Liquid Crystal Display Interfacing:

- LCD (Liquid Crystal Display) is a flat panel display technology commonly used in TVs, computer monitors, and mobile phones.
- It operates by aligning liquid crystal molecules between transparent electrodes and polarizing filters.
- LCDs are widely replacing LEDs due to their lower power consumption and compact size, making them ideal for space-constrained applications.
- Compared to CRTs, LCDs are more energy-efficient and lightweight, as they lack a bulky picture tube.
- The HITACHI 44780 is a widely used LCD controller that facilitates communication between microprocessors or microcontrollers and LCDs.

LCD Pin Description:

An LCD requires **3 control lines** (RS, R/W, and EN) and **8 or 4 data lines**, depending on the mode of operation. **8-bit mode** (faster) uses all 8 data lines, while **4-bit mode** uses only 4. In total, an 8-bit mode LCD has **14 pins**, including **3 power supply pins (Vcc, Vss, Vee)**.

1. Power Supply Pins:

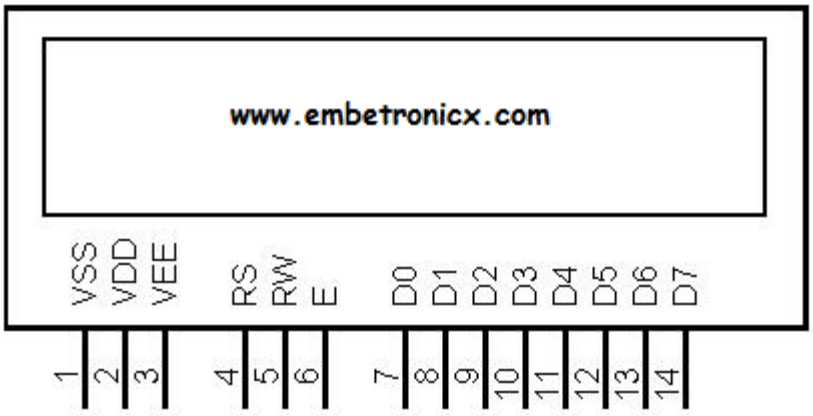
- **Vcc** – Provides **+5V** power.
- **Vss** – **Ground (0V)** connection.
- **Vee** – Adjusts **LCD contrast**.

2. Control Pins:

- **RS (Register Select):**
    - **0** → Selects **command register** (for LCD instructions).
    - **1** → Selects **data register** (for displaying data).
- **R/W (Read/Write):**
    - **0** → **Write** operation.
    - **1** → **Read** operation.
- **EN (Enable):** Triggers data latching with a **high-to-low pulse**.

3. Data Lines (D0–D7):

- Used for sending **data/commands** to the LCD or **reading** LCD registers.
- Operates in either **8-bit or 4-bit mode**.



Pin Description for LCD:

Pin No	Symbol	Description
1	Vss	Ground
2	Vcc	+5V power supply
3	Vee	power supply to control the contrast
4	RS	register select
5	R/W	Read/Write
6	EN	Enable
7-14	D0-D7	8- bit data lines

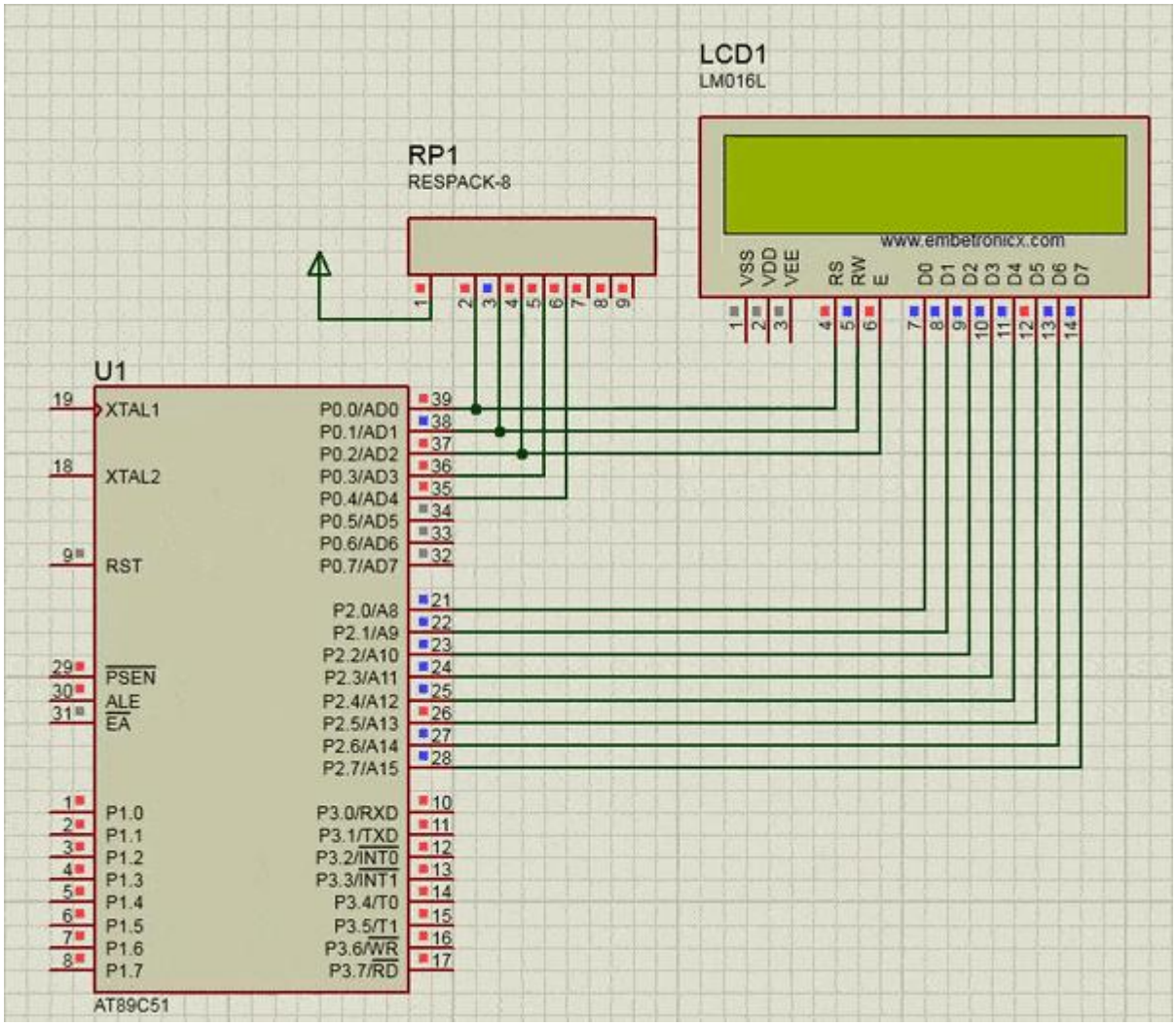
LCD Command Words:

LCD commands are special instructions sent to the **command register** to control display operations such as clearing the screen, moving the cursor, or blinking the text. To send a command, **RS = 0** and **R/W = 0**.

Hex code	Description
01	Clear display screen

Hex code	Description
02	Return home
04	Decrement cursor (shift cursor from right to left)
06	Increment cursor (shift cursor from left to right)
05	Shift display right
07	Shift display left
08	Display off. cursor off
0A	Display off. cursor on
0C	Display on. cursor off
0E	Display on, cursor blinking
0F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to the right
18	Shift the entire display to left
1C	Shift the entire display to the right

Hex code	Description
80	Force cursor to the beginning of 1″ line
C0	Force cursor to the beginning of 2 line
38	2 lines and 5×7 matrix



LCD Interfacing with 8085 Microprocessor

Interfacing an LCD with an **8085 microprocessor** is useful for displaying alphanumeric characters. Since the **8085 does not have built-in support for LCDs**, the **8255 Programmable Peripheral Interface (PPI) IC** is used as a bridge.

Working Principle

The **8085 processor** sends **ASCII-coded data** (A–Z, a–z, 0–9) to the LCD via **8255 PPI**. The control signals ensure proper data transfer, allowing the LCD to display characters correctly.

Connections and Working

1. Data Transfer

- **Port A of 8255** → Sends **8-bit ASCII data** (D0–D7) to the LCD.
- **Port B of 8255** → Controls the LCD using **RS and EN pins**.

2. Control Signals

- **RS (Register Select):**
    - **0 → Command Mode** (Sends instructions like clear display, cursor shift).
    - **1 → Data Mode** (Sends actual text to display).
- **R/W (Read/Write):** Always **0** (Grounded) since LCD is only used for writing.
- **EN (Enable):** Used to latch the data (A **high-to-low pulse** is required).

3. Power Supply and Contrast Adjustment

- **Vcc (+5V)** → Power supply.
- **Vss (0V)** → Ground connection.
- **Vee (Contrast Control):** Connected to a **potentiometer** (adjusts display clarity).

Steps for LCD Interfacing in 8085

1. **Initialize LCD** (Send initialization commands like setting cursor, display mode).
2. **Send Data or Commands** (Decide between writing a command or displaying text).
3. **Latch the Data** (Use EN signal for proper data transfer).
4. **Repeat** the process for continuous display updates.

Advantages of Using LCD with 8085

- **Clear alphanumeric display** for real-time data.
- **Low power consumption** compared to LED displays.
- **Easier user interaction** in embedded systems.

This interfacing method is widely used in **microprocessor-based systems**, digital meters, and embedded applications. 🚀

External Memory Interfacing in 8051 Microcontroller:

- The 8051 microcontroller has limited internal memory, but it can be expanded by connecting external RAM and ROM. This allows it to run larger programs and store more data.
- External memory is connected using a data bus (for transferring data) and an address bus (for selecting memory locations). The 8051 uses special pins like ALE, PSEN, RD, and WR to manage communication with external memory.
- While the 8051 has built-in memory for basic tasks, some applications require more space. In such cases, external ROM (for program storage) and RAM (for data storage) can be added to enhance the microcontroller's capability.

Why is External Memory Interfacing Needed in the 8051 Microcontroller?

External memory interfacing is essential in the 8051 microcontroller due to the following reasons:

- **Limited Internal Memory:** The 8051 has only 128 bytes of RAM and 4KB of ROM, which may not be enough for complex applications.
- **Larger Programs:** For running advanced algorithms or multiple functions, external memory provides the extra space needed.
- **Data Storage:** Applications like data logging or analysis require more memory to store large amounts of data.
- **Flexibility:** Adding external memory allows customization and adaptability for different project needs.
- **Cost-Effective:** External RAM and ROM are affordable, making them a budget-friendly way to expand memory instead of relying on expensive built-in options.

External Program Memory in 8051 Microcontroller

The 8051 microcontroller can use external ROM/EPROM for program storage when its internal memory is insufficient.

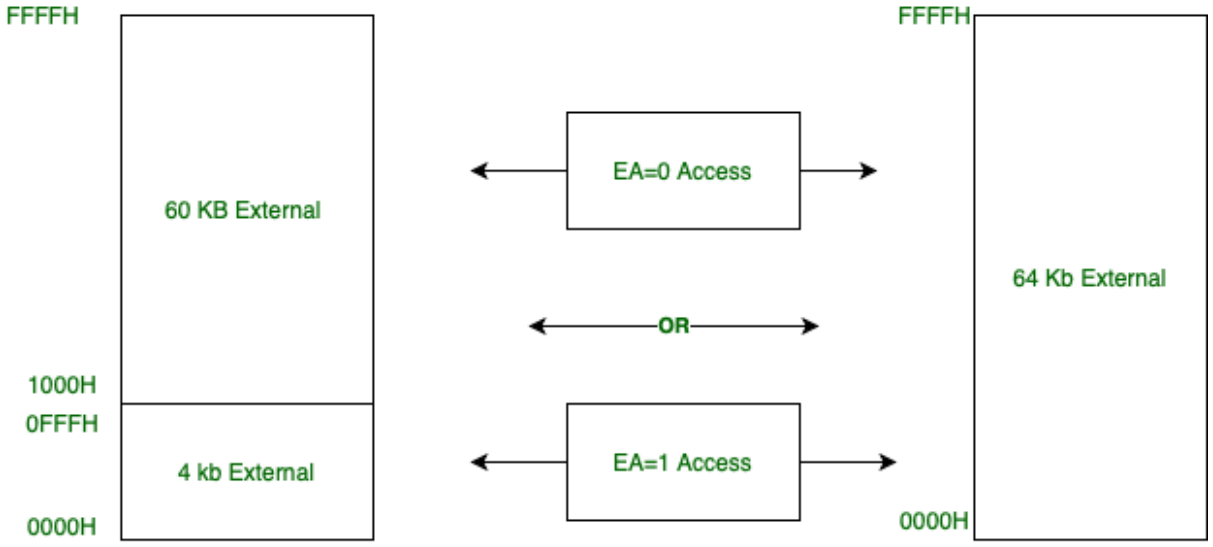
- **Memory Access Control:**
    - If the **EA (External Access) pin** is high, the 8051 fetches programs from internal ROM for addresses **0000H to 0FFFH** and from external memory for **1000H to FFFFH**.
    - If the **EA pin is low**, all program memory accesses (0000H to FFFFH) go to external memory.
- **Data Bus and Addressing:**
    - **Port 0** is used as a multiplexed address and data bus.
    - During the first clock cycle (T-cycle), it provides the **lower 8-bit address**.
    - Later, it functions as a **data bus** to transfer data between the microcontroller and external memory.

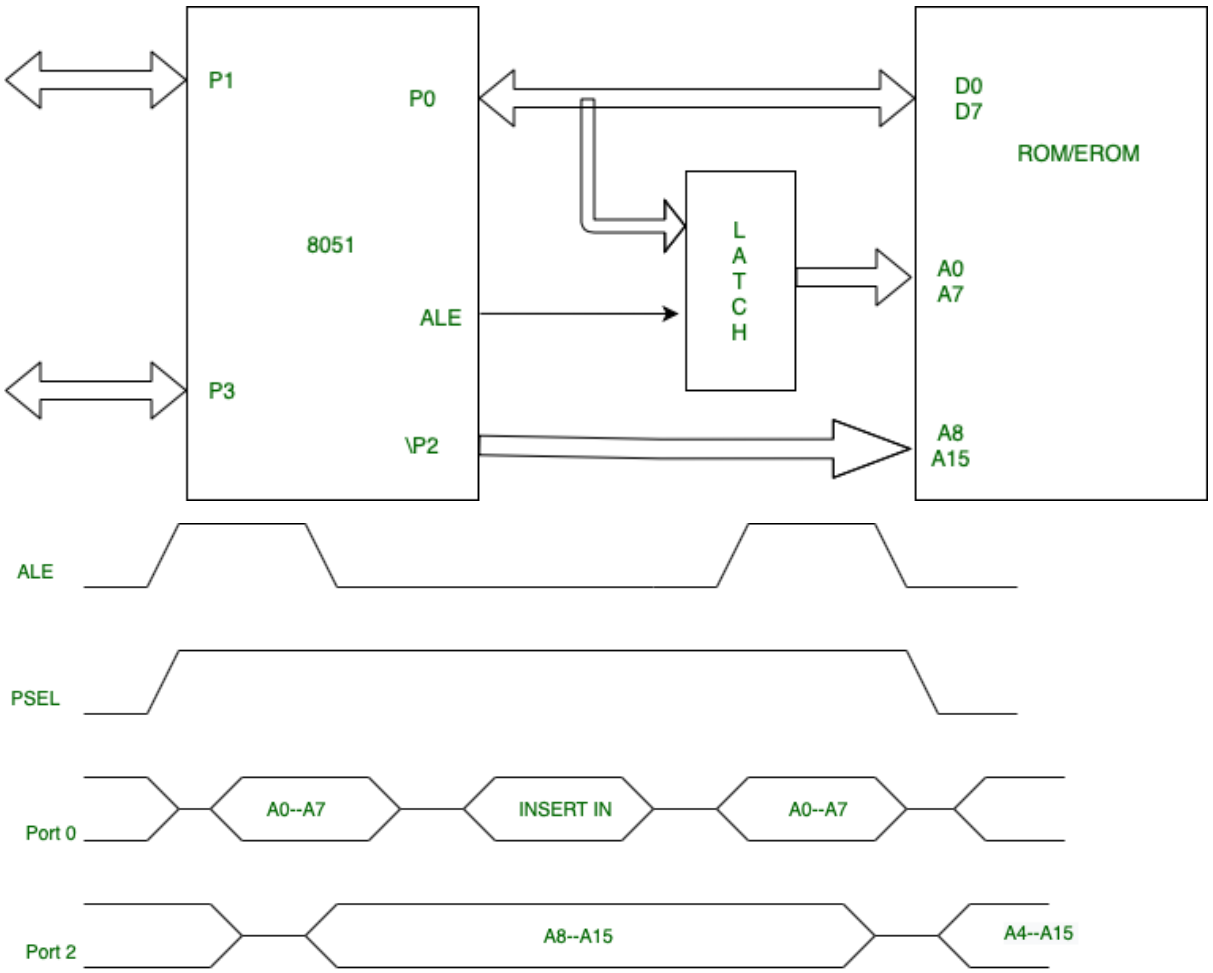
- An **external latch** and the **ALE (Address Latch Enable) signal** are used to store the 8-bit address.

- **Program Memory Timing:**
  - The **PSEN (Program Store Enable)** signal activates external ROM/EPROM, ensuring proper program execution.
  - The external memory is read in sync with the microcontroller's timing to fetch instructions correctly.

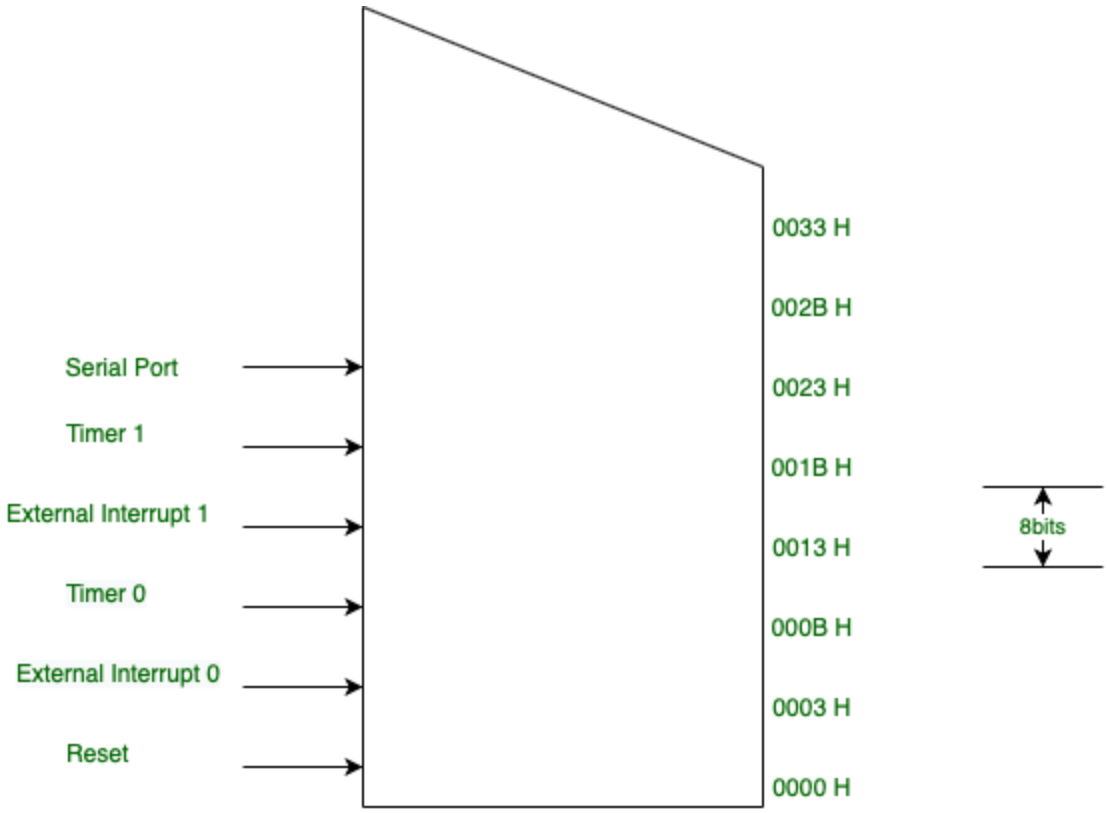
- **Interrupt Vector Table:**
  - Specific memory locations are reserved for interrupts. Examples include:
    - External Interrupt 0: **0003H**
    - Timer 0: **000BH**
    - External Interrupt 1: **0013H**
    - Timer 1: **001BH**
  - If an interrupt is not used, its reserved memory space can be used for general program storage.

External program memory interfacing allows the 8051 to run larger applications by extending its code storage capacity efficiently.





timing waveform for external data memory write cycle



Instructions to Access External ROM / Program Memory :
This table is explaining the instructions to access external ROM/program memory.

Mnemonic	**Operation**
MOVC A, @ A+DPTR	Copy the contents of the external ROM address formed by adding A and the DPTR, to A
MOVC A, @ A + PC	This operation will do copy This operation contents of the external ROM address formed by adding A and the PC, to A.

External Memory Interfacing in 8051 Microcontroller

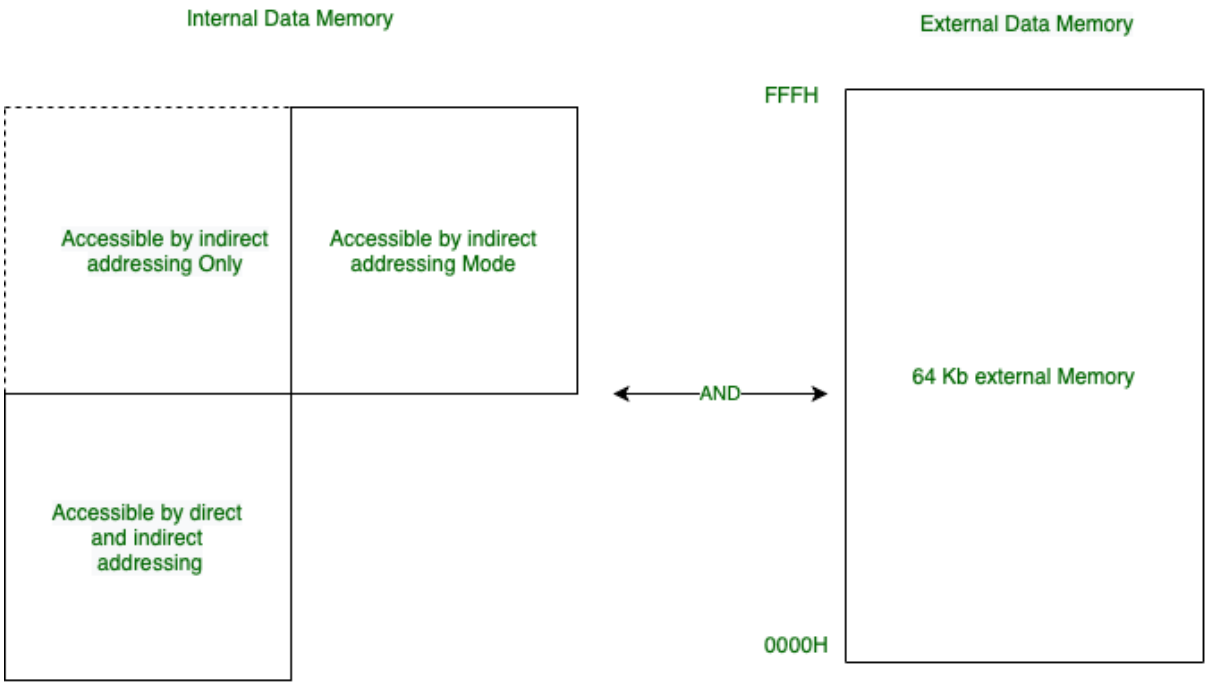
The 8051 microcontroller can address up to **64KB of external data memory**, expanding its storage capabilities for complex applications.

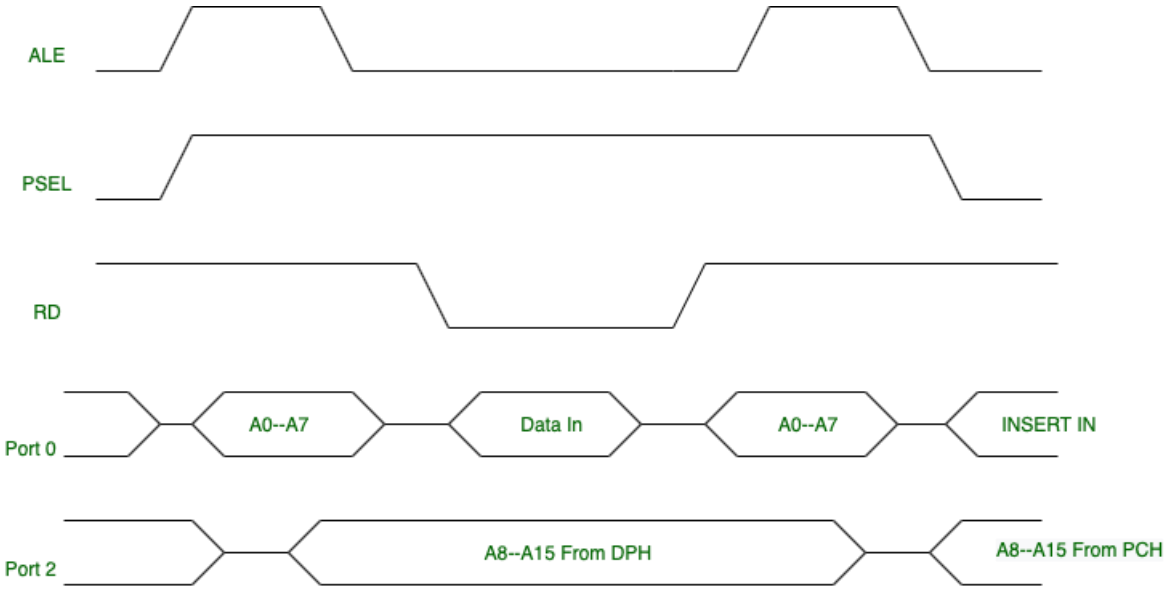
- **Accessing External Data Memory:**
    - The **MOVX instruction** is used to read from or write to external memory.
    - External RAM or other memory devices can be connected for data storage beyond the internal limits.
- **Internal Data Memory Structure:**

- The internal memory is divided into three sections:
    1. **Lower 128 bytes (00H–7FH)** – General-purpose RAM.
    2. **Upper 128 bytes (80H–FFH)** – Accessible only via **indirect addressing**.
    3. **Special Function Registers (SFRs) (80H–FFH)** – Accessible only via **direct addressing**.

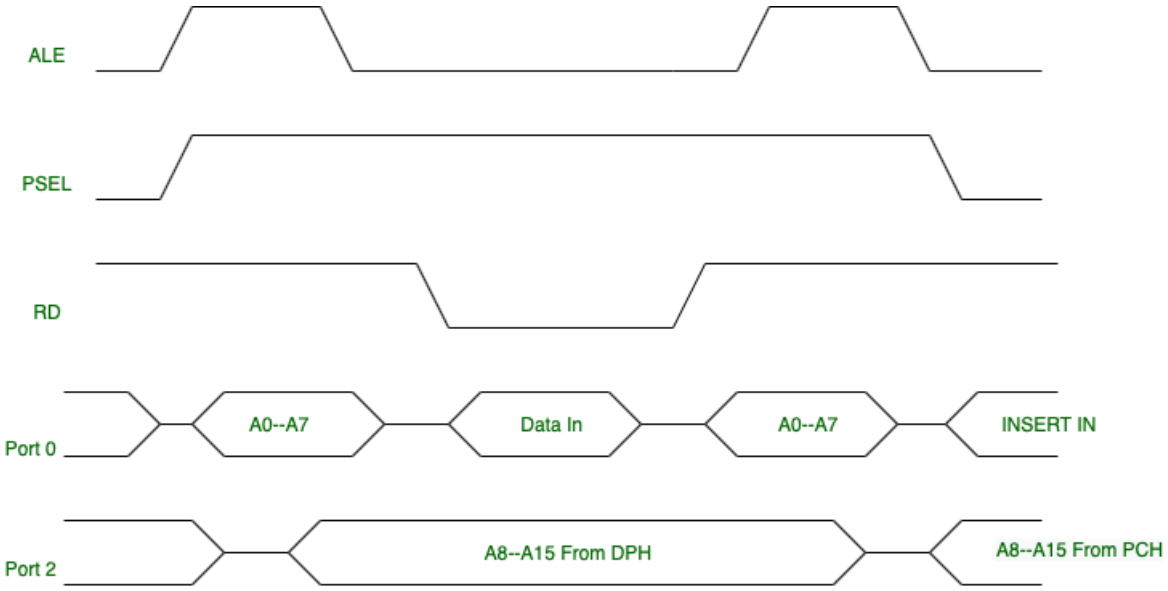
- **Addressing Modes:**
    - The **upper address space (80H–FFH)** is shared between **SFRs and Upper RAM**.
    - **SFRs** are accessed using **direct addressing**.
    - **Upper RAM** is accessed using **indirect addressing**.
    - **External memory** can be accessed using **both direct and indirect addressing**.
- **Timing for External Memory Access:**
    - The microcontroller follows a specific read and write cycle to interact with external memory.
    - Timing waveforms (as shown in figures) illustrate how data is transferred during read and write operations.

By interfacing external memory, the 8051 can efficiently handle larger programs and store more data, making it suitable for advanced embedded applications.





timing waveform for external data memory read cycle



timing waveform for external data memory write cycle

Instructions to Access External Data Memory :

The Table explains the instruction to access external data memory.

Mnemonic	Operation
MOVX A, @Rp	In this operation, it will copy the contents of the external address in Rp to A.
MOVX A. @DPTR	Copy the contents of the external address in DPTR to A.
MOVX @Rp. A	Copy data from A to the external address in Rp
MOVX DPTR, A	Copy data from A to the external address in DPTR.

Important Points to Remember in Accessing External Memory :

- In the Case of accessing external memory, All external data moves with external RAM or ROM involve the A register.
- While accessing external memory, R can address 256 bytes and DPTR can address 64 k-bytes
- MOV X instruction is used to access external RAM or 1/O addresses.

Note –
It must be noted that while the Program counter(PC) will be used then to access external ROM, it will be incremented by 1 (to point to the next instruction) before it is added to A to form the physical address of external ROM.

Memory Address Decoding in 8051 Microcontroller

Memory address decoding is essential for identifying and selecting specific memory locations in external RAM or ROM. Each memory register has a unique address, and the number of registers depends on the number of address lines.

Example Calculations for Memory Addressing:

1. If a memory has **12 address lines** and **8 data lines**, the number of memory locations is:

212=4096 (4K locations)2^{12} = 4096 \text{ (4K locations)}212=4096 (4K locations)

The word length is **8 bits** (since it has 8 data lines).

2. If a memory has **8192 locations**, the number of required address lines is:

213=81922^{13} = 8192213=8192

So, **13 address lines** are needed.

Memory Capacity and Address Lines Required:

Memory Capacity	Address Lines Required
1K (1024)	10
2K (2048)	11
4K (4096)	12
8K (8192)	13
16K (16384)	14
32K (32768)	15
64K (65536)	16

Address Decoding Process:

- The memory chip typically has address lines (e.g., **A0 to A11 for a 4K memory**), a **chip select (CS)**, and **control signals (RD, WR)**.
- **EPROM (Read-Only Memory) does not need the WR signal** because it only supports reading.
- Since the **microprocessor/microcontroller shares data, address, and control buses among multiple devices**, decoding is necessary to ensure only one device is accessed at a time.

Address Decoding Techniques:

1. **Absolute Decoding (Full Decoding):**
    - Uses **all address lines** for decoding.
    - Provides **precise** memory selection.
    - Requires **more hardware** (additional logic gates or decoders).
2. **Linear Decoding (Partial Decoding):**
    - Uses **fewer address lines**, ignoring some higher-order bits.
    - Reduces hardware complexity.
    - May lead to **memory aliasing**, where multiple addresses refer to the same location.

Address decoding ensures proper memory access and prevents conflicts when multiple devices share the system bus.