Unit 5: Basic I/O, Memory R/W and Interrupt Operations

Syllabus

- **Memory mapped I/O, I/O Mapped I/O and Hybrid I/O**
- **Direct Memory Access (DMA)**
  - Introduction, Advantage and Application
  - 8237 DMA Controller and Interfacing
- **Interrupt**
  - 8085 Interrupt Pins and Priority
  - Maskable and Non-maskable Interrupts
  - RST Instructions
  - Vector and Polled Interrupt
- **8259 Interrupt Controller**
  - Block Diagram and Explanation
  - Priority Modes and Additional Features

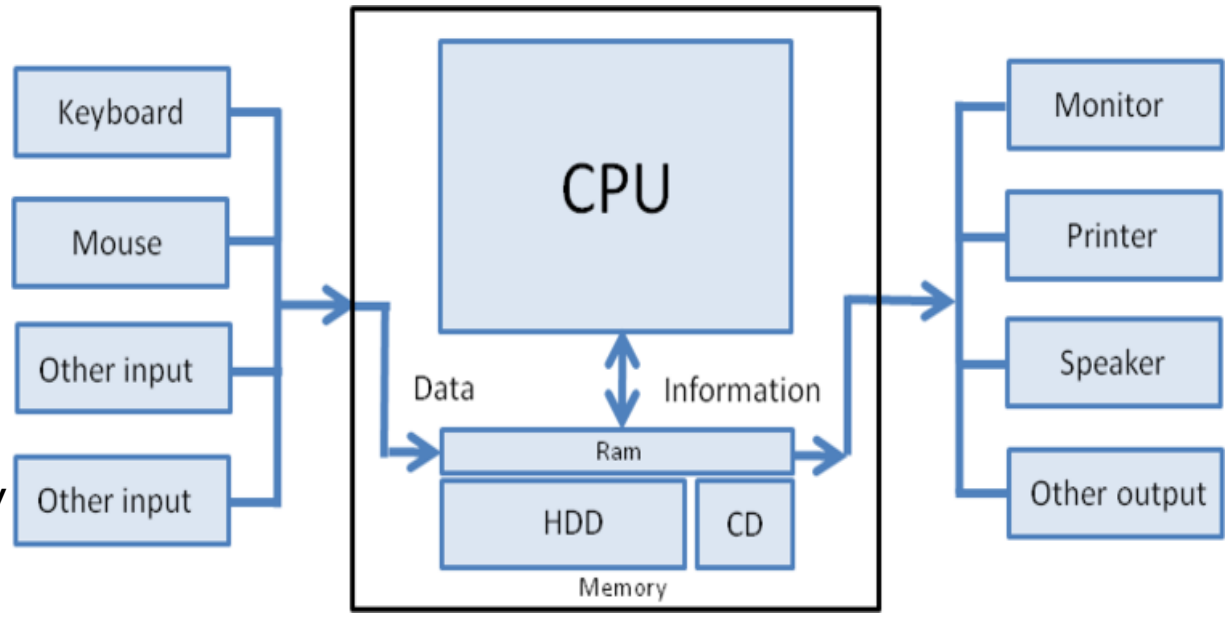
Introduction

- The microprocessor cannot do anything by itself therefore, It needs to be linked with memory, extra peripherals, or IO devices. This linking is called **Interfacing**.
- The interfacing of the I/O devices in 8085 can be done in two ways :

1. **Memory-Mapped I/O Interfacing:** In this kind of interfacing, we assign a memory address that can be used in the same manner as we use a normal memory location.
2. **I/O Mapped I/O Interfacing:** A kind of interfacing in which we assign an 8-bit address value to the input/output devices which can be accessed using IN and OUT instruction is called I/O Mapped I/O Interfacing.

I/O Operations

- CPU uses two methods to perform input/output operations between the CPU and peripheral devices in the computer. These two methods are called **memory mapped IO** and **IO mapped IO**.
- **Memory-mapped IO** uses the same address space to address both memory and I/O devices.
- On the other hand, **IO mapped IO** uses separate address spaces to address memory and IO devices.

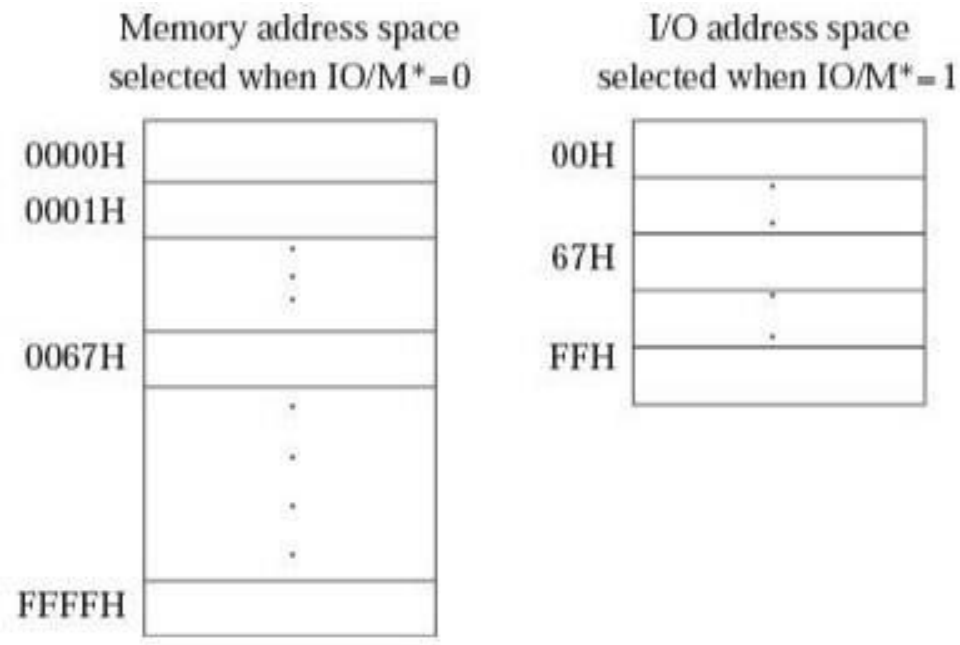


Memory Mapped I/O

- **Memory mapped IO** uses one address space for memory and input and output devices. In other words, some addresses are assigned to memory while others are assigned to store the addresses of IO devices.
- There is one set of read and write instruction lines. The same set of instructions work for both memory and IO operations. Therefore, the instructions used to manipulate memory can be used for IO devices too. Hence, it can lessen the addressing capability of memory because some are occupied by the IO.

Isolated (I/O) Mapped I/O

- **IO mapped IO** uses two separate address spaces for memory locations and for IO devices. There are two separate control lines for both memory and IO transfer. In other words, there are different read-write instruction for both IO and memory.
- IO read and IO write are for IO transfer whereas memory read and memory write are for memory transfer. IO mapped IO is also called port-mapped IO or isolated IO.



Hybrid I/O

- Combination of both previous I/O.
- Uses same address in some cases and two address space in some cases.

I/O Mapped I/O and Memory Mapped I/O

I/O Mapped I/O	Memory Mapped I/O
I/O device is treated as an I/O device and hence given an I/O address.	I/O device is treated like a memory device and hence given a memory address.
I/O device has an 8 bit address.	I/O device has a 16 bit Memory address.
I/O device is given $\overline{IOR}$ and $\overline{IOW}$ control signals.	I/O device is given $\overline{MEMR}$ and $\overline{MEMW}$ control signals.
Decoding is easier due to lesser address lines.	Decoding is more complex due to more address lines.
Decoding is cheaper	Decoding is more expensive
Works faster due to less delays	More gates add more delays hence slower
Allows max. $2^8 = 256$ I/O devices	Allows many more I/O devices as I/O addresses are 16 bits($2^{16} = 64K$).
I/O devices can only be accessed by IN and OUT instructions.	I/O devices can be accessed using any memory instruction.
Popular technique in Microprocessors.	Popular technique in Microcontrollers.

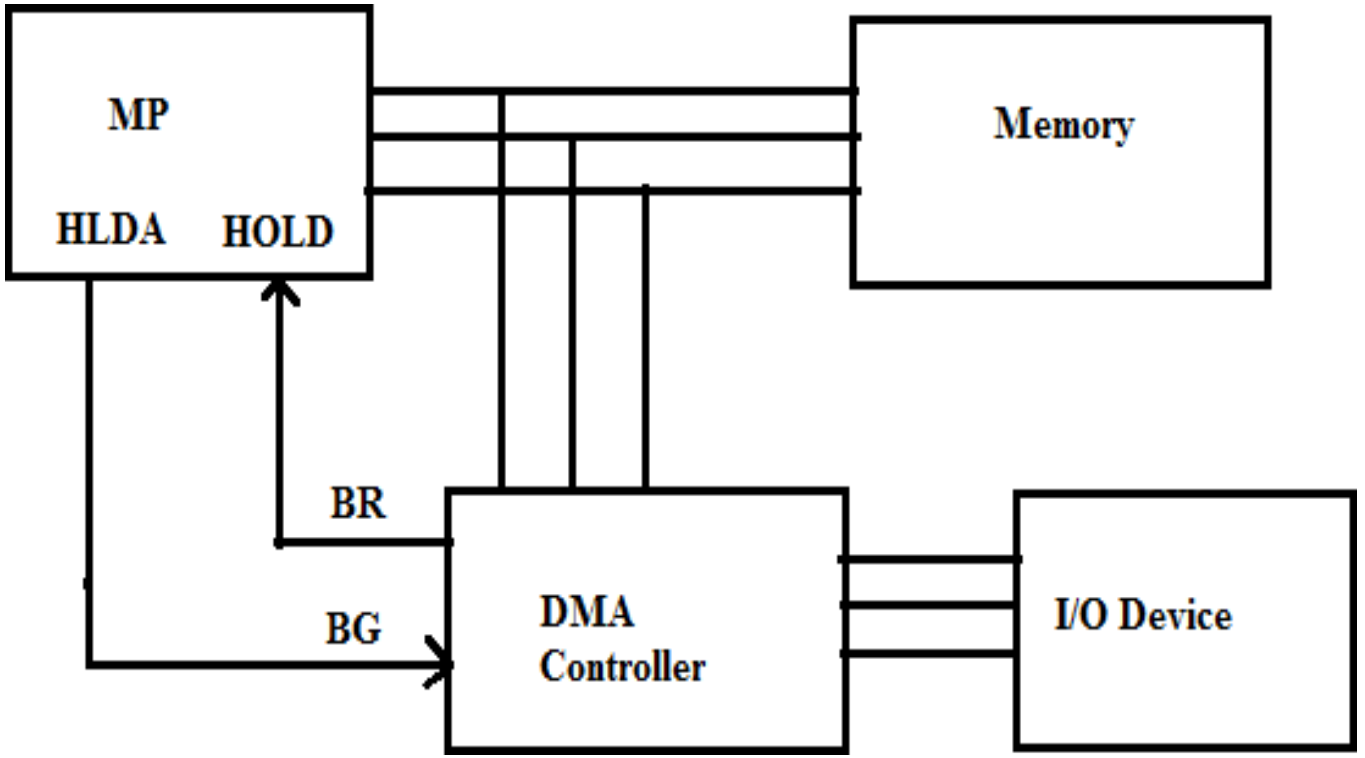
Memory Mapped IO and IO Mapped IO in 8085

Memory Mapped IO	IO Mapped IO
❖ Here IO devices treated as Memory	❖ Here IO devices treated as IO.
❖ 16 bits addressing ($A_0 - A_{15}$)	❖ 8 bits addressing ($A_0 - A_7$)
❖ It can address = 2^{16} = 64K Address	❖ It can address = 2^8 = 256 Address
❖ Number of devices = 64K = 65536	❖ Number of devices = 256
❖ Due to many devices we need more decoder hardware in Memory Mapped IO.	❖ Due to less devices we need less decoder hardware in IO Mapped IO.
❖ Due to limited Address lines (16) with 8085 available memory less with memory mapped IO.	❖ Available memory more with IO mapped IO as number of devices are less in IO mapped IO.
❖ Here we use $\overline{MEMR}$ and $\overline{MEMW}$ control signals.	❖ Here we use $\overline{IOR}$ and $\overline{IOW}$ control signals.
❖ Data transfer happens between any registers and IO.	❖ Data transfer only between Accumulator and IO.
❖ ALU operation is done with all registers.	❖ Not Available.
❖ Instructions Example: LDA XXXXH, STA XXXXH, MOV A,M	❖ Instructions Example: IN XXXXH, OUT XXXXH

Conclusion

- Memory mapped IO and IO mapped IO are two methods to perform input/output operations between the CPU and peripheral devices in the computer.
- The basic difference between memory mapped IO and IO mapped IO is that memory mapped IO uses the same address space for both memory and IO device while IO mapped IO uses two separate address spaces for memory and IO device.

Direct Memory Access (DMA)



Direct Memory Access (DMA) Operation

- DMA is a process of communication for data transfer between memory and input/output, controlled by an external circuit called **DMA controller**, without involvement of CPU.
- 8085 MP has two pins **HOLD** and **HLDA** which are used for DMA operation.
- First, DMA controller sends a request by making Bus Request (BR) control line high. When MP receives high signal to HOLD pin, it first completes the execution of current machine cycle, it takes few clocks and sends HLDA signal to the DMA controller.
- After receiving HLDA through Bus Grant (BG) pin of DMA controller, the DMA controller takes control over system bus and transfers data directly between memory and I/O without involvement of CPU. During DMA operation, the processor is free to perform next job which does not need system bus.
- At the end of data transfer, the DMA controller terminates the request by sending low signal to HOLD pin and MP regains control of system bus by making HLDA low.

Advantages & Disadvantages of DMA

- **Advantages**
  - DMA allows a peripheral device to read from/write to memory without going through the CPU.
  - DMA allows faster processing since the processor can be working on something else while the peripheral can be populating memory.
  - DMA enables more efficient use of interrupts.
  - High transfer rates.
  - DMA capable device can communicate directly with memory.
- **Disadvantages**
  - Cost of DMA hardware.
  - Data has to be stored in continuous memory locations.
  - DMA controller is slow in comparison to CPU.

Application of DMA

- DMA has been a built-in feature of PC architecture since the introduction of the original IBM PC.
- PC-based DMA was used for floppy disk I/O in the original PC and for hard disk I/O in later versions.
- PC-based DMA technology, along with high speed bus technology, is driven by data storage, communications, and graphics needs-all of which require the highest rates of data transfer between system memory and I/O devices.
- Applications areas are**: cinemas, theatres, hotels, railway stations, shopping centres, trade shows, museums & many more**.

Features of 8257 DMA Controller

Here is a list of some of the prominent features of 8257:

- It has four channels which can be used over four I/O devices.
- Each channel has 16-bit address and 14-bit counter.
- Each channel can transfer data up to 64kb.
- Each channel can be programmed independently.
- Each channel can perform read transfer, write transfer and verify transfer operations.
- It generates MARK signal to the peripheral device that 128 bytes have been transferred.
- It requires a single phase clock.
- Its frequency ranges from 250Hz to 3MHz.
- It operates in 2 modes, i.e., **Master mode** and **Slave mode**.

8257 Direct Memory Access

❖ Basics of 8257 Direct Memory Access

- ❑ 8257 is used to for high speed data transfer in between in IO devices and memory.
- ❑ Using Microprocessor data transfer is slow, as microprocessor have to execute instructions and it have to check interrupt as well.
- ❑ Using 8257, MPU releases the control of the buses to the DMA.
- ❑ Here, Data transfer between memory and IO is been done by bypassing MPU.
- ❑ To take control of Address and Data bus from MPU, it has HOLD and HLDA control terminals which are used by DMA.

8257 Direct Memory Access

❖ HOLD and HLDA

❑ HOLD :

- This is active high signal input MPU.
- It gives request to MPU for address and data bus control.
- After receiving HOLD signal, MPU relinquishes the buses in following machine cycle.
- All buses are tri stated and HLDA sent out by MPU.
- MPU regains the control of buses after HOLD goes low.

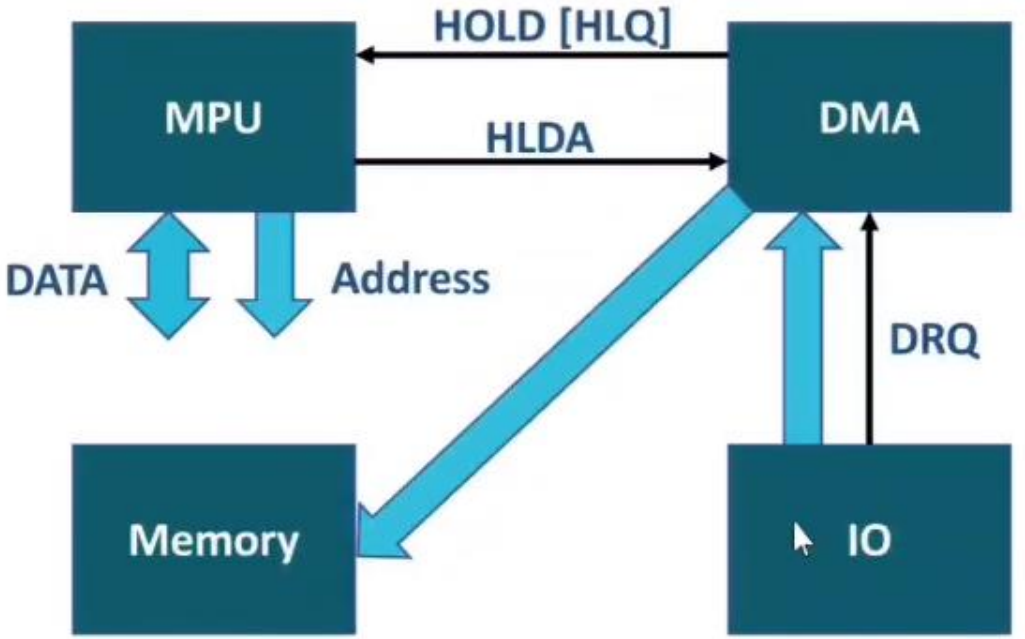
❑ HLDA :

- This is active high signal indicating that the MPU is relinquishing control of buses.

8257 Direct Memory Access

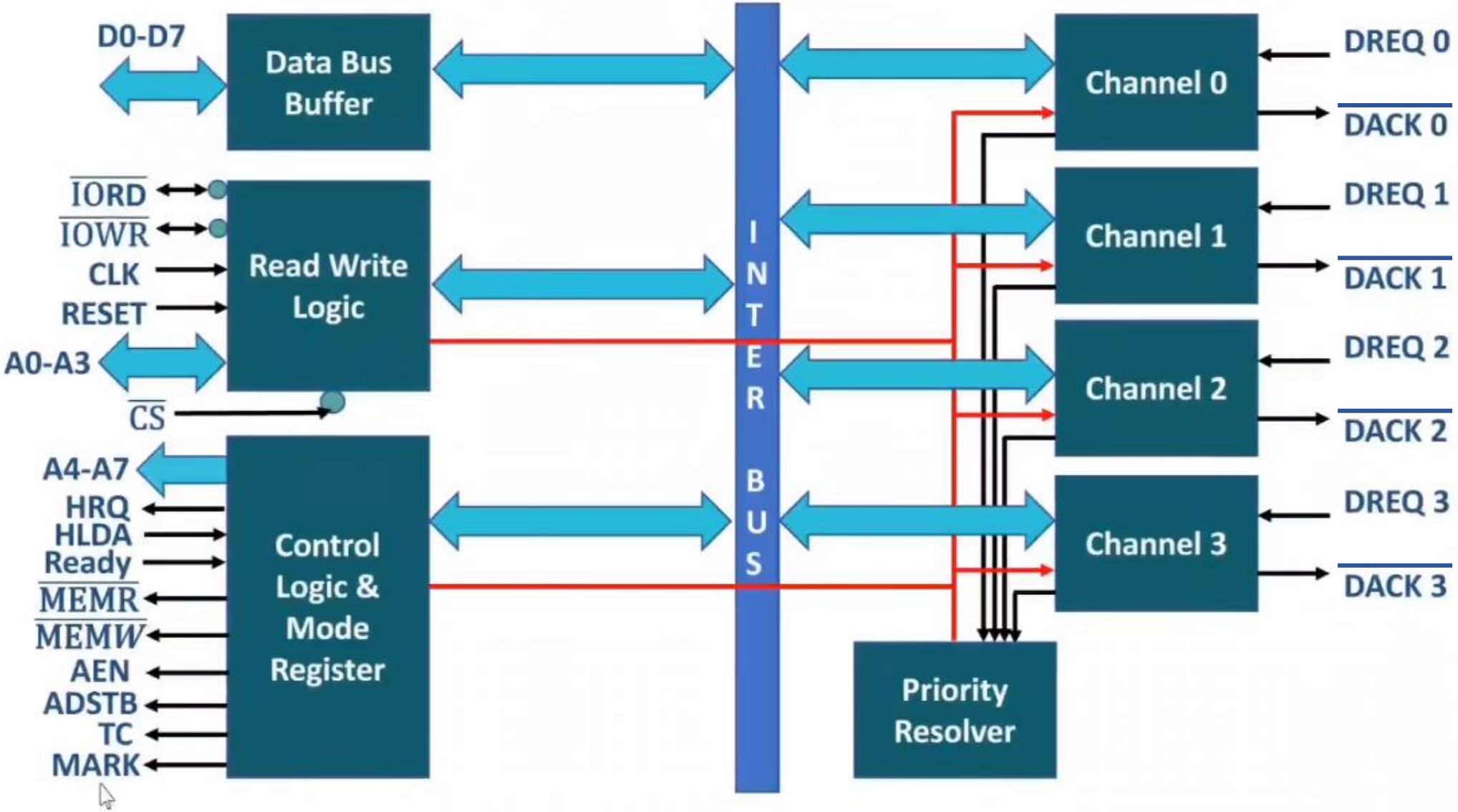
❖ Working of DMA

- If IO wants to send data to memory, then it will send request to DMA. [DRQ]
- To take control of system buses, DMA will send HOLD signal to MPU.
- To give control of address and data, MPU will give HLDA [HOLD Acknowledge]. Which indicates that now DMA is master of Buses. So now buses will be managed by DMA for memory and IO.



- Now data transfer can happen without involvement of MPU. Here now MPU don't need to execute instructions, so data exchange will be faster.
- Once HOLD signals goes low, MPU will take control of system buses and then MPU becomes master.

Block Diagram of 8257



Pin Description

- **$DREQ_0$ – $DREQ_3$**(DMA REQUEST)
  - These are the four individual channel DMA request inputs, which are used by the peripheral devices for using DMA services. When the fixed priority mode is selected, then DRQ0 has the highest priority and DRQ3 has the lowest priority among them.

- **$\overline{DACK_0}$– $\overline{DACK_3}$** (DMA ACKNOWLEDGE)
  - These are the active-low DMA acknowledge lines, which updates the requesting peripheral about the status of their request by the CPU. These lines can also act as strobe lines for the requesting devices.

- **D_0 – D_7** (DATA BUS)
  - These are bidirectional, data lines which are used to interface the system bus with the internal data bus of DMA controller. In the Slave mode, it carries command words to 8257 and status word from 8257. In the master mode, these lines are used to send higher byte of the generated address to the latch. This address is further latched using ADSTB signal.

- **$\overline{IOR}$** (I/O READ)
  - It is an active-low bidirectional tri-state input line, which is used by the CPU to read internal registers of 8257 in the Slave mode. In the master mode, it is used to read data from the peripheral devices during a memory write cycle.

Pin Description

- **$\overline{\text{IOW}}$** (I/O WRITE)
  - It is an active low bi-direction tri-state line, which is used to load the contents of the data bus to the 8-bit mode register or upper/lower byte of a 16-bit DMA address register or terminal count register. In the master mode, it is used to load the data to the peripheral devices during DMA memory read cycle.
- **CLK** (CLOCK INPUT)
  - It is a clock frequency signal which is required for the internal operation of 8257.
- **RESET**
  - This signal is used to RESET the DMA controller by disabling all the DMA channels.
- **A_0 - A_3** (ADDRESS)
  - These are the four least significant address lines. In the slave mode, they act as an input, which selects one of the registers to be read or written.
  - In the master mode, they are the four least significant memory address output lines generated by 8257.
- **$\overline{\text{CS}}$** (CHIP SELECT)
  - It is an active-low chip select line. In the Slave mode, it enables the read/write operations to/from 8257. In the master mode, it disables the read/write operations to/from 8257.

Pin Description

- **A_4 - A_7** (ADDRESS)
  - These are the higher nibble of the lower byte address generated by DMA in the master mode.
- **READY**
  - It is an active-high asynchronous input signal, which makes DMA ready by inserting wait states.
- **HRQ** (HOLD REQUEST)
  - This signal is used to receive the hold request signal from the output device. In the slave mode, it is connected with a DRQ input line 8257. In Master mode, it is connected with HOLD input of the CPU.
- **HLDA** (HOLD ACKNOWLEDGE)
  - It is the hold acknowledgement signal which indicates the DMA controller that the bus has been granted to the requesting peripheral by the CPU when it is set to 1.
- **$\overline{\text{MEMR}}$** (MEMORY READ)
  - It is the low memory read signal, which is used to read the data from the addressed memory locations during DMA read cycles.
- **$\overline{\text{MEMW}}$** (MEMORY WRITE)
  - It is the active-low three state signal which is used to write the data to the addressed memory location during DMA write operation.

Pin Description

- **ADSTB** (ADDRESS STROBE)
  - Address strobe output signal is used to latch the high-byte memory address issued through DB7 to DB0 lines by 8257 during DMA mode into an external latch.
- **AEN** (ADDRESS ENABLE)
  - It is used to enable the address latch connected to D7 – D0 pins of 8257. It is also used to disable any buffers in the system connected to the processor.
- **TC** (TERMINAL COUNT)
  - It stands for 'Terminal Count', which indicates the present DMA cycle to the present peripheral devices.
- **MARK**
  - The mark will be activated after each 128 cycles or integral multiples of it from the beginning. It indicates the current DMA cycle is the 128th cycle since the previous MARK output to the selected peripheral device.
- **V_{cc}** (POWER SUPPLY)
  - It is the power signal which is required for the operation of the circuit.

8085 Interrupt

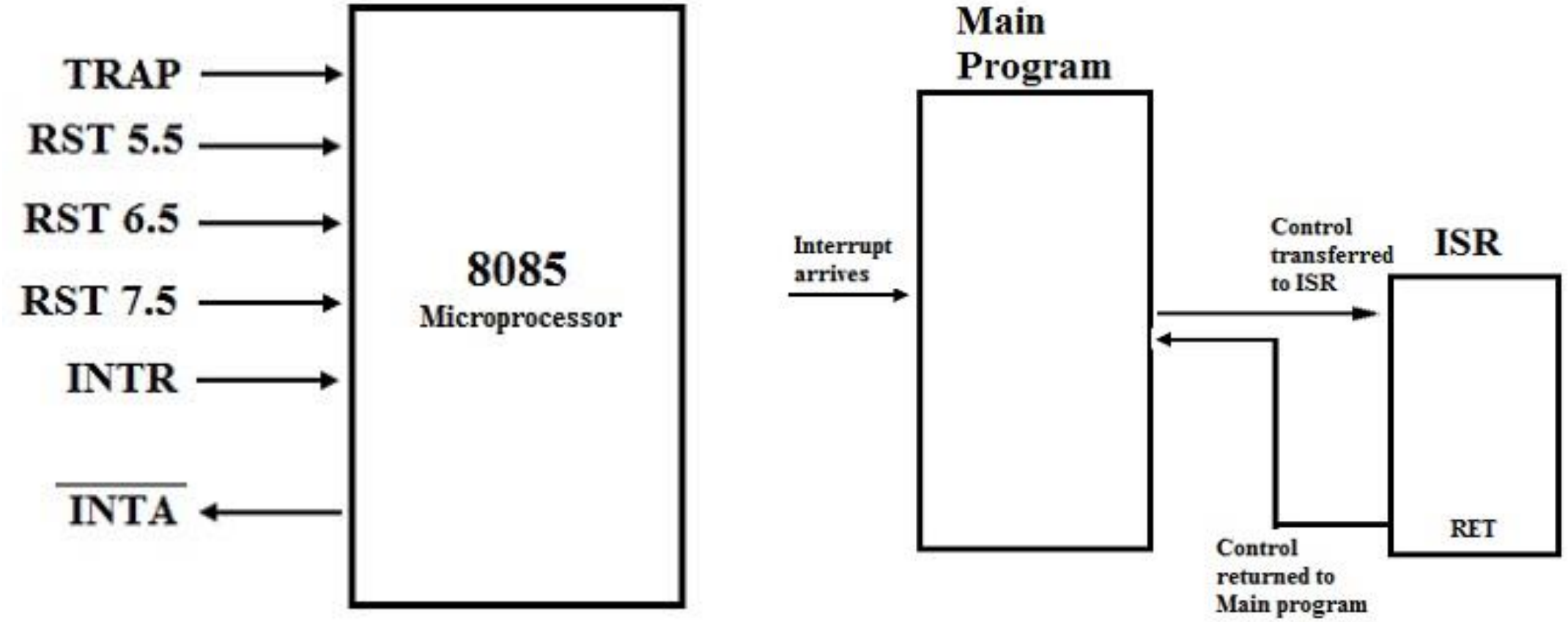
- Interrupt is a signal send by an external device informing the processor that it is ready for communication and requests attention to perform a particular task. It is useful to interface I/O device that provide or require data.
- Interrupt is a process where an external device can get the attention of the microprocessor. The process starts from the I/O device. An interrupt is considered to be an emergency signal that may be serviced. The Microprocessor may respond to it as soon as possible.
- When the Microprocessor receives an interrupt signal, it suspends the currently executing program and jumps to an **Interrupt Service Routine (ISR)** to respond to the incoming interrupt. Each interrupt will most probably have its own ISR.
- Responding to an interrupt may be immediate or delayed depending on whether the interrupt is maskable or non-maskable and whether interrupts are being masked or not. There are two ways of redirecting the execution to the ISR depending on whether the interrupt is vectored or non-vectored.
- In 8085 microprocessor, there are 5 interrupts such as **TRAP, RST 5.5, RST 6.5, RST 7.5, INTR** as shown in figure.

8085 Interrupt Pins and Interrupt Priority

- There are five interrupt pins in 8085 and one interrupt acknowledge ($\overline{INTA}$) pin.
- Pin 6 to pin 10 interrupts have the priorities from highest to lowest in decreasing order.
- Priority means which interrupt gets the acknowledgement first if more than one are interrupting the microprocessor.

Pin No.	Name	Type	Priority	ISR Location
6	TRAP	Vectored	Highest	CALL 0024H (3-byte call)
7	RST 7.5	Vectored	:	CALL 003CH (3-byte call)
8	RST 6.5	Vectored		CALL 0034H (3-byte call)
9	RST 5.5	Vectored		CALL 002CH (3-byte call)
10	INTR	Non-Vectored	Lowest	RST (Restart instructions) – 1 byte call

8085 Interrupt Pins



Notes: In additional to these hardware interrupts, 8085 microprocessor has eight software interrupts. The RESTART instructions RST 0 to RST 7 are software interrupt instructions.

8085 Interrupt

- **Vector Interrupt:** In this type of interrupt, Processor knows the address of Interrupt. In other word processor knows the address of interrupt service routine. The examples of vector interrupt are RST 7.5, RST 6.5, RST 5.5, TRAP.
- **Non-Vector Interrupt:** In this type of interrupt, Processor cannot know the address of Interrupt. It should give externally. The device will have to send the address of interrupt service routine to processor for performing Interrupt. In other word, if the ISR address of an interrupt is to be taken from the interrupting source itself, it is called a non-vectored interrupt. The example of Non-vector interrupt is INTR.
- When a device interrupts, it actually wants the MP to give a service which is equivalent to asking the MP to call a subroutine. This subroutine is called ISR (Interrupt Service Routine).
- **Interrupt Service Routine (ISR):** A small program or a routine that when executed, services the corresponding interrupting source is called an ISR.

8085 Interrupt

- **Software Interrupt:** It is an instruction based Interrupt which is completely controlled by software. That means programmer can use this instruction to execute interrupt in main program. There are eight software interrupts available in 8085 microprocessor. See the example with their hex code and vector address.

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

8085 Interrupt

- **Hardware Interrupt:** As name suggests it is interrupt which can get the interrupt request in hardware pin of microprocessor 8085. There are mainly six dedicated pins available for interrupt purpose.
- Those are TRAP, RST 7.5, RST 6.5, RST 5.5, INTR, $\overline{\text{INTA}}$ (It is not an Interrupt pin but it is used to send acknowledgement of the Interrupt request getting from other interrupt pin).

INTERRUPT	VECTOR ADDRESS
TRAP (RST 4.5)	0024 H
RST 5.5	002C H
RST 6.5	0034 H
RST 7.5	003C H

Software Interrupt Vs Hardware Interrupt

S. N.	Software Interrupt	Hardware Interrupt
1.	It is a synchronous event.	It is an asynchronous event.
2.	This interrupt is requested by executing instruction.	This interrupt is requested by an external device on a pin.
3.	PC is incremented.	PC is not incremented.
4.	The microprocessor does not execute any interrupt acknowledge cycle to acknowledge this interrupt. The microprocessor executes a normal instruction cycle.	The microprocessor executes either interrupt acknowledge cycle bus or idle machine cycle to acknowledge this interrupt.
5.	It cannot be ignored or masked.	It can be masked except for TRAP.
6.	It has the highest priority among all interrupts.	The priority is lower than that of a software interrupt.
7.	It does not affect interrupt control logic.	It affects interrupt control logic.
8.	It is not used to interface peripherals that means it does not improve the throughput of the system. It is used in program debugging.	It is used to interface peripherals in interrupt-driven I/O. It improves the throughput of the system.

Maskable and Non-Maskable Interrupt

- **Maskable interrupts:** An interrupt which can be disabled by software that means we can disable the interrupt by sending appropriate instruction, is called a maskable interrupt. RST 7.5, RST 6.5, and RST 5.5 are the examples of Maskable Interrupt.
- **Non-Maskable interrupts:** As name suggests we cannot disable the interrupt by sending any instruction is called Non Maskable Interrupt. TRAP interrupt is the non-maskable interrupt for 8085. It means that if an interrupt comes via TRAP, 8085 will have to recognize the interrupt we cannot mask it.

Comparison Chart

BASIS OF COMPARISON	MASKED INTERRUPTS	NON-MASKABLE INTERRUPTS
Basic Concept	The processor can accept or reject the maskable interrupts.	The processor has to accept the non-maskable interrupts.
Processing	The maskable interrupt is processed immediately if the interrupt occurred has higher priority than currently executing instruction. Else the interrupt is processed after completing the current execution.	The non-maskable interrupt has to be processed as soon as it occurs by suspending the current execution.
Priority	Interrupts with low priority are the maskable interrupts.	Interrupts with high priority are non-maskable interrupts.
Vector Address	The vector address of maskable interrupt can be altered by programming the Interrupt Controller.	The vector address of non-maskable interrupt are fixed and are predefined by the manufacturer of the processor.
Use	Maskable interrupts are used by the device controllers.	Used by watch dog timer and during power failure.
Example	In 8085 microprocessor all the hardware interrupt excluding TRAP are maskable interrupt.	In 8085 the interrupts raised TRAP pin and all software interrupts are non-maskable interrupt.

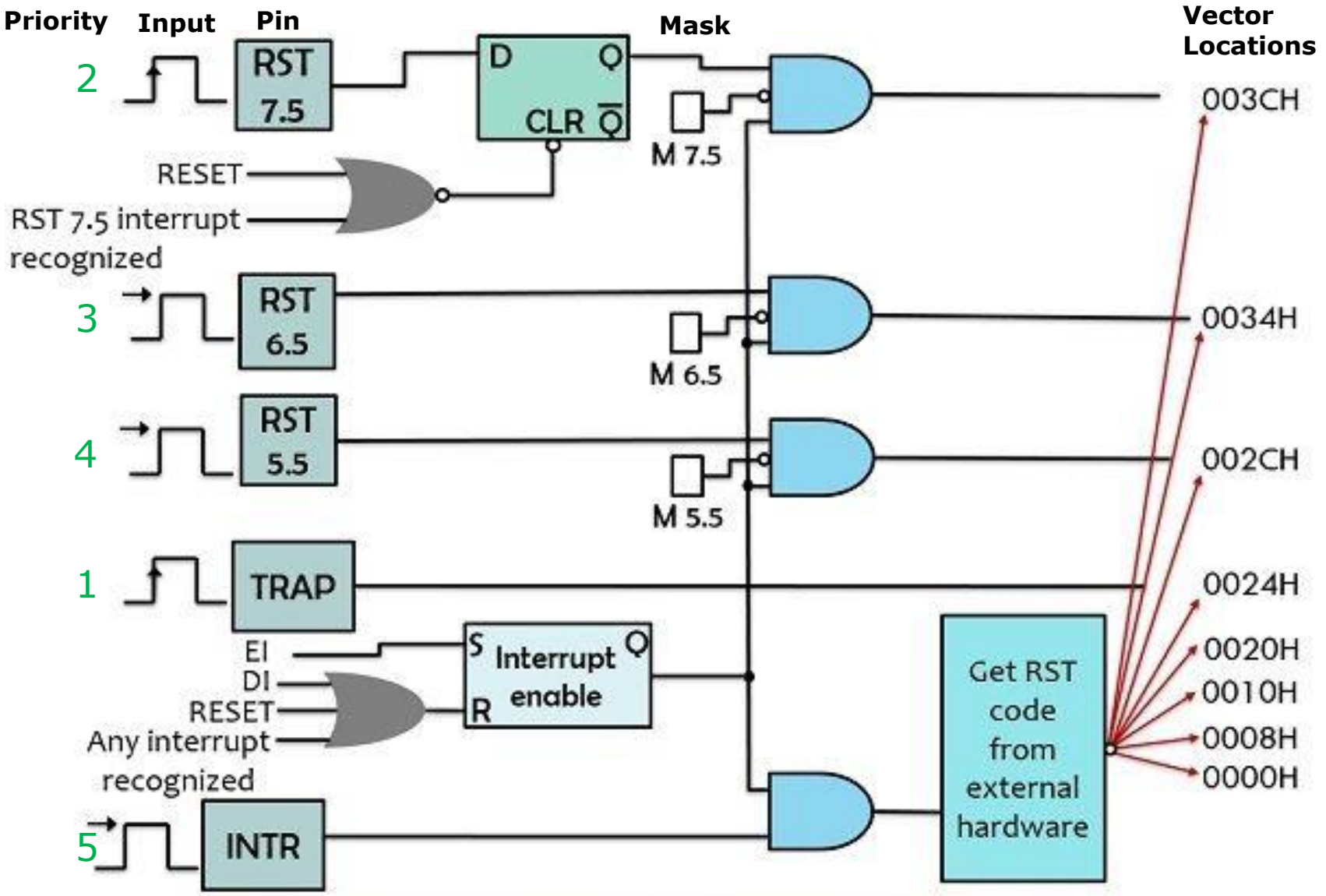
Interrupt priority

- **TRAP**
  - This interrupt can be considered as a non-maskable interrupt. Any mask or interrupt enable cannot affect this.
  - It is a vectored interrupt. The interrupt vector address is 0024H.
  - TRAP has the highest priority level.
  - We can say that TRAP interrupt is level and edge triggered. This means that till the acknowledgement, the TRAP must go high and remain high.
  - In emergency situations like sudden power failure, it executes an ISR and sends the data from main memory to backup memory.
- **RST 7.5**
  - The RST 7.5 can be considered as a maskable interrupt.
  - It is a vectored interrupt. The interrupt vector address is 003CH.
  - It has second highest priority.
  - It is edge triggered. i.e., Input attains at high and no need to retain the high state until it is recognized and acknowledged.

Interrupt priority

- **RST 6.5**
  - The RST 6.5 interrupt is a maskable interrupt.
  - It is a vectored interrupt. The interrupt vector address is 0034H.
  - It has the third highest priority.
  - It is level triggered. i.e., Input goes to high and stays high until it is recognized and acknowledged.
- **RST 5.5**
  - The RST 5.5 interrupt is a maskable interrupt.
  - It is a vectored interrupt. The interrupt vector address is 002CH.
  - It has the fourth highest priority.
  - It is level triggered. i.e., Input goes to high and stays high until it is recognized and acknowledged.
- **INTR**
  - INTR is a maskable interrupt.
  - It is a non- vectored interrupt. After receiving $\overline{\text{INTA}}$, the peripheral has to supply the address of ISR.
  - It has the lowest priority. It is a level triggered. i.e., Input goes to high and it is necessary to maintain high state until it is recognized and acknowledged.

8085 Interrupt and Vector Locations



Interrupt structure of 8085 microprocessor

Process of INTR interrupt

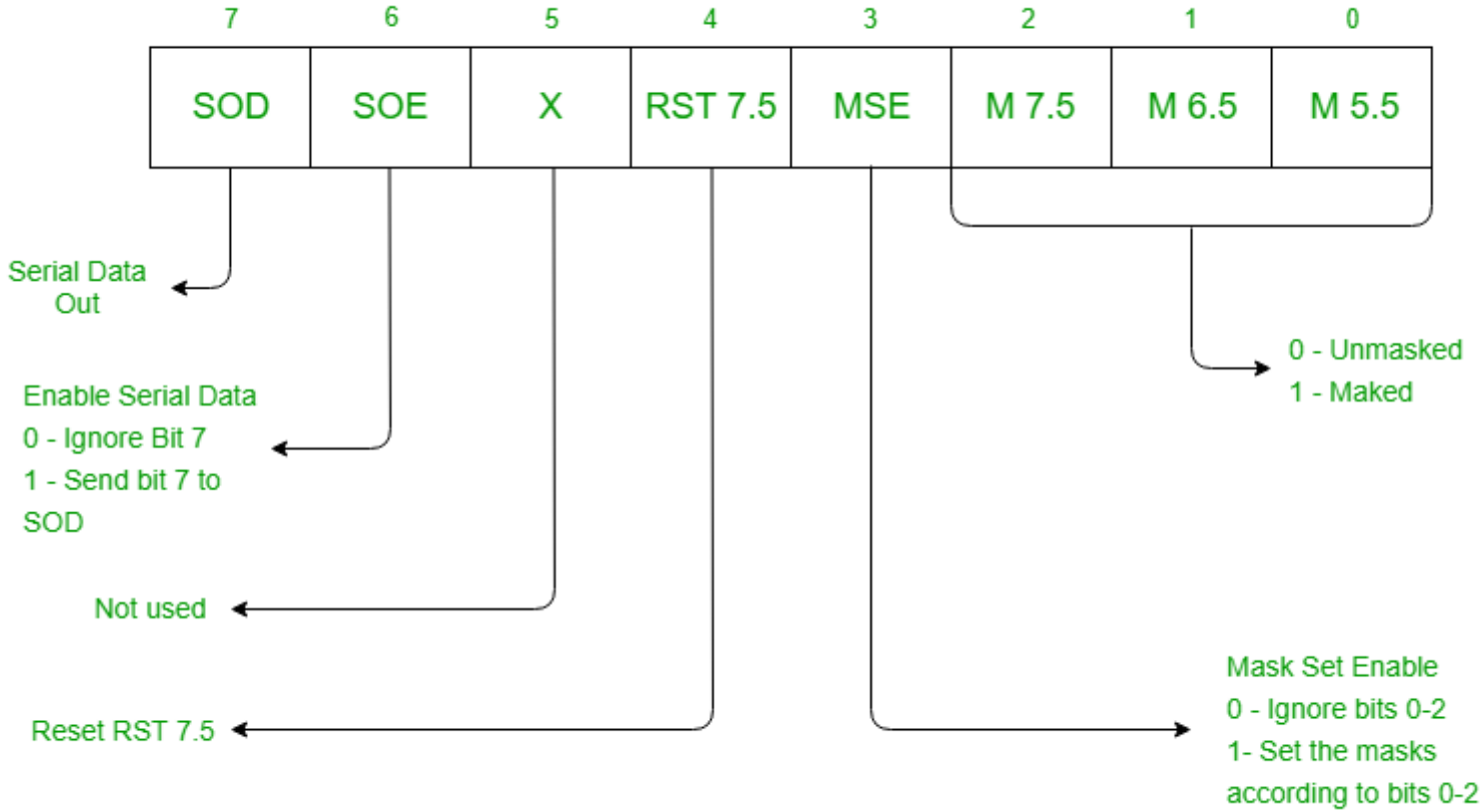
1. With the use of the EI instruction, the interrupt process should be enabled.
2. Whenever an instruction is executed, the 8085 checks for an interrupt signal.
3. If INTR is high, the microprocessor completes current instruction, disables the interrupt and sends $\overline{\text{INTA}}$ signal to the peripheral device.
4. $\overline{\text{INTA}}$ allows the peripheral device to send an RST instruction through data bus.
5. Upon receiving the $\overline{\text{INTA}}$ signal, the microprocessor saves the memory location of the next instruction on the stack and the program is transferred to 'call' location (ISR Call) specified by the RST instruction.
6. Microprocessor executes the ISR.
7. Within the program, In order to enable the further interrupt, ISR must include the 'EI' instruction.
8. The RET instruction at the end of the ISR retrieves the return address from the stack and the program is transferred back to main program which was interrupted.

Instruction for Interrupts

1. **Enable Interrupt (EI):** The interrupt enable flip-flop is set and all interrupts are enabled following the execution of next instruction followed by EI. No flags are affected. After a system reset, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to enable the interrupts again (except TRAP).
2. **Disable Interrupt (DI):** This instruction is used to reset the value of enable flip-flop hence disabling all the interrupts. No flags are affected by this instruction.

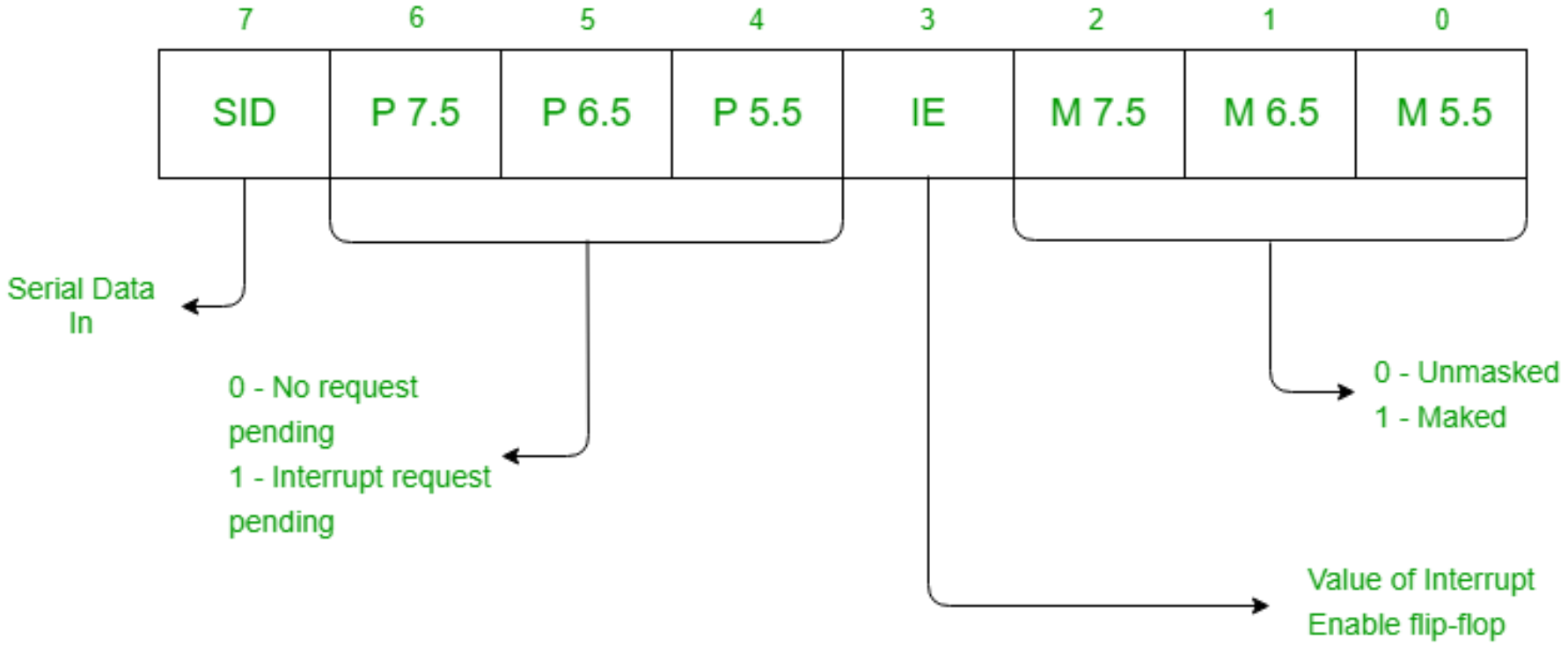
Instruction for Interrupts

3. **Set Interrupt Mask (SIM):** It is used to implement the hardware interrupts (RST 7.5, RST 6.5, RST 5.5) by setting various bits to form masks or generate output data via the Serial Output Data (SOD) line. First the required value is loaded in accumulator then SIM will take the bit pattern from it.



Instruction for Interrupts

4. **Read Interrupt Mask (RIM):** This instruction is used to read the status of the hardware interrupts (RST 7.5, RST 6.5, RST 5.5) by loading into the A register a byte which defines the condition of the mask bits for the interrupts. It also reads the condition of SID (Serial Input Data) bit on the microprocessor.



Uses of Interrupts in 8085 microprocessor

Interrupts in the 8085 microprocessor are used for various purposes, including:

1. **Real-time processing:** Interrupts allow the microprocessor to respond quickly to external events in real-time, such as user input or hardware signals. This is particularly useful in applications such as control systems and data acquisition systems where time-critical operations are required.
2. **Multi-tasking:** Interrupts enable the microprocessor to perform multiple tasks simultaneously by temporarily suspending the current task and executing the ISR for the interrupting event. This allows the microprocessor to switch between different tasks and maximize the utilization of system resources.
3. **Input/output operations:** Interrupts can be used for handling input/output operations, such as data transfer between the microprocessor and external devices. This allows the microprocessor to perform other tasks while waiting for input/output operations to complete.
4. **Error handling:** Interrupts can be used for error handling and exception handling, such as detecting and recovering from hardware or software errors.
5. **Power management:** Interrupts can be used for power management, such as putting the microprocessor into a low-power mode when it is not needed and waking it up when an interrupt occurs.

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

- In a computer, a polled interrupt is a specific type of I/O interrupt that notifies the part of the computer containing the I/O interface that a device is ready to be read or otherwise handled but does not indicate which device. The interrupt controller must poll (send a signal out to) each device to determine which one made the request.
- Polled interrupts are handled using mostly software and are therefore slower compared to vectored (hardware) interrupts. The processor responds to an interrupt by executing one general service routine for all devices. The priority of these devices is determined by the order in which the routine polls each device. Once the processor determines the source of interrupt, it branches to the service routine for that device. The typical configuration of the polled interrupt is shown in figure below.
- As shown in figure, several external devices (Device1, Device2,……., Device N) are connected to a single interrupt line (INT) of the processor. When one or more devices activate the INT line high, the processor saves the content of the PC and other registers and then branches to an address defined by the manufacturer of the processor. The user can write a program at this address in order to poll each device starting with highest priority device in order to find the source of the interrupt.
- Polled interrupts are very simple. But for a large number of devices, the time required to poll each device may exceed the time to service the device.

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.
- Here several external 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.

Polled Interrupt

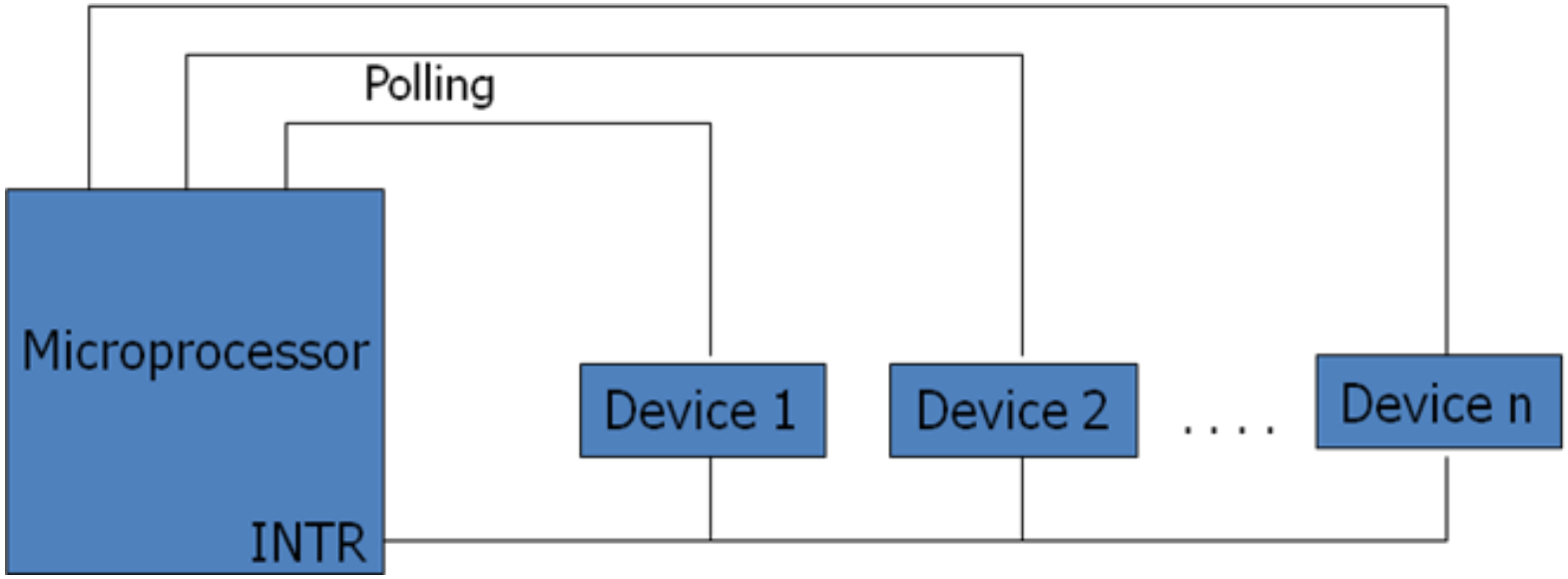


Fig: Polled Interrupt

Vectored(Daisy Chain) Interrupt

- In a computer, a vectored interrupt is an I/O interrupt that tells the part of the computer that handles I/O interrupts at the hardware level that a request for attention from an I/O device has been received and also identifies the device that sent the request.
- The device is connected in a chain as shown in figure below for setting up the priority systems. Suppose that one or more devices interrupt the processor at a time. In response, the processor saves its current status and then generates an interrupt acknowledge ($\overline{\text{INTA}}$) signal to the highest priority device, which is device1, in this case.
- If this device has generated the interrupt, it will accept the $\overline{\text{INTA}}$ signal from the processor; otherwise, it will pass $\overline{\text{INTA}}$ on to the next device until $\overline{\text{INTA}}$ is accepted by the interrupting device.

Vectored(Daisy Chain) 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 $\overline{\text{INTA}}$ signal to the highest priority device. If this device has generated the interrupt, it will accept the INTA; otherwise it will push $\overline{\text{INTA}}$ to the next priority device until the $\overline{\text{INTA}}$ is accepted by the interrupting device.
- When $\overline{\text{INTA}}$ 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.

Vectored(Daisy Chain) Interrupt

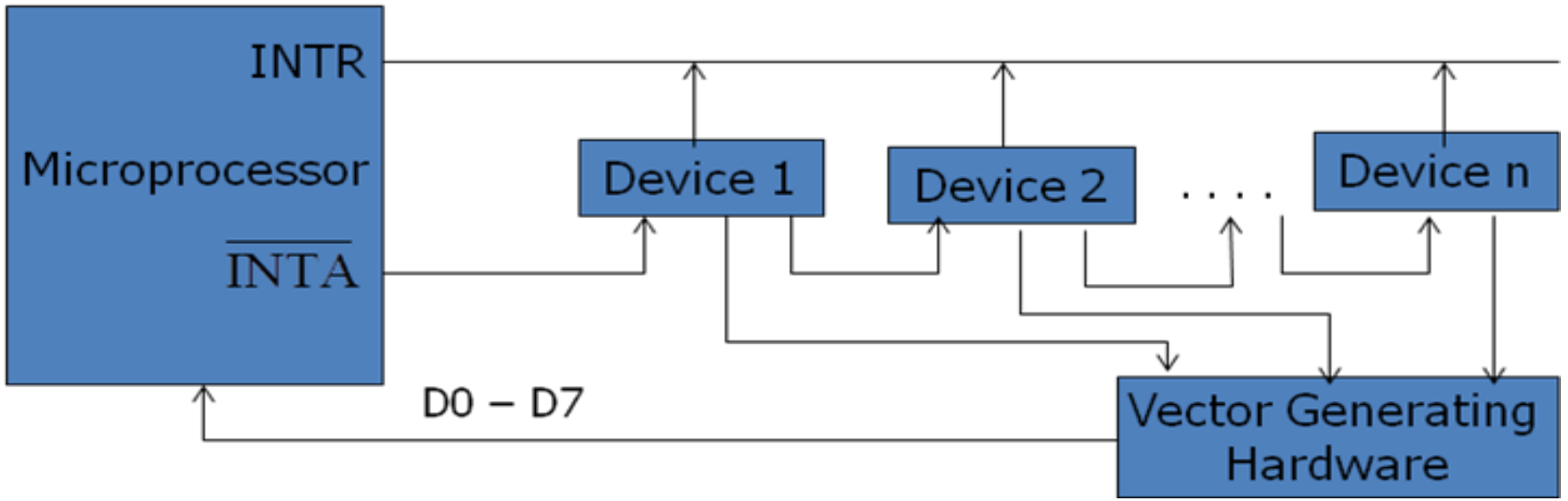


Fig: Vectored (Daisy Chain) Interrupt

Q. If INTR is non vectored interrupt, then why it is used as vectored in Daisey-chaining interrupt handling mechanism?

Answer:

While the INTR itself is technically a non-vectored interrupt because it doesn't directly specify an interrupt vector (memory address) like some other interrupt systems, it can be used in a vectored manner when combined with external hardware, such as a priority encoder or daisy-chaining logic or The 8259 Interrupt Controller.

Summary of 8085 Interrupts

Interrupt	Vector address	Priority	Type
TRAP	0024H	1 (highest priority)	Hardware interrupt Vectored interrupt Non-maskable interrupt
RST 7.5	003CH	2	Hardware interrupt Vectored interrupt Maskable interrupt
RST 6.5	0034H	3	Hardware interrupt Vectored interrupt Maskable interrupt
RST 5.5	002CH	4	Hardware interrupt Vectored interrupt Maskable interrupt
INTR	-	5(lowest priority)	Hardware interrupt Non - Vectored interrupt Maskable interrupt
RST instruction	RST 0 – 0000H RST 1 – 0008H RST 2 – 0010H RST 3 – 0018H RST 4 – 0020H RST 5 - 0028H RST 6 – 0030H RST 7 – 0038H	-	Software interrupt Vectored interrupt Maskable interrupt

The 8259 Interrupt Controller/ 8259 Programmable Interrupt Controller

- It is also known as a **priority interrupt controller** and was designed by Intel to increase the interrupt handling ability of the microprocessor.
- The 8259A is a programmable interrupt controller designed to work with Intel microprocessors 8085, 8086, and 8088.
- An **8259 PIC** never services an interrupt; it simply forwards the interrupt to the processor for the execution of interrupt service routine.
- The advanced version of **8259 PIC is 8259A PIC** which can be used with Intel's 8086/88 16-bit Microprocessor.

Features of 8259

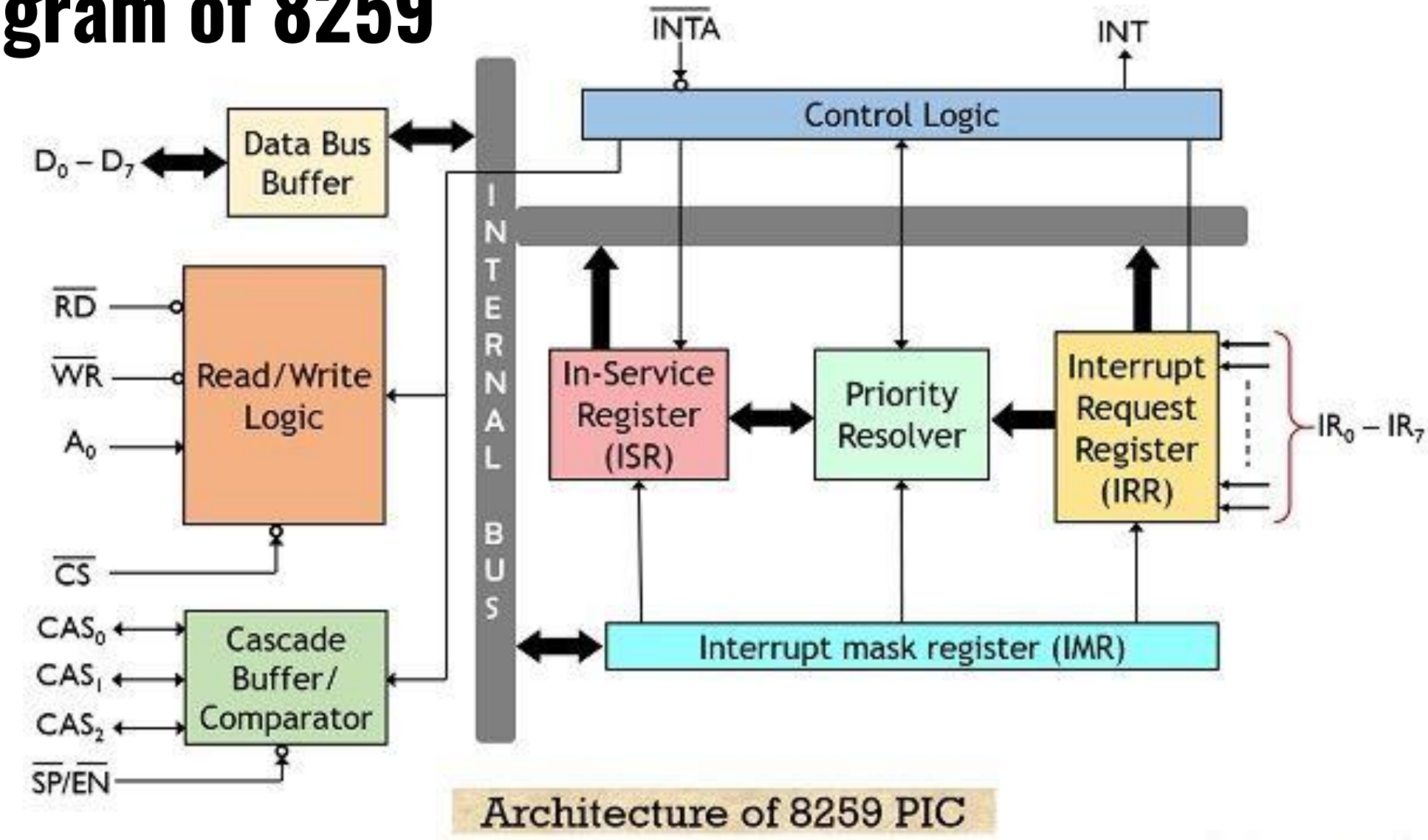
- The 8259 programmable interrupt controller has 8 interrupt pins thus can handle 8 interrupt inputs.
- The priority of interrupts in 8259 can be programmed. The priority of interrupts is decided by the different operating modes.
- We know that a single 8259 can handle 8 interrupt inputs but by cascading multiple 8259, it can handle maximal 64 interrupt inputs.
- 8259 allows individual masking of each generated interrupt using interrupt mask register.
- 8259 is programmed in a way that it can handle either edge-triggered or level-triggered interrupt request at a time.
- If multiple interrupts are generated, then 8259 holds the status of interrupts that are masked, in-service and pending.
- It reduces the software and real-time overhead generated due to handling multilevel priority interrupts.

Features of 8259

- The 8259A is a programmable interrupt controller designed to work with Intel micro-processors 8085, 8086, and 8088The 8259A interrupt controller can
  - manage eight interrupts according to the instructions written into its control registers. This is equivalent to providing eight interrupt pins on the processor in place of one INTR (8085) pin.
  - vector an interrupt request anywhere in the memory map. However, all eight interrupts are spaced at the interval of either four or eight locations. This eliminates the major drawback of the 8085 interrupts in which all interrupts are vectored to memory locations on page OOH.
  - resolve eight levels of interrupt priorities in a variety of modes, such as fully nested mode, automatic rotation mode, and specific rotation mode (to be explained later).
  - mask each interrupt request individually.
  - read the status of pending interrupts, in-service interrupts, and masked interrupts.
  - be set up to accept either the level-triggered or the edge-triggered interrupt request.
  - be expanded to 64 priority levels by cascading additional 8259As.
  - be set up to work with either the 8085 microprocessor mode or the 8086/8088 microprocessor mode.

Block Diagram of 8259

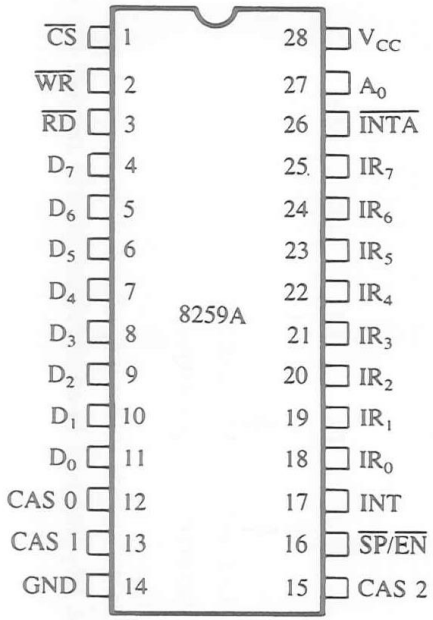
- The figure shows the architectural representation of 8259 programmable interrupt controller:



Architecture of 8259 PIC

Block Diagram of 8259A

Pin Configuration



Pin Names

D_7–D_0	Data Bus (Bidirectional)
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A_0	Command Select Address
$\overline{CS}$	Chip Select
CAS_2–CAS_0	Cascade Lines
$\overline{SP}/\overline{EN}$	Slave Program/Enable Buffer
INT	Interrupt Output
$\overline{INTA}$	Interrupt Acknowledge Input
IR_0–IR_7	Interrupt Request Inputs

Block Diagram

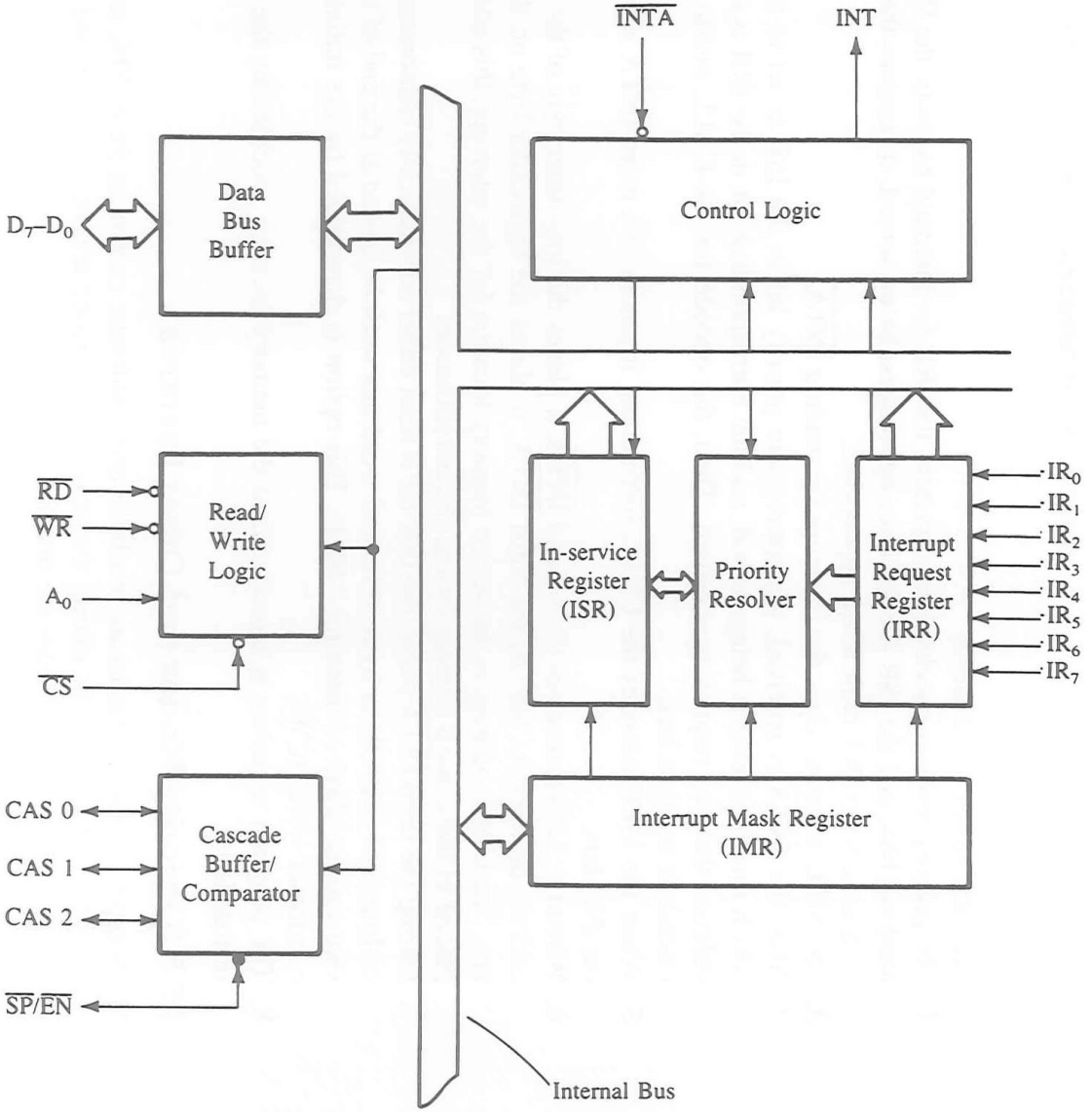


FIGURE 15.29
The 8259A Block Diagram

SOURCE: Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), pp. 3–171, 3–172.

Block Diagram of the 8259A

- Figure in previous slide shows the internal block diagram of the 8259A. It includes eight blocks: **control logic, Read/Write logic, data bus buffer, three registers (IRR, ISR, and IMR), priority resolver, and cascade buffer**.
- This diagram shows all the elements of a programmable device, plus additional blocks.
- The functions of some of these blocks is given below.

1. **READ/WRITE LOGIC:** This is a typical Read/Write control logic. When the address line A_0 is at logic 0, the controller is selected to write a command or read a status. The Chip Select logic and A_0 determine the port address of the controller.

2. **CONTROL LOGIC:** This block has two pins: INT (Interrupt) as an output, and $\overline{\text{INTA}}$ (Interrupt Acknowledge) as an input. The INT is connected to the interrupt pin of the MPU. Whenever a valid interrupt is asserted, this signal goes high. The $\overline{\text{INTA}}$ is the Interrupt Acknowledge signal from the MPU.

Block Diagram of the 8259A

3. **INTERRUPT REGISTERS AND PRIORITY RESOLVER:** The Interrupt Request Register (IRR) has eight input lines (IR0-IR7) for interrupts. When these lines go high, the requests are stored in the register. The In-Service Register (ISR) stores all the levels that are currently being serviced, and the Interrupt Mask Register (IMR) stores the masking bits of the interrupt lines to be masked. The Priority Resolver (PR) examines these three registers and determines whether INT should be sent to the MPU.
4. **CASCADE BUFFER/COMPARATOR:** This block is used to expand the number of interrupt levels by cascading two or more 8259As. To simplify the discussion, this block will not be mentioned again.

Block Diagram of 8259

- It has an 8-bit of data bus. As we have already discussed that 8259 never services the interrupt, it simply forwards the interrupts to the microprocessor.
- Thus, the above architecture has different units that combinedly functions to increase the interrupts handled by the processor.
- Let us understand the operation performed by each unit in detail:

1) **Data Bus Buffer:** 8259 has tri-stated bidirectional 8-bit data bus buffer (i.e., D0 to D7) that interfaces with the internal bus of the processor. The 8085 microprocessor sends/ receives, control or status words to/ from the 8259 using data bus buffer.
2) **Read/ Write Logic:**
   - This unit is responsible for controlling the internal read-write operations of the system. It holds initialization command word register and operation command word register inside which various control formats exist that are needed for the device operation.
   - $\overline{RD}$, $\overline{WR}$, A_0 and $\overline{CS}$ are the pins that are associated with this unit. Basically, these pins are used by the processor for read and write operations.
   - A low signal at $\overline{CS}$ i.e., chip select shows that now the communication has been set up between the processor and 8259.

Block Diagram of 8259

3) Control Logic:

- This unit is the heart of the architecture of 8259. It controls the overall operation of the system by sending the INTR signal to the processor whenever an interrupt request is generated.
- Also, it receives $\overline{\text{INTA}}$ signal by the processor when microprocessor demands for the address of the interrupt service routine. The control logic is responsible for sending the address of the desired interrupt service routine through the data bus.

4) Interrupt request register (IRR): This unit stores the interrupt requests generated by the peripheral devices. We know that 8259 has 8 interrupt request pins (i.e., IR0 to IR7). So, the unit can store 8 interrupt requests that are requesting the service from the processor.

5) Priority Resolver:

- This logic unit decides that among all the interrupt request present in the IRR which holds the highest priority and needs to be executed first.
- Suppose at the time of servicing an interrupt, another incoming interrupt request gets generated then that request will be ignored as the one in-service is holding the highest priority.
- But in case the incoming request has greater priority than the one which is being in current execution then that respective bit will be set in ISR and INTR signal is sent to the microprocessor. This simply means that only the interrupt holding the highest priority will be forwarded by the 8259 to the processor.

Block Diagram of 8259

6) **In-service register:**
   - ❑ Here the name of the unit is itself indicating the operation performed by it. This register unit stores the interrupts which are currently being executed by the processor.
   - ❑ The priority resolver sets each bit of ISR and after getting interrupt word command by the processor, the bits get reset. As the processor holds the ability to directly read the status of in-service register.
7) **Interrupt mask register:** This register unit holds the masking bit of those interrupts which are to be masked. Through operation command word (OCW) the processor sends the required information and programs the interrupt mask register.
8) **Cascade buffer/comparator:** As we have already discussed that by cascading multiple 8259, the number of interrupts handled by 8259 can be expanded up to 64. The unit allows the comparison of IDs of different 8259s cascaded together.

8259A Interrupt Operation

- To implement interrupts, the Interrupt Enable flip-flop in the microprocessor should be enabled by writing the EI instruction, and the 8259A should be initialized by writing control words in the control register.
- The 8259A requires two types of control words: Initialization Command Words (ICWs) and Operational Command Words (OCWs).
- The ICWs are used to set up the proper conditions and specify RST vector addresses.
- The OCWs are used to perform functions such as masking interrupts, setting up status-read operations, etc. After the 8259A is initialized, the following sequence of events occurs when one or more interrupt request lines go high:

8259A Interrupt Operation

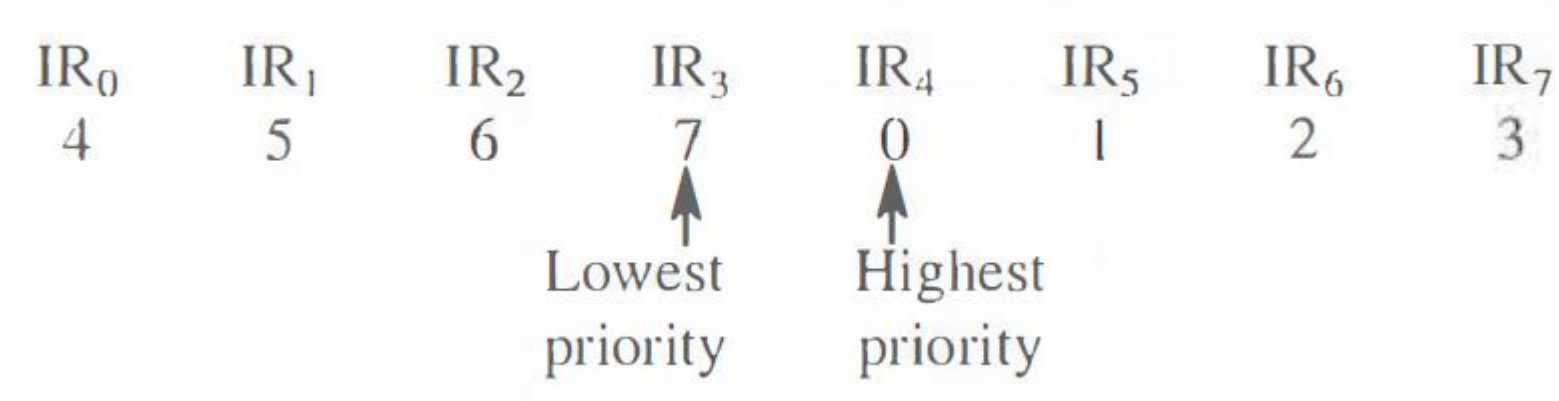
1. The IRR stores the requests.
2. The priority resolver checks three registers: the IRR for interrupt requests, the Ilv1R for masking bits, and the ISR for the interrupt request being served. It resolves the priority and sets the INT high when appropriate.
3. The MPU acknowledges the interrupt by sending $\overline{\text{INTA}}$.
4. After the $\overline{\text{INTA}}$ is received, the appropriate priority bit in the ISR is set to indicate which interrupt level is being served, and the corresponding bit in the IRR is reset to indicate that the request is accepted. Then, the opcode for the CALL instruction is placed on the data bus.
5. When the MPU decodes the CALL instruction, it places two more $\overline{\text{INTA}}$ signals on the data bus.
6. When the 8259A receives the second $\overline{\text{INTA}}$, it places the low-order byte of the CALL address on the data bus. At the third $\overline{\text{INTA}}$, it places the high-order byte on the data bus. The CALL address is the vector memory location for the interrupt; this address is placed in the control register during the initialization.
7. During the third $\overline{\text{INTA}}$ pulse, the ISR bit is reset either automatically (Automatic-End of-Interrupt — AEOI) or by a command word that must be issued at the end of the service routine (End-of-Interrupt-EOI). This option is determined by the initialization command word (ICW).
8. The program sequence is transferred to the memory location specified by the CALL instruction.

Priority Modes

- Many types of priority modes are available under software control in the 8259A, and they can be changed dynamically during the program by writing appropriate command words.

1. Fully Nested Mode:

- This is a general-purpose mode in which all IRs (Interrupt Requests) are arranged from highest to lowest, with IR0 as the highest and IR7 as the lowest.
- In addition, any IR can be assigned the highest priority in this mode; the priority sequence will then begin at that IR. In the example below, IR4 has the highest priority, and IR3 has the lowest priority:



Priority Modes

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
1	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, thus fixing all other priorities.

End of Interrupt

After the completion of an interrupt service, the corresponding ISR bit needs to be reset to update the information in the ISR. This is called the End-of-Interrupt (EOI) command. It can be issued in three formats:

1. **Nonspecific EOI Command** When this command is sent to the 8259A, it resets the 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 $\overline{\text{INTA}}$, the ISR bit is reset. The major drawback with this mode is that the ISR does not have information on which IR is being serviced. Thus, any IR can interrupt the service routine, irrespective of its priority, if the Interrupt Enable flip-flop is set.

Additional Features

- The 8259A is a complex device with various modes of operation. These modes are listed below for reference;
  - **Interrupt Triggering:** The 8259A can accept an interrupt request with either the edge triggered mode or the level-triggered mode. The mode is determined by the initialization instructions.
  - **Interrupt Status:** The status of the three interrupt registers (IRR, ISR, and IMR) can be read, and this status information can be used to make the interrupt process versatile.
  - **Poll Method:** The 8259A can be set up to function in a polled environment. The MPU polls the 8259A rather than each peripheral.

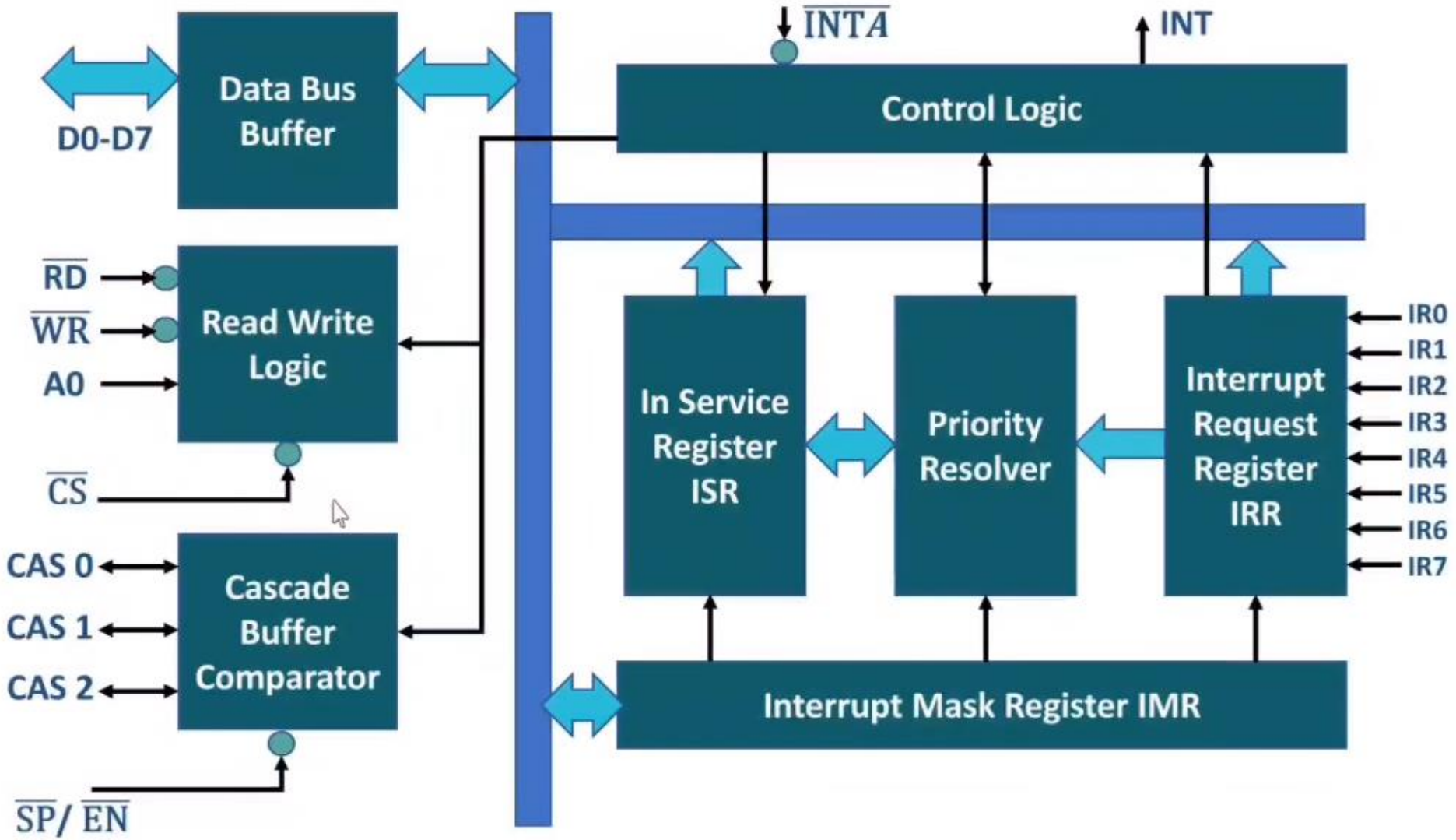
8259 Programmable Interrupt Controller

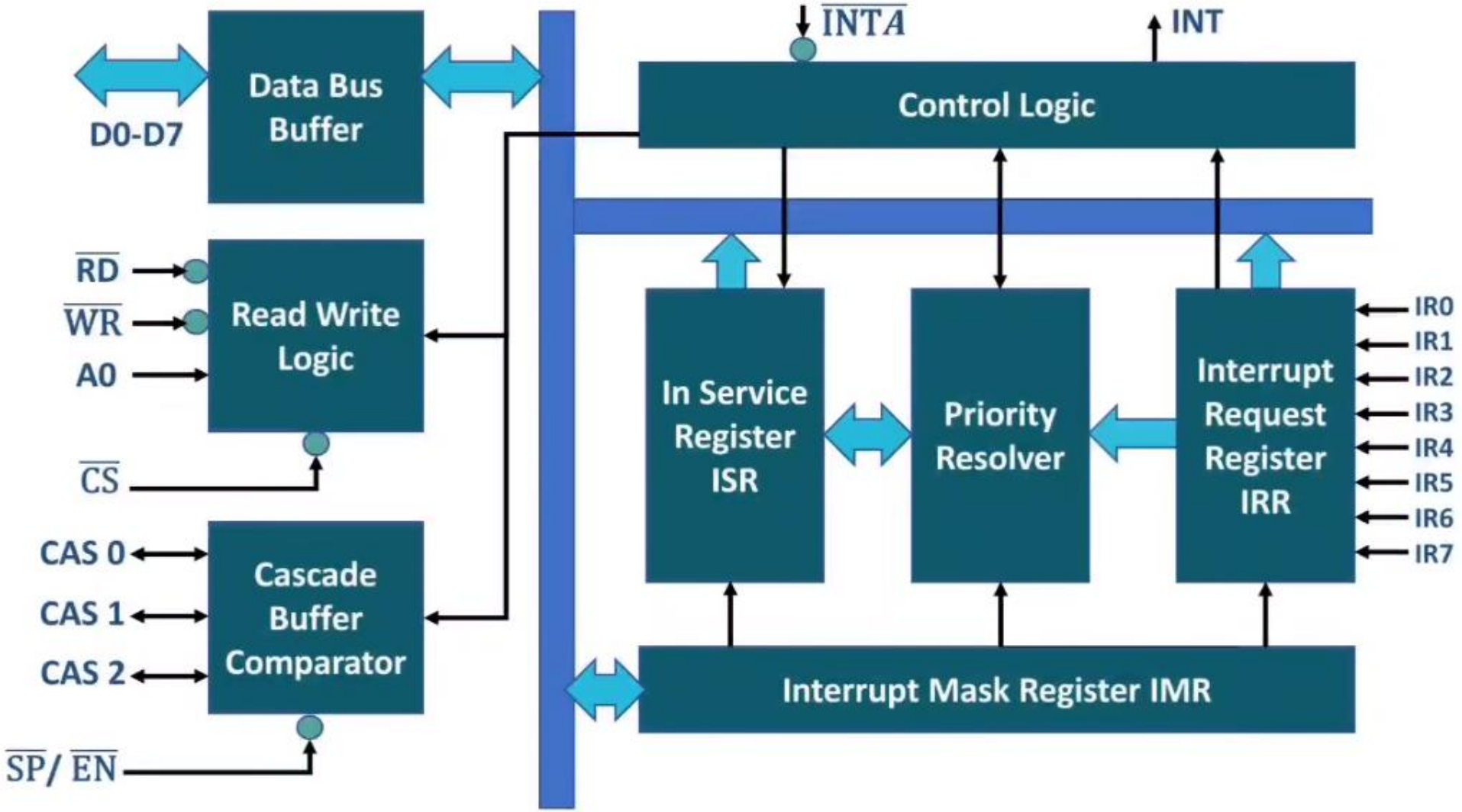
❖ Features of 8259 Programmable Interrupt Controller

- ❑ 8259 is designed to work with various microprocessors like 8085, 8086 etc.
- ❑ 8259 is designed to increase capacity of interrupts.
- ❑ 8259 can handle 8 interrupts with single IC.
- ❑ A cascade connection of 8259 can handle 64 interrupts.
  - ▪ One Master 8259 can works with eight Slave 8259, so total capacity will be 64.
- ❑ 8259 has flexible interrupt priority structure.
- ❑ Using 8259, we can mask interrupt as well.
- ❑ The vector address of 8259 is programmable.
- ❑ Status of interrupt can be observed by microprocessor :
  - ▪ Pending status
  - ▪ In service status
  - ▪ Masked status

8259 Programmable Interrupt Controller

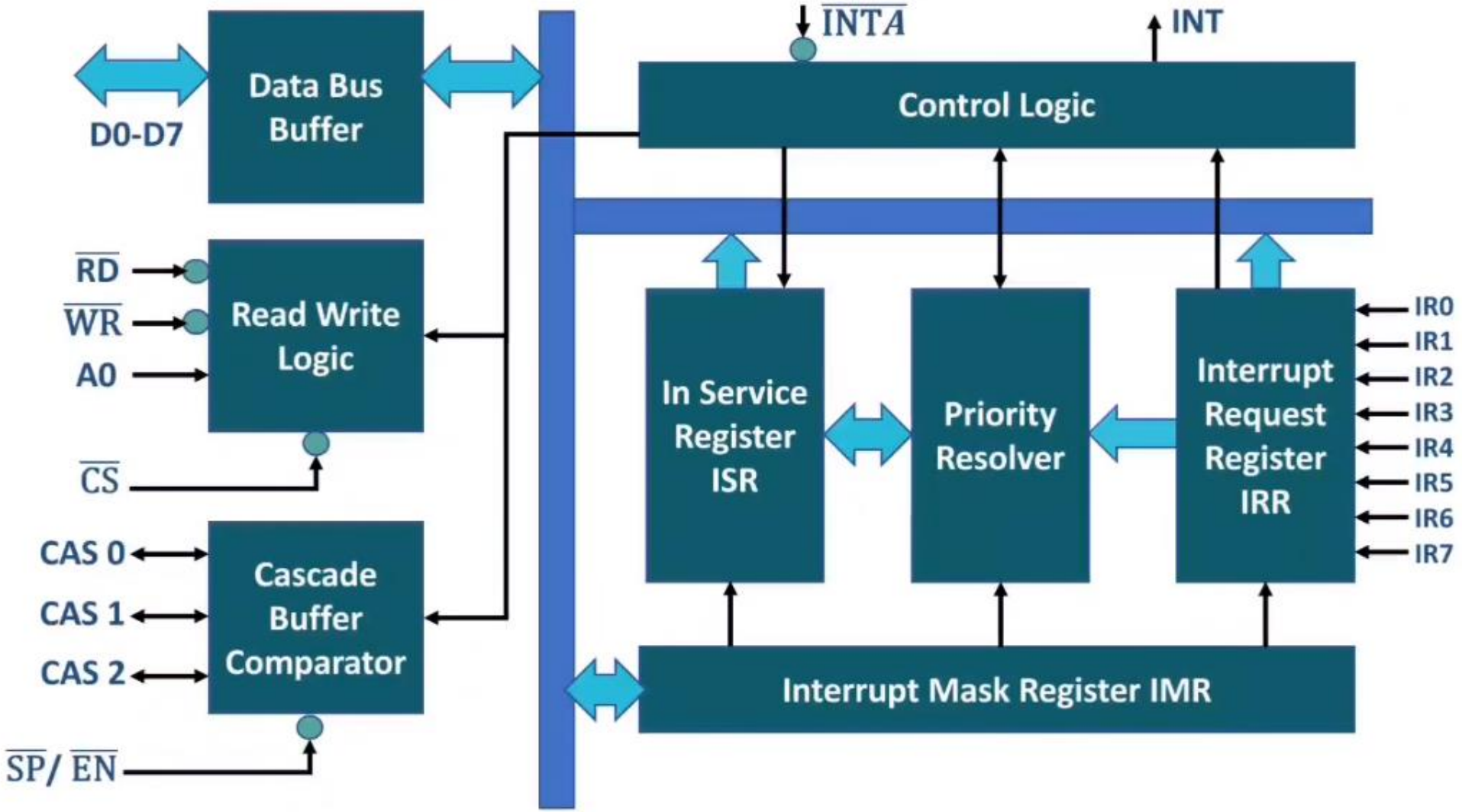
❖ Block Diagram / Architecture of 8259





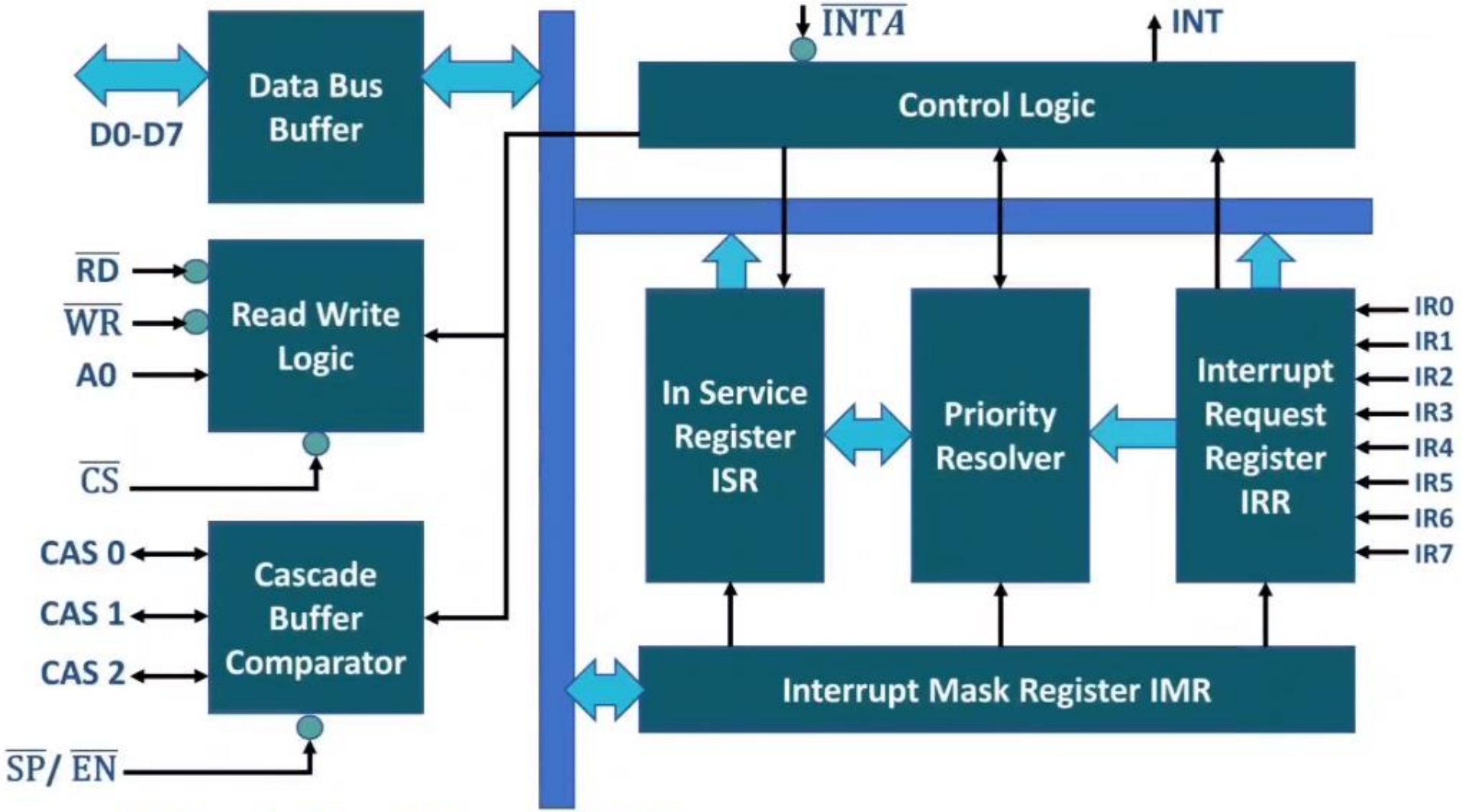
❖ Interrupt Request Register - IRR

- ❑ Here, we have 8 interrupt lines IR7 – IR0.
- ❑ IRR is Eight bits register, each bit stores individual interrupt.
- ❑ When interrupt occurs on any lines, corresponding bit will get set to One.



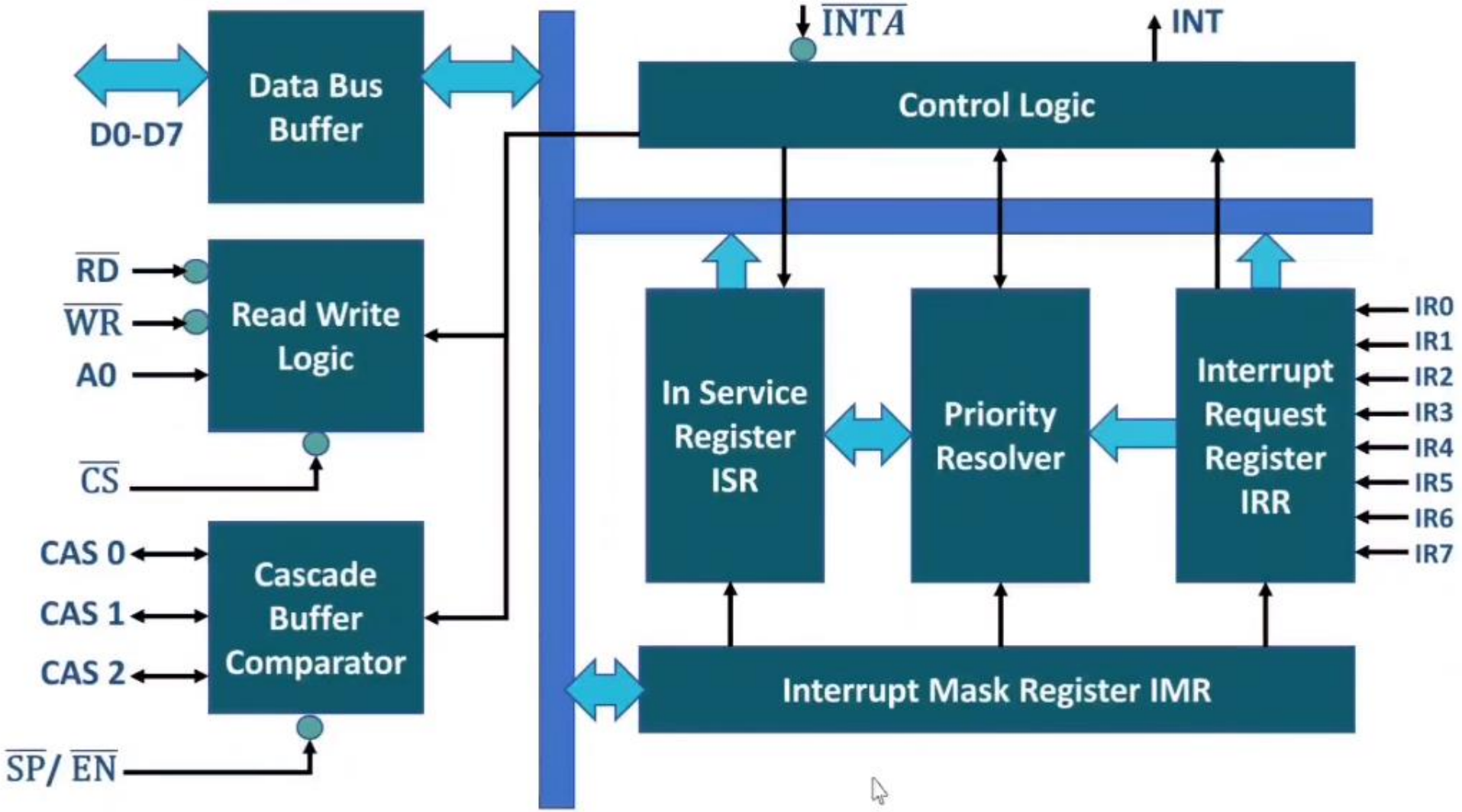
❖ In Service Register - ISR

- ❑ It is 8 bits register.
- ❑ It stores the data of currently served interrupt.



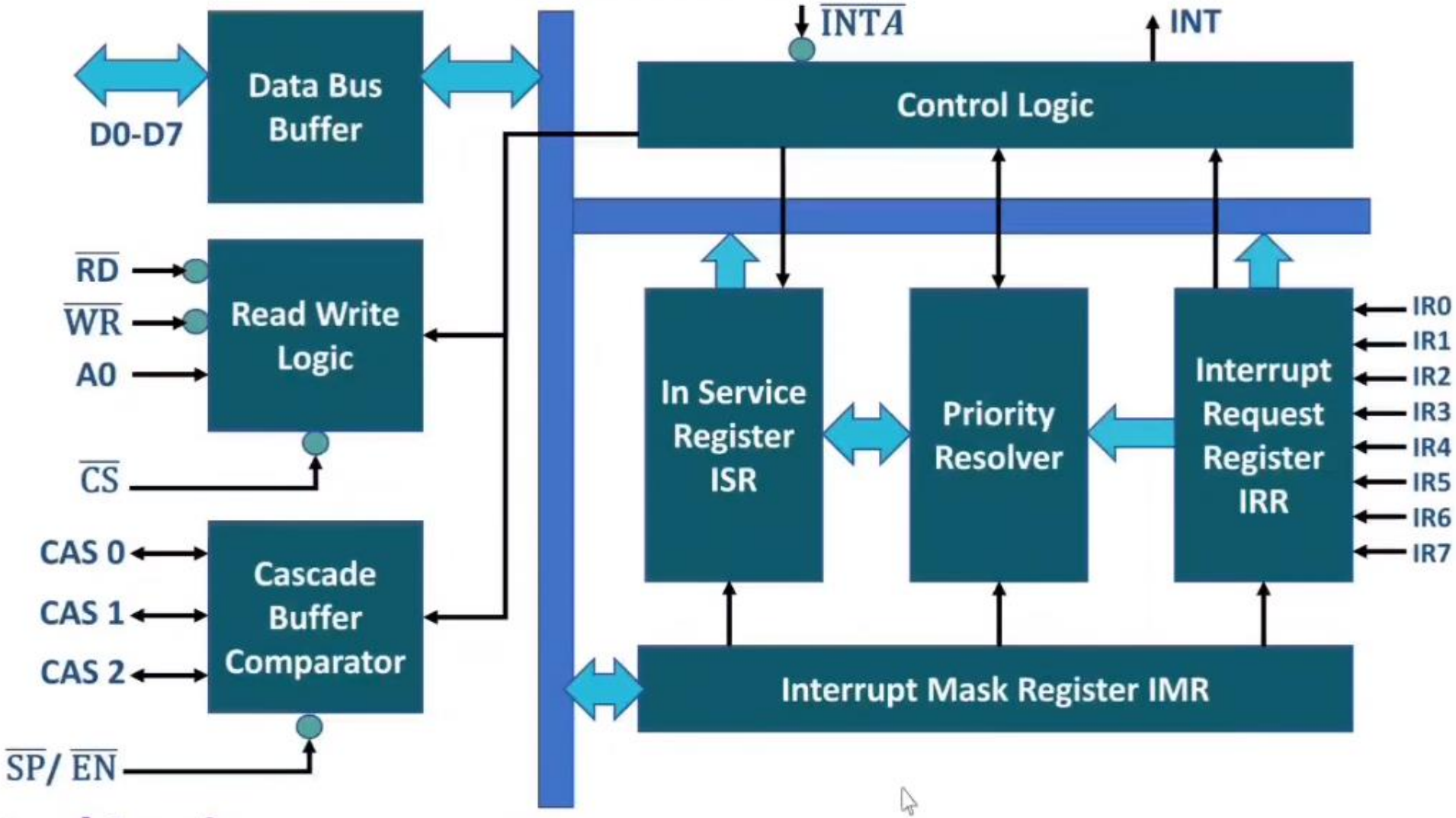
❖ Interrupt Mask Register - IMR

- ❑ It is 8 bits register.
- ❑ It stores the masking pattern of 8259.
- ❑ Each bits holds masking of individual interrupt.



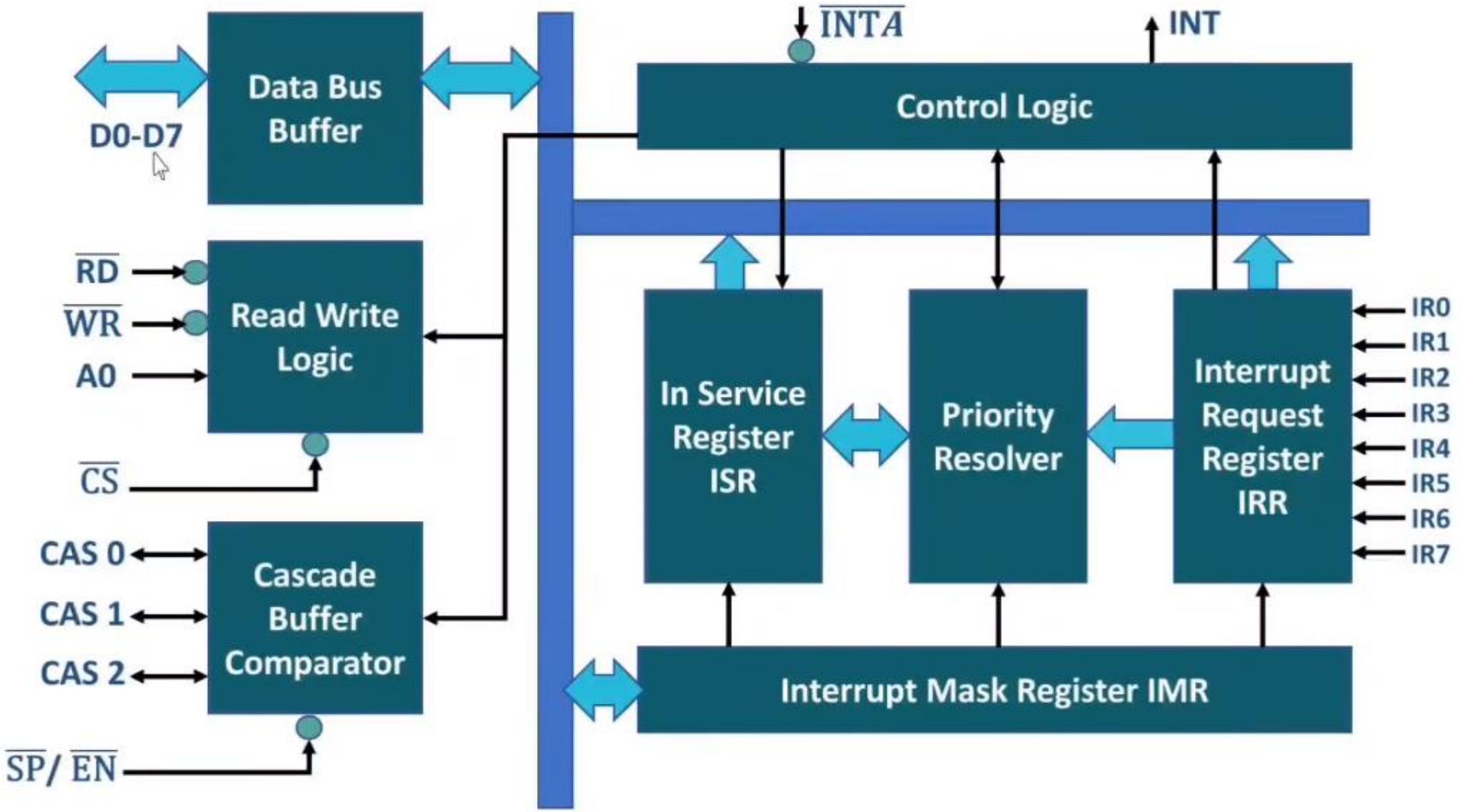
❖ Priority Resolver

- ❑ It examines IRR, ISR and IMR. Based on that determines which interrupt has maximum priority and should be sent to microprocessor.



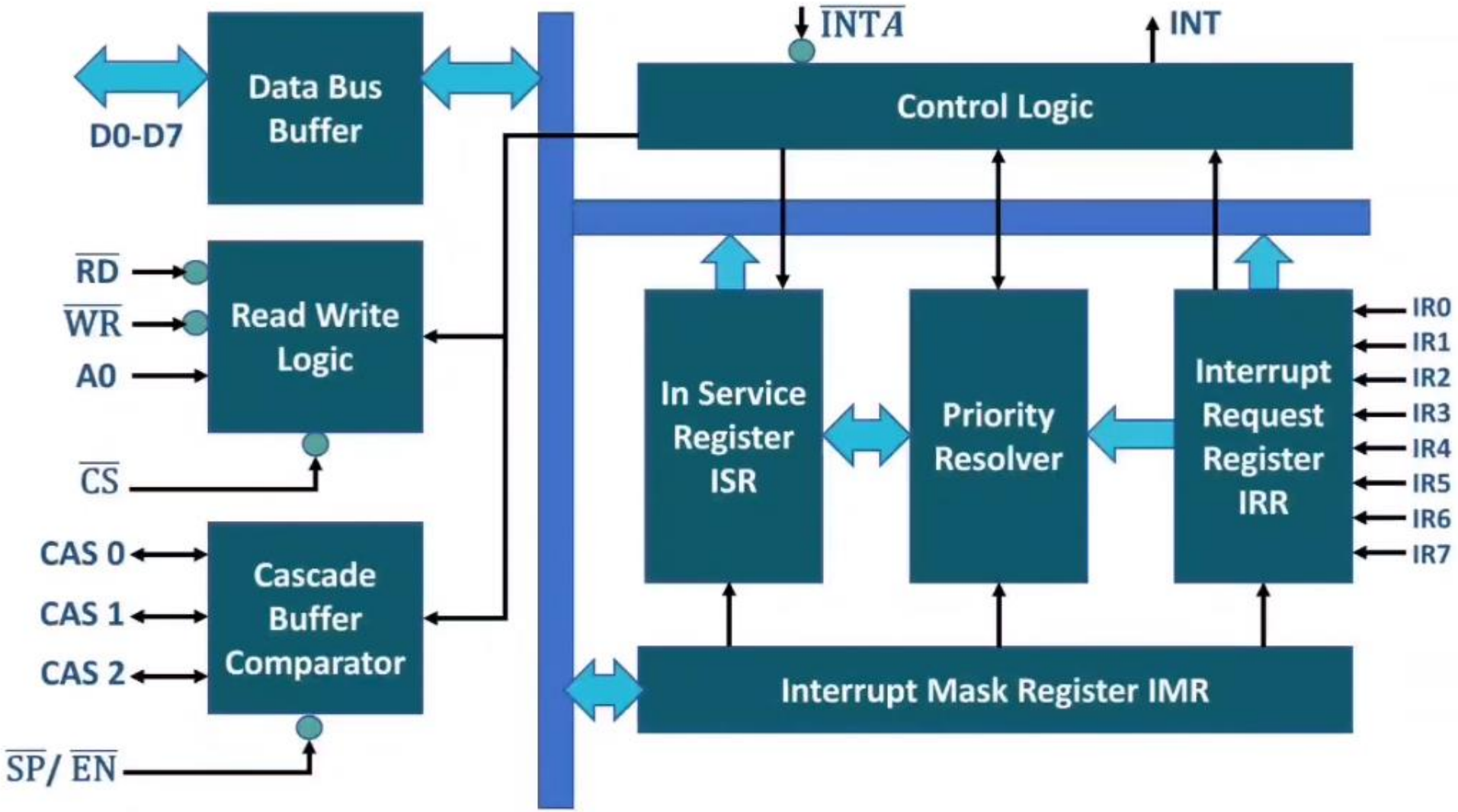
❖ Control Logic

- ❑ It has INT pin connected with INTR of microprocessor to send interrupt request.
- ❑ It has $\overline{INTA}$ pin connected with $\overline{INTA}$ of microprocessor to receive acknowledgment.
- ❑ It is also used to control remaining blocks.



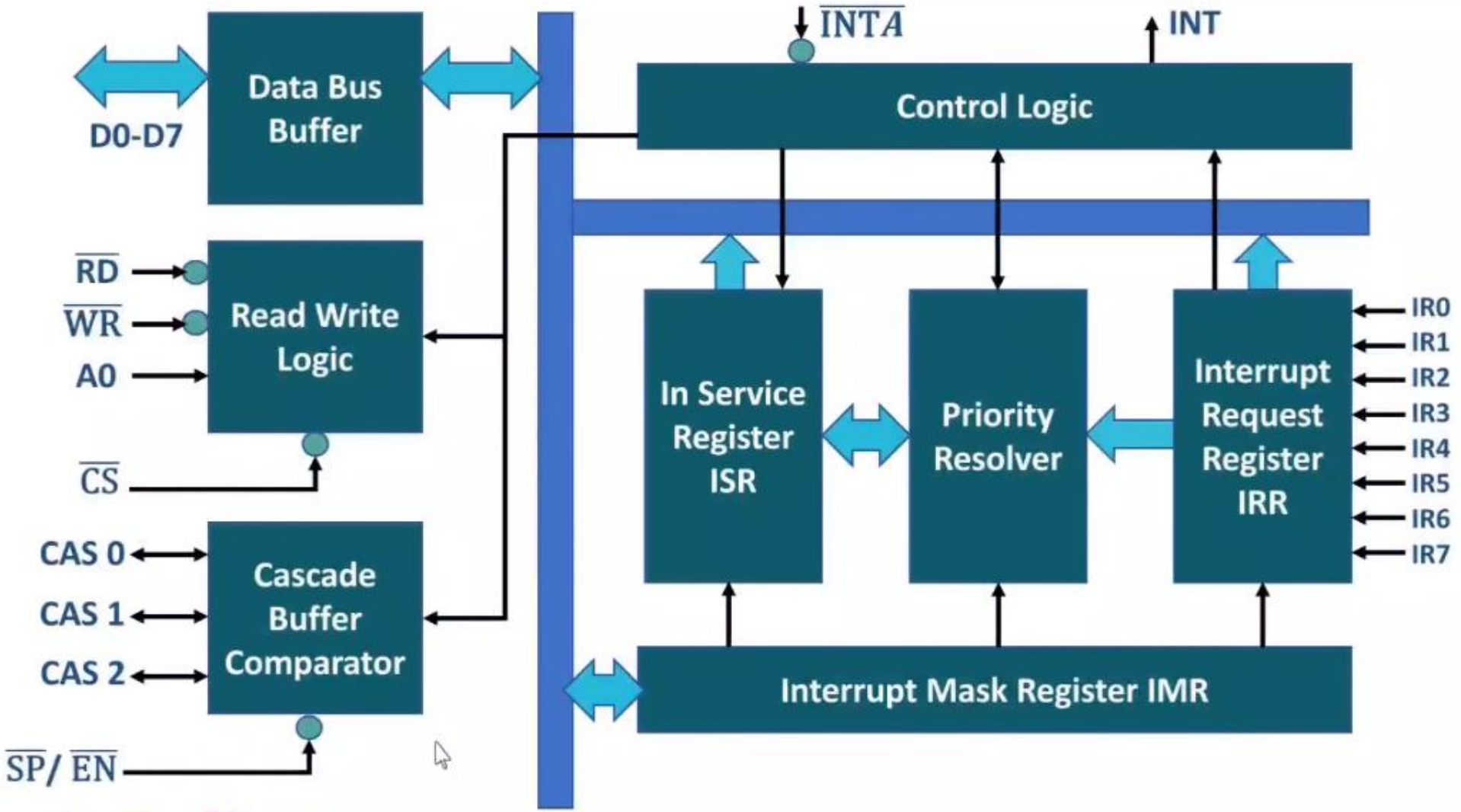
❖ Data Bus Buffer

- ❑ It has bidirectional data bus D0 – D7.
- ❑ D0 – D7 is interfaced with system data bus of Microprocessor.



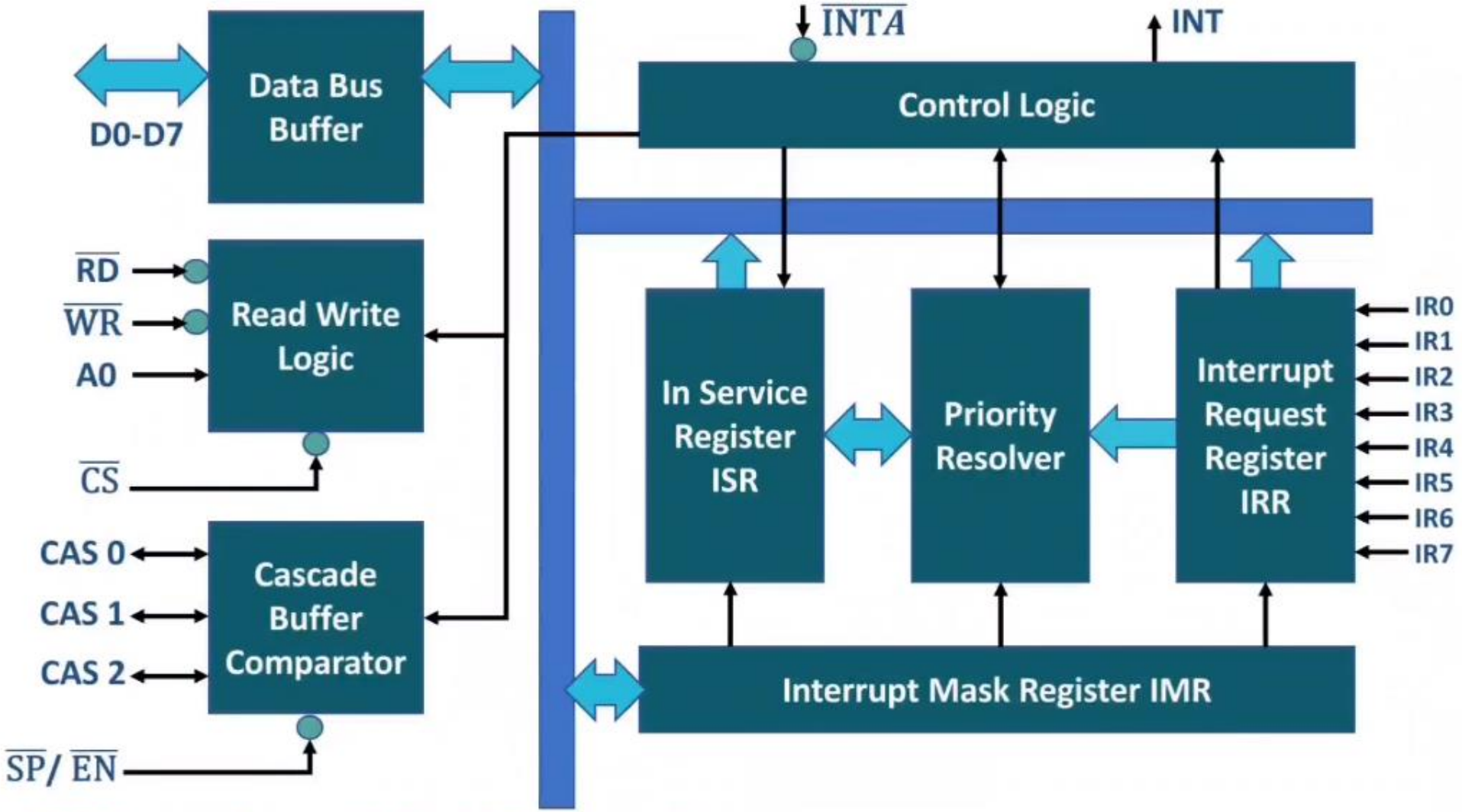
❖ Read Write Logic

- ❑ It is used to take read, write, A0 and Chip select.
- ❑ It also holds Initialization Command Words (ICW) and Operational Command Words (OCW).

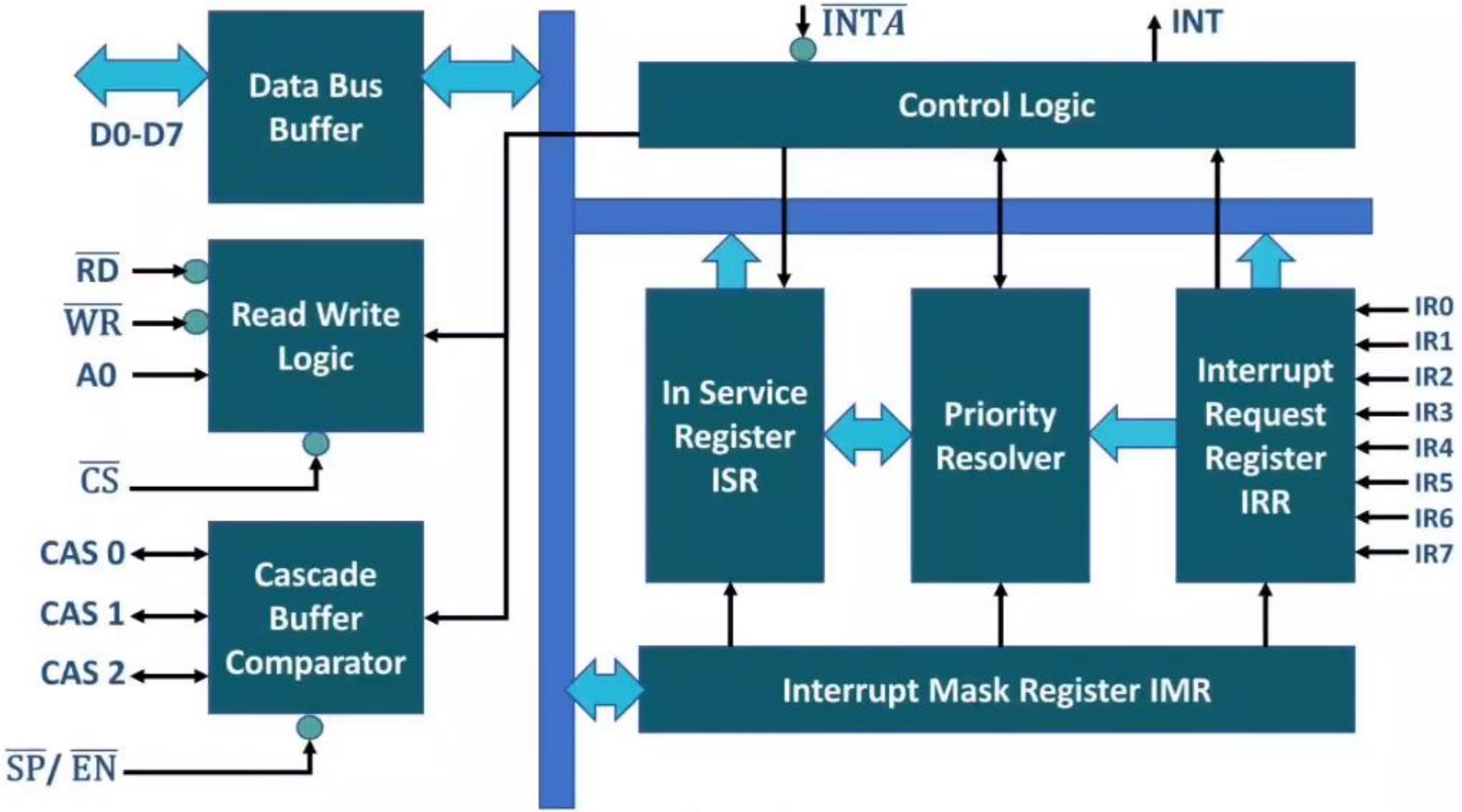


❖ Cascade Buffer Comparator

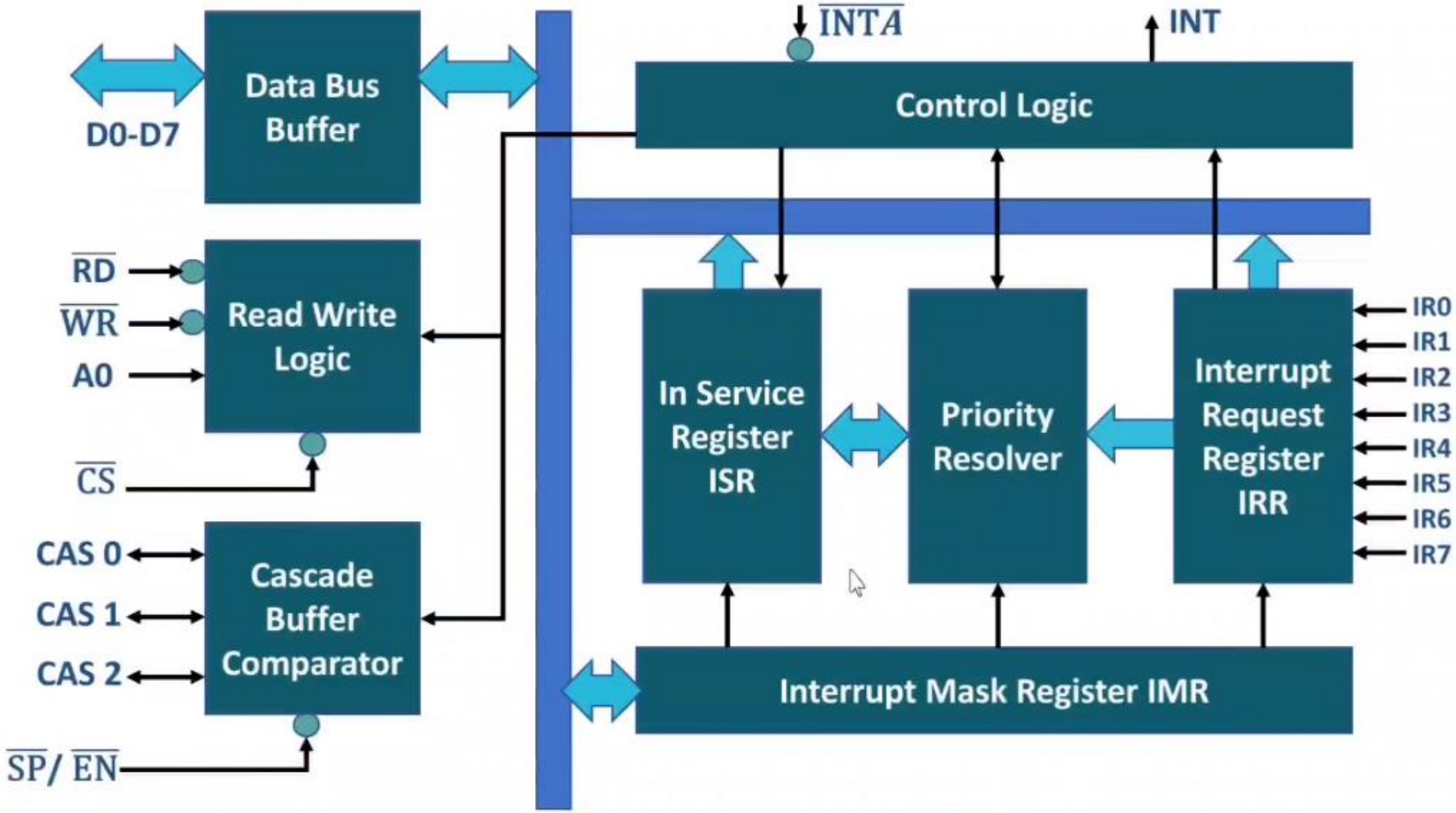
- ❑ It is used in cascade mode operation.
- ❑ It is used for Master Slave Interrupt control from multiple 8259.
- ❑ $\overline{SP}/\overline{EN}$ – It is Slave Program/ Master Enable line. In buffer Mode, it function as enable line and In non buffer mode it functions as SP enable line.



1. INTR of 8085 must be enabled with EI Instruction.
2. 8259 is initialized by necessary commands. [ICW]
3. Once 8259 is initialized, if multiple interrupts comes then corresponding bit of IRR is set to one.
4. Priority resolver checks IRR, IMR & ISR. Based on that, it will decide highest priority and then it gives signal to Microprocessor 8085 at INTR.



5. The Microprocessor completes current instruction, after that it give acknowledgement to 8259 on $\overline{INTA}$.
6. On receiving $\overline{INTA}$ from Microprocessor, ISR will set corresponding bit to one in ISR to indicate service to this interrupt is started and the bit in IRR is reset to indicate request is accepted. Now 8259 can give opcode of CALL instruction to Microprocessor.



7. The microprocessor decodes the CALL instruction and sends two more $\overline{INTA}$ to 8259.
8. In response to $\overline{INTA}$ signals, 8059 sends address of interrupt service routine. So it completes three bytes CALL instruction.
9. Now Microprocessor perform Interrupt Service Routine by pushing content of PC on stack.
10. At the end of interrupt, Microprocessor will send EOI command to 8259, that makes corresponding bit 0 in ISR of 8259.

The I/O interfacing numerical problems are not on your syllabus, so they are not important. ***NO NEED TO STUDY AFTER THIS SLIDE.***

I/O Interfacing

- In a microprocessor system, there are two methods of interfacing input/output (I/O) devices: **memory-mapped I/O** and **I/O mapped I/O**.
- In **memory-mapped I/O**, input/output devices are mapped to the memory address space of the microprocessor. This means that the I/O devices are treated like memory locations and can be accessed using the same read and write instructions as memory. In other words, the same bus and control signals used for memory access are used for I/O access as well.
- On the other hand, in **I/O mapped I/O**, input/output devices are mapped to a separate I/O address space that is different from the memory address space. The microprocessor uses special instructions to access the I/O devices using specific I/O address signals, which are separate from the memory address signals.

I/O Interfacing

- In the case of the 8085 microprocessor, it uses memory-mapped I/O for accessing input/output devices. The input/output devices are mapped to specific memory locations in the address space, and the microprocessor can access these devices using the same instructions it uses for memory access.
- For example, if a device is connected to the memory address 0x8000, the microprocessor can read from or write to the device by using the same instructions it uses to access memory locations, such as:
  - MOV A, M ; Load accumulator with data from memory location 0x8000
  - MOV M, A ; Write the contents of the accumulator to memory location 0x8000
- This method of I/O access is simpler and more efficient in terms of hardware design, as it does not require separate I/O address signals. However, it limits the number of memory locations available for use by the microprocessor.

I/O-Mapped I/O Interfacing Problems

Example 1. Interface an 8-bit DIP switch with the 8085 such that the address assigned to the DIP switch is FOH.

Solution:

The instruction **IN FOH** is used to get the data from the DIP switch and store it in the accumulator. The steps involved in the execution of the instruction IN FOH are as follows.

i. The address in the IN instruction, which is FOH in this case, is placed in the address lines A0-A7 and a copy of it in the address lines A8-A15.

ii. The $\overline{IOR}$ signal is activated (i.e., $\overline{IOR} = 0$), which makes the selected input device to place its data in the data bus.

iii. The data in the data bus is read and stored in the accumulator.

Figure 6.11 shows the interfacing of a DIP switch so as to assign the address FOH to it.

I/O-Mapped I/O Interfacing Problems

Example 1. Interface an 8-bit DIP switch with the 8085 such that the address assigned to the DIP switch is FOH.

Solution:

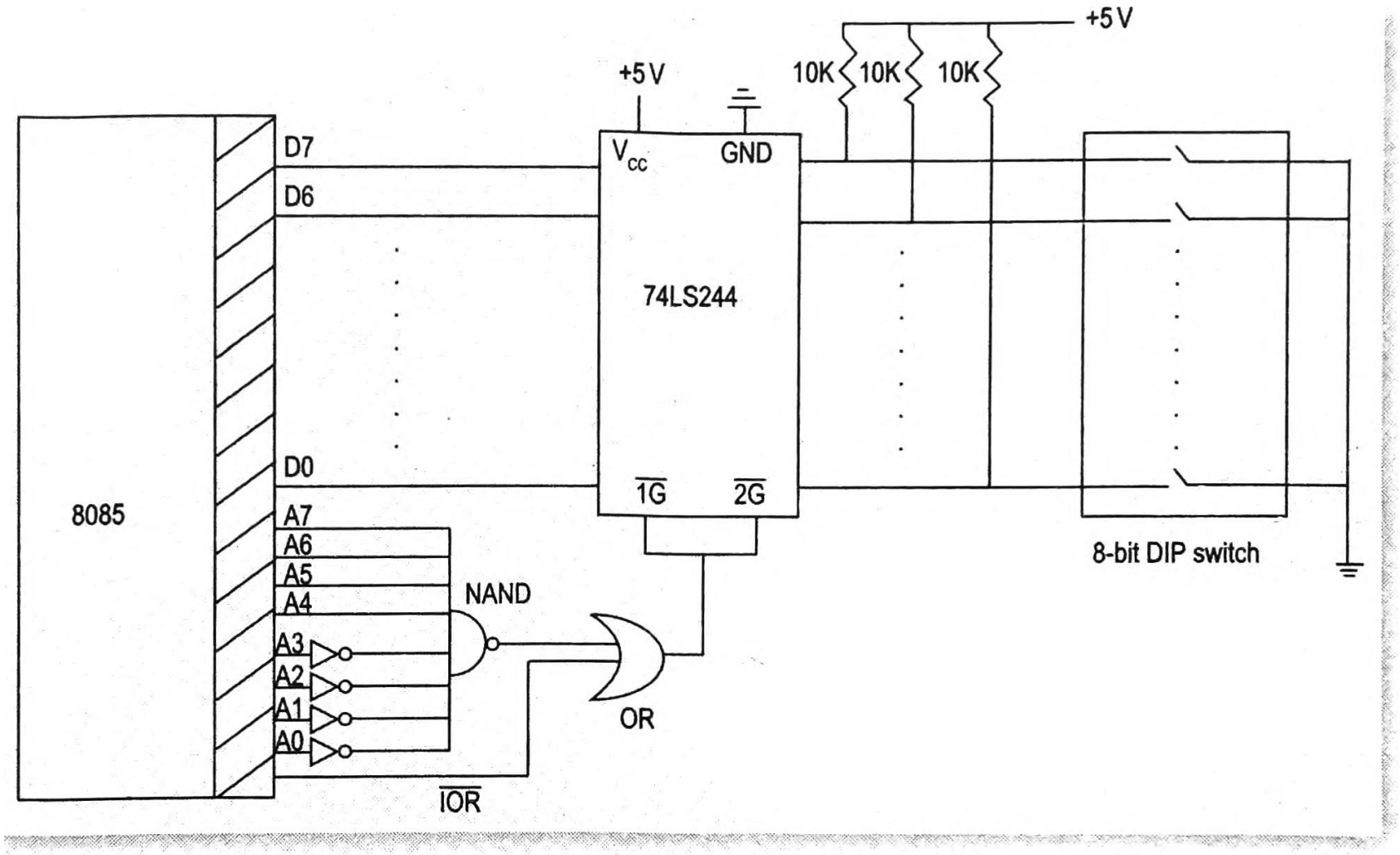


Fig. 6.11 Interfacing 8-bit DIP switch with the 8085 (Example 6.7)

I/O-Mapped I/O Interfacing Problems

Example 1. Interface an 8-bit DIP switch with the 8085 such that the address assigned to the DIP switch is FOH.

Solution:

When the address FOH is placed in the address bus (A0-A7) during the execution of IN FOH by the 8085, the value in the address lines A0-A7 is as follows:

A7	A6	A5	A4	A3	A2	A1	A0	
1	1	1	1	0	0	0	0	= FOH

These address lines are connected to a NAND gate address decoder such that the output of the NAND gate is 0 under this condition. The output of the NAND gate decoder is ORed with the $\overline{IOR}$ signal and the output of the OR gate is connected to the enable inputs $\overline{1G}$ and $\overline{2G}$ of the 74LS244. When the NAND gate decoder's output and $\overline{IOR}$ are both 0, the 74LS244 is enabled and data from the DIP switch is placed in the data bus of the 8085. The 8085 reads that data and places it in the accumulator. Thus, the data from the DIP switch is transferred to the accumulator.

I/O-Mapped I/O Interfacing Problems

Example 2. Interface a set of eight LEDs in common cathode configuration with the 8085 such that the address assigned to them is F8H.

Solution:

The instruction OUT F8H is used to send the data in the accumulator to the LEDs. The steps involved in the execution of the instruction OUT F8H are as follows:

i. The address F8H is placed in the address lines A0-A7 and a copy of it in the lines A8-A15.

ii. The data in the accumulator is placed in the data bus (D0-D7).

iii. The $\overline{\text{IOW}}$ signal (i.e., $\overline{\text{IOW}}$= 0) is activated for some duration, during which the data gets stored in the external latch.

Figure 6.12 shows the interfacing of a set of eight LEDs with the 8085.

I/O-Mapped I/O Interfacing Problems

Example 2. Interface a set of eight LEDs in common cathode configuration with the 8085 such that the address assigned to them is F8H.

Solution:

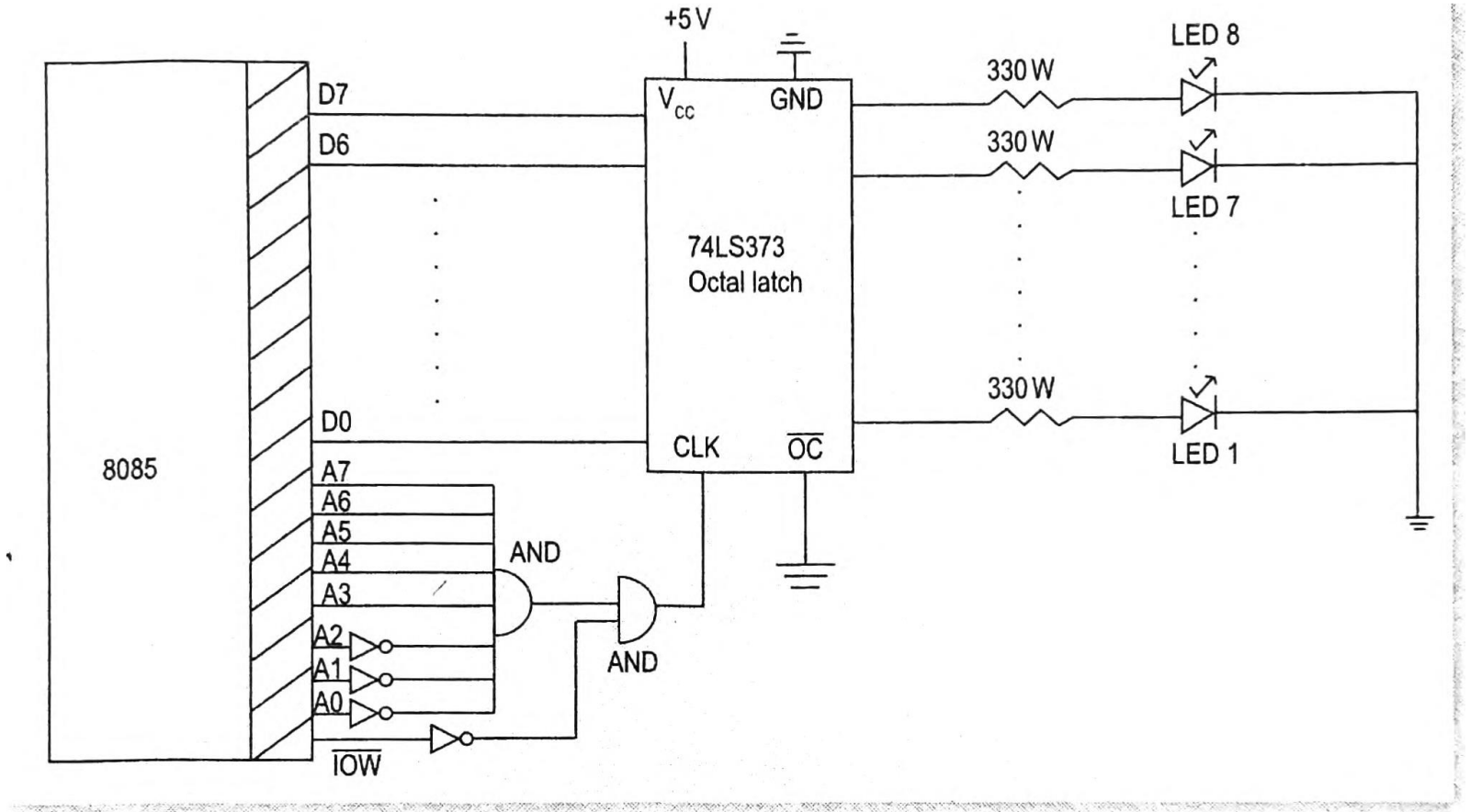


Fig. 6.12 Interfacing set of eight LEDs with the 8085 using 74LS373

I/O-Mapped I/O Interfacing Problems

Example 2. Interface a set of eight LEDs in common cathode configuration with the 8085 such that the address assigned to them is F8H.

Solution:

When the address F8H is placed in the address bus (A0-A7) during the execution of the instruction OUT F8H by the 8085, the value in the address lines A0-A7 is as follows:

A7	A6	A5	A4	A3	A2	A1	A0	
1	1	1	1	1	0	0	0	= F8H

When the instruction OUT F8H is executed by the 8085, the address F8H is placed in the address bus (A0-A7), the data in the accumulator is placed in the data bus (D0-D7), and the $\overline{IOW}$ signal is activated. This causes the activation of the clock input (CLK) of the 74LS373 (i.e., it is made 1) since both AND gates

produce logic 1 output. This latches the value in the data lines.

Memory-Mapped I/O Interfacing Problems

Example 1. Interface an 8-bit DIP switch with the 8085 using logic gates such that the address assigned to it is FOFOH.

Solution:

Since a 16-bit address has to be assigned to a DIP switch, the memory-mapped I/O technique must be used. Using LDA FOFOH instruction, the data from the 8-bit DIP switch can be transferred to the accumulator. The steps involved in the execution of the instruction LDA FOFOH are as follows:

i. The address FOFOH is placed in the address lines A0-A15.

ii. The MEMR signal is made low (i.e. logic 0) for some time.

iii. The data in the data bus is read and stored in the accumulator.

Figure 6.13 shows the interfacing of an 8-bit DIP switch with the 8085 so as to assign the address FOFOH to it.

Memory-Mapped I/O Interfacing Problems

Example 1. Interface an 8-bit DIP switch with the 8085 using logic gates such that the address assigned to it is FOFOH.

Solution:

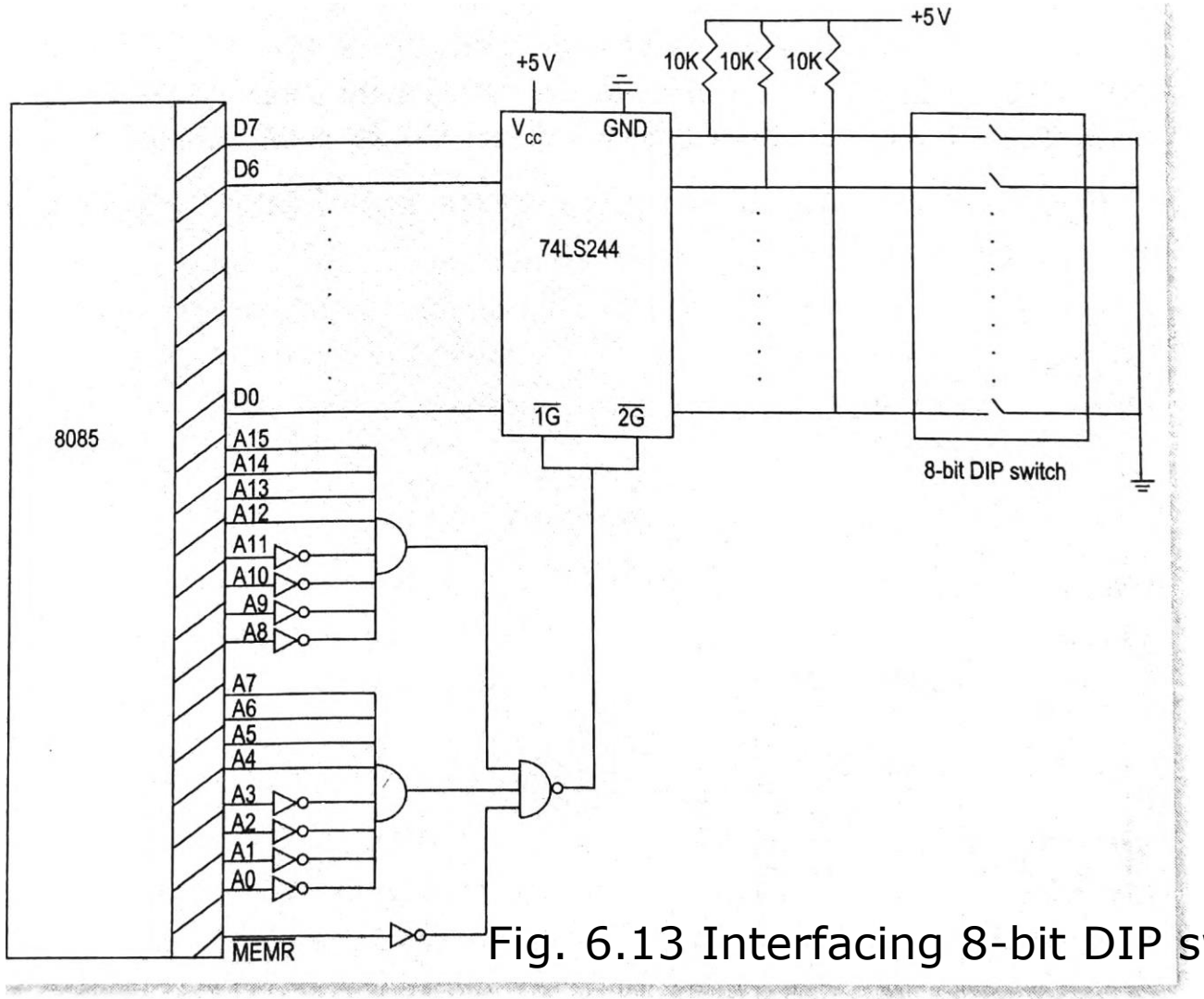


Fig. 6.13 Interfacing 8-bit DIP switch with 8085

Memory-Mapped I/O Interfacing Problems

Example 1. Interface an 8-bit DIP switch with the 8085 using logic gates such that the address assigned to it is FOFOH.

Solution:

When the 8085 executes the instruction LDA FOFOH, it places the address FOFOH in the address lines A0-A15 as follows:

A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 = FOFOH

Since 8-input AND gate ICs such as CD4068B are available, these address lines are connected to two 8-input AND gates so that the outputs of both the AND gates are 1 when the address FOFOH is placed in the address bus (A0-A15) by the 8085. These two outputs along with the inverted MEMR signal are given to a NAND gate so that its output becomes 0 when $\overline{\text{MEMR}}$ = 0, thereby enabling the buffer (74LS244). The data from the DIP switch is placed in the 8085's data bus. The 8085 reads the data from the data bus and stores it in the accumulator.

Memory-Mapped I/O Interfacing Problems

Example 2. Interface a seven-segment LED display with common cathode connection to the 8085 so as to assign the address FFF3H to it.

Solution:

Since a 16-bit address has to be assigned to the seven-segment LED display, memory-mapped I/O technique must be used. Using instructions such as STA FFF3H, the data in the accumulator can be sent to the LED to display a particular digit. The steps involved in the execution of the instruction STA FFF3H are as follows:

i. The address FFF3H is placed in the address lines A0-A15.

ii. The contents of the accumulator are placed in the data bus (D0-D7).

iii. The $\overline{\text{MEMW}}$ signal is activated (i.e., made 0) for some duration.

Figure 6.14 shows the interfacing of a seven-segment LED with the 8085 using memory-mapped I/O scheme.

Memory-Mapped I/O Interfacing Problems

Example 2. Interface a seven-segment LED display with common cathode connection to the 8085 so as to assign the address FFF3H to it.

Solution:

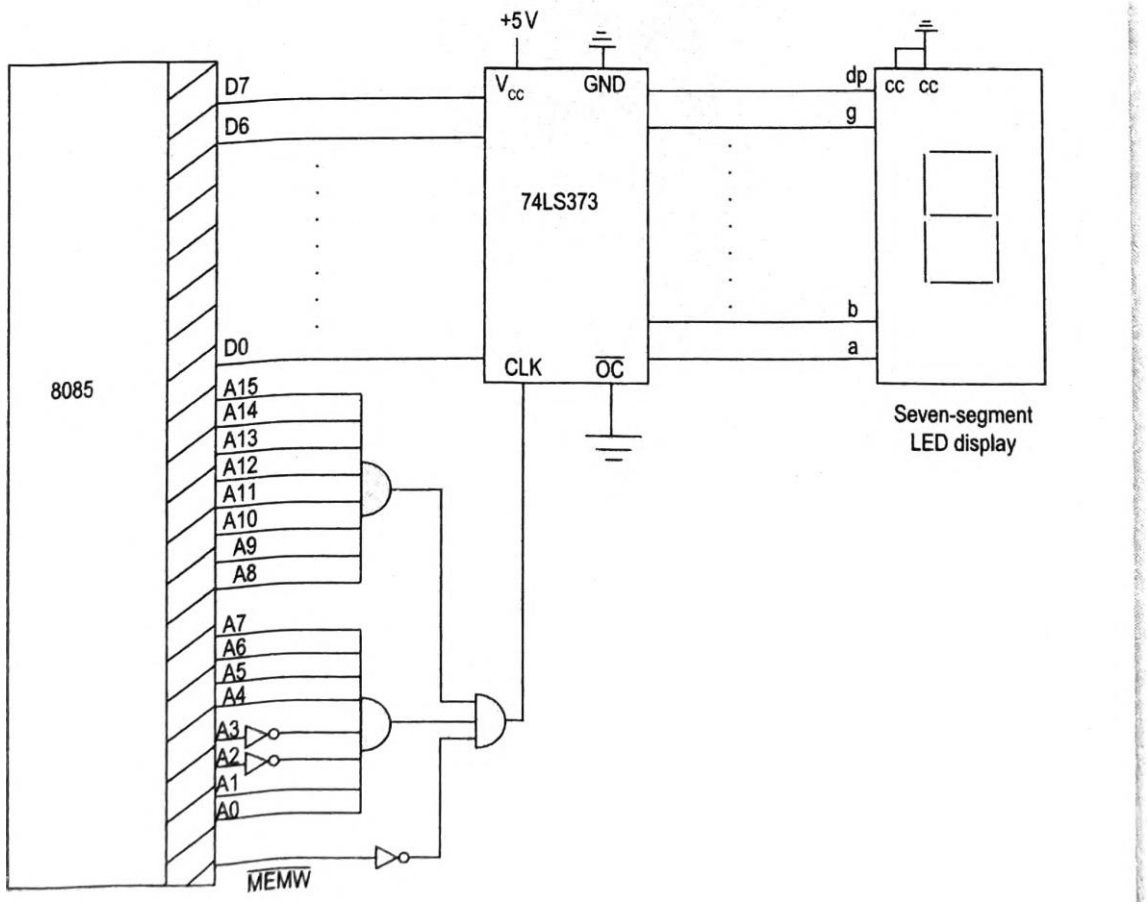


Fig. 6.14 Interfacing seven-segment LED with 8085

Memory-Mapped I/O Interfacing Problems

Example 2. Interface a seven-segment LED display with common cathode connection to the 8085 so as to assign the address FFF3H to it.

Solution:

When the 8085 executes the instruction STA FFF3H, the address FFF3H is placed in the address lines A0-A15 as follows:

A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 =FFF3H

The address lines are connected to the inputs of two AND gates such that both the AND gates produce 1 at their output for the address FFF3H. These two outputs along with the inverted $\overline{\text{MEMW}}$ signal from the 8085 are given as inputs to another AND gate whose output is connected to the CLK input of the latch. Therefore, when the 8085 executes the instruction STA FFF3H, the clock input of the 74LS374 becomes 1.The data in the data bus, which is the content of the accumulate or, is stored in the latch.

End of Unit 5
Thank You