Computer Organisation and Architecture

Course Code: CO206

Module- 4- Input Output Organisation

Contents of Module 4

Input Output Subsystems

- I/O Subsystem **provides an efficient mode of communication between** the **central system** and the **outside environment**.

- **Programs and data must be entered into computer** memory for processing and **results obtained** from computer **must be recorded and displayed to user**.

Peripheral Devices

- **Devices** that are **under direct control of computer** are said to be **connected on-line.**
- **Input or output devices** that are **attached to the computer** are called **peripherals.**
- Among the **common peripheral** are **keyboards**, **mouse**, **display units**, and **printers** etc.
- **Peripherals that provide auxiliary storage** for the systems are **magnetic disks** and **tapes**.
- Peripherals are electromechanical and electromagnetic devices.

Classification of Peripheral Devices

- ❑Input Peripherals
- ❑Output Peripherals
- ❑Input Output Peripherals

Classification of Peripheral Devices(based on readability)

- **Human Readable**
  - Printers, Monitors
- **Machine Readable**
  - Disks, Tapes, Sensors, Controllers
- **Communication**
  - Modem, Network Interface Card

Examples of Peripheral Devices

Input Devices

- Keyboard
- Optical input devices
  - Card Reader
  - Paper Tape Reader
  - Bar code reader
  - Optical Mark Reader
- Magnetic Input Devices
  - Magnetic Stripe Reader
- Screen Input Devices
  - Touch Screen
  - Light Pen
  - Mouse
- Analog Input Devices

Output Devices

- Card Puncher, Paper Tape Puncher
- CRT
- Printer (Impact, Ink Jet, Laser, Dot Matrix)
- Plotter
- Analog (voice)

Input Output Interface

- I/O devices **needs special communication links for interfacing them with the CPU.**

- The **purpose** of the communication link is **to resolve the differences between the computer and peripheral devices**.

Input Output Interface

- ❑The **major differences** are:-

  (1) Peripherals – **Electromechanical and electromagnetic Devices**

  CPU or Memory - **Electronic Device**

  Therefore a **conversion of signal values may be required**.

  (2) Data Transfer Rate

  - Peripherals - **Usually slower**
  - CPU or Memory - **Usually faster than peripherals**

  Some kinds of **synchronization mechanism may be needed**

  (3) Unit of Information

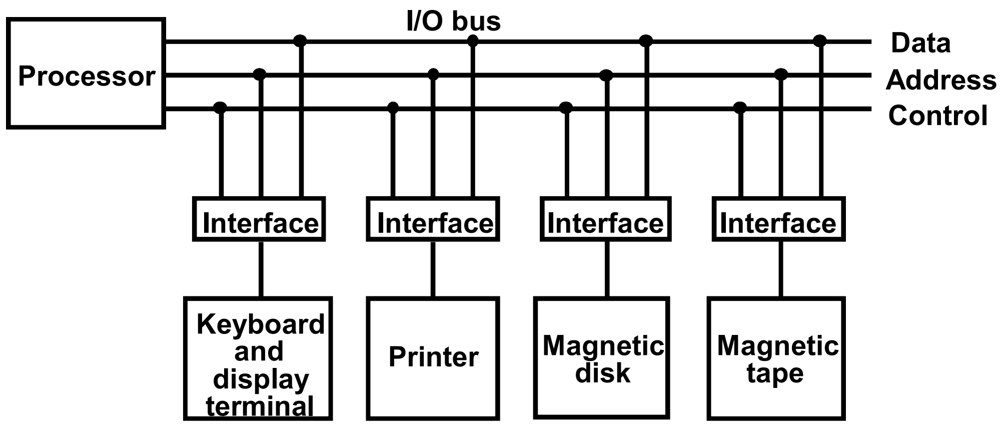
  - Peripherals – **Byte, Block**, …
  - CPU or Memory – **Word**

  (4) **Data representations may differ**.

  (5) **Operating mode of peripherals differ from each other** and **each must be controlled** so as **not to disturb the operation of others peripherals** connected to the CPU.

- ❑ **To resolve these differences**, computer system **includes special hardware components** between CPU and I/O devices **to supervise and synchronize all input and output transfers**. These components are called **interface units**.

Input Output Interface



Each peripheral has an interface module associated with it to do the following:

- **Decodes the device address (device code)**
- **Decodes the commands (operation)**
- **Provides signals for the peripheral controller**
- **Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory**

Typical I/O instruction

Op. code	Device address	Function code
		(Command)

Input Output Bus and Commands

- The **I/O bus from the processor** is **attached to** all the **I/O devices interface**.
- **To communicate** with a particular device, the **processor puts a device address** on the **address line**.
- **When interface detects** its **own address**, it **activates the path between the bus line and the device** that it controls.
- **At the same time** that the **address is made available in the address lines**, the **processor provides a function code in the control lines**.
- The **interface selected respond to** the **function code** and **proceeds to execute** it.
- The **function code** is **referred** to **as an I/O command**.
- There are **four types of command that an interface may receive**.
    - **Control**
    - **Status**
    - **Data Output**
    - **Data Input**

Input Output Commands

- **I/O Command** is an **instruction** that is **executed in** the **interface** and its **attached peripheral units.**

- **Control command** : It is issued to **activate peripheral** and to **inform what to do**

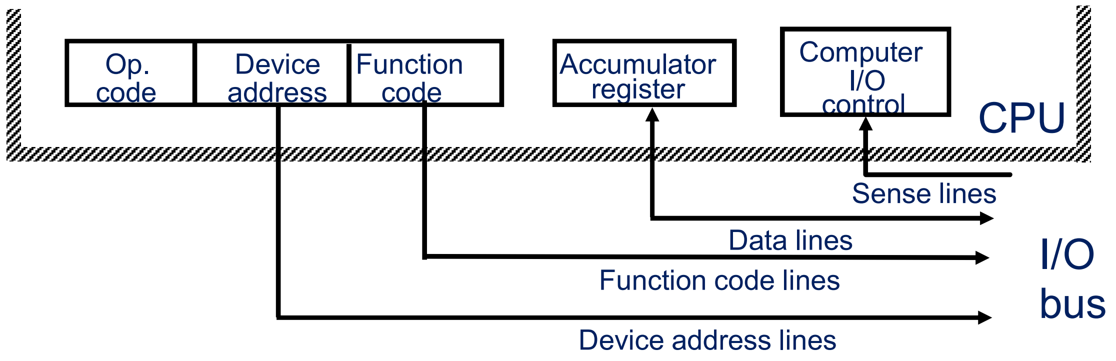
- **Status command :** It is used to **test various status condition** in the interface and the peripherals

- **Data o/p command** : It **causes the interface to respond by transferring data from the bus into one of its registers**

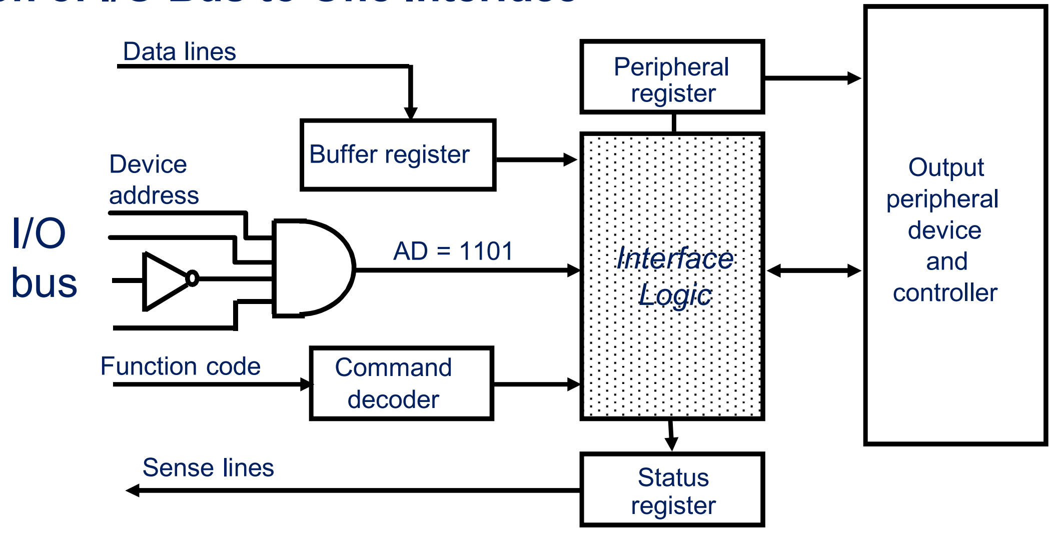
- **Data i/p command** : It **causes the interface to receive an item of data from the peripheral and places it in its buffer register**.

Connection of I/O Bus

Connection of I/O Bus to CPU



Connection of I/O Bus to One Interface



Input Output Bus and Memory Bus

- In addition to communicating with I/O, the processor must communicate with the memory unit.
- Like the memory bus, I/O bus contains data, address, and read/write control lines.
- Functions of Buses
  - MEMORY BUS is for information transfers between CPU and the Main Memory
  - I/O BUS is for information transfers between CPU and I/O devices through their I/O interface

Input Output Bus and Memory Bus

- **Input Output Bus**
  - Communication between CPU and all interface units is via a common I/O Bus
  - An interface connected to a peripheral device may have a number of data registers , a control register, and a status register
  - A command is passed to the peripheral by sending to the appropriate interface register
  - Transfer of data, control, and status information is always via the common I/O Bus

Input Output Bus and Memory Bus

There are three ways that computer buses can be used to communicate with memory and I/O:

- ❑Use two separate buses
  - one to communicate with memory and
  - the other with I/O interfaces
- ❑Use one common bus but both memory and I/O but have separate control lines for each.
- ❑Use one common bus with common control lines for both functions

Isolated and Memory Mapped I/O

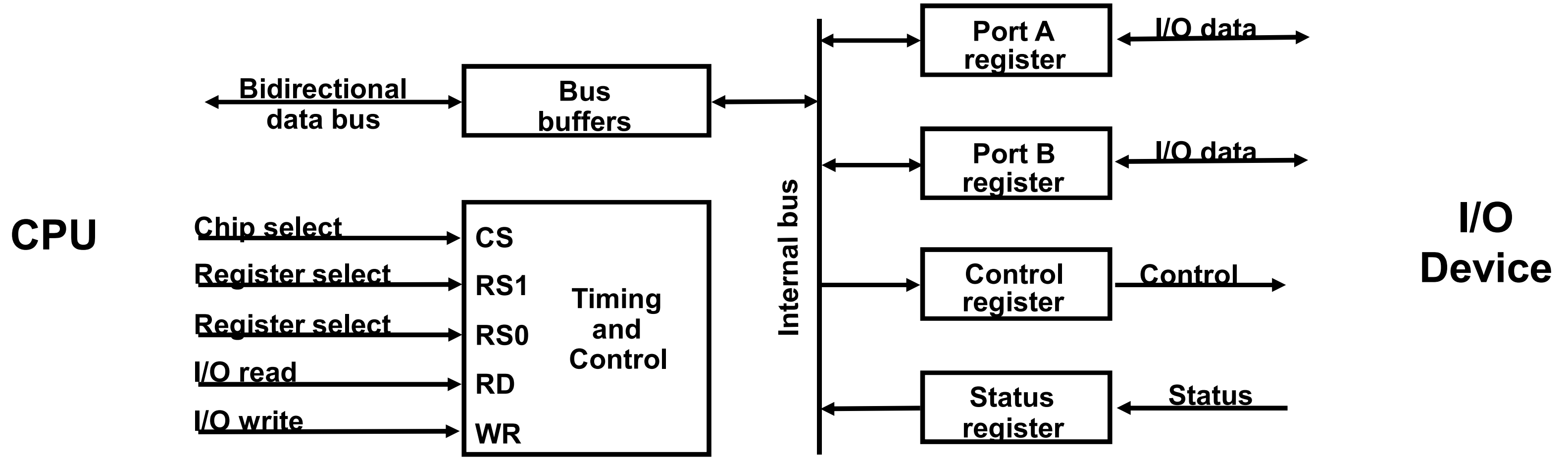
Isolated I/O

- Separate I/O read/write control lines in addition to memory read/write control lines
- Separate (isolated) memory and I/O address spaces
- Distinct input and output instructions

Memory-mapped I/O

- A single set of read/write control lines
  ->no distinction between memory and I/O transfer
- Memory and I/O addresses share the common address space
  -> reduces memory address range available
- No specific input or output instruction
  -> The same memory reference instructions can be used for I/O transfers
- Considerable flexibility in handling I/O operations

Input Output Interface and Ports



CS	RS1	RS0	Register selected
0	x	x	None - data bus in high-impedence
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

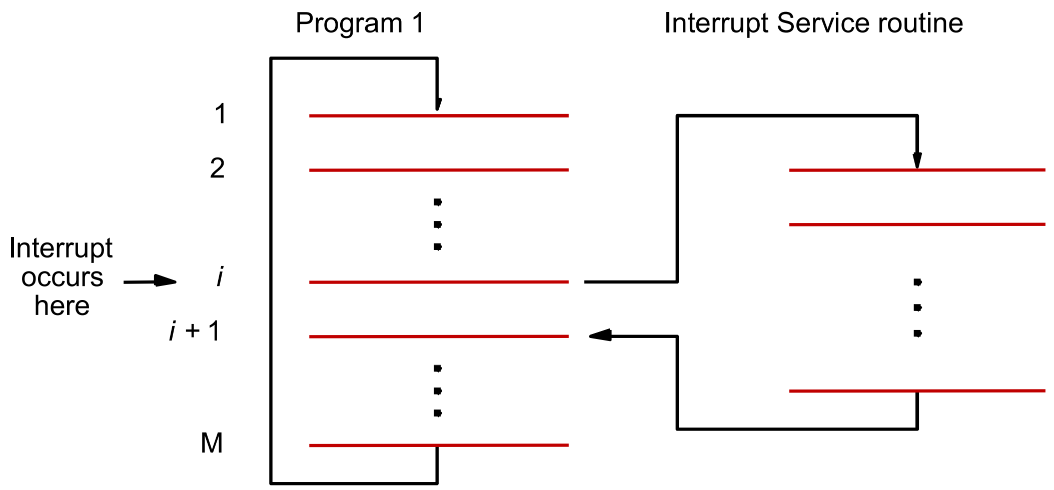
Programmable Interface

- Information in each port can be assigned a meaning depending on the mode of operation of the I/O device
  → Port A = Input Data; Port B = Output Data;
- CPU initializes(loads) each port by transferring a byte to the Control Register
  → Allows CPU to define the mode of operation of each port
  → *Programmable Port*: By changing the bits in the control register, it is possible to change the interface characteristics

Interrupts

- Interrupt is **a signal emitted by hardware or software when** a **process or an event needs immediate attention**.
- It is a **change in program defined flow of execution.**
- It **alerts the processor** to a **high priority process requiring interruption** of the current working process.
- **When** an **interrupt occurs**, the **hardware executes the instructions at a specified address** instead of following the normal program flow.
- **Atleast one** of the **bus control lines** called an **Interrupt request line** is usually **dedicated for this purpose**.
- The **routine executed in response to an interrupt request** is called the **Interrupt Service routine.**

Interrupts



- **Processor** is **executing** the **instruction** located at **address i when an interrupt occurs**.
- **Routine executed** in **response to an interrupt request** is called the **interrupt-service routine.**
- **When** an **interrupt occurs**, **control** must be **transferred to** the **interrupt service routine.**
- But **before transferring control**, the current **contents of the PC (i+1**), **must be saved** in a known location.
- **This** will **enable** the **return-from-interrupt instruction** to **resume execution at i+1**.
- **Return address**, or the **contents of the PC** are **usually stored** on the **processor stack**.

Types of Interrupts

- ❑External Interrupts
- ❑Internal Interrupts
- ❑Software Interrupts

External Interrupts

- ❑**Initiated** from **outside of CPU and Memory**
- ❑It **can come from**
  - **Input-Output devices**
  - **timing device**
  - **circuit monitoring the power supply**
  - **Any other external source**
- ❑Some **examples** include

  - I/O Device → Data transfer request or Data transfer complete

  - Timing Device → Timeout

  - Power Failure

Internal Interrupts

- Also known as **Traps**
- **Caused by** the **currently running program**
- It **arises from illegal or erroneous use** of an instruction or data.
- The service program(**ISR**) **determines the corrective measures** to be taken
- Some **examples** include
  - Register/Stack Overflow
  - Divide by zero
  - Invalid operation code
  - Protection Violation

Software Interrupts

- Both **External** and **Internal Interrupts** are **initiated by** the **computer Hardware**.
- **Software Interrupt** is **initiated by executing** an **instruction**.
- It is **a special call instruction** that **behaves like an interrupt rather than a subroutine call**.
- Used to transfer control to the operating system
- Supervisor Call → Switching from a user mode to the supervisor mode
- Allows to execute a certain class of operations which are not allowed in the user mode
- Example
  - Complex I/O transfer operation

Masking Interrupts

- Some interrupts can be temporarily disabled.
- Most of the processors can disable external interrupts.
- **Most internal interrupts cannot be disabled**.
- It is generally **problematic to disable the interrupts** for a long period of time.

Interrupts

- Saving and restoring information can be done automatically by the processor or explicitly by program instructions.
- Saving and restoring registers involves memory transfers:
  - Increases the total execution time.
  - Increases the delay between the time an interrupt request is received, and the start of execution of the interrupt-service routine. This delay is called interrupt latency.
- In order to reduce the interrupt latency, most processors save only the minimal amount of information:
  - This minimal amount of information includes **Program Counter** and **processor status registers**.
- Any additional information that must be saved, must be saved explicitly by the program instructions at the beginning of the interrupt service routine.

Interrupts

- When a processor receives an interrupt-request, it must branch to the interrupt service routine.
- It must also inform the device that it has recognized the interrupt request.
- This can be accomplished in two ways:
  - Some processors have an **explicit interrupt-acknowledge control signal** for this purpose.
  - In other cases, the **data transfer** that takes place between the device and the processor can be used to **inform the device**.

Interrupts

- Interrupt-requests interrupt the execution of a program, and may alter the intended sequence of events:
  - Sometimes such alterations may be undesirable, and must not be allowed.
  - For example, the processor may not want to be interrupted by the same device while executing its interrupt-service routine.
- Processors generally provide the ability to enable and disable such interruptions as desired.
- One simple way is to provide machine instructions such as *Interrupt-enable* and *Interrupt-disable* for this purpose.
- To avoid interruption by the same device during the execution of an interrupt service routine:
  - First instruction of an interrupt service routine can be Interrupt-disable.
  - Last instruction of an interrupt service routine can be Interrupt-enable.

Interrupts

- Multiple I/O devices may be connected to the processor and the memory via a bus. Some or all of these devices may be capable of generating interrupt requests.
  - Each device operates independently, and hence no definite order can be imposed on how the devices generate interrupt requests?
- How does the processor know which device has generated an interrupt?
- How does the processor know which interrupt service routine needs to be executed?
- When the processor is executing an interrupt service routine for one device, can other device interrupt the processor?
- If two interrupt-requests are received simultaneously, then how to break the tie?

Interrupts

- Consider a simple arrangement where all devices send their interrupt-requests over a single control line in the bus.
- When the processor receives an interrupt request over this control line, how does it know which device is requesting an interrupt?
- This information is available in the status register of the device requesting an interrupt:
  - The status register of each device has an *IRQ* bit which it sets to 1 when it requests an interrupt.
- Interrupt service routine can poll the I/O devices connected to the bus. The first device with *IRQ* equal to 1 is the one that is serviced.
- Polling mechanism is easy, but time consuming to query the status bits of all the I/O devices connected to the bus.

Interrupts

- The device requesting an interrupt may identify itself directly to the processor.
  - Device can do so by sending a special code (4 to 8 bits) to the processor over the bus.
  - Code supplied by the device may represent a part of the starting address of the interrupt-service routine.
  - The remainder of the starting address is obtained by the processor based on other information such as the range of memory addresses where interrupt service routines are located.
- Usually the location pointed to by the interrupting device is used to store the starting address of the interrupt-service routine.

Interrupts

- Previously, before the processor started executing the interrupt service routine for a device, it disabled the interrupts from the device.
- In general, same arrangement is used when multiple devices can send interrupt requests to the processor.
  - During the execution of an interrupt service routine of device, the processor does not accept interrupt requests from any other device.
  - Since the interrupt service routines are usually short, the delay that this causes is generally acceptable.
- However, for certain devices this delay may not be acceptable.
  - Which devices can be allowed to interrupt a processor when it is executing an interrupt service routine of another device?

Interrupts

- I/O devices are organized in a priority structure:
  - An interrupt request from a high-priority device is accepted while the processor is executing the interrupt service routine of a low priority device.
- A priority level is assigned to a processor that can be changed under program control.
  - Priority level of a processor is the priority of the program that is currently being executed.
  - When the processor starts executing the interrupt service routine of a device, its priority is raised to that of the device.
  - If the device sending an interrupt request has a higher priority than the processor, the processor accepts the interrupt request.

Interrupts

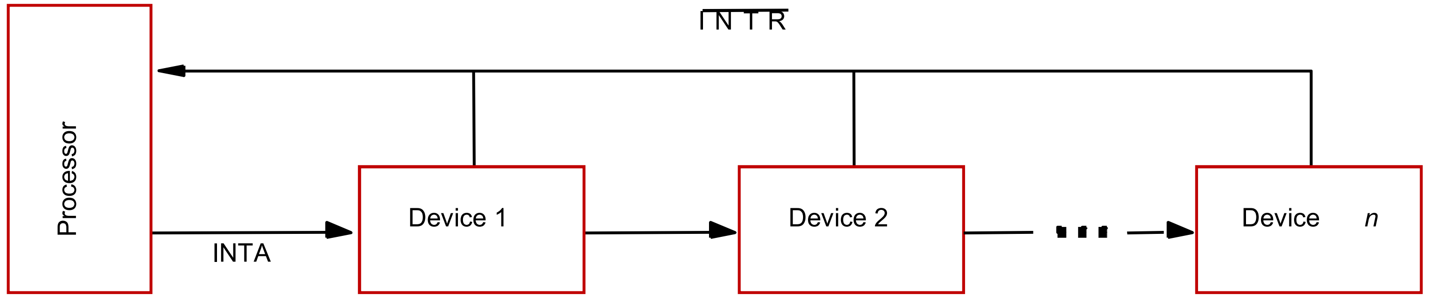
- Processor’s priority is encoded in a few bits of the processor status register.
  - Priority can be changed by instructions that write into the processor status register.
  - Usually, these are privileged instructions, or instructions that can be executed only in the supervisor mode.
  - Privileged instructions cannot be executed in the user mode.
  - Prevents a user program from accidentally or intentionally changing the priority of the processor.
- If there is an attempt to execute a privileged instruction in the user mode, it causes a special type of interrupt called as privilege exception.

Interrupts

- If multiple devices have generated the interrupt, then to break the tie one of the following method can be used.
  - Daisy Chaining.
  - Polling
  - Priority Arbitration
  - Hybrid

Interrupts

Daisy chain scheme:



- ❑ Devices are connected to form a daisy chain.
- ❑ Devices share the interrupt-request line, and interrupt-acknowledge line is connected to form a daisy chain.
- ❑ When devices raise an interrupt request, the interrupt-request line is activated.
- ❑ The processor in response activates interrupt-acknowledge Received by device 1, if device 1 does not need service, it passes the signal to device 2.
- ❑ Device that is electrically closest to the processor has the highest priority.

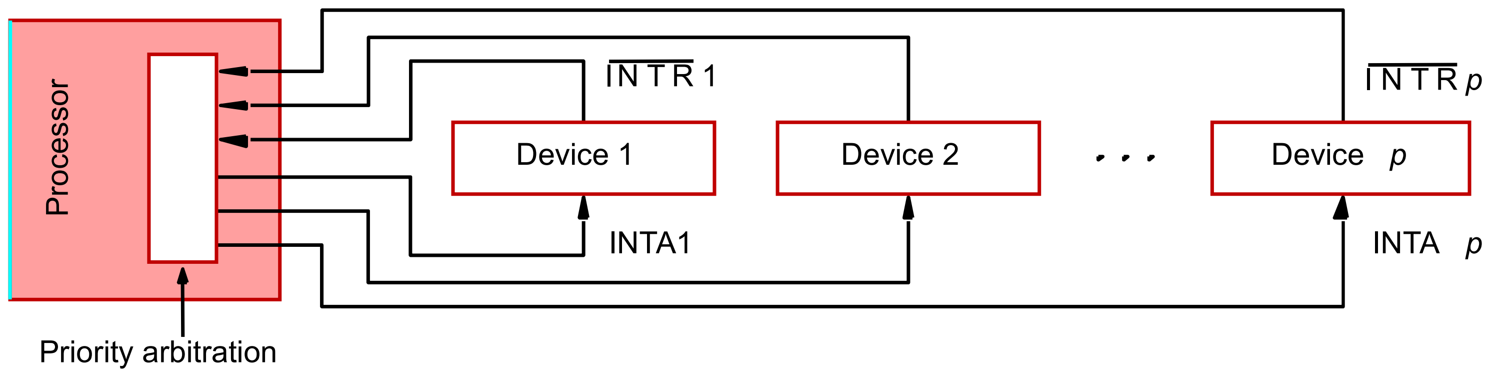
Interrupts

Polling scheme:

- The processor uses a polling mechanism to poll the status registers of I/O devices to determine which device is requesting an interrupt.
- In this case the priority is determined by the order in which the devices are polled.
- The first device with status bit set to 1 is the device whose interrupt request is accepted.

Interrupts

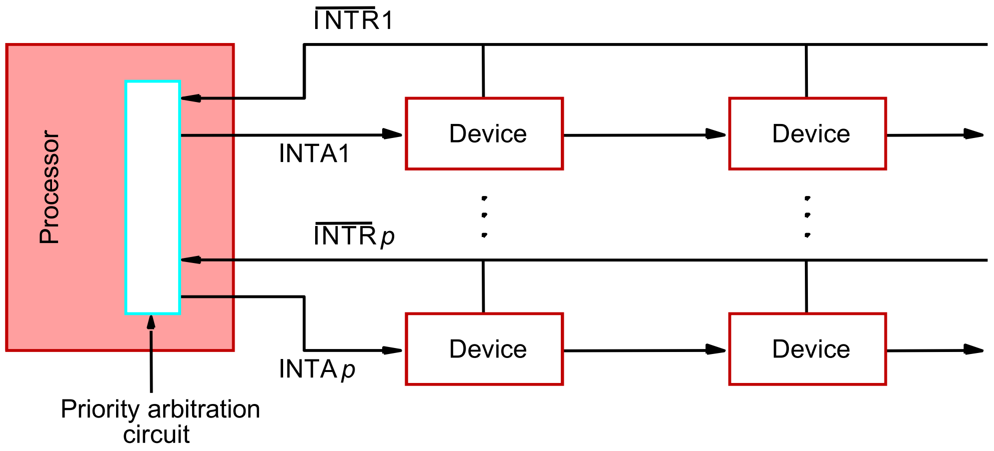
Priority Arbitration scheme:



- ❑Each device has a separate interrupt-request and interrupt-acknowledge line.
- ❑Each interrupt-request line is assigned a different priority level.
- ❑Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor.
- ❑If the interrupt request has a higher priority level than the priority of the processor, then the request is accepted.

Interrupts

- ❑ When I/O devices were organized into a priority structure, each device had its own interrupt-request and interrupt-acknowledge line.
- ❑ When I/O devices were organized in a daisy chain fashion, the devices shared an interrupt-request line, and the interrupt-acknowledge propagated through the devices.
- ❑ A combination of priority structure and daisy chain scheme can also used.


$\overline{\text{INTR}}1$
Processor
INTA1
Device
Device
$\overline{\text{INTR}}p$
Device
Device
INTAp
Priority arbitration circuit

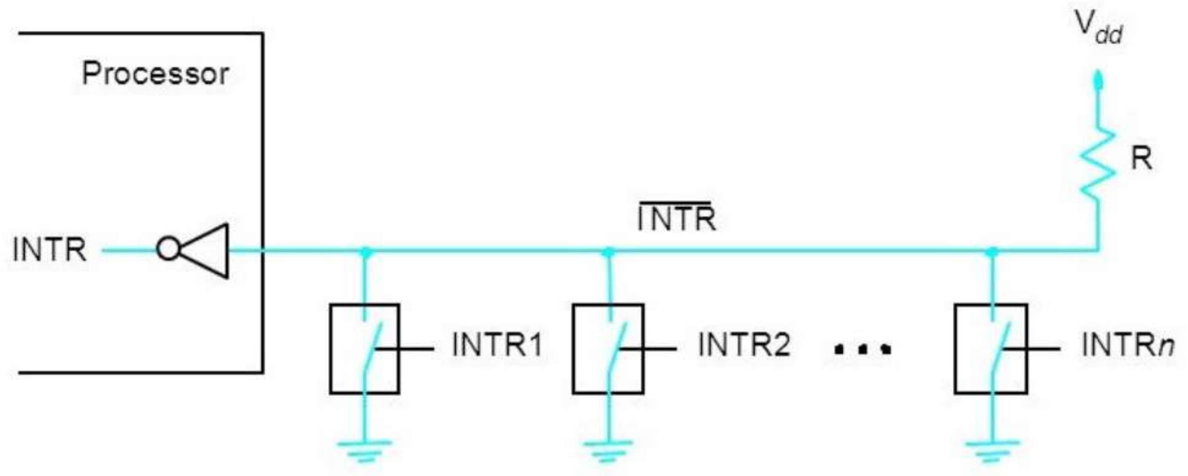
- ❑ Devices are organized into groups.
- ❑ Each group is assigned a different priority level.
- ❑ All the devices within a single group share an interrupt-request line, and are connected to form a daisy chain.

Interrupts

- Only those devices that are being used in a program should be allowed to generate interrupt requests.
- To control which devices are allowed to generate interrupt requests, the interface circuit of each I/O device has an interrupt-enable bit.
  - If the interrupt-enable bit in the device interface is set to 1, then the device is allowed to generate an interrupt-request.
- Interrupt-enable bit in the device’s interface circuit determines whether the device is allowed to generate an interrupt request.
- Interrupt-enable bit in the processor status register or the priority structure of the interrupts determines whether a given interrupt will be accepted.

Interrupts Hardware

- ❑ Many computers have facility to connect two or more input and output devices to it like laptop may have 3 USB slots.
- ❑ So there is a common interrupt line for all N input/output devices
- ❑ All these input and output devices are connected via switches as shown below



Interrupts Hardware

- When no interrupt is issued by the input/output devices then all the switches are open and the entire voltage from Vdd is flown through the single line INTR and reaches the processor which means the processor gets a voltage of 1V.
- When the interrupt is issued by the input/output devices then the switch associated with the input/output device is closed, so the entire current now passes via the switches which means the hardware line reaching the processor i.e INTR line gets 0 voltage. This is an indication for the processor that an interrupt has occurred and the processor needs to identify which input/output device has triggered the interrupt
- The value of INTR is a logical OR of the requests from individual devices.
- The resistor R is called as a pull up resistor because it pulls the line voltage to high voltage state when all switches are open(no interrupt state).

Exceptions

- The term exception is used to refer to any event that causes an interruption.
  - Interrupt-requests from I/O devices is one type of an exception.
- Other types of exceptions are:
  - Recovery from errors
  - Debugging
  - Privilege exception

Exceptions

- Many sources of errors in a processor. For example:
  - Error in the data stored.
  - Error during the execution of an instruction.
- When such errors are detected, exception processing is initiated.
  - Processor takes the same steps as in the case of I/O interrupt-request.
  - It suspends the execution of the current program, and starts executing an exception-service routine.
- Difference between handling I/O interrupt-request and handling exceptions due to errors:
  - In case of I/O interrupt-request, the processor usually completes the execution of an instruction in progress before branching to the interrupt-service routine.
  - In case of exception processing however, the execution of an instruction in progress usually cannot be completed.

Exceptions

- Debugger uses exceptions to provide important features:
  - Trace,
  - Breakpoints.
- Trace mode:
  - Exception occurs after the execution of every instruction.
  - Debugging program enables the user to examine the contents of registers, memory location and so on.
- Breakpoints:
  - Exception occurs only at specific points selected by the user.
  - Trap/Software interrupt is usually provided for this purpose
  - Debugging program is used as the exception-service routine.

Exceptions

- ❑ To protect the operating system of a computer from being corrupted by user programs, certain instructions can be executed only the processor is in supervisor mode. These are called privileged instructions.
- ❑When the processor is running in user mode, it will not execute an instruction that changes the priority level of processor or area in computer memory which is not allocated to the program.
- ❑An attempt to execute such an instruction will produce privilege exception.
- ❑It causes the processor to switch to supervisor mode and begin executing an appropriate routine in operating system.