# COMPUTER ORGANIZATION AND ARCHITECTURE (BCS302)

## UNIT-5 Input/Output

### Today's Target

- ALL TOPICS COVERED
- All AKTU PYQs
- Both 2 MARKS AND 7 MARKS COVERED

By PRAGYA RAJVANSHI

B.Tech, M.Tech( C.S.E)

# Computer Science & Engineering , Information Technology & Computer Science Allied

## Unit-V : Input / Output

**AKTU : Syllabus**

**Input / Output:** Peripheral devices, I/O interface, I/O ports, Interrupts: interrupt hardware, types of interrupts and exceptions. Modes of Data Transfer: Programmed I/O, interrupt initiated I/O and Direct Memory Access., I/O channels and processors. Serial Communication: Synchronous & asynchronous communication, standard communication interfaces.

# AKTU PYQs

| | | |
|---|---|---|
| Q1 | What are the function performed by input/output unit ? | AKTU 2022-23 |
| Q2 | Why DMA get priority over CPU when request memory transfer? (fast data transfer) | AKTU 2022-23 |
| Q3 | With neat schematic diagram , explain about the DMA controller and its mode of data transfer.<br>Or<br>Draw and explain the block diagram of typical DMA controller.<br>Or<br>What is DMA ? Describe how DMA IS used to transfer data from peripherals. | AKTU 2022-23<br>AKTU 2018-19<br>AKTU 2020-21<br>AKTU 2019-20 |
| Q4 | Discuss the design of typical input or output interface. | AKTU 2022-23<br>AKTU 2019-20 |
| Q5 | Explain how the computer buses can be used to communicate with memory and I/O .Also draw the block diagram for CPU-IOP Communication. | AKTU 2021-22 |
| Q6 | What are the different methods of asynchronous data transfer? Explain in detail? | AKTU 2021-22 |
| Q7 | Explain the destination-initiated transfer using handshaking method. | AKTU 2021-22 |
| Q8 | Describe the serial communication techniques in detail. Explain and compare the synchronous and asynchronous communication. | AKTU 2021-22 |

# AKTU PYQs....

| | | |
|---|---|---|
| Q9 | Explain the interrupt driven/o technique and design its flowchart in detail along with its merits and demerits. | AKTU 2021-22 |
| Q10 | Discuss interrupt processing. differentiate between the following<br>1. Trap vs exception<br>2. maskable and non-maskable interrupt | AKTU 2021-22 |
| Q11 | Explain cycle stealing. (DMA) | AKTU 2020-21<br>AKTU 2018-19 |
| Q12 | What do you means by asynchronous data transfer? Explain the strobe and hand shanking mechanism? | AKTU 2020-21<br>AKTU 2018-19 |
| Q13 | Discuss different modes of data transfer. (Progra, In, DMA) | AKTU 2020-21<br>AKTU 2018-19 |
| Q14 | What is i/o control? | AKTU 2019-20 |
| Q15 | What are interrupt? How they are handled? | AKTU 2019-20 |
| Q16 | Write down the difference isolated/o and memory i/o . Also discuss advantages and disadvantages of isolated/o and memory i/o. | AKTU 2019-20 |

# AKTU PYQs....

| | | |
|---|---|---|
| Q17 | Define the role of MIMD in computer architecture. 2 Marks | AKTU 2019-20 |
| Q18 | Explain the different vectored and non-vectored interrupt. Explain using example of each. | AKTU 2019-20 |
| Q19 | Differentiate between synchronous and asynchronous communication taking suitable example. | AKTU 2019-20 |
| Q20 | List and explain different modes of data of data transfer . | Aktu 2019-20 |
| Q21 | Explain the role of MIMD in computer architecture. | AKTU 2018-19 |
| Q22 | Give the block diagram of DMA controller? Why read and write lines in DMA controller is Bi- directional<br>0r<br>What is DMA based data transfer | AKTU 2018-19<br>AKTI2017-18<br>AKTU 2015-16 |
| Q23 | Difference between serial and parallel communication. | AKTU 2017-18 |

# AKTU PYQs

| | | |
|---|---|---|
| Q23 | DIFFERENCE BETWEEN-<br>➢ Strobe and handshaking asynchronous data transfer modes<br>➢ Processor and IOP<br>➢ Synchronous and asynchronous transmission<br>➢ DMA and interrupt initiated i/o interrupt | AKTU 2015-16 |
| Q24 | Why input output interface is required? | AKTU 2015-16 |
| Q25 | What is the use of modem in synchronous communication? | AKTU 2015-16 |
| Q26 | what is i/o bus architecture | AKTU 2018-19 |

Flynn's taxonomy

DMA

AKTU 2015-16

# Peripheral Devices:

- A peripheral device provides input/output (I/O) functions for a computer and serves as an auxiliary computer device without computing-intensive functionality
- Mouse, Keyboard, Printer, Monitor, Webcam etc
- FUNCTION OF INPUT-OUTPUT UNIT
- An input device sends information to a computer system for processing, and an output device reproduces or displays the results of that processing.

# Input - Output Interface

- Input Output Interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit.
- The purpose of communication links is to resolve the differences that exist between the central computer and each peripheral.

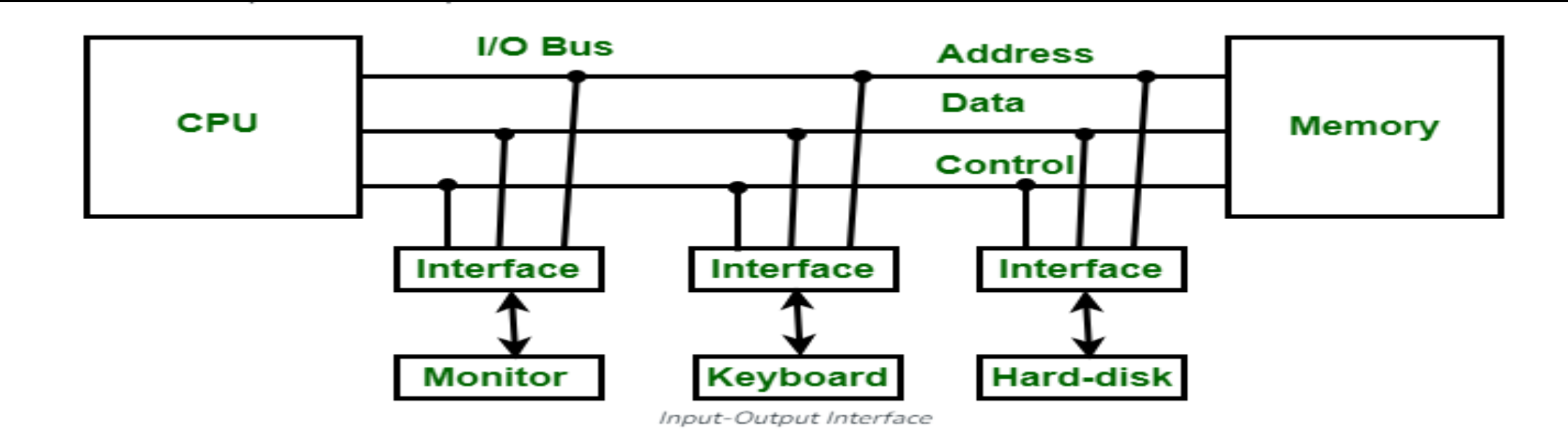
**The Major Differences are:-**

1. Peripherals are electro-mechanical and electromagnetic devices and CPU and memory are electronic devices. Therefore, a conversion of signal values may be needed.
2. The data transfer rate of peripherals is usually slower than the transfer rate of CPU and consequently, a synchronization mechanism may be needed.
3. Data codes and formats in the peripherals differ from the word format in the CPU and memory.

# Input - Output Interface

4. The operating modes of peripherals are different from each other and must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

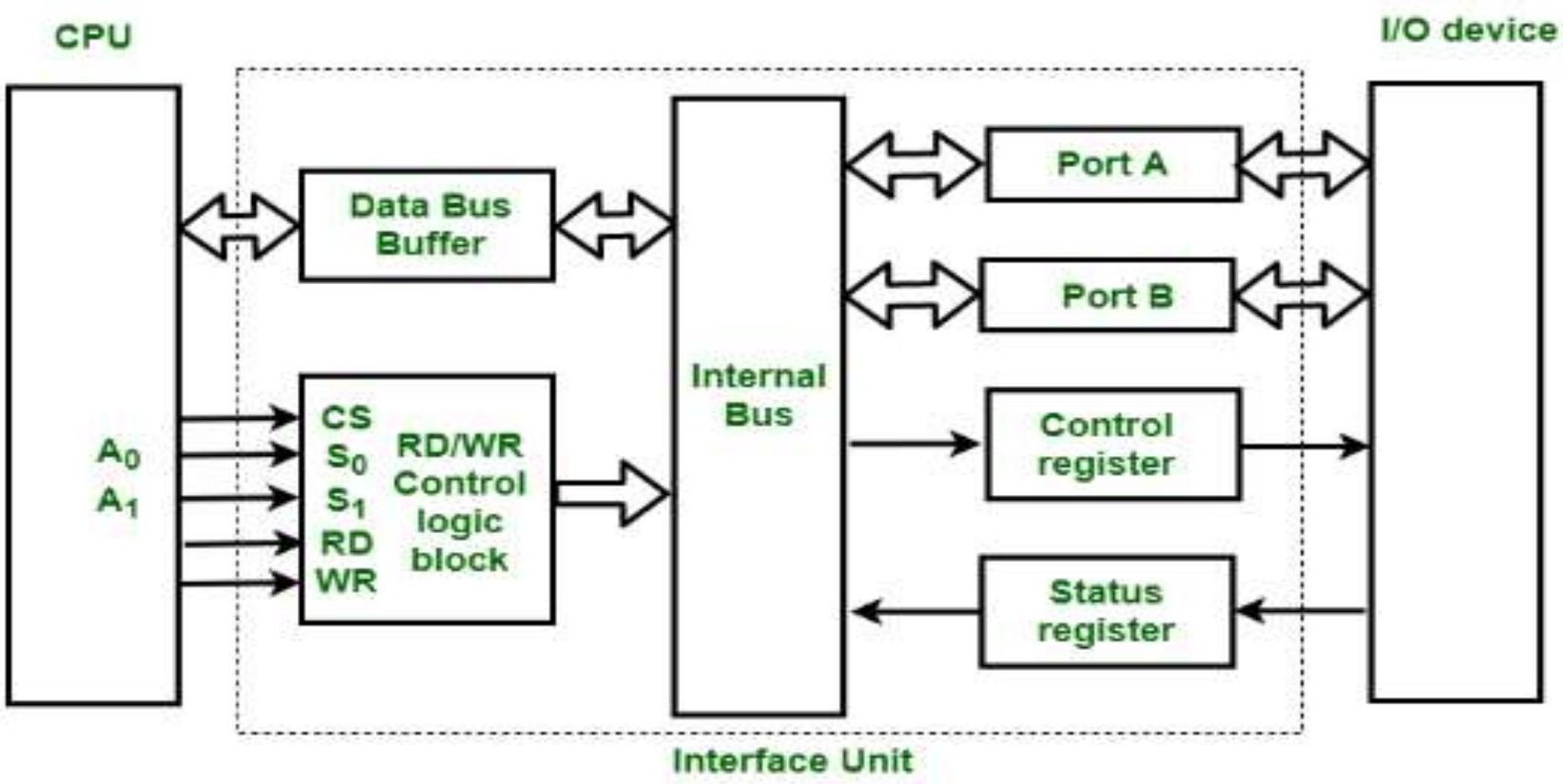
- To Resolve these differences, computer systems include special hardware components between the CPU and Peripherals to supervise and synchronize all input and out transfers These components are called Interface Units because they interface between the processor bus and the peripheral devices.



Input-Output Interface

# Structure of Input-Output Interface

## Components of a typical i/o device interface



# Structure of Input-Output Interface

**Port A and Port B :**

Port A and Port B are used to transfer data between the Input-Output device and the Interface Unit. Each port consists of a bi-directional data input buffer and a bi-directional data output buffer. The interface unit connects directly with an input device and output disk or with a device that requires both input and output through Port A and Port B i.e. modem, external hard drive, magnetic disk

**Control and Status Register :**

CPU gives control information to the control register based on control information. Interface unit control input and output operation between CPU and input-output device. Bits that are present in the status register are used for checking status conditions. The status register indicates the status of the data register, port A, and port B, and also records errors that may occur during the transfer of data.

# Structure of Input-Output Interface

## Read/Write Control Logic :

This block generates necessary control signals for overall device operations. All commands from the CPU are accepted through this block. It also allows the status of the interface unit to be transferred onto the data bus through this block to accept CS, read, and write control signals from the system bus and S0 , S1 from the system address bus. Read and Write signals are used to define the direction of data transfer over a data bus.

Read Operation: CPU <---- I/O device

Write Operation: CPU ----> I/O device

The read signals direct data transfer from the interface unit to the CPU and the writes signal direct data transfer from the CPU to the interface unit through the data bus.

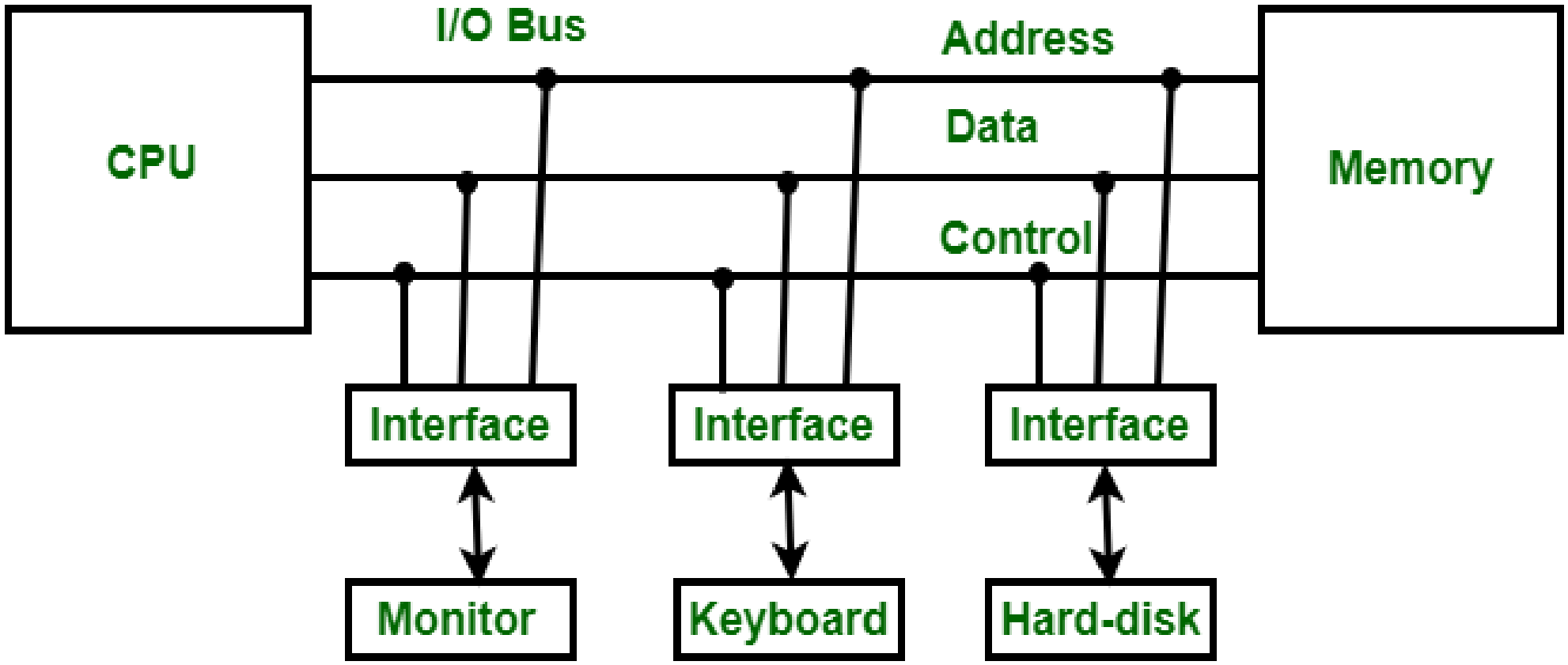
## Data Bus Buffer :

The bus buffer uses a bi-directional data bus to communicate with the CPU. All control word data and status information between the interface unit and CPU are transferred through the data bus.

# Purpose/Functions of Input-Output Interface:

- it is used to synchronize the operating speed of CPU with respect to input-output devices.
- It selects the input-output device which is appropriate for the interpretation of the input-output signal.
- It is capable of providing signals like control and timing signals.
- In this data buffering can be possible through data bus.
- There are various error detectors.
- It converts serial data into parallel data and vice-versa.
- It also convert digital data into analog signal and vice-versa.

# I/O BUS ARCHITECTURE




*Input-Output Interface*

# I/O BUS and I/O COMMAND

- The I/O Bus consists of data lines, address lines, and control lines.
- The I/O bus from the processor is attached to all peripherals interfaces.
- To communicate with a particular device, the processor places a device address on address lines.
- The control lines are referred to as I/O commands. The commands are as follows:

**Control command-** A control command is issued to activate the peripheral and to inform it what to do.

**Status command-** A status command is used to test various status conditions in the interface and the peripheral.

**Data Output command-** A data output command causes the interface to respond by transferring data from the bus into one of its registers.

**Data Input command-** The data input command is the opposite of the data output.

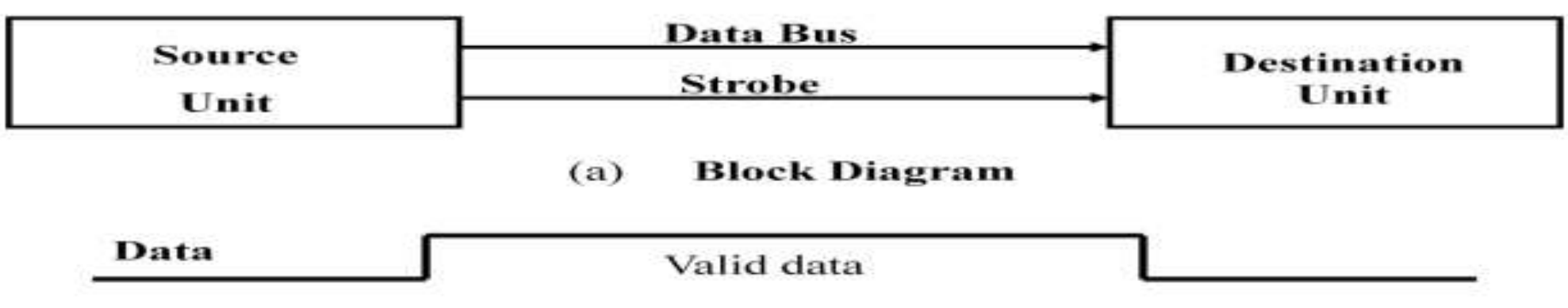
# Asynchronous Data Transfer :

- This Scheme is used when the speed of I/O devices does not match with a microprocessor, and the timing characteristics of I/O devices are not predictable. In this method, the process initiates the device and checks its status. As a result, the CPU has to wait till the I/O device is ready to transfer data. When the device is ready CPU issues instructions for I/O transfer. In this method, two types of techniques are used based on signals before data transfer.
- Strobe Control/signal
- Handshaking

# Strobe Signal

- The strobe control method of Asynchronous data transfer employs a single control line to each time transfer. The strobe may be activated by either the source or the destination unit.

**Data Transfer Initiated by Source Unit:**




Source-Initiated strobe for Data Transfer

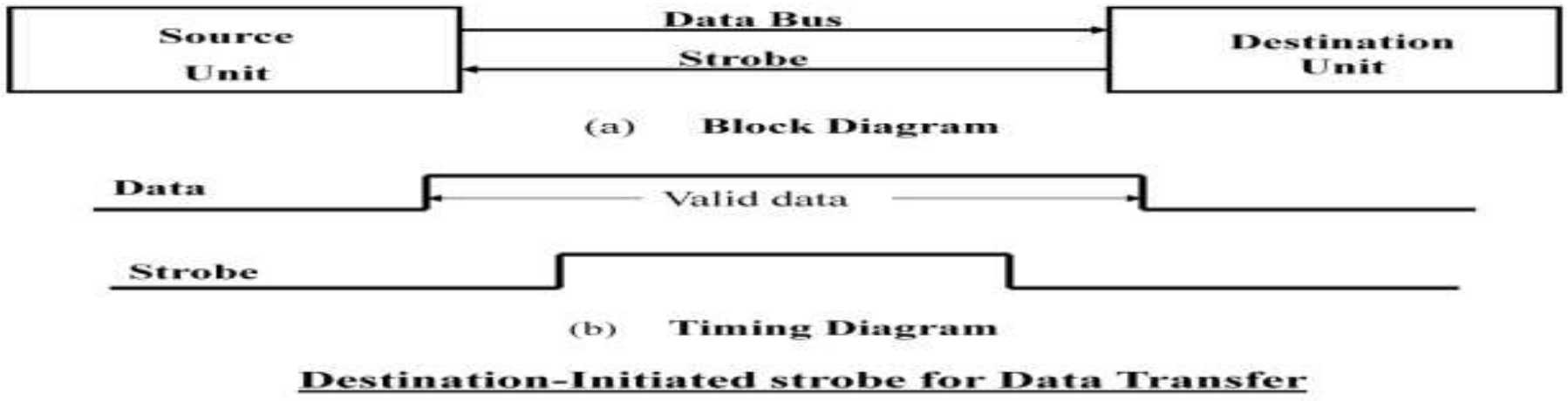
# Strobe Signal…

- In the block diagram fig. (a), the data bus carries the binary information from the source to the destination unit. Typically, the bus has multiple lines to transfer an entire byte or word. The strobe is a single line that informs the destination unit when a valid data word is available.
- In the timing diagram fig. (b) the source unit first places the data on the data bus. The information on the data bus and strobe signal remain in the active state to allow the destination unit to receive the data.

**Data Transfer Initiated by Destination Unit:**

- In this method, the destination unit activates the strobe pulse, to inform the source to provide the data. The source will respond by placing the requested binary information on the data bus.
- The data must be valid and remain in the bus long enough for the destination unit to accept it.
- When accepted the destination unit then disables the strobe and the source unit removes the data from the bus.

# Strobe Signal….




Destination-Initiated strobe for Data Transfer

## Disadvantage of Strobe Signal :

- The disadvantage of the strobe method is that, the source unit initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was places in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on bus. The Handshaking method solves this problem.

# Handshaking:

- The handshaking method solves the problem of the strobe method by introducing a second control signal that provides a reply to the unit that initiates the transfer.

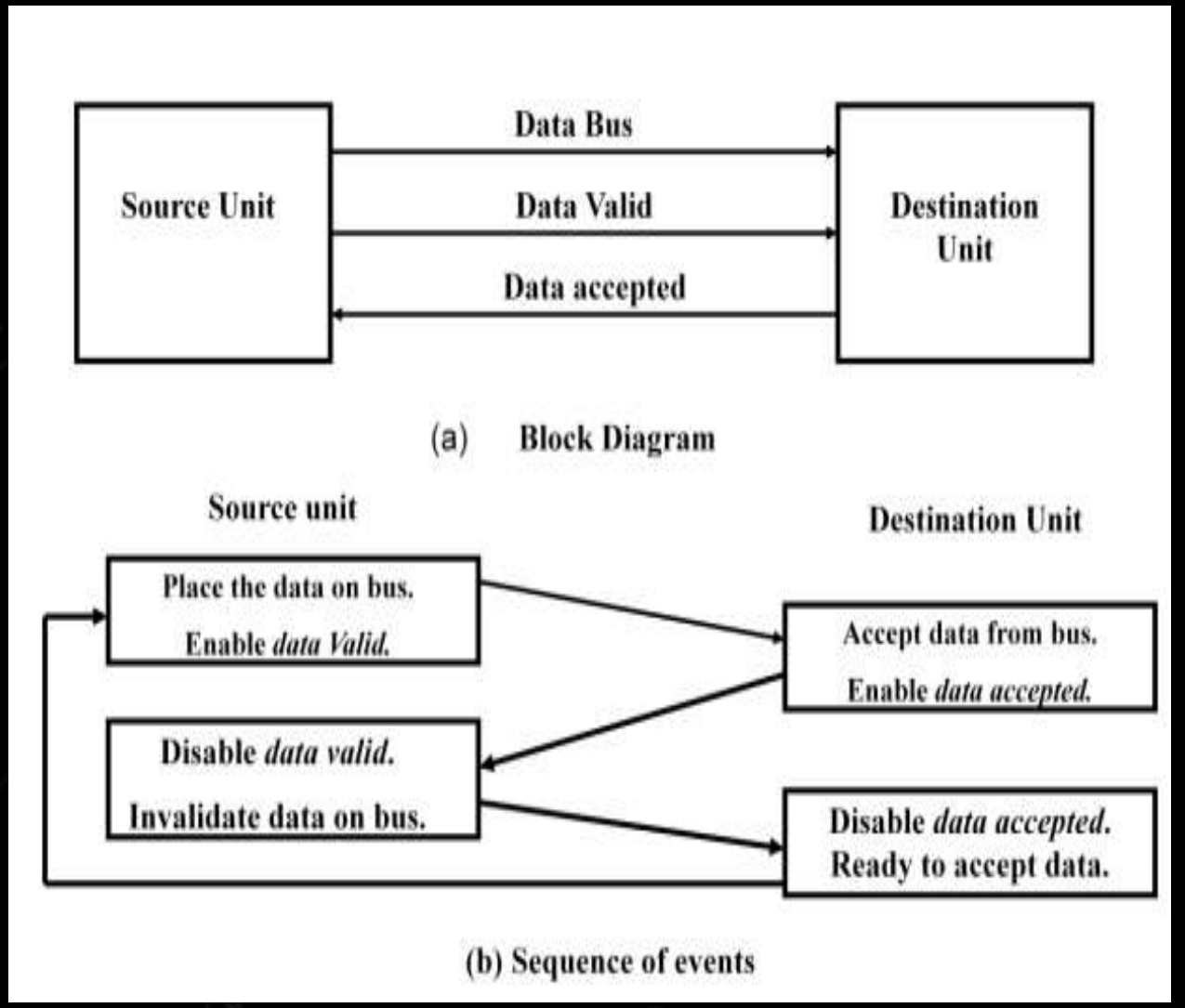
**Principle of Handshaking:**

- The basic principle of the two-wire handshaking method of data transfer is as follows:
- One control line is in the same direction as the data flows in the bus from the source to the destination.
- It is used by the source unit to inform the destination unit whether there a valid data in the bus.
- The other control line is in the other direction from the destination to the source.
- It is used by the destination unit to inform the source whether it can accept the data.
- The sequence of control during the transfer depends on the unit that initiates the transfer.

# Handshaking

## Source Initiated Transfer using Handshaking:

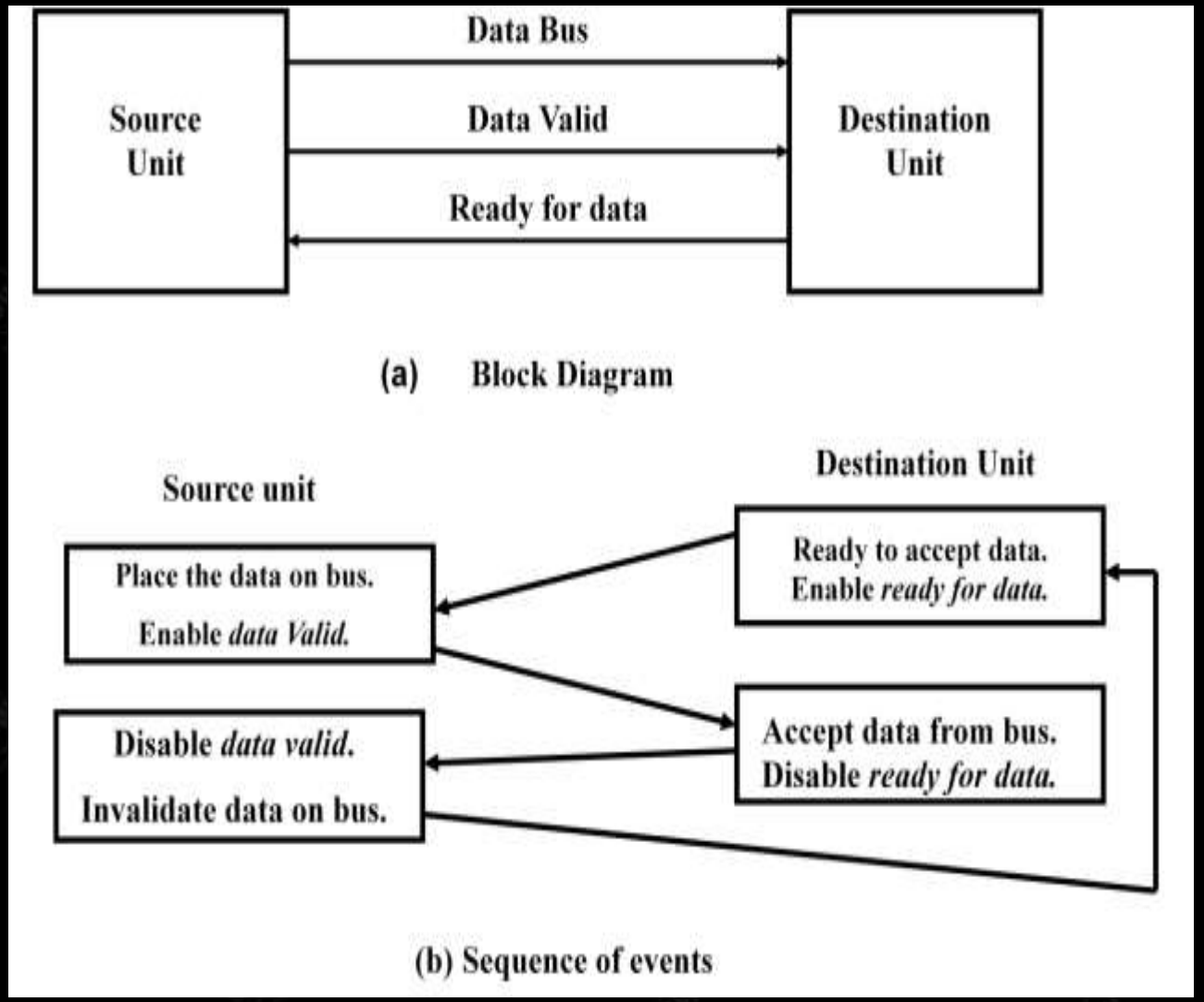
- The sequence of events shows four possible states that the system can be at any given time.
- The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal. The data accepted signal is activated by the destination unit after it accepts the data from the bus. The source unit then disables its data accepted signal and the system goes into its initial state..



# Handshaking…

**Destination Initiated Transfer Using Handshaking:**

- The name of the signal generated by the destination unit has been changed to ready for data to reflect its new meaning.
- The source unit in this case does not place data on the bus until after it receives the ready for data signal from the destination unit.
- From there on, the handshaking procedure follows the same pattern as in the source-initiated case.



# Handshaking….

- The only difference between the Source Initiated and the Destination Initiated transfer is in their choice of Initial state...

**Advantages of the Handshaking method:**

- The Handshaking scheme provides a degree of flexibility and reliability because the successful completion of data transfer relies on active participation by both units.
- If any of one unit is faulty, the data transfer will not be completed. Such an error can be detected by means of a Timeout mechanism which provides an alarm if the data is not completed within time.

# Modes of Data Transfer

- Transfer of data is required between CPU and peripherals or memory or sometimes between any two devices or units of your computer system. To transfer data from one unit to another one should be sure that both units have a proper connection and at the time of data transfer the receiving unit is not busy. This data transfer with the computer is Internal Operation.
- All the internal operations in a digital system are synchronized by means of clock pulses supplied by a common clock pulse Generator. The data transfer can be

i. Synchronous or

ii. Asynchronous

- When both the transmitting and receiving units use the same clock pulse then such a data transfer is called a Synchronous process. On the other hand, if there is no concept of clock pulses and the sender operates at a different moment than the receiver then such a data transfer is called Asynchronous data transfer.

# Modes of Data Transfer….

- The data transfer can be handled by various modes. some of the modes use the CPU as an intermediate path, while others transfer the data directly to and from the memory unit and this can be handled in 3 following ways:

i. Programmed I/O

ii. Interrupt-Initiated I/O

iii. Direct Memory Access (DMA)

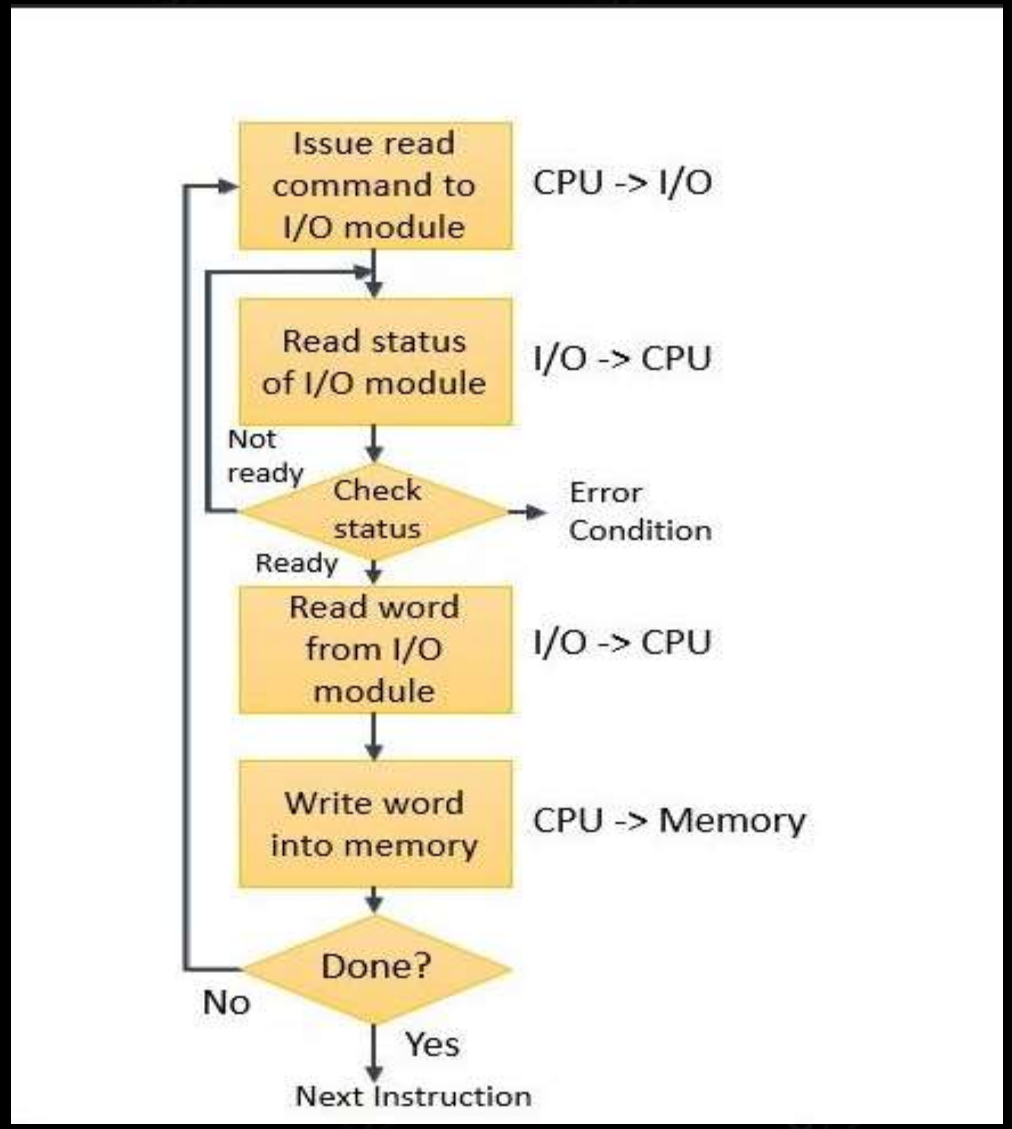
# Programmed I/O

- The programmed I/O was the most simple type of I/O technique for the exchanges of data or any types of communication between the processor and the external devices.
- With programmed I/O, data are exchanged between the processor and the I/O module.
- The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.
- When the processor issues a command to the I/O module, it must wait until the I/O operation is complete. If the processor is faster than the I/O module, this is wasteful of processor time.

# Programmed I/O

- Consider the situation that the processor is busy executing any program. Meanwhile, it encounters an *I/O instruction*. To execute the encountered instruction the process supply an appropriate *I/O command* to the corresponding I/O module. Accepting the issued command, the I/O module performs the desired task and sets some appropriate bits of its I/O status registers.
- Further, the I/O module does not notify the processor that it has performed the desired task. Moreover, it's the processors' responsibility to periodically check the status of the I/O module till it finds that the I/O has successfully completed the desired task.

# Programmed I/O Mode Input Data Transfer




➢you have observed the function of the programmed I/O it involves two things the I/O command that is provided by the processor to the I/O module and the I/O instruction that is encountered and executed by the processor. Let us discuss these two things.

➢**I/O Commands**

➢Whenever the processor experience the I/O related instruction, to execute this I/O instruction the processor issues two things I/O command and address on the bus which is decoded by every I/O module connected to the system. Whichever I/O module is addressed by the processor recognizes this address respond to the issued I/O command.

# i/o commands

The processor issue the I/O commands to the I/O module can be of four types.

- **Control:** This I/O command activates the I/O module addressed by the processor and directs it to the task it has to perform. This command can be customized depending upon the type of peripherals.
- **Test:** This I/O command tests the status of the I/O module and its peripherals to ensure that the addressed peripheral is powered on and available for the task. This command also tests whether the most recent I/O operation has completed successfully or if any error has occurred.
- **Read:** This I/O command lets the I/O module extract the data from the corresponding peripheral and store it in its internal buffer. Further, the I/O module can place this data over the data bus on the processor's demand.
- **Write:** This I/O command lets the I/O module accept the data over the data bus and transmit it to the corresponding peripheral.

# I/O Instructions

The I/O instruction encountered by the processor is issued to the processor by the main memory. And to execute this I/O instruction the processor provides the I/O command to the corresponding I/O device.

Thereby the I/O instruction cab is simply mapped onto the I/O command. Usually, there is a simple one-to-one relationship between I/O instruction and I/O command.

The I/O instruction can also be customized depending on the peripherals addressed,

when the processor, main memory, and I/O module share the common bus then addressing can be achieved in two ways memory-mapped I/O and isolated I/O.

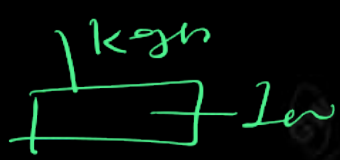
With the **memory-mapped I/O,** the processor access memory and I/O using a single address space. Here the processor uses the same address, data, and control bus. So, the same set of machine instructions addresses both memory and I/O.

With **isolated I/O** the address space for memory is isolated from the address space of I/O. Though the processor uses the same data and address line for memory and I/O devices it uses a separate control line for memory and I/O devices.

# Interrupt Driven I/O

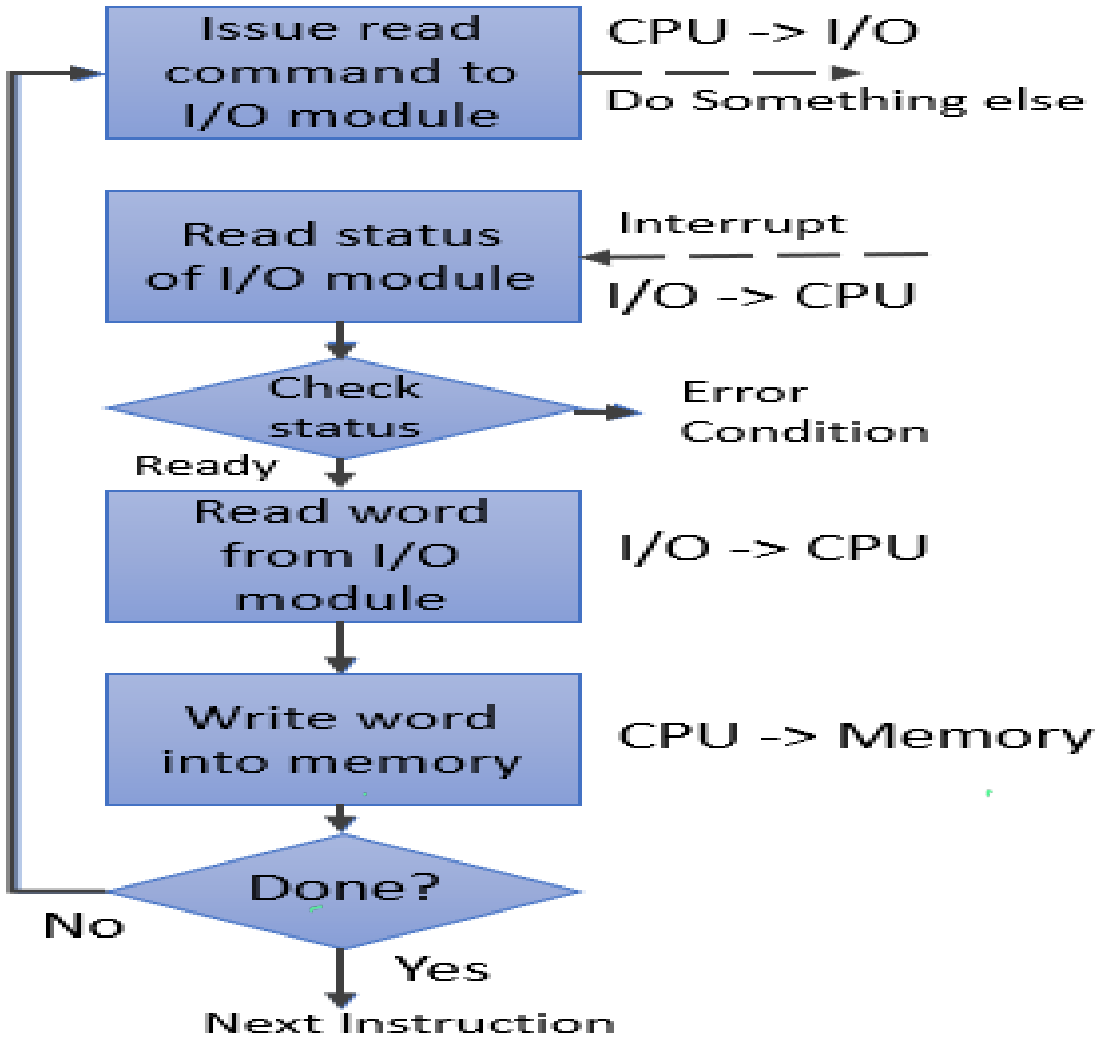
- Interrupt-driven I/O is an approach to transfer data between ‘memory’ and ‘I/O devices’ through the ‘processor’. The other two techniques for the same are programmed I/O and direct memory access (DMA). The interrupt-driven I/O involves the use of interrupt to exchange data between I/O and memory.
- programmed I/O we have seen it is a processor who keeps on checking whether the I/O module is ready for reception and transmission of data or whether the I/O module has completed the desired task or not. This long waiting of the processor deteriorates the performance of the system.
- To improve the performance of the system an alternative approach can be used where after issuing the I/O command to the I/O module the processor can get itself busy doing some other work. In this way, the valuable time of the processor can be utilized

# Interrupt Driven I/O



- Consider that the data has to be stored in the main memory from the I/O module as input from the I/O module's point of view.

1. For this, the processor issues a READ I/O command to the corresponding I/O module and proceeds with some other useful tasks. It does not wait for the I/O module to get ready with the desired data.
2. The I/O module then processes this READ I/O command and reads the data from the addressed peripheral device. The I/O module stores the read data into its data register and issues an interrupt signal to the processor over the control line in the system bus. By sending the interrupted signal, the I/O module indicates to the processor that now it is ready to transmit the data. But, the I/O module has to wait until the processor asks for the data from the I/O module.
3. When the processor requests the data from the I/O, it places the data over the data line of the system bus. Once the I/O module transfers the data to the processor it sets itself ready for another I/O transfer.




Interrupted I/O to Transfer Data from I/O Module to Memory

# Interrupt Driven I/O

## Advantages

- system can do useful work when the device is not ready.

## Disadvantages

- lots of interrupts
- interrupts are expensive to do
- Have to run interrupts code segment every time

Diffn Mods Deti

# Difference between- (2 MARKS)

| Programmed i/o | Interrupt driven i/o |
|---|---|
| Data transfer is initiated by the means of instructions stored in the computer program. Whenever there is a request for I/O transfer the instructions are executed from the program. | The I/O transfer is initiated by the interrupt command issued to the CPU. |
| The CPU stays in the loop to know if the device is ready for transfer and has to continuously monitor the peripheral device. | There is no need for the CPU to stay in the loop as the interrupt command interrupts the CPU when the device is ready for data transfer. |
| This leads to the wastage of CPU cycles as CPU remains busy needlessly and thus the efficiency of system gets reduced. | The CPU cycles are not wasted as CPU continues with other work during this time and hence this method is more efficient. |

# Difference between-

| Programmed i/o | Interrupt driven i/o |
|---|---|
| CPU cannot do any work until the transfer is complete as it has to stay in the loop to continuously monitor the peripheral device. | CPU can do any other work until it is interrupted by the command indicating the readiness of device for data transfer |
| Its module is treated as a slow module. | Its module is faster than programmed I/O module. |
| It is quite easy to program and understand. | It can be tricky and complicated to understand if one uses low level language. |
| The performance of the system is severely degraded. | The performance of the system is enhanced to some extent. |

# How interrupt handle by processor

1. The I/O device completes the task that the processor has issued to it through the I/O command and then it signals an interrupt to the processor.
2. The processor executes its current instruction and then check for the interrupt.
3. When the processor determines an interrupt ,it signals the corresponding I/O device with an acknowledgment that it has received an interrupt. Receiving the acknowledgment from the processor, the I/O device withdraw its interrupt signal.
4. The processor now has to transfer the control to the interrupt service routine. But before switching the control it has to save the information it requires to resume the execution of the program it was working on at the point of interrupt. The least information that the processor will require to resume its work at the point of interrupt is the current status of the processor and the address of the next instruction to be executed. The processor saves its status in the register program status word (PSW) and the address of the next instruction to be executed in the program counter. The processor put this information onto the system control stack.

# How interrupt handle by processor

5. The processor loads the program counter with the address of the entry location of the appropriate interrupt service routine that will serve the occurred interrupt.

6. Well depending on the computer architecture and the type of operating system you have there may be a single interrupt handling program for all types of interrupt or there may be a different program for each type of interrupt.

7. In the case of several interrupt-handling programs the processor has to determine which interrupt-handling routine it has to invoke and this information is present in the interrupt signal issued to the processor by the I/O module. Once this information is acquired the processor starts the execution of the program service routine.

8. The interrupt service routine serves the occurred interrupt so that the processor checks the status of the I/O device that signals the interrupt or the event that caused the interrupt. While serving the interrupt the processor may send some additional commands or acknowledge to the I/O device.

# How interrupt handle by processor

9. When the interrupt is processed completely the processor restores its register values from the stack.

10. Finally, the values of the program counter and program status word are restored from the stack and resume the execution of the interrupted program.

# Interrupt and its types

The interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process. In I/O devices one of the bus control lines is dedicated for this purpose and is called the Interrupt Service Routine (ISR).

→ Hardware & Software
→ Maskable & Non-Maskable
→ Vectored Interrupt & Non-Vectored Interrupt

# Difference between

| S.No. | SOFTWARE INTERRUPT | HARDWARE INTERRUPT |
|---|---|---|
| 1 | A sort of interrupt called a software interrupt is one that is produced by software or a system as opposed to hardware. Traps and exceptions are other names for software interruptions. They serve as a signal for the operating system or a system service to carry out a certain function or respond to an error condition. | If the signal for the processor is from external device or hardware is called hardware interrupts. Example: from keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts. Hardware interrupts can be classified into two types they are |
| 2 | A particular instruction known as a “interrupt instruction” is used to create software interrupts. When the interrupt instruction is used, the processor stops what it is doing and switches over to a particular interrupt handler code. The interrupt handler routine completes the required work or handles any errors before handing back control to the interrupted application. | They are of two interrupt-<br>Maskable interrupt<br>Non-maskable interrupt |

# Difference between

| Aspect | Vectored Interrupts | Non-Vectored Interrupts |
|---|---|---|
| Definition | Provides direct information about interrupt source | Does not provide direct information about interrupt source |
| Functionality | Jump to specific interrupt handler routine | Iterate through interrupt sources based on priority |
| Handling Time | Faster and more efficient handling | Potential delays in handling due to iteration |
| Prioritization | Directly identifies highest-priority interrupt | Relies on predefined priority scheme |
| Implementation Complexity | Requires complex hardware and software mechanisms | Relatively simpler implementation |

# Difference between

| S.No. | MASKABLE INTERRUPT | NON-MASKABLE INTERRUPT |
|---|---|---|
| 1 | Maskable interrupt is a hardware Interrupt that can be disabled or ignored by the instructions of CPU. | A non-maskable interrupt is a hardware interrupt that cannot be disabled or ignored by the instructions of CPU. |
| 2 | When maskable interrupt occur, it can be handled after executing the current instruction. | When non-maskable interrupts occur, the current instructions and status are stored in stack for the CPU to handle the interrupt. |
| 3 | Maskable interrupts help to handle lower priority tasks. | Non-maskable interrupt help to handle higher priority tasks such as watchdog timer |
| 4 | In maskable interrupts, response time is high. | In non maskable interrupts, response time is low. |
| 5 | It may be vectored or non-vectored. | All are vectored interrupts. |

# NOTE

- In **programmed I/O,** the processor keeps on scanning whether any device is ready for data transfer. If an I/O device is ready, the processor **fully dedicates** itself in transferring the data between I/O and memory. It transfers data at a **high rate, but it can't get involved in any other activity** during data transfer. This is the major **drawback** of programmed I/O.
- In **Interrupt driven I/O,** whenever the device is ready for data transfer, then it raises an **interrupt to processor**. Processor completes executing its ongoing instruction and saves its current state. It then switches to data transfer which causes a **delay**. Here, the processor doesn't keep scanning for peripherals ready for data transfer. But, it is **fully involved** in the data transfer process. So, it is also not an effective way of data transfer.
- The above two modes of data transfer are not useful for transferring a large block of data. But, the DMA controller completes this task at a faster rate and is also effective for transfer of large data block.

# DMA

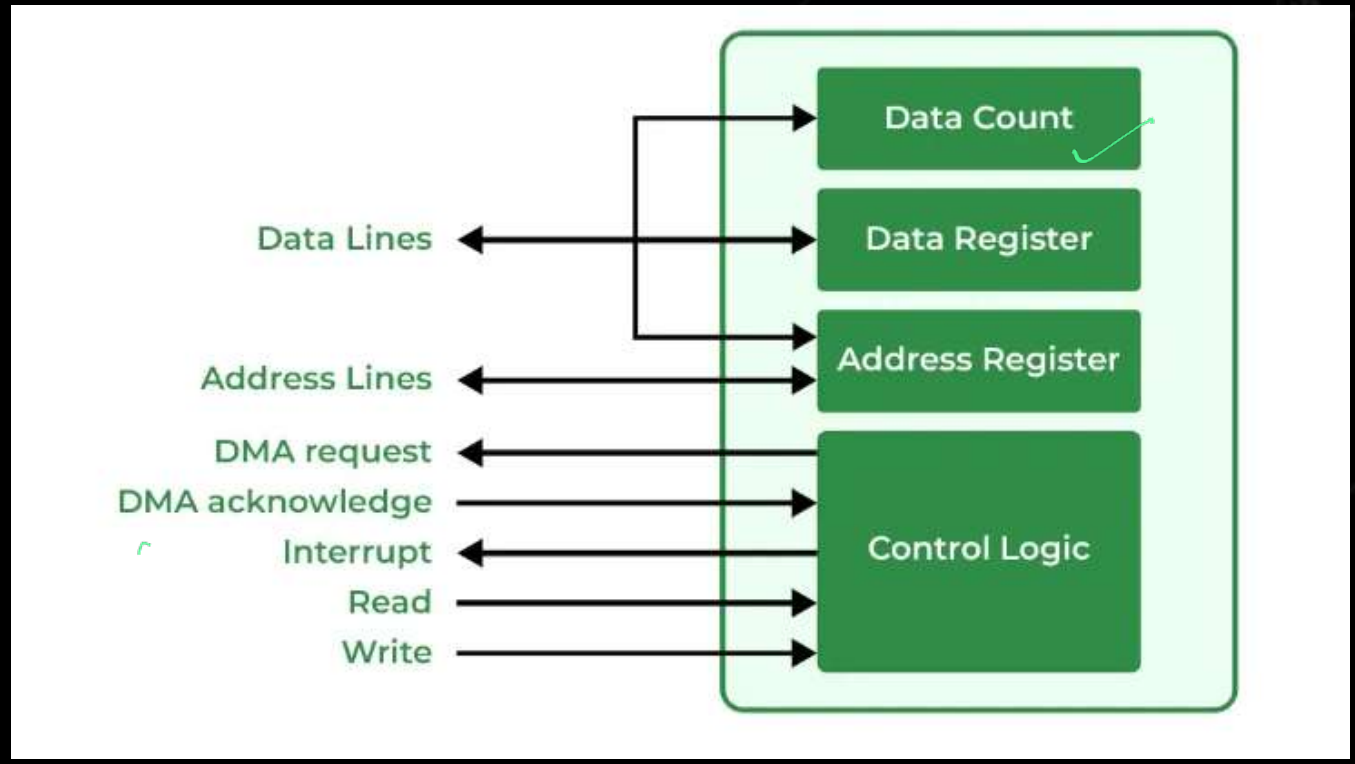
- Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations.
- The process is managed by a chip known as a DMA controller (DMAC).

## What is a DMA Controller?

- Direct Memory Access uses hardware for accessing the memory, that hardware is called a DMA Controller. It has the work of transferring the data between Input Output devices and main memory with very little interaction with the processor. The direct Memory Access Controller is a control unit, that the work to transfer data

# DMA Controller Diagram in Computer Architecture

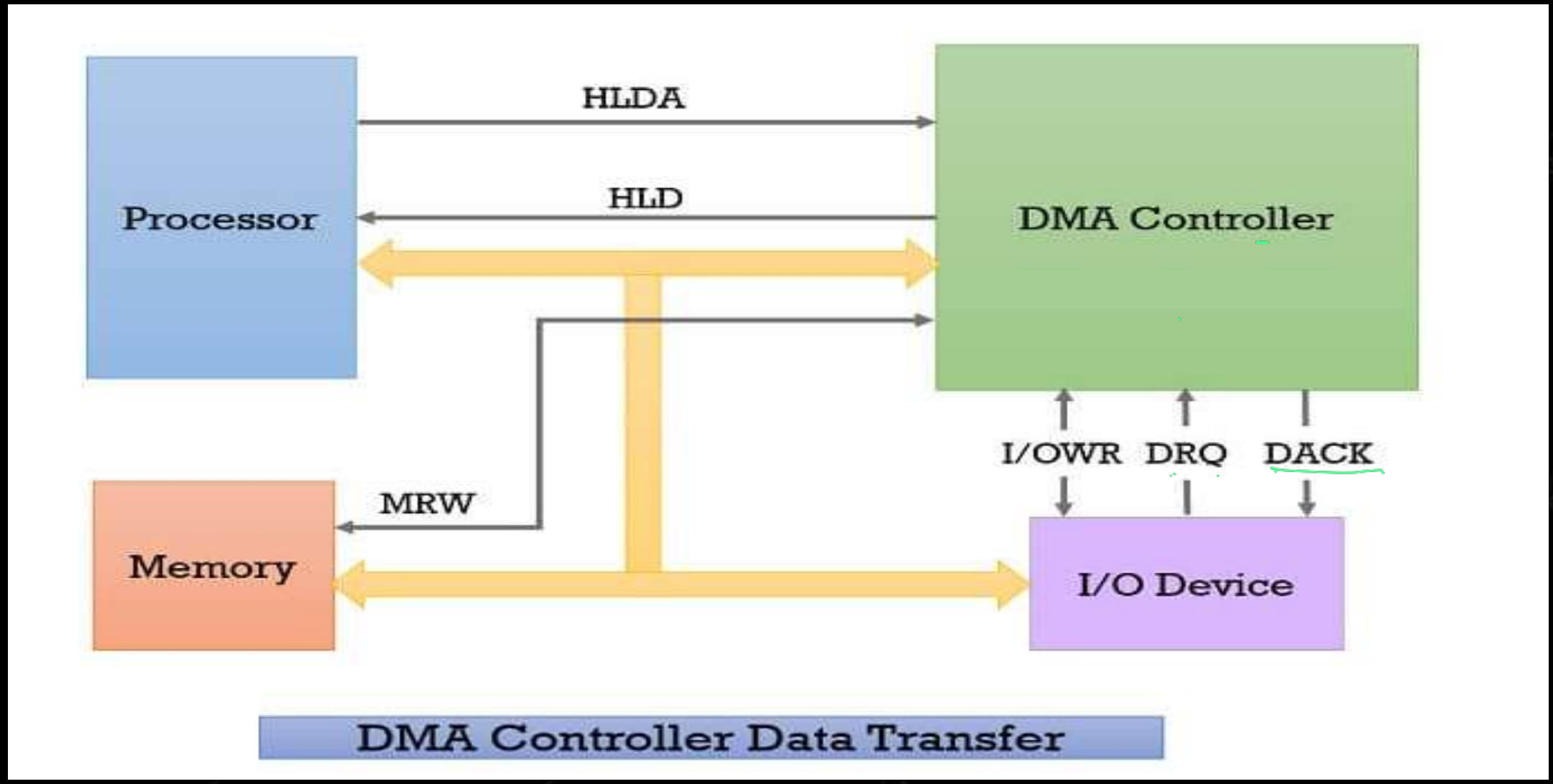
- DMA Controller is a type of control unit that works as an interface for the data bus and the I/O Devices. As mentioned, DMA Controller has the work of transferring the data without the intervention of the processors, processors can control the data transfer.
- DMA Controller also contains an address unit, which generates the address and selects an I/O device for the transfer of data. Here we are showing the block diagram of the DMA Controller.



# DMA Controller

- The DMA controller registers have three registers as follows.
- Address register – It contains the address to specify the desired location in memory.
- Word count register – It contains the number of words to be transferred.
- Control register – It specifies the transfer mode.
- Note: All registers in the DMA appear to the CPU as I/O interface registers. Therefore, the CPU can both read and write into the DMA registers under program control via the data bus.
- The figure below shows the block diagram of the DMA controller. The unit communicates with the CPU through the data bus and control lines. Through the use of the address bus and allowing the DMA and RS register to select inputs, the register within the DMA is chosen by the CPU. RD and WR are two-way inputs. When BG (bus grant) input is 0, the CPU can communicate with DMA registers. When BG (bus grant) input is 1, the CPU has relinquished the buses and DMA can communicate directly with the memory..

# Direct Memory Access Controller & it's Working



# DMA Controller data transfer( working)

- DMA controller is a **hardware unit** that allows I/O devices to access memory directly without the participation of the processor. Here, we will discuss the working of the DMA controller...
- DMA controller. DMA controller accepts this DRQ and asks the CPU to hold for a few clock cycles by sending it the Hold request (**HLD**).
- CPU receives the Hold request (HLD) from DMA controller and relinquishes the bus and sends the Hold acknowledgement (**HLDA**) to DMA controller.
- After receiving the Hold acknowledgement (HLDA), DMA controller acknowledges I/O device **(DACK)** that the data transfer can be performed and DMA controller takes the charge of the system bus and transfers the data to or from memory.
- When the data transfer is accomplished, the DMA raise an **interrupt** to let know the processor that the task of data transfer is finished and the processor can take control over the bus again and start processing where it has left

# Modes of Data Transfer in DMA

The DMA controller transfers the data in three modes

**Mode-1: Burst Mode –**

- In this mode, the Burst of data (entire data or burst of block containing data) is transferred before the CPU takes control of the buses back from DMAC.
- This is the quickest mode of DMA Transfer since at once a huge amount of data is being transferred.
- Since at once only a huge amount of data is being transferred so time will be saved in huge amounts.
- once the DMA controller gains the charge of the system bus, then it releases the system bus only after **completion** of data transfer. Till then the CPU has to wait for the system bus.

**Pros:**

- Fastest mode of DMA Transfer

**Cons:**

- Less user friendly because during the DMA transfer CPU will be blocked.

# Modes of Data Transfer in DMA...

## Mode-2: Cycle Stealing Mode – (2 Marks)

- In this mode, the DMA controller **forces** the CPU to stop its operation and **relinquish the control over the bus** for a **short term** to DMA controller. After the **transfer of every byte**, the DMA controller **releases** the **bus** and then again requests for the system bus. In this way, the DMA controller steals the clock cycle for transferring every byte
- Slow IO device will take some time to prepare data (or words) and within that time CPU keeps the control of the buses.
- Once the data or the word is ready CPU gives back control of system buses to DMAC for 1-cycle in which the prepared word is transferred to memory.
- As compared to Burst mode this mode is a little bit slowest since it requires a little bit of time which is consumed by the IO device while preparing the data.

# Modes of Data Transfer in DMA…

## Mode-2: Cycle Stealing Mode –

**Pros:**

- Most Efficient way for DMA Transfer.
- CPU won't be blocked the entire time.

**Cons:**

- The rate of DMA Transfer will be less.

# Modes of Data Transfer in DMA....

## Mode-3: Interleaving Mode –

- Whenever the CPU does not require the system buses then only control of buses will be given to DMAC.
- In this mode, the CPU will not be blocked due to DMA at all.
- This is the slowest mode of DMA Transfer since DMAC has to wait for so long time to just even get access of system buses from the CPU itself.
- Hence less amount of data will be transferred.

**Pros:**

- CPU will not be blocked at all.

**Cons:**

- Slowest DMA transfer rate.

# Modes of Data Transfer in DMA

## Mode-3: Interleaving Mode /transparent mode–

- Whenever the CPU does not require the system buses then only control of buses will be given to DMAC.
- In this mode, the CPU will not be blocked due to DMA at all.
- This is the slowest mode of DMA Transfer since DMAC has to wait for so long time to just even get access to system buses from the CPU itself.
- Hence less amount of data will be transferred.

**Pros:**

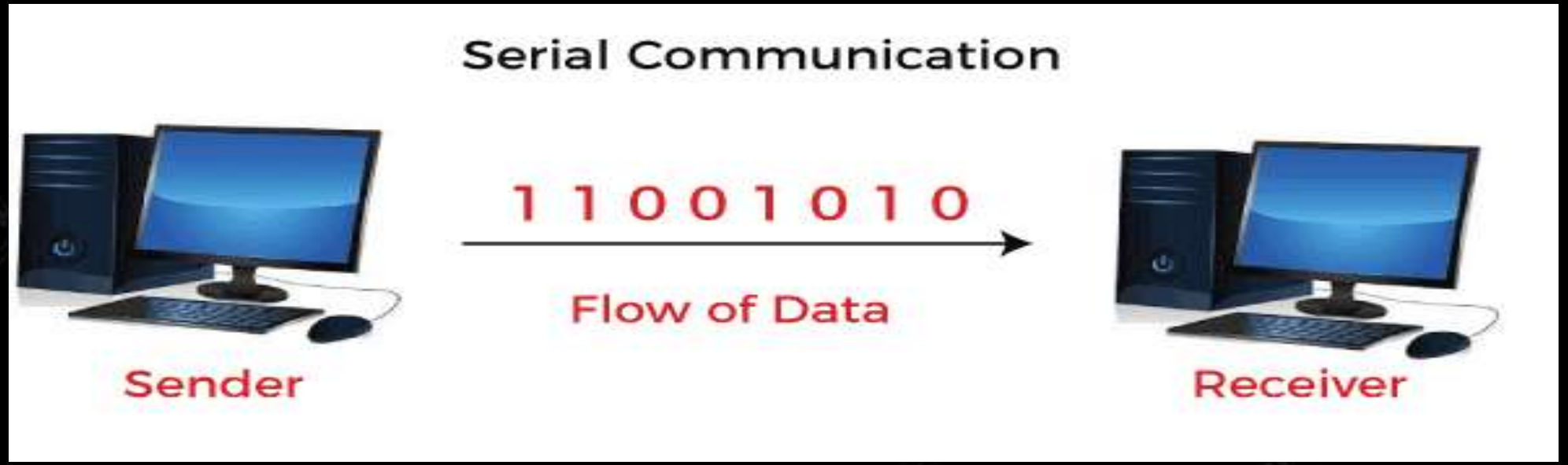
- CPU will not be blocked at all.

**Cons:**

- Slowest DMA transfer rate.

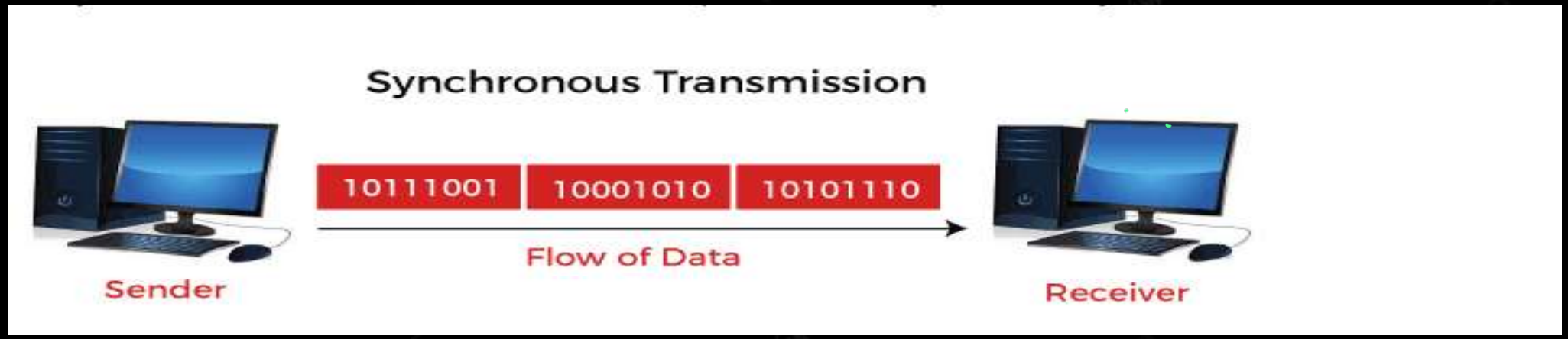
# SERIAL COMMUNICATION

- Serial communication is the process of sequentially transferring the information/bits on the same channel. Due to this, the cost of wire will be reduced, but it slows the transmission speed. Generally, communication can be described as the process of interchanging information between individuals in the form of audio, video, verbal words, and written documents.
- Binary contains the two numbers 0 and 1. 0 is used to show the LOW or 0 Volts, and 1 is used to show the HIGH or 5 Volts. The serial communication can either be asynchronous or synchronous.



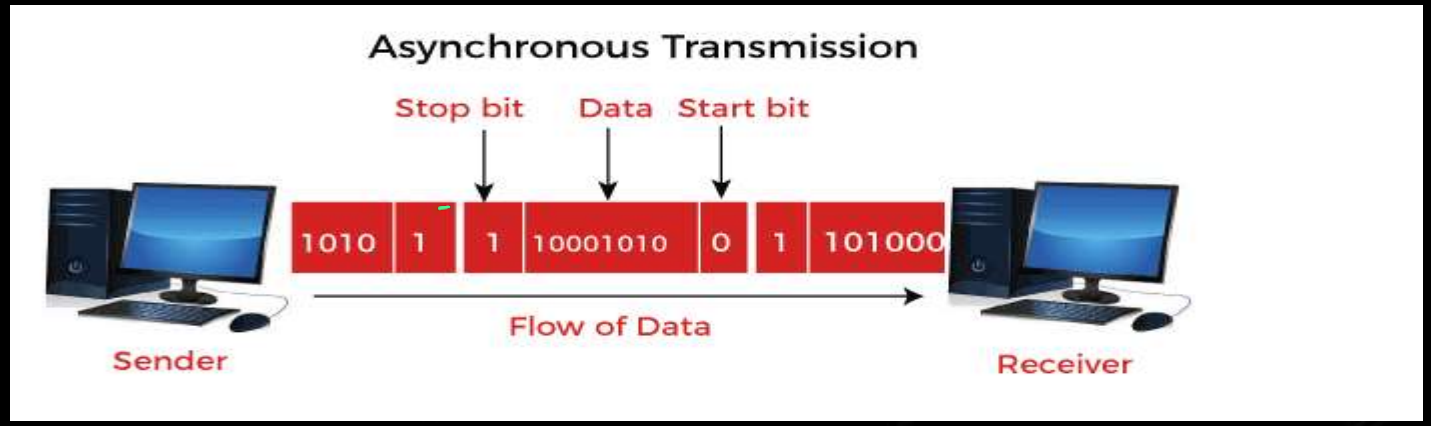
# Synchronous Communication

- In synchronous communication, the frames or data will be constructed with the help of combining the groups of bits. Those frames will be continuously sent in time with a master clock. It uses a synchronized clock frequency to operate the data of the sender or receiver.
- In synchronous communication, there is no need to use the gaps, start bits, and stop bits. The time taken by the sender and receiver is synced based on the timing being synced correctly between the sender and receiver devices, the data accuracy is dependent.
- The synchronous serial transmission is more expensive as compared to asynchronous serial transmission.



# Asynchronous Communication

- In asynchronous communication, the groups of bits will be treated as an independent unit, and these data bits will be sent at any point in time. In order to make synchronization between sender and receiver, the stop bits and start bits are used between the data bytes.
- These bits are useful to ensure that the data is correctly sent. The time taken by data bits of sender and receiver is not constant, and the time between transmissions will be provided by the gaps. In asynchronous communication, we don't require synchronization between the sender and receiver devices, which is the main advantage of asynchronous communication. This method is also cost-effective. In this method, there can be a case when data transmission is slow, but it is not compulsory, and it is the main disadvantage of the asynchronous method.



# Transmission Mode

## 1. Simplex

- In the simplex method, the data transmission can be performed only in one direction. At a time, only one client (either sender or receiver) can be active. That means among the two devices, one device can only transmit a link while the other device can only receive it. A sender can only transmit the data, and the receiver can only accept that data. The receiver cannot reply back to the sender.
- **Example 1:** Keyboard and CPU

## 2. half-duplex

walkitakki

- the sender and receiver can communicate in both directions, but not at the same time. If a sender sends some data, the receiver is able to accept it, but at that time, the receiver cannot send anything to the receiver. Same as if the receiver sends data to the sender, the sender cannot send. If there is a case where we don't need to communicate at a time in both the direction, we can use the half-duplex

# Transmission Mode

## 3.FULL DUPLEX

- In the full-duplex, the sender and the receiver are able to send and receive at the same time. The communication mode of full-duplex is widely used in the world

## Why DMA get priority over CPU when request memory transfer

- Because the CPU always has access to the memory bus when a DMA transfer is not taking place, DMA appears to have priority. DMA operates by "cycle stealing" memory bus access cycles from the CPU.
- DMA (Direct Memory Access) allows a hardware device to access system memory without any help from the CPU (Central Processing Unit). By doing so a device can transfer data to/from memory much faster than through the CPU

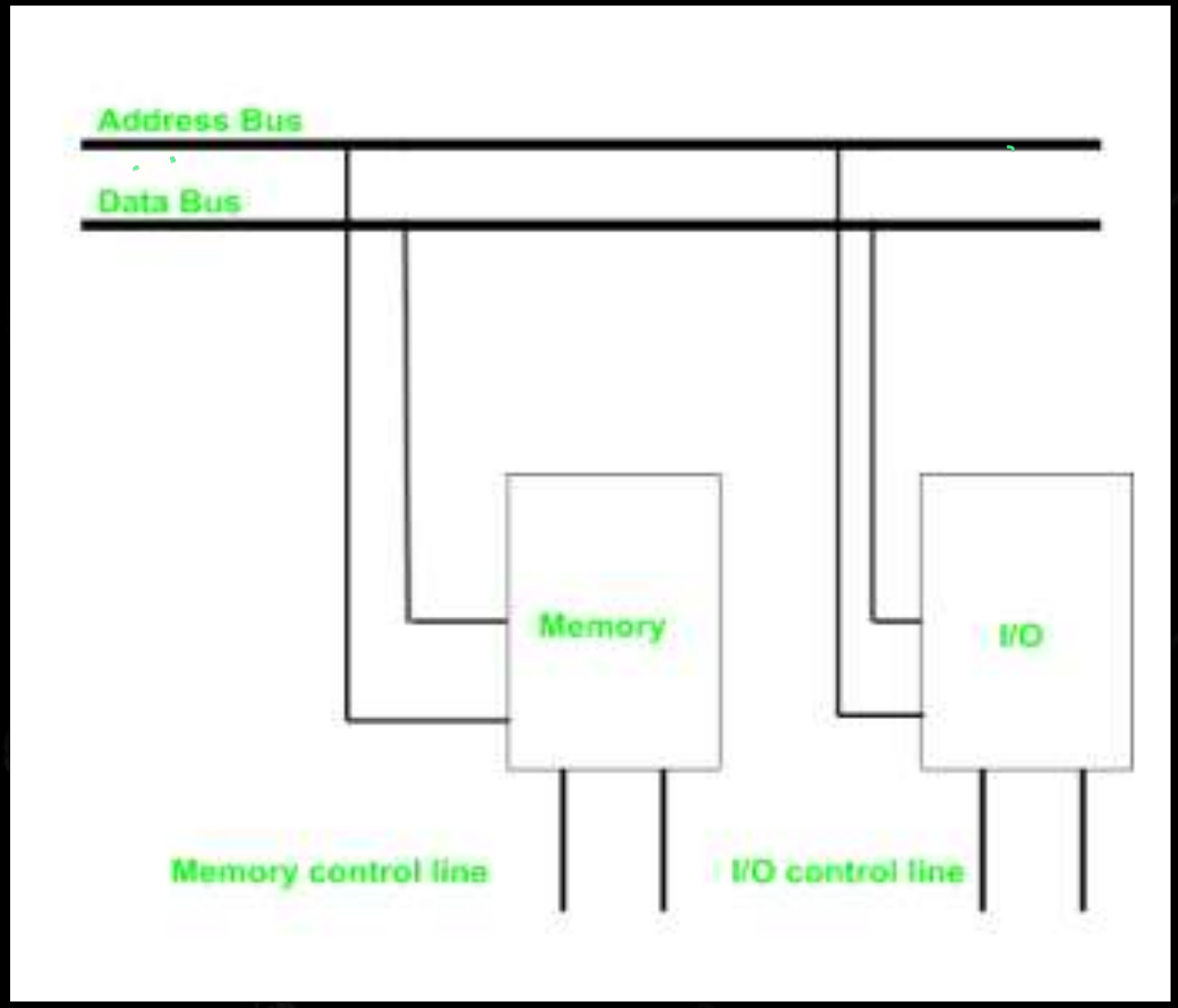
# Explain how the computer buses can be used to communicate with memory and I/O

Imp 7 Marks

- As a CPU needs to communicate with the various memory and input-output devices (I/O) as we know data between the processor and these devices flow with the help of the system bus. There are three ways in which system bus can be allotted to them :
- Separate set of address, control and data bus to I/O and memory.
- Have common bus (data and address) for I/O and memory but separate control lines.
- Have common bus (data, address, and control) for I/O and memory.
- In first case it is simple because both have different set of address space and instruction but require more buses.

# Explain how the computer buses can be used to communicate with memory and I/O

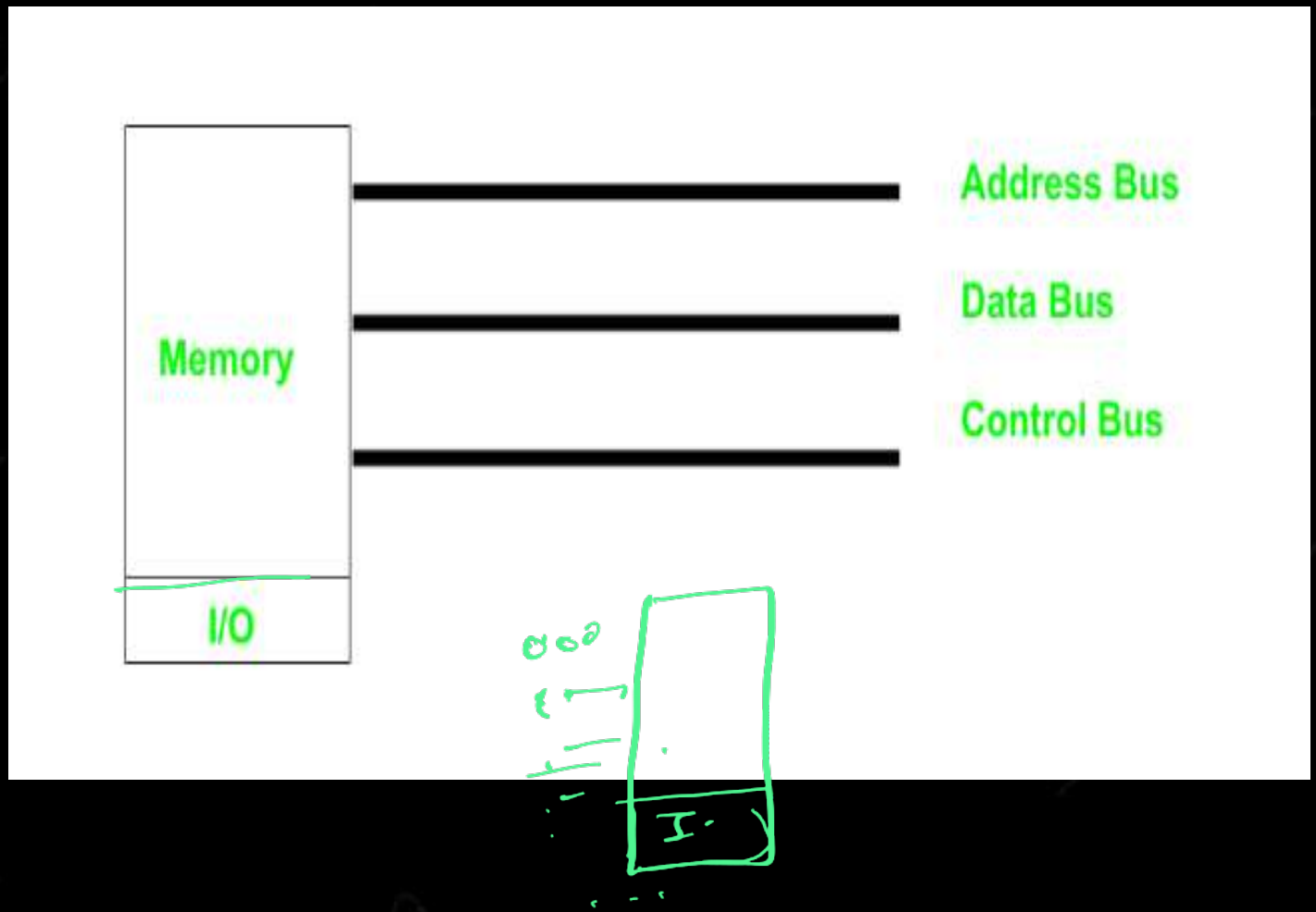
**Isolated I/O –**Then we have Isolated I/O in which we Have a common bus(data and address) for I/O and memory but separate read and write control lines for I/O. So when the CPU decodes instructions if data is for I/O then it places the address on the address line and sets the I/O read or write control line due to which data transfer occurs between CPU and I/O. As the address space of memory and I/O is isolated and the name is so. The address for I/O here is called ports. Here we have different read-write instructions for both I/O and memory.



# Explain how the computer buses can be used to communicate with memory and I/O

## Memory Mapped I/O –

- In this case, every bus is in common due to which the same set of instructions work for memory and I/O.
- Hence we manipulate I/O the same as memory and both have the same address space, due to which the addressing capability of memory becomes less because some part is occupied by the I/O



# DIFFERENCE BETWEEN-

| Isolated I/O | Memory Mapped I/O |
| --- | --- |
| Memory and I/O have separate address space | Both have same address space |
| All address can be used by the memory | Due to addition of I/O addressable memory become less for memory |
| Separate instruction control read and write operation in I/O and Memory | Same instructions can control both I/O and Memory |
| In this I/O address are called ports. | Normal memory address are for both |
| More efficient due to separate buses | Lesser efficient |
| Larger in size due to more buses | Smaller in size |
| It is complex due to separate logic is used to control both. | Simpler logic is used as I/O is also treated as memory only. |

# Advantages of Memory-Mapped I/O

**Faster I/O Operations:** Memory-mapped I/O allows the CPU to access I/O devices at the same speed as it accesses memory. This means that I/O operations can be performed much faster compared to isolated I/O.

**Simplified Programming:** Memory-mapped I/O simplifies programming as the same instructions can be used to access memory and I/O devices. This means that software developers do not have to use specialized I/O instructions, which can reduce programming complexity.

**Efficient Use of Memory Space:** Memory-mapped I/O is more memory-efficient as I/O devices share the same address space as the memory. This means that the same memory address space can be used to access both memory and I/O devices

# Disadvantages of Memory-Mapped I/O:

**Limited I/O Address Space:** Memory-mapped I/O limits the I/O address space as I/O devices share the same address space as the memory. This means that there may not be enough address space available to address all I/O devices.

**Slower Response Time:** If an I/O device is slow to respond, it can delay the CPU's access to memory. This can lead to slower overall system performance

# Advantages of Isolated I/O

GW
GATEWAY CLASSES

**Large I/O Address Space:** Isolated I/O allows for a larger I/O address space compared to memory-mapped I/O as I/O devices have their own separate address space.

**Greater Flexibility:** Isolated I/O provides greater flexibility as I/O devices can be added or removed from the system without affecting the memory address space.

**Improved Reliability:** Isolated I/O provides better reliability as I/O devices do not share the same address space as the memory. This means that if an I/O device fails, it does not affect the memory or other I/O devices.

# Disadvantages of Isolated I/O

**Slower I/O Operations:** Isolated I/O can result in slower I/O operations compared to memory-mapped I/O as it requires the use of specialized I/O instructions.

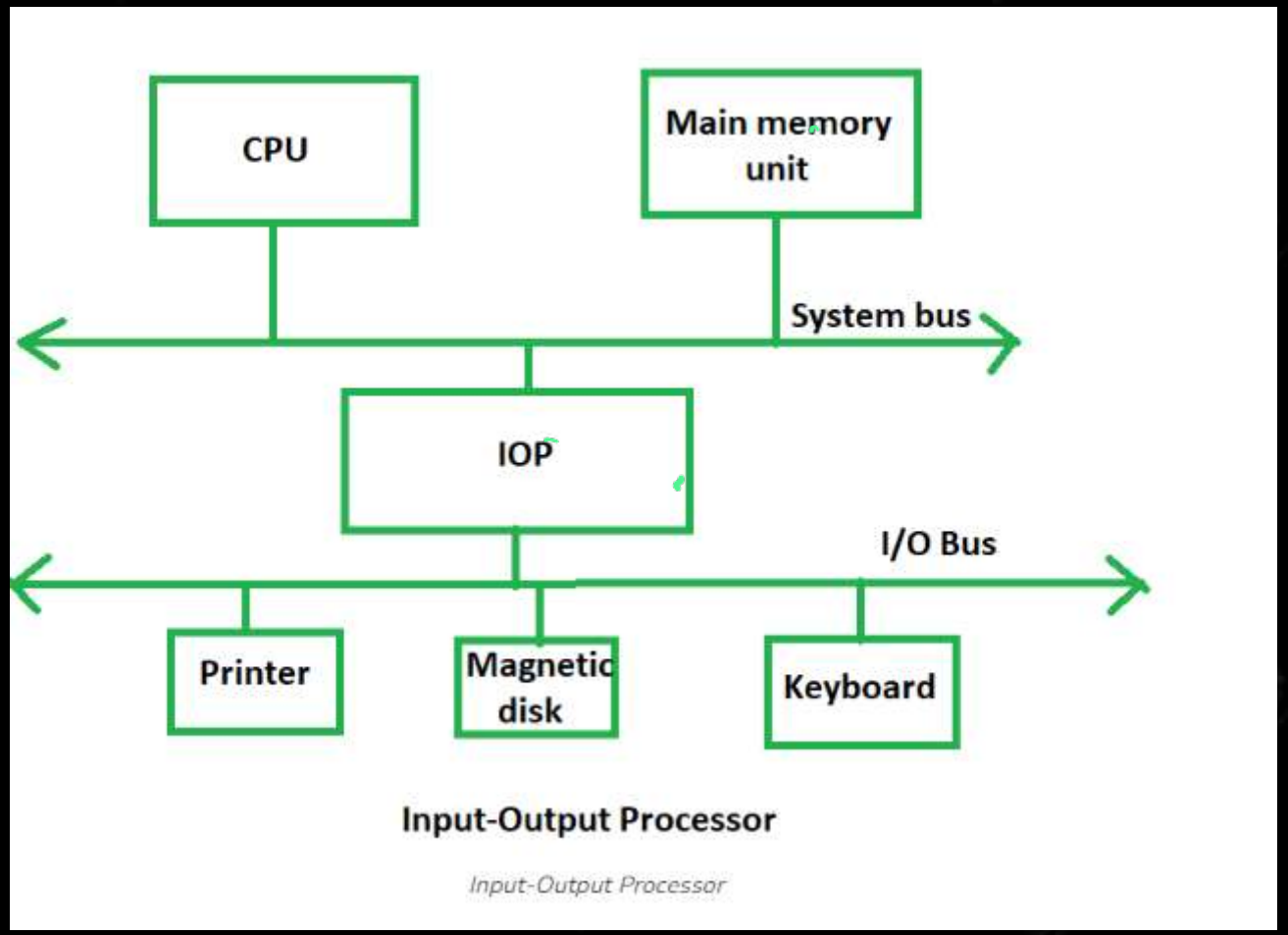
**More Complex Programming:** Isolated I/O requires specialized I/O instructions, which can lead to more complex programming.

# Input-Output Processor

- The DMA mode of data transfer reduces the CPU's overhead in handling I/O operations. It also allows parallelism in CPU and I/O operations. Such parallelism is necessary to avoid the wastage of valuable CPU time while handling I/O devices whose speeds are much slower as compared to CPU.
- The concept of DMA operation can be extended to relieve the CPU further from getting involved with the execution of I/O operations. This gives rise to the development of special purpose processors called Input-Output Processor (IOP) or IO channels.
- The Input-Output Processor (IOP) is just like a CPU that handles the details of I/O operations. It is more equipped with facilities than those available in a typical DMA controller. The IOP can fetch and execute its own instructions that are specifically designed to characterize I/O transfers.
- In addition to the I/O-related tasks, it can perform other processing tasks like arithmetic, logic, branching, and code translation. The main memory unit takes a pivotal role. It communicates with the processor by means of DMA.

# Input-Output Processor

- The Input-Output Processor is a specialized processor that loads and stores data in memory along with the execution of I/O instructions.
- It acts as an interface between the system and devices. It involves a sequence of events to execute I/O operations and then store the results in memory.



Input-Output Processor

# Features of an Input-Output Processor

**Specialized Hardware:** An IOP is equipped with specialized hardware that is optimized for handling input/output operations. This hardware includes input/output ports, DMA controllers, and interrupt controllers.

**DMA Capability:** An IOP has the capability to perform Direct Memory Access (DMA) operations. DMA allows data to be transferred directly between peripheral devices and memory without going through the CPU, thereby freeing up the CPU for other tasks.

**Interrupt Handling:** An IOP can handle interrupts from peripheral devices and manage them independently of the CPU. This allows the CPU to focus on executing application programs while the IOP handles interrupts from peripheral devices.

**Protocol Handling:** An IOP can handle communication protocols for different types of devices such as Ethernet, USB, and SCSI. This allows the IOP to interface with a wide range of devices without requiring additional software support from the CPU.

# Features of an Input-Output Processor…

**Buffering:** An IOP can buffer data between the CPU and peripheral devices. This allows the IOP to handle large amounts of data without overloading the CPU or the peripheral devices.

**Command Processing:** An IOP can process commands from peripheral devices independently of the CPU. This allows the CPU to focus on executing application programs while the IOP handles peripheral device commands.

**Parallel Processing:** An IOP can perform input/output operations in parallel with the CPU. This allows the system to handle multiple tasks simultaneously and improve overall system performance.

# Advantages of Input-Output Processor

- The I/O devices can directly access the main memory without the intervention of the processor in I/O processor-based systems.
- It is used to address the problems that arise in the Direct memory access method.

**Reduced Processor Workload:** With an I/O processor, the main processor doesn't have to deal with I/O operations, allowing it to focus on other tasks. This results in more efficient use of the processor's resources and can lead to faster overall system performance.

**Improved Data Transfer Rates**: Since the I/O processor can access memory directly, data transfers between I/O devices and memory can be faster and more efficient than with other methods.

**Increased System Reliability:** By offloading I/O tasks to a dedicated processor, the system can be made more fault-tolerant. For example, if an I/O operation fails, it won't affect other system processes.

**Scalability**: I/O processor-based systems can be designed to scale easily, allowing for additional I/O processors to be added as needed. This can be particularly useful in large-scale data centers or other environments where the number of I/O devices is constantly changing.

# Disadvantages of Input-Output Processor

**Cost:** I/O processors can add significant costs to a system due to the additional hardware and complexity required. This can be a barrier to adoption, especially for smaller systems.

**Increased Complexity:** The addition of an I/O processor can increase the overall complexity of a system, making it more difficult to design, build, and maintain. This can also make it harder to diagnose and troubleshoot issues.

**Limited Performance Gains:** While I/O processors can improve system performance by offloading I/O tasks from the main processor, the gains may not be significant in all cases. In some cases, the additional overhead of the I/O processor may actually slow down the system.

**Synchronization Issues:** With multiple processors accessing the same memory, synchronization issues can arise, leading to potential data corruption or other errors.

**Lack of Standardization:** There are many different I/O processor architectures and interfaces available, which can make it difficult to develop standardized software and hardware solutions. This can limit interoperability and make it harder for vendors to develop compatible products..

# Trap

- A trap is a mechanism that allows a program to intercept certain events or signals during its execution.
- Traps are often associated with low-level operations, such as hardware interrupts or signals from the operating system.
- They are used for handling events like division by zero, memory access violations, or other hardware/software-related errors.
- Traps are generally part of the lower-level aspects of the system and are not typically used for high-level error handling within the application code.

# Exception

- An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions.
- Exceptions are a high-level construct and are typically used for error handling within the application code.
- When an exceptional situation occurs, an exception is thrown, and the program can catch and handle the exception in a structured way.
- Exceptions are part of the programming language's design and are explicitly used by developers to handle errors and abnormal situations.

2 Marks

# I/O control

- I/O control, short for Input/Output control, refers to the management and coordination of input and output operations in a computer system. Input refers to data entering the system, while output refers to data leaving the system. I/O control is crucial for ensuring efficient and orderly data transfer between the computer and its external devices, such as storage devices, displays, keyboards, and network interfaces.

**Key aspects of I/O control include:**

**Device Management:**

- Managing communication between the central processing unit (CPU) and various peripheral devices.
- Handling device-specific details, such as device drivers, to facilitate data transfer.

**Buffering:**

- Using buffers to temporarily store data during I/O operations to cope with speed differences between the CPU and I/O devices. Buffers help smooth out the flow of data and reduce the impact of speed variations.

**Error Handling:**

- Implementing mechanisms to detect and handle errors that may occur during input or output operations.
- Error handling is crucial to maintain the integrity of data and prevent system failures.

# Architectural Classification

## Flynn's classification

- Based on the multiplicity of Instruction Streams and Data Streams
- **Instruction Stream**
- • Sequence of Instructions read from memory

» **Data Stream**

- • Operations performed on the data in the processor

7 Marks 2015-16 2016-17

# Architectural Classification:

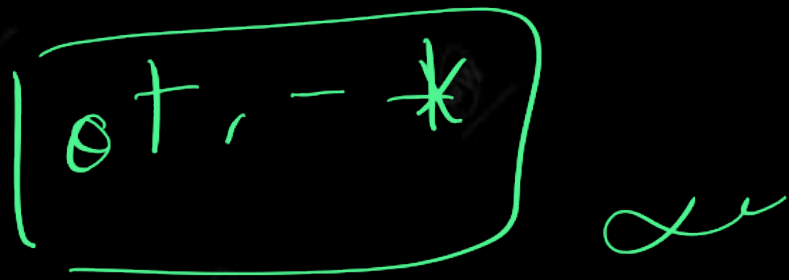
2 Marks

GW GATEWAY CLASSES

- Parallel computing is computing where the jobs are broken into discrete parts that can be executed concurrently. Each part is further broken down into a series of instructions.
- Instructions from each piece are executed simultaneously on different CPUs. The breaking up of different parts of a task among multiple processors will help to reduce the amount of time to run a program.
- Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both.
- Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized. The difficult problem of parallel processing is portability.

# Architectural Classification

- An Instruction Stream is a sequence of instructions that are read from memory. The Data Stream is the operations performed on the data in the processor.
- Flynn's taxonomy is a classification scheme for computer architectures proposed by Michael Flynn in 1966. The taxonomy is based on the number of instruction streams and data streams that can be processed simultaneously by a computer architecture.



# Architectural Classification

There are four categories in Flynn's taxonomy:

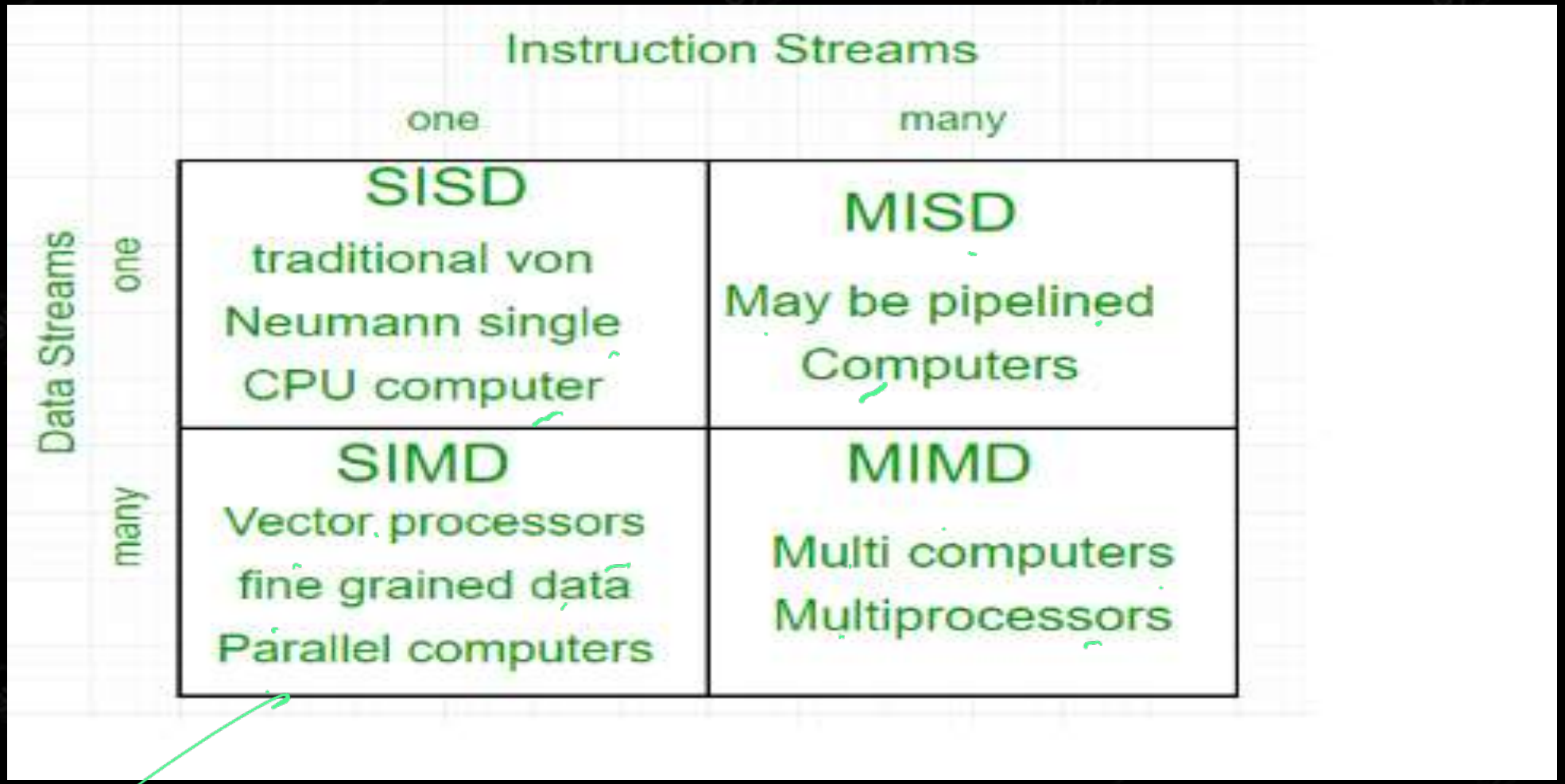
**Single Instruction Single Data (SISD):** In a SISD architecture, there is a single processor that executes a single instruction stream and operates on a single data stream. This is the simplest type of computer architecture and is used in most traditional computers.

**Single Instruction Multiple Data (SIMD):** In a SIMD architecture, there is a single processor that executes the same instruction on multiple data streams in parallel. This type of architecture is used in applications such as image and signal processing.

**Multiple Instruction Single Data (MISD):** In a MISD architecture, multiple processors execute different instructions on the same data stream. This type of architecture is not commonly used in practice, as it is difficult to find applications that can be decomposed into independent instruction streams.

**Multiple Instruction Multiple Data (MIMD):** In a MIMD architecture, multiple processors execute different instructions on different data streams. This type of architecture is used in distributed computing, parallel processing, and other high-performance computing applications.

# Architectural Classification



# Why read and write lines in DMA controller is Bi- directional

- The read and write control lines running through the data buses have control signals. The microprocessor can read data from memory or write data to the memory so the data buses are bidirectional.
- One direction is needed to write the data while another direction is required for the read command to fetch the data.

2 Marks

# Difference between-

| PROCESSOR | IOP |
|---|---|
| Processor is CPU | IOP is a port of i/o processing |
| Handles arithmetic and logical task | Handles input- output processing |
| DMA controller is set up by CPU | IOP is a processor with DMA |

2 Marks

## What is the use of modem in synchronous communication

- A modem converts digital signals into audio tones to be transmitted over telephone lines and also converts audio tones from the line to digital signals for machine use

GW GATEWAY CLASSES

2 Marks

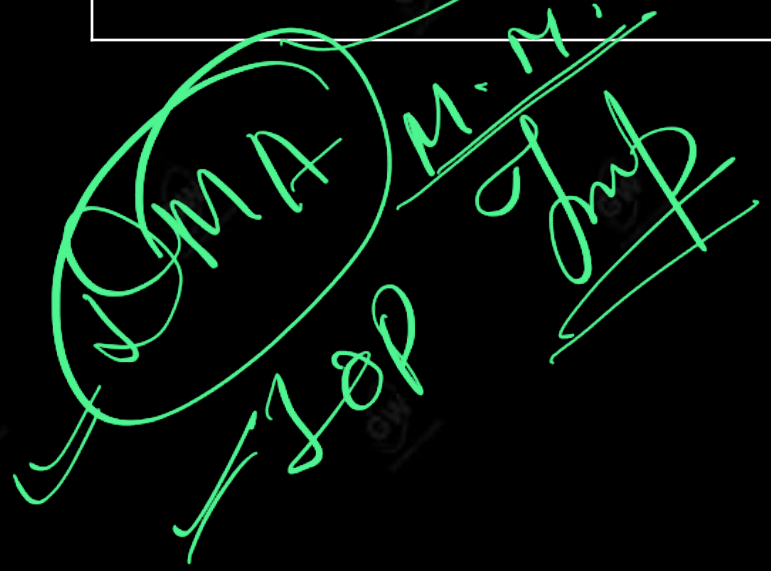
# Difference between-

| SERIAL COMMUNICATION | PARALLEL COMMUNICATION |
|---|---|
| Data flow in bi directional bit by bit | Multiples lines are used to send data i.e. 8 bit or 1 byte at a time |
| Not so costly | expensive |
| 1 bit transferred at 1 clock pulse | 8 bit or 1 byte transferred at 1 clock pulse |
| Speed is slow | Speed is fast |
| Used for long distance communication example computer to computer communication | Short distance communication for example computer to printer |

GW GATEWAY CLASSES

# Difference between-

| i/o program controlled transfer | DMA TRANSFER |
|---|---|
| Software controlled data transfer | Hardware controlled data transfer |
| Data transfer speed is low | Data transfer speed is fast |
| CPU is involved in the transfer | CPU is not involved in the transfer |
| Extra hardware is not required | DMA controller is require d to carry out data transfer |



AKTU Full Courses (Paid)
Download Gateway Classes Application
From the Google Play
store All Subjects

Link in Description

# Thank You