Chapter-3.3

Input Output Interface

Introduction

- Interfacing the I/O devices with the microprocessor using latches and buffers.
- Using I/O devices data can be transferred between the microprocessor and the outside world.
- This can be done in group of 8 bits(1 byte) using the entire data bus. This is **parallel I/O.**
- Likewise one bit data transferred at a time using SID pins is serial I/O.
- Parallel transmission over long distances becomes expensive, thus serial transmission is preferred since one line required

Communication Channel

1. Simplex

- A simplex communication channel only sends information in one direction. For example, a radio station usually sends signals to the audience but never receives signals from them, thus a radio station is a simplex channel.

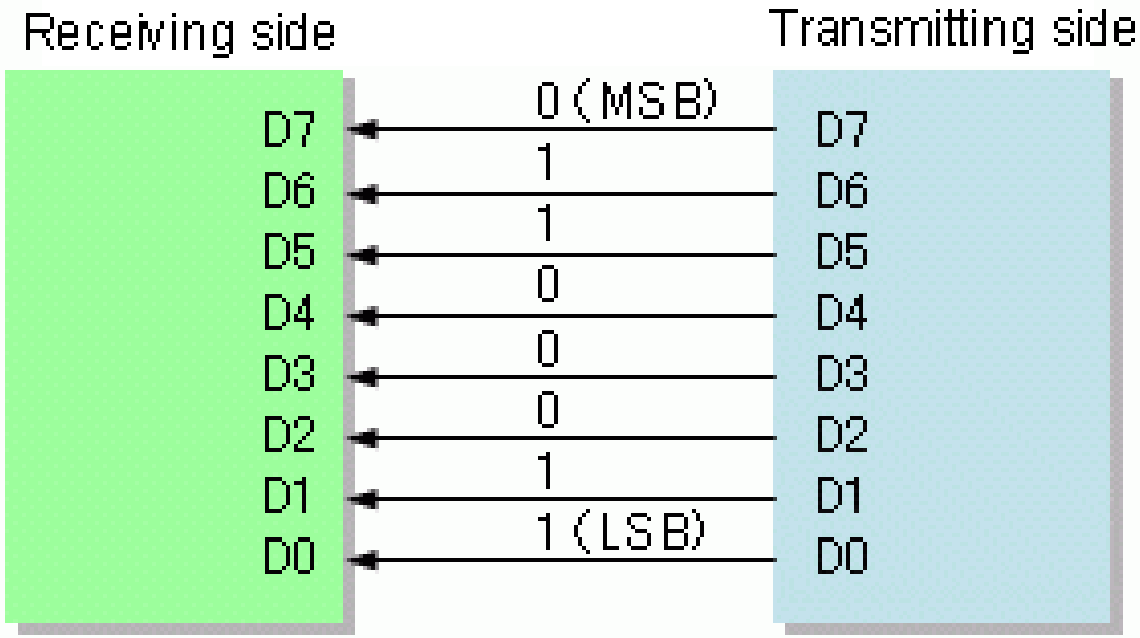
2. Half duplex

- In half duplex mode, data can be transmitted in both directions on a signal carrier except not at the same time. At a certain point, it is actually a simplex channel whose transmission direction can be switched. Walkie-talkie is a typical half duplex device.
- Then think about **Full Duplex???**

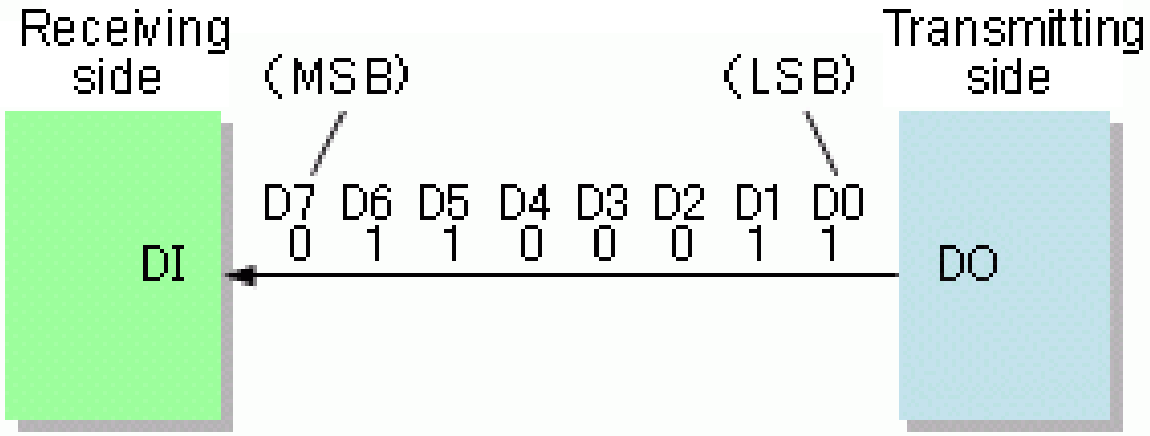
Parallel Communication

- In data transmission, **parallel communication** is a method of conveying multiple binary digits (bits) simultaneously. It contrasts with serial communication, which conveys only a single bit at a time; this distinction is one way of characterizing a communications link.
- A **parallel port** is a type of interface found on computers for connecting peripherals. The name refers to the way the data is sent; parallel ports send multiple bits of data at once
- Search for the applications of parallel communication??

Parallel interface example



Serial interface example (MSB first)

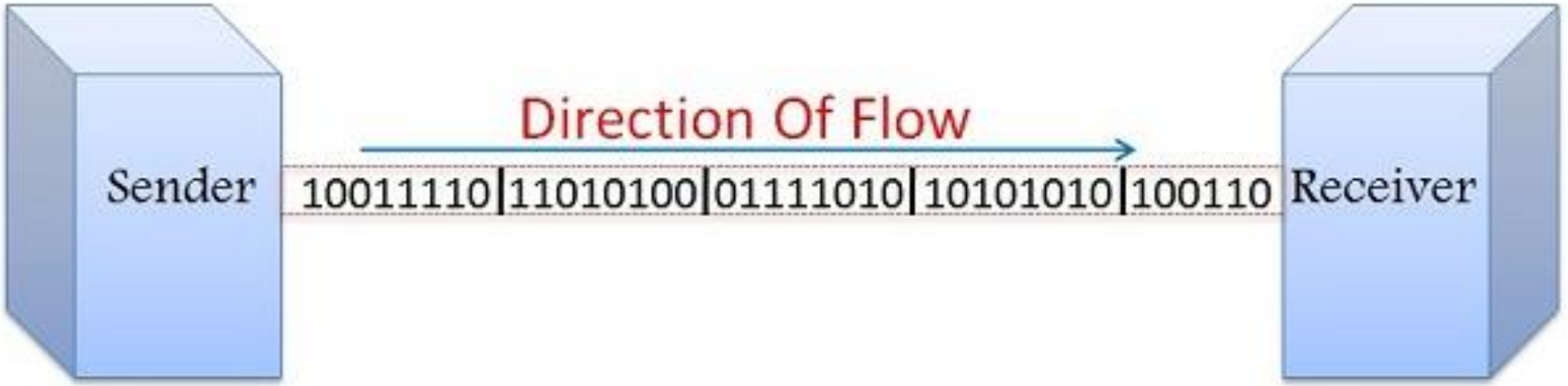


Serial Communication

- Data transferred one bit at a time.
- Single wire used and low cost but slow in speed.
- Byte of data must be converted to serial bits using a parallel-in-serial-out shift register, then it can be transmitted over a single data lines.
- What about applications??

Modes of Serial Transmission
i. Synchronous Transmission

- In Synchronous Transmission, data flows in a full duplex mode in the form of blocks or frames. Synchronization between the sender and receiver is necessary so that the sender know where the new byte starts (since there is no gap between the data).

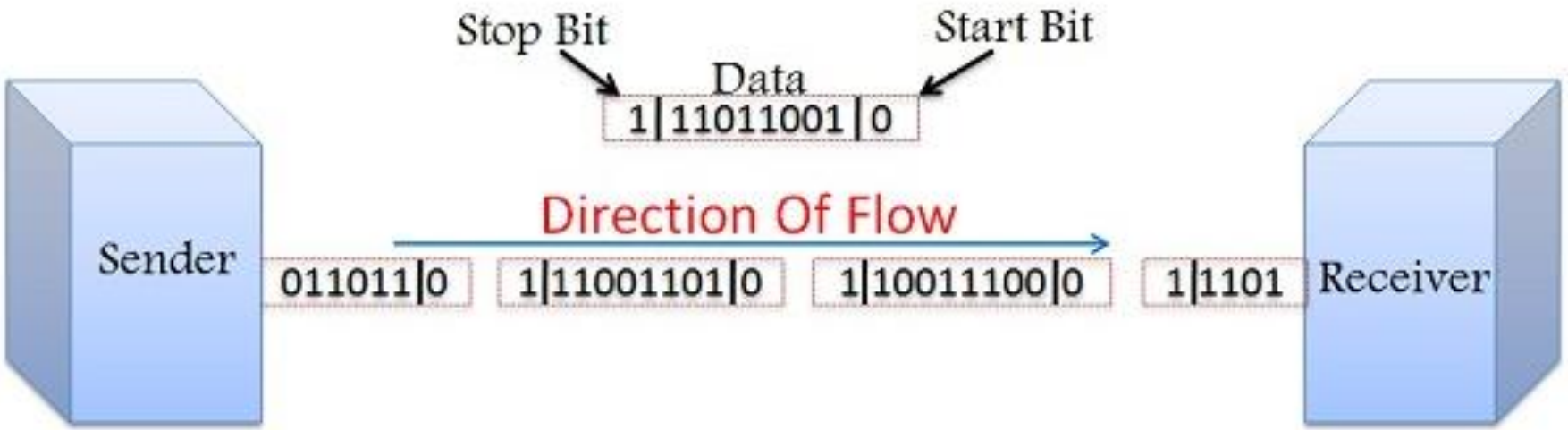


Cont..

- Synchronous Transmission is efficient, reliable and is used for transferring a large amount of data.
- It provides real-time communication between connected devices.

ii. Asynchronous Transmission

- In Asynchronous Transmission data flows in a half duplex mode, 1 byte or a character at a time.
- It transmits the data in a continuous stream of bytes.
- In general, the size of a character sent is 8 bits to which a parity bit is added i.e. a start and a stop bit that gives the total of 10 bits.
- It does not require a clock for synchronization; rather it uses the parity bits to tell the receiver how to interpret the data.

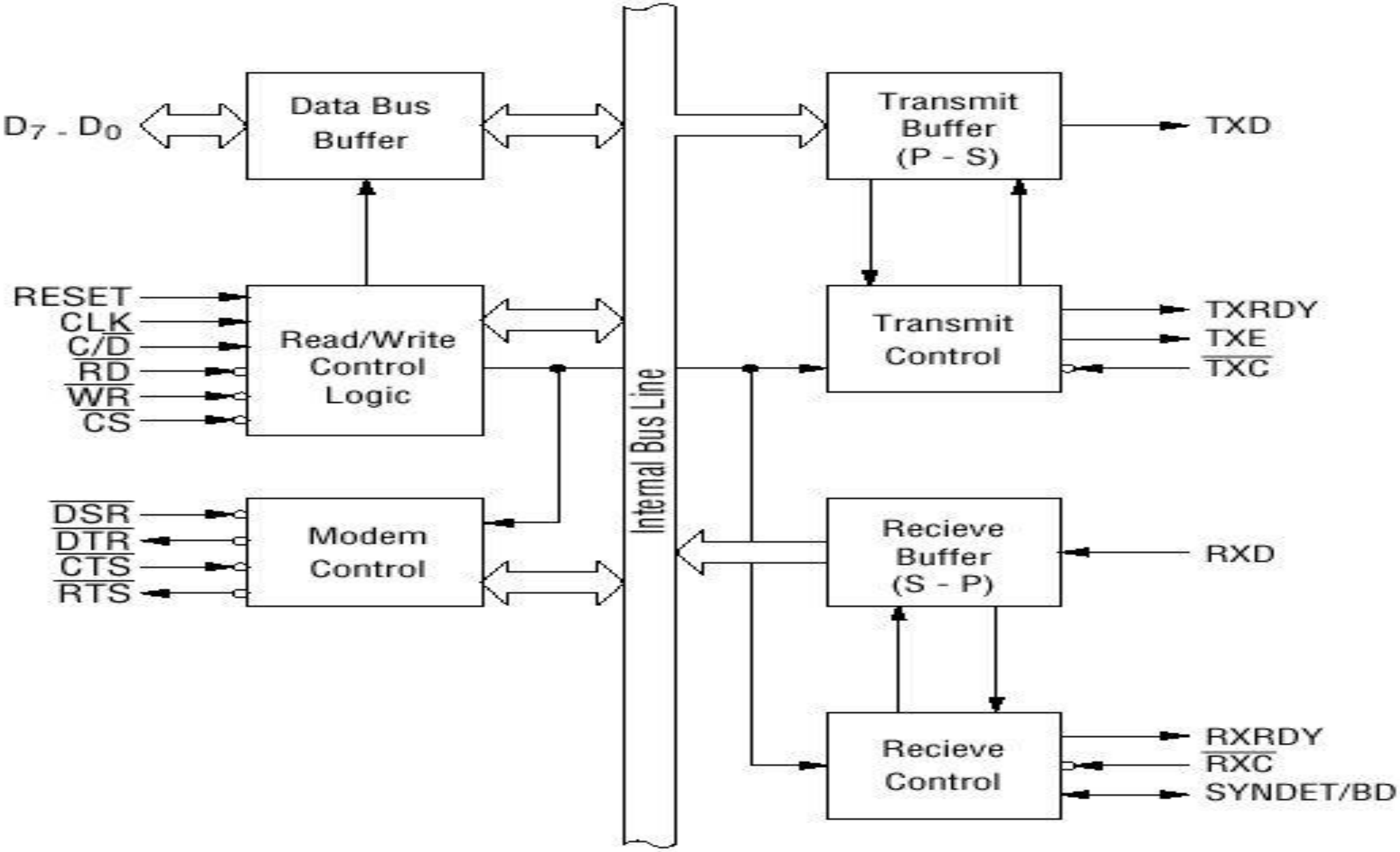


- It is simple, fast, economical and does not require a 2-way communication. Letters, emails, forums, televisions and radios are some of the examples of Asynchronous Transmission.

Introduction to 8251A PCI (Programmable Communication Interface)

- The 8251A is a **programmable serial communication interface** chip designed for synchronous and asynchronous serial data communication.
- It supports the serial transmission of data.
- It is packed in a 28 pin DIP.
- It is also called USART (Universal Synchronous Asynchronous Receiver Transmitter).

Block


D7 - D0
Data Bus Buffer
RESET
CLK
C/D
RD
WR
CS
Read/Write Control Logic
DSR
DTR
CTS
RTS
Modem Control
Internal Bus Line
Transmit Buffer (P - S)
TXD
Transmit Control
TXRDY
TXE
TXC
Recieve Buffer (S - P)
RXD
Recieve Control
RXRDY
RXC
SYNDET/BD

Explanation

Read/Write control logic:

- The Read/Write Control logic interfaces the 8251A with CPU, determines the functions of the 8251A according to the control word written into its control register.
- It monitors the data flow.
- This section has three registers

and they are **control register,**

status register and data buffer.

.

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	Operation
1	X	X	X	Invalid
0	0	0	1	data CPU< ----- 8251
0	0	1	0	data CPU ----- > 8251
0	1	0	1	Status word CPU < ----------8251
0	1	1	0	Control word CPU---------- > 8251

Cont..

Transmitter section:

- The transmitter section accepts parallel data from CPU and converts them into serial data.
- The transmitter section is double buffered, i.e., it has a **buffer register** to hold an 8-bit parallel data and another register called **output register** to convert the parallel data into serial bits

Receiver Section:

- The receiver section accepts serial data and convert them into parallel data.
- The receiver section is double buffered, i.e., it has an input register to receive serial data and convert to parallel, and a buffer register to hold the parallel data.

MODEM Control:

- The MODEM control unit allows to interface a MODEM to 8251A and to establish data communication through MODEM over telephone lines.
- This unit takes care of handshake signals for MODEM interface.
  - DSR - Data Set Ready : Checks if the Data Set is ready when communicating with a modem.
  - DTR - Data Terminal Ready : Indicates that the device is ready to accept data when the 8251 is communicating with a modem.
  - CTS - Clear to Send : If its low, the 8251A is enabled to transmit the serial data .
  - RTS - Request to Send Data : Low signal indicates that the modem is ready to receive a data byte.

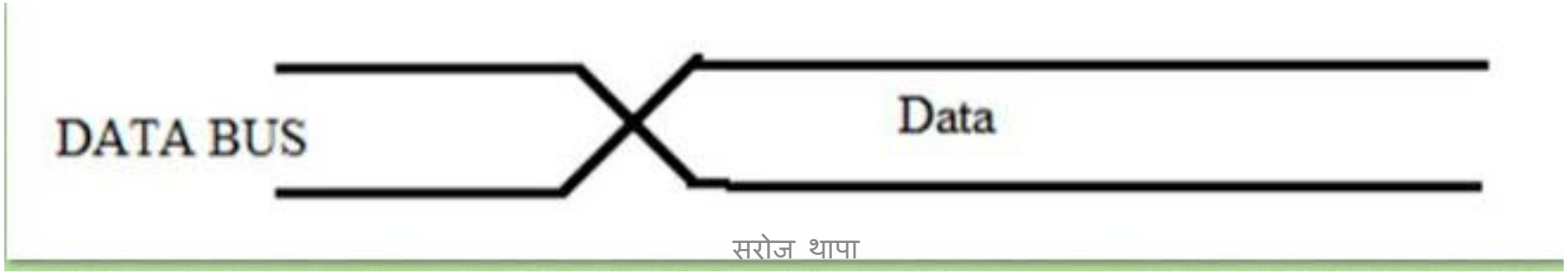
Something to research yourself!!!!

Control word in 8251 PCi

Methods of parallel data transfer

i) Simple I/O

- When you need to get digital data from simple switch, such as thermostat, into microprocessor, all you have to do is connect the switch to an I/O port line and read the port.
- Likewise, when you need to output data to simple display device, such as LED, all you have to do is connect the input of the LED buffer on an output port pin and output the logical level required to turn on the light.



ii) Strobe I/O

- In many applications, valid data is present on an external device only at a certain time, so it must be read in at that time.
- E.g. the ASCII-encoded keyboard. When a key is pressed, circuitry on the keyboard sends out the ASCII code for the pressed key on eight parallel data lines, and then sends out a strobe signal on another line to indicate that valid data is present on the eight data lines.

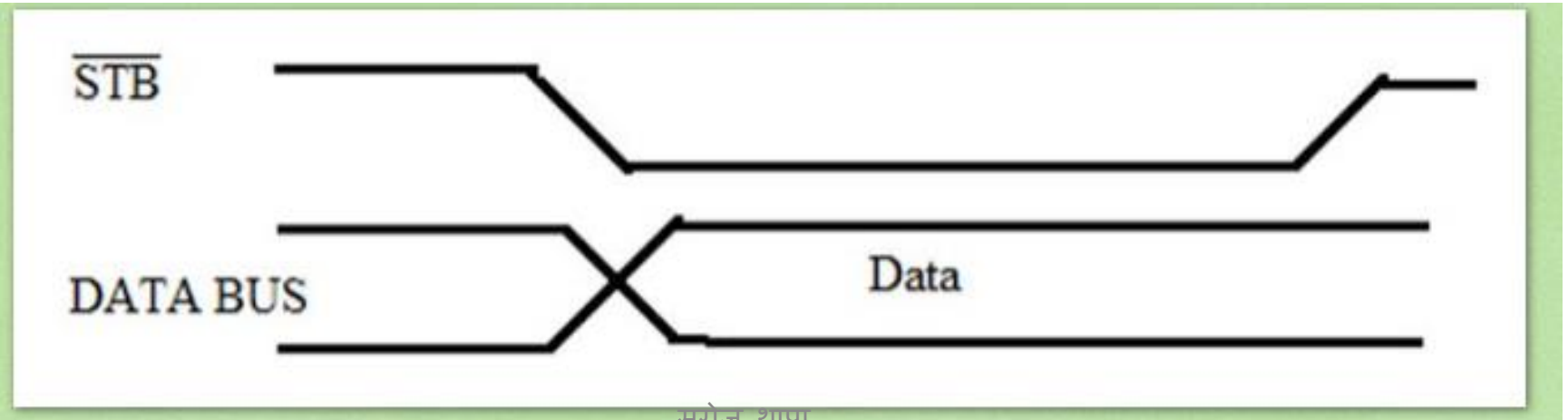


Fig: Strobe I/O

Cont..

- The sending device, such as a keyboard, outputs a parallel data on the data lines, and then outputs an STB signal to let you know that valid data is present.
- For low rates of data transfer, such as from a keyboard to a MP, a simple strobe transfer works well.
- However, for higher speed data transfer, this method does not work because there is no signal which tells the sending device when it is safe to send the next data byte.
- In other words, the sending system might send data bytes faster than the receiving system could read them.
- To prevent this problem, a handshake data transfer scheme is used

iii. Single handshake

- The peripheral outputs some parallel data and sends an STB signal to the MP.
- The MP detects the asserted STB signal on a polled or interrupts basis and reads in the bytes of data.
- Then, the MP sends ACK (acknowledge) signal to the peripheral to indicate that the data has been read and that the peripheral can send next byte of data.
- The point of this method is that the sending device or system is designed so that it does not send the next byte until the receiving device or system indicates with an ACK signal that it is ready to receive the next byte.

Cont..

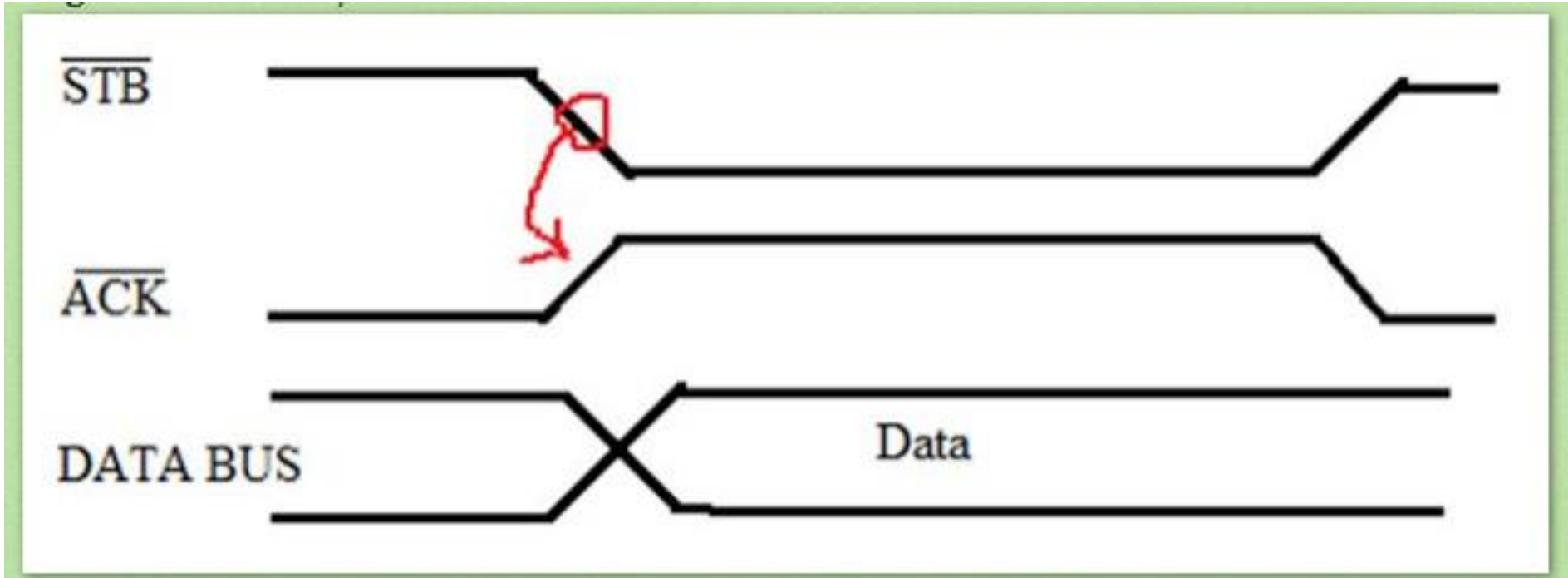


Fig: Single Handshake I/O

It shows the timing waveform for a handshake data transfer from a peripheral device to a MP.

iv. Double Handshake

- For data transfer where even more coordination is required between the sending system and the receiving system, a double handshake is used.
- The sending (peripheral) device asserts its STB line low to ask the receiving device whether it is ready or not for data reception.
- The receiving system raises its ACK line high to indicate that it is ready.
- The peripheral device then sends the byte of data and raises its STB line high to assure that the valid data is available for the receiving device (MP).
- When MP reads the data, it drops its ACK line low to indicate that it has received the data and requests the sending system to send next byte of data.

Cont…

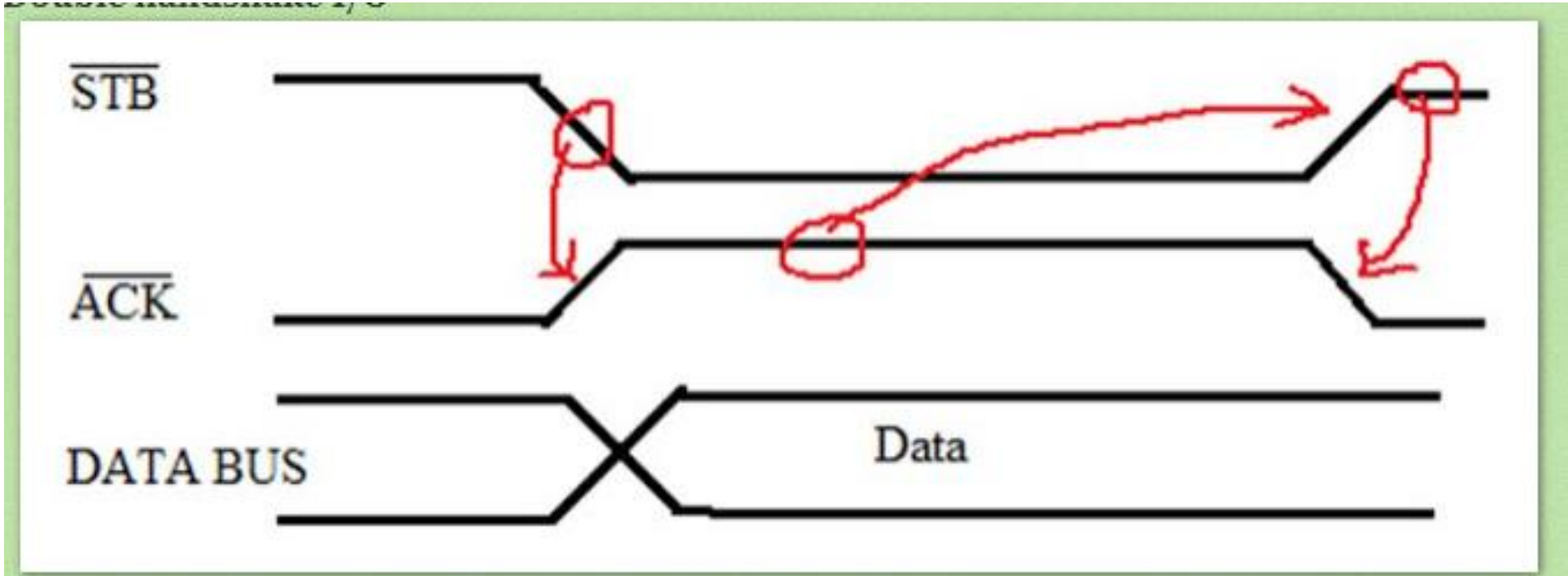
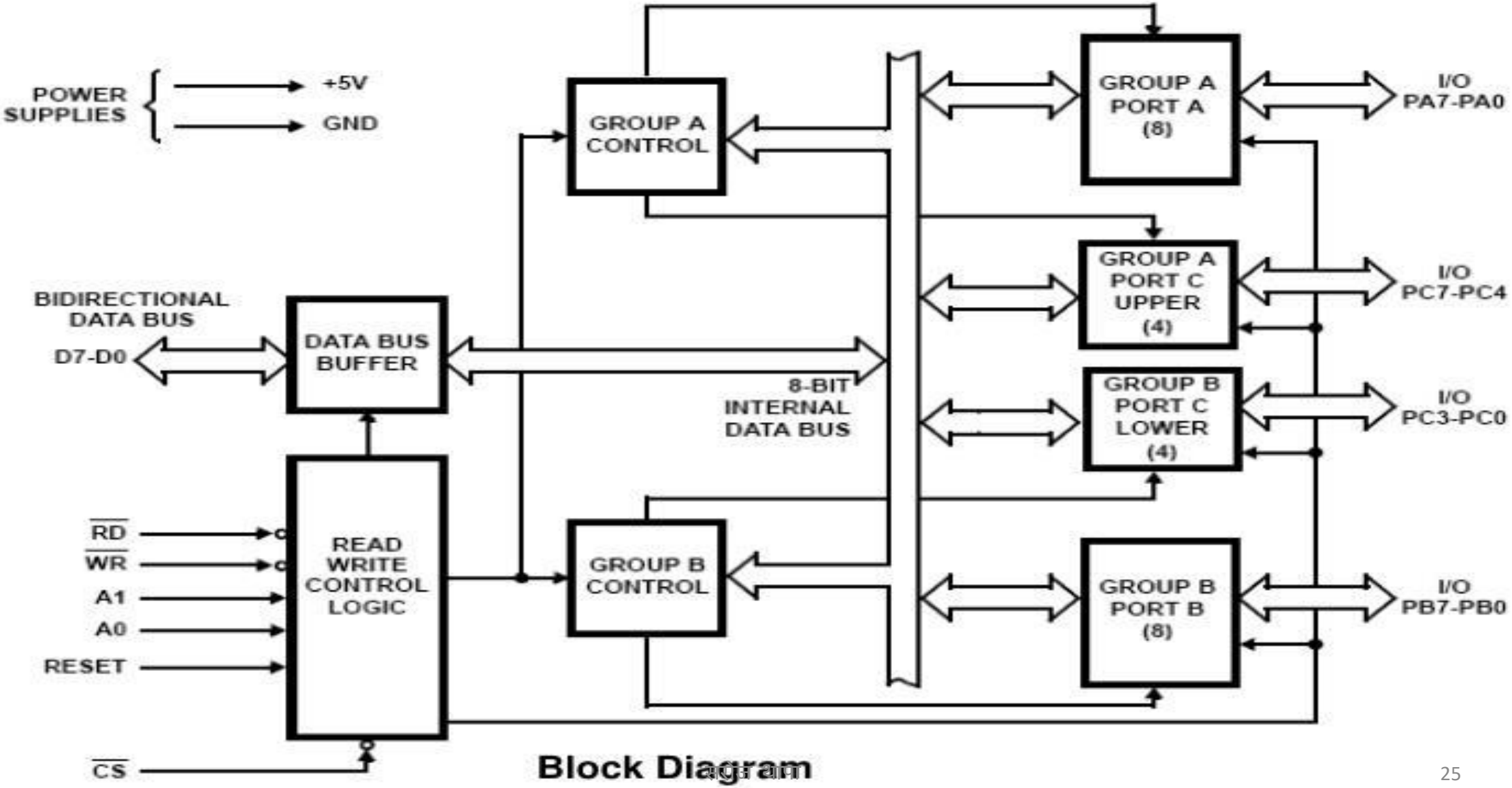


Fig: Double Handshake I/O

8255A Programmable Peripheral Interface

- The 8255A is an LSI peripheral designed to permit easy implementation of parallel I/O in the 8085 and 8086-microcomputer system.
- It consists of three 8-bit bidirectional I/O ports (24I/O lines) which can be configured as per the requirement.
- Timing of the data transfers to the 8255A is controlled by the ***read/write control*** *(RD and WR)* signals.

8255:Programmable Peripheral Interface



Block Diagram

Explanation

- **Data bus(D0-D7)**:These are 8-bit bi-directional buses, connected to 8085 data bus for transferring data.
- **CS**: This is Active Low signal. When it is low, then data is transfer from 8085.
- **Read**: This is Active Low signal, when it is Low read operation will be start.
- **Write**: This is Active Low signal, when it is Low Write operation will be start.

Cont..

Data Bus buffer:

- It is a 8-bit bidirectional Data bus.
- Used to interface between 8255 data bus with system bus.
- The internal data bus and Outer pins D0-D7 pins are connected in internally.
- The direction of data buffer is decided by Read/Control Logic.

Cont..

Read/Write Control Logic:

- This is getting the input signals from control bus and Address bus
- Control signal are RD and WR.
- Address signals are A0,A1,and CS.
- 8255 operation is enabled or disabled by CS.

Cont..

- **RESET**: This is used to reset the device. That means clear control registers.
- **PA0-PA7**:It is the 8-bit bi-directional I/O pins used to send the data to peripheral or to receive the data from peripheral.
- **PB0-PB7**:Similar to PA
- **PC0-PC7**:This is also 8-bit bidirectional I/O pins. These lines are divided into two groups.
  - PC0 to PC3(Lower Groups)
  - PC4 to PC7 (Higher groups)
- These two groups working in separately using 4 data's.

Port selection

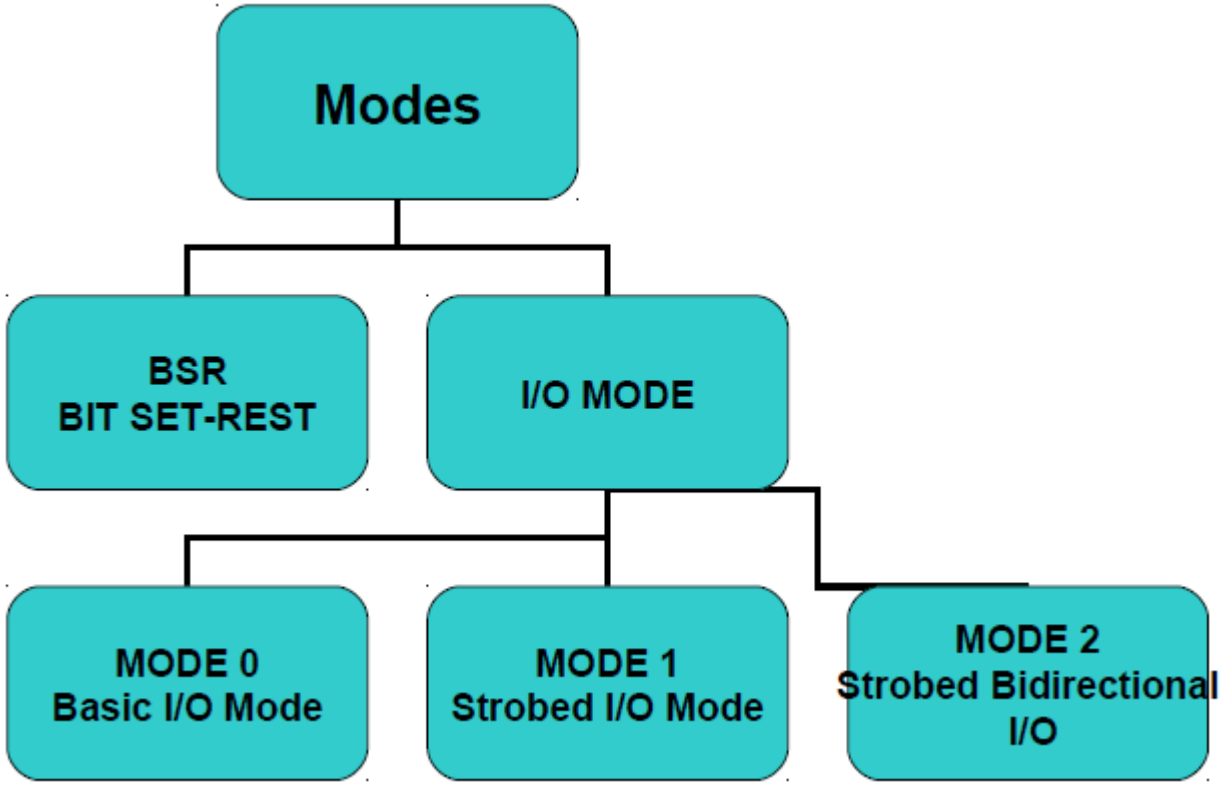
CS	A1	A0	Selects
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	x	x	8255 is not selected

Cont..

Group A and Group B control:

- Group A and B get the Control Signal from CPU and send the command to the individual control blocks.
- Group A send the control signal to port A and Port C (Upper) PC7-PC4.
- Group B send the control signal to port B and Port C (Lower) PC3-PC0.

8255A-PPI Modes of operation

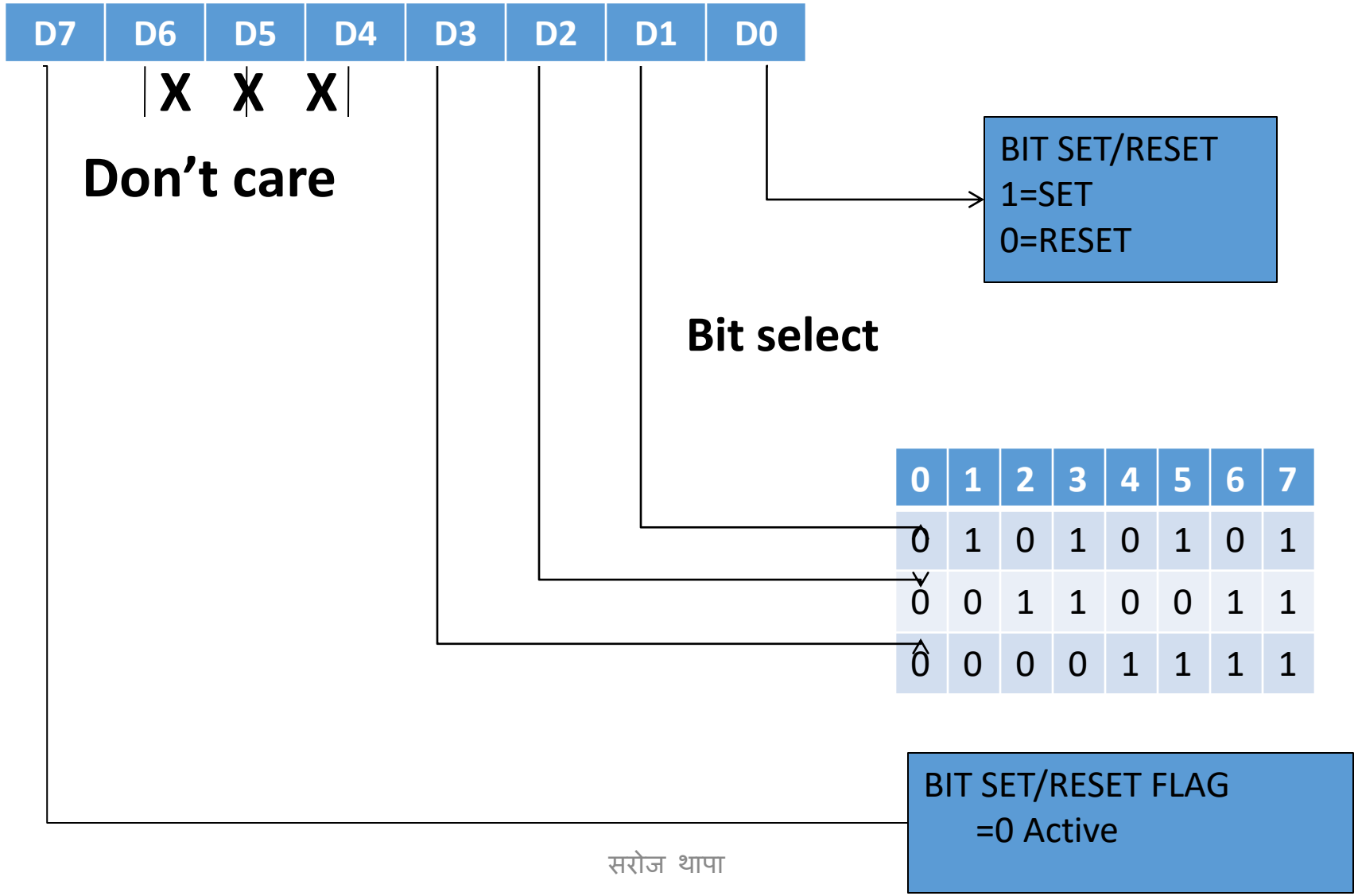


BIT Set Reset Mode (BSR Mode)

- The PORT C can be Set or Reset by sending OUT instruction to the CONTROL registers.
- The control word format for the BSR mode is shown.

FOR BIT SET/RESET MODE:

- This is bit set/reset control word format.



Input/output mode (I/O) –

- This mode is selected when the most significant bit (D7) in the control register is 1.

Mode 0 – Simple or basic I/O mode:

Port A, B and C can work either as input function or as output function.

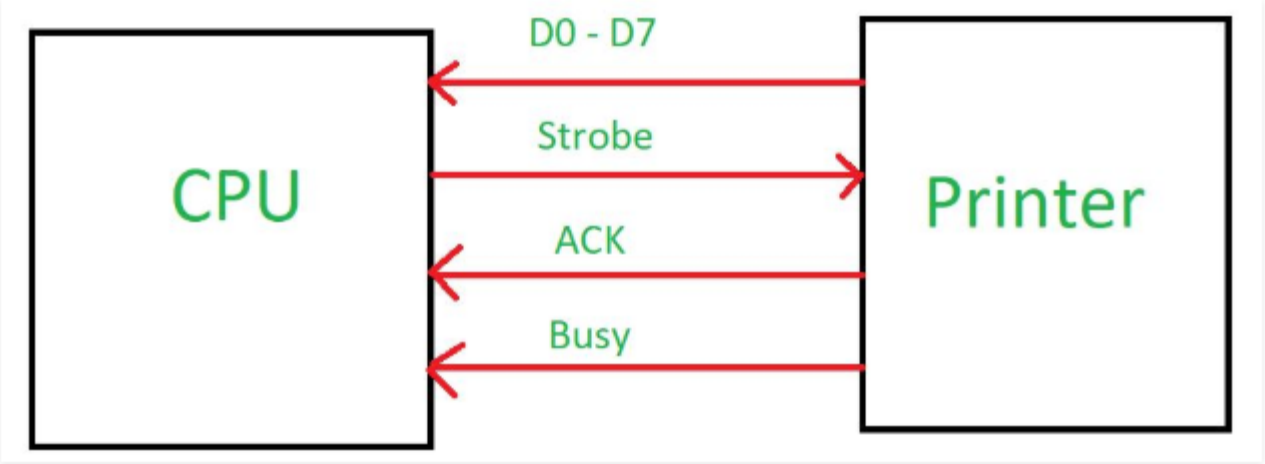
- Data can be simply read from and written to the input and output ports respectively, after appropriate initialisation.
- The outputs are latched but the inputs are not latched. It has interrupt handling capability.

Mode 1 – Handshake or strobbed I/O:

- In this mode the handshaking control the input and output action of the specified port.
- Port C lines PC0-PC2, provide strobe or handshake lines for port B.This group which includes port B and PC0-PC2 is called as group B for Strobed data input/output.
- Port C lines PC3-PC5 provide strobe lines for port A. This group including port A and PC3-PC5 from group A and the lines PC6, PC7 may be used as independent data lines.

Example:

- When CPU wants to send data to slow peripheral device like printer, it will send handshaking signal to printer to tell whether it is ready or not to transfer the data. When printer will be ready it will send one acknowledgement to CPU then there will be transfer of data through data bus.



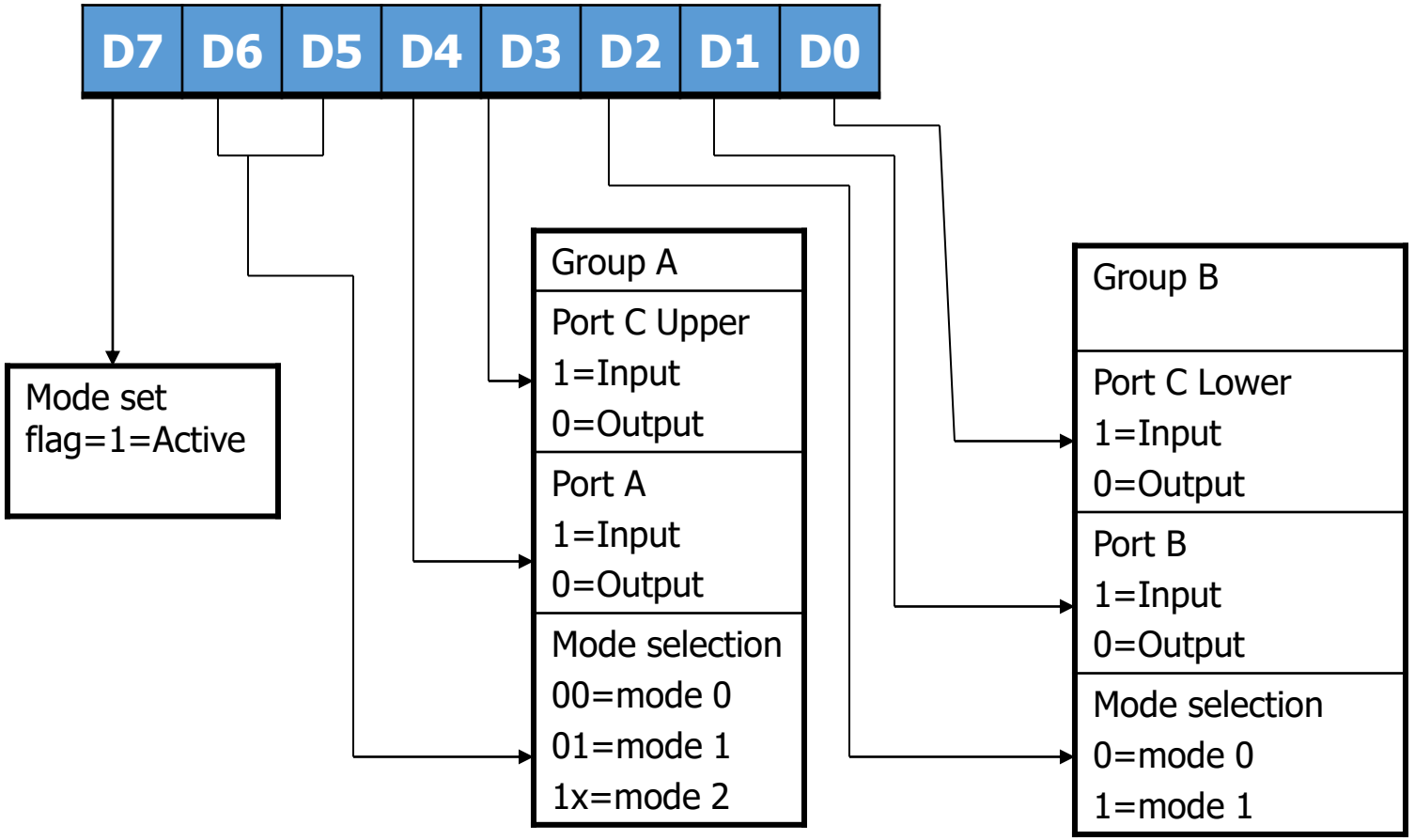
Mode 2 – Bidirectional I/O:

- This mode of operation of 8255 is also called as strobed bidirectional I/O.
- This mode of operation provides 8255 with an additional features for communicating with a peripheral device on an 8-bit data bus.
- Handshaking signals are provided to maintain proper data flow and synchronization between the data transmitter and receiver.
- The interrupt generation and other functions are similar to mode 1.
- In this mode, 8255 is a bidirectional 8-bit port with handshake signals.
- The RD and WR signals decide whether the 8255 is going to operate as an input port or output port.

Cont..

- The single 8-bit port in group A is available.
- The 8-bit port is bidirectional and additionally a 5-bit control port is available.
- Three I/O lines are available at port C.(PC2 – PC0)
- Inputs and outputs are both latched.
- The 5-bit control port C (PC3-PC7) is used for generating / accepting handshake signals for the 8-bit data transfer on port A.

Control Word format



Cont..

- The most significant bit (**D7**) is 1 for the I/O mode and 0 for the BSR mode.
- **D6 & D5** It is used to set the port A mode
- **D4** is used to tell whether port A is taking input or displaying the result. If it is 1 then it is taking input otherwise displaying output.
- **D3** is used to tell whether port C higher bites is taking input or displaying the result. If it is 1 then it is taking input otherwise displaying output.
- **D2** tells the mode of port B. If it is 0 then port B is in m0 mode otherwise in m1 mode.
- **D1** is used to tell whether port B is taking input or displaying the result. If it is 1 then it is taking input otherwise displaying output.
- **D0** is used to tell whether port C lower bits is taking input or displaying the result. If it is 1 then it is taking input otherwise displaying output.

DMA -Introduction

- DMA is acronym to Direct Memory Access.
- Direct Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU).
- It is also a fast way of transferring data within (and sometimes between) computer.
- The DMA I/O technique provides direct access to the memory while the microprocessor is temporarily disabled.
- The DMA controller temporarily borrows the address bus, data bus and control bus from the microprocessor and transfers the data directly from the external devices to a series of memory locations (and vice versa).

Cont…

- I/O devices are connected to system bus via a special interference circuit known as DMA CONTROLLER.
- Both CPU and DMA have access to main memory via a shared system bus having data address and control line
- The controller manages data transfer between memory and peripheral directly by borrowing the address bus, control bus and data bus from microprocessor and bypassing it.

DMA operation

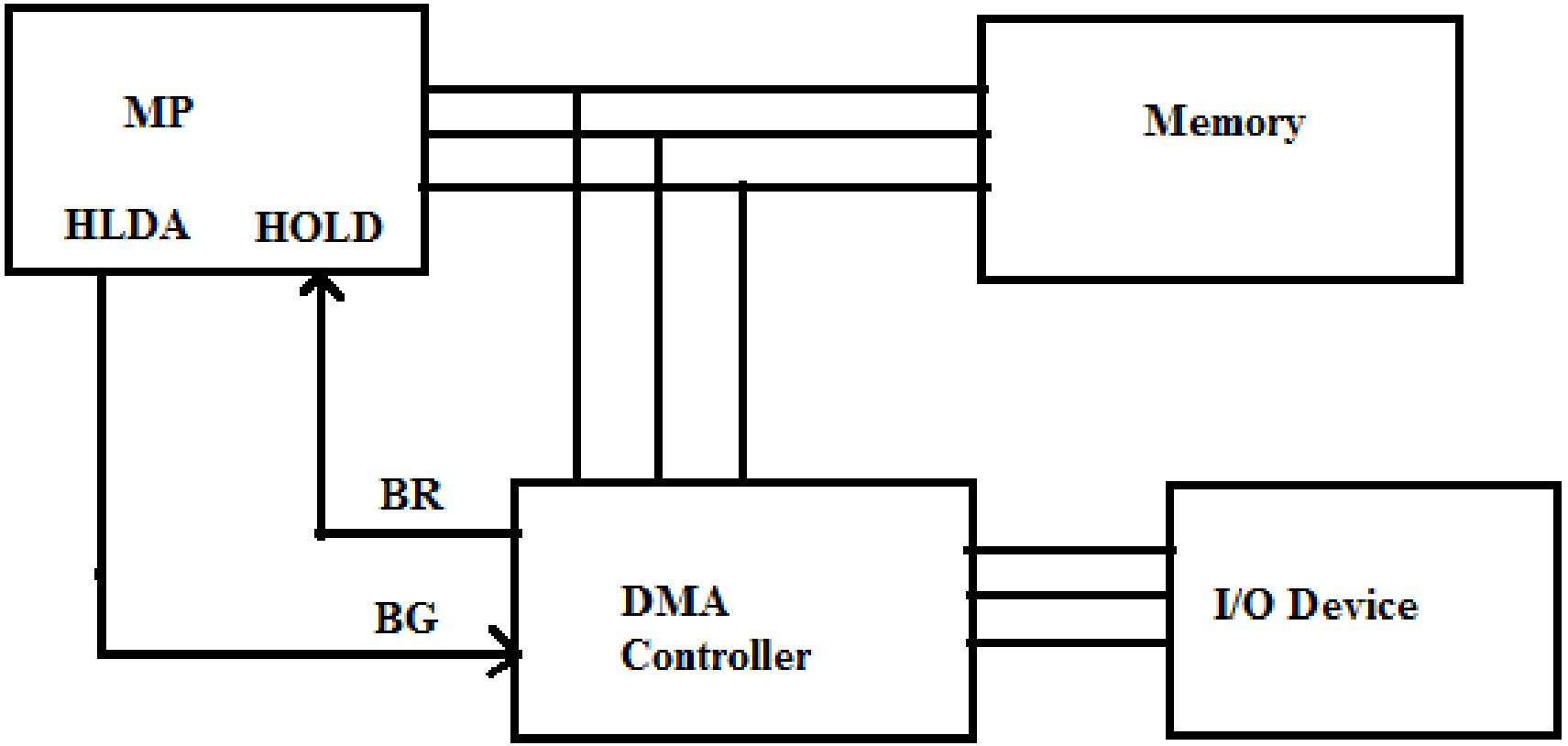
- Two control signals are used to request and acknowledge a direct memory access (DMA) transfer in the microprocessor-based system.
- The **HOLD signal** as an input(to the processor) is used to request a DMA action.
- The **HLDA signal** as an output that acknowledges the DMA action.
- When the processor recognizes the hold, it stops its

execution and enters hold cycles.

Cont..

- HOLD input has higher priority than INTR or NMI.
- The only microprocessor pin that has a higher priority than a HOLD is the RESET pin.
- HLDA becomes active to indicate that the processor has placed its buses at high impedance state.
- A **DMA read** transfers data from the memory to the I/O device.
- A **DMA write** transfers data from an I/O device to memory.
- The system contains separate memory and I/O control signals.

DMA controller



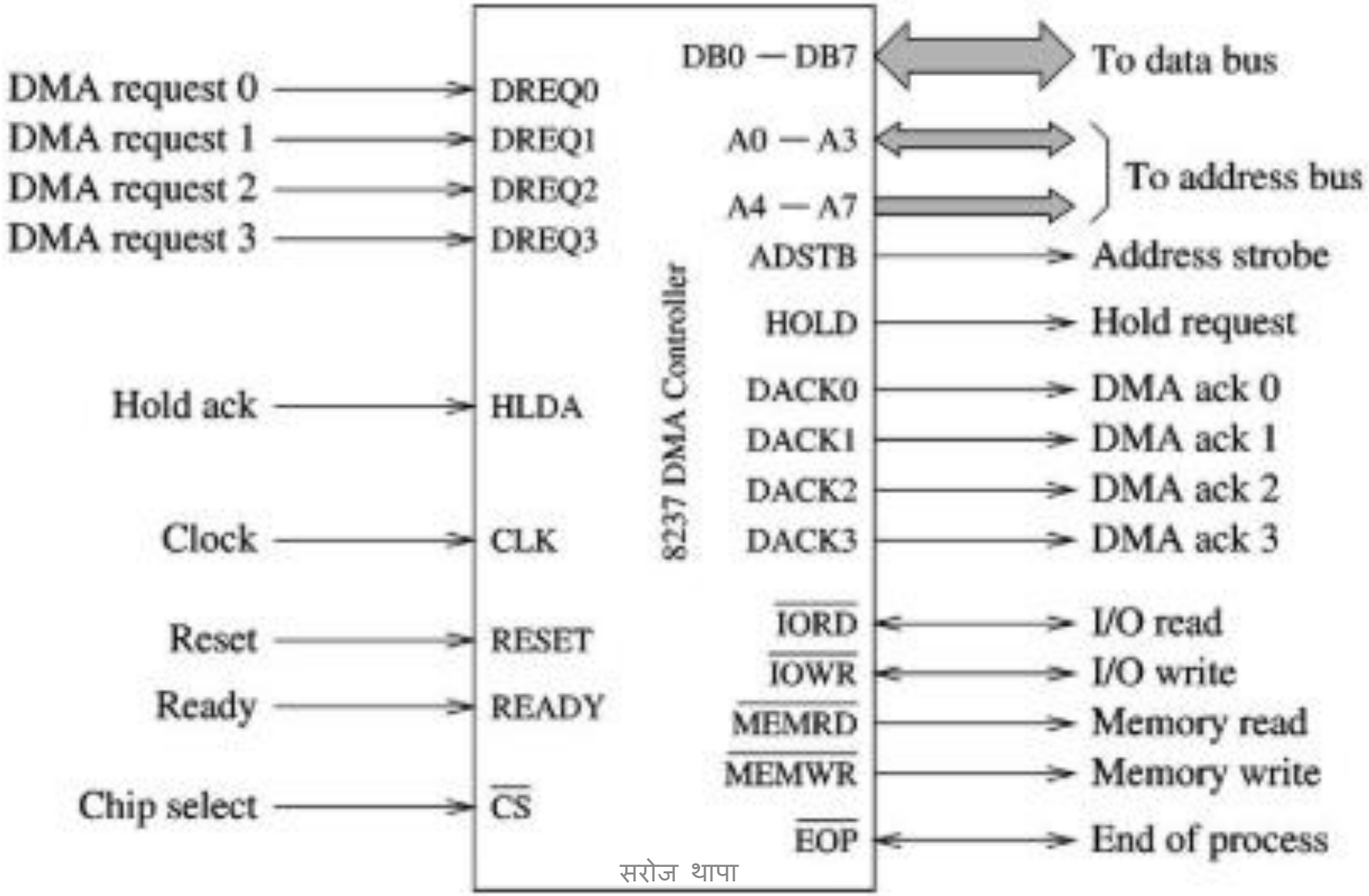
Questions

- What are the advantages, disadvantages and applications of DMA??
- Draw the timing diagram of DMA operation.

8237 DMA controller

- 8237 is a programmable Direct Memory Access controller (DMA) housed in a 40-pin package
- It has four independent channels with each channel capable of transferring 64K bytes
- It must interface with MPU and a peripheral device
- It is an I/O device to MPU
- It is a data transfer processor to peripheral device
- Many of its signals that are input in the I/O mode become outputs in the processor mode

8237 DMA Controller



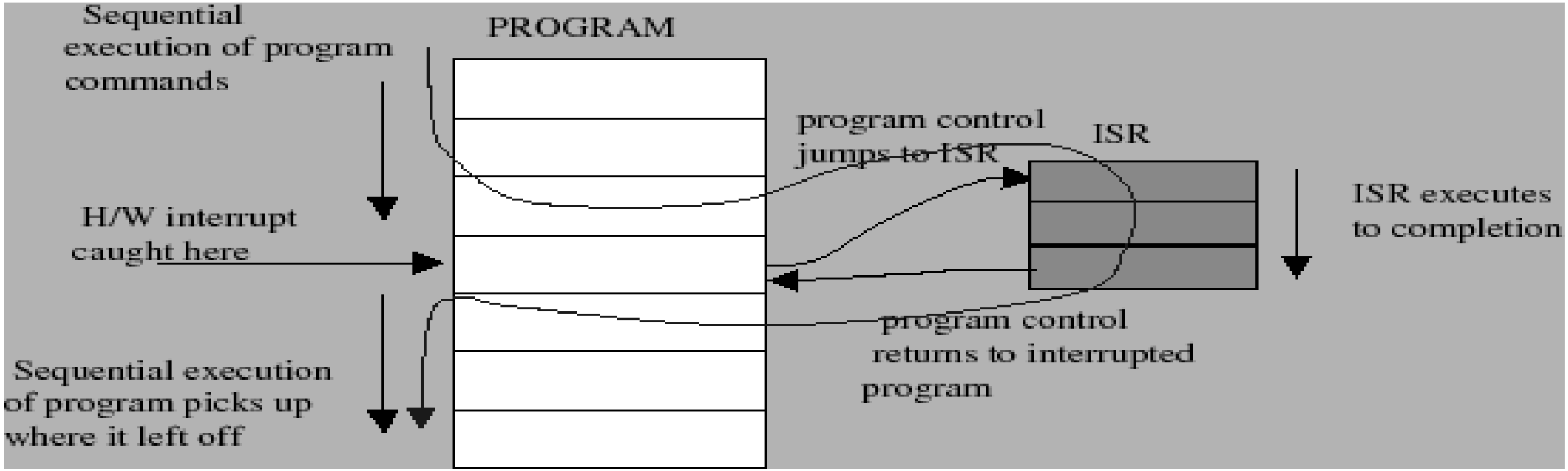
Polling vs Interrupt

- 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.
- 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.
- Search for the advantages of Interrupt over polling???
- What is Interrupt service routine?

Interrupt

- 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 RET instruction

Control flow in the presence of a hardware interrupt



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

1. Polled interrupts and
2. Daisy chain (vectored) interrupts.

1. Polled interrupts

- In this method, all interrupts are serviced by branching to the same service program.
- This program then checks with each device if it is the one generating the interrupt.
- The order of checking is determined by the priority that has to be set. The device having the highest priority is checked first and then devices are checked in descending order of priority.
- If the device is checked to be generating the interrupt, another service program is called which works specifically for that particular device
- Here several devices are connected to a single interrupt line (INTR) of the microprocessor.
- The major disadvantage of this method is that it is quite slow. To overcome this, we can use hardware solution, one of which involves connecting the devices in series. This is called Daisy-chaining method.

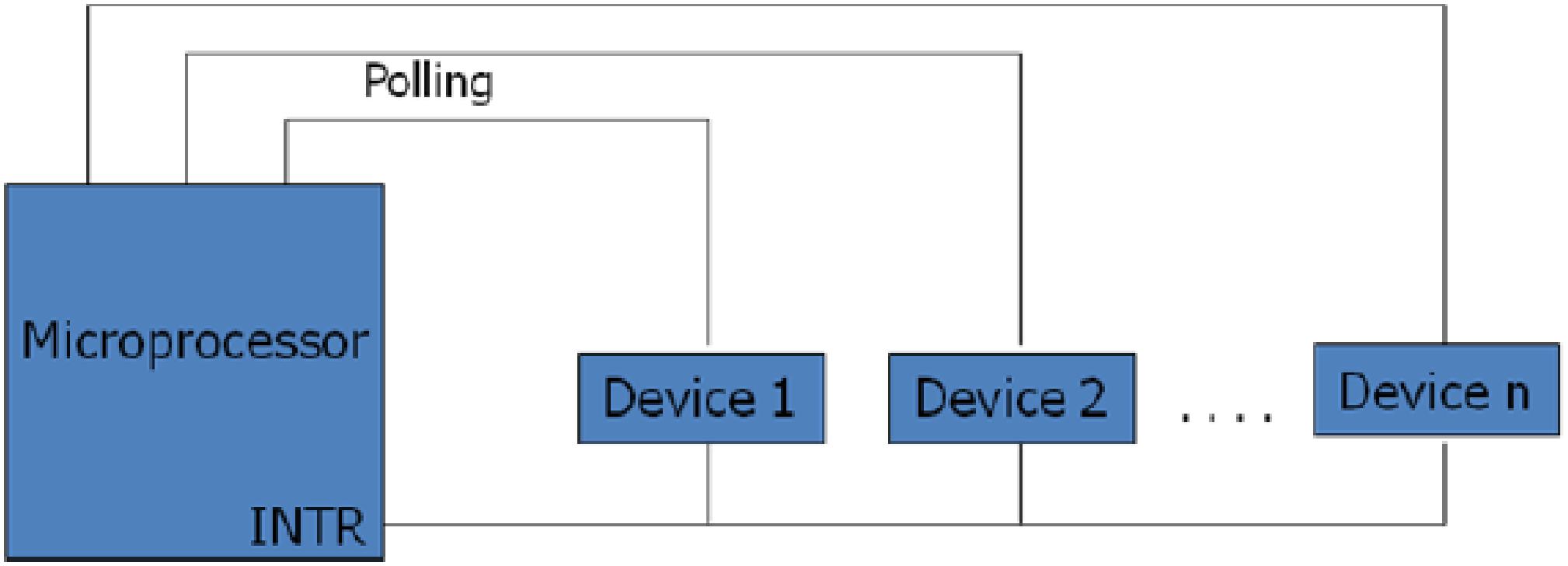


Fig: Polled Interrupt

2. 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 to the INTA next priority device until it is accepted by the interrupting device.
- 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

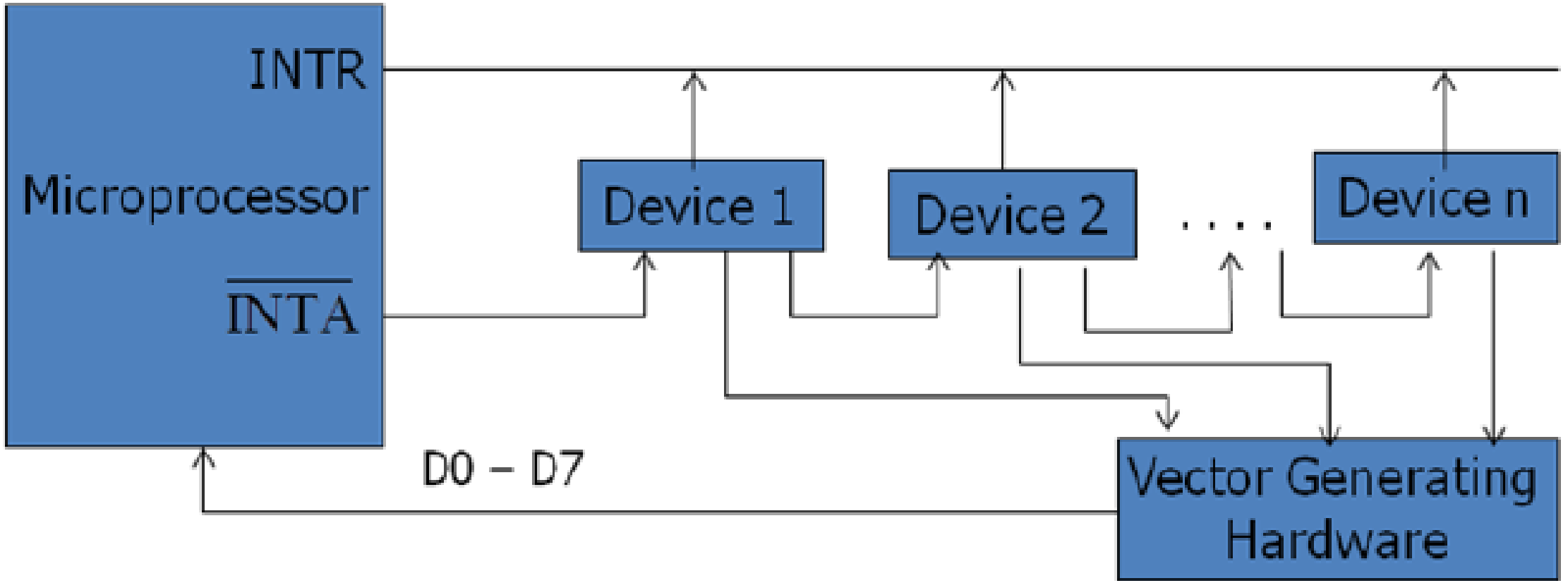


Fig: Vectored (Daisy Chain) Interrupt

Types of interrupt in 8085

- It supports two types of interrupts.

1. Hardware

2. 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	0000_H
RST 1	0008_H
RST 2	0010_H
RST 3	0018_H
RST 4	0020_H
RST 5	0028_H
RST 6	0030_H
RST 7	0038_H

Hardware Interrupts in 8085

- 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

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.
  1. By resetting microprocessor (External signal)
  2. By giving a high TRAP ACKNOWLEDGE (Internal signal)

RST 7.5:

- The RST 7.5 interrupt is a maskable interrupt.
- It has the second highest priority.
- It is edge sensitive. ie. 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.

RST 6.5 and 5.5:

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

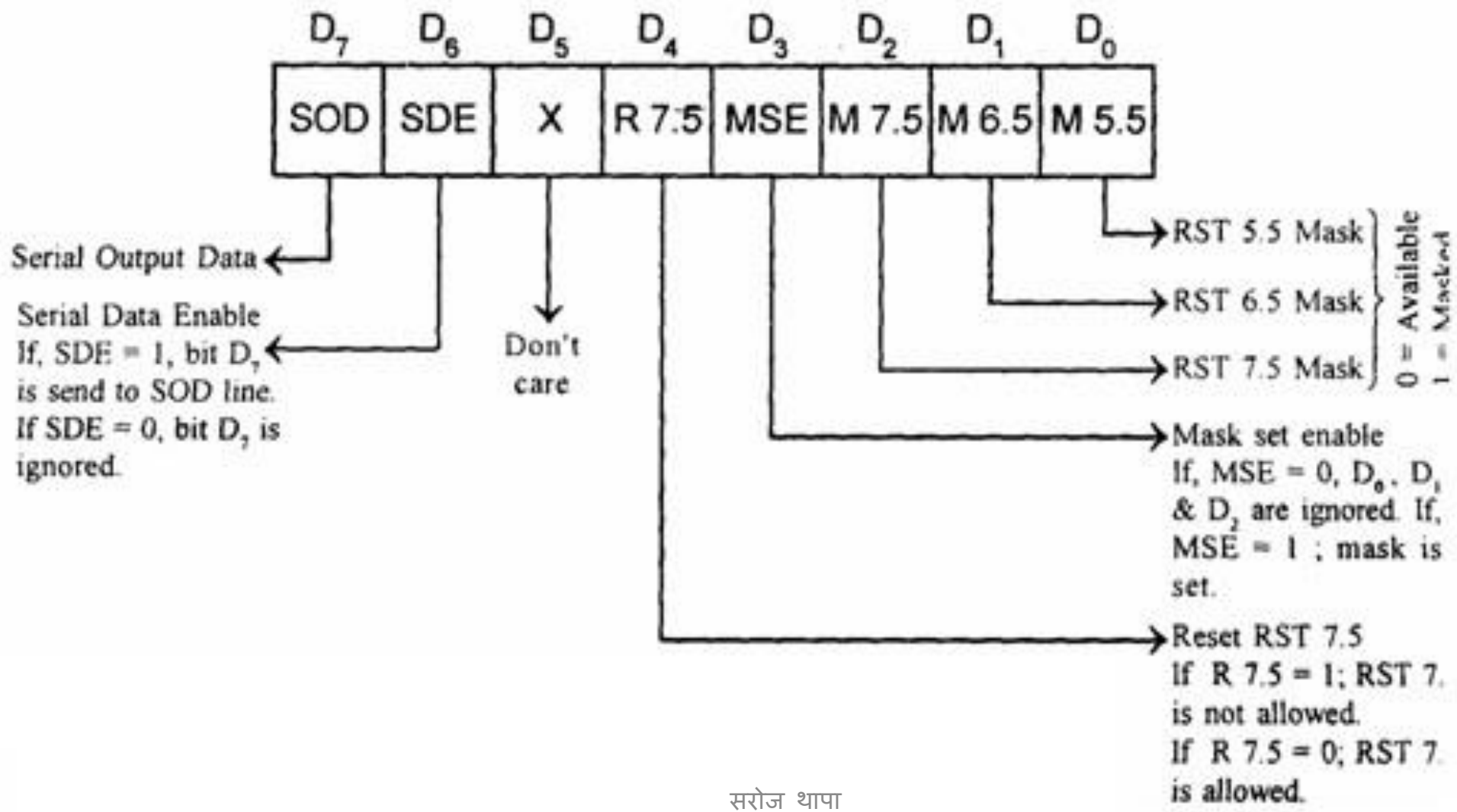
INTR:

- 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

Interrupt instructions

- **SIM 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.

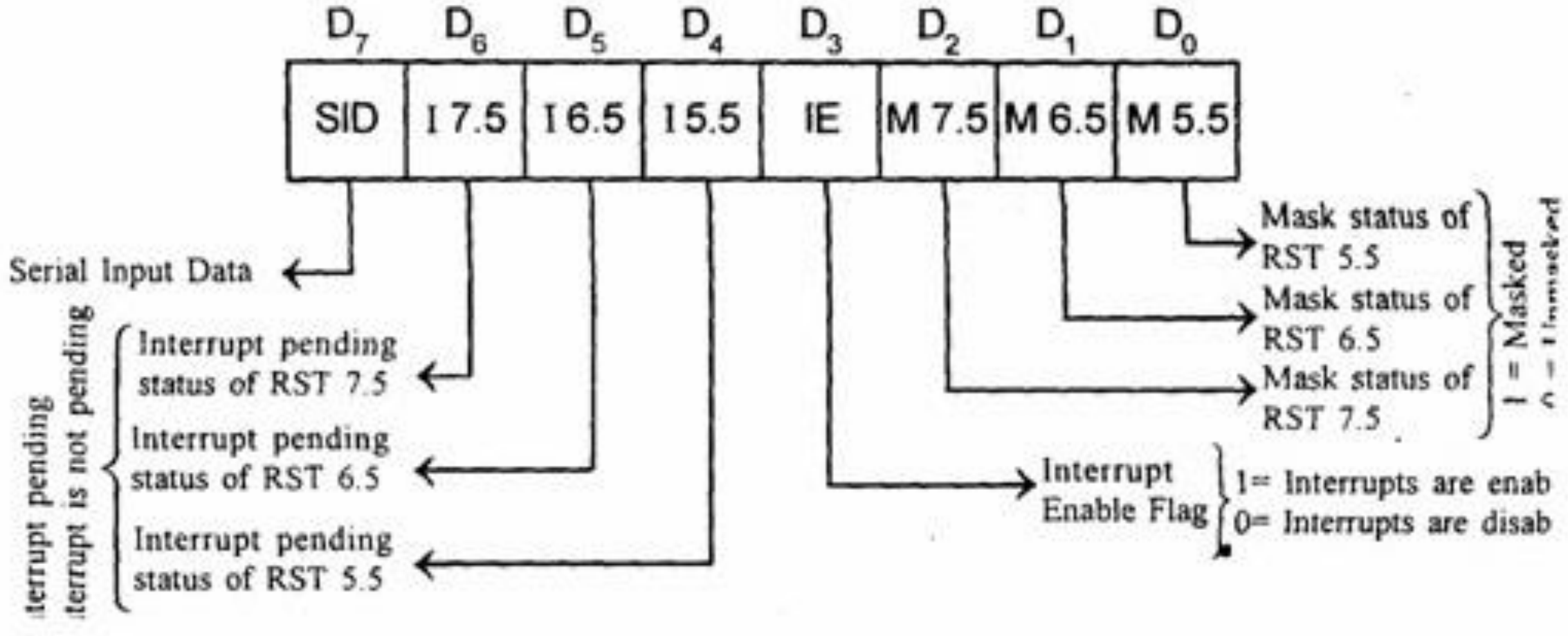
SIM Instruction



RIM instruction(Read interrupt mask)

- 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, RST 6.5, RST 5.5 and read serial data input bit.
- When RIM instruction is executed, 8 bit data is located in accumulator.

RIM Instruction



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

Programmable Interrupt Controller (PIC)

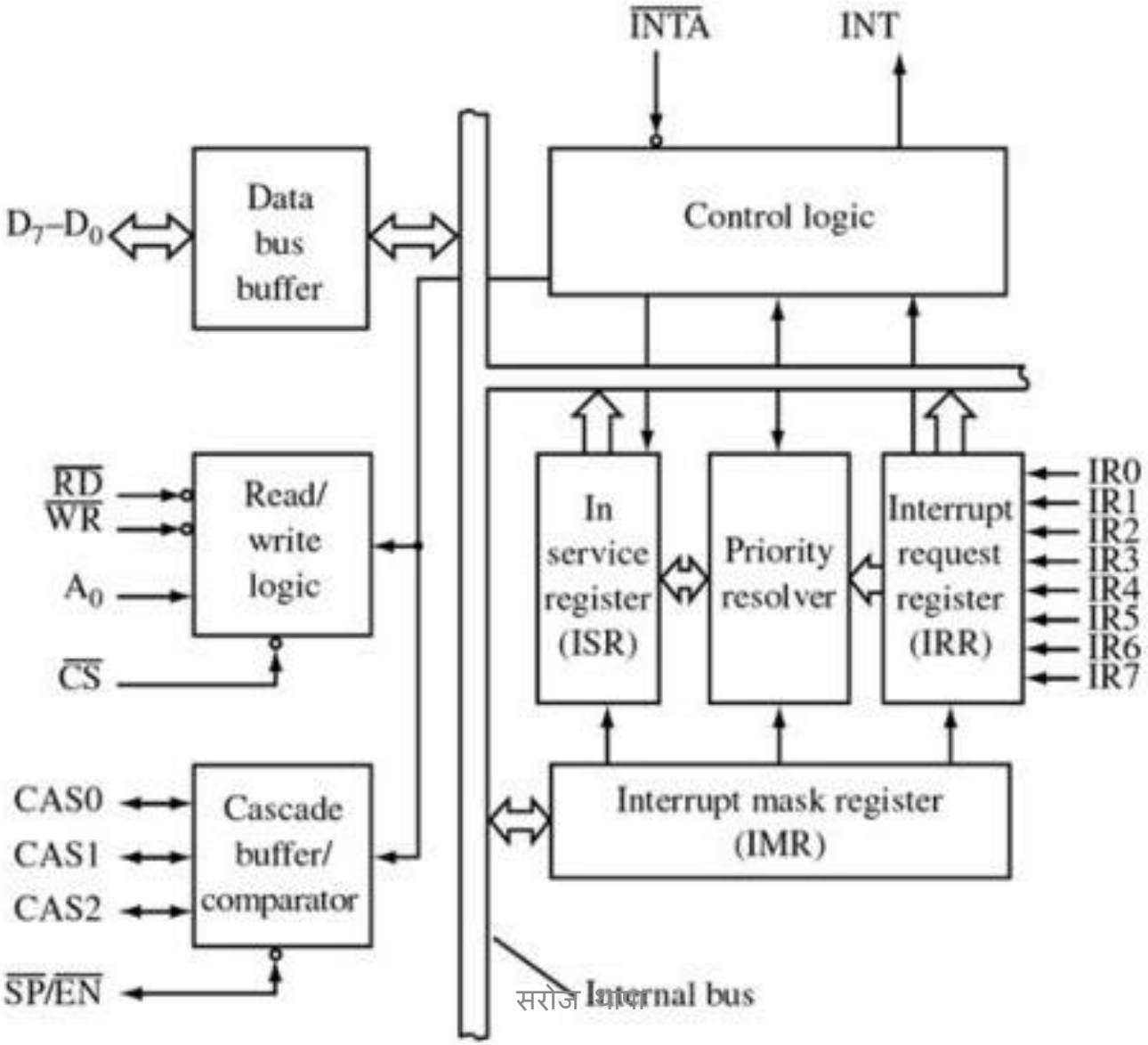
- The INTR pin can be used for multiple peripherals and to determine priorities among these devices when two or more peripherals request interrupt service simultaneously, PIC is used.
- If there are simultaneous requests, the priorities are determined by the encoder, it responds to the higher level input, ignoring the lower level input.
- The drawback of the scheme is that the interrupting device connected to input I7 always has the highest priority. The PIC includes a status register

Fig: Multiple Interrupts using PIC

8259 interrupt controller

- It is programmed to work with either 8085 or 8086 processor.
- It manage 8-interrupts according to the instructions written into its control registers.
- The interrupts can be masked or unmasked individually.
- The 8259s can be cascaded to accept a maximum of 64 interrupts.

8259A Interrupt Controller



Explanation

Data bus buffer-

- It is used to transfer data between microprocessor and internal bus.

Read/write logic-

- It sets the direction of data bus buffer.
- It controls all internal read/write operations.
- It contains initialization and operation command registers.

Cascaded buffer and comparator-

- To increase the Interrupt handling capability, we can further cascade more number of pins by using cascade buffer. So, during increment of interrupt capability, CSA lines are used to control multiple interrupt structure.
- SP/EN (Slave program/Enable buffer) pin is when set to high, works in master mode else in slave mode.

Cont..

Control logic-

- It controls read/write control logic, cascade buffer/comparator, in service register, priority resolver and IRR.

Interrupt request register-

- It is used to store all pending interrupt requests.

In service register (InSR)-

- It is used to store all interrupt levels currently being serviced.
- Each bit of this register is set by priority resolver and reset by end of interrupt command word.
- The microprocessor can read contents of this register by issuing appropriate command word.

Cont..

Priority resolver-

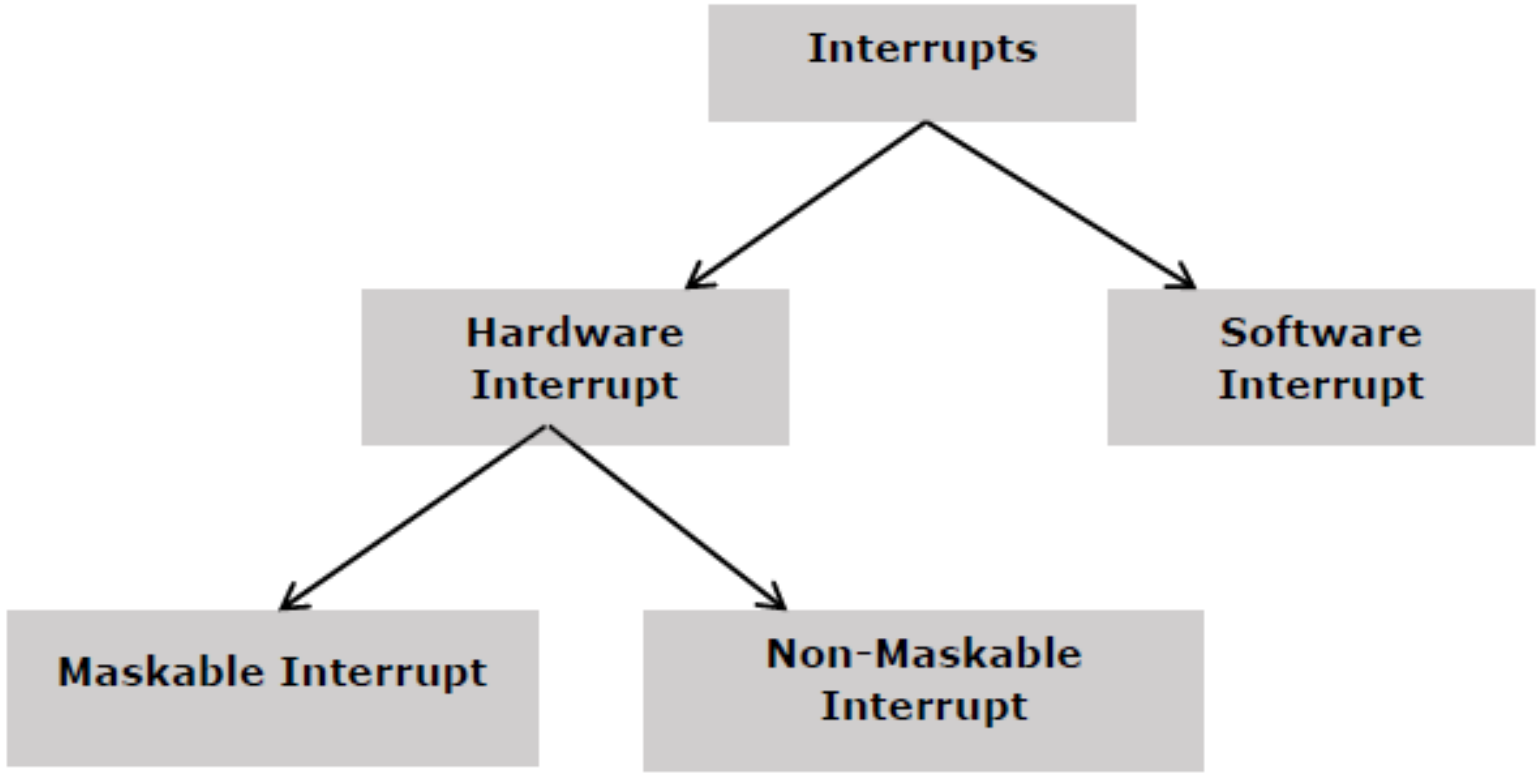
- It determines the priorities of the bit set in the IRR. To make decision, the priority resolver looks at the ISR.
- If the higher priority bit in the ISR is set then it ignores the new request.

Interrupt mask register (IMR)-

- It is a programmable register.
- It is used to mask unwanted interrupt request by writing appropriate command word.
- The microprocessor can read contents of this register without issuing any command word.

Interrupts in 8086

The following image shows the types of interrupts we have in a 8086 microprocessor −



Hardware Interrupts

- Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.
- The 8086 has two hardware interrupt pins, i.e. NMI and INTR. NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority. One more interrupt pin associated is INTA called interrupt acknowledge.

NMI

- It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)and it is of type 2 interrupt.

When this interrupt is activated, these actions take place –

- Completes the current instruction that is in progress.
- Pushes the Flag register values on to the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.

- IP is loaded from the contents of the word location 00008H.
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.
- INTR

INTR

- The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction.
- It should not be enabled using clear interrupt Flag instruction.
- The INTR interrupt is activated by an I/O port.
- If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends ‘0’ on INTA pin twice.
- The first ‘0’ means INTA informs the external device to get ready and during the second ‘0’ the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

INTR cont…

- These actions are taken by the microprocessor −
- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location X × 4
- CS is loaded from the contents of the next word location.
- Interrupt flag and trap flag is reset to 0

Software Interrupts

- Some instructions are inserted at the desired position into the program to create interrupts. These interrupt instructions can be used to test the working of various interrupt handlers. It includes –
- INT- Interrupt instruction with type number
- It is 2-byte instruction. First byte provides the op-code and the second byte provides the interrupt type number. There are 256 interrupt types under this group.
- Its execution includes the following steps –
- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location ‘type number’ × 4
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

Cont....

- The starting address for type0 interrupt is 000000H, for type1 interrupt is 00004H similarly for type2 is 00008H and ……so on. The first five pointers are dedicated interrupt pointers. i.e. –
- **TYPE 0** interrupt represents division by zero situation.
- **TYPE 1** interrupt represents single-step execution during the debugging of a program.
- **TYPE 2** interrupt represents non-maskable NMI interrupt.
- **TYPE 3** interrupt represents break-point interrupt.
- **TYPE 4** interrupt represents overflow interrupt.
- The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interrupts from 32 to Type 255 are available for hardware and software interrupts.

Dedicated interrupts (INT 0.....INT 4)

- **INT 0**
- During division an error occurs and hence invoking this interrupt. The error can arise in cases when the divisor is smaller or zero than the dividend.
- This ISR is at location 00000H in the interrupt vector table.
- **INT 1**
- Whenever the TF bit is set this interrupt is executed.
- The ISR address for this interrupt is having memory location 00004H in the interrupt vector table.
- **INT 2**
- The INT2 is a non maskable interrupt. It has an ISR address of 00008H in the interrupt vector table.

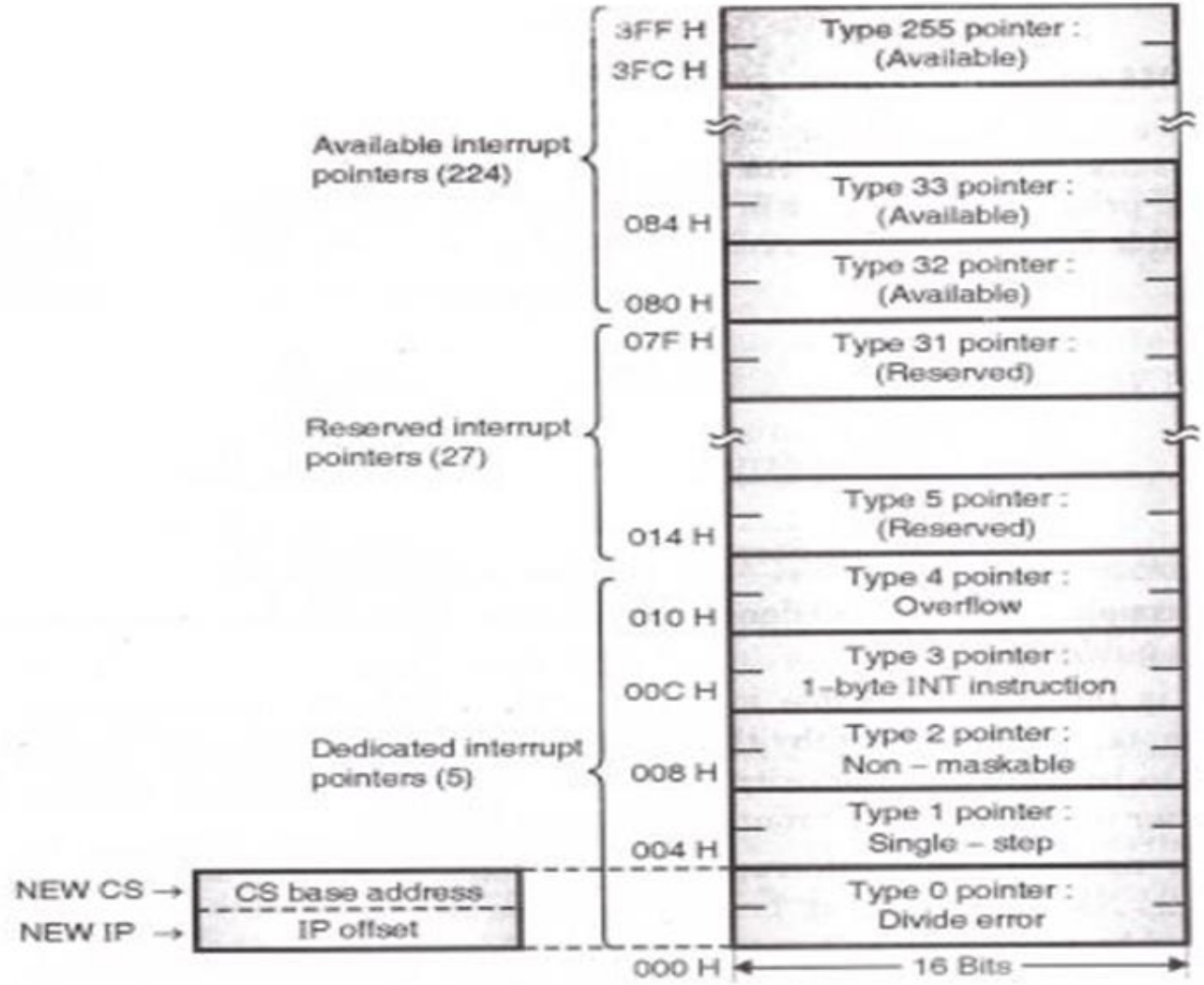
INT 3

- This interrupt can be active through instruction INT or INT 03H and it also causes breakpoints in the program.
- If the programs are very large then this interrupt helps in debugging it.
- This interrupt has the ISR location of 0000CH in the interrupt vector table.

INT 4

- This interrupt is overflow interrupt.
- When overflow bit is set then they are active. This interrupt is active after the execution of INTO instruction.
- Then overflow bit is usually set after the execution of signed arithmetic operations.
- This interrupt also has the ISR location of 00010H in the interrupt vector table.

Interrupt Vector table



Interrupt vector table

Description

- The interrupt pointer is the connection between the interrupt type code and the process which is also assigned to service interrupt.
- The four byte memory is assigned for each interrupt.
- The NEW CS is the base address of the segment which has a higher address word. Hence, NEW CS is responsible for providing the new address from where the interrupt service routine starts.
- The lower address word has the procedure offset which has NEW IP.
- We can segregate the interrupt vector table into three groups such as Dedicated interrupts which are from (INT 0…..INT 4). The second is Reserved interrupts which is from (INT 5…..INT 31)and lastly Available interrupts (INT 32…..INT 225).

End of 3.3