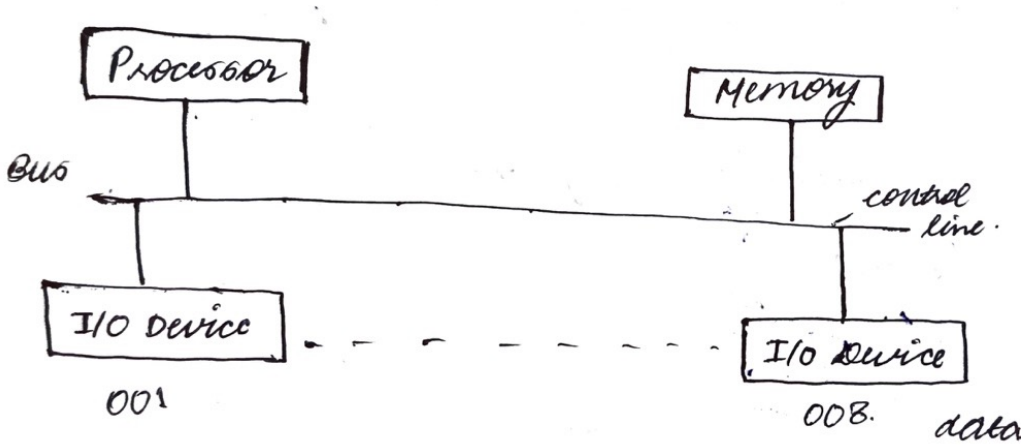
UNIT 3 Input-Output Organization

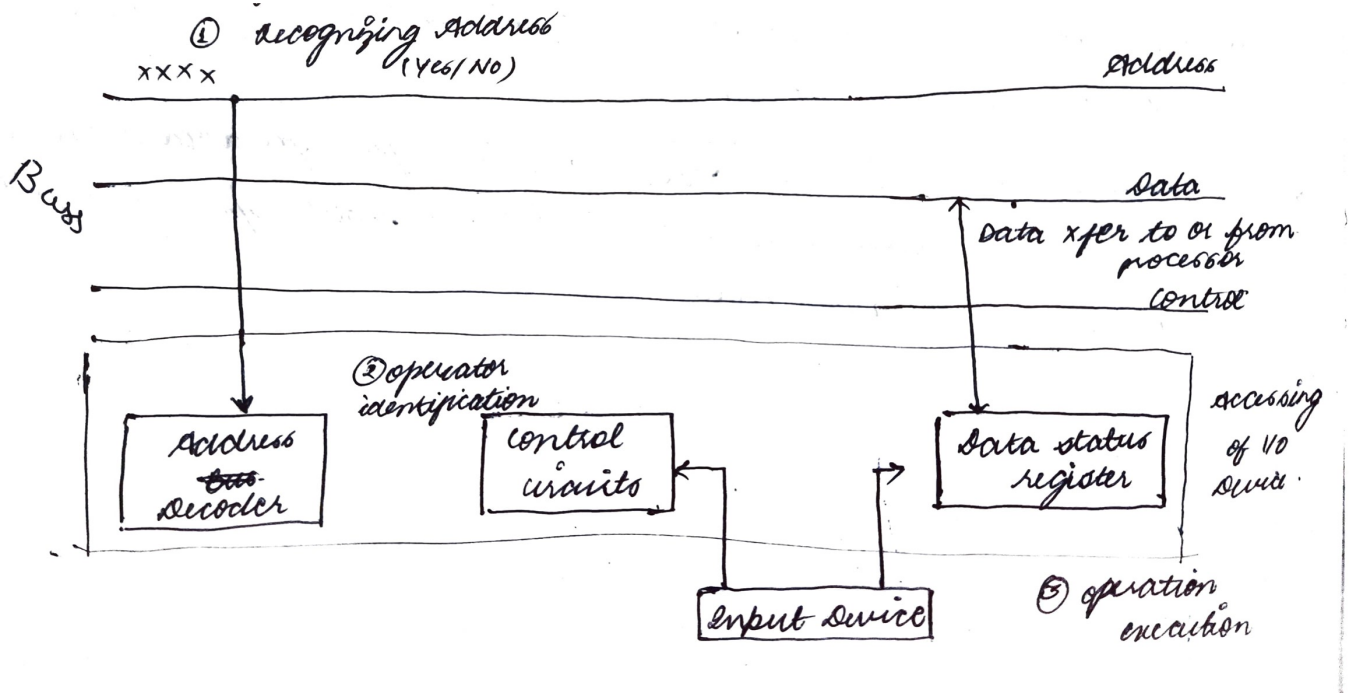
Single bus structure



- The bus is used to enable all the devices connected to it for information exchange.
- The information transmitting over the bus maybe an address, data and control signal.
- The processor use bus as per the instructions specified from the word address.
- Each i/o device is assigned with a unique address when processor put this address on the address line, the device recognises it and respond to the command.
- The processor request either read or write operation letter data is transferred with the help of data lines. When a bus is used as both the data line and address line then it is known as the Multiplex bus.

Memory Mapped I/O Device

- When I/O devices and the memory of processor share the same address space and instructions then the arrangement is called as memory mapped I/O.
- With memory mapped I/O arrangement any machine instructions that can access memory can also be used to transfer data to or from I/O devices.



Difference between Memory Mapped I/O and Port mapped I/O

Memory Mapped I/O	Port mapped I/O
16 bit addressing mode is used (as of computer m/c)	Separate 8-bit addresses are used
More hardware in the processor	Separate hardware
More number of instructions	Limited instructions are available
Available I/O connections for interfacing: 2^{16}	Available I/O connections for interfacing: 2^{8}

Interrupts:

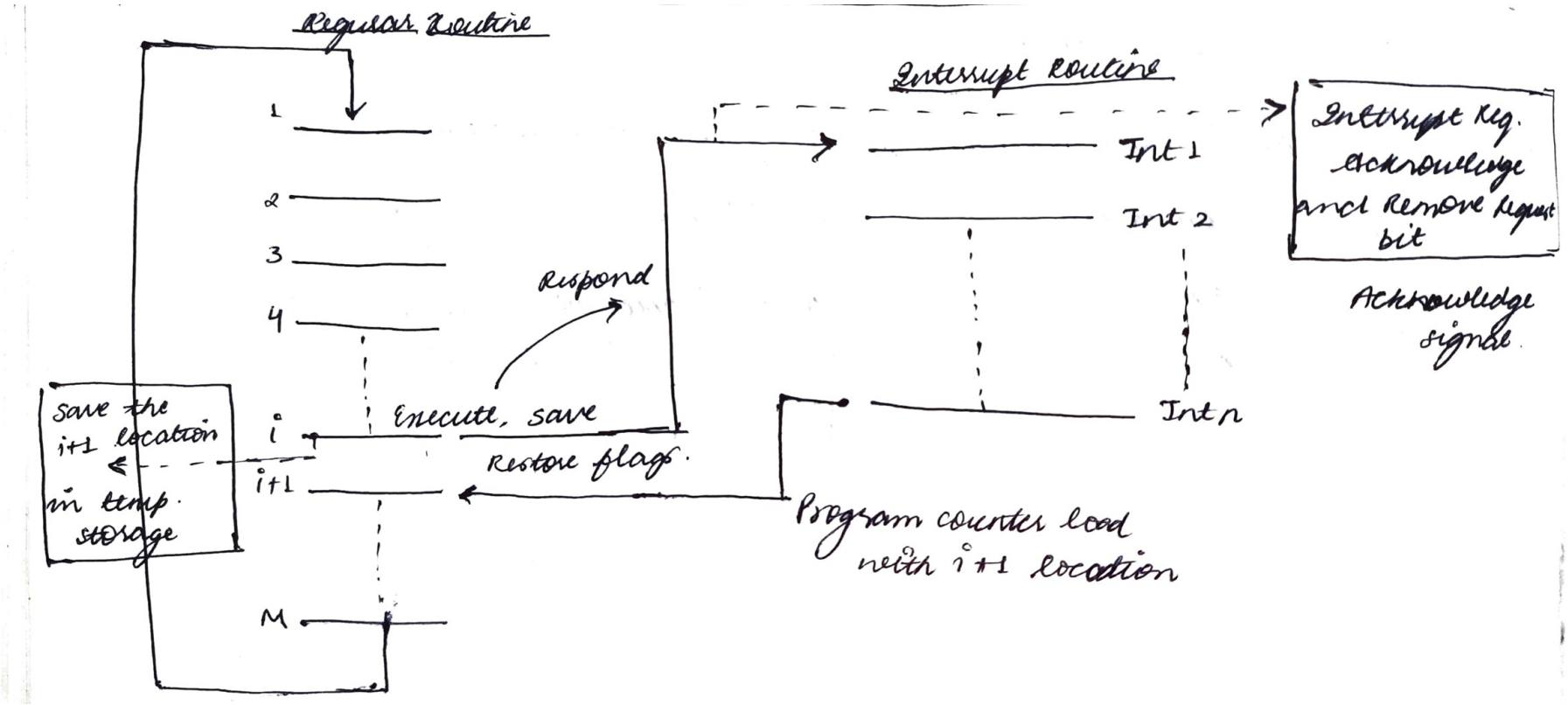
- Interrupt can be termed as an interfere request given to the processor to interrupt its ongoing process of execution (if allowed) so that processor can process all the processing in a timely manner.

- When the interrupt request is accepted by the processor, then the processor will suspend its current activity, save its state and execute a function called as interrupt handler or interrupt service routine.

- Mainly this interruption is often temporarily, allowing the processor to resume on its routine activity post finishing the interrupt handling function.

- The main function of the interrupt is to indicate any special electronic or physical state change that requires sensitive attention by the processor.

Control transfer for the use of interrupt:



Role of Interrupts:

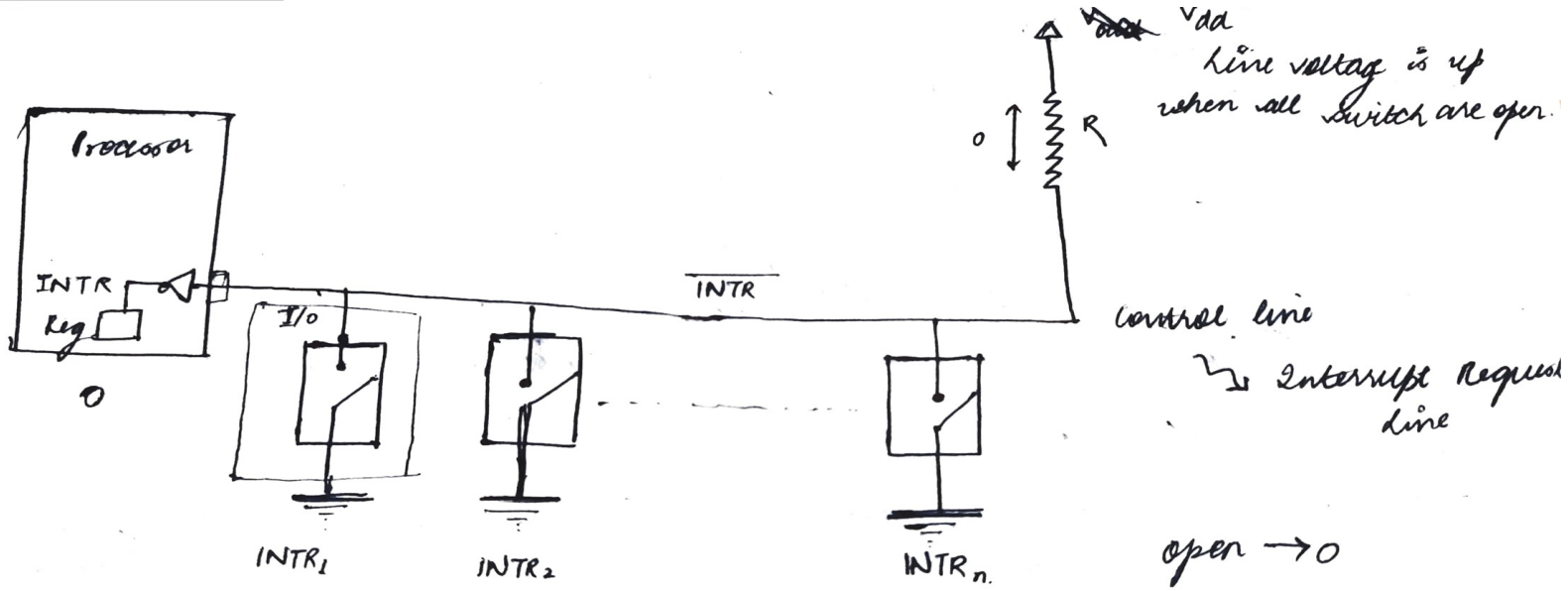
- To indicate physical or electronic changes occurred usually from I/O devices.
- The interrupt task is an asynchronous task and can occur at any time instant irrespective of regular program execution of the processor.
- Indication of any external event.
- Indication of completion of any task.
- Time allocation from CPU time for timely execution of all the activities.
- Indication of any abnormal event or Hazard.

Note: for better operation, at least one of the bus control line should be dedicated for interrupt request purpose, this bus line is called as interrupt request line.

Operation of Interrupts:

- Before handling or starting the interrupt routine, program process should execute, restore, and save the information.
- Information that needs to be saved and restored typically include the condition code flags and content of registers used.
- The saving and re-storing of information is done by the processor automatically or through program instructions.
- Modern processors save the minimum amount of information needed to maintain the integrity of the program execution.
- The processor should work, re-store, and save information such that overall execution time should be minimum.
- Typically, the data stored by the processor is program counter data and program register data in use.

Interrupt Hardware:



- When all switches are open

  INTR' = V_{dd} = Logic '1'
  INTR = '0'

- When all switches are closed

  INTR' = Logic '0'
  INTR = '1'

Working of Interrupt Hardware:

- Most computer have several interface I/O devices that can request an interrupt signal.
- Single control lone (interrupt request line) may be used to serve n devices for interrupt request.
- These I/O devices are connected in parallel combination with 1 terminal at same voltage (say V_1) another terminal on ground
- To request an interrupt signal, the devices need to close its associated switch. Causing the control line to drop its voltage from V_1 to 0.
- Making INTR or Logic '1'. When there is no interrupt request, all switches are open the voltage is brought back from 0 to V_1. This in known as pulling up line voltage.
- The register between V_{dd} and V_1 is known as pull up register.

Enabling and Disabling of Interrupts:

- A fundamental facility which should be in all the computers is the ability to enable and disable interruption. This is desired for better control over the program execution.

- The interrupt which has lower priority should be ignored for the interrupt request.

Example: Sometimes it is necessary to guarantee that a particular sequence of instruction is executed to the end without any interruption because ISR may change some of the data used by the instruction in question. For such reasons, enabling and disabling of interrupt must be available to the programmer.

Problem Associated with Interrupts:

- When device activate the INTR signal, it keeps the signal activated until its gets acknowledged that processor has accepted its request.

- The INTR signal will be active during the execution of ISR bit it should ensure that active request signal does not lead to successive interruption causing the system to certain infinite loop.

Solution:

Case 1: using ISR program

I. Processor hardware should ignore the INTR line until the execution of first instruction of the ISR has been completed.
II. Interrupt disable instruction can be used as the first instruction in the ISR. With this, the programmer can ensure that no further interruption will occur until interrupt enable instruction is executed.

Case 2: using processor register

I. This option is suitable for a simple processor with INTR line.
II. Processor should automatically disable interrupt before starting of the execution of ISR.
III. After saving the content of program counter and processor status register, the processor perform B equal and process of executing and interrupt disable function.
IV. Open one bit of Processor status registers used as interrupt enable which indicates whether interrupt request our enable or not.

Case3: using edge triggering

Handling Multiple Devices:

To understand the operation of interrupts few basic questions need to be understood.

1. Can the processor recognise the device requesting an interrupt?

2. Should a device be allowed to interrupt the processor while another interrupt is being solved?

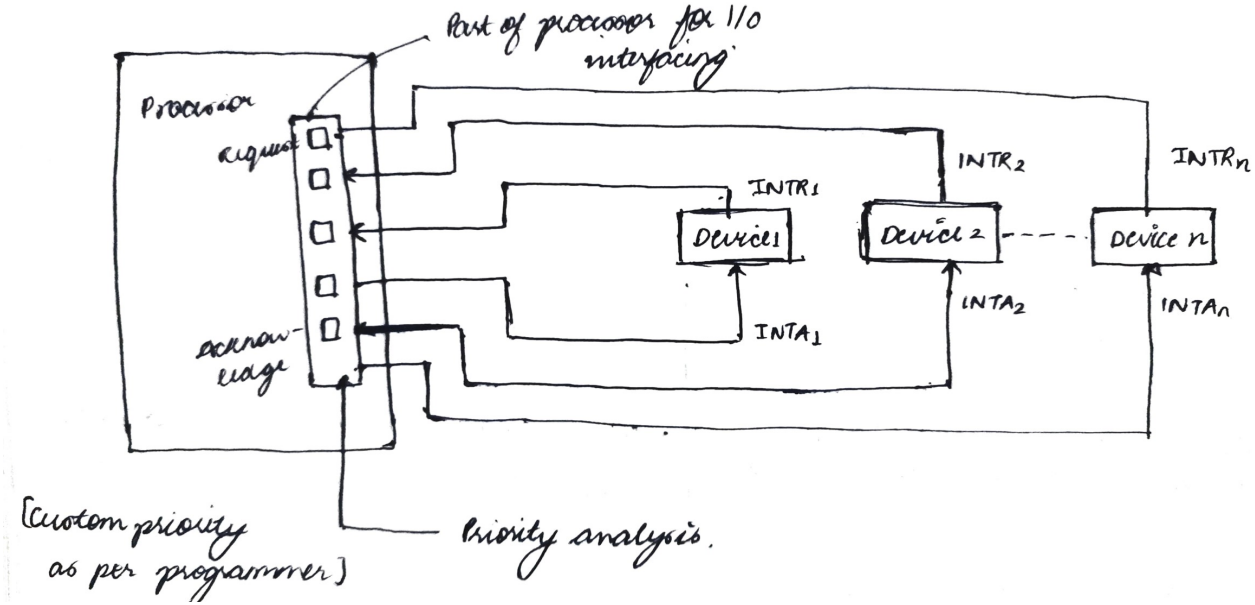
3. How two or more simultaneous interrupt request be handled?

Vectored interrupt	Polled Interrupt
1. I/O device raising request, also provide information of branch called vector	1. CPU regularly check all the devices continuously looking for service request flag
1. It does have an address along with its request	1. No addresses provided only information about whether device is available or not is there
1. These are a synchronous in nature and can occur at any point of time	1. CPU regularly check the device status and these are synchronous in nature
1. Last time of execution	1. Much time is wasted in identifying the address and regular status check
1. It is priority based activity	1. There is no priority in polling

Interrupt nesting:

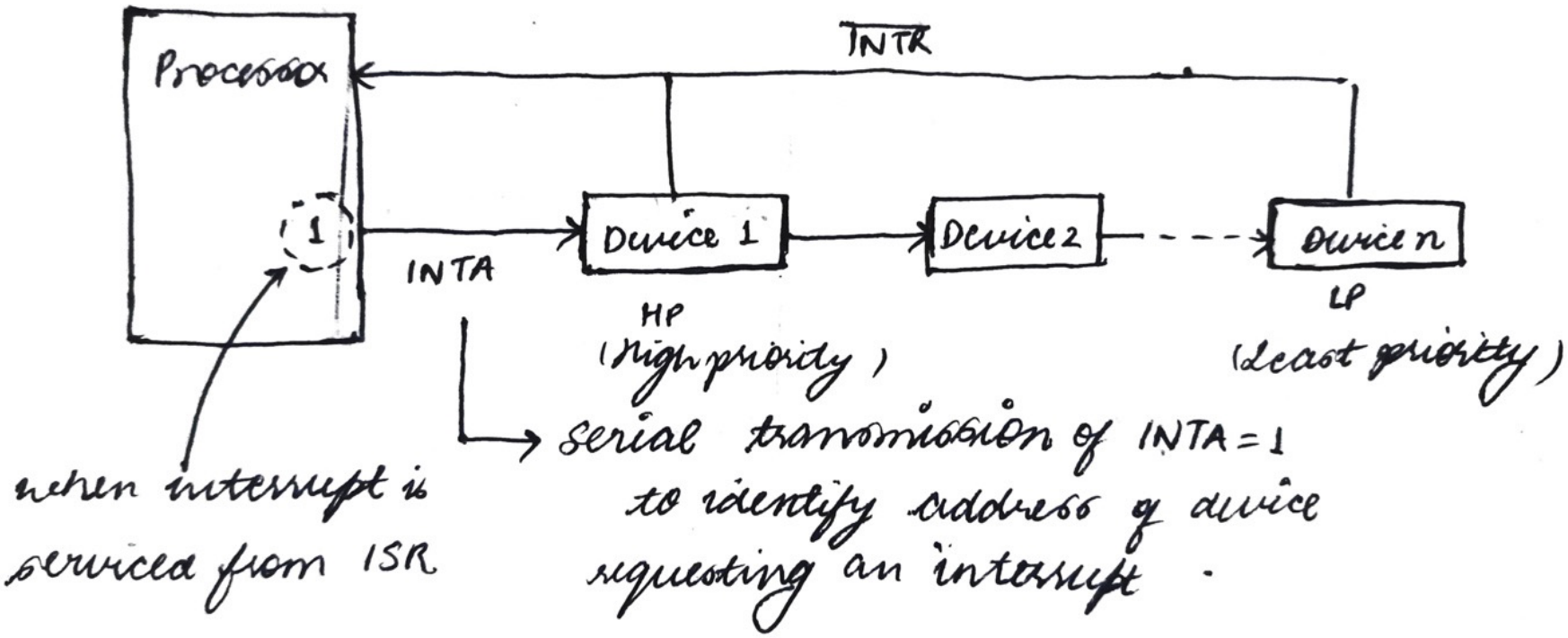
- It is a method of interconnecting I/O devices with the part of processor used for I/O interfacing.
- It is used to identify priority of any device at both hardware and software level

1. **Individual interrupt request and acknowledge line**



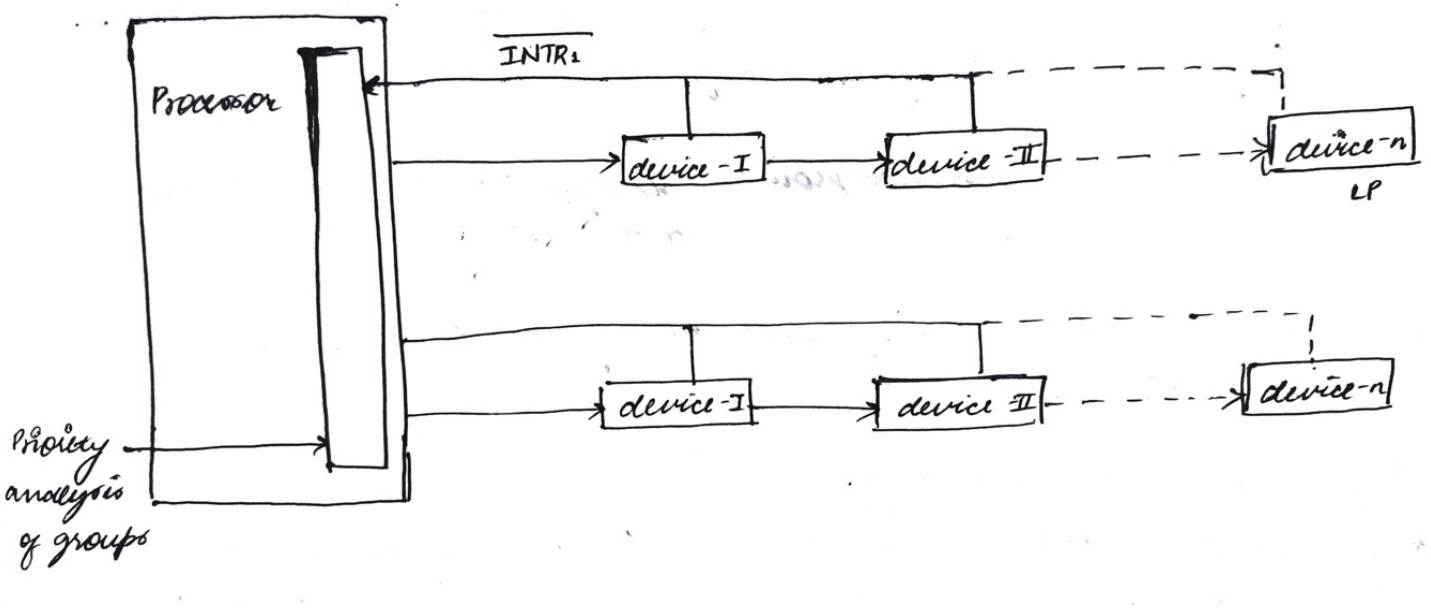
   - Customized priority based on program

2. Daisy chain



- Highest priority is given to the nearest device connected to the processor

3. Hybrid Method



- In this priority of multiple daisy chains used can be customized. But device-I will be always at higher priority compared to the other devices

Direct Memory Access

An instruction to transfer input/output data is executed only after the processor determines that the I/O device is ready.

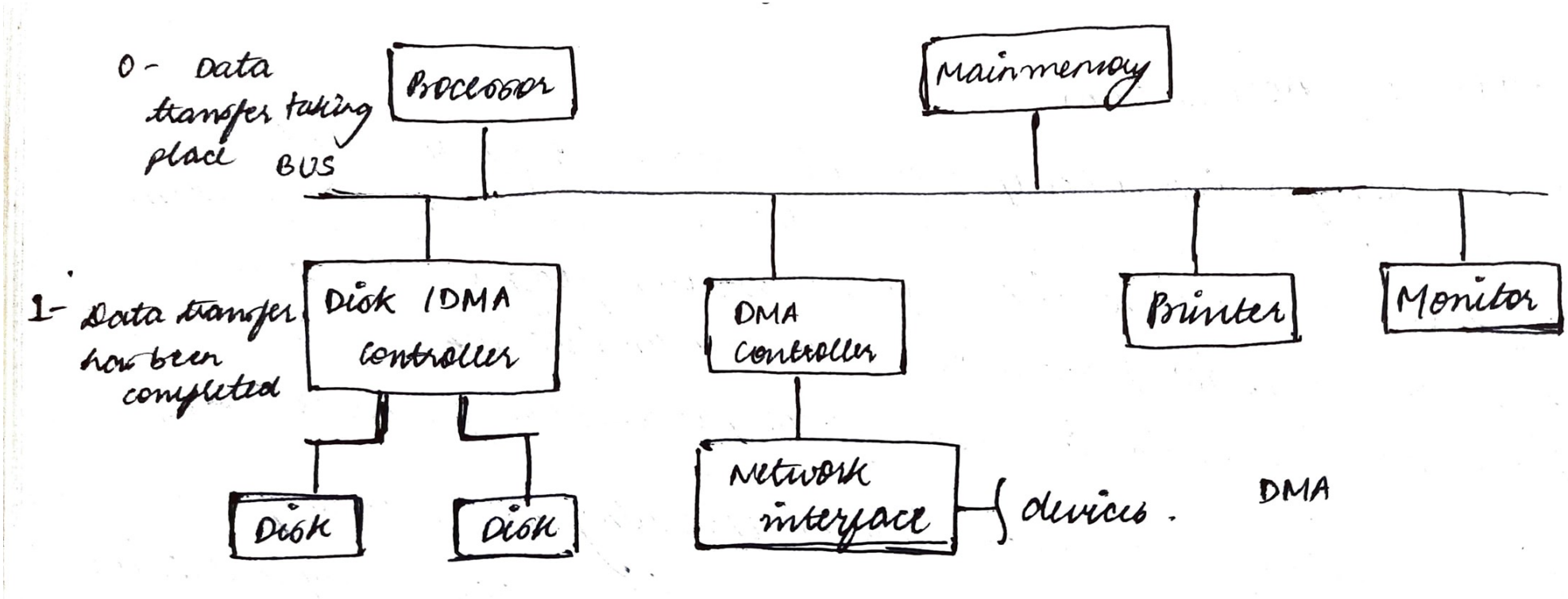
To perform this process, the processor either bowl a status plug in the device interface or wait for the device to send an interrupt request in both the cases considerable time of the processor is procured

In polling, much time is wasted on obtaining the status of devices.
In interrupt, much time is wasted during execution, saving, and restoring of conditional flags.

To transfer a large block of data at high speed and alternative approaches used by providing a special control unit to allow transfer of data directly between an external audio device and the main memory without continuous intervention by the processors this approach is known as **direct memory access (DMA)**

The controller circuit used for DMA transfer is called as **DMA Controller.**

Functioning of Direct Memory Access



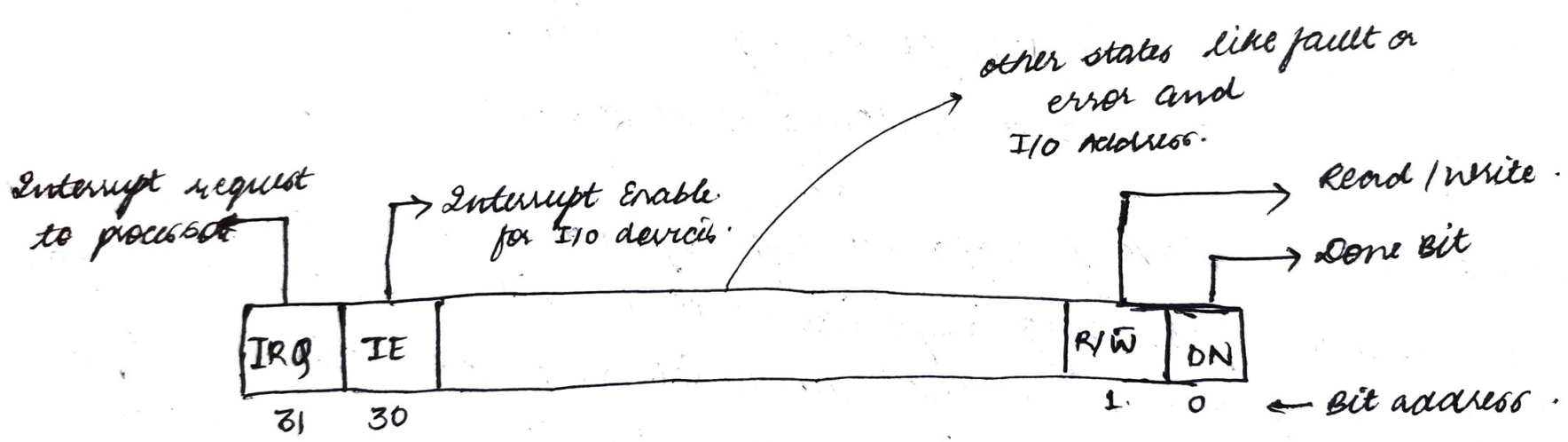
Functions of DMA Controller

- To perform operations that are normally performed by the processor only when accessing memory.
- For each transfer, it should provide memory address and all the bus signals that control data transfer.
- It should control and implement memory addresses on each transfer.
- Keep record of total number of transfers
- It should raise INTR to the processor for accessing the bus mastership when any I/O device is about to perform memory operations.

DMA Registers

There are a total of three registers used by the DMA controller

Two registers: for storing the starting address and the word count
One register: for providing the information regarding status and control flags



Status and control flags

R/W': 0 = write operation
R/W': 1 = Read operation

DN: 0 = Data Transfer is taking place
DN: 1 = Data Transfer has been completed

IE: 0 = I/O device interrupt request will be ignored
IE: 1 = I/O devices or allowed to raise interrupt request

IRQ: 0 = no request from DMA
IRQ: 1 = DMA controller raised request to processor

Bus Master:
The device that is allowed to initiate data transfer on the bus at any given time is called as bus master.

Bus Arbitration:
It is the process of granting the bus mastership to a device based on its priority is known as bus arbitration.

Arbitration is mainly classified into two category based on functioning

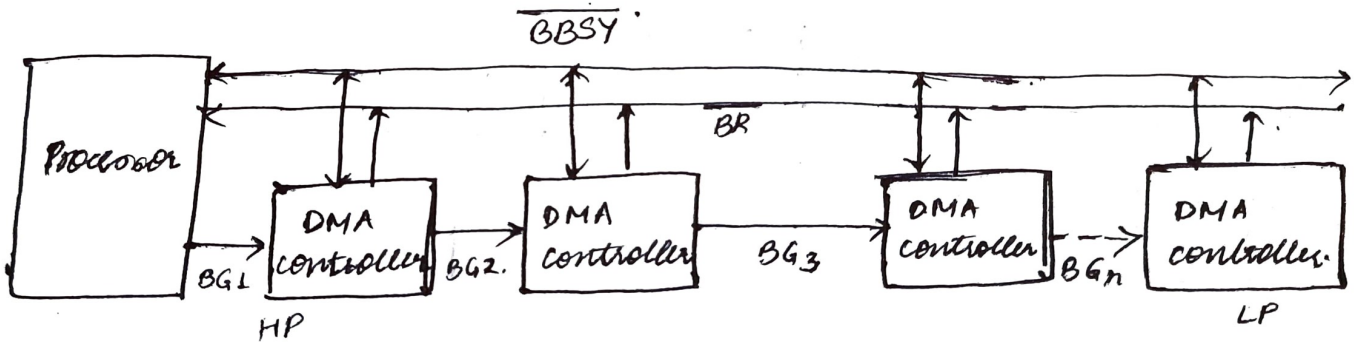
1. Centralized bus arbitration (Daisy chain arrangement of multiple DMA Controllers)
2. Distributed Arbitration (Individual Arrangement)

Centralized bus arbitration:

BBSY' : Bus Busy

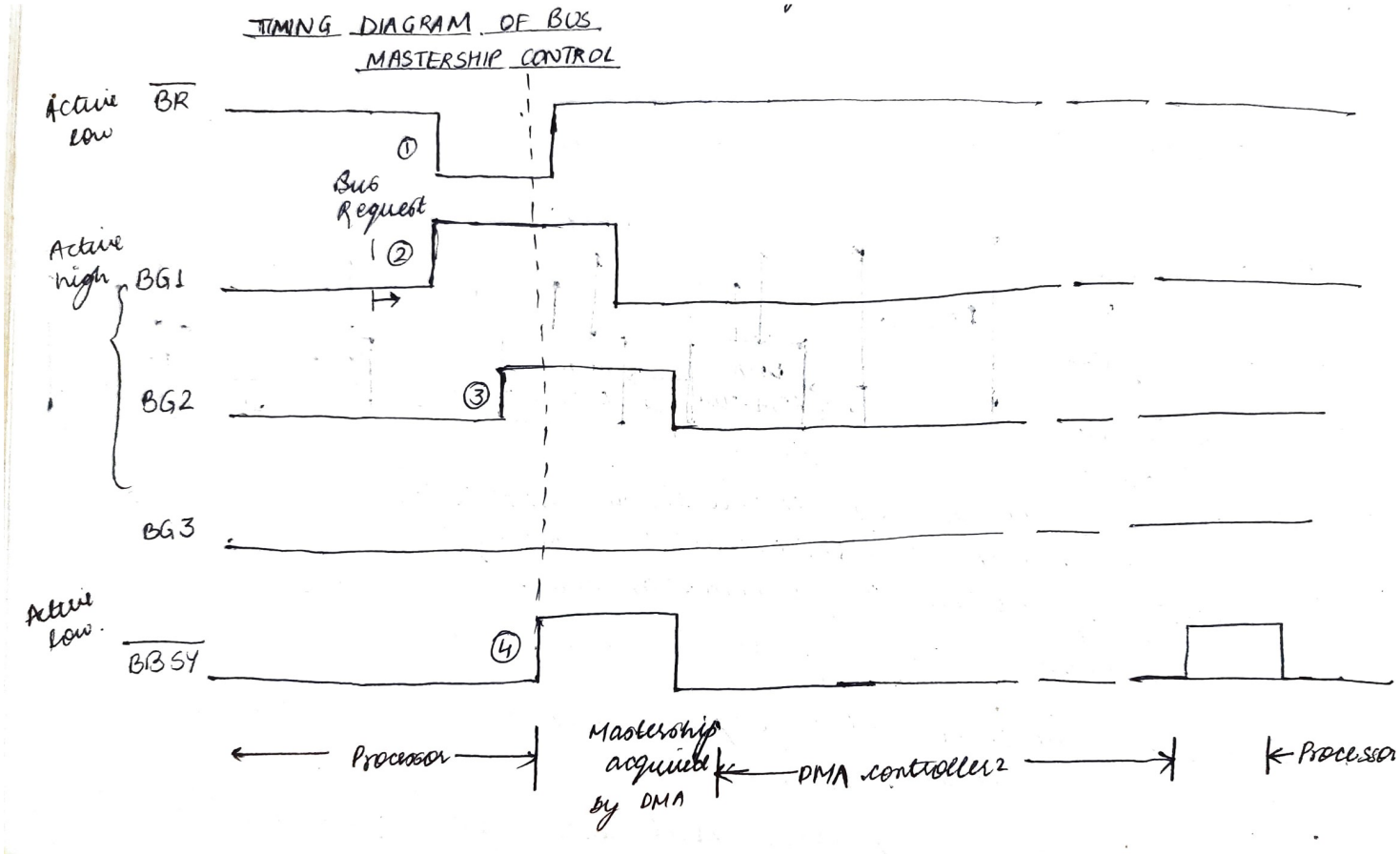
BR' : Bus Request

BG' = Bus Grant



- First the interrupt enable bit is made High so as the IO device says which are connected to the DM a can raise INTR for memory operations
- Once a DMA receive INTR from the I/O device the interrupt enable bit is made to 0
- IRQ is said to be high to request and interrupt from DMA controller to the processor
- The DMA also sends the information about starting address, word count, status register information and I/O device address
- Once the processor gives access of bus to DM a controller the memory transfer operation starts
- Post completion of data transfer that done but is made High and interrupt enable is said high

Timing Diagram of Bus Arbitration:



Distributed bus arbitration:

Interconnection Standards

- A typical desktop or notebook computer has several ports that can be used to connect I/O devices, such as a mouse, a memory key, or a disk drive.
- Standard interfaces have been developed to enable I/O devices to use interfaces that are independent of any particular processor.
- For example, a memory key that has a USB connector can be used with any computer that has a USB port.
- IEEE (Institute of Electrical and Electronics Engineers) develops these standards further and publishes them as IEEE Standards.

Universal Serial Bus (USB)

- Most widely used interconnection standard.
- large variety of devices are available with a USB connector, including mice, memory keys, disk drives, printers, cameras, and many more.
- The commercial success of the USB is due to its simplicity and low cost.
- The original USB specification supports two speeds of operation, called low-speed (1.5 Megabits/s) and full-speed (12 Megabits/s).
- Later, USB 2, called High-Speed USB, was introduced. It enables data transfers at speeds up to 480 Megabits/s. As I/O devices continued to evolve with even higher speed requirements, USB 3 (called Superspeed) was developed. It supports data transfer rates up to 5 Gigabits/s.

Key objectives:

- Provide a simple, low-cost, and easy to use interconnection system
- Accommodate a wide range of I/O devices and bit rates, including Internet connections, and audio and video applications
- Enhance user convenience through a “plug-and-play” mode of operation

USB Device Characteristics

- **Plug-and-Play:** USB devices can be connected or disconnected from a computer without the need to restart the system. This feature allows for easy and convenient device connectivity.

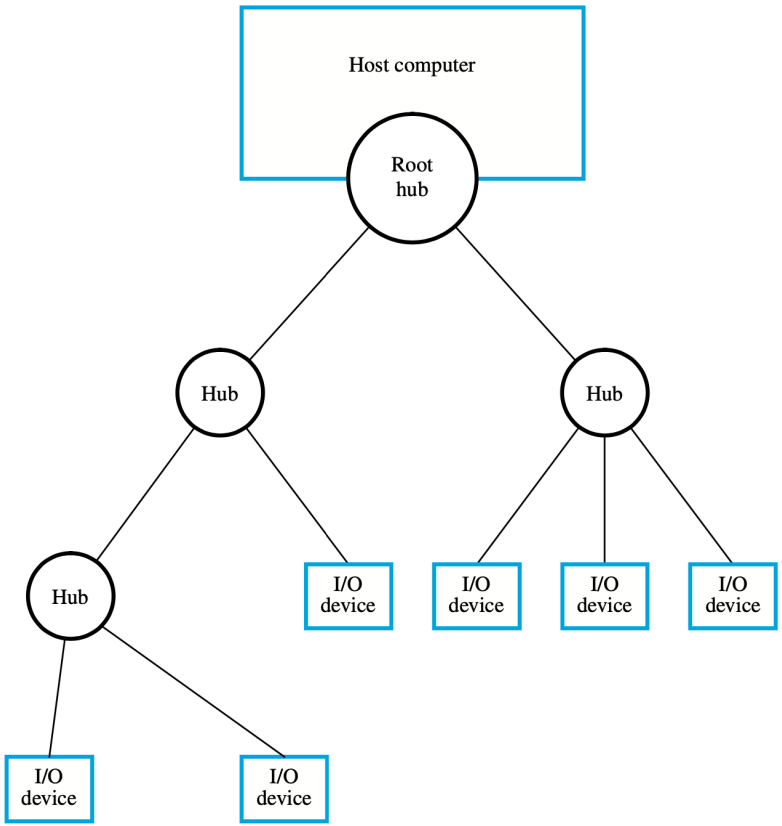
- **Data Transfer Speed:** USB supports various data transfer speeds, which have evolved over time. The different versions of USB include USB 1.0, USB 2.0, USB 3.0, USB 3.1, USB 3.2, and USB 4.0. Each version offers improved data transfer rates, with USB 4.0 being the fastest.

- **Connector Types:** USB connectors come in different shapes and sizes, each designed for specific applications. The most common types include USB Type-A, USB Type-B, USB Type-C, and Mini-USB/Micro-USB connectors. USB Type-C is becoming increasingly prevalent due to its small size, and reversibility.

USB Device Characteristics

- **Power Delivery:** USB can provide power to connected devices, eliminating the need for separate power adapters.
- **Device Compatibility:** USB is a widely supported standard and is compatible with a vast range of devices. It can connect devices like keyboards, mouse, printers, external hard drives, cameras, smartphones, audio devices, and many more. The USB standard ensures interoperability between devices from different manufacturers.
- **Backward Compatibility:** USB versions are designed to be backward compatible, allowing newer devices to work with older USB ports and vice versa. However, the data transfer speed will be limited by the slowest device or port in the connection.
- **Multiple Device Support:** USB supports the use of hubs, which can expand the number of USB ports available on a computer. USB hubs allow users to connect multiple devices simultaneously by providing additional USB ports.
- **Data and Power Cables:** USB cables carry both data and power signals. Depending on the device and USB version, cables can transmit data at various speeds and deliver power to charge or power the connected device. USB Type-C cables can support high-speed data transfer, power delivery, and video output.

USB Architecture



- USB uses point-to-point connections and a serial transmission
- When multiple devices are connected, they are arranged in a tree structure as shown
- Each node of the tree has a device called a *hub*, which acts as an intermediate transfer point between the host computer and the I/O devices
- a *root hub* connects the entire tree to the host computer
- The leaves of the tree are the I/O devices
- The tree structure makes it possible to connect many devices using simple point-to-point serial links

Question: How multiple I/O device data is handled?
If I/O devices are allowed to send messages at any time, two messages may reach the hub at the same time and interfere with each other.

Answer: For this reason, the USB operates strictly on the basis of polling. A device may send a message only in response to a poll message from the host processor. Hence, no two devices can send messages at the same time. This restriction allows hubs to be simple, low-cost devices.

Traffic on USB

- USB supports four different transfer types, which are Control, Interrupt, Bulk, and Isochronous.
- isochronous transfers are primarily used for time-sensitive data streams.
- **Isochronous transfers** in USB are designed for continuous data streaming with a guaranteed data rate and timing. They are commonly used for audio, video, and real-time communication applications where a constant and uninterrupted data flow is crucial.
- Key Functionalities

    1. Constant Data Rate
    2. Guaranteed Bandwidth
    3. No Error Correction or Retransmission (Small Error can be tolerated)
    4. Time-Sensitive Applications (time-critical applications, where maintaining synchronization)
    5. Limited Buffering

Evolution of USB Speeds

USB 1.0/1.1: 1.5 Mbps (low-speed), 12 Mbps (full-speed)

USB 2.0: 480 Mbps (high-speed)

USB 3.0/3.1 Gen 1: 5 Gbps (SuperSpeed)

USB 3.1 Gen 2: 10 Gbps (SuperSpeed+)

USB 3.2 Gen 1x2: 10 Gbps

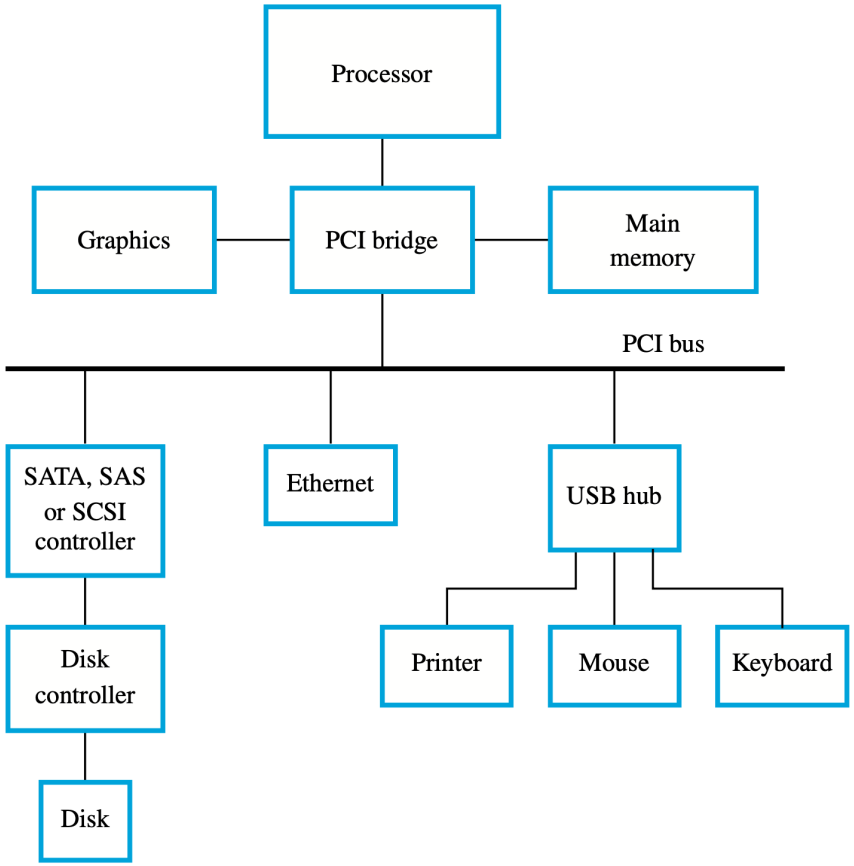
USB 3.2 Gen 2x2: 20 Gbps

USB 4: 40 Gbps

PCI Bus (Peripheral Component Interconnect Bus)

- A low-cost, processor-independent bus
- Housed on the motherboard of a computer
- Used to connect I/O interfaces for a wide variety of devices.
- Device connected to the PCI bus appears to the processor as if it is connected directly to the processor bus
- Its interface registers are assigned addresses in the address space of the processor

PCI Bus Structure



- The PCI bus is connected to the processor bus via a controller called a bridge.
- The bridge has a special port for connecting the computer's main memory.
- It may also have another special high-speed port for connecting graphics devices.
- The bridge translates and relays commands and responses from one bus to the other and transfers data between them.
- When the processor sends a Read request to an I/O device, the bridge forwards the command and address to the PCI bus.
- When the bridge receives the device's response, it forwards the data to the processor using the processor bus.

- The PCI bus supports three independent address spaces: (1) Memory (2) I/O and (3) Configuration.
- Memory Mapping with I/O: The system designer may choose to use memory-mapped I/O even with a processor that has a separate I/O address space. This approach is recommended for wider compatibility.
- Configuration space: This is intended to give the PCI its plug-and-play capability. A 4-bit command that accompanies the address identifies which of the three spaces is being used in a given data transfer operation.
- Data transfers on a computer bus often involve bursts of data rather than individual words. Words stored in successive memory locations are transferred directly between the memory and an I/O device such as a disk or an Ethernet connection.
- Data transfers are initiated by the interface of the I/O device, which acts as a bus master. The PCI bus is designed primarily to support multiple-word transfers. A Read or a Write operation involving a single word is simply treated as a burst of length one.

PCI Bus Key Features

- Wide Compatibility
  - Compatible with various computer systems and operating systems
- Higher Data Transfer Speed
  - A 32-bit PCI bus operating at 33 MHz provides a maximum bandwidth of 133 MB/s
  - A 64-bit PCI bus running at 66 MHz offers a maximum bandwidth of 533 MB/s
- Feature expansion slots on the motherboard
- Plug-and-Play
- Wider Supports for Peripherals
- Interrupt Request (IRQ) lines Sharing
- Hot Swapping
- Backward Compatibility
- DMA Support

Data Transfer on PCI Bus

- The bus master: It is the device that initiates data transfers by issuing Read and Write commands, is called the *initiator* in PCI terminology.
- The addressed device that responds to these commands is called a *target*.
- There are 32 or 64 lines that carry address and data using a synchronous signalling scheme.
- A complete transfer operation on the PCI bus, involving an address and a burst of data, is called a *transaction*.

Table 7.1 Data transfer signals on the PCI bus.

Name	Function
CLK	A 33-MHz or 66-MHz clock
FRAME#	Sent by the initiator to indicate the duration of a transmission
AD	32 address/data lines, which may be optionally increased to 64
C/BE#	4 command/byte-enable lines (8 for a 64-bit bus)
IRDY#, TRDY#	Initiator-ready and Target-ready signals
DEVSEL#	A response from the device indicating that it has recognized its address and is ready for a data transfer transaction
IDSEL#	Initialization Device Select

A signal whose name ends with the symbol # is asserted when in the low-voltage state.

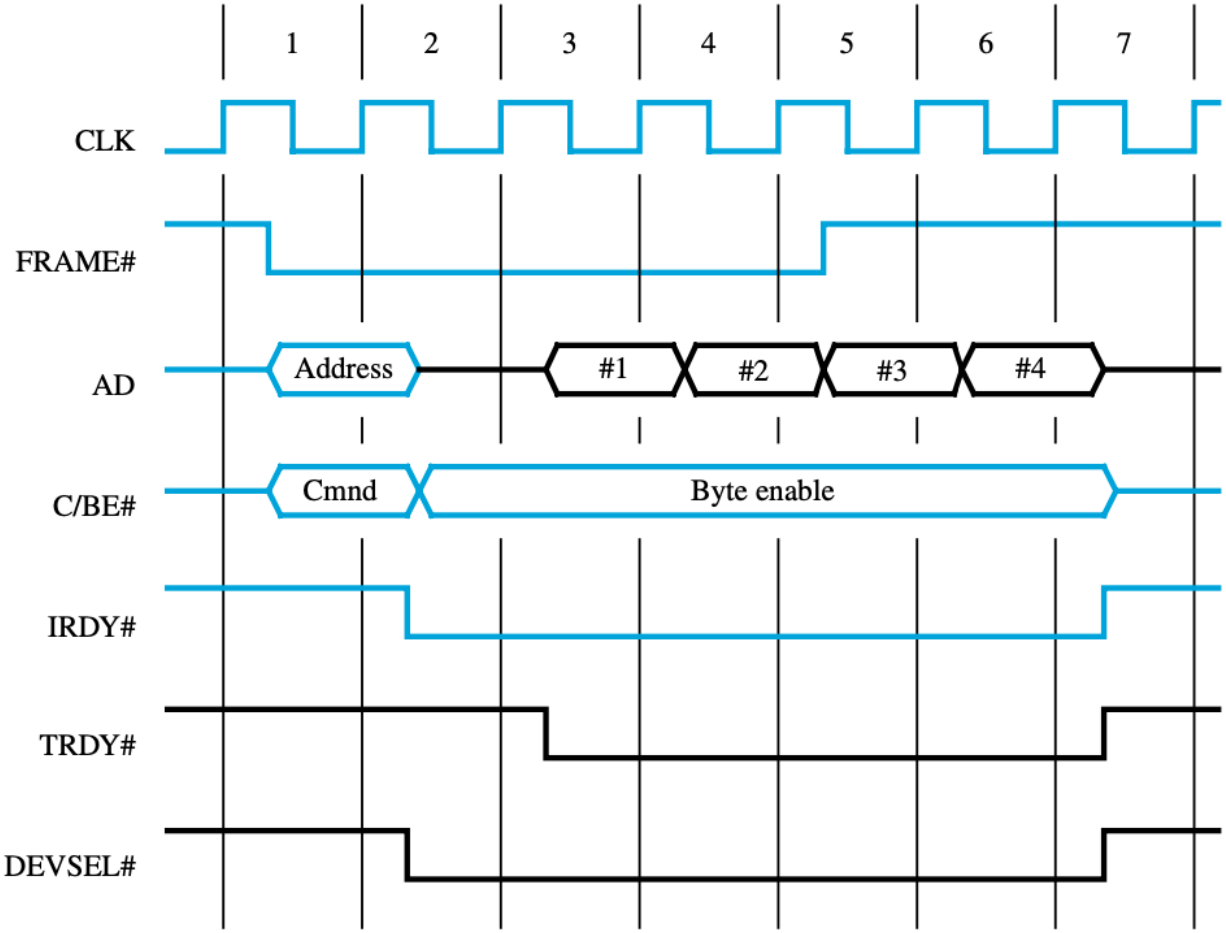


Figure 7.19 A Read operation on the PCI bus.

Evolution of PCI Speed

PCI 1.0: 133 MB/s (32-bit, 33 MHz)

PCI 2.0: 266 MB/s (32-bit, 66 MHz)

PCI-X:

PCI-X 1.0: 533 MB/s (64-bit, 66 MHz)

PCI-X 2.0: 1.06 GB/s (64-bit, 133 MHz)

PCI-X 3.0: 2.12 GB/s (64-bit, 133 MHz)

PCI-X 4.0: 4.24 GB/s (64-bit, 133 MHz)

PCI Express (PCIe):

PCIe 1.0/1.1: 250 MB/s per lane (per direction)

PCIe 2.0: 500 MB/s per lane (per direction)

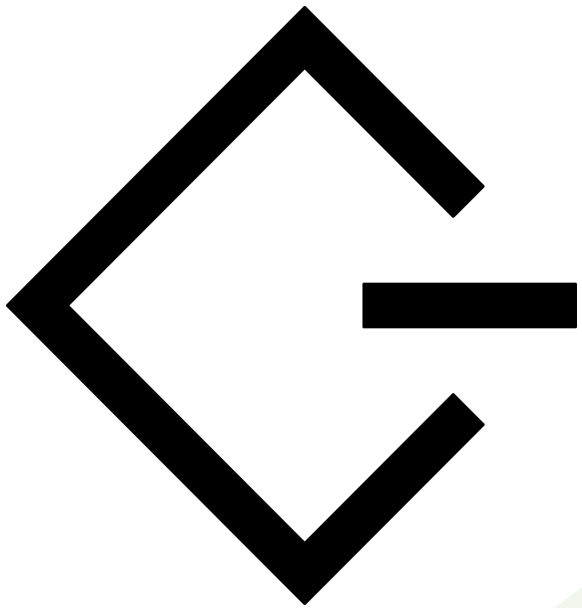
PCIe 3.0: 1 GB/s per lane (per direction)

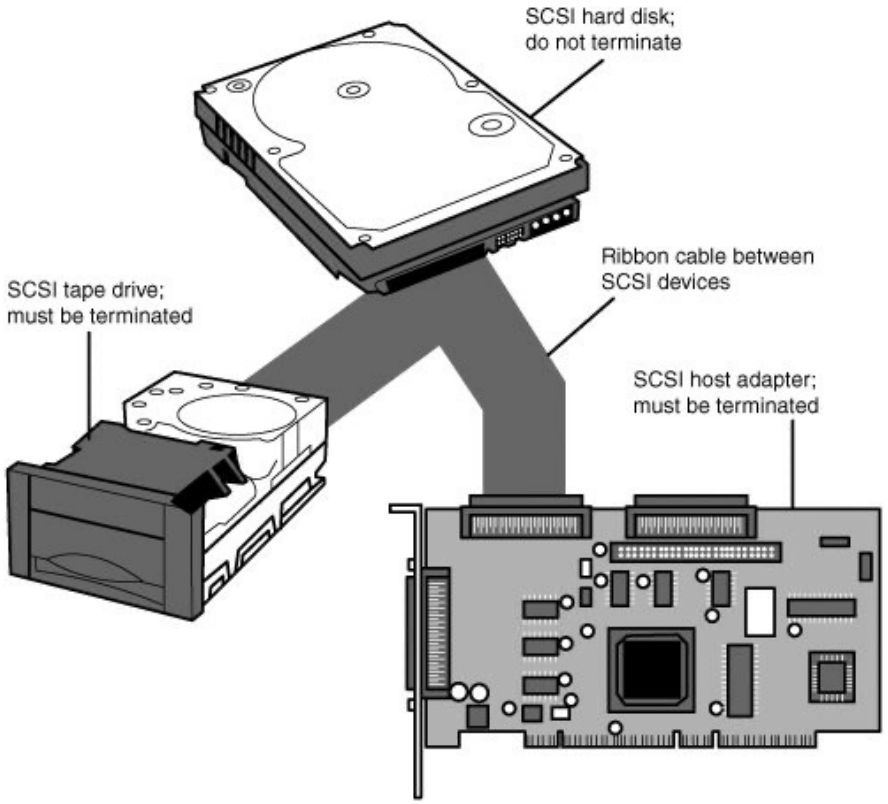
PCIe 4.0: 2 GB/s per lane (per direction)

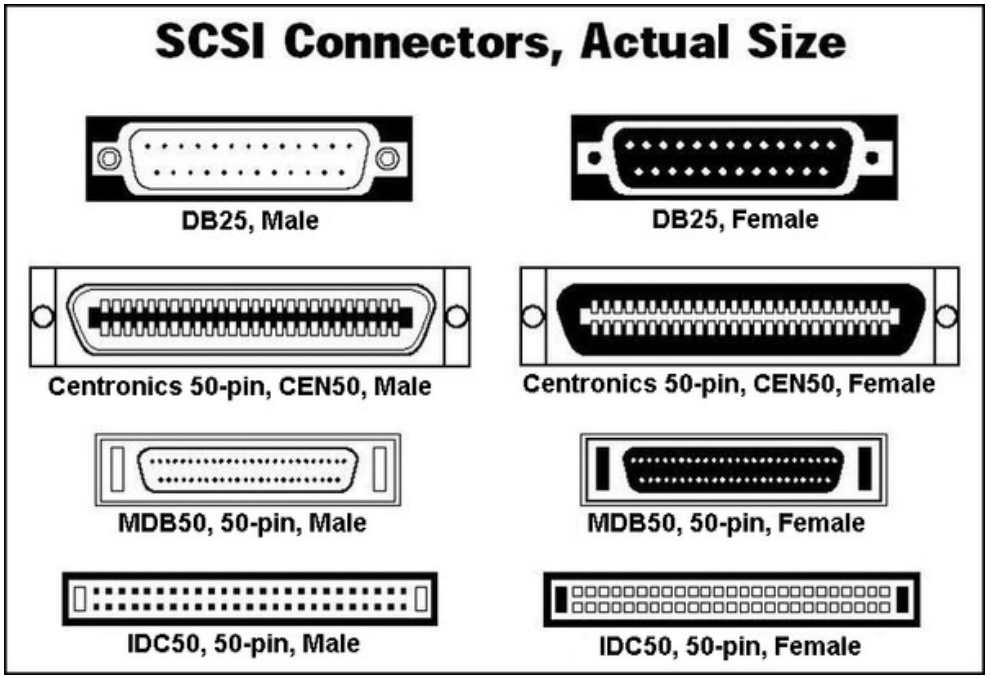
PCIe 5.0: 4 GB/s per lane (per direction)

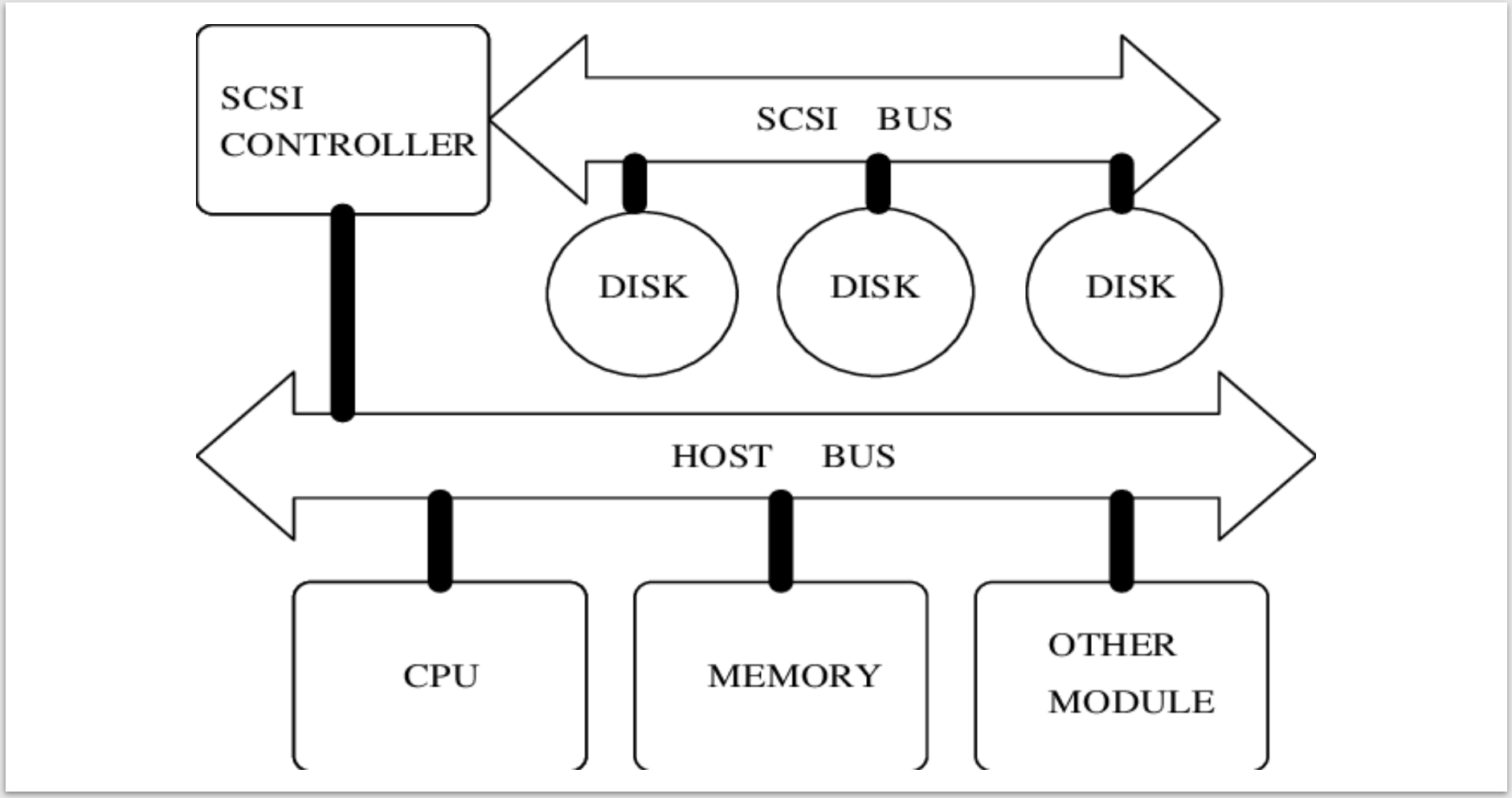
SCSI Bus (Small Computer System Interface)

- A standard bus defined by the American National Standards Institute (ANSI).
- SCSI bus may be used to connect a variety of devices to a computer.
- Particularly well-suited for use with disk drives.
- often found in installations such as institutional databases or email systems where many disks drives are used
- In the original specifications of the SCSI standard, devices are connected to a computer via a 50-wire cable, which can be up to 25 meters in length and can transfer data at rates of up to 5 Megabytes/s.




SCSI hard disk;
do not terminate
Ribbon cable between
SCSI devices
SCSI tape drive;
must be terminated
SCSI host adapter;
must be terminated


SCSI Connectors, Actual Size
DB25, Male
DB25, Female
Centronics 50-pin, CEN50, Male
Centronics 50-pin, CEN50, Female
MDB50, 50-pin, Male
MDB50, 50-pin, Female
IDC50, 50-pin, Male
IDC50, 50-pin, Female


SCSI
CONTROLLER
SCSI BUS
DISK
DISK
DISK
HOST BUS
CPU
MEMORY
OTHER
MODULE

Assume that the processor wishes to read a block of data from a disk drive and that these data are stored in two disk sectors that are not contiguous. The processor sends a command to the SCSI controller, which causes the following sequence of events to take place:

1. The processor wants to read some data from a disk drive, but the data is stored in two separate sectors on the disk.

2. The processor sends a command to the SCSI controller, saying it wants to perform a Read operation.

3. The disk controller (a device that manages communication with the disk itself) needs to move its read head to the first sector where the data is stored before it can start transferring the data. So, it tells the SCSI controller that it needs to temporarily pause the connection between them. This means the SCSI bus (the pathway for communication between devices) is available for other devices to use.

4. The disk controller sends a command to the disk drive, instructing it to move the read head to the first sector where the data is stored.

5. It reads the data from that sector and stores it in a special area called a data buffer.

6. When it's ready to transfer the data, the disk controller requests control of the SCSI bus.

7. Once it wins the right to use the bus (by a process called arbitration), it re-establishes the connection with the SCSI controller, sends the data from the buffer, and then temporarily pauses the connection again.

8. The same process is repeated to read and transfer the data from the second disk sector where the rest of the required data is stored.
9. The SCSI controller receives the requested data from the disk controller and transfers it to the main memory of the computer. It also sends a signal to the processor to let it know that the data is now available. The processor can then continue with its tasks using the data it received.