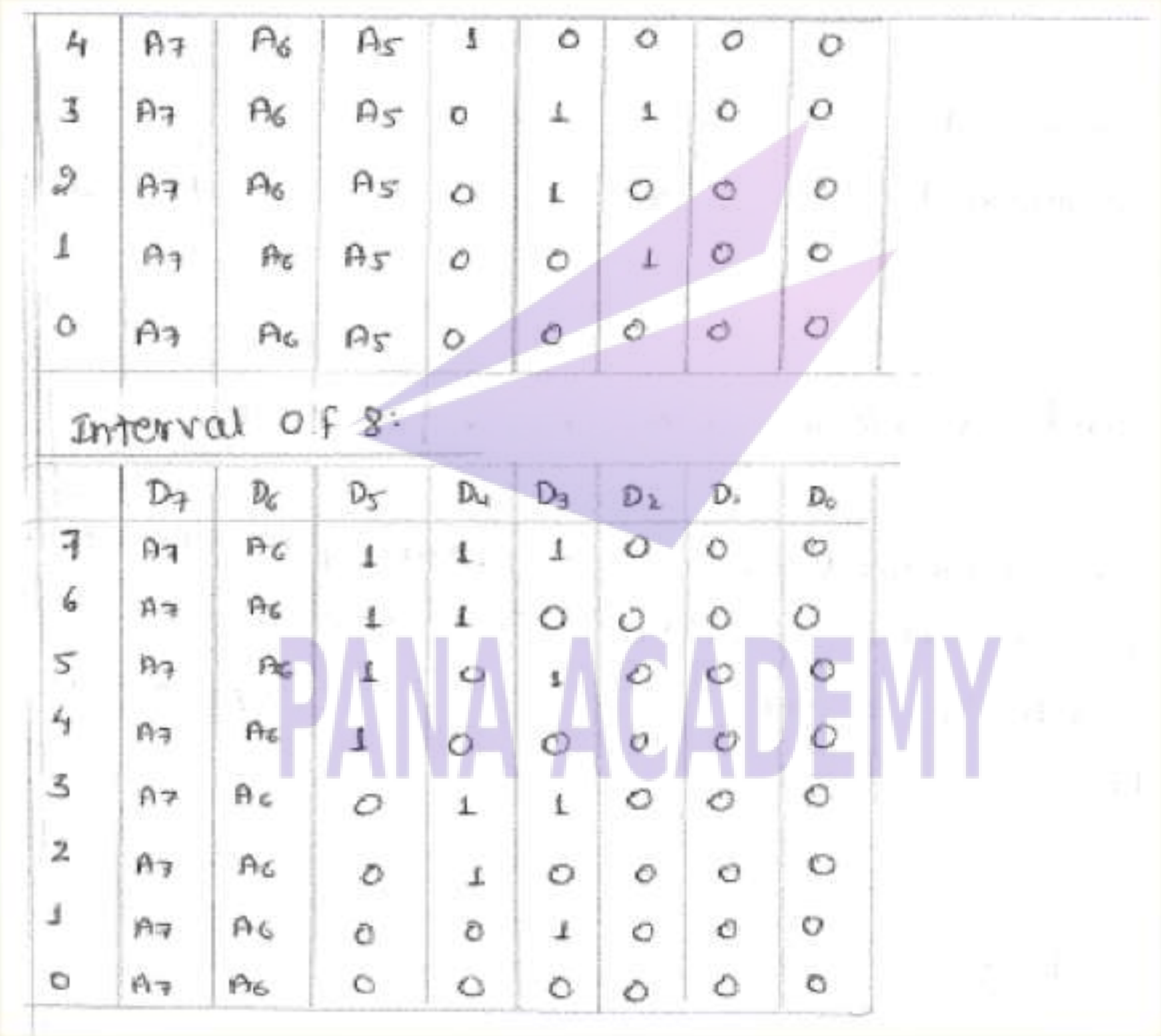
IF ADI = 0. interval of 8 is used (i.e. 0.8, 16, ...)
= 1. interval of 4 is used (i.e. 0, 4, 8, ..)

IF LTIM = 1. the interrupt operation takes place at level trigger
= 0, the interrupt operation takes place at edge trigger.

Interval of 4:

	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	A5	1	1	1	0	0
6	A7	A6	A5	1	1	0	0	0
5	A7	A6	A5	1	0	1	0	0

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (7)

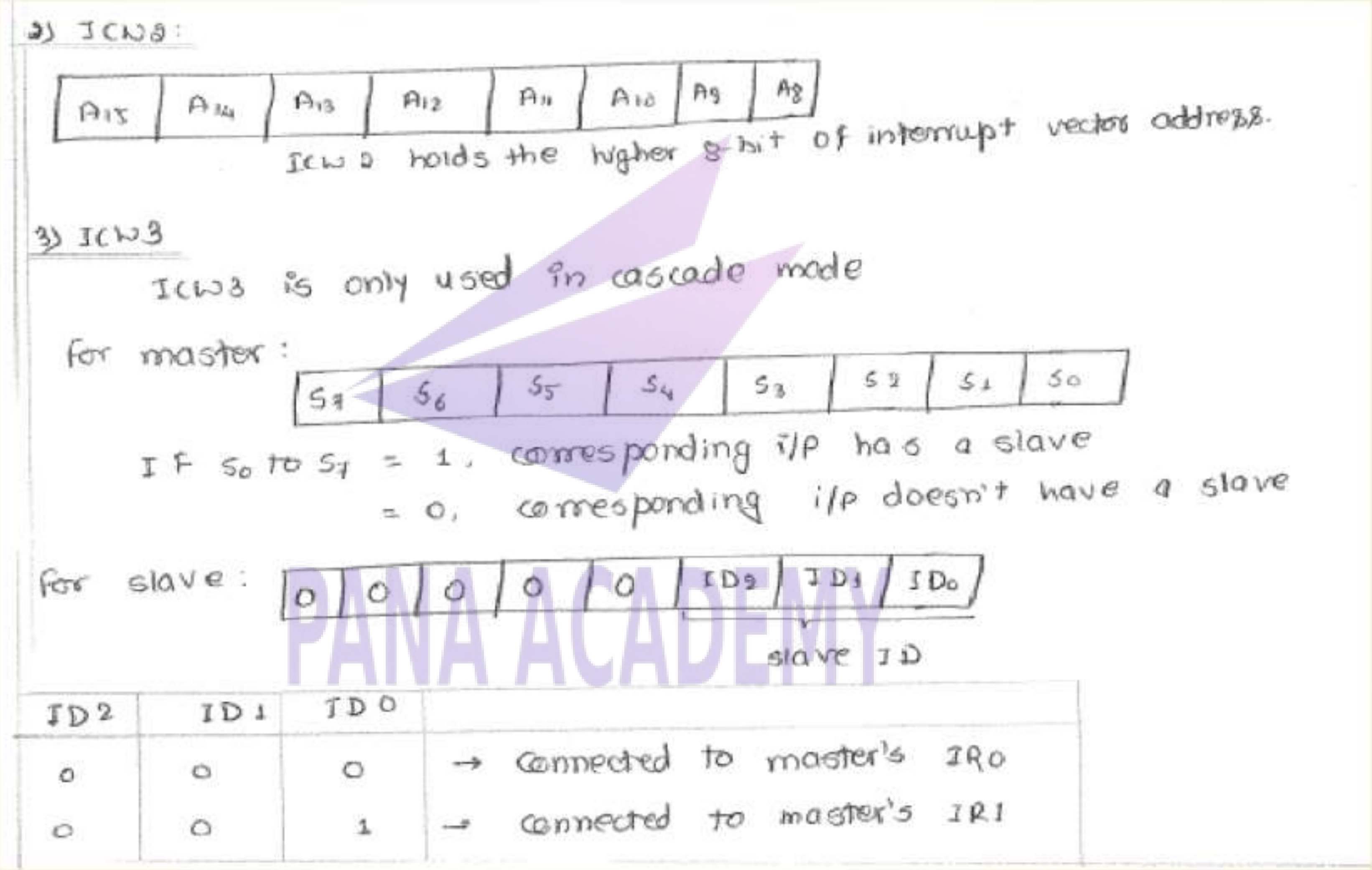


4	A_7	A_6	A_5	1	0	0	0	0
3	A_7	A_6	A_5	0	1	1	0	0
2	A_7	A_6	A_5	0	1	0	0	0
1	A_7	A_6	A_5	0	0	1	0	0
0	A_7	A_6	A_5	0	0	0	0	0

Interval of 8:

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
7	A_7	A_6	1	1	1	0	0	0
6	A_7	A_6	1	1	0	0	0	0
5	A_7	A_6	1	0	1	0	0	0
4	A_7	A_6	1	0	0	0	0	0
3	A_7	A_6	0	1	1	0	0	0
2	A_7	A_6	0	1	0	0	0	0
1	A_7	A_6	0	0	1	0	0	0
0	A_7	A_6	0	0	0	0	0	0

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (8)



2) ICW2:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8

ICW2 holds the higher 8 bit of interrupt vector address.

3) ICW3

ICW3 is only used in cascade mode

for master:

S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

IF S_0 to S_7 = 1, corresponding i/p has a slave

= 0, corresponding i/p doesn't have a slave

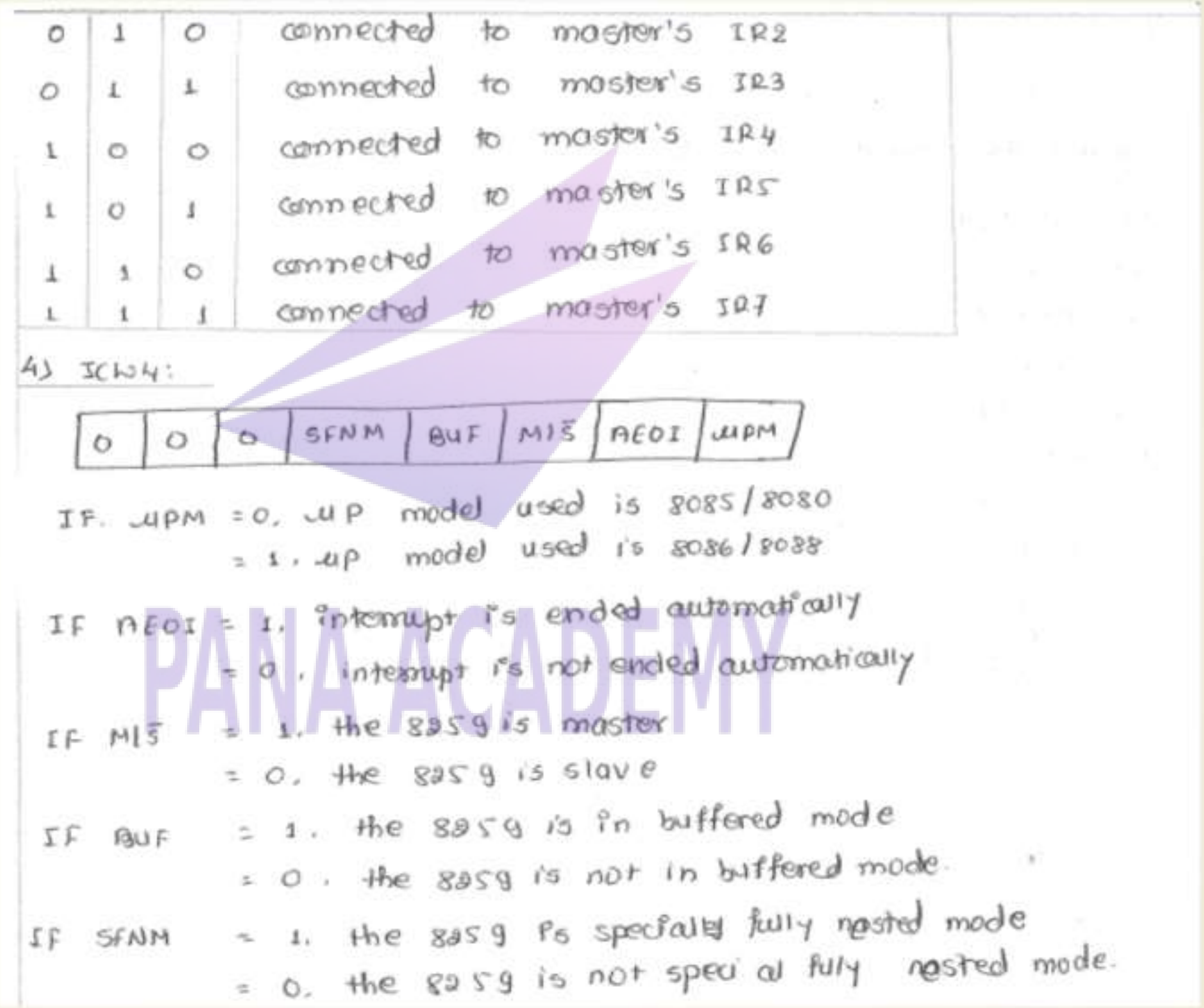
for slave:

0	0	0	0	0	ID_2	ID_1	ID_0

slave ID

ID2	ID1	ID0	
0	0	0	→ Connected to master's IR0
0	0	1	→ connected to master's IR1

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (9)



0	1	0	connected to master's IR2
0	1	1	connected to master's IR3
1	0	0	connected to master's IR4
1	0	1	connected to master's IR5
1	1	0	connected to master's IR6
1	1	1	connected to master's IR7

4) ICW4:

0	0	0	SFNM	BUF	M/$\overline{S}$	AEOI	μPM

IF μPM = 0, μP model used is 8085/8080
= 1, μP model used is 8086/8088

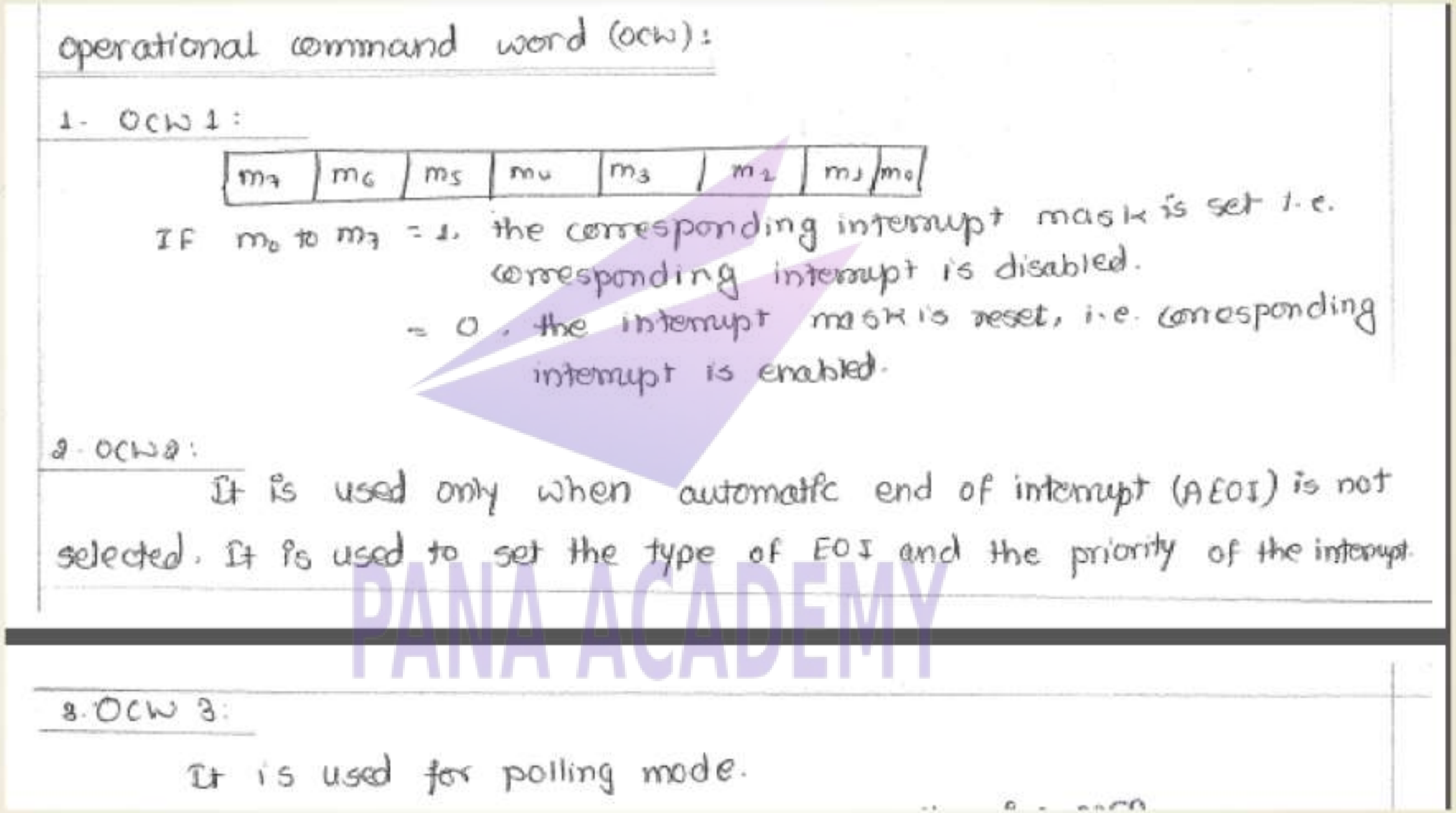
IF AEOI = 1, interrupt is ended automatically
= 0, interrupt is not ended automatically

IF M/$\overline{S}$ = 1, the 8259 is master
= 0, the 8259 is slave

IF BUF = 1, the 8259 is in buffered mode
= 0, the 8259 is not in buffered mode.

IF SFNM = 1, the 8259 is specially fully nested mode
= 0, the 8259 is not special fully nested mode.

8259 A (PROGRAMMABLE INTERRUPT CONTROLLER) (10)



operational command word (ocw):

1. OCW 1:

m_7	m_6	m_5	m_4	m_3	m_2	m_1	m_0

IF m_0 to m_7 = 1, the corresponding interrupt mask is set i.e. corresponding interrupt is disabled.

= 0, the interrupt mask is reset, i.e. corresponding interrupt is enabled.

2. OCW 2:

It is used only when automatic end of interrupt (AEOI) is not selected. It is used to set the type of EOI and the priority of the interrupt.

3. OCW 3:

It is used for polling mode.

INTERRUPT PROCESSING IN 8086

- The meaning of ‘interrupts’ is to break the sequence of operation.
- While the CPU is executing a program, on ‘interrupt’ breaks the normal sequence of execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR).
- After executing ISR , the control is transferred back again to the main program.
- Interrupt processing is an alternative to polling.

INTERRUPT PROCESSING IN 8086

Interrupt Pins

- **INTR and NMI**
  - INTR is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the FLAGS register with the help of the POPF instruction.
  - When an interrupt occurs, the processor stores FLAGS register into stack, disables further interrupts, fetches from the bus one byte representing interrupt type, and jumps to interrupt processing routine address of which is stored in location 4 * <interrupt type>. Interrupt processing routine should return with the IRET instruction.
  - NMI is a non-maskable interrupt. Interrupt is processed in the same way as the INTR interrupt. Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h. This interrupt has higher priority than the maskable interrupt.
  - – Ex: NMI, INTR.

INTERRUPT VECTOR TABLE AND ITS ORGANIZATION

- An interrupt vector is a pointer to where the ISR is stored in memory.
- All interrupts (vectored or otherwise) are mapped onto a memory area called the Interrupt Vector Table (IVT).
    - The IVT is usually located in memory page 00 (0000H - 00FFH).
    - The purpose of the IVT is to hold the vectors that redirect the microprocessor to the right place when an interrupt arrives.
- Interrupt Vector Table (IVT) is a 1024 bytes sized table that contains addresses of interrupts.
- Each address is of 4 bytes long of the form offset:segment, which represents the address of a routine to be called when the CPU receives an interrupt.
- IVT can hold maximum of 256 addresses (0 to 255).
- The interrupt number is used as an index into the table to get the address of the interrupt service routine.

INTERRUPT VECTOR TABLE AND ITS ORGANIZATION

- IVT act as pointers, unlike function call IVT need number as an argument then as a result IVT point us to interrupt service routine (ISR).
- ISR executes its code, when ISR finished then returns back to original statement. Interrupt vector table is a global table situated at the address 0000:0000H.
- The interrupt vector table is a feature of the Intel 8086/8088 family of microprocessors.

INTERRUPT VECTOR TABLE AND ITS ORGANIZATION

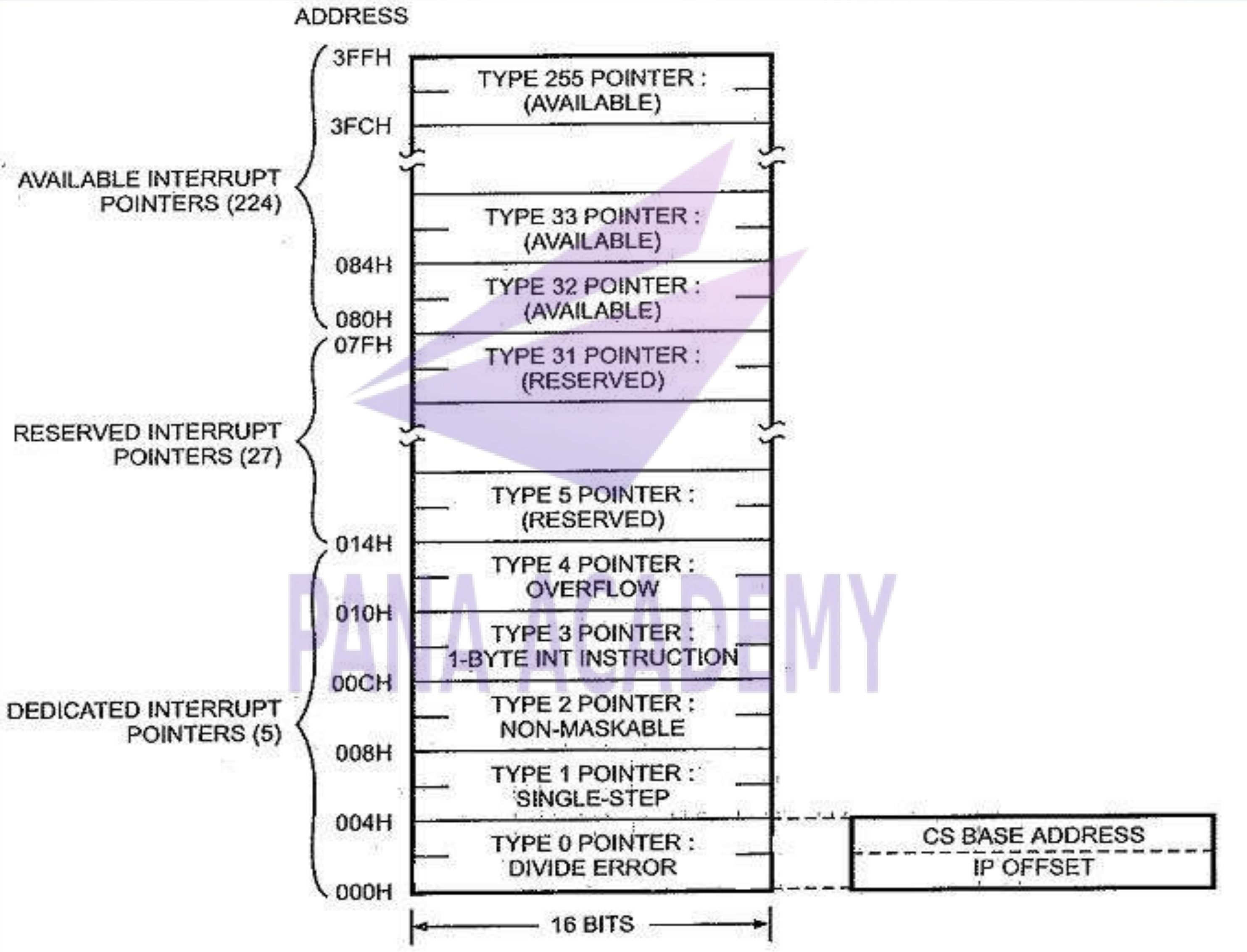
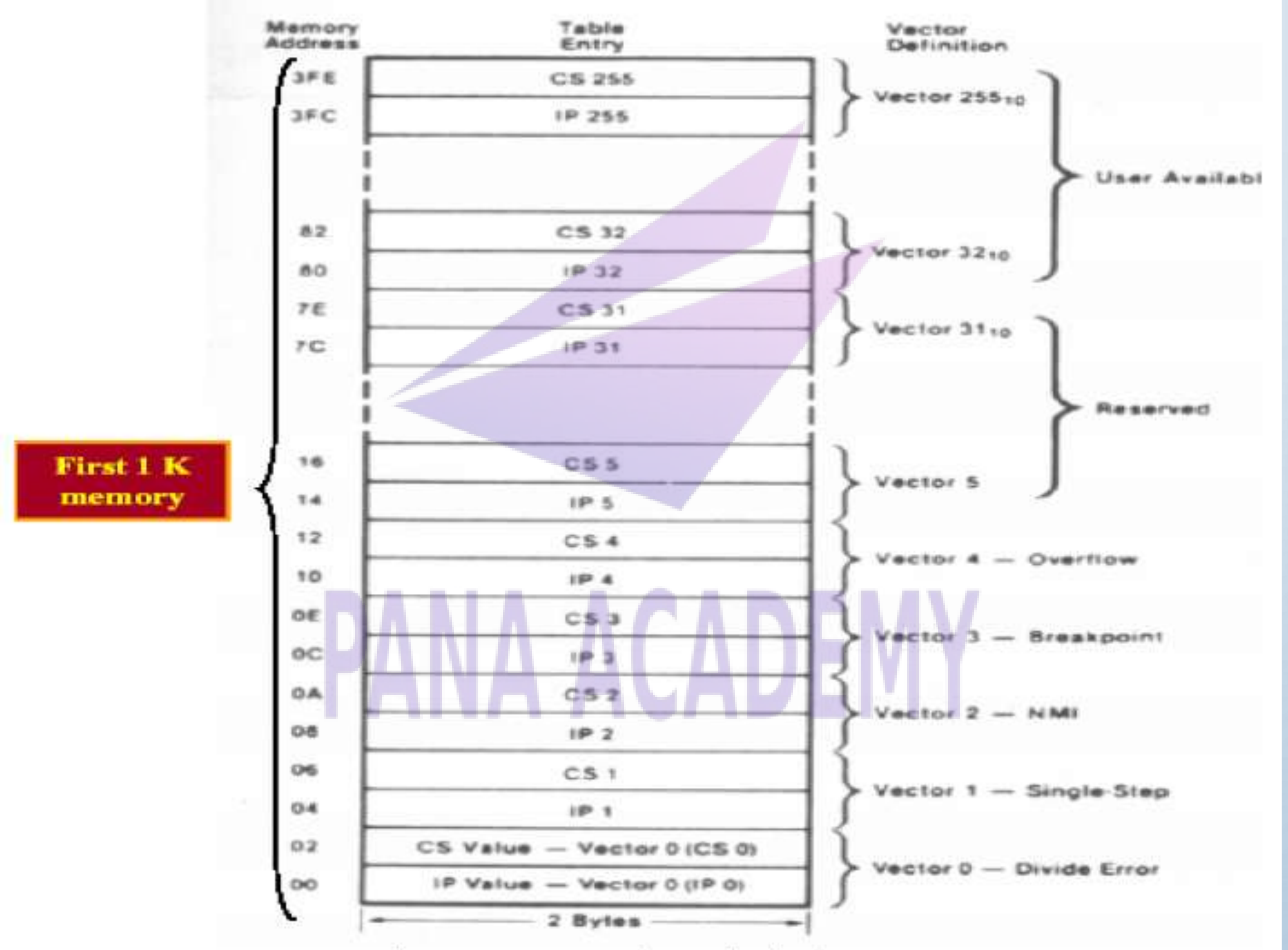

ADDRESS
3FFH
3FCH
AVAILABLE INTERRUPT POINTERS (224)
TYPE 255 POINTER : (AVAILABLE)
TYPE 33 POINTER : (AVAILABLE)
084H
TYPE 32 POINTER : (AVAILABLE)
080H
07FH
TYPE 31 POINTER : (RESERVED)
RESERVED INTERRUPT POINTERS (27)
TYPE 5 POINTER : (RESERVED)
014H
TYPE 4 POINTER : OVERFLOW
010H
TYPE 3 POINTER : 1-BYTE INT INSTRUCTION
00CH
DEDICATED INTERRUPT POINTERS (5)
TYPE 2 POINTER : NON-MASKABLE
008H
TYPE 1 POINTER : SINGLE-STEP
004H
TYPE 0 POINTER : DIVIDE ERROR
000H
CS BASE ADDRESS
IP OFFSET
16 BITS

Fig. 9.2 8086 interrupt vector table

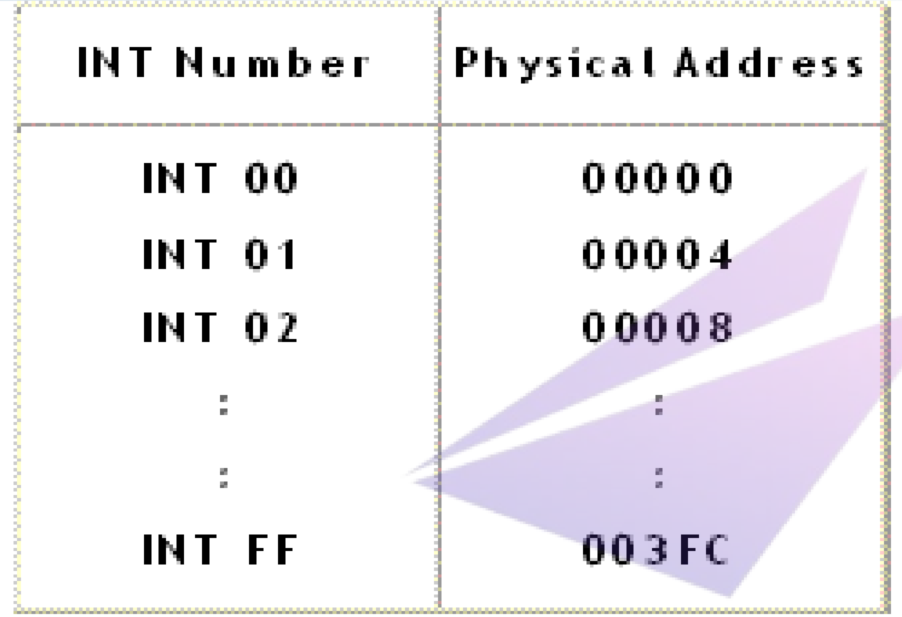
INTERRUPT VECTOR TABLE AND ITS ORGANIZATION



Memory Address	Table Entry	Vector Definition
3FE	CS 255	Vector 255_{10}
3FC	IP 255	
82	CS 32	Vector 32_{10}
80	IP 32	
7E	CS 31	Vector 31_{10}
7C	IP 31	
16	CS 5	Vector 5
14	IP 5	
12	CS 4	Vector 4 — Overflow
10	IP 4	
0E	CS 3	Vector 3 — Breakpoint
0C	IP 3	
0A	CS 2	Vector 2 — NMI
08	IP 2	
06	CS 1	Vector 1 — Single-Step
04	IP 1	
02	CS Value — Vector 0 (CS 0)	Vector 0 — Divide Error
00	IP Value — Vector 0 (IP 0)	

Fig: IVT Structure (Organization)

INTERRUPT VECTOR TABLE AND ITS ORGANIZATION



INT Number	Physical Address
INT 00	00000
INT 01	00004
INT 02	00008
:	:
:	:
INT FF	003FC

DEDICATED INTERRUPTS

INT 00 (divide error)

- INT00 is invoked by the microprocessor whenever there is an attempt to divide a number by zero.
- ISR is responsible for displaying the message “Divide Error” on the screen

INT 01 (Single step interrupt)

- For single stepping the trap flag must be 1
- After execution of each instruction, 8086 automatically jumps to 00004H to fetch 4 bytes for CS: IP of the ISR.
- The job of ISR is to dump the registers on to the screen

DEDICATED INTERRUPTS

INT 02 (Non maskable Interrupt)

- When ever NMI pin of the 8086 is activated by a high signal (5v), the CPU Jumps to physical memory location 00008 to fetch CS:IP of the ISR associated with NMI.

INT 03 (break point)

- A break point is used to examine the cpu and memory after the execution of a group of Instructions.
- It is one byte instruction whereas other instructions of the form “INT nn” are 2 byte instructions.

INT 04 (Signed number overflow)

- There is an instruction associated with this INT 0 (interrupt on overflow).
- If INT 0 is placed after a signed number arithmetic as IMUL or ADD the CPU will activate INT 04 if OF = 1. (OF = Overflow Flag)
- In case where OF = 0 , the INT 0 is not executed but is bypassed and acts as a NOP.

SOFTWARE AND HARDWARE INTERRUPT

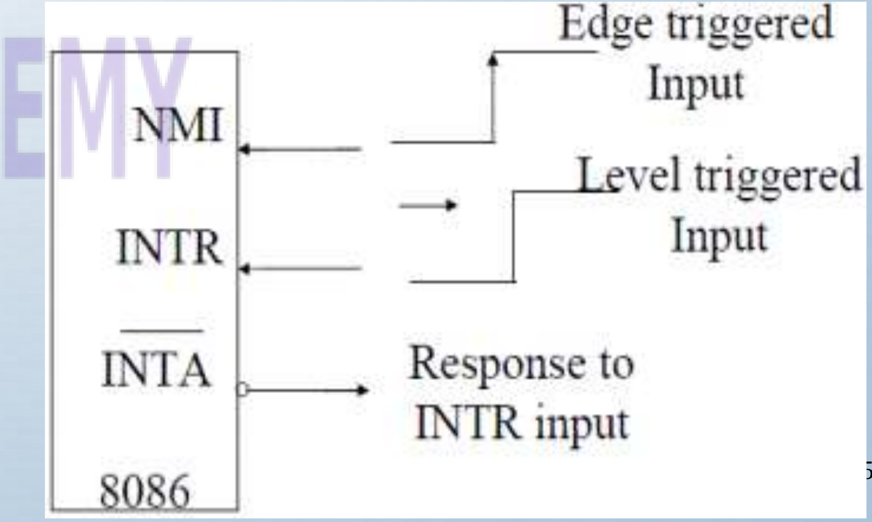
- **Types of Interrupts**:
  - There are two types of Interrupts in 8086. They are:

1) Hardware Interrupts (External Interrupts).

- The Intel microprocessors support hardware interrupts through:
  - Two pins that allow interrupt requests, INTR and NMI
  - One pin that acknowledges, INTA, the interrupt requested on INTR.

Performance of Hardware Interrupts

- NMI : Non maskable interrupts - TYPE 2 Interrupt
- INTR : Interrupt request - Between 20H and FFH



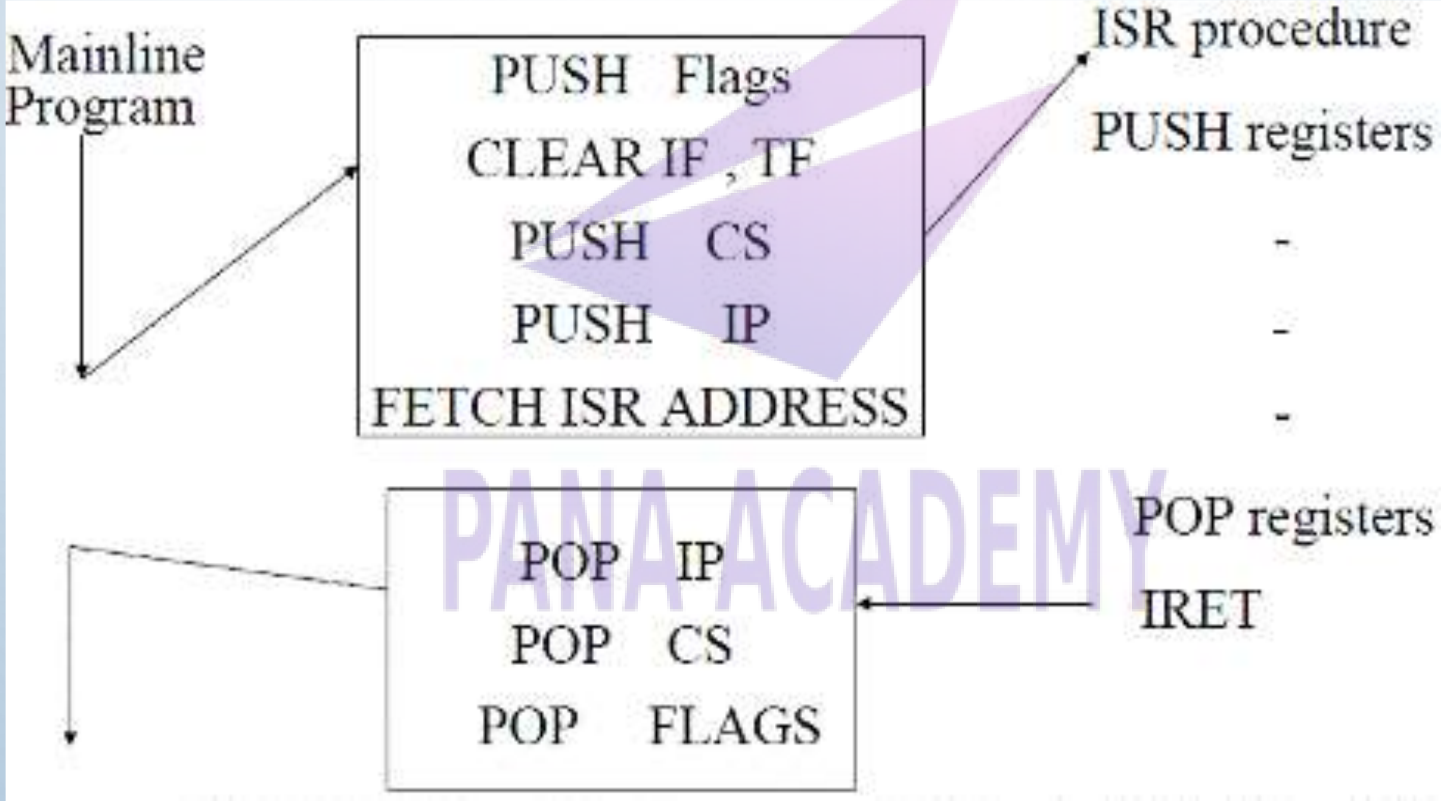
SOFTWARE AND HARDWARE INTERRUPT

2) Software Interrupts (Internal Interrupts and Instructions) .

- Software interrupts can be caused by:
- INT instruction - breakpoint interrupt. This is a type 3 interrupt.
- INT <interrupt number> instruction - any one interrupt from available 256 interrupts.
- INT0 instruction - interrupt on overflow
- Single-step interrupt - generated if the TF flag is set. This is a type 1 interrupt. When the CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.
- Processor exceptions: Divide Error (Type 0), Unused Opcode (type 6) and Escape opcode (type 7).
- Software interrupt processing is the same as for the hardware interrupts.
- Ex: INT n (Software Instructions)
- Control is provided through:
  - IF and TF flag bits
  - IRET and IRETD

SOFTWARE AND HARDWARE INTERRUPT

Performance of Software Interrupts



Mainline Program

PUSH Flags

CLEAR IF , TF

PUSH CS

PUSH IP

FETCH ISR ADDRESS

ISR procedure

PUSH registers

-

-

-

POP registers

IRET

POP IP

POP CS

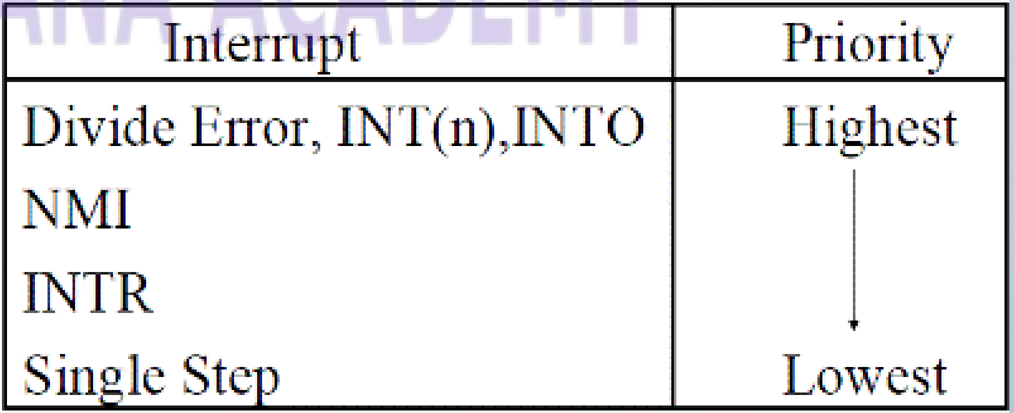
POP FLAGS

SOFTWARE AND HARDWARE INTERRUPT

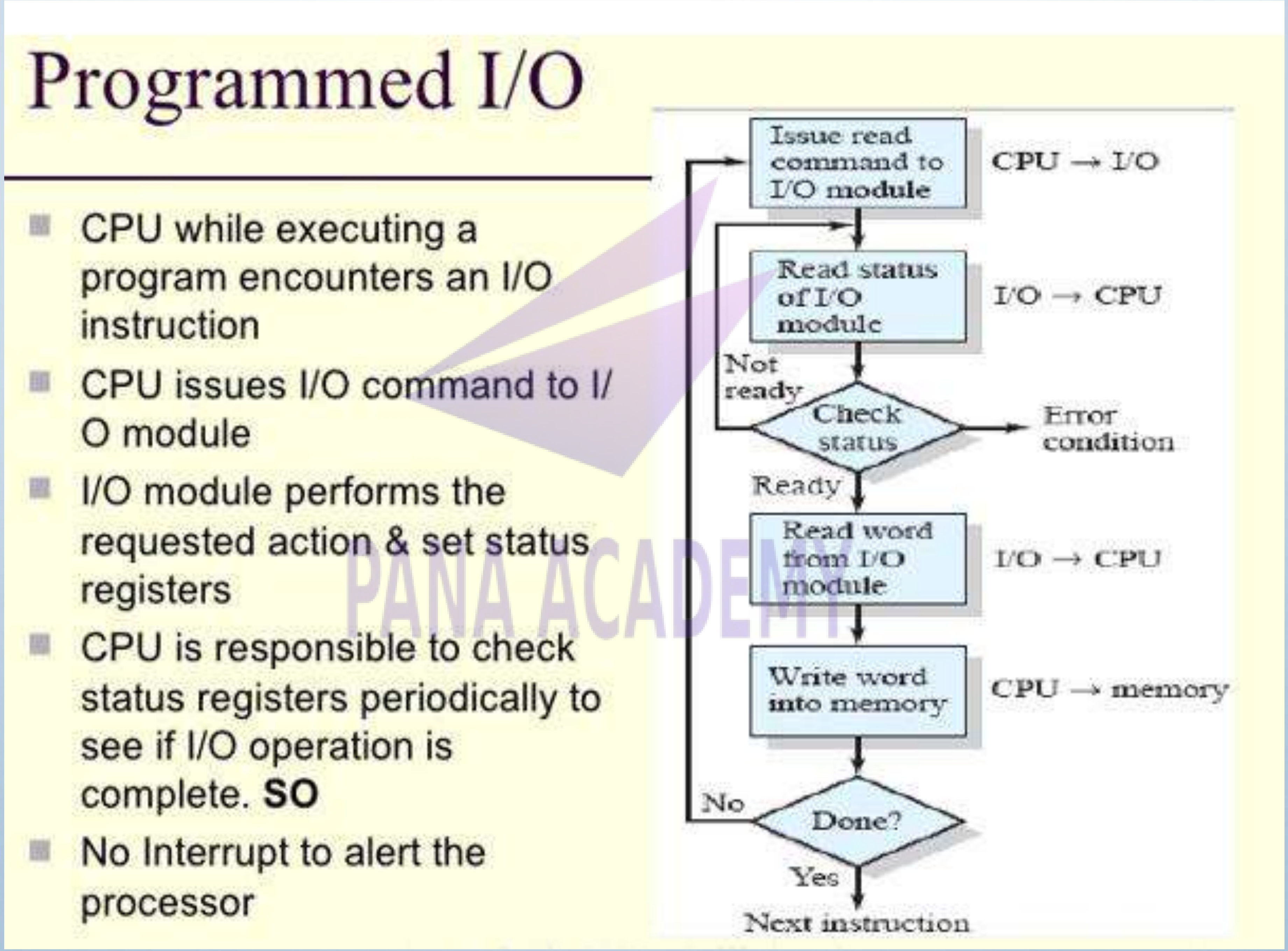
Performance of Software Interrupts

- It decrements SP by 2 and pushes the flag register on the stack.
- Disables INTR by clearing the IF.
- It resets the TF in the flag Register.
- It decrements SP by 2 and pushes CS on the stack.
- It decrements SP by 2 and pushes IP on the stack.
- Fetch the ISR address from the interrupt vector table.

Interrupt Priorities

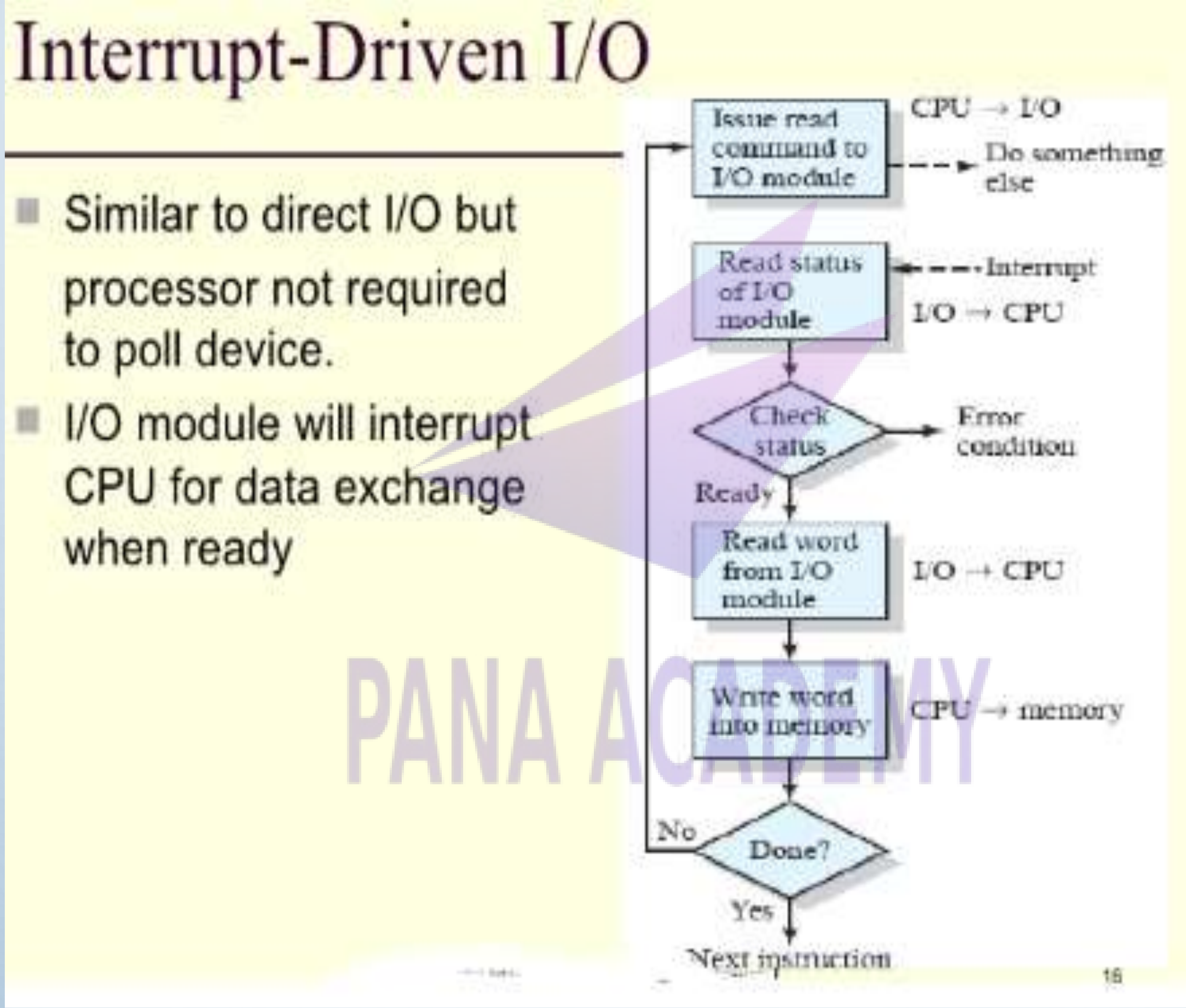


Interrupt	Priority
Divide Error, INT(n),INTO	Highest
NMI	↓
INTR	
Single Step	Lowest



Programmed I/O

- CPU while executing a program encounters an I/O instruction
- CPU issues I/O command to I/O module
- I/O module performs the requested action & set status registers
- CPU is responsible to check status registers periodically to see if I/O operation is complete. **SO**
- No Interrupt to alert the processor



Interrupt-Driven I/O

- Similar to direct I/O but processor not required to poll device.
- I/O module will interrupt CPU for data exchange when ready

Issue read command to I/O module

CPU → I/O

Do something else

Read status of I/O module

Interrupt

I/O → CPU

Check status

Error condition

Ready

Read word from I/O module

I/O → CPU

Write word into memory

CPU → memory

Done?

No

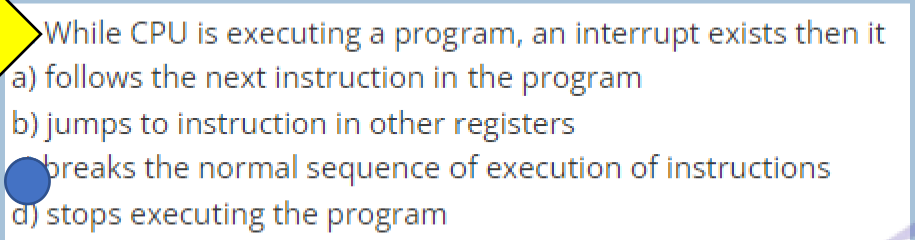
Yes

Next instruction

16

Interrupt operations

1



While CPU is executing a program, an interrupt exists then it

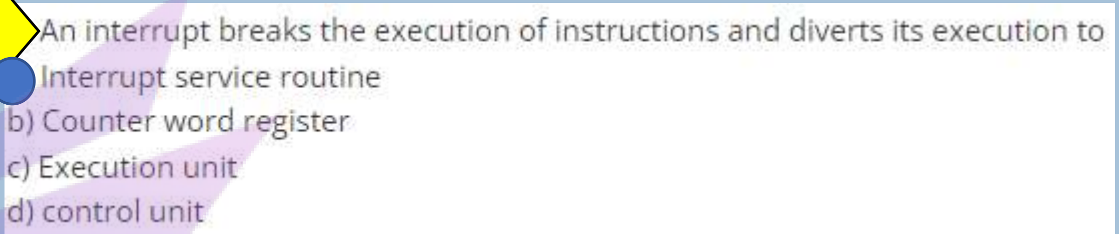
a) follows the next instruction in the program

b) jumps to instruction in other registers

[✓ breaks the normal sequence of execution of instructions]

d) stops executing the program

2



An interrupt breaks the execution of instructions and diverts its execution to

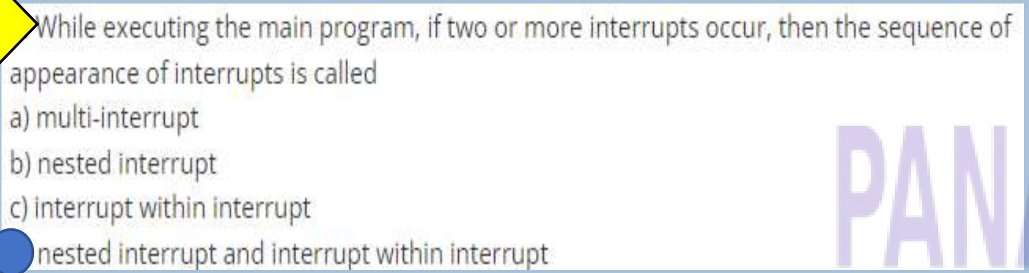
[✓ Interrupt service routine]

b) Counter word register

c) Execution unit

d) control unit

3



While executing the main program, if two or more interrupts occur, then the sequence of appearance of interrupts is called

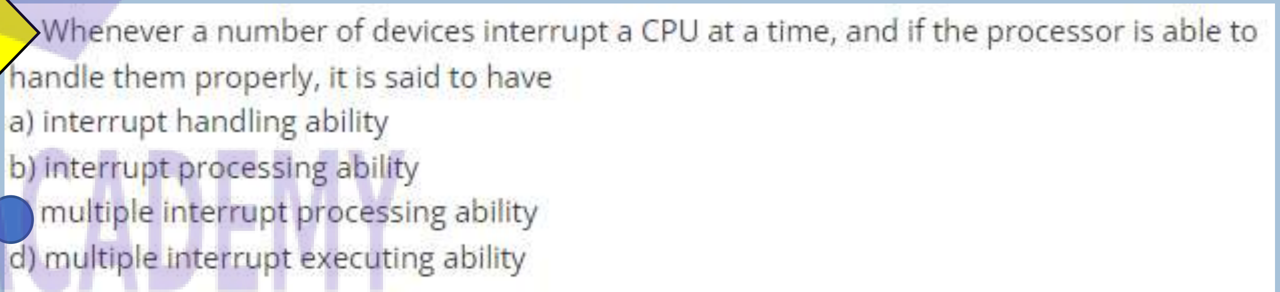
a) multi-interrupt

b) nested interrupt

c) interrupt within interrupt

[✓ nested interrupt and interrupt within interrupt]

4



Whenever a number of devices interrupt a CPU at a time, and if the processor is able to handle them properly, it is said to have

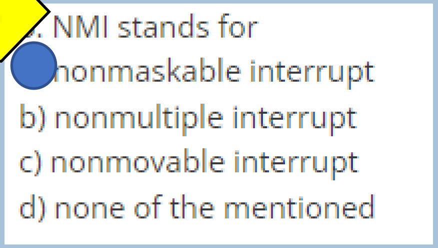
a) interrupt handling ability

b) interrupt processing ability

[✓ multiple interrupt processing ability]

d) multiple interrupt executing ability

5



. NMI stands for

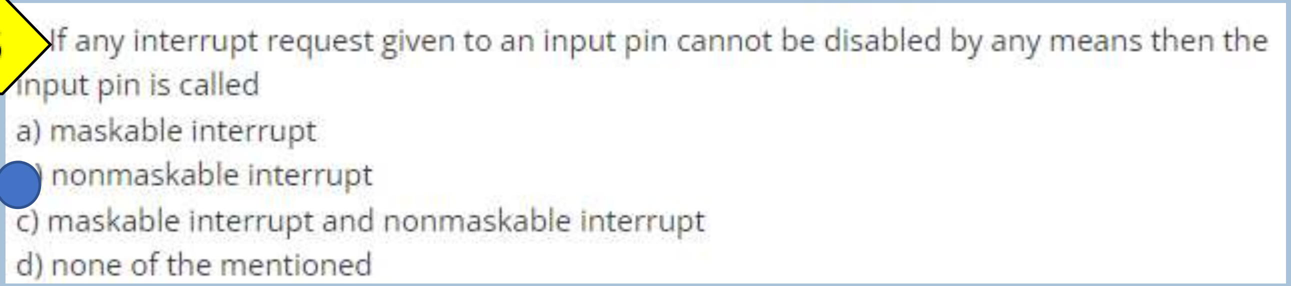
[✓ nonmaskable interrupt]

b) nonmultiple interrupt

c) nonmovable interrupt

d) none of the mentioned

6



If any interrupt request given to an input pin cannot be disabled by any means then the input pin is called

a) maskable interrupt

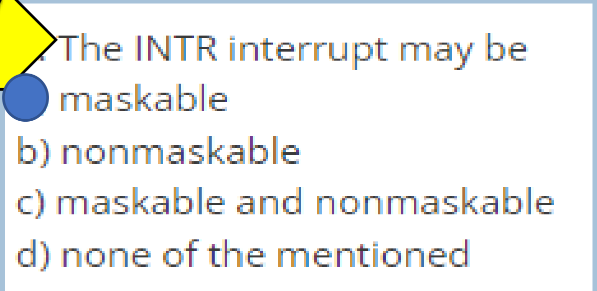
[✓ nonmaskable interrupt]

c) maskable interrupt and nonmaskable interrupt

d) none of the mentioned

Interrupt operations

7



The INTR interrupt may be

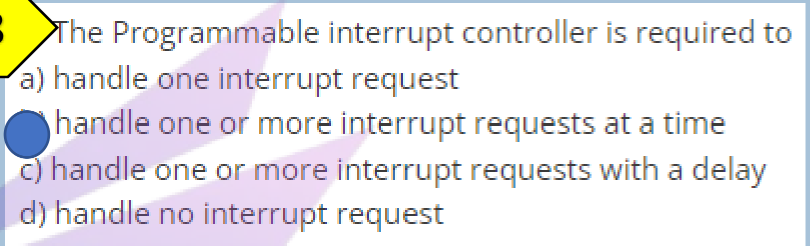
[✓ maskable]

b) nonmaskable

c) maskable and nonmaskable

d) none of the mentioned

8



The Programmable interrupt controller is required to

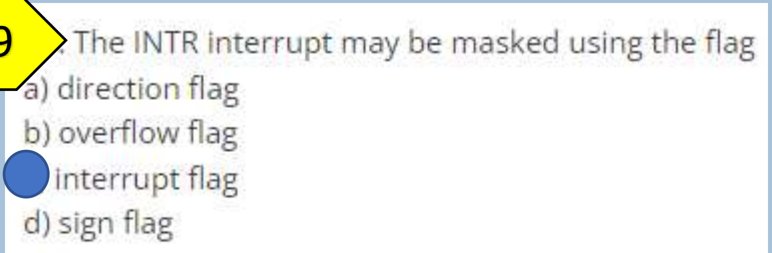
a) handle one interrupt request

[✓ handle one or more interrupt requests at a time]

c) handle one or more interrupt requests with a delay

d) handle no interrupt request

9



The INTR interrupt may be masked using the flag

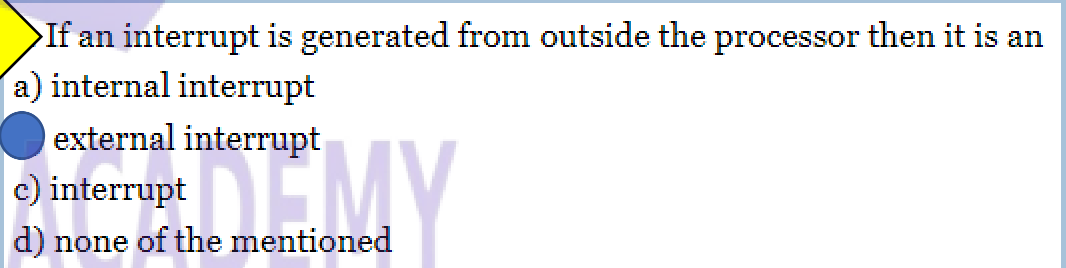
a) direction flag

b) overflow flag

[✓ interrupt flag]

d) sign flag

10



If an interrupt is generated from outside the processor then it is an

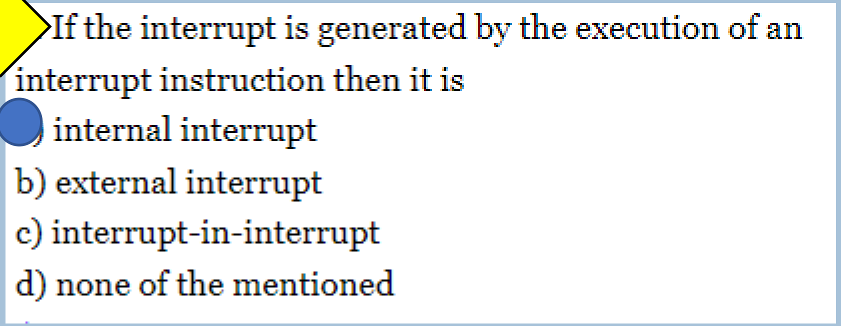
a) internal interrupt

[✓ external interrupt]

c) interrupt

d) none of the mentioned

11



If the interrupt is generated by the execution of an interrupt instruction then it is

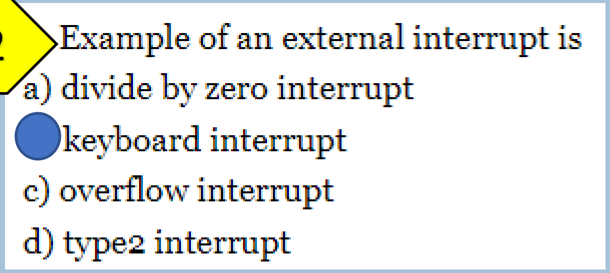
[✓ internal interrupt]

b) external interrupt

c) interrupt-in-interrupt

d) none of the mentioned

12



Example of an external interrupt is

a) divide by zero interrupt

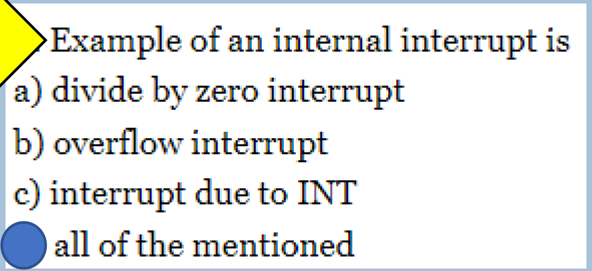
[✓ keyboard interrupt]

c) overflow interrupt

d) type2 interrupt

Interrupt operations

13


Example of an internal interrupt is

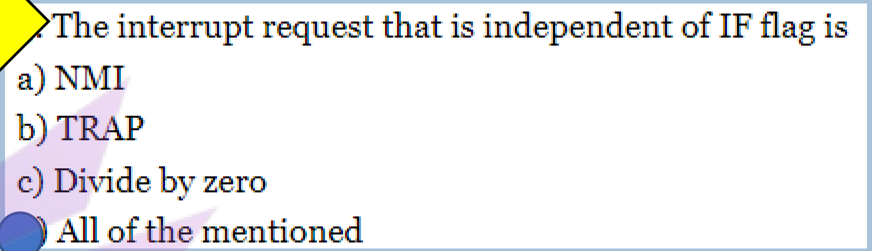
a) divide by zero interrupt

b) overflow interrupt

c) interrupt due to INT

[✓ all of the mentioned]

14


The interrupt request that is independent of IF flag is

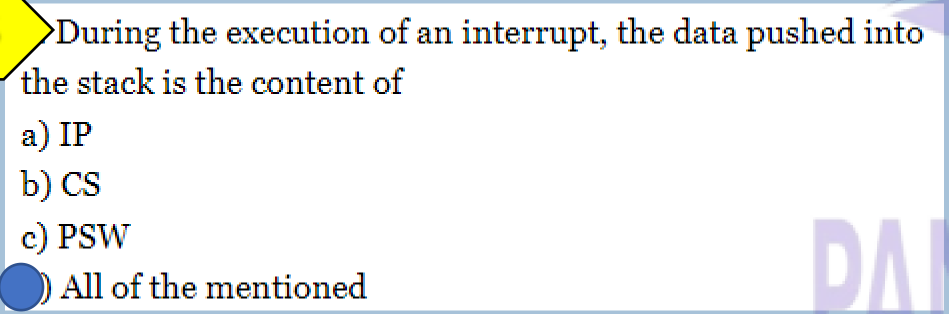
a) NMI

b) TRAP

c) Divide by zero

[✓) All of the mentioned]

15


During the execution of an interrupt, the data pushed into the stack is the content of

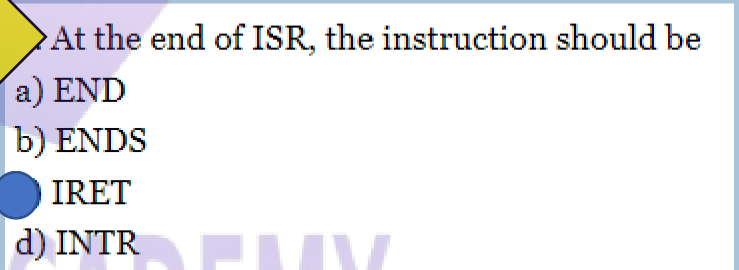
a) IP

b) CS

c) PSW

[✓) All of the mentioned]

16


At the end of ISR, the instruction should be

a) END

b) ENDS

[✓ IRET]

d) INTR

17

The register that stores all the interrupt requests in it in order to serve them one by one on a priority basis is

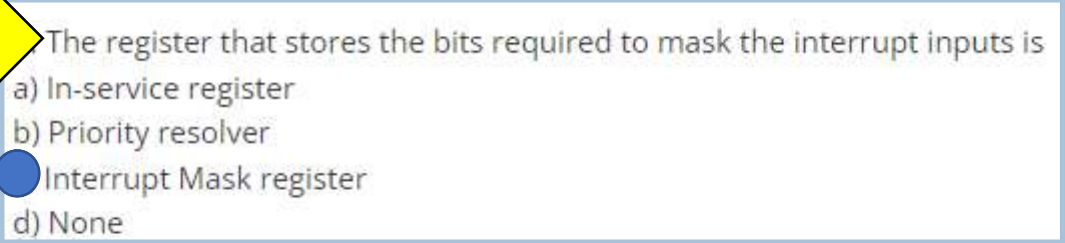
[✓ Interrupt Request Register]

b) In-Service Register

c) Priority resolver

d) Interrupt Mask Register

18


The register that stores the bits required to mask the interrupt inputs is

a) In-service register

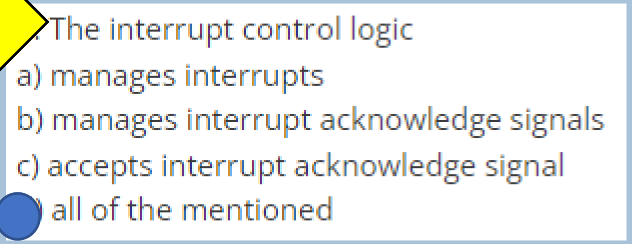
b) Priority resolver

[✓ Interrupt Mask register]

d) None

Interrupt operations

19



The interrupt control logic

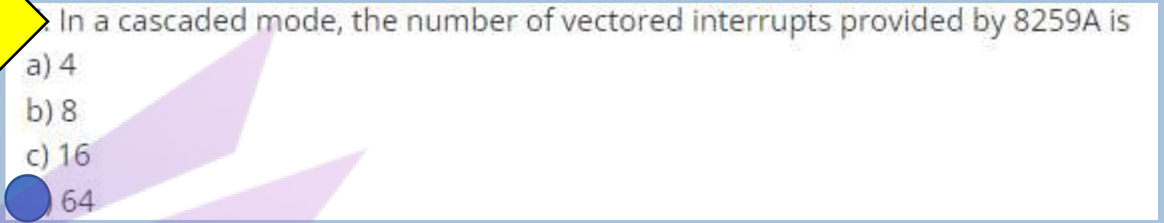
a) manages interrupts

b) manages interrupt acknowledge signals

c) accepts interrupt acknowledge signal

[✓) all of the mentioned]

20



In a cascaded mode, the number of vectored interrupts provided by 8259A is

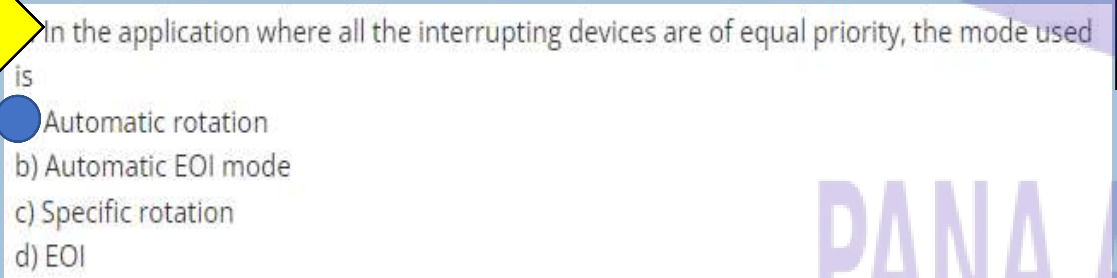
a) 4

b) 8

c) 16

[✓) 64]

21



In the application where all the interrupting devices are of equal priority, the mode used is

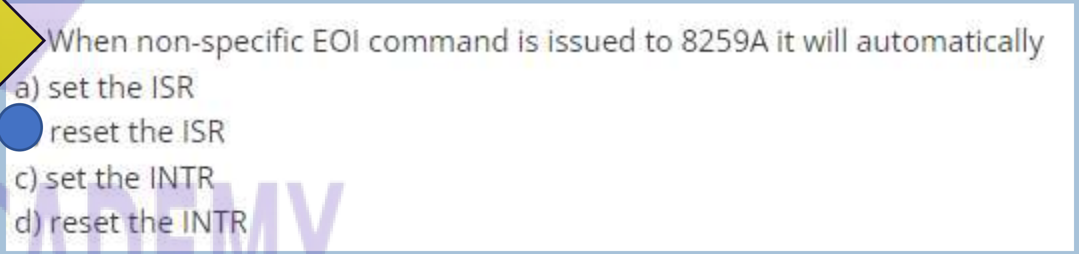
[✓ Automatic rotation]

b) Automatic EOI mode

c) Specific rotation

d) EOI

22



When non-specific EOI command is issued to 8259A it will automatically

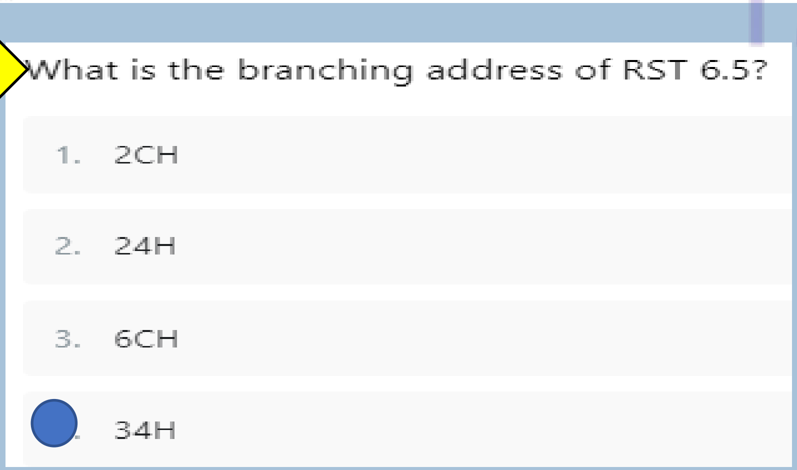
a) set the ISR

[✓ reset the ISR]

c) set the INTR

d) reset the INTR

23

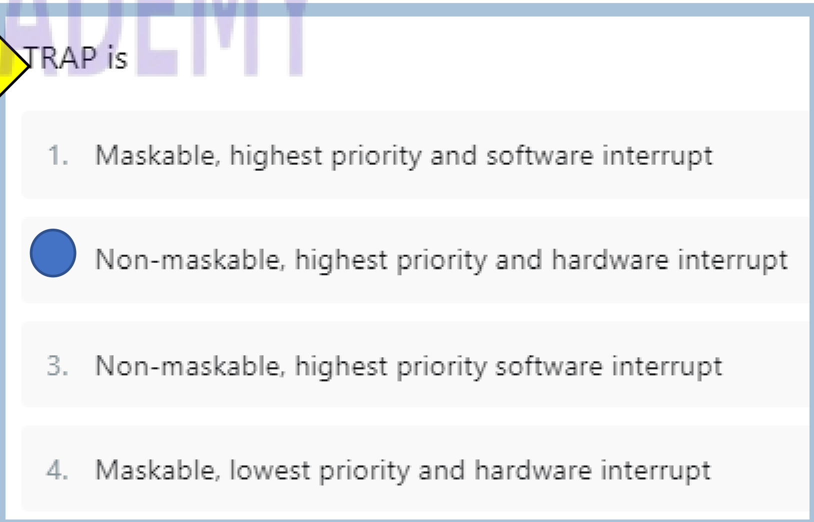


What is the branching address of RST 6.5?

1. 2CH
2. 24H
3. 6CH

[✓ 34H]

24



TRAP is

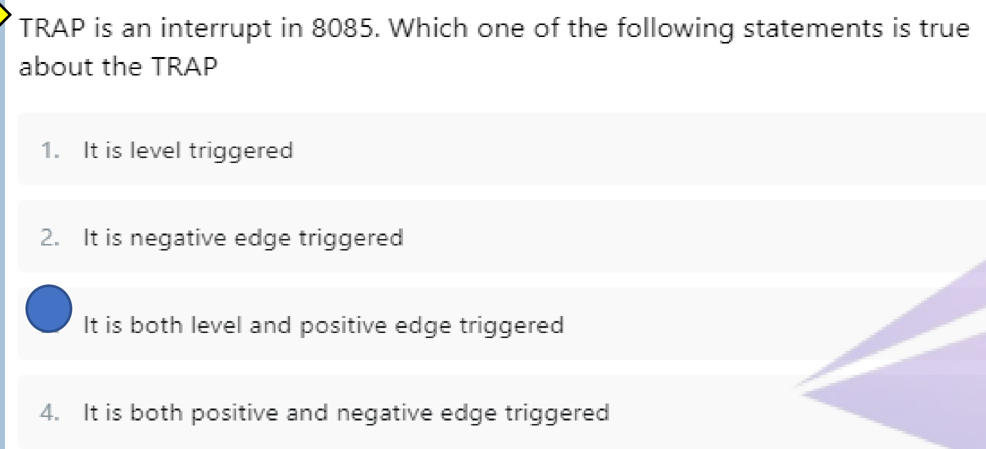
1. Maskable, highest priority and software interrupt

[✓ Non-maskable, highest priority and hardware interrupt]

3. Non-maskable, highest priority software interrupt
4. Maskable, lowest priority and hardware interrupt

Interrupt operations

25

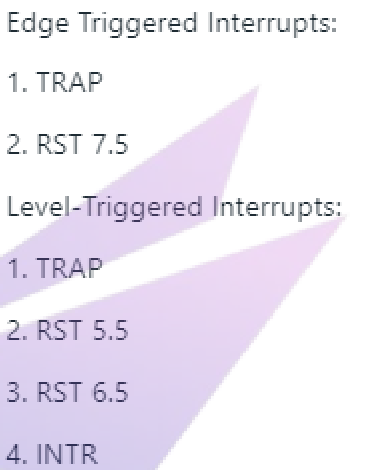


TRAP is an interrupt in 8085. Which one of the following statements is true about the TRAP

1. It is level triggered
2. It is negative edge triggered

● It is both level and positive edge triggered

4. It is both positive and negative edge triggered



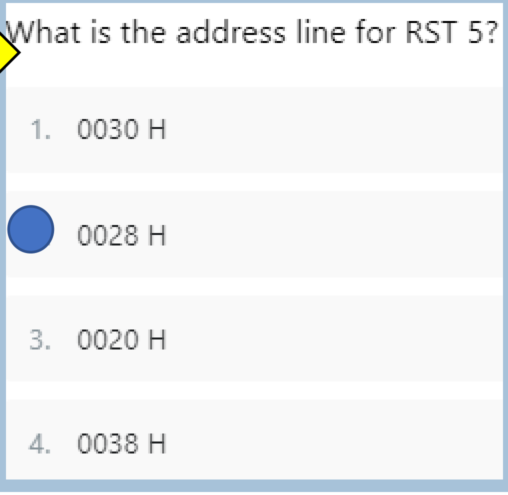
Edge Triggered Interrupts:

1. TRAP
2. RST 7.5

Level-Triggered Interrupts:

1. TRAP
2. RST 5.5
3. RST 6.5
4. INTR

26



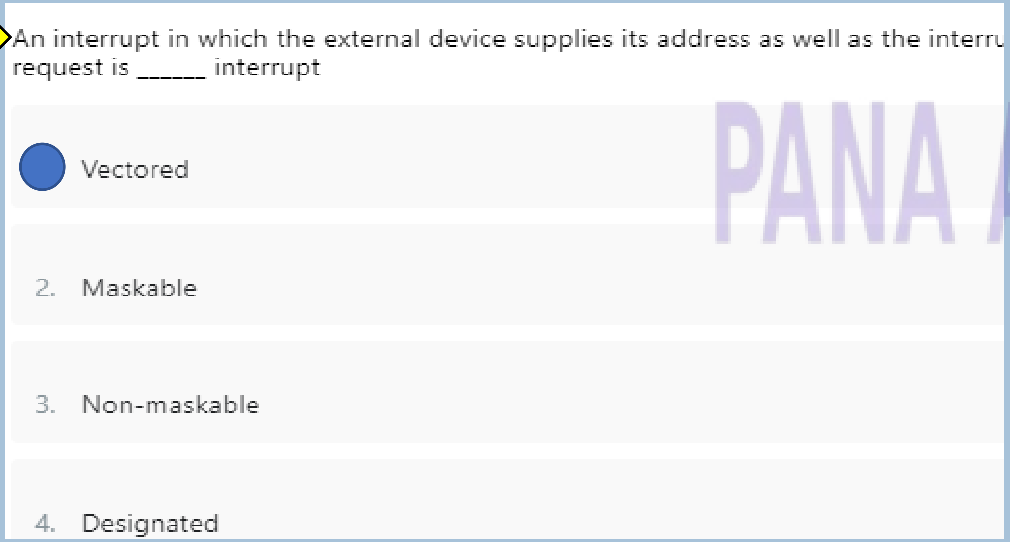
What is the address line for RST 5?

1. 0030 H

● 0028 H

3. 0020 H
4. 0038 H

27

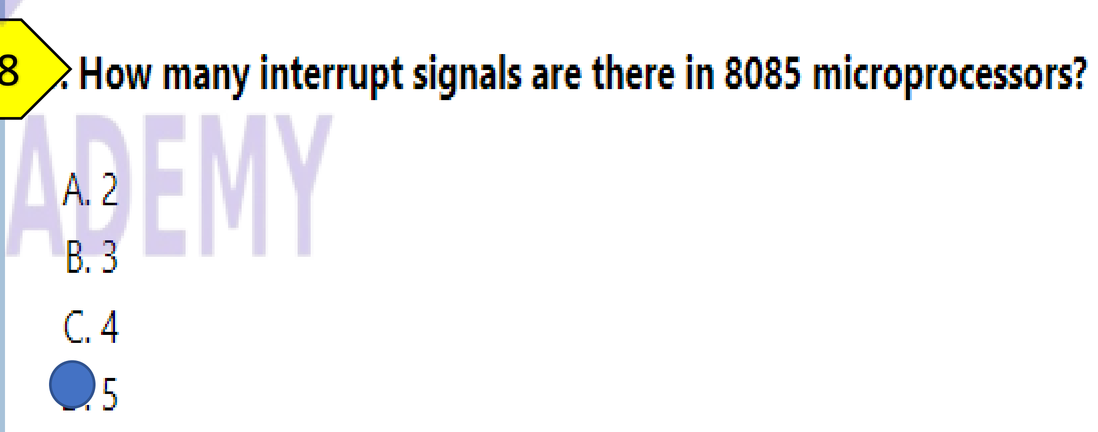


An interrupt in which the external device supplies its address as well as the interru request is ______ interrupt

● Vectored

2. Maskable
3. Non-maskable
4. Designated

28

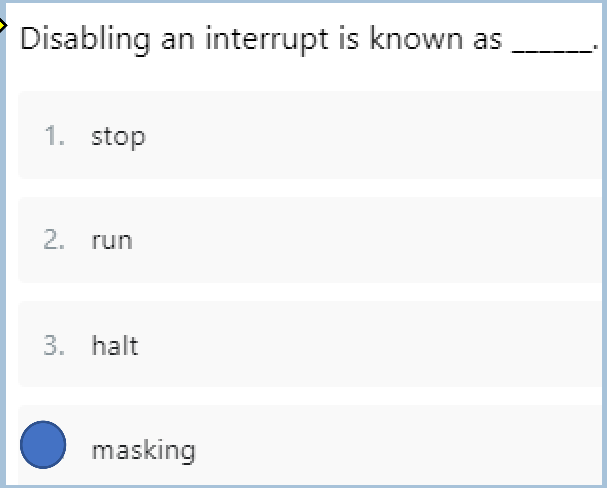


. How many interrupt signals are there in 8085 microprocessors?

A. 2
B. 3
C. 4
● 5

Interrupt operations

29

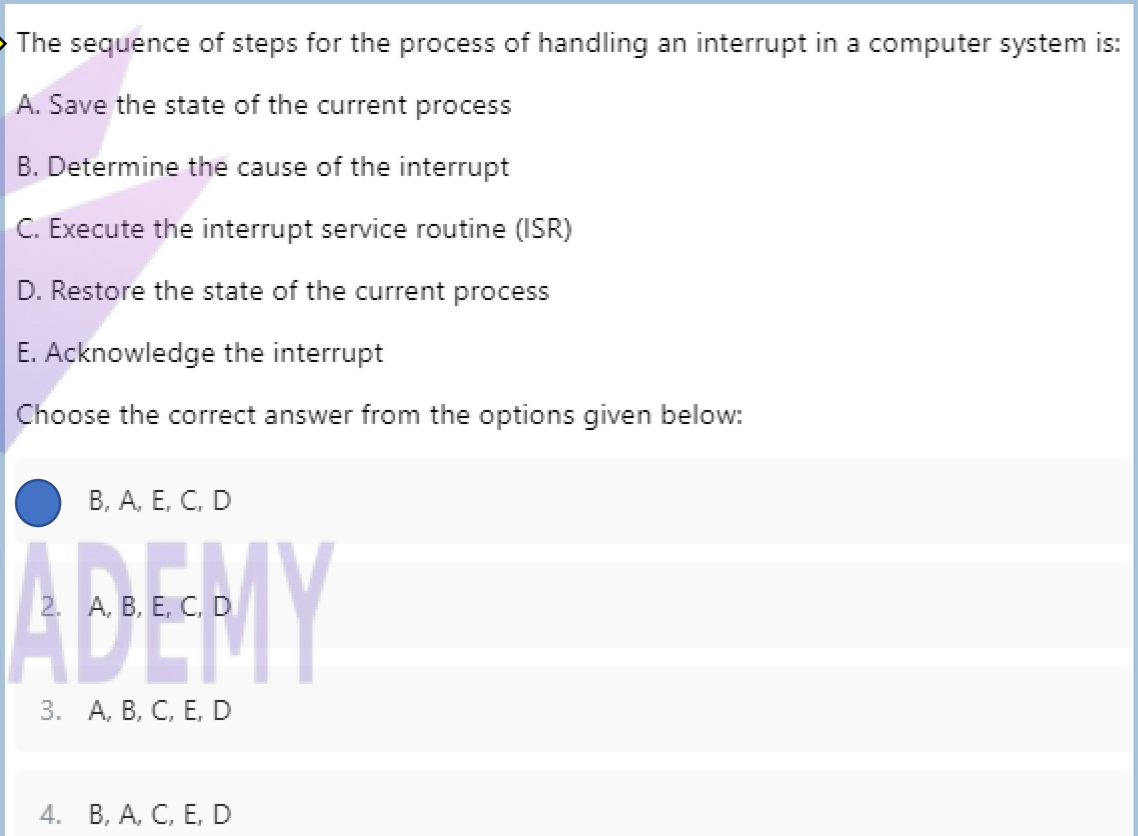


Disabling an interrupt is known as ______.

1. stop
2. run
3. halt

● masking

30



The sequence of steps for the process of handling an interrupt in a computer system is:

A. Save the state of the current process

B. Determine the cause of the interrupt

C. Execute the interrupt service routine (ISR)

D. Restore the state of the current process

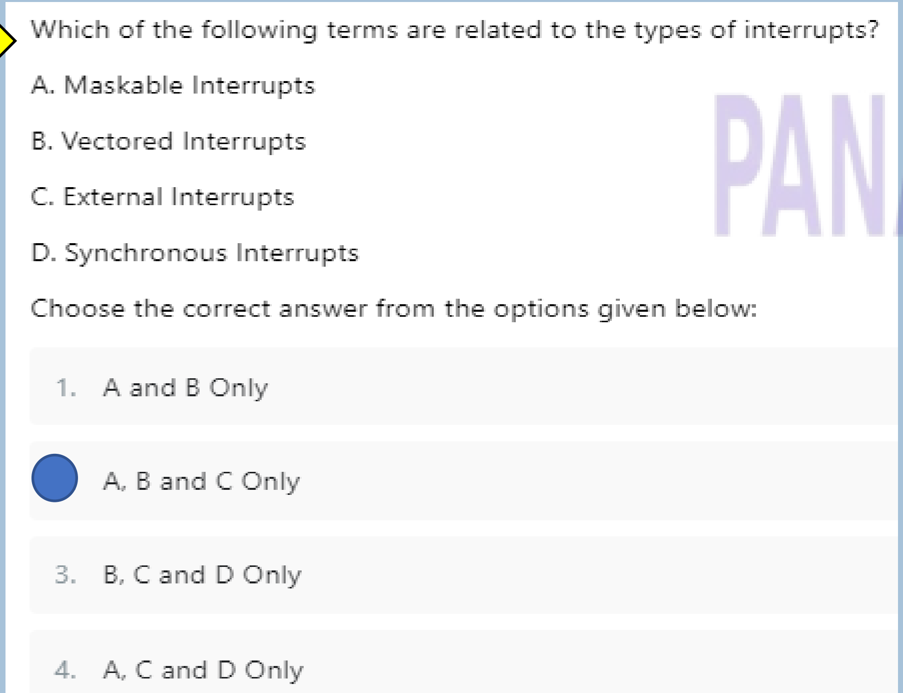
E. Acknowledge the interrupt

Choose the correct answer from the options given below:

● B, A, E, C, D

2. A, B, E, C, D
3. A, B, C, E, D
4. B, A, C, E, D

31



Which of the following terms are related to the types of interrupts?

A. Maskable Interrupts

B. Vectored Interrupts

C. External Interrupts

D. Synchronous Interrupts

Choose the correct answer from the options given below:

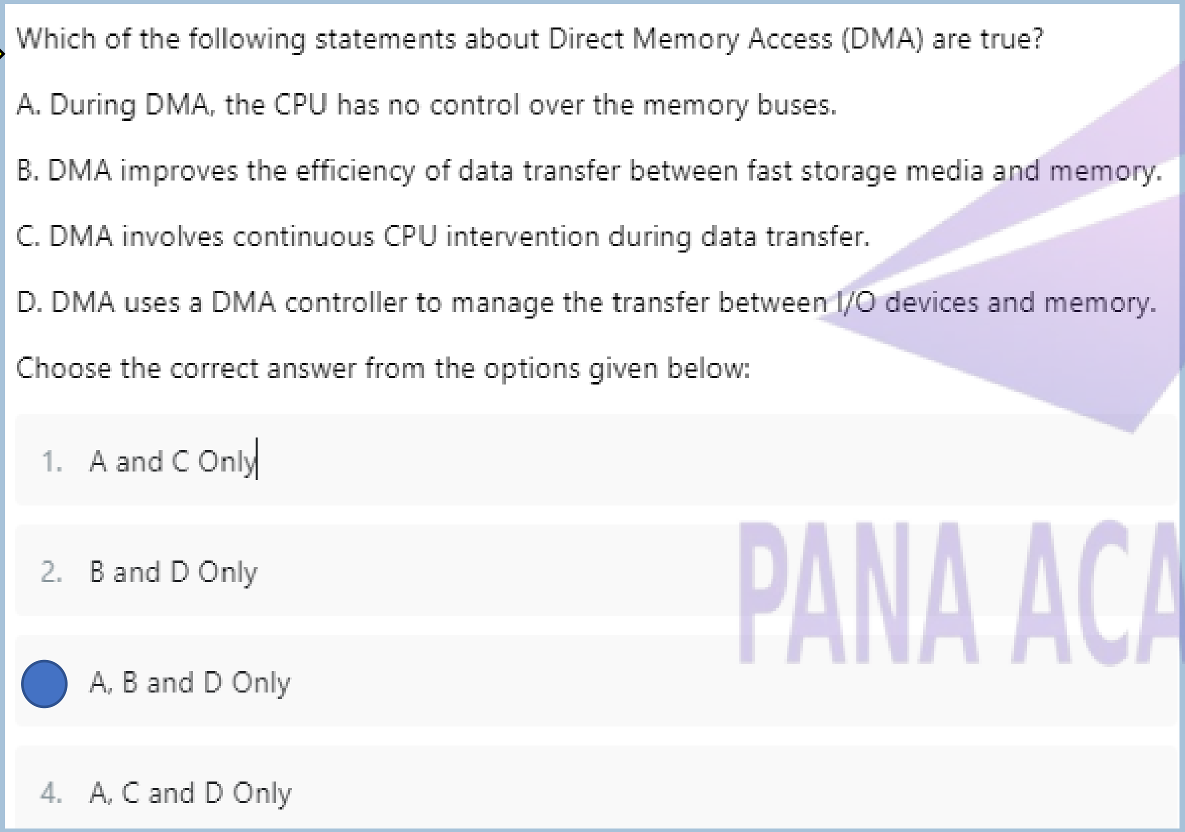
1. A and B Only

● A, B and C Only

3. B, C and D Only
4. A, C and D Only

Interrupt operations

32



Which of the following statements about Direct Memory Access (DMA) are true?

A. During DMA, the CPU has no control over the memory buses.

B. DMA improves the efficiency of data transfer between fast storage media and memory.

C. DMA involves continuous CPU intervention during data transfer.

D. DMA uses a DMA controller to manage the transfer between I/O devices and memory.

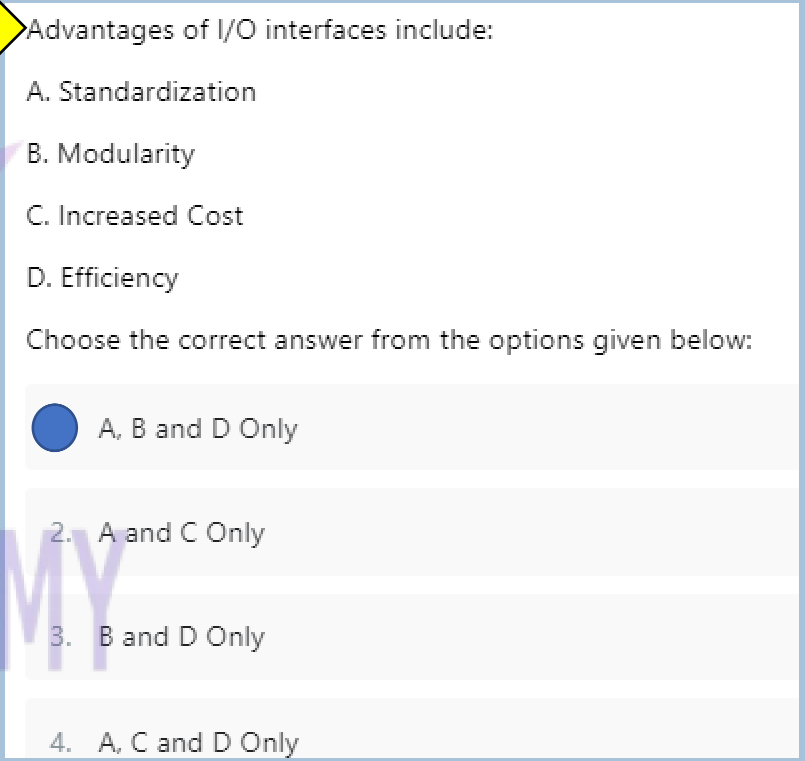
Choose the correct answer from the options given below:

1. A and C Only
2. B and D Only

● A, B and D Only

4. A, C and D Only

33



Advantages of I/O interfaces include:

A. Standardization

B. Modularity

C. Increased Cost

D. Efficiency

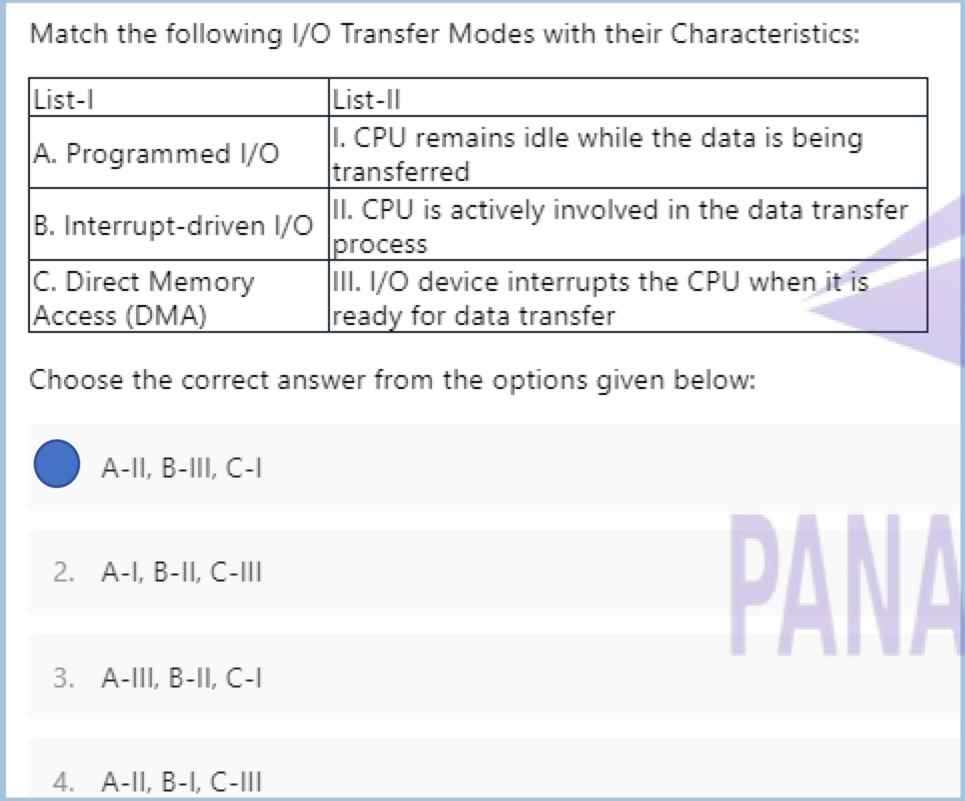
Choose the correct answer from the options given below:

● A, B and D Only

2. A and C Only
3. B and D Only
4. A, C and D Only

Interrupt operations

34



Match the following I/O Transfer Modes with their Characteristics:

List-I	List-II
A. Programmed I/O	I. CPU remains idle while the data is being transferred
B. Interrupt-driven I/O	II. CPU is actively involved in the data transfer process
C. Direct Memory Access (DMA)	III. I/O device interrupts the CPU when it is ready for data transfer

Choose the correct answer from the options given below:

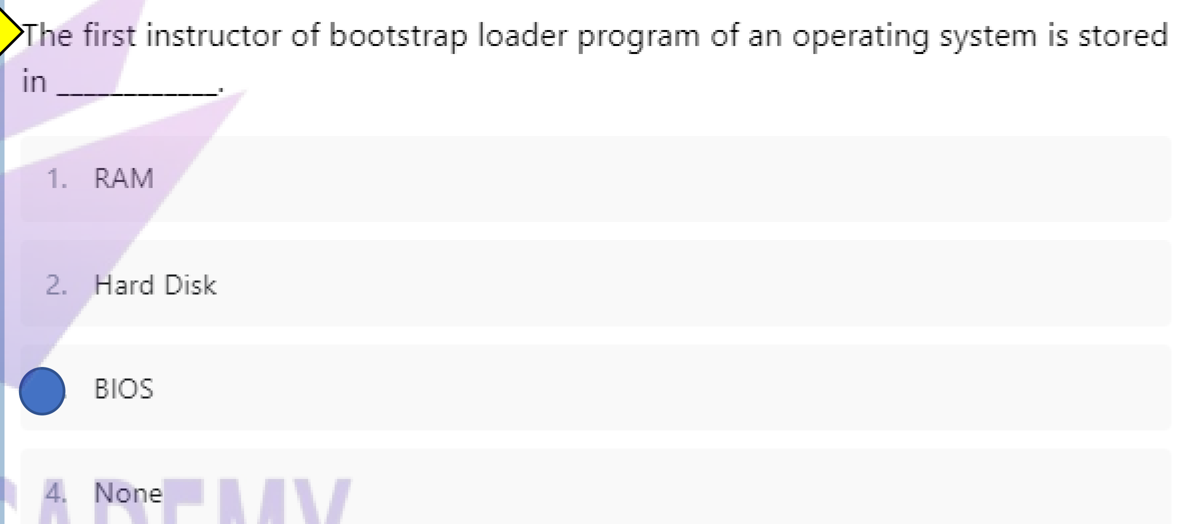
● A-II, B-III, C-I

2. A-I, B-II, C-III

3. A-III, B-II, C-I

4. A-II, B-I, C-III

35



The first instructor of bootstrap loader program of an operating system is stored in __________.

1. RAM

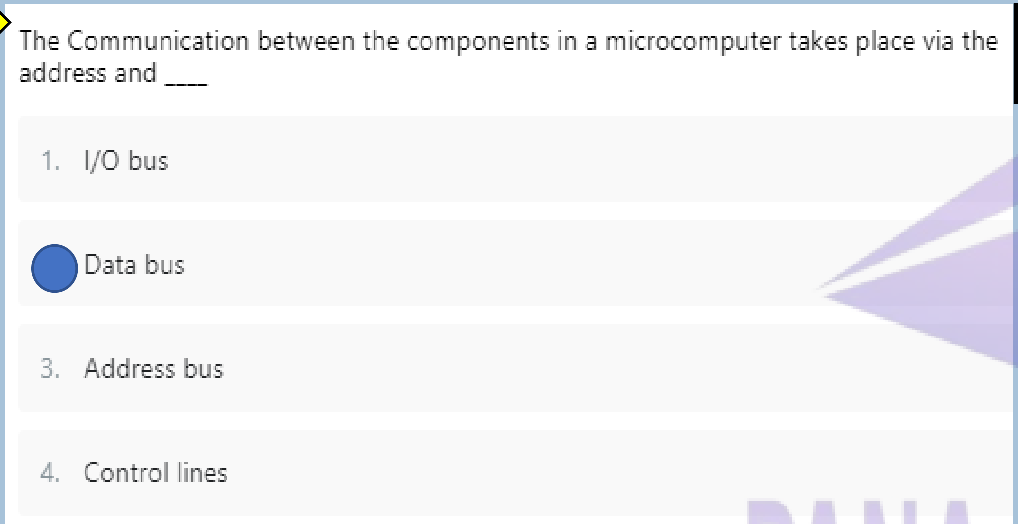
2. Hard Disk

● BIOS

4. None

Interrupt operations

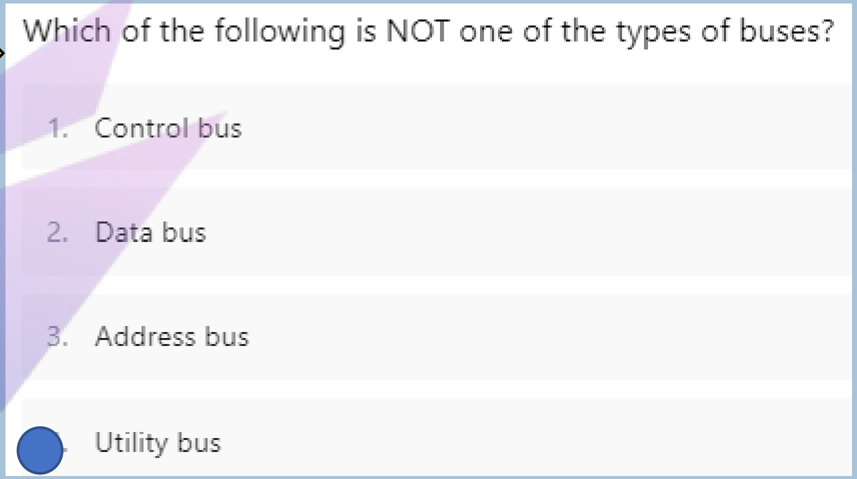
36



The Communication between the components in a microcomputer takes place via the address and ____

1. I/O bus
2. [✓ Data bus]
3. Address bus
4. Control lines

37

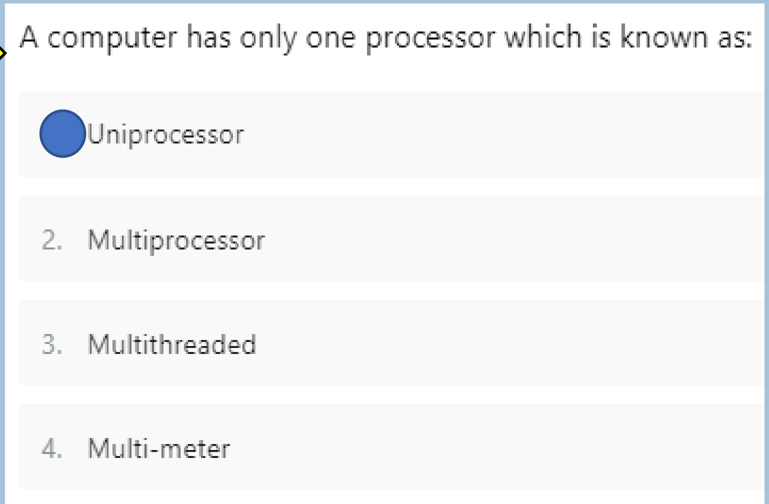


Which of the following is NOT one of the types of buses?

1. Control bus
2. Data bus
3. Address bus
4. [✓ Utility bus]

Interrupt operations

38



A computer has only one processor which is known as:

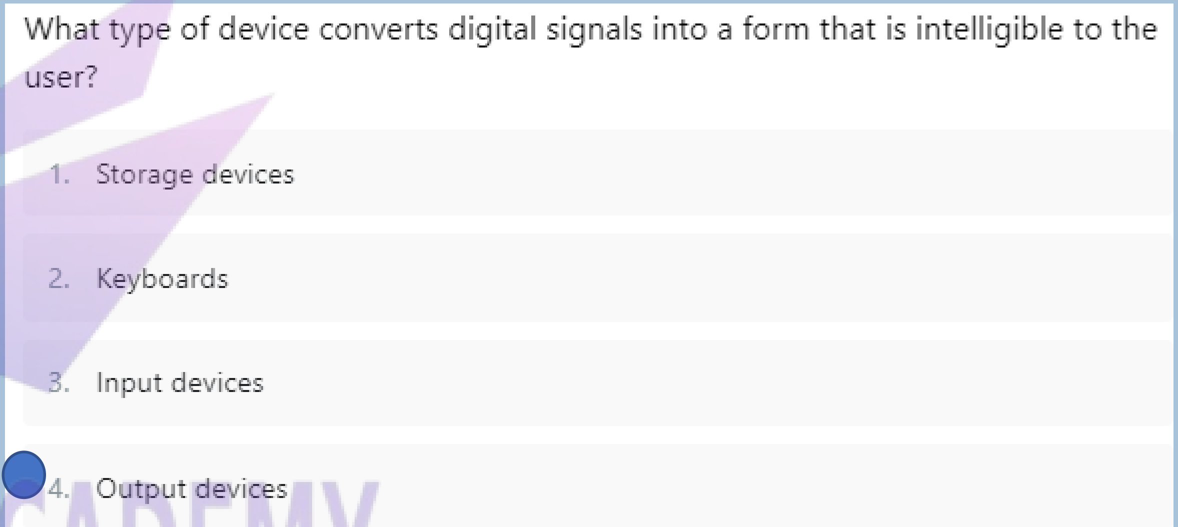
[✓ Uniprocessor]

2. Multiprocessor

3. Multithreaded

4. Multi-meter

39



What type of device converts digital signals into a form that is intelligible to the user?

1. Storage devices

2. Keyboards

3. Input devices

[✓ 4. Output devices]

Interrupt operations

40 Which interrupt has the highest priority in the 8085 microprocessor?

a) INTR

b) RST 7.5

c) RST 6.5

[✓ d) TRAP]

41 The RST 7.5 interrupt is:

a) Non-maskable

[✓ b) Maskable and edge-triggered]

c) Maskable and level-triggered

d) None of the above

Interrupt operations

42 Which instruction is used to enable all interrupts in the 8085 microprocessor?

a) RIM

b) NOP

[✓ c) EI]

d) DI

43 What is the function of the Interrupt Flag (IF) in the 8086 microprocessor?

a) Enables the NMI

[✓ b) Enables/disables maskable interrupts]

c) Clears pending interrupts

d) None of the above

Interrupt operations

44 Which interrupt in the 8086 microprocessor is a non-maskable? interrupt?

a) INTR

b) RST

[✓ c) NMI]

d) INT 0

45 When the PS(active low)/EN(active low) pin of 8259A used in buffered mode, then it can be used as a

a) input to designate chip is master or slave

[✓ b) buffer enable]

c) buffer disable

d) none

When the pin is used in buffered mode, then it can be used as a buffer enable to control buffer transreceivers. If it is not used in buffered mode, then the pin is used as input to designate whether the chip is used as a master or a slave

Interrupt operations

46 The interrupt vector table in the 8086 microprocessor starts at:

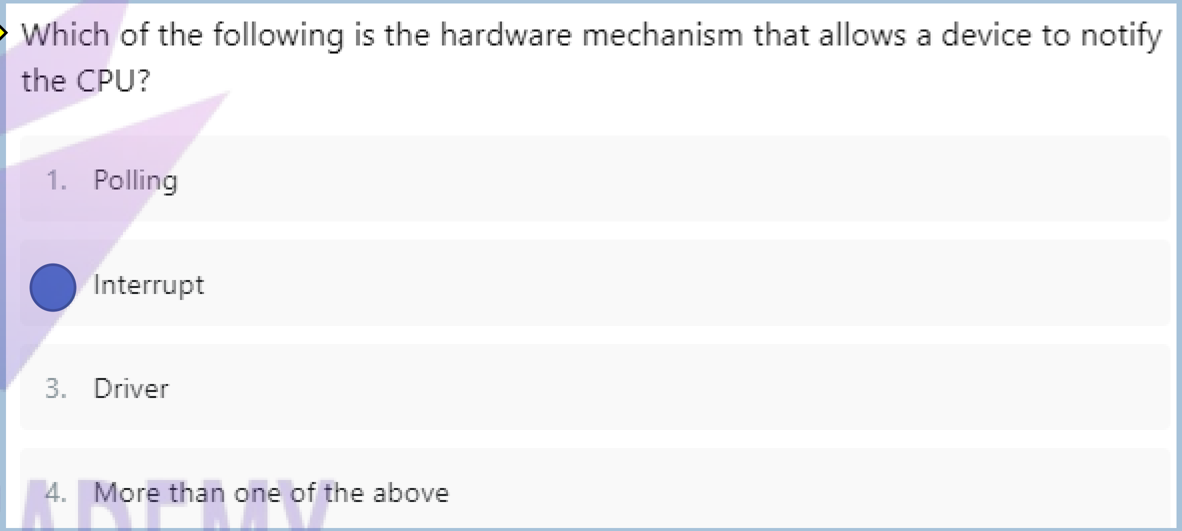
a) 1000H

b) 8000H

[✓) **0000H**]

d) FFFFH

47



Which of the following is the hardware mechanism that allows a device to notify the CPU?

1. Polling

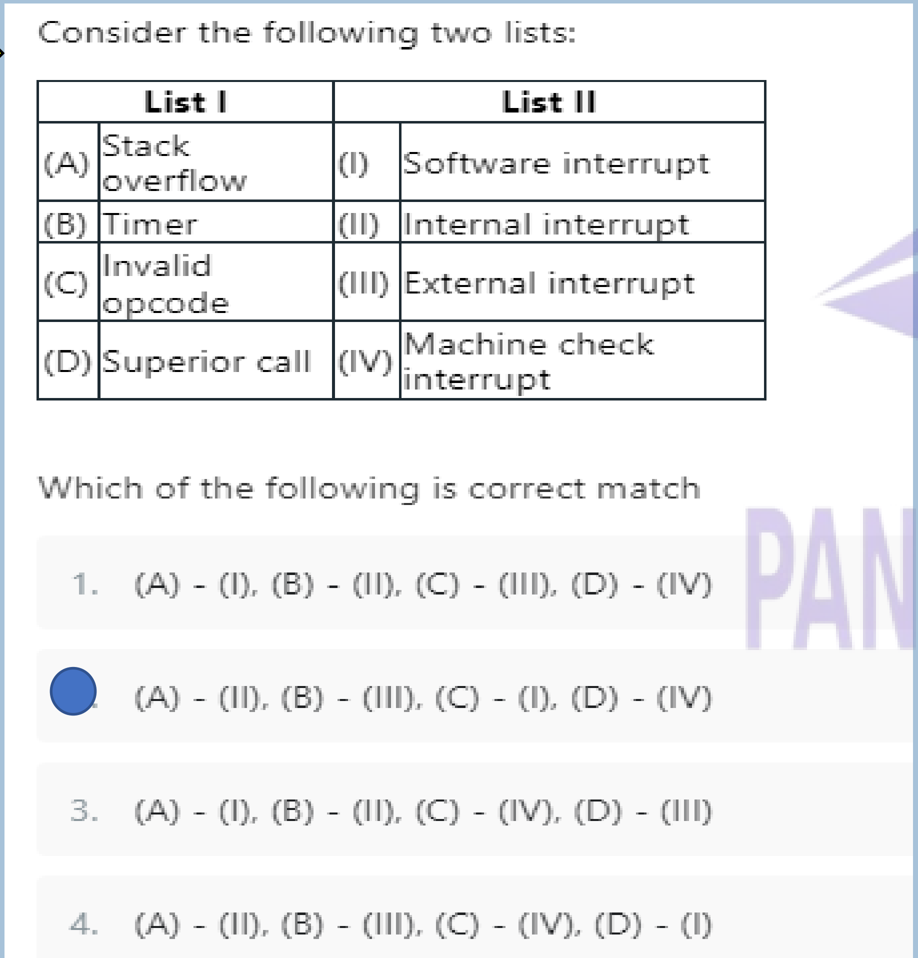
[✓ Interrupt]

3. Driver

4. More than one of the above

Interrupt operations

48



Consider the following two lists:

List I		List II	
(A)	Stack overflow	(I)	Software interrupt
(B)	Timer	(II)	Internal interrupt
(C)	Invalid opcode	(III)	External interrupt
(D)	Superior call	(IV)	Machine check interrupt

Which of the following is correct match

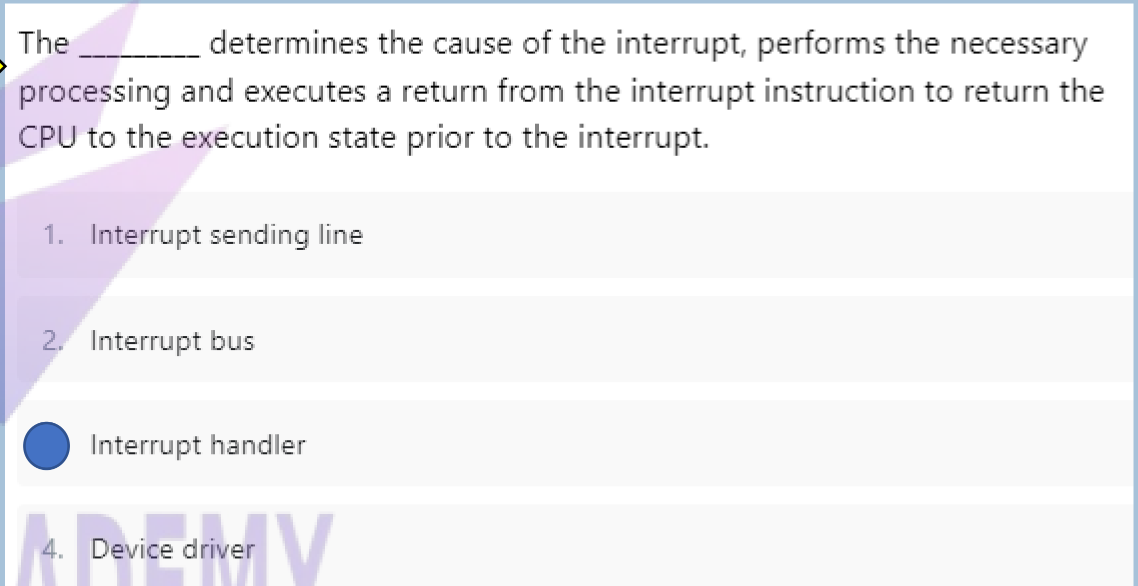
1. (A) - (I), (B) - (II), (C) - (III), (D) - (IV)

[✓ (A) - (II), (B) - (III), (C) - (I), (D) - (IV)]

3. (A) - (I), (B) - (II), (C) - (IV), (D) - (III)

4. (A) - (II), (B) - (III), (C) - (IV), (D) - (I)

49



The ________ determines the cause of the interrupt, performs the necessary processing and executes a return from the interrupt instruction to return the CPU to the execution state prior to the interrupt.

1. Interrupt sending line

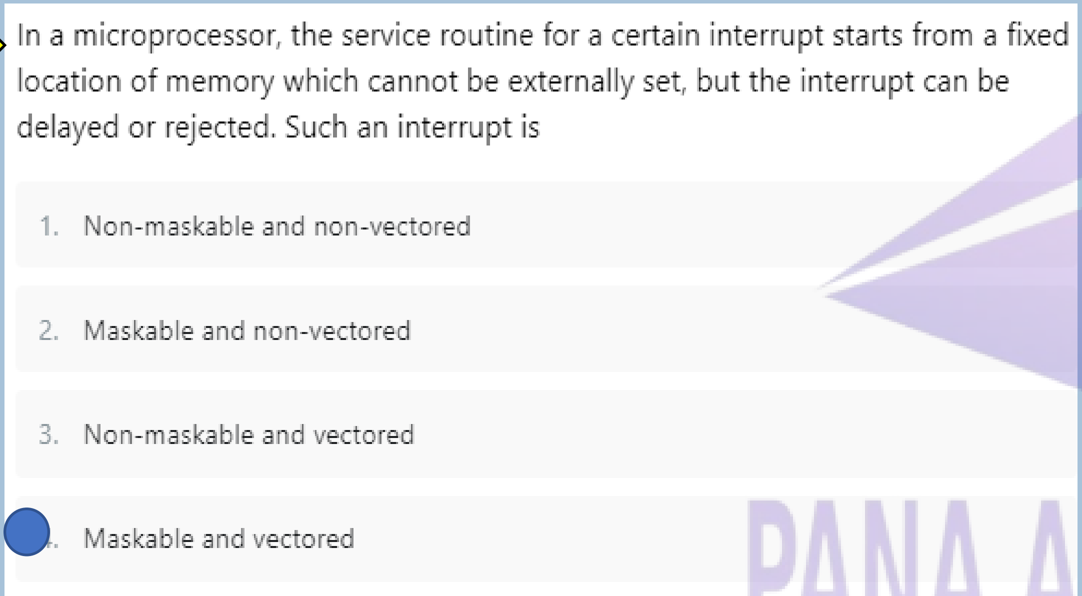
2. Interrupt bus

[✓ Interrupt handler]

4. Device driver

Interrupt operations

50

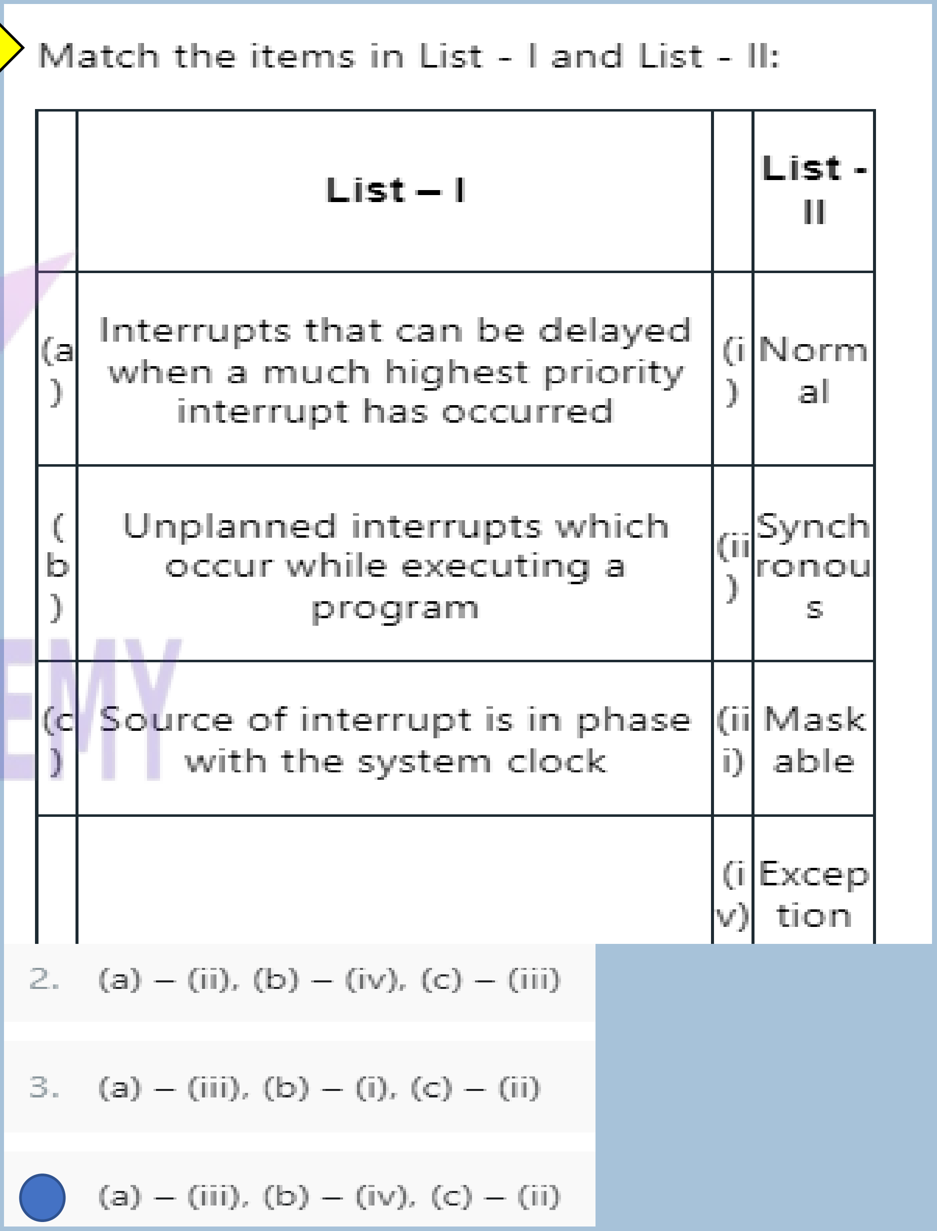


In a microprocessor, the service routine for a certain interrupt starts from a fixed location of memory which cannot be externally set, but the interrupt can be delayed or rejected. Such an interrupt is

1. Non-maskable and non-vectored
2. Maskable and non-vectored
3. Non-maskable and vectored

[✓ Maskable and vectored]

51



Match the items in List - I and List - II:

	List – I		List - II
(a)	Interrupts that can be delayed when a much highest priority interrupt has occurred	(i)	Normal
(b)	Unplanned interrupts which occur while executing a program	(ii)	Synchronous
(c)	Source of interrupt is in phase with the system clock	(iii)	Maskable
		(iv)	Exception

2. (a) – (ii), (b) – (iv), (c) – (iii)
3. (a) – (iii), (b) – (i), (c) – (ii)

[✓ (a) – (iii), (b) – (iv), (c) – (ii)]

Interrupt operations

52 What is the purpose of the 8259A Programmable Interrupt Controller (PIC) in conjunction with the 8086 microprocessor?

a) To generate clock pulses

b) To control memory access

[✓ c)] To manage multiple interrupt requests

d) To provide serial communication

53 The software interrupt INT 21H in the 8086 microprocessor is used for:

[✓ a)] DOS function calls

b) b) Arithmetic operations

c) c) Memory management

d) d) None of the above

Interrupt operations

54 The 8086 microprocessor uses how many bytes of memory for the interrupt vector table?

a) 256 bytes

●) 1024 bytes

c) 512 bytes

d) 2048 bytes

55 What does the 8086 microprocessor do when it encounters an interrupt?

a) Halts the current operation

●) Saves the state of the current execution

c) Ignores the interrupt

d) Resets the system

Interrupt operations

56 The trap flag (TF) in the 8086 microprocessor is used to:

[✓ a) Enable single-step execution]

b) Mask all interrupts

c) Disable single-step execution

d) Enable priority interrupts

57 The 8086 microprocessor's interrupt mechanism allows for how many distinct interrupt vectors?

a) 128

b) 192

[✓ c) 256]

d) 512

Interrupt operations

58 What is the purpose of the Initialization Command Word (ICW) in the 8259 PIC?

[✓ a) To set up the 8259 PIC during initialization]

b) To reset the interrupt flags

c) To read the status of interrupts

d) To mask all interrupts

59 Which pin on the 8259 PIC is used for cascading multiple PICs?

a) INTR

b) INTA

[✓ c) CAS]

d) IR7

Interrupt operations

60 What happens when the 8259 PIC receives an interrupt request (IR) signal?

a) It ignores the request

[✓ ●) It sends an interrupt acknowledge (INTA) signal to the CPU]

c) It resets the interrupt line

d) It performs a memory read operation

61 The 8259 PIC can operate in which of the following modes to handle priority?

a) Binary Priority Mode

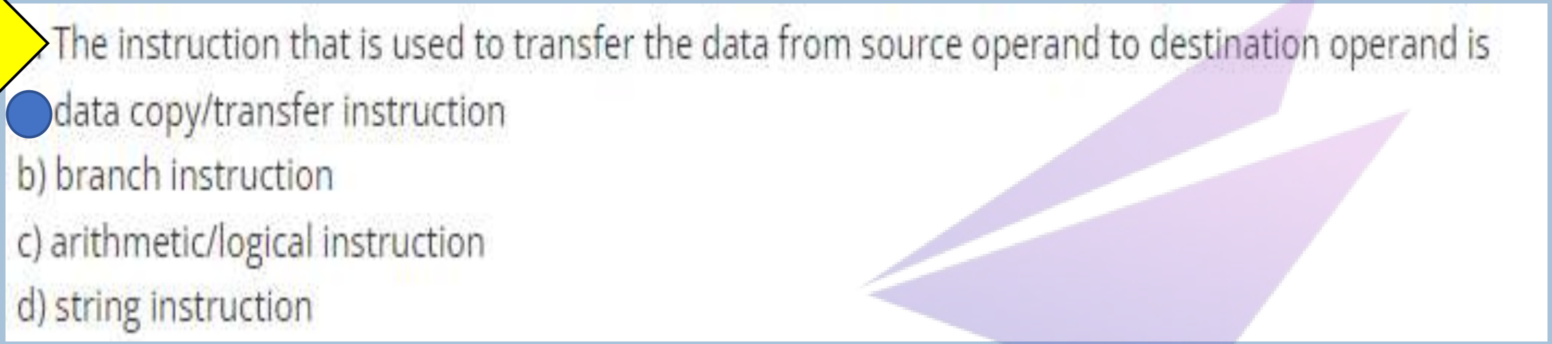
b) LIFO Priority Mode

[✓ ●) Rotating Priority Mode]

d) FIFO Priority Mode

Instruction 8085 and 8086

1



The instruction that is used to transfer the data from source operand to destination operand is

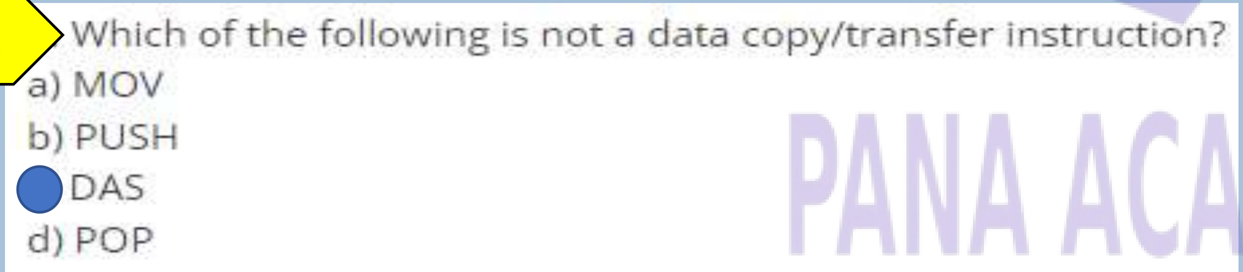
● data copy/transfer instruction

b) branch instruction

c) arithmetic/logical instruction

d) string instruction

2



Which of the following is not a data copy/transfer instruction?

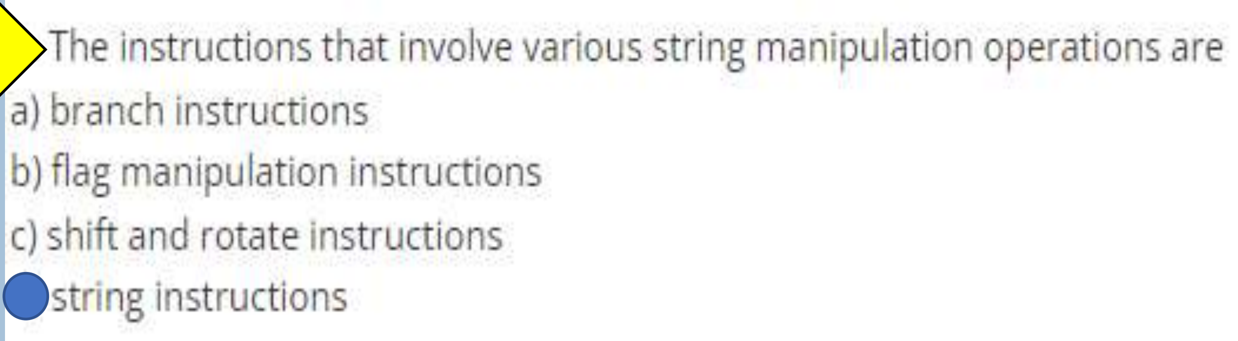
a) MOV

b) PUSH

● DAS

d) POP

3



The instructions that involve various string manipulation operations are

a) branch instructions

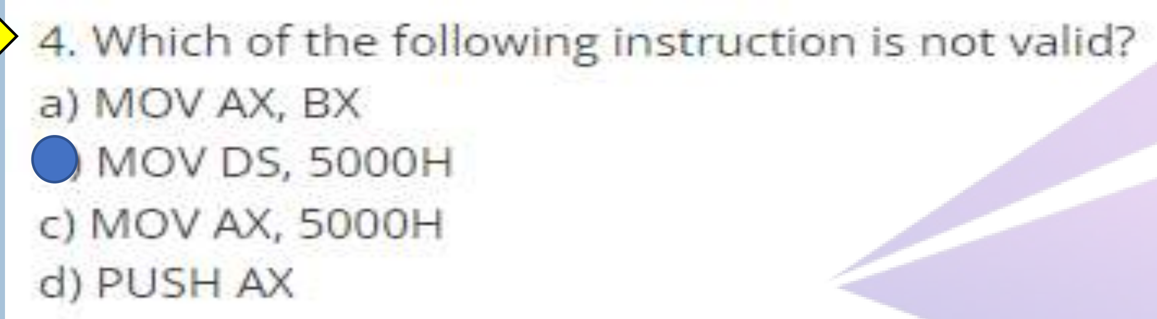
b) flag manipulation instructions

c) shift and rotate instructions

● string instructions

Instruction 8085 and 8086

4



4. Which of the following instruction is not valid?

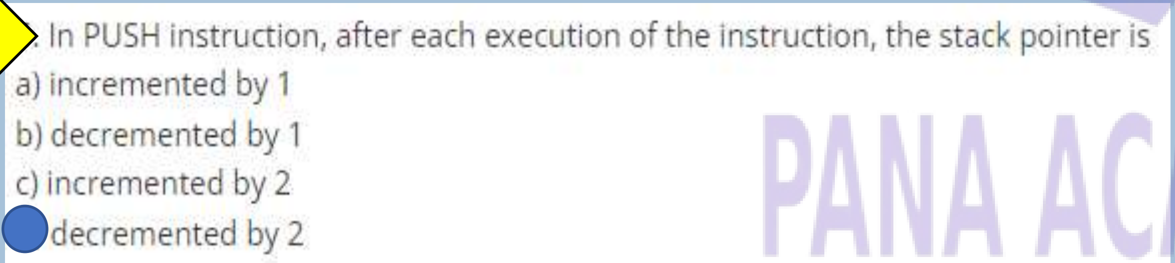
a) MOV AX, BX

[✓ MOV DS, 5000H]

c) MOV AX, 5000H

d) PUSH AX

5



In PUSH instruction, after each execution of the instruction, the stack pointer is

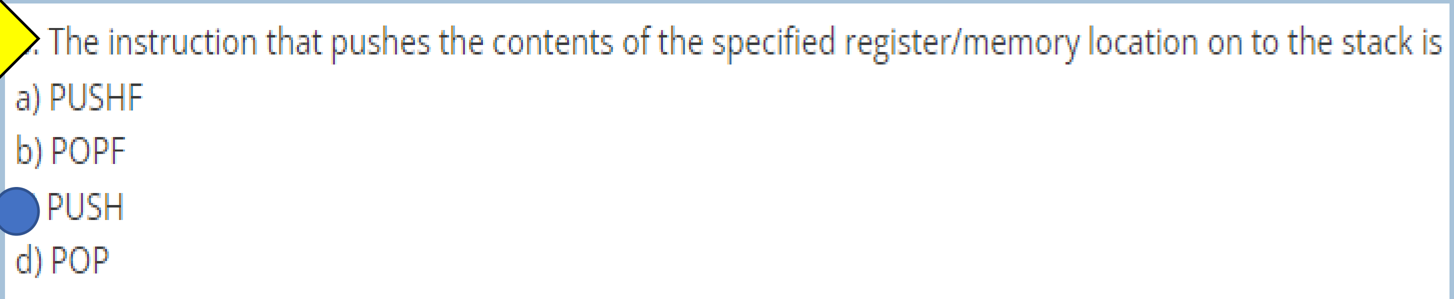
a) incremented by 1

b) decremented by 1

c) incremented by 2

[✓ decremented by 2]

6



The instruction that pushes the contents of the specified register/memory location on to the stack is

a) PUSHF

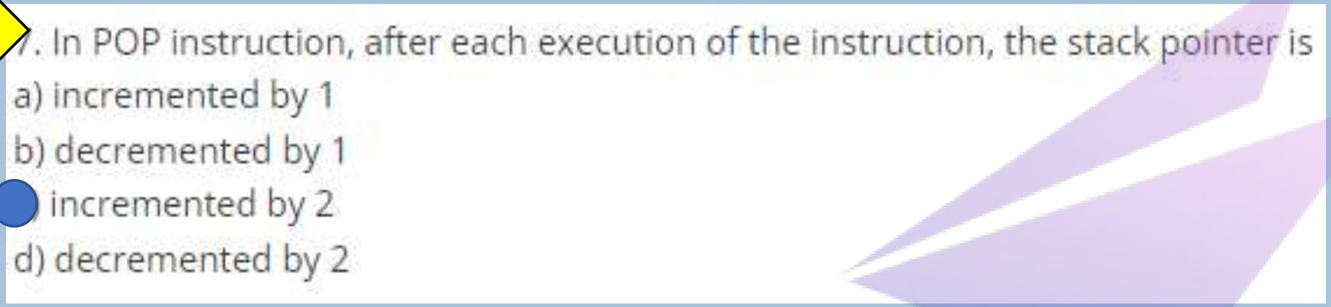
b) POPF

[✓ PUSH]

d) POP

Instruction 8085 and 8086

7



7. In POP instruction, after each execution of the instruction, the stack pointer is

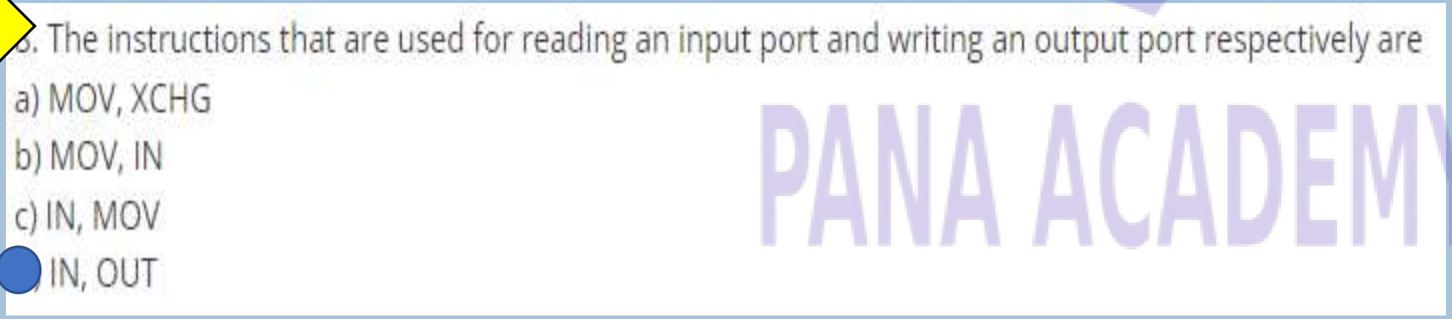
a) incremented by 1

b) decremented by 1

[✓ incremented by 2]

d) decremented by 2

8



8. The instructions that are used for reading an input port and writing an output port respectively are

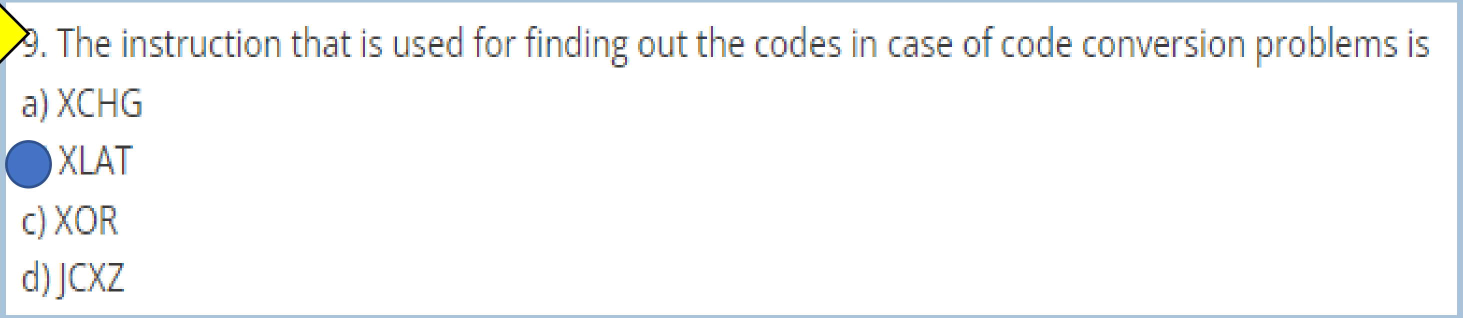
a) MOV, XCHG

b) MOV, IN

c) IN, MOV

[✓ IN, OUT]

9



9. The instruction that is used for finding out the codes in case of code conversion problems is

a) XCHG

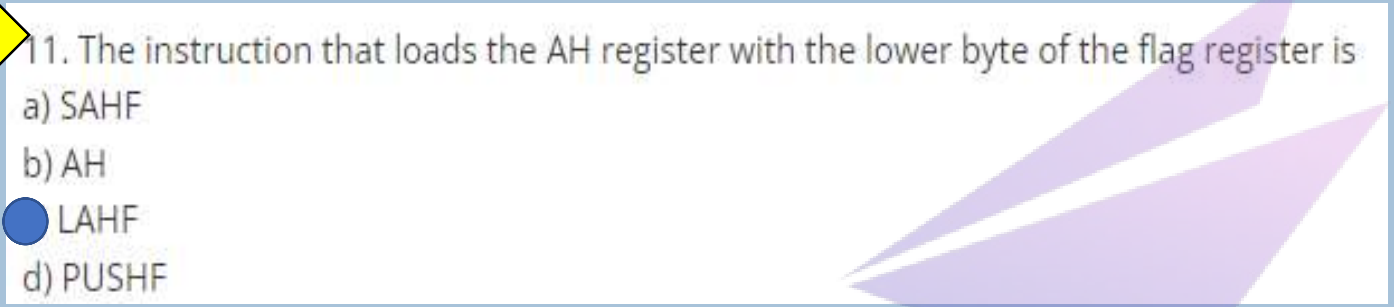
[✓ XLAT]

c) XOR

d) JCXZ

Instruction 8085 and 8086

10



11. The instruction that loads the AH register with the lower byte of the flag register is

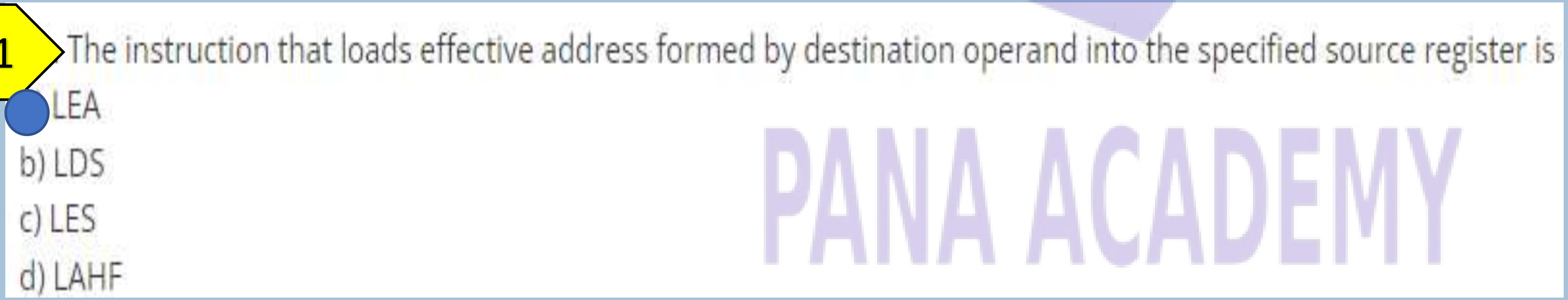
a) SAHF

b) AH

[✓ LAHF]

d) PUSHF

11



The instruction that loads effective address formed by destination operand into the specified source register is

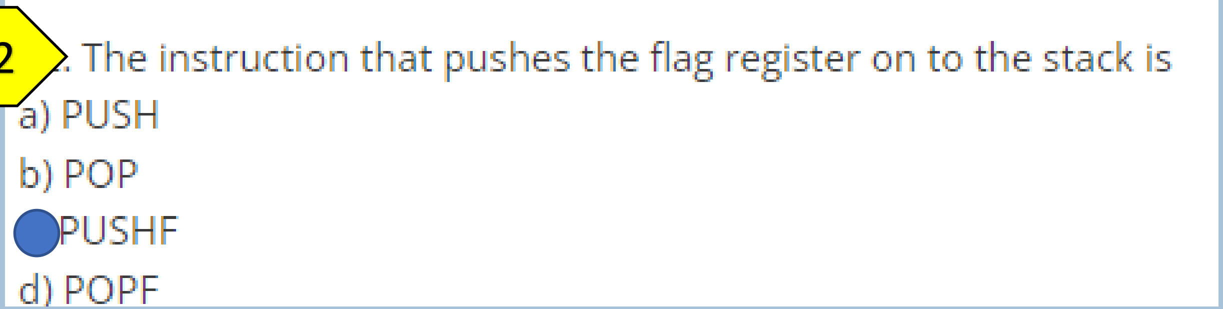
[✓ LEA]

b) LDS

c) LES

d) LAHF

12



. The instruction that pushes the flag register on to the stack is

a) PUSH

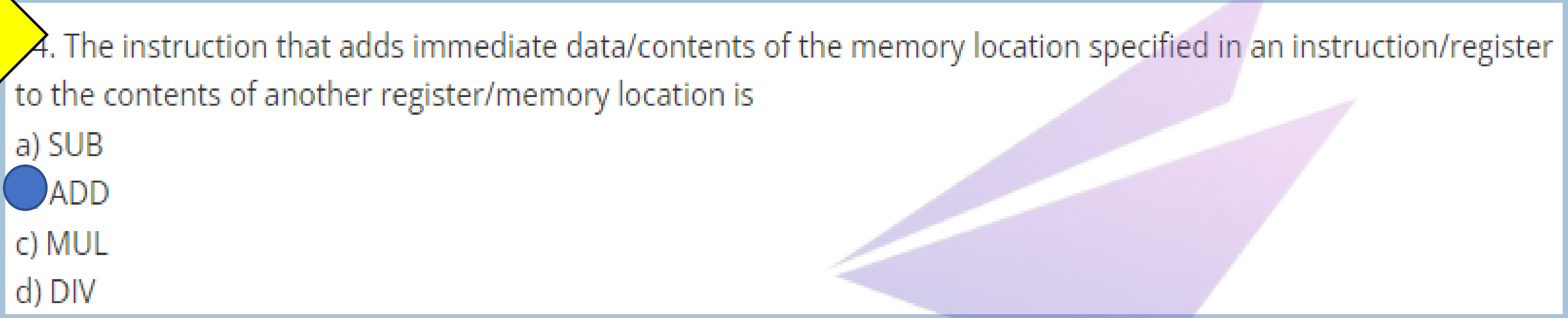
b) POP

[✓ PUSHF]

d) POPF

Instruction 8085 and 8086

13



4. The instruction that adds immediate data/contents of the memory location specified in an instruction/register to the contents of another register/memory location is

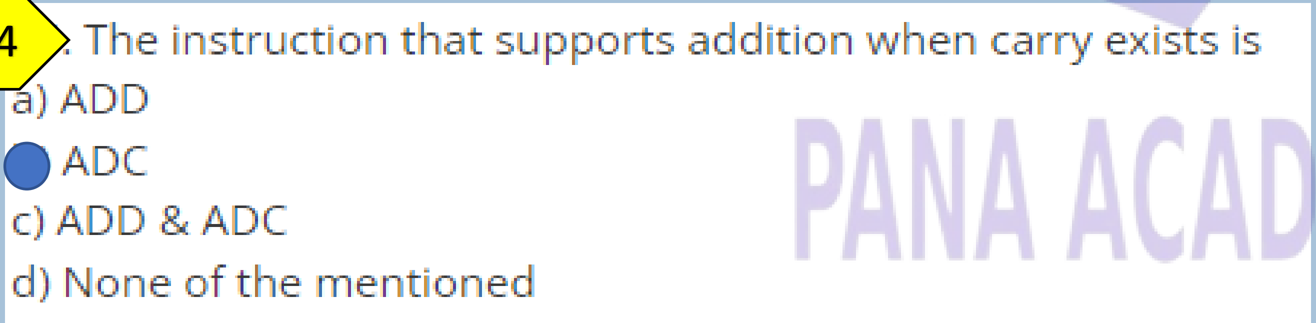
a) SUB

[✓ ADD]

c) MUL

d) DIV

14



. The instruction that supports addition when carry exists is

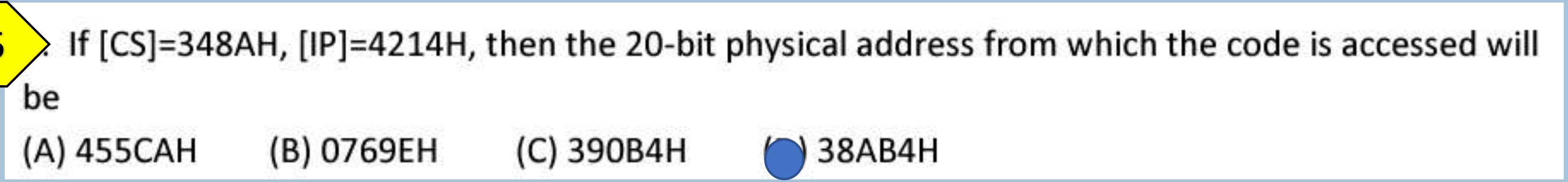
a) ADD

[✓ ADC]

c) ADD & ADC

d) None of the mentioned

15

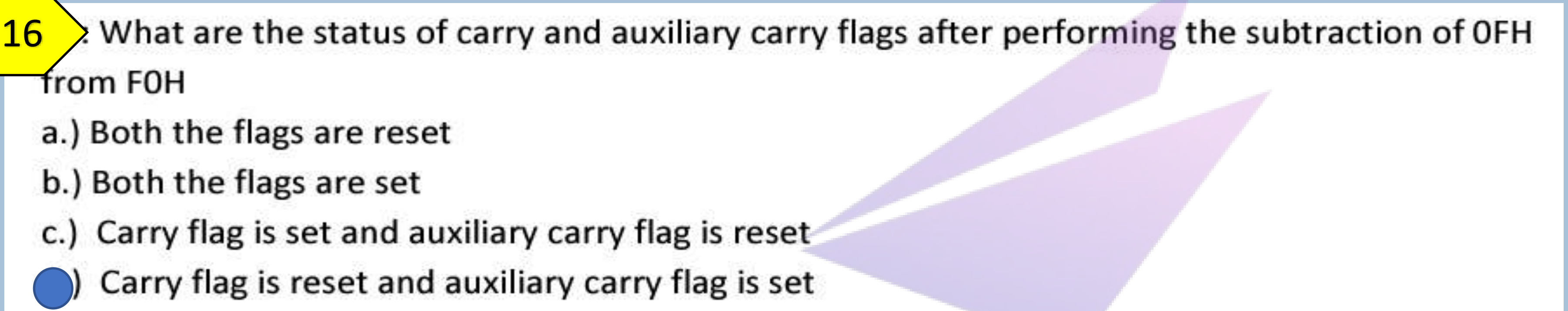


. If [CS]=348AH, [IP]=4214H, then the 20-bit physical address from which the code is accessed will be

(A) 455CAH (B) 0769EH (C) 390B4H [✓ 38AB4H]

Instruction 8085 and 8086

16



: What are the status of carry and auxiliary carry flags after performing the subtraction of 0FH from F0H

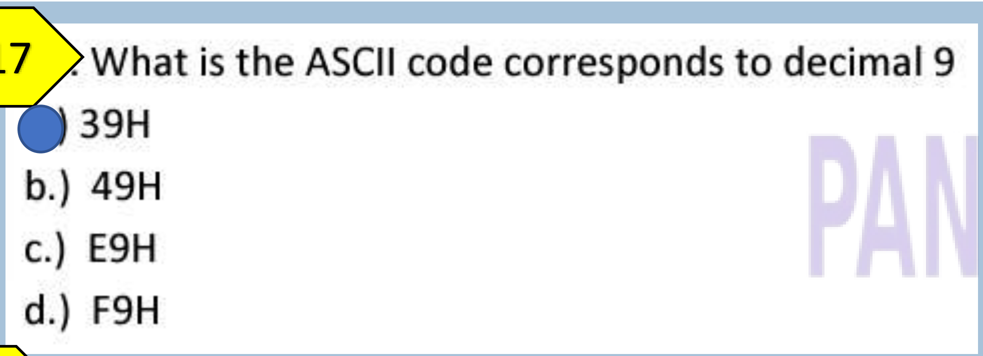
a.) Both the flags are reset

b.) Both the flags are set

c.) Carry flag is set and auxiliary carry flag is reset

[✓) Carry flag is reset and auxiliary carry flag is set]

17



: What is the ASCII code corresponds to decimal 9

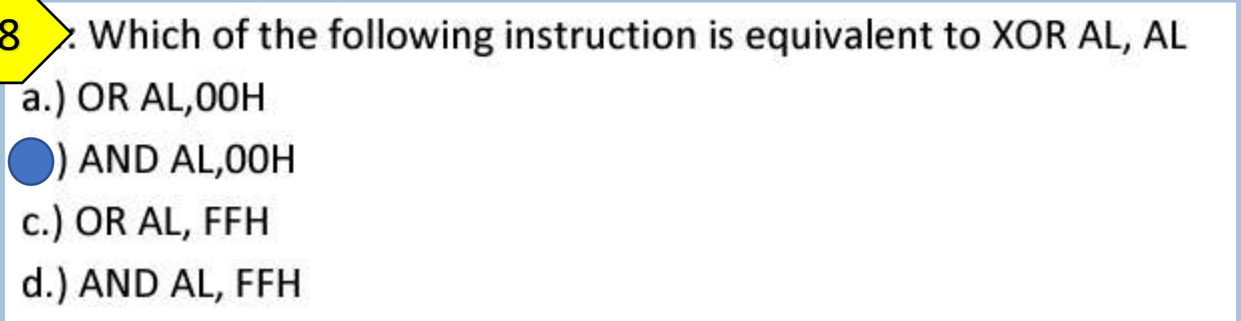
[✓) 39H]

b.) 49H

c.) E9H

d.) F9H

18



: Which of the following instruction is equivalent to XOR AL, AL

a.) OR AL,00H

[✓) AND AL,00H]

c.) OR AL, FFH

d.) AND AL, FFH

Interrupt operations

19 Which of the following instructions is used to complement the accumulator in 8085?

[✓ CMA]

b) CPI

c) INX

d) ANI

20 The instruction MVI A, 05H is an example of a:

a) Register to register transfer

[✓ Immediate addressing]

c) Direct addressing

d) Indirect addressing

Interrupt operations

21 Which of the following is a one-byte instruction?

a) MVI A, 32H

b) LDA 2500H

[✓ c) MOV B, C]

d) LXI H, 2500H

22 The instruction STA 2500H in the 8085 microprocessor is an example of:

a) Register addressing

b) Indirect addressing

[✓ c) Direct addressing]

d) Immediate addressing

Interrupt operations

23 The instruction ADD AX, 0005H is an example of:

a) Direct addressing

[✓) Immediate addressing]

c) Register addressing

d) Indirect addressing

24 Which of the following instructions is used to jump to a specified address if the zero flag (ZF) is set?

a) JNZ

b) JNC

[✓) JZ]

d) JMP

Interrupt operations

25 Which instruction is used to exchange the contents of AX and BX registers?

●) XCHG

b) MOV

c) SWAP

d) CMP

26 Which of the following instructions is used to jump to a specified address if the zero flag (ZF) is set?

a) JNZ

b) JNC

●) JZ

d) JMP

THANK YOU