8259 PROGRAMMABLE INTERRUPT CONTROLLER

SALIENT FEATURES

- 8259 is a **Programmable Interrupt Controller** (PIC) designed to work with various microprocessors such as **8085**, **8086** etc.
- **8259 is basically used to increase the number of interrupts.**
- A **single 8259** can handle **8 interrupts**
- A **cascaded** configuration of 8 slave 8259's and 1 master 8259 can handle **64 interrupts**.
- 8259 has a **flexible priority** structure.
- 8259 can **handle edge** as well as **level triggered** interrupts.
- In 8259 interrupts can be **masked** individually.
- The **Vector address** of the interrupts is **programmable**.
- Status of interrupts (pending, In-service, masked) can be easily read by the μP.

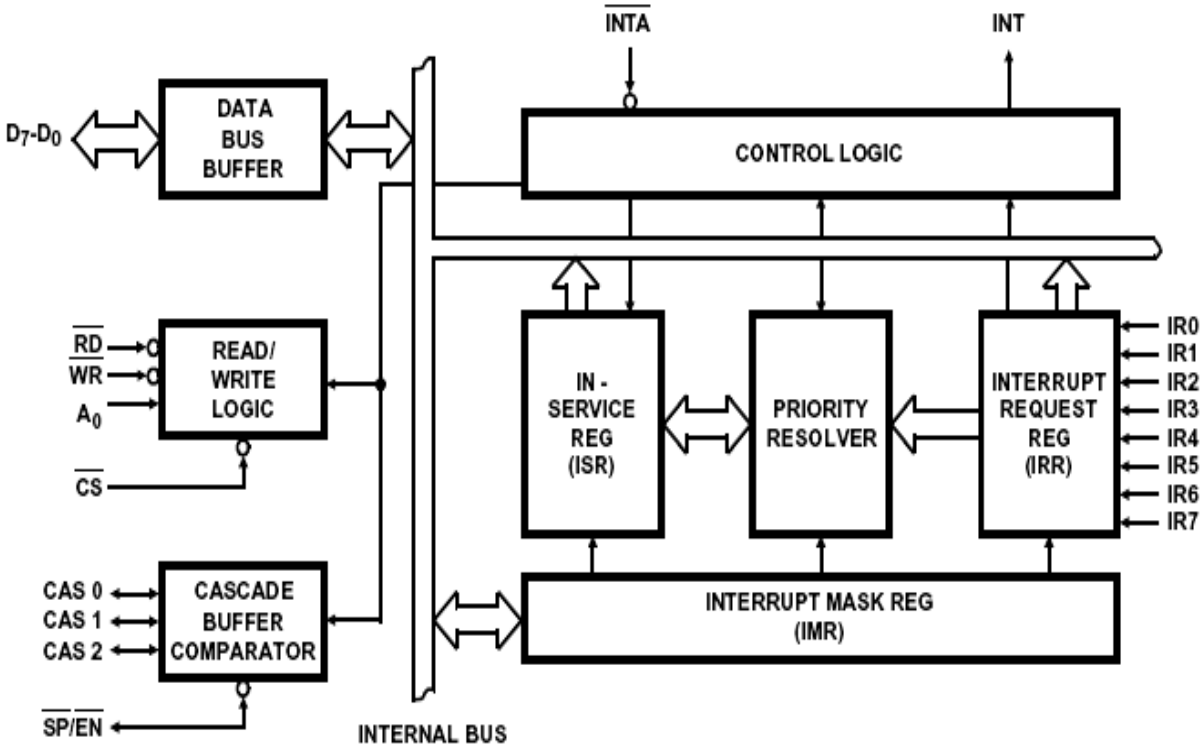
Note: Initialization of 8259

- 8259 must be compulsorily initialized.
- During Initialization we give commands to 8259 by which we decide priority, trigger, masking etc.
- The MOST important thing we give are vector addresses for the interrupts.
- In a cascaded configuration every 8259 must be individually initialized.

Special Note:

If you are learning this by piracy, then you are not my student.
You are simply a thief! #PoorUpbringing

ARCHITECTURE OF 8259



1) **Interrupt Request Register (IRR)**
   - 8259 has **8 interrupt** input lines **IR_7 … IR_0**.
   - The IRR is an **8-bit register** having **one bit** for **each** of the **interrupt** lines.
   - When an **interrupt request** occurs on any of these lines, the **corresponding bit** is **set** in the Interrupt Request Register (**IRR**).

2) **In-Service Register (InSR)**
   - It is an **8-bit** register, which **stores** the **level** of the Interrupt Request, which is **currently** being **serviced**.

3) **Interrupt Mask Register (IMR)**
   - It is an **8-bit** register, which stores the **masking pattern** for the interrupts of 8259.
   - It stores **one bit per interrupt level**.

4) Priority Resolver

- It **examines** the **IRR**, **InSR**, and **IMR** and determines which interrupt is of **highest priority** and should be sent to the µP.

5) Control Logic

- It has **INT output** signal **connected to** the **INTR** of the µP, to **send** the **Interrupt** to the µP.
- It also has the **INTA input** signal **connected to** the **INTA** of the µP, to **receive** the interrupt **acknowledge**.
- It is also used to control the remaining blocks.

6) Data Bus Buffer

- It is a bi-directional buffer used to **interface** the internal **data bus** of 8259 with the external (system) data bus.

7) Read/Write Logic

- It is used to accept the RD, WR, A_0 and CS signal.
- It also holds the Initialization Command Words (ICW's) and the Operational Command Words (OCW's).

8) Cascade Buffer / Comparator

- It is used in **cascaded mode** of operation.
- It has two components:

i. CAS_2, CAS_1, CAS_0 lines:

- These lines are **output for the master**, **input for the slave**.
- The **Master sends** the **address of** the **slave** on these lines (hence output).
- The **Salves read** the **address** on these lines (hence input).
- As there are 8 interrupt levels for the Master, there are **3 CAS lines ($\because 2^3 = 8$).**

ii. SP/EN (Slave Program/Master Enable):

- In **Buffered Mode**, it **functions** as the **EN line** and is used to **enable** the **buffer**.
- In **Non buffered mode**, it **functions** as the **SP output line**.
- **For Master** 8259 **SP** should be **high**, and **for** the **Slave SP** should be **low**.

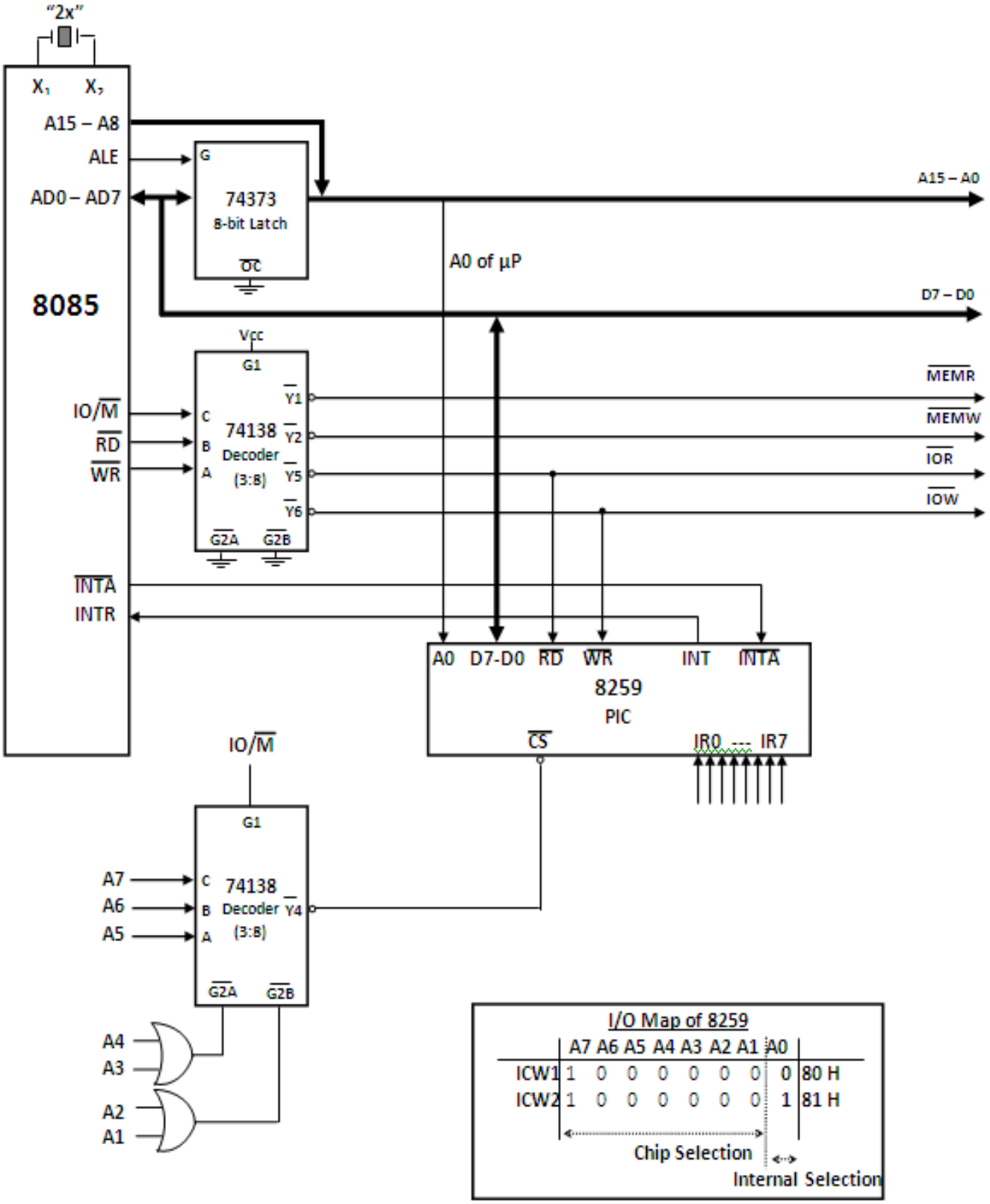
INTERFACING AND WORKING OF A SINGLE 8259

The working of a single 8259 with 8085, 8259 is explained below.

1. **First of all, interrupt INTR of 8085 must be enabled** by the **EI** instruction.
2. **8259** is **initialized** by giving the necessary commands. (**ICWs**)
3. **Once** 8259 is **initialized**, the **following sequence** of events takes place when one or more **interrupts occur** on the IR lines of the 8259.
4. The **corresponding bit** for an interrupt is **set in IRR**.
5. The **Priority Resolver checks** the 3 registers:
   IRR (for highest interrupt request)
   IMR (for the masking Status)
   InSR (for the current level serviced)
   and **determines** the **highest priority** interrupt.
   It **sends** the **INT** signal **to** the **µP**.
6. The **µP finishes** the **current instruction** and **acknowledges** the interrupt by **sending** the **first INTA pulse**.
7. On receiving the INTA signal, the **corresponding bit** in the **InSR** is **set** (indicating that the service of this interrupt is started) and the **bit** in the **IRR** is **reset** (to indicate that the request is accepted).
   8259 now **sends** the **Opcode of CALL** instruction **to the µP** on the data bus.
8. The **µP decodes** the **CALL instruction** and **sends 2 more INTA pulses** to the 8259.
9. In response to the two INTA pulses, the **8259 sends the address of the ISR to the µP**. First the lower byte and then the higher byte.
10. Thus, the complete 3-byte CALL Instruction code is released by the 8259.
11. The **µP pushes** the contents of **PC** onto the **Stack** and **transfers program to** the **address** of the **ISR** sent by the 8259. **The ISR thus begins**.
12. At the end of the ISR, 8085 will give an **EOI command t**o make the corresponding bit 0 in In Service Register. *In case of doubts WhatsApp: +919136428051 ()Only for regsitered students!)*

Special Note:

You will learn the interfacing diagram in the next page in later lectures. But the explanation of the interface is already covered as you learn the architecture.



I/O Map of 8259

	A7	A6	A5	A4	A3	A2	A1	A0	
ICW1	1	0	0	0	0	0	0	0	80 H
ICW2	1	0	0	0	0	0	0	1	81 H

Chip Selection
Internal Selection