# System Organization

- A general purpose computer should have the ability to exchange information with a wide range of devices in varying environments.

- Computers can communicate with other computers over the Internet and access information around the globe.

- They are an integral part of home appliances, manufacturing equipment, transportation systems, banking and point-ofsale terminals

Accessing I/O Devices
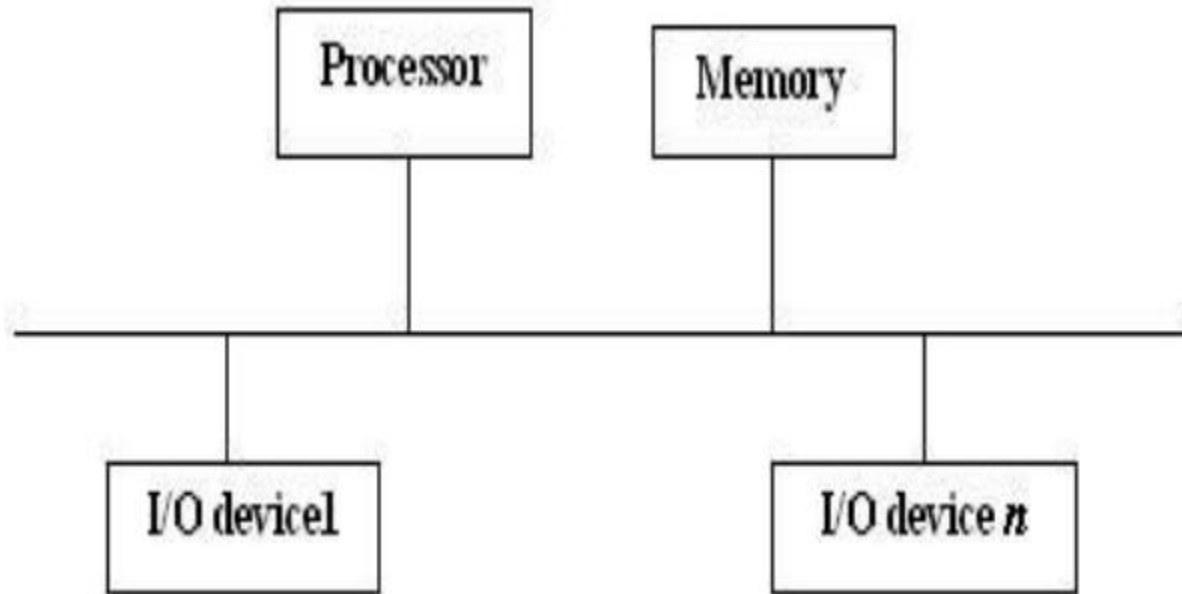
A single-bus structure

- A simple arrangement to connect I/O devices to a computer is to use a single bus arrangement, as shown in above figure. Each I/O device is assigned a unique set of address.

- When the processor places a particular address on the address lines, the device that recognizes this address responds to the commands issued on the control lines.

- The processor requests either a read or a write operation which is transferred over the data lines.

- When I/O devices and the memory share the same address space, the arrangement is called memory-mapped I/O.

- Consider, for instance, with memory-mapped I/O, if DATAIN is the address of the input buffer of the keyboard

Move DATAIN, R0

Reads the data from DATAIN and store them into processor register R0

And DATAOUT is the address of the output buffer of the display/printer

Move R0, DATAOUT

Sends the contents of R0 to location DATAOUT, which may be the output data buffer of a display unit or a print.

Most computer systems use memory-mapped I/O. Some processors have special I/O instructions to perform I/O transfers. The hardware required to connect an I/O device to the bus is shown below:
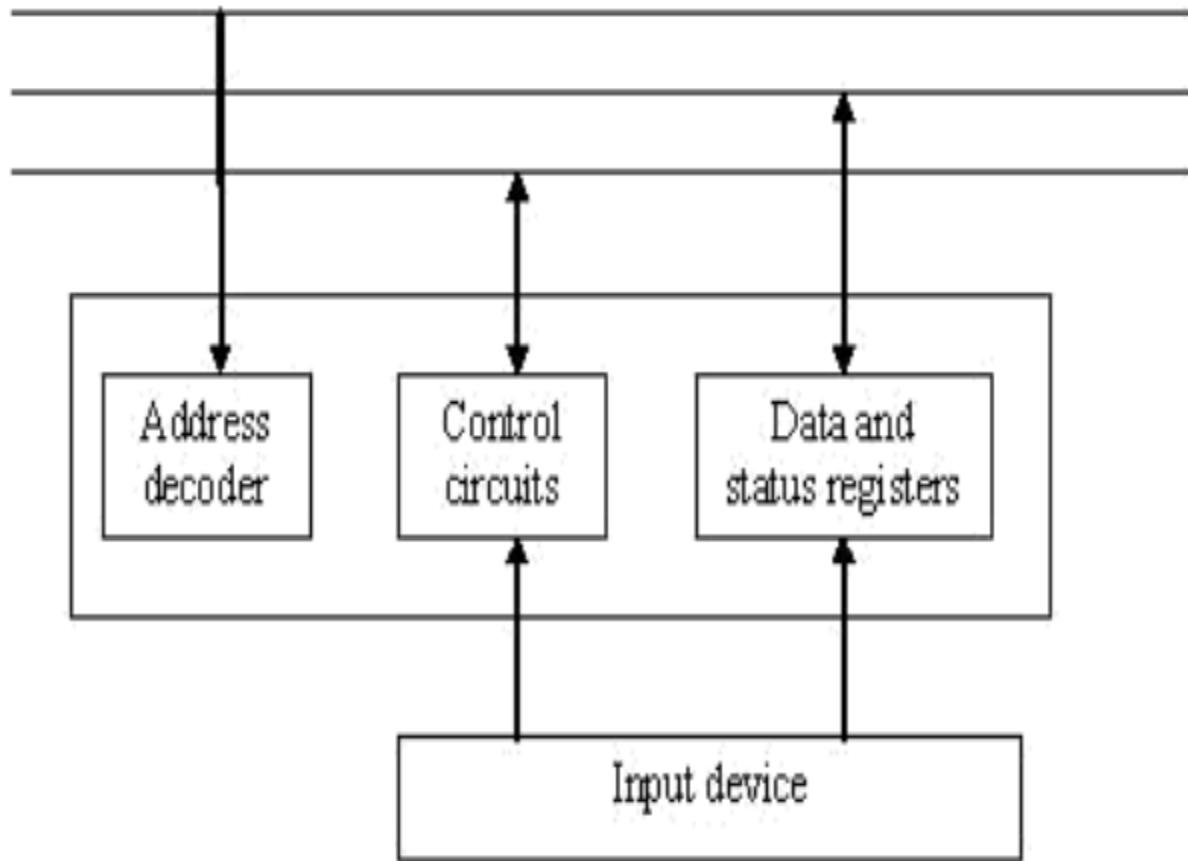
I/O interface for an input device

- The address decoder enables the device to recognize its address when this address appears on the address lines.
- The data register holds the data.
- The status register contains information.
- The address decoder, data and status registers and controls required to coordinate I/O transfers constitutes interface circuit
- For eg: Keyboard, an instruction that reads a character from the keyboard should be executed only when a character is available in the input buffer of the keyboard interface. The processor repeatedly checks a status flag to achieve the synchronization between processor and I/O device, which is called as program-controlled I/O.

- Two commonly used mechanisms for implementing I/O operations are:

- Interrupts

- Direct memory access

Introduction

❖ In program-controlled I/O, processor enters a waiting loop in which it continuously monitors the status of the device. During this period, it does not perform any useful tasks.

❖ An alternate arrangement for the I/O device to alert the processor when it becomes ready.

  • It can be possible by sending a hardware signal called an interrupt to the processor.

  • At least one of the bus control lines, called an interrupt-request line is dedicated for this purpose.

❖ Processor can perform other useful tasks while it is waiting for the device to be ready.

Interrupts

➤An I/O device requests an interrupt by activating a buss line called **Interrupt request** when I/O device is ready to perform it's task.

➤When a processor receives an interrupt-request, it must branch to the interrupt service routine.

➤It must also inform the device that it's request has been recognized. This may be accomplished by means of a special control signal called **interrupt-acknowledge signal**

Main program

Sub Program(ISR)

Processor

INTR

INTA

I/O devices
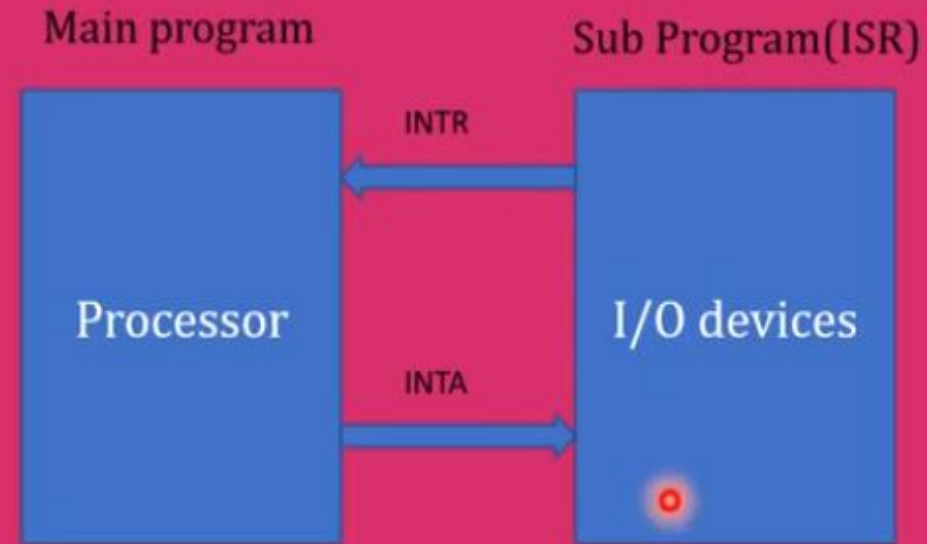
Transfer of control through the use of interrupts:

- 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.

- Before branching to the interrupt-service routine, PC (i+1), and also other information such as condition code flags, processor registers used by both the interrupted program and the interrupt service routine must be stored on the processor stack.

- This will enable the return-from-interrupt instruction to resume execution at i+1.
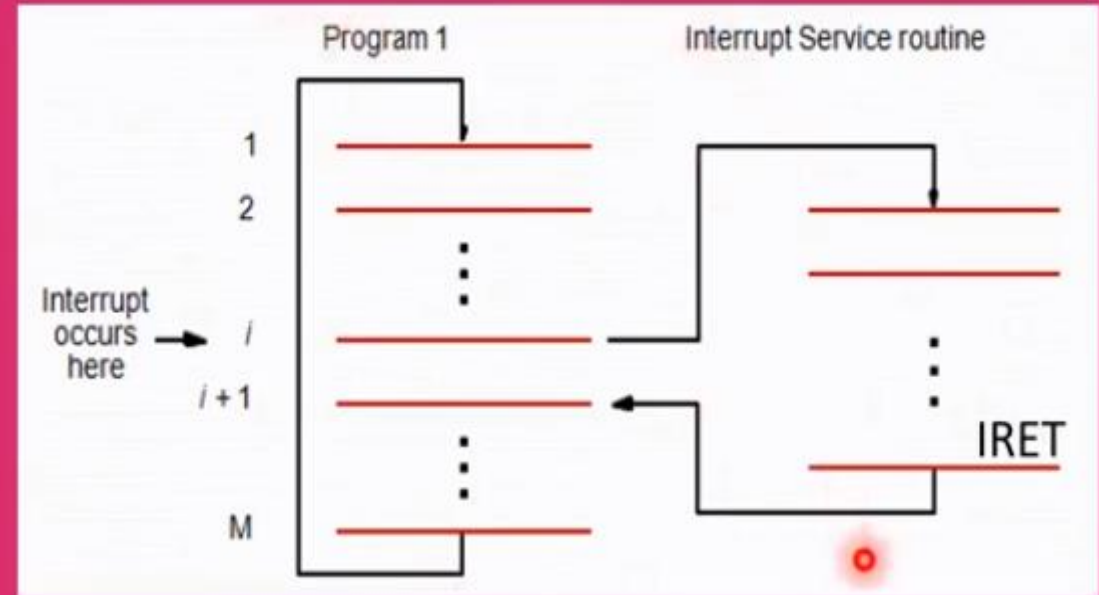
Interrupt Hardware

- Saving and restoring information can be done automatically by the processor or explicitly by program instructions.
- Saving and restoring registers involves memory transfers:
  - 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 <u>interrupt latency</u>.
- 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.

An equivalent circuit for an open-drain bus used to implement a common interrupt-request line

If all switches are open

INTR 1, INTR 2,...... INTR n are inactive
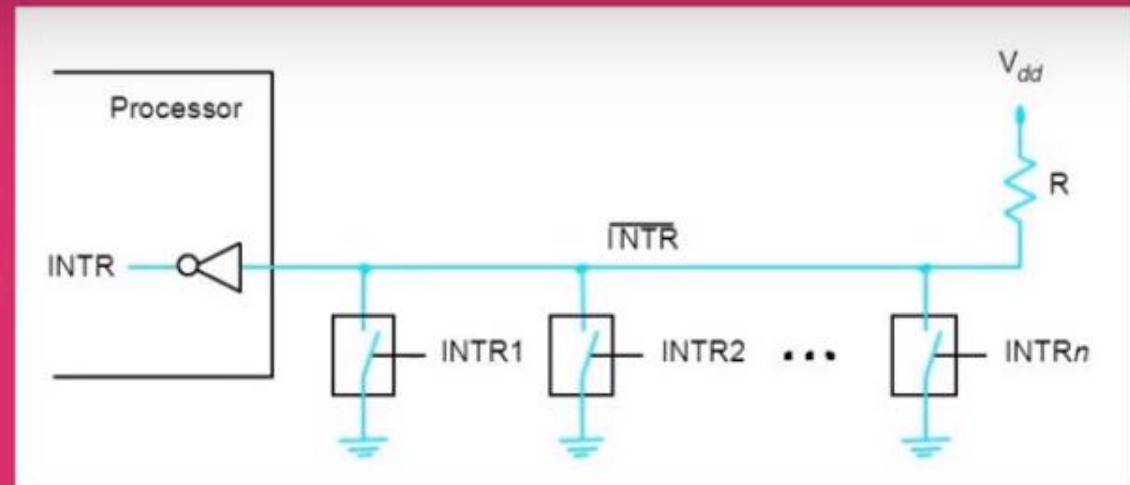
Then the voltage on the INTR line =Vdd

INTR=0

If all the switch are closed

INTR 1, INTR 2,...... INTR n are active

Then the voltage on the INTR line =0

INTR=1

INTR= INTR 1+ INTR 2+...... INTR n

Enabling and Disabling Interrupt

- Interrupt can occur any time which alter the execution sequence.
- To provide programmer complete control over program execution event.
- So interruption of program carefully controlled i. e. by Enable and disable interrupt as desire
- Some situation interrupt have to ignore e.g. PRINT interrupt the processor even COMPUTE is not ready with text to print.

Enabling and Disabling Interrupt

- A single interrupt request from one device by activating the interrupt-request signal, it keeps this signal activated until it learns that the processor has accepted its request. This means that the interrupt-request signal will be active during execution of the interrupt-service routine.

- It is essential to ensure that this active request signal does not lead to successive interruptions, causing the system to enter an infinite loop from which it cannot recover.

- So three possibilities used to handle one or more interrupt request.

Enabling and Disabling Interrupt

- First Possibility
  - Ignore the IR until complete the current ISR
  - i.e. first instruction of ISR is Interrupt Disable and last instruction is Interrupt Enable

Enabling and Disabling Interrupt

- Second Possibility
  - Processor first saves the contents of the program counter (PC) and the processor status (PS) register with IE=1 on stack and automatically disable interrupts before starting the execution of the interrupt-service routine.
  - When return from interrupt instruction is executed the contents of PC and PS is popped with IE=1 from stack.

Enabling and Disabling Interrupt

- Third Possibility
  - IR line must accept only leading edge of the signal (Edge triggered line)
  - Processor receive only one request regardless of how long the line is activated.
  - So no question of multiple interruption or no need of explicit instruction for enable/disable interrupt.

Interrupt hardware

Most computers have several I/O devices that can request an interrupt. A single interrupt request line may be used to serve n devices.

Enabling and Disabling Interrupts

All computers fundamentally should be able to enable and disable interruptions as desired. Again reconsider the COMPUTE and PRINT example. When a device activates the interrupt-request signal, it keeps this signal activated until it learns that the processor has accepted its request. When interrupts are enabled, the following is a typical scenario:

- The device raises an interrupt request.
- The processor interrupts the program currently being executed.
- Interrupts are disabled by changing the control bits in the processor status register (PS).
- The device is informed that its request has been recognized and deactivates the interrupt request signal.
- The action requested by the interrupt is performed by the interrupt-service routine.
- Interrupts are enabled and execution of the interrupted program is resumed.

Handling Multiple Devices

- Multiple I/O devices may be connected to the processor and the memory via a bus. These devices may be capable of generating interrupt requests.

Example: Device X may request an interrupt while device Y being served.

Or several devices may request interrupts at exactly the same time.

Questions may be raised :

- How can the processor recognize which device requesting an interrupt?
- How can the processor obtain the starting address of the appropriate interrupt service routine in each case?
- Should a device be allowed to interrupt the processor while another interrupt being served?
- How should two or more simultaneous interrupt-requests be handled?

Handling Multiple Devices

When a request is received over the common interrupt-request line as shown in figure..
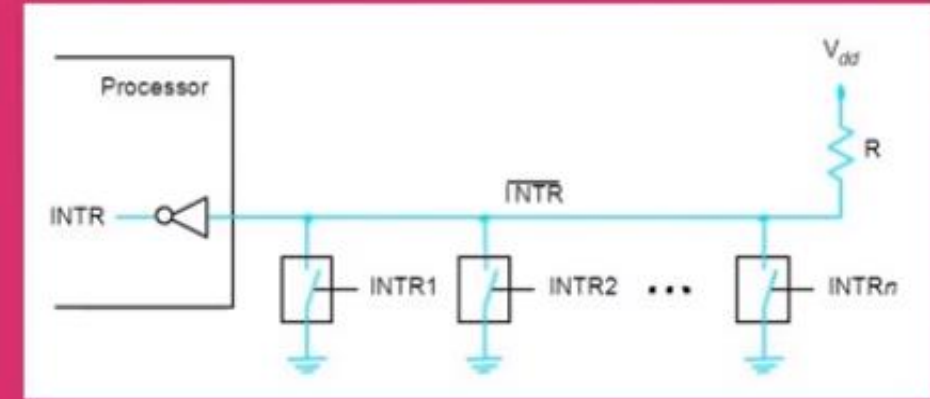
- **Polling mechanism:**
- Additional information needed to identify the particular device that activated the line.
- information is available in the status register of the device
  - The status register of each device has an *IRQ* bit which it sets to 1 when it requests an interrupt.
- Interrupt service routine can poll all the I/O devices connected to the bus to identify the interrupting device.
- 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.

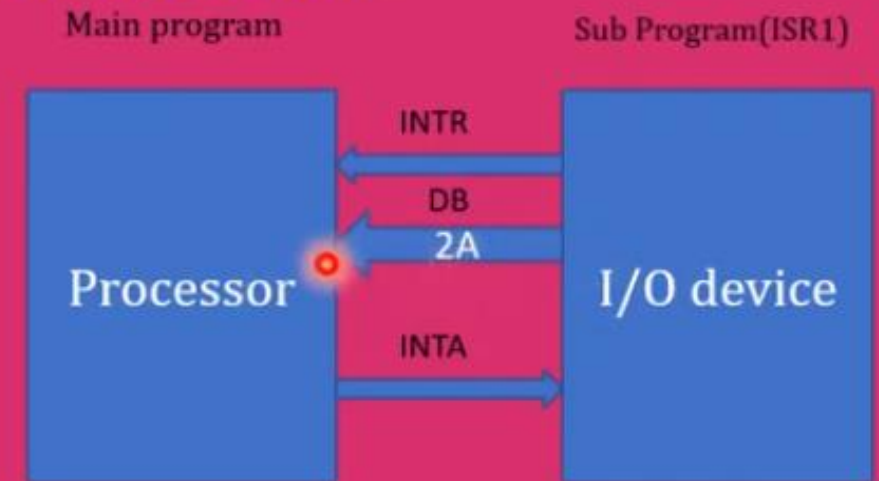An alternative approach is to use vectored interrupts.

Vectored Interrupts

To reduce the time involved in the polling process,

- The device requesting an interrupt can identify itself by sending a special code (4 to 8 bits) to the processor over the data bus. This enables the processor to identify individual devices.

- Code supplied by the device may represent a starting address of the interrupt-service routine.

➤Starting address of the interrupt-service routine called interrupt vector.

Vector Table

| 2A |
| 3A |
| 4A |

ISR1   2A

ISR2   3A

ISR3   4A

Main program

Sub Program(ISR1)

Processor — INTR — DB — 2A — INTA — I/O device

Interrupt nesting

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.

❖ Processor's priority is encoded in a few bits of the processor status register.

• Each device has a separate INTR and INTA 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.
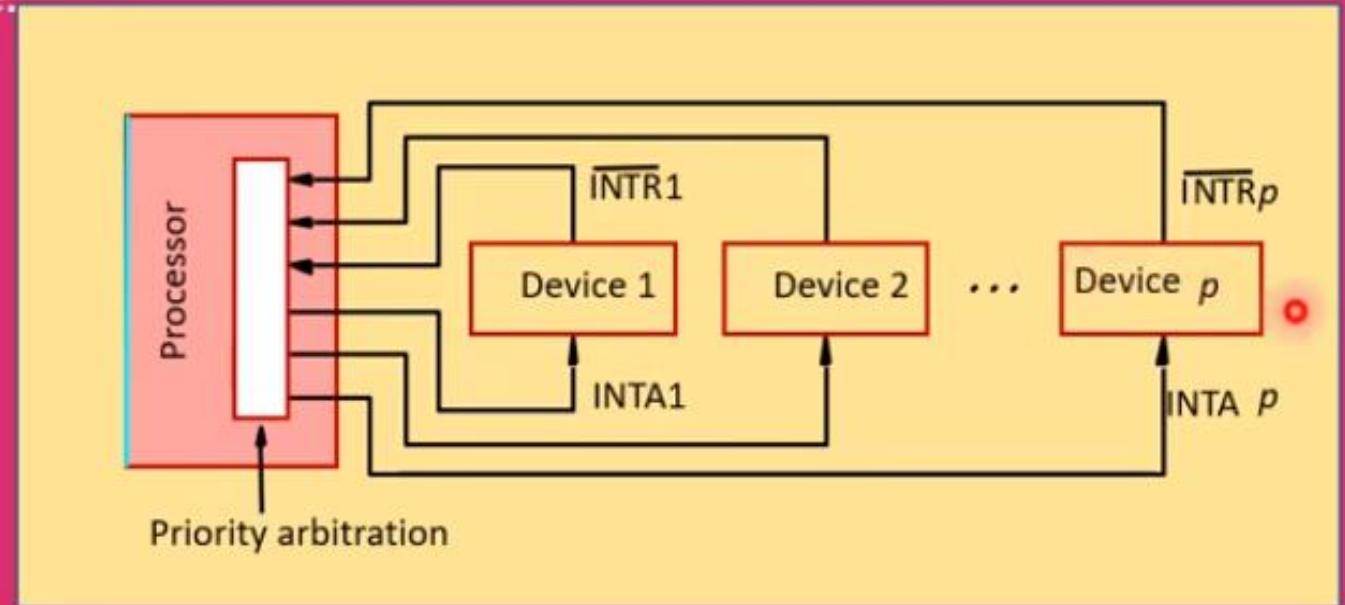
Implementation of Interrupt priority

Daisy Chain Scheme

- let us consider the problem of simultaneous arrivals of interrupt requests from two or more devices, if the devices share an interrupt request line, then how does the processor decide which interrupt request to accept?.

- This problem can be solved by connecting the devices to form a Daisy chain.

•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.
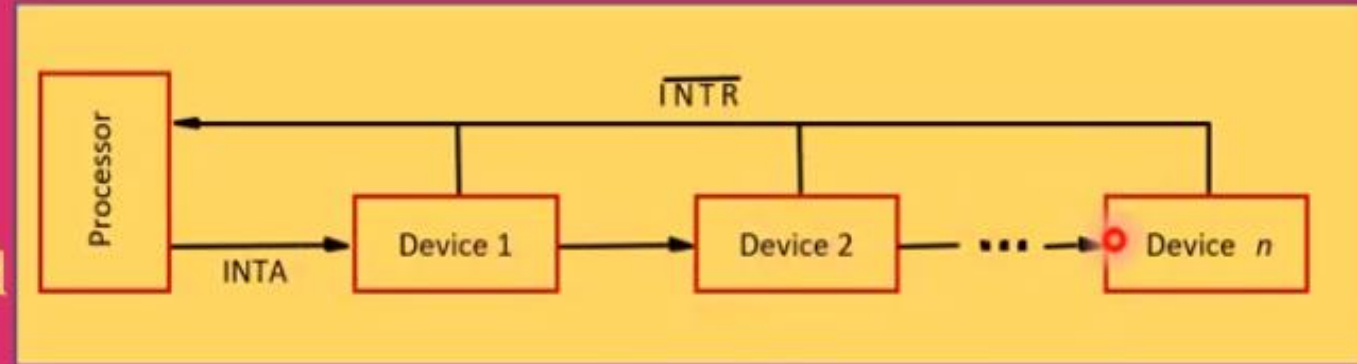
•When devices raise an interrupt request, the interrupt-request line is activated.

•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.

Arrangement of priority groups
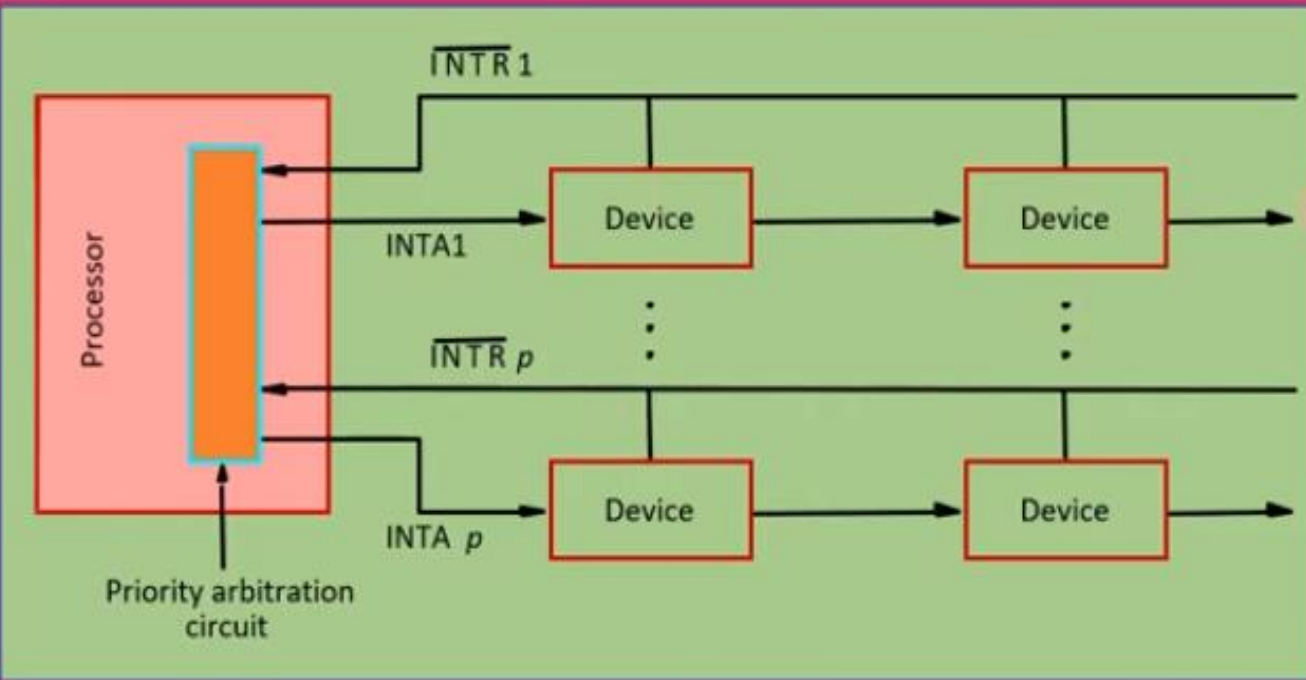
- •When I/O devices were organized into a priority structure, each device had its own interrupt-request and interrupt- acknowledge line.
- •When I/O devices were organized in a daisy chain fashion, the devices shared an interrupt-request line, and the interrupt-acknowledge propagated through the devices.

A combination of priority structure and daisy chain scheme can also used.

- •Devices are organized into groups.
- •Each group is assigned a different priority level.
- •All the devices within a single group are connected in a daisy chain.
- •This organization is used in many computer system.

Controlling Device Request-1

- It is important to ensure that interrupt requests are generated only by those I/O devices that the processor is currently willing to recognize.
- Need a mechanism in the interface circuits of individual devices to control whether a device is allowed to interrupt the processor.
- The control needed is usually provided in the form of an *interrupt-enable* bit in the device's interface circuit.
- To reduce unnecessary interrupt from other I/O devices which are not used by current executing program.

Controlling device requests At the device end, an interrupt enable bit determines whether it is allowed to generate an interrupt request. At the processor end, it determines whether a given interrupt request will be accepted.

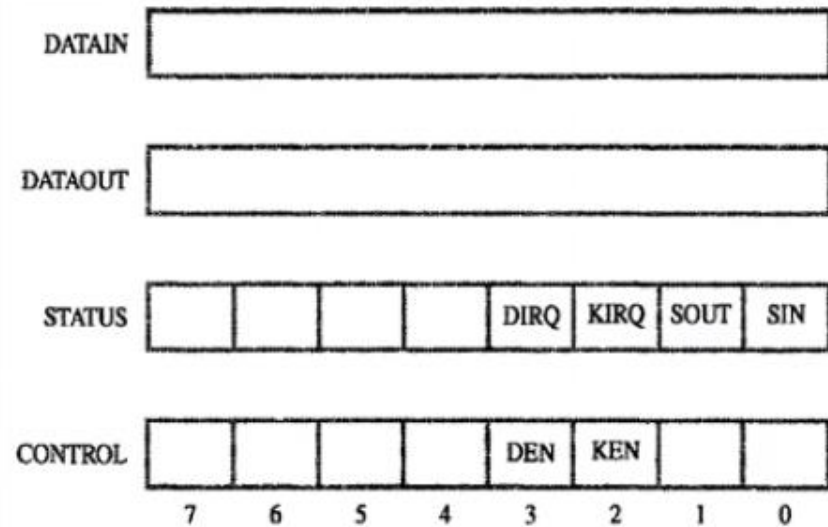# Controlling Device Request-2

Figure 4.3 Registers in keyboard and display interfaces.

- Keyboard interrupt enable bit KEN

- Display interrupt enable bit DEN

- KEN and/or DEN = 1 **then only** interface circuit generate an IR.

- Same time KIRQ and/or DIRQ = 1 **indicate that both devices is requesting an interrupt**.

Controlling Device Request-3

- Example
- When a program wish to read an input line from the keyboard and store the character in successive byte location

Exceptions

The term exception is used to refer to any event that causes an interruption. Hence, I/O interrupts are one example of an exception.

- Recovery from errors – These are techniques to ensure that all hardware components are operating properly.
- Debugging – find errors in a program, trace and breakpoints (only at specific points selected by the user).
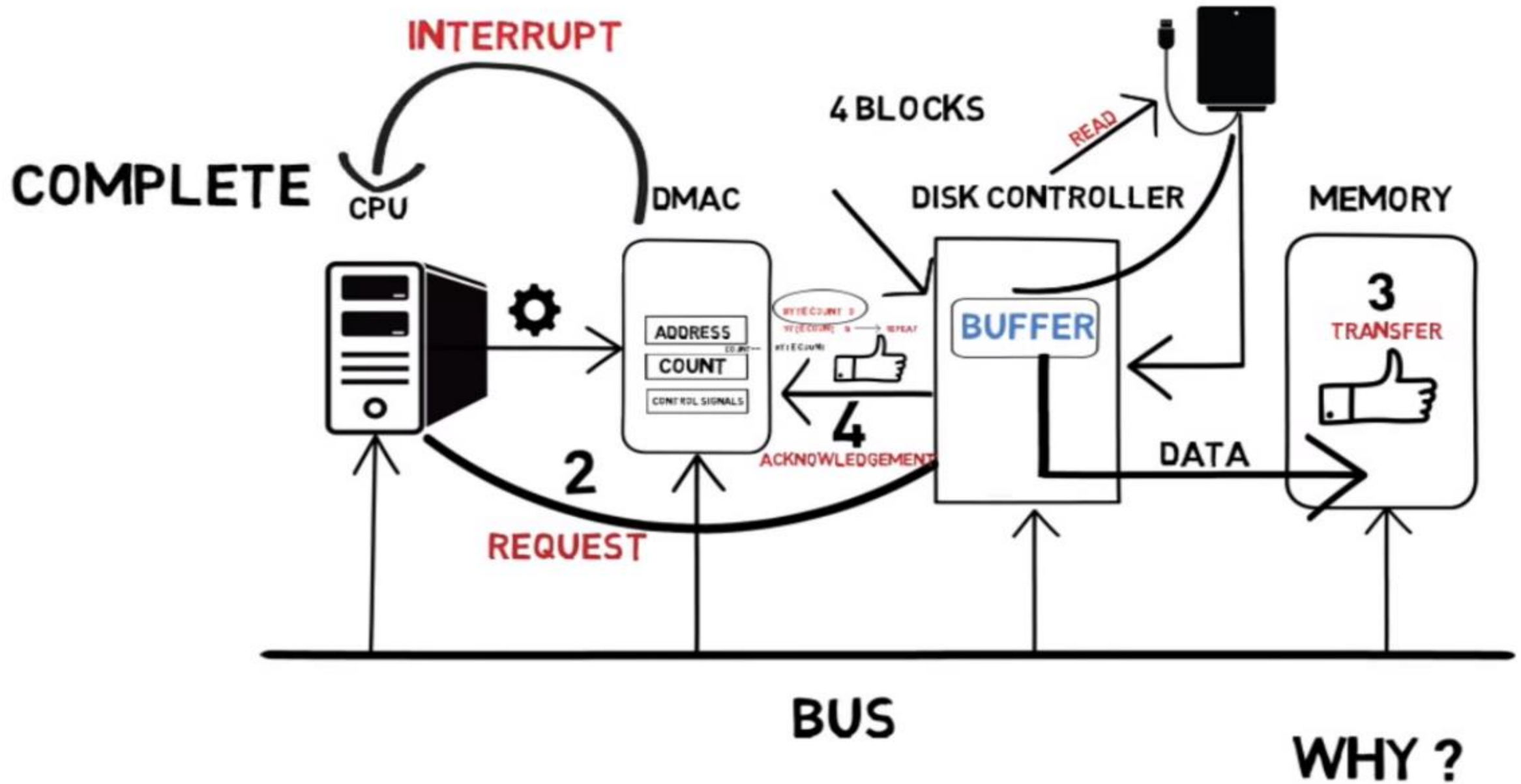
DMA (Direct Memory Access)

Direct Memory Access (DMA) :
DMA Controller is a hardware device that allows I/O devices to directly access memory with less participation of the processor.

CPU

DMAC

4 BLOCKS

READ

DISK CONTROLLER

MEMORY

ADDRESS

COUNT

CONTROL SIGNALS

BUFFER

BUS

WHY ?
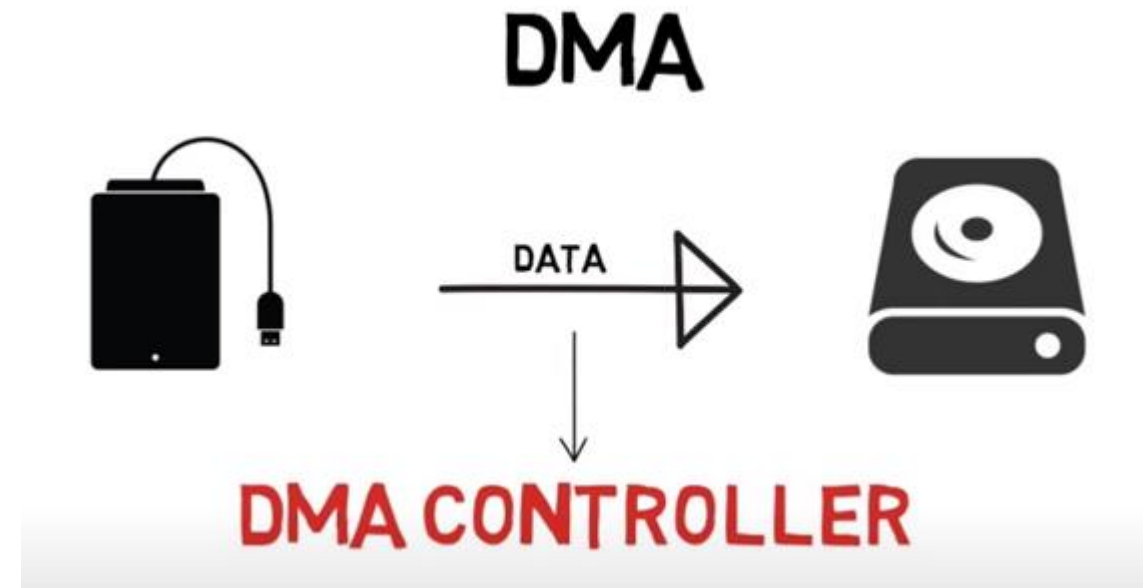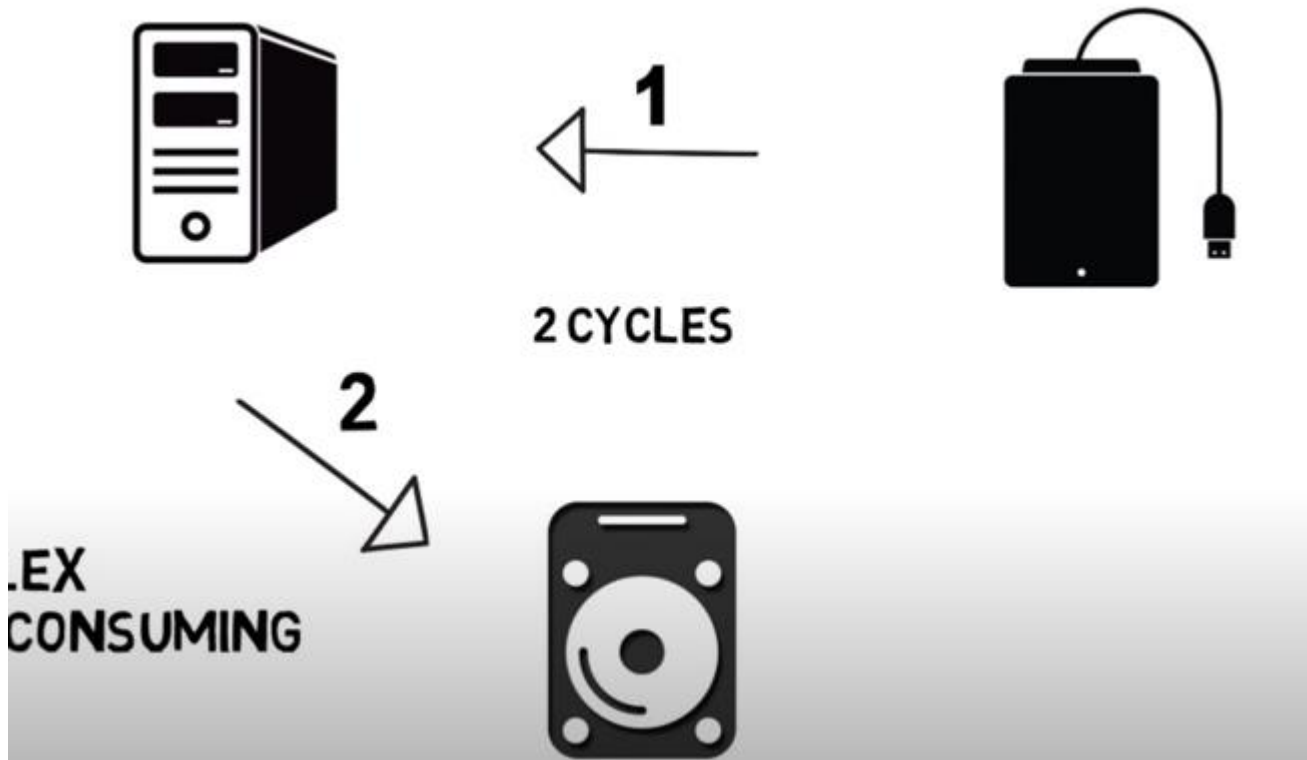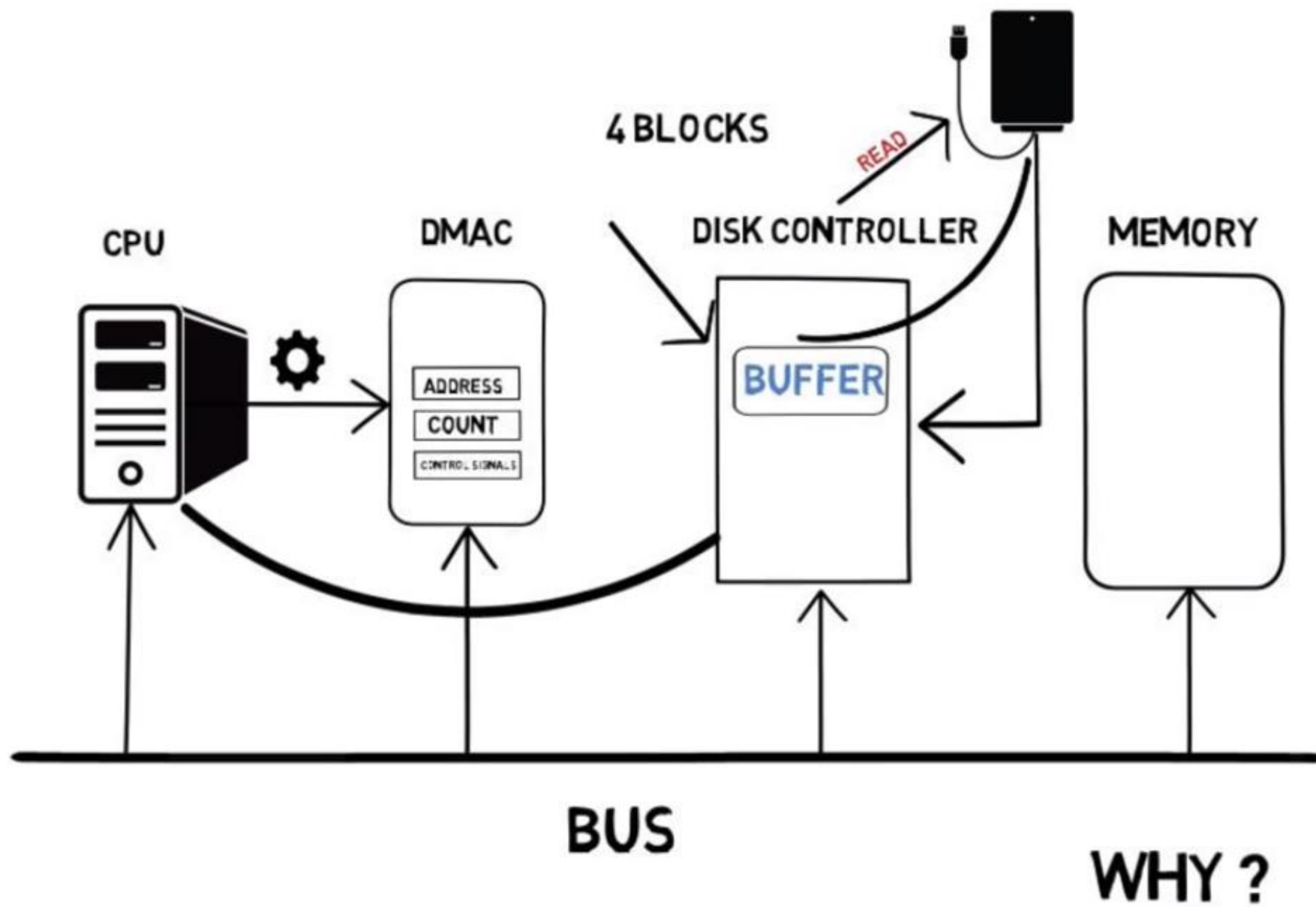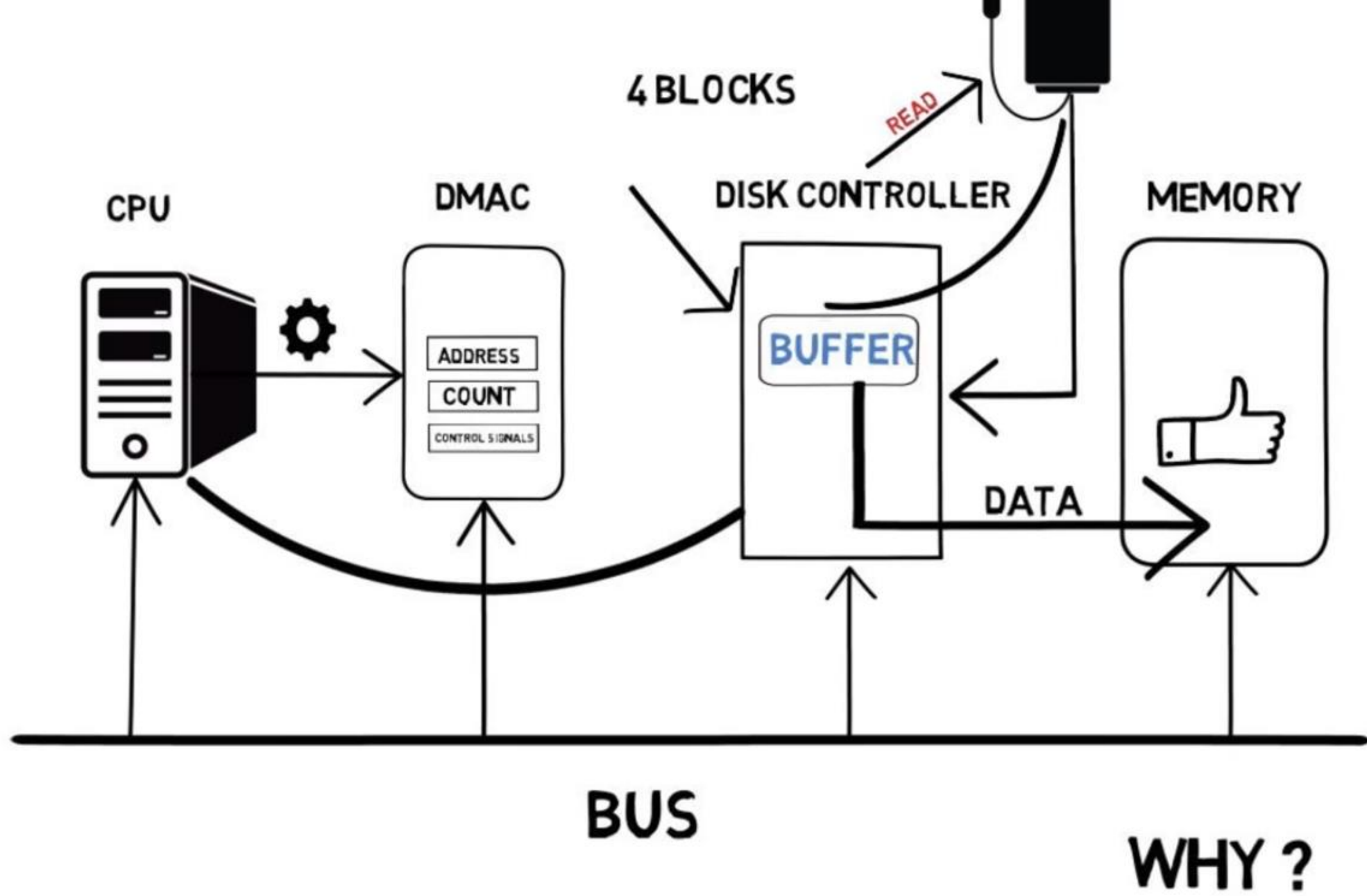
DMA

As we have seen earlier, the two commonly used mechanisms for implementing I/O operations are:

- Interrupts and
- Direct memory access

Interrupts: synchronization is achieved by having the I/O device send a special signal over the bus whenever it is ready for a data transfer operation

Direct memory access:

Basically for high speed I/O devices, the device interface transfer data directly to or from the memory without informing the processor. When interrupts are used, additional overhead involved with saving and restoring the program counter and other state information. To transfer large blocks of data at high speed, an alternative approach is used. A special control unit will allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor.
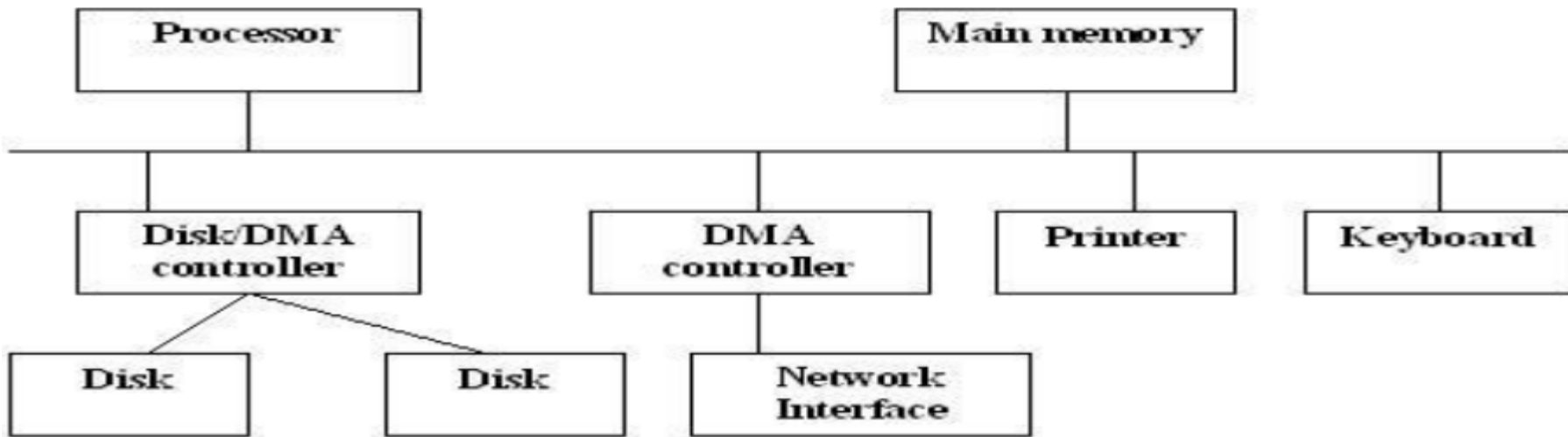
Use of DMA controllers in a computer system

- DMA controller is a control circuit that performs DMA transfers, is a part of the I/O device interface.
- It performs functions that normally be carried out by the processor.
- DMA controller must increment the memory address and keep track of the number of transfers. The operations of DMA controller must be under the control of a program executed by the processor.
- To initiate the transfer of block of words, the processor sends the starting address, the number of words in the block and the direction of the transfer.
- On receiving this information, DMA controller transfers the entire block and informs the processor by raising an interrupt signal. While a DMA transfer is taking place, the processor can be used to execute another program. After the DMA transfer is completed, the processor can return to the program that requested the transfer

Bus Structure in Computer Architecture

A system bus has typically from fifty to hundreds of distinct lines where each line is meant for a certain function. These lines can be categories into three functional groups i.e., data lines, address lines, and control lines. Let us discuss them one by one each.
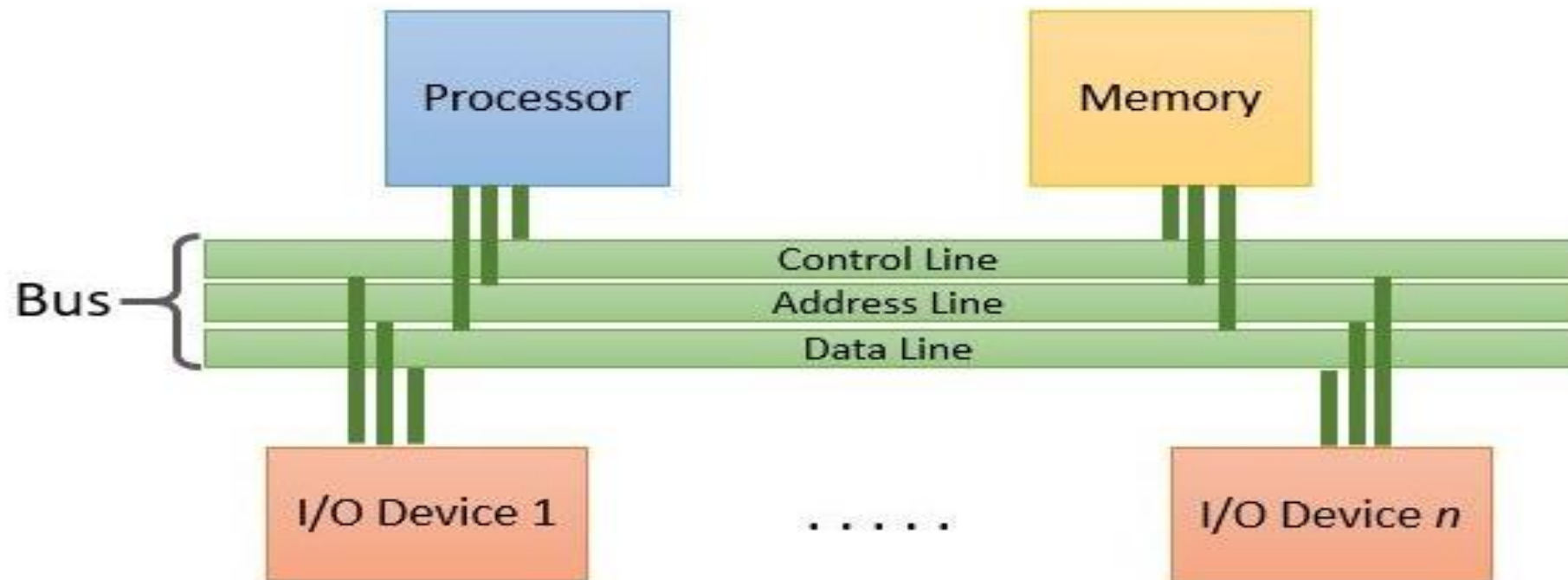
Bus Structure

- **1. Data Lines**
- Data lines coordinate in transferring the data among the system components. The data lines are collectively called data bus. A data bus may have 32 lines, 64 lines, 128 lines, or even more lines. The number of lines present in the data bus defines the *width* of the data bus.
- Each data line is able to transfer only one bit at a time. So the number of data lines in a data bus determines how many bits it can transfer at a time. The performance of the system also depends on the width of the data bus.

Address Lines

- The content of the address lines of the bus determines the source or destination of the data present on the data bus. The number of address lines together is referred to as address bus. The number of address lines in the address bus determines its *width*.

- The width of the address bus determines the memory capacity of the system. The content of address lines is also used for addressing I/O ports.

- Whenever the processor has to read a word from the memory it simply places the address of the corresponding word on the address line.

- **3. Control Lines**
- The address lines and data lines are shared by all the components of the system so there must some means to control the use and access of data and address lines.
- The control signals placed on the control lines control the use and access to address and data lines of the bus.
- The control signal consists of the *command* and *timing information*. Here the command in the control signal specify the *operation* that has to be performed. And the timing information over the control signals specify till when the data and address information is valid .

- The control lines include the lines for:
- **Memory Write:** This command causes the data on the data bus to be placed over the addressed memory location.
- **Memory Read:** This command causes the data on the addressed memory location to be placed on the data bus.
- **I/O Write:** The command over this control line causes the data on the data bus to be placed over the addressed I/O port.
- **I/O Read:** The command over this control line causes the data from the addressed I/O port to be placed over the data bus.
- **Transfer ACK:** This control line indicates the data has been received from the data bus or is placed over the data bus.
- **Bus Request:** This control line indicates that the component has requested control over the bus.
- **Bus Grant:** This control line indicates that the bus has been granted to the requesting component.
- **Interrupt Request:** This control line indicates that interrupts are pending.
- **Interrupt ACK:** This control line provides acknowledgment when the pending interrupt is serviced.
- **Clock:** This control line is used to synchronize the operations.
- **Reset:** The bit information issued over this control line initializes all the modules.

Introduction

- Processor, main memory, and I/O devices are interconnected by means of a bus.

- Bus provides a communication path for the transfer of data.

- A **bus protocol** is the set of rules that govern the behavior of various devices connected to the bus, as to when to place information on the bus, when to assert control signals, etc.

Cont...

- Bus lines may be grouped into three types:
  - Data
  - Address
  - Control
- Control signals specify:
  - Whether it is a read or a write operation.
  - Required size of the data, when several operand sizes (byte, word, long word) are possible.
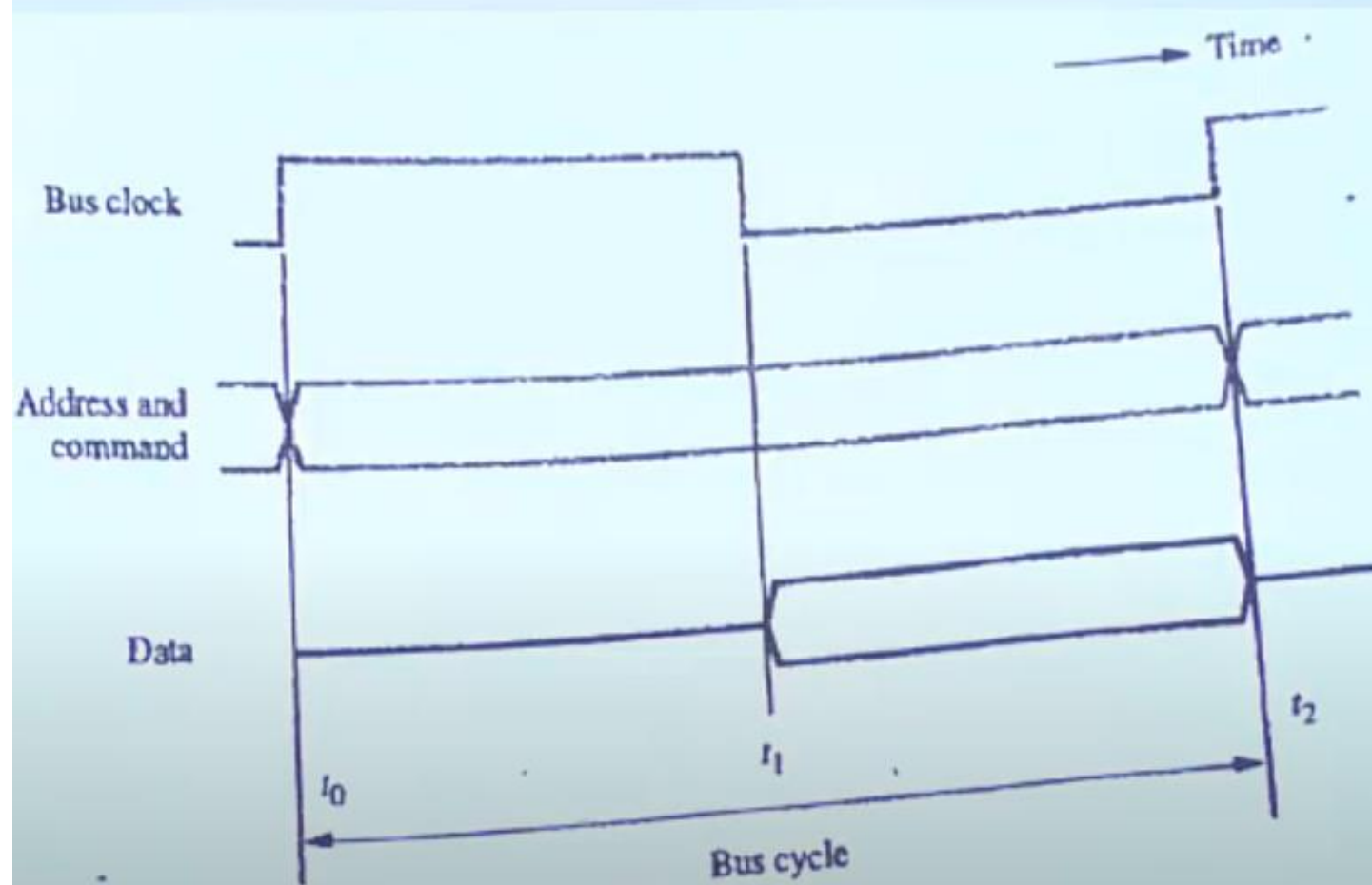
Cont...

- Timing information to indicate when the processor and I/O devices may place data or receive data from the bus.

- Schemes for timing of data transfers over a bus can be classified into:

  - Synchronous Bus
  - Asynchronous Bus

Synchronous Bus

- In synchronous buses, the steps of data transfer take place at fixed clock cycles.
- Everything is synchronized to bus clock.
- The clock signals are made available to both master and slave.
- The bus clock is a square wave signal.
- A transfer may take multiple bus cycles depending on the speed parameters of the bus and two ends of the transfer.

Cont...

Cont...

- In case of write operation, the master places the data on the bus along with the address and commands at time t0.

- The slave strobes the data into its input buffer at time t2.

- Once master places the device address and command on the bus, it takes time for this information to propagate to the devices.

- This time depends on the physical and electrical characteristics of the bus.

Cont...

- Also, all the devices have to be given enough time to decode the address and control signals, so that the addressed slave can place data on the bus.
- Width of the pulse depends on:
- 1. Maximum propagation delay between two devices connected to the bus.
- 2. Time taken by all the devices to decode the address and control signals, so that the addressed slave can respond at time t1.

Asynchronous Bus

Introduction

- Data transfers on the bus is controlled by a **handshake** between the master and the slave.

- Common clock in the synchronous bus case is replaced by two timing control lines **Master-ready and Slave-ready.**

- **Master-ready signal** is asserted by the **master** to indicate that master is ready to participate in a data transfer.

Cont...

- **Slave-ready signal** is asserted by the **slave** in response to the master-ready.

- It indicates to the master that the slave is ready to participate in a data transfer.

- **Data transfer using the handshake protocol:**

- 1. **Master** places the **address** and **command** information on the bus.

Cont...

- 2. Asserts the **Master-ready** signal to indicate to slaves that the address and command information has been placed on the bus.
- 3. All devices on the bus **decode the address.**
- 4. **Addressed slave** performs the required operation, and informs the master it has done so by asserting the Slave-ready.
- 5. **Master removes all the signals from the bus, once Slave-ready is asserted.**

Cont...

⊙ 6. If the operation is a **Read operation**, Master also **strobes** the data into its input buffer.