Microprocessors & Interfacing

IO Interface

BITS Pilani

Dr. Gargi Prabhu
Department of CS & IS

Fixed Addresses

- This term typically means that each individual I/O device is assigned a specific, predetermined address.
- In systems with fixed-address I/O, the address assigned to each device remains constant and does not change.
- For example, if an 8-bit I/O device is connected to the system, it might be assigned a fixed address such as 0x1000. This means that regardless of the data being transferred, the device is always accessed at address 0x1000.

```
IN AL, 0200h  ; Read data from the fixed address 0200h into AL register

MOV AL, 0xFF  ; Data to be written
OUT 0200h, AL ; Write data from AL register to the fixed address 0200h
```

Variable Addresses

- In contrast, variable-address I/O means that the addresses at which I/O devices are connected can vary.
- This could be due to various factors such as the configuration of the system or the dynamic allocation of resources.
- In such systems, the addresses assigned to I/O devices may not be constant and could change under different conditions or configurations.

```
IN AX, 0300h  ; Read data from the variable address 0300h into AX register;

MOV AX, 0xFFFF  ; Data to be written
OUT 0400h, AX   ; Write data from AX register to the variable address 0400h
```

Addressing of I/O Device

- The address of an I/O device is determined by the system architecture and is usually assigned by the system designer or the operating system.
- Each I/O device is assigned a unique address within the I/O address space, which is separate from the memory address space.
- The I/O address space can range from 0x0000 to 0xFFFF (64 KB) in the case of the Intel 8086 architecture, for example.
- The address lines of the microprocessor are used to specify the address of the I/O device being accessed

Port Size

- The port size refers to the width of the data bus used for communication between the microprocessor and the I/O device.
- It determines the number of data lines over which data can be transferred in parallel between the microprocessor and the I/O device.
- For example, an 8-bit port has 8 data lines, a 16-bit port has 16 data lines, and a 32-bit port has 32 data lines

How microprocessor connects IO device

- When accessing an I/O device, the microprocessor generates the appropriate address on the address bus to select the desired device, and the data lines are used for transferring data between the microprocessor and the device.
- The size of the I/O port (data bus width) determines how many bits of data can be transferred in parallel during each I/O operation.
- However, the specific address of the I/O device is determined independently of the port size

8-bit I/O Ports

- Data bus width: 8 bits.
- Can transfer 8 bits of data in parallel.
- Suitable for interfacing with devices that require transferring small amounts of data at a time, such as simple sensors, LEDs, or switches.
- Limited in data transfer capability compared to wider ports but can be more efficient for simple tasks and reduce hardware complexity.
- IN AL, 0x100 ; Read data from port 0x100 into AL register ;

- MOV AL, 0xFF ; Data to be written
- OUT 0x100, AL ; Write data from AL register to port 0x100

16-bit I/O Ports

- Data bus width: 16 bits.
- Can transfer 16 bits of data in parallel.
- Provides higher data transfer capability compared to 8-bit ports.
- Suitable for interfacing with devices that require transferring larger amounts of data or need higher throughput, such as LCD displays, some types of memory devices, or certain communication interfaces.
- IN AX, 0x200 ; Read data from port 0x200 into AX register ;

- MOV AX, 0xFFFF ; Data to be written
- OUT 0x200, AX ; Write data from AX register to port 0x200

32-bit I/O Ports

- Data bus width: 32 bits.
- Can transfer 32 bits of data in parallel.
- Offers even higher data transfer capability compared to 16-bit ports.
- Often used in more advanced systems requiring fast data transfer rates, such as graphics cards, high-speed communication interfaces (e.g., Ethernet), or memory-mapped devices with large data buses.
- IN EAX, 0x300 ; Read data from port 0x300 into EAX register ;

- MOV EAX, 0xFFFFFFFF ; Data to be written
- OUT 0x300, EAX ; Write data from EAX register to port 0x300

Basic Input and Output Interfaces

- The basic input device is a set of three-state buffers. The basic output device is a set of data latches.
- The term IN refers to moving data from the I/O device into the microprocessor and the term OUT refers to moving data out of the microprocessor to the I/O device.

Three-State Bus Buffers

- Three-state bus buffers, also known as tri-state buffers, are electronic components commonly used in digital circuits to control the flow of data on a bus.
- A bus is a collection of wires that carry multiple signals simultaneously.
- Tri-state buffers allow a device connected to a bus to either drive data onto the bus, receive data from the bus, or effectively disconnect from the bus altogether, entering a high-impedance state.

Three-State Bus Buffers

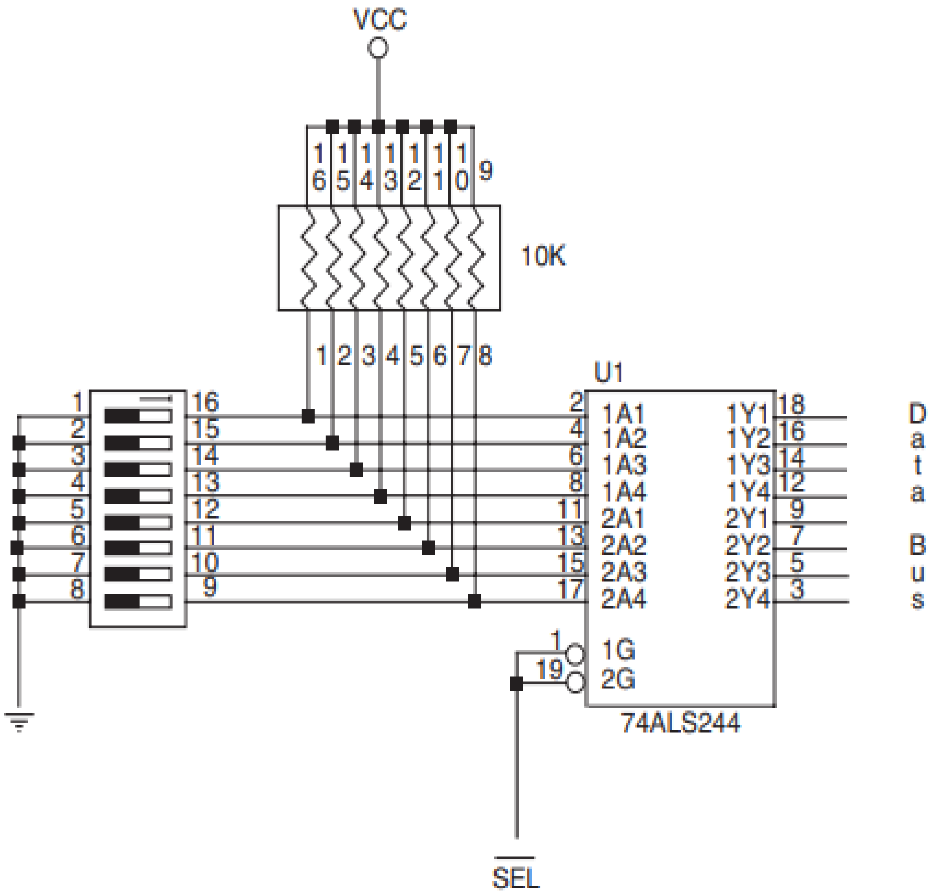
Here's how a tri-state buffer typically operates:

Active High Enable: Tri-state buffers have an enable input. When the enable input is active (usually high), the buffer operates normally, allowing data to pass through from its input to its output.

Active Low Enable: Some tri-state buffers have an active low enable input. In such cases, the buffer operates when the enable input is low.

High-Impedance State (Hi-Z): When the enable input is inactive (low or high depending on the design), the buffer enters a high-impedance state. In this state, the buffer effectively disconnects its input from its output, allowing other devices connected to the bus to drive the signals without interference.

Basic Input Interface

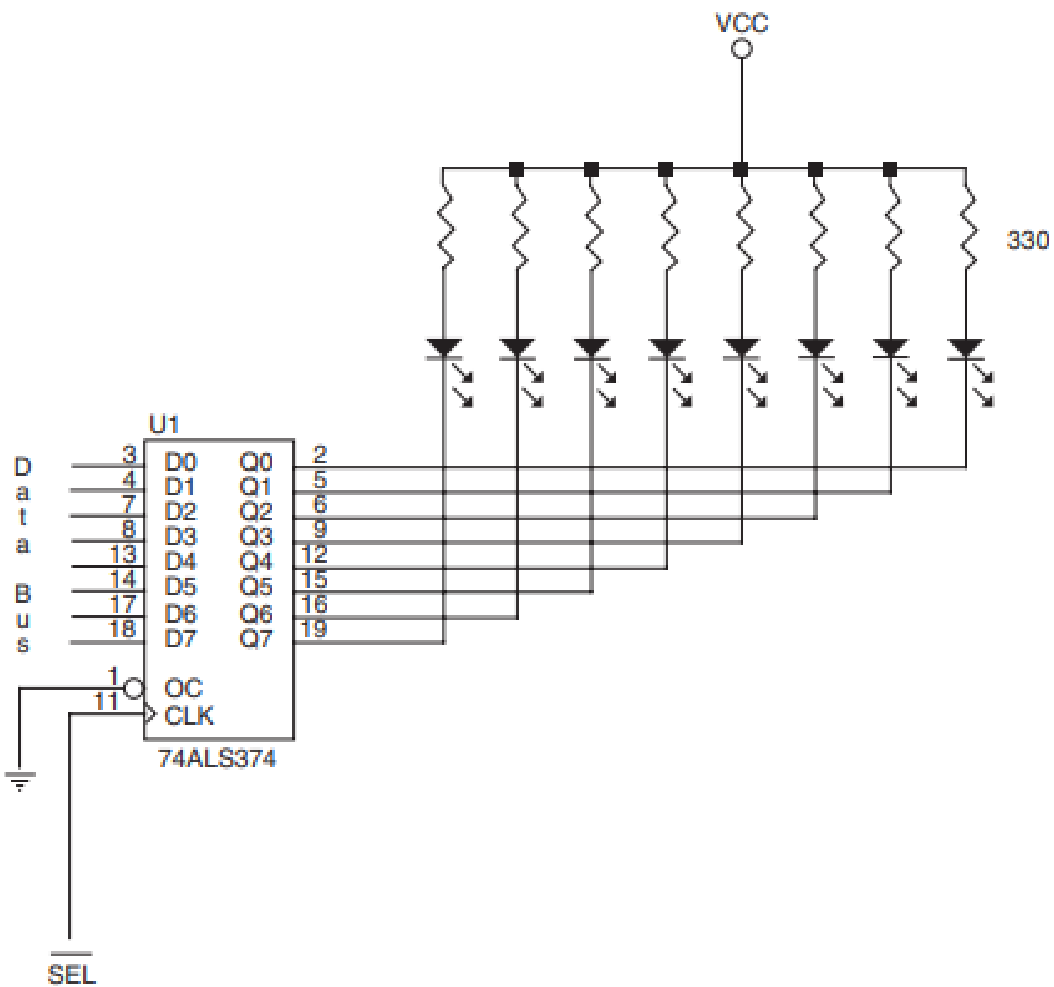


- When the microprocessor executes an IN instruction, the I/O port address is decoded to generate the logic 0 on SEL'.
- A 0 placed on the output control inputs (1G and 2G) of the 74ALS244 buffer causes the data input connections (A) to be connected to the data output (Y) connections.

Basic Output Interface

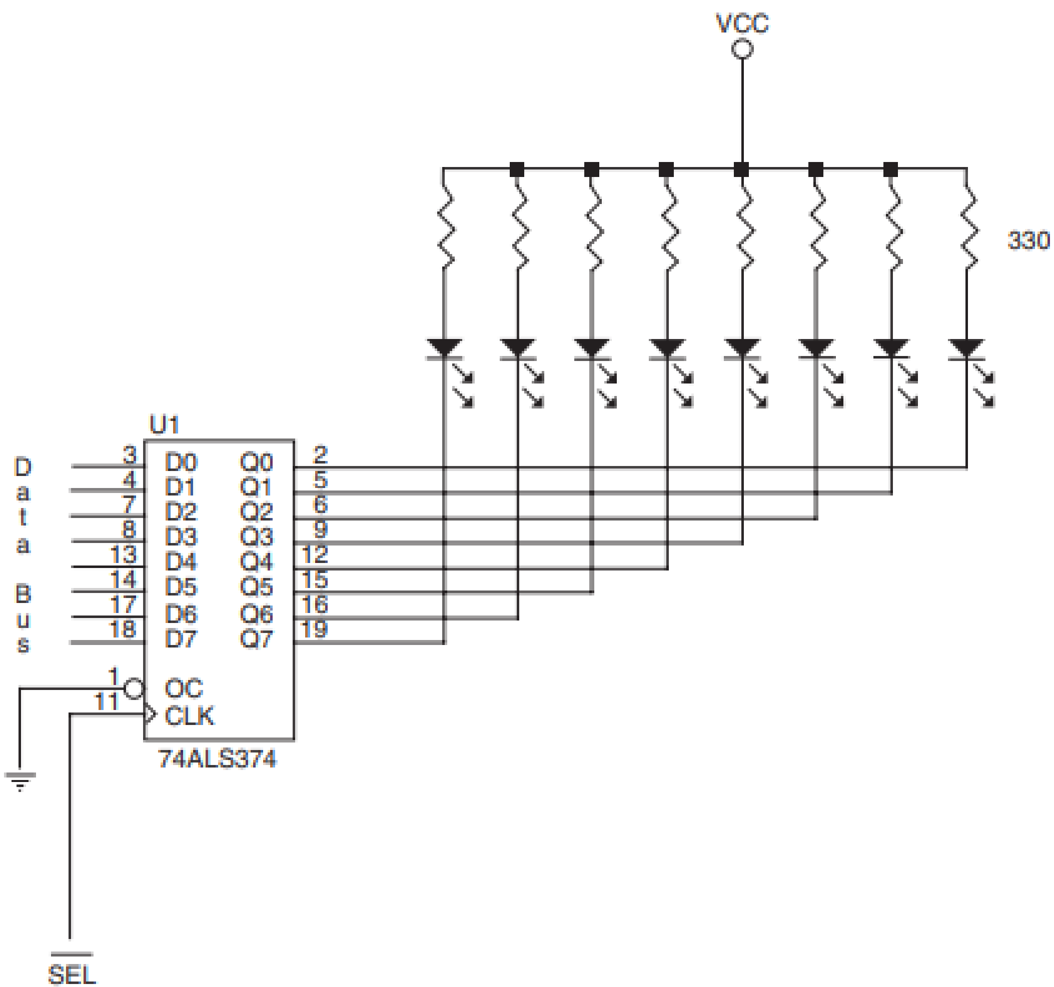
- The basic output interface receives data from the microprocessor and usually must hold it for some external device.
- Its latches or flip-flops, like the buffers found in the input device, are often built into the I/O device

Basic Output Interface



- When the OUT instruction executes, the data from AL, AX, or EAX are transferred to the latch via the data bus
- 8 Emitting LEDs are connected.
- D inputs of a 74ALS374 octal latch are connected to the data bus to capture the output data, and the Q outputs of the latch are attached to the LEDs.
- When a Q output becomes a logic 0, the LED lights.

Basic Output Interface



- Each time that the OUT instruction executes, the signal to the latch activates, capturing the data output to the latch from any 8-bit section of the data bus.
- The data are held until the next OUT instruction executes.
- Thus, whenever the output instruction is executed in this circuit, the data from the AL register appear on the LEDs

Handshaking

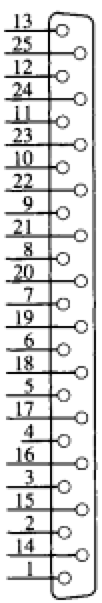
- Many I/O devices accept or release information at a much slower rate than the microprocessor.
- Another method of I/O control, called handshaking or polling, synchronizes the I/O device with the microprocessor.
- Example: parallel printer that prints a few hundred characters per second (CPS)
- It is obvious that the microprocessor can send more than a few hundred CPS to the printer, so a way to slow the microprocessor down to match speeds with the printer must be developed

Handshaking

- Data are transferred through a series of data connections (D7–D0).
- BUSY indicates that the printer is busy.
- STB' is a clock pulse used to send data to the printer for printing
- The ASCII data to be printed by the printer are placed on D7–D0, and a pulse is then applied to the connection.
- The strobe signal sends or clocks the data into the printer so that they can be printed.
- As soon as the printer receives the data, it places a logic 1 on the BUSY pin, indicating that the printer is busy printing data.
- The microprocessor software polls or tests the BUSY pin to decide whether the printer is busy.
- If the printer is busy, the microprocessor waits; if it is not busy, the microprocessor sends the next ASCII character to the printer
- This process of interrogating the printer, or any asynchronous device like a printer, is called handshaking or polling.

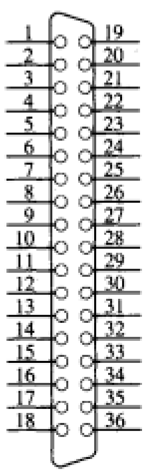
Handshaking

- A simple procedure that tests the printer BUSY flag and then sends data to the printer if it is not busy.
- Here, the PRINT procedure prints the ASCII-coded contents of BL only if the BUSY flag is a logic 0, indicating that the printer is not busy.
- This procedure is called each time a character is to be printed.



Connector DB25

(microprocessor)



Connector CENT36

(printer)

DB25 Pin number	CENT36 Pin number	Function	DB25 Pin number	CENT36 Pin number	Function
1	1	$\overline{\text{Data Strobe}}$	12	12	Paper empty
2	2	Data 0 (D0)	13	13	Select
3	3	Data 1 (D1)	14	14	Afd
4	4	Data 2 (D2)	15	32	$\overline{\text{Error}}$
5	5	Data 3 (D3)	16	—	$\overline{\text{RESET}}$
6	6	Data 4 (D4)	17	31	Select in
7	7	Data 5 (D5)	18—25	19—30	Ground
8	8	Data 6 (D6)	—	17	Frame ground
9	9	Data 7 (D7)	—	16	Ground
10	10	$\overline{\text{Ack}}$	—	33	Ground
11	11	Busy			

Example

```
;An assembly language procedure that prints the ASCII contents of BL.

PRINT   PROC    NEAR

        .REPEAT                              ;test the busy flag
                IN AL,BUSY
                TEST AL,BUSY_BIT
        .UNTIL ZERO
        MOV AL,BL                            ;position data in AL
        OUT PRINTER,AL                       ;print data
        RET

PRINT   ENDP
```

innovate achieve lead

BITS Pilani
Pilani Campus

Thank You