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 needs special communication links for interfacing them with CPU.

The purpose of communication links to resolve the differences that exist between central computer and each peripheral.

The major differences are :-

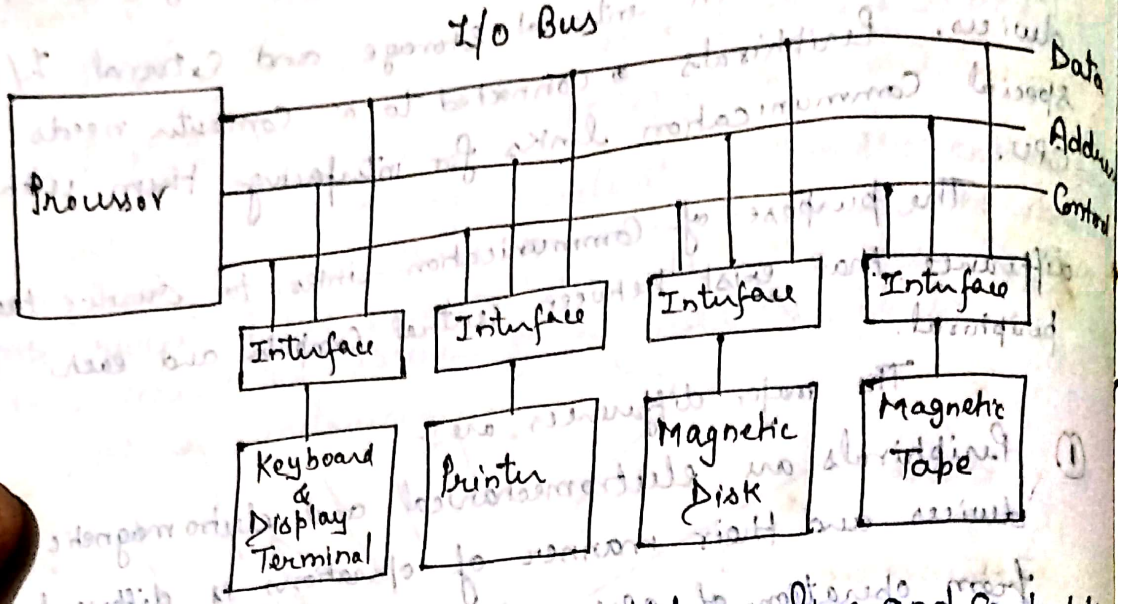
(1) Peripherals are electromechanical and electromagnetic devices and their manner of operations is different from operation of CPU and memory which are electronic devices, therefore a conversion of signals values may be required.

(2) The data transfer rate of peripherals is usually slower than transfer rate of CPU so synchronization is required.

(3) Data codes and formats in peripherals differ from word format in CPU and memory.

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

To resolve these difference, computer system include special hardware components b/w CPU and peripherals to supervise and synchronize all input and output transfer. These components are called Interface.



I/O bus Consist of data line, address line and Control line. Each peripheral device has associated with an interface unit. Interface synchronize the data flow and supervise the transfer b/w peripheral and processor.

The I/O Bus from processor is attached to all peripheral interface. Each interface decodes the address and Control received from I/O Bus, interprets them for peripheral and provides signal for peripheral ~~Controller~~ devices.

To Communicate with particular device, the processor places a device address on address lines. Each interface attached to I/O bus Contains an address decoder that monitors the address lines. when interface detects its own address, it activates the path b/w bus lines and device that it Controls. All peripherals whose address does not Corresponds to address in bus are disabled by their interface.

At the same time that the address is made available in address lines, the processor provides a function code in Control lines, The

interface selected responds to function code and proceeds to execute it. This is called I/O Commands. It is classified in Control, Status, data output and data input.

Control Command :- It is issued to activate the peripheral and to inform it what to do.

Status Command :- It is used to test various status conditions in the interface and peripheral.

Eg:- The Computer may wish to check status of peripheral before a transfer is initiated.

Output data :- It causes the interface to respond by transferring data from the bus into one of its registers.

Input data :- In this case, the interface receives an item of data from peripheral and places it in its buffer register. The processor checks if data are available by means of status Command and then issues a data input Command. The interface places the data on data lines, where they are accepted by processor.

Asynchronous Data Transfer

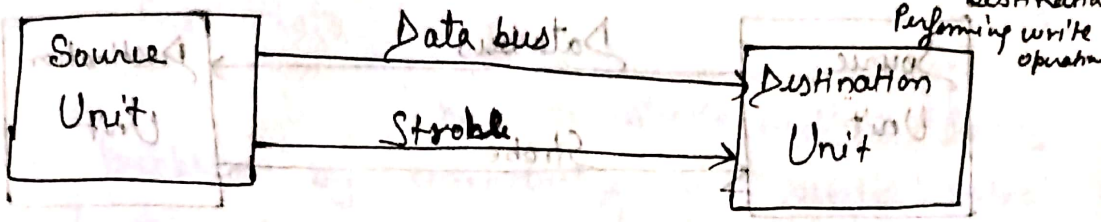
The internal operations in a digital system are synchronize by means of clock pulses supplied by a common pulse generator. If the registers in interface share a common clock with CPU registers, the transfer between two units is said to be synchronous. In most cases the internal timing in each unit is independent from other in that each uses its own private clock for internal Registers.

Asynchronous data Transfer b/w two independent units requires that Control signals be transmitted b/w Communicating units to indicate the time at which data is being transmitted.

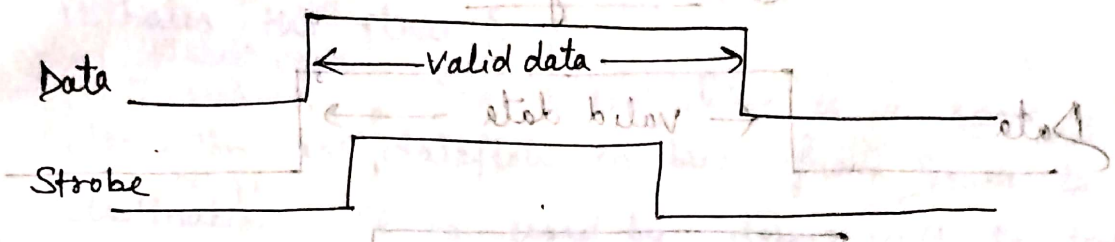
One way of achieving this is by means of Strobe pulse supplied by one of units to indicate the other unit when the transfer has to occur. Another method Commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in bus. The unit receiving the data item responds with another Control signal to acknowledge receipt of data. This type of agreement b/w two independent units is refferd to as Handshaking.

STROBE CONTROL :- This method of asynchronous data transfer employs a single Control line to time each transfer. The Strobe may be activated by either the source or destination unit.

Eg:- CPU → Memory
Source — Destination
Performing write operation



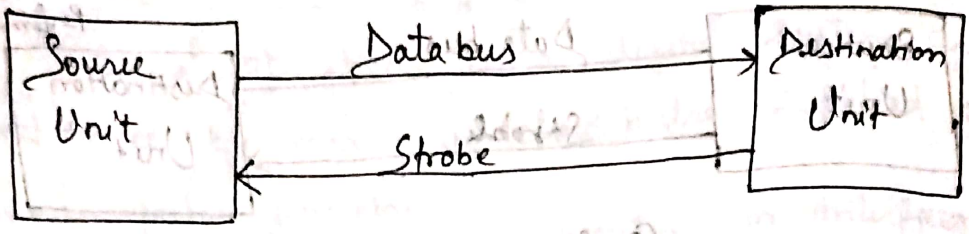
(a) Block diagram



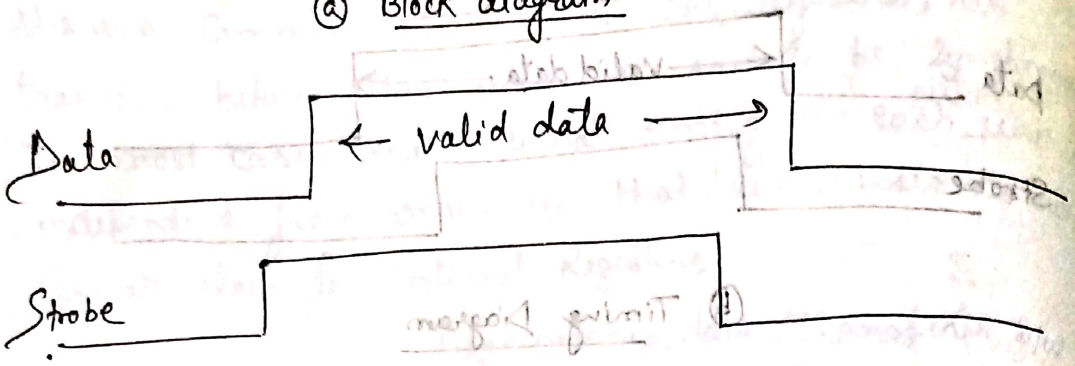
(b) Timing Diagram

Fig:- Source initiated Strobe for data Transfer

The data bus carries binary information from source unit to the distination unit. The Strobe is a single line that informs the distination unit when a valid data is available in data bus. As in the timing diagram, the source unit first places the data on the data bus. After a brief delay to ensure that the data settle to a steady value, the source activates the strobe pulse. The information on the databus and strobe signal remain in active state for a sufficient time period to allow the distination unit to receive the data. The source removes the data from bus a brief period after it disable the clock pulse. Actually the source does not have to change the information in databus. The fact that the strobe signal is disabled indicates that databus does not contain valid data. New valid data will be available only after strobe is enabled again.



(a) Block diagram



(b) Timing diagram

Fig:- Destination initiated Strobe for data Transfer.

In this case, the destination unit activates the Strobe pulse, informing the source to provide the data. The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain in bus long enough for destination unit to accept it. The destination unit then disable the Strobe. The source removes the data from bus after a predetermine time interval.

HANDSHAKING

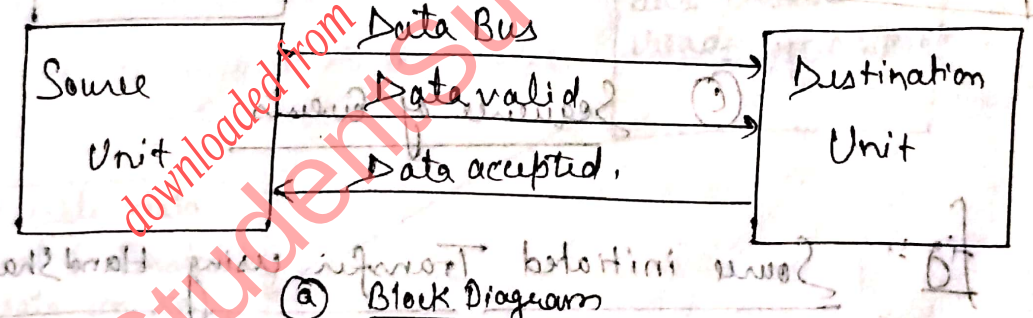
The disadvantage of Strobe method is that the source unit that indicate the transfer has no way of knowing whether the destination unit has actually received the data that was placed in bus. Similarly, a destination unit that initiates the transfer has no way

knowing whether the source unit has actually placed the data on bus.

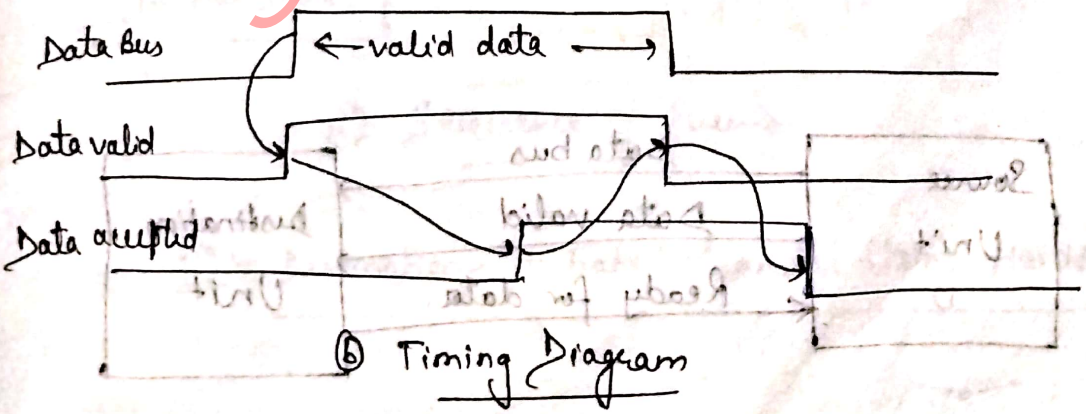
The Handshake method solve this problem by introducing a second control signal that provides a reply to unit that initiates the transfer.

One control line is in same direction as dataflow in bus from source to destination. It is used by source unit to inform the destination unit whether there are valid data in the bus. The other control line is in other direction from destination to source. It is used by destination unit to inform the source whether it can accept data.

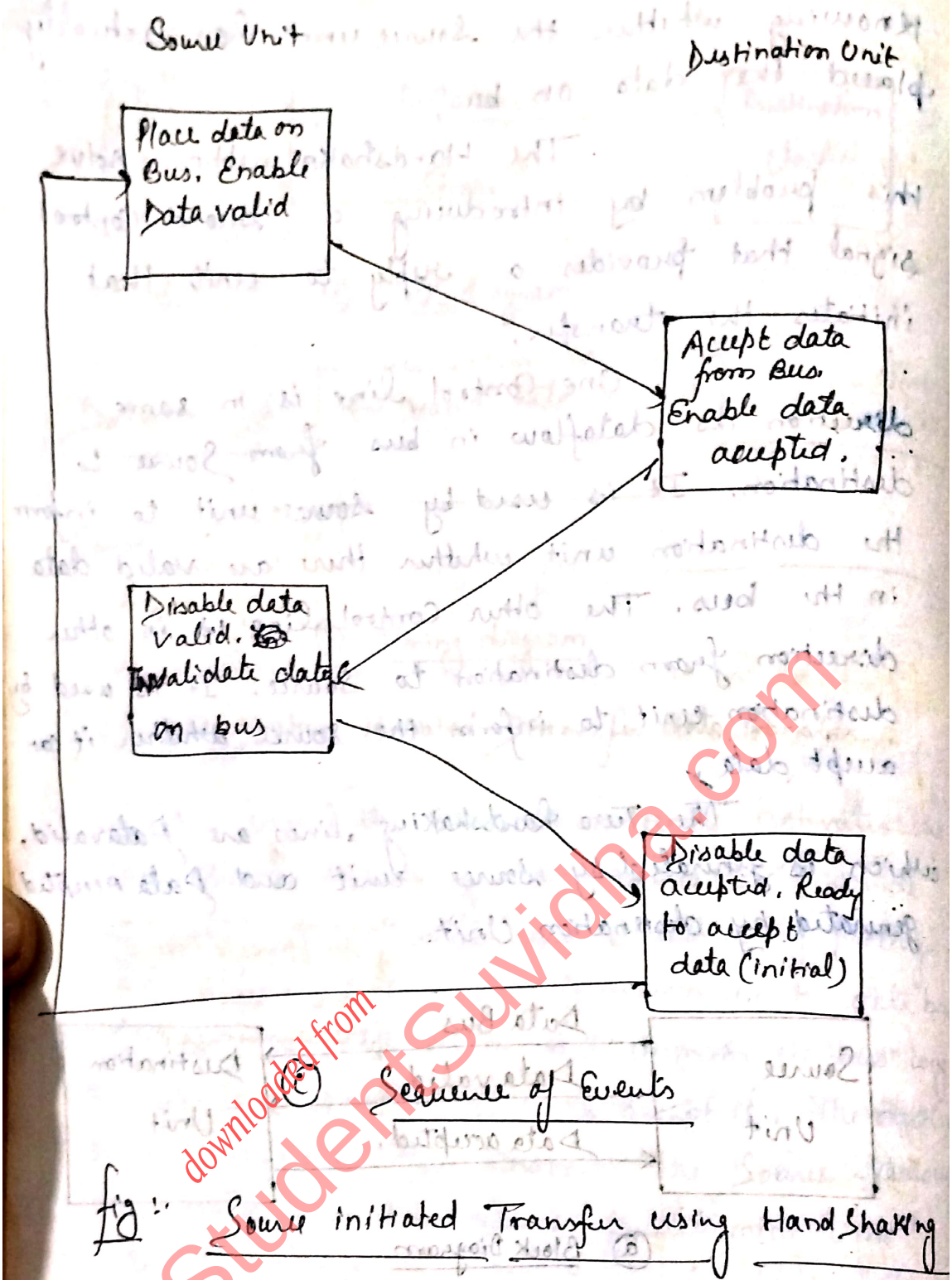
The two handshaking lines are Datavalid, which is generated by source unit and Data Accepted, generated by destination Unit.



(a) Block Diagram

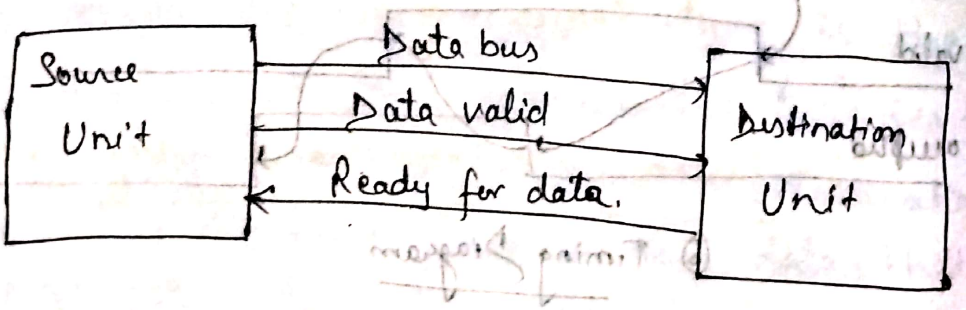


(b) Timing Diagram

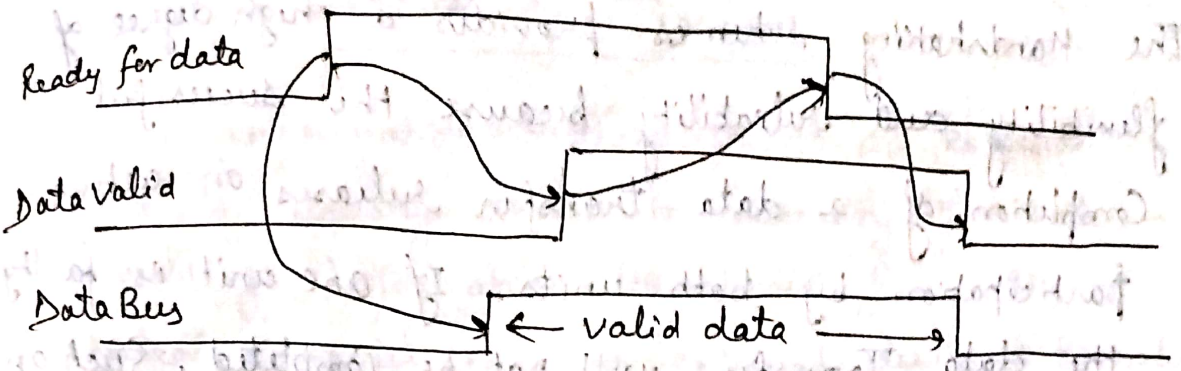


(a) Sequence of Events

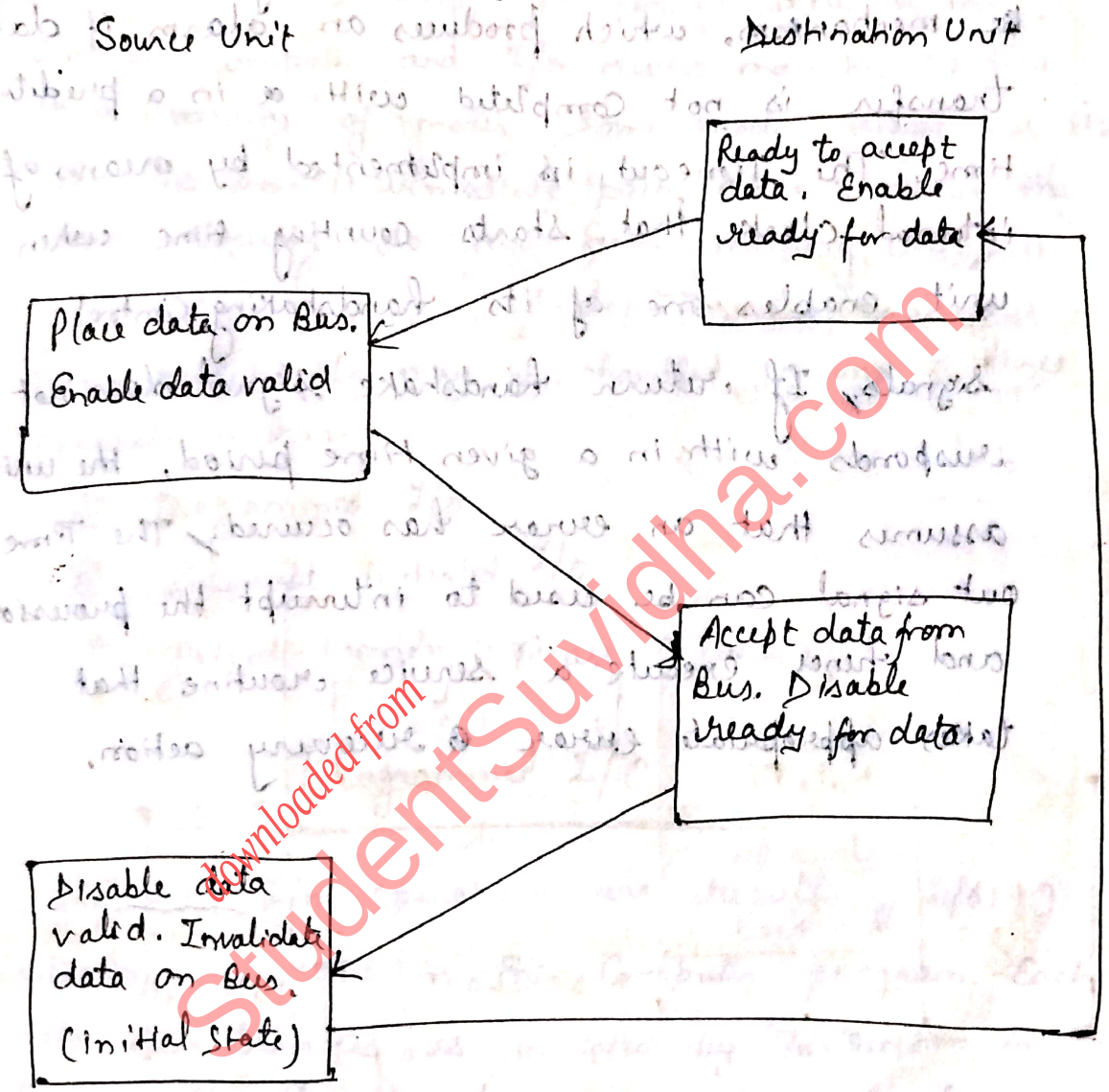
fig: Source initiated Transfer using Handshaking



(b) Block diagram



(b) Timing Diagram



(c) Sequence of Events

fig:- Destination-initiated transfer Using Handshaking

The Handshaking schemes provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units. If one unit is faulty, the data Transfer will not be Completed. Such an error can be detected by means of timeout mechanism, which produces an alarm if data transfer is not Completed with in a predetermined time. The Timeout is implemented by means of internal clock that starts counting time when unit enables one of its handshaking Control signals. If return handshake signal does not responds with in a given time period, the unit assumes that an error has occured. The Time-out signal can be used to interrupt the processor and there execute a service routine that takes appropriate error & recovery action.

Modes of Transfer

Binary information received from an external device is usually stored in memory for later processing. Information transferred from the central computer into an external device originates in memory unit.

Data transferred between the central computer and I/O device may be handled in a variety of modes. Some modes use the CPU as an intermediate path, other transfer the data directly to and from memory unit.

Data transfer to and from peripherals may be handled in one of three possible modes:-

1. Programmed I/O
2. Interrupt initiated I/O
3. Direct Memory Access (DMA).

Programmed I/O

Programmed I/O operations are result of I/O instruction written in the computer program. Each data item transfer is initiated by instructions in program. Usually the transfer is to and from a CPU register and peripherals. Other instructions are needed to transfer the data to and from CPU and memory. Once the data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made. It is upto the program instructions executed in CPU.

In programmed I/O method, the I/O device does not have direct access to memory. A Transfer from I/O devices to memory requires the execution of several instructions by CPU, including an input instruction to transfer data from input device to the CPU and store instruction to transfer data from CPU to memory;

In the figure of data transfer from I/O device through an interface to CPU the device transfer bytes of data one at a time as they are available. When a byte of data is available the device place it in the I/O bus and enable the data valid line,

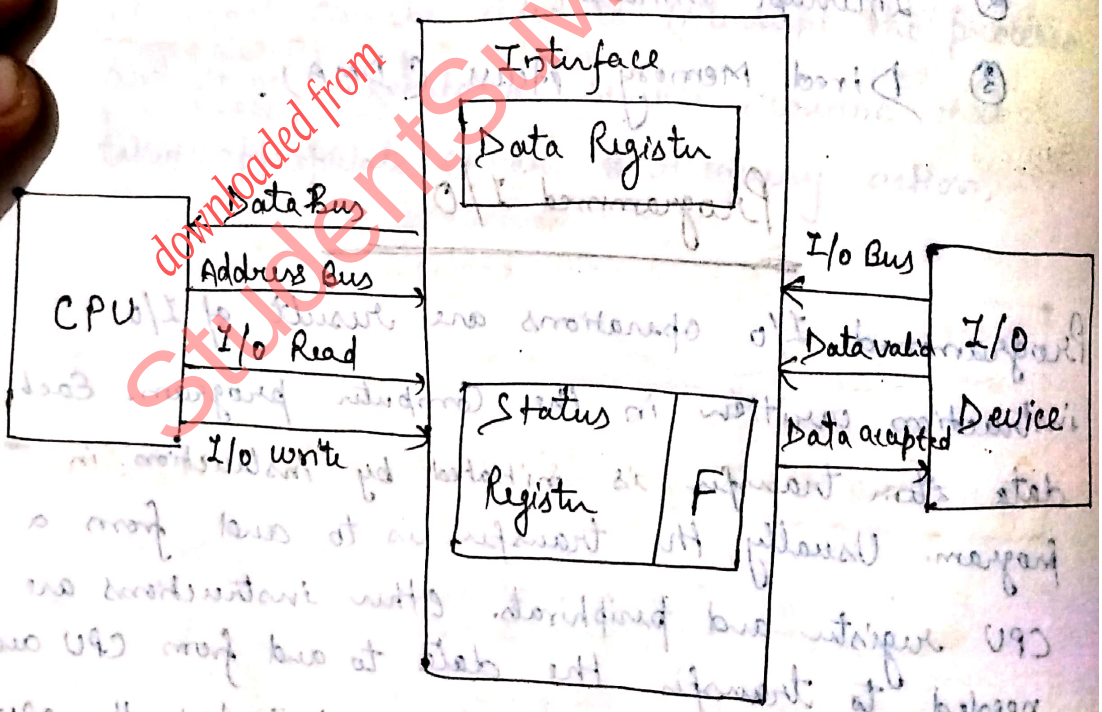


fig: Data Transfer from I/O device to CPU

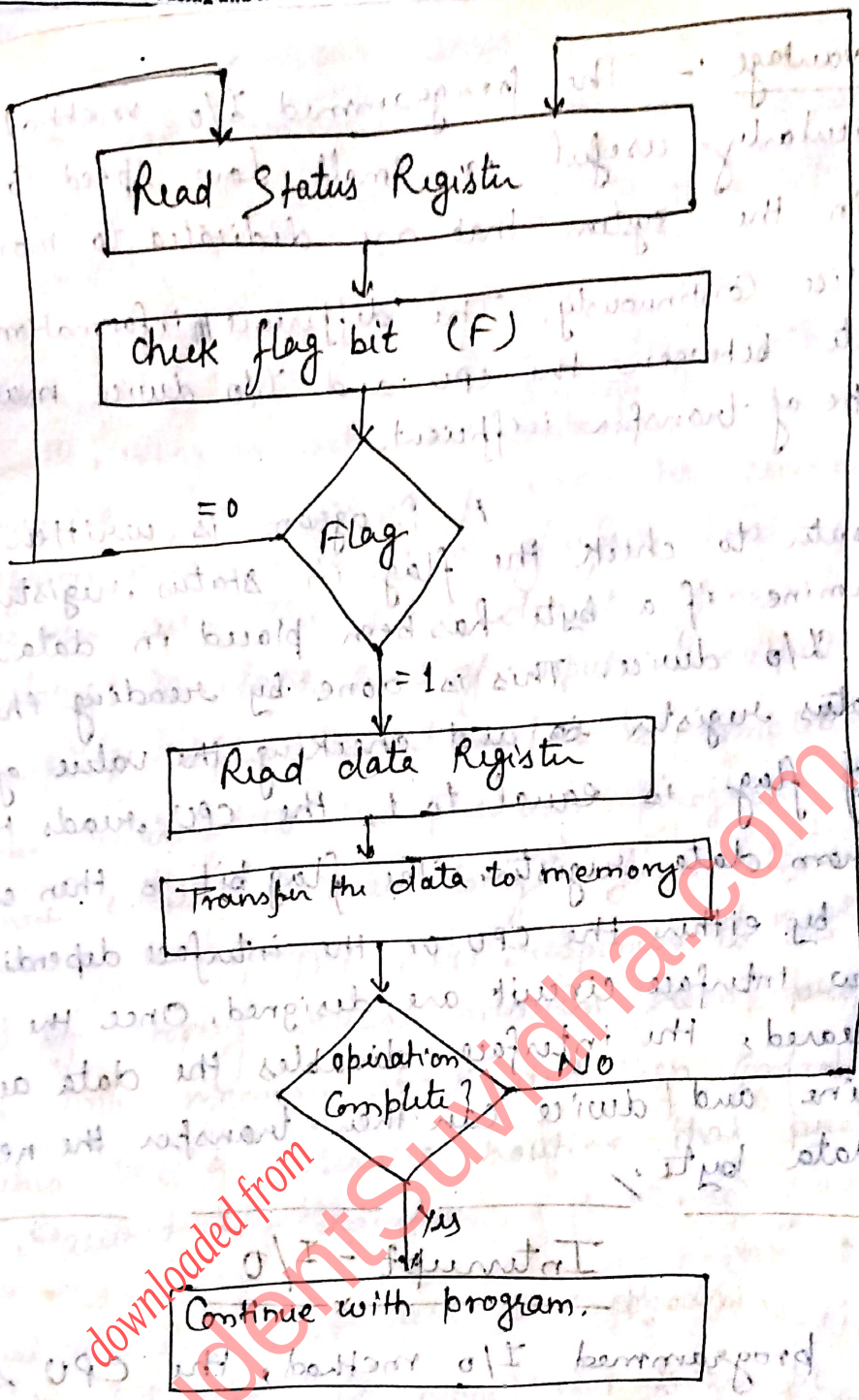


Fig: flow chart for CPU program to input data

The interface accepts the bytes into its data register and enable its data accepted line. The interface sets a bit (F) in the status register. The device can now disable the data valid line, but it will not transfer another byte until the data accepted line is disabled by interface.

Disadvantage :- The programmed I/O method is particularly useful in small low speed Computer or in the system that are dedicated to monitor a device continuously. The difference in information transfer rate between the CPU and I/O device make this type of transfer inefficient.

A Program is written for Computer to check the flag in status register to determine if a byte has been placed in data register by I/O device. This is done by reading the status register and checking the value of flag bit. If flag is equal to 1, the CPU reads the data from data register. The flag bit is then cleared to 0 by either the CPU or the interface depending on how interface circuit are designed. Once the flag is cleared, the interface disables the data accepted line and device can then transfer the next data byte.

Interrupt - I/O

In programmed I/O method, the CPU stays in a programmed loop until the I/O units indicates that it is ready for data transfer. This is time consuming process since it keeps the processor busy needlessly. It can be avoided by using interrupt facility and special commands to inform the interface to issue an interrupt request signal, when data are available from device. In mean time, the CPU can proceed to execute another progra

The interface meanwhile keeps monitoring the device. When interface determines that device is ready for data transfer, it generates an interrupt request to the Computer.

In this, while CPU is running the programs, it does not check the flag. However, when the flag is set, the ~~Computer~~ processor is momentarily interrupted from proceeding with the current program and is informed of fact that flag has been set. The CPU deviates from what it is doing to take care of input or output transfer. After the transfer is completed, the Computer returns to the previous program to continue what it was doing before interrupt.

The CPU responds to the interrupt signal by storing return address from program Counter into memory stack and then control branches to a service routine that processes the required I/O transfer.

There are 2 methods.

(1) Vector Interrupt (2) Non Vector Interrupt.

In Non vector Interrupt, the branch address is assigned to a fixed location in memory.

In a vector Interrupt, the source that interrupts supplies the branch information to the Computer. This information is called Interrupt vector.

In some Computer, the interrupt vector is first address of I/O service routine.

In other Computers, the interrupt vector is an address that points to a location in memory where beginning address of I/O service routine is stored.

Priority Interrupt :- A Priority interrupt is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrives simultaneously.

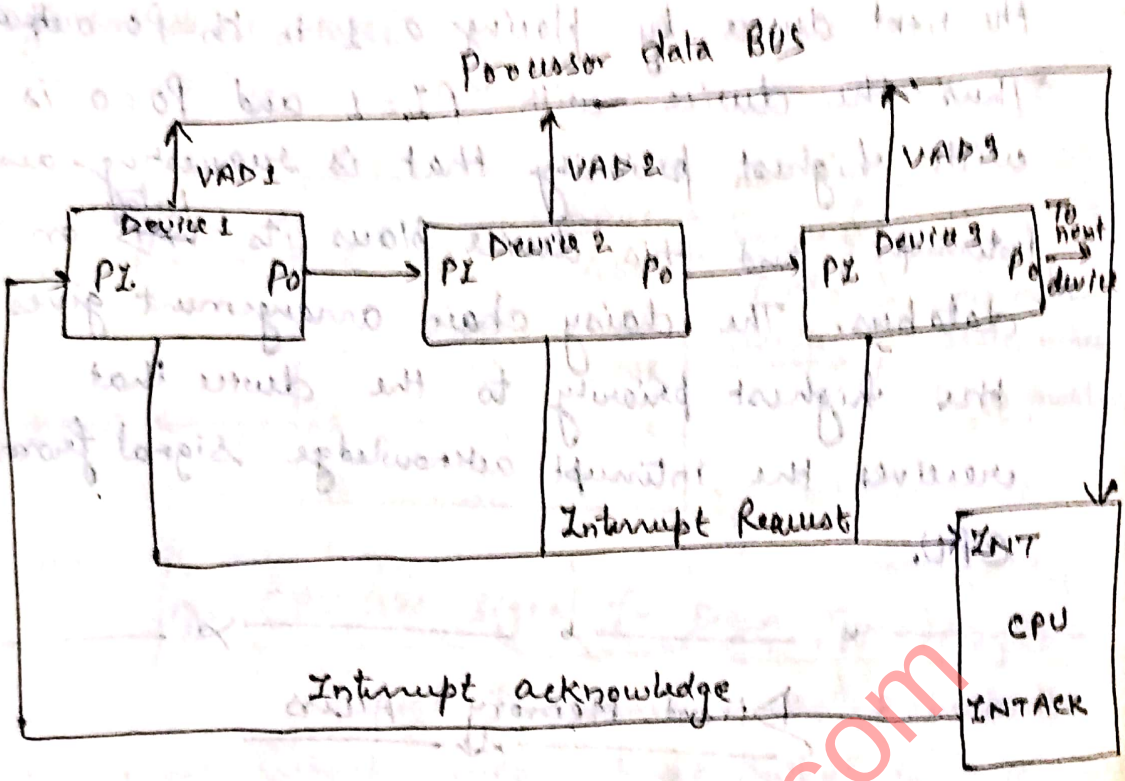
Devices with high speed transfers such as magnetic disks are given high priority and slow devices such as keyboards receive low priority. When two devices interrupts the computer at same time, the computer services the device with higher priority first.

Daisy chain Priority :- This method of establishing priority consists of a serial connection of all devices that request an interrupt. The device with highest priority is placed at first position, followed by lower priority devices upto the device with lowest priority which is placed last in chain.

The interrupt request line is common to all devices. If any device has its interrupt signal in low level state, the interrupt line goes to low level state and enables the interrupt input in CPU.

When no interrupts are pending the interrupt line stays in the high level state and no interrupts are recognized by CPU.

The CPU responds to interrupt request by enabling the interrupt acknowledge line.



This signal is received by device 1 at its PI (Priority input). The acknowledge signal passes on next device through PO (Priority output) only if device 1 is not requesting an interrupt. If device 1 has a pending interrupt, it blocks the acknowledge signal ~~from~~ to next device by placing a 0 in PO output. It then proceeds to insert its own interrupt Data vector Address (VAD) into data bus for CPU to use during interrupt Cycle.

A device with a 0 in its PI input generates a 0 in its PO output to inform the next lower priority device that acknowledge signal has been blocked. A device that is requesting an interrupt and has 1 in its PI input will intercept the acknowledge signal by placing a 0

in its PO output. If device does not have pending interrupts, it transmits the acknowledge signal to the next device by placing a 1 in its PO output. Thus the device with PI=1 and PO=0 is one with highest priority that is requesting an interrupt, and this device places its data on databus. The daisy chain arrangement gives the highest priority to the device that receives the interrupt acknowledge signal from CPU.

Direct Memory Access

The Transfer of data between a fast storage device such as Magnetic disk and memory is often limited by speed of CPU.

Removing the CPU from path and peripheral device manage the memory buses directly would improve the speed of Transfer. This transfer technique is called Direct Memory Access (DMA).

During DMA transfer, the CPU is idle and has no control of memory buses. A DMA Controller takes over the buses to manage the transfer directly between the I/O device and Memory.

figure, shows two Control signals Bus request (BR) input is used by DMA Controller to request the CPU to relinquish control of buses. When this input is active, the CPU

terminates the execution of current instruction and place the data bus, address bus, read & write lines into a High impedance state.

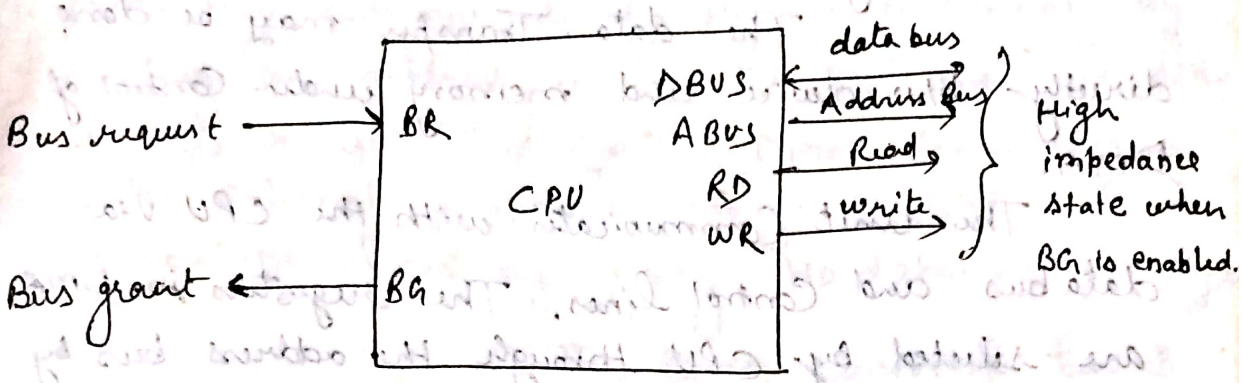


fig:- CPU Bus signal for DMA Transfer

High Impedance state behave like open circuit which means that output is disconnected and does not have logic significance.

The CPU activates the Bus grant (BG) output to inform the external DMA that buses are in High impedance state. The DMA that originates the bus request can now take control of buses to conduct memory transfer without processor intervention.

When DMA terminates the bus transfer. It disables the bus request line. The CPU disables the bus grant, take control of buses and returns to its normal operation.

When DMA takes control of bus system, it communicates directly with memory.

DMA Controller :-

The DMA Controller needs the usual circuits of an interface to communicate with CPU and I/O device. In addition, it needs an address register, a word count register and a set of address lines. The

address register and address lines are used for direct communication with memory.

The word count register specifies the number of words that must be transferred.

The data Transfer may be done directly b/w device and memory under Control of DMA.

The unit Communicates with the CPU via data bus and Control lines. The registers in DMA are selected by CPU through the address bus by enabling the DS (DMA Select) and RS (Register Select) inputs. The RD (Read) and WR (Write) inputs are bidirectional.

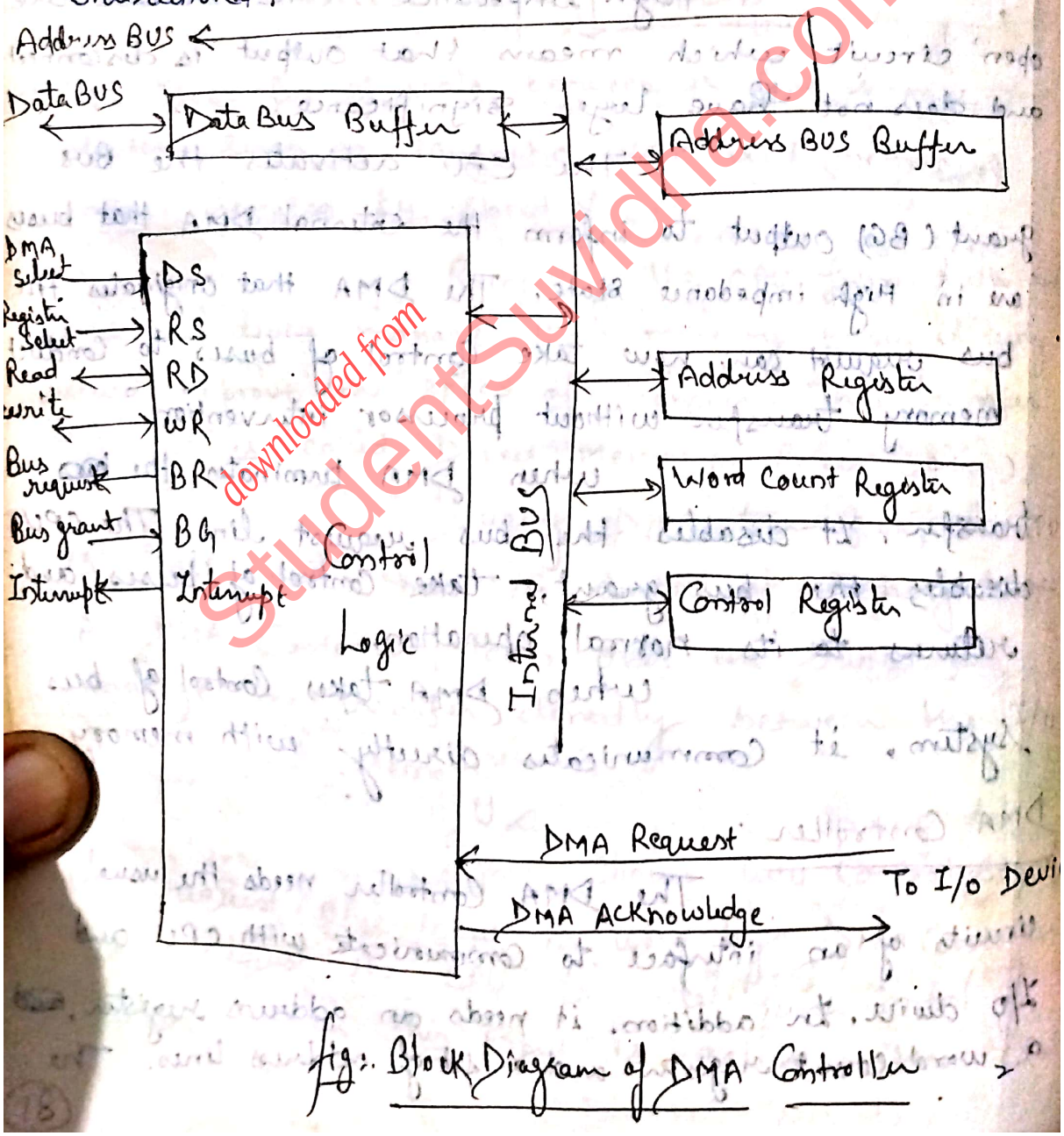


fig: Block Diagram of DMA Controller

when BG (Bus Grant) input is 0. The CPU can Communicate with DMA registers through the data bus to read from or write to DMA Registers. When BG=1, the CPU has relinquished the buses and DMA can Communicate directly with memory by specifying an address in address BUS and activating the RD or WR Control. The DMA Communicates with external peripheral through request and acknowledge lines by using Hand shaking procedure.

The DMA Controller has 3

Ref] Registers :- An address Register, a word Count register and a Control Register. The address register Contains an address to specify the desired location in memory. The address bits go through bus buffers into address BUS. The address register is incremented after each word that is transferred to memory.

The word Count Register holds the number of words to be transferred. This register is decremented by one after each word transferred. The Control register specifies the mode of transfer (RD/WR).

The DMA is first initialized by CPU. After that, DMA starts and Continues to transfer data between memory and peripheral until an entire block is transferred. The CPU initialize the DMA by sending following information through Data BUS:-

(1) The Starting address of memory block where data are available (for read) or where data are to be stored (for write).

(2) The word count, which is number of words in memory block.

(3) Control to specify the mode of transfer such as Read or write.

(4) A Control to start the DMA transfer.

DMA Transfer

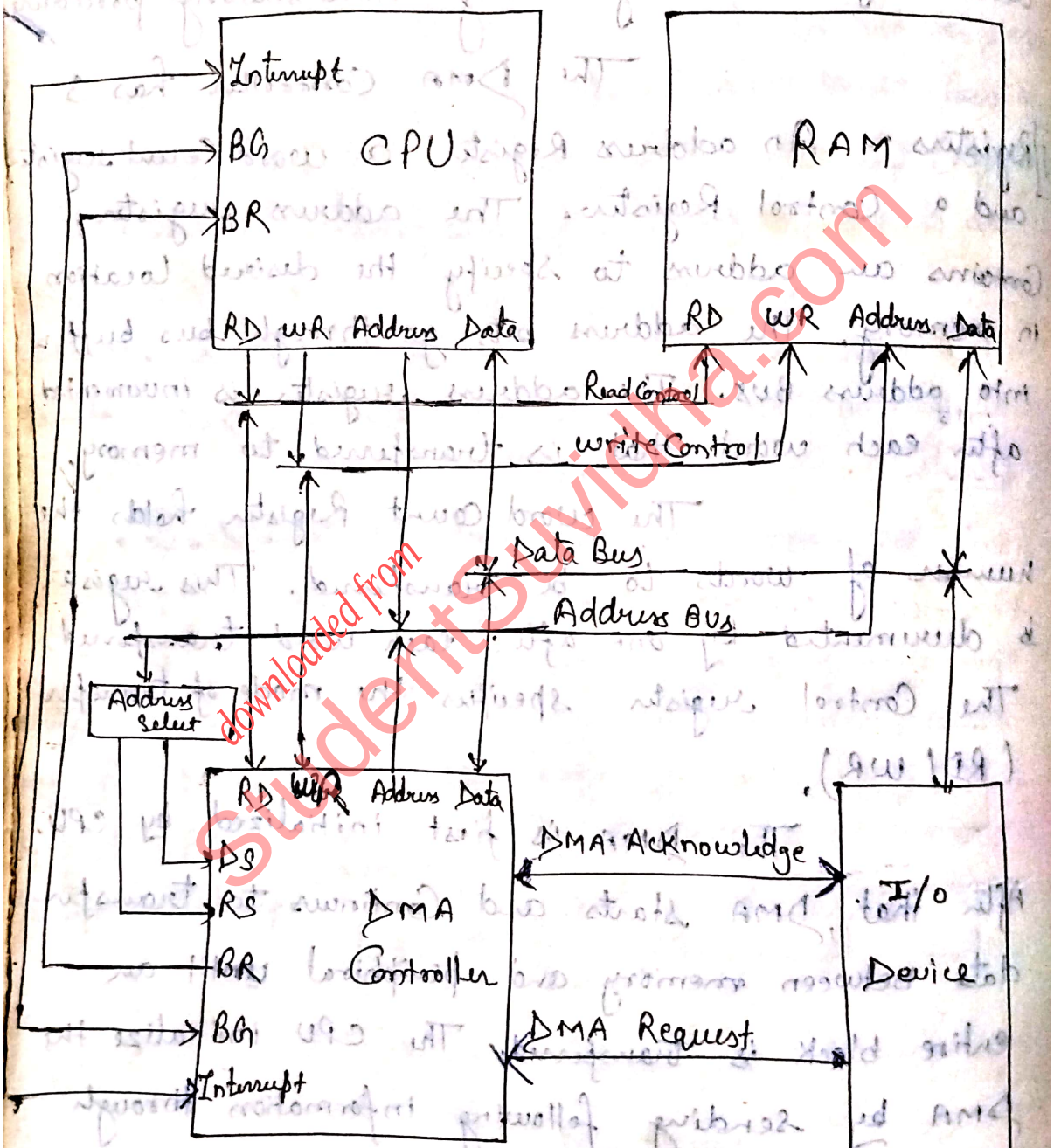


Fig: DMA Transfer in Computer

The CPU Communicates with DMA through address and data buses. The DMA has its own address, which activates the DS and RS lines. The CPU initialize the DMA through data Bus. Once the DMA receives the start Control Command, it can start the transfer b/w peripheral device and memory.

when peripheral device sends a DMA request, the DMA Controller activates the BR line, informing the CPU to relinquish the buses. The CPU responds with its BG line, informing the DMA that its Buses are disabled. The DMA then puts the current value of its address register into address bus, initiates the RD or WR Signal and sends a DMA Acknowledge to the peripheral device.

When BG = 0, the RD and WR are input lines allowing the CPU to communicate with internal DMA registers.

When BG = 1, the RD and WR are output lines from DMA Controller to Random Access Memory to specify the read or write operation for data.

when peripheral device receives a DMA Acknowledge, it puts a word in data bus (write) or receives a word from data bus (read).

Thus DMA Controls the read or write operations and supplies the address for the memory. The peripheral unit can then

Communicate with memory through data Bus for direct transfer between two units while CPU is momentarily disabled.

For each word transferred, the DMA increments its address register and decrements its word Count register.

If word Count does not reach zero, the DMA checks the request line coming from peripheral. For a high speed device the line will be active as soon as previous transfer is Completed.

If peripheral speed is slower, the DMA request may come some what later. In this case, the DMA disable the bus request line so that the CPU can Continue to execute its program. When peripheral requests a transfer, the DMA requests the buses again.

If word Count register reaches zero, the DMA stops any further transfer and removes its bus request. It also informs the CPU of termination by interrupt. When CPU responds to the interrupt, it reads the Content of word Count register. The zero value of this register indicates that all words were transferred successfully.

Input-output Processor (IOP)

IOP may be classified as a processor with Direct Memory Access Capability that Communicate with I/O devices.

IOP is similar to a CPU except that it is designed to handle the details of I/O Processing. Unlike the DMA Controller that must be set up entirely by CPU, IOP can fetch and execute its own instructions. IOP instructions are spcifically designed to facilitate I/O Transfers.

The IOP provides a path for transferring of data between various peripheral devices and memory unit. The CPU is usually assigned the task of initiating the I/O program. From then on IOP operates independent of the CPU and continues to transfer data from external devices and memory.

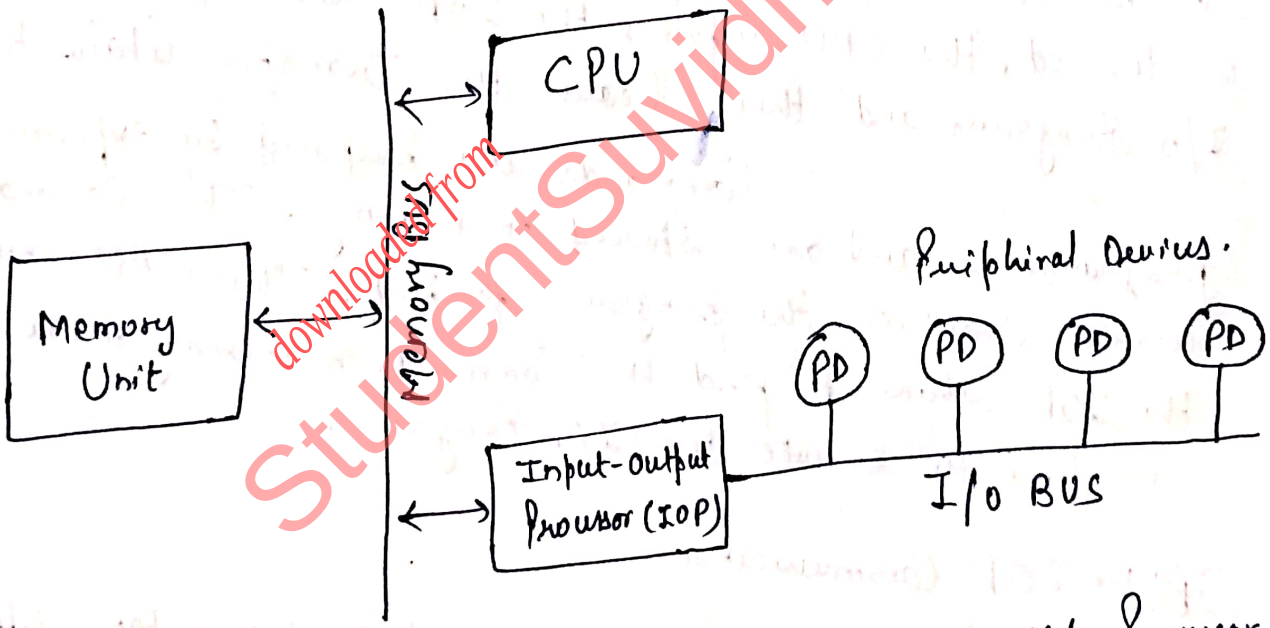


fig:- Block Diagram of Computer with I/O Processor

The Data formats of peripheral devices differ from memory and CPU data formats. The IOP must structure data words from many different sources. Example, it may necessary to take 4 bytes from Input device and pack them into one 32 bit word before transfer to memory.

Data are gathered in IOP at device rate and bit capacity while CPU is executing its own program. After input data are assembled into memory word, they are transferred from IOP directly into memory by "stealing" one memory cycle from CPU.

The Communication between IOP and devices attached to it is similar to the program control method of transfer. In most Computer system, the CPU is master while IOP is slave processor. The CPU is assigned the task of initiating all operations, but I/O instructions are executed in the IOP. CPU instructions provides operations to start I/O Transfer and also to test I/O status Conditions needed for making decisions on various I/O activities by means of interrupt. It also responds to CPU requests by placing a status word in a prescribed location in memory to be examined later by CPU program. When I/O operation is desired, the CPU informs the IOP where to find the I/O program and then leaves the transfer details to IOP.

Commands are prepared by experienced programmers and are stored in memory. The Command words constitute the program for IOP. The CPU informs the IOP where to find the Commands in memory when it is time to execute the I/O program.

CPU - IOP Communication:-

The Communication between CPU and IOP may take different forms, depending on the particular Computer considered. In most cases, memory unit acts as a message center where each processor leaves information for the other.

CPU operations.

IOP operations.

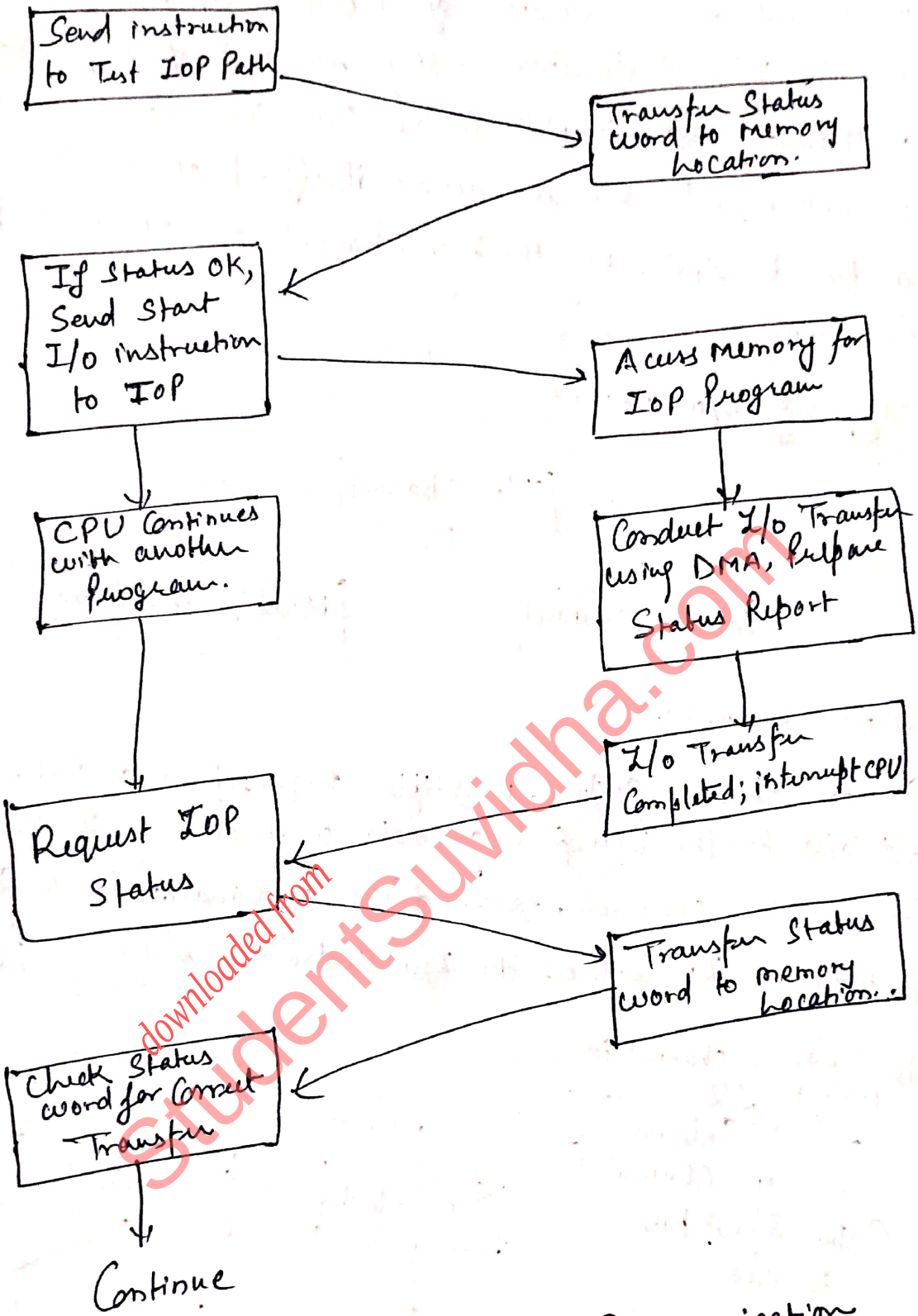
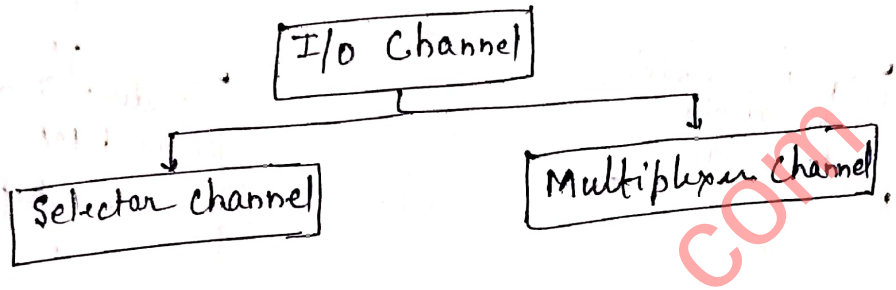


fig:- CPU - IOP Communication

I/O Channel and its Types:

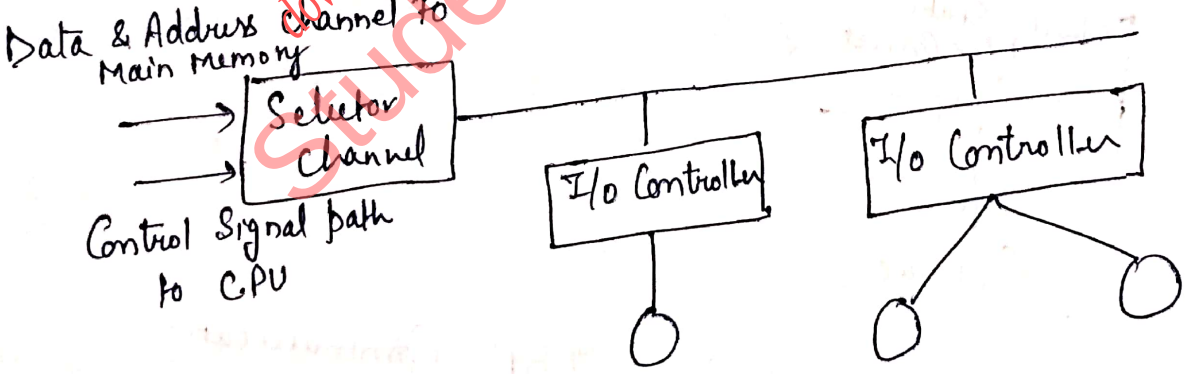
I/O channel is an extension of DMA concept. It has ability to execute I/O instructions using special purpose processor on I/O channel and complete control over I/O operations. Processor does not execute I/O instructions itself. Processor initiates I/O transfer by instructing the I/O channel to execute a program in memory.

Types of I/O channel:



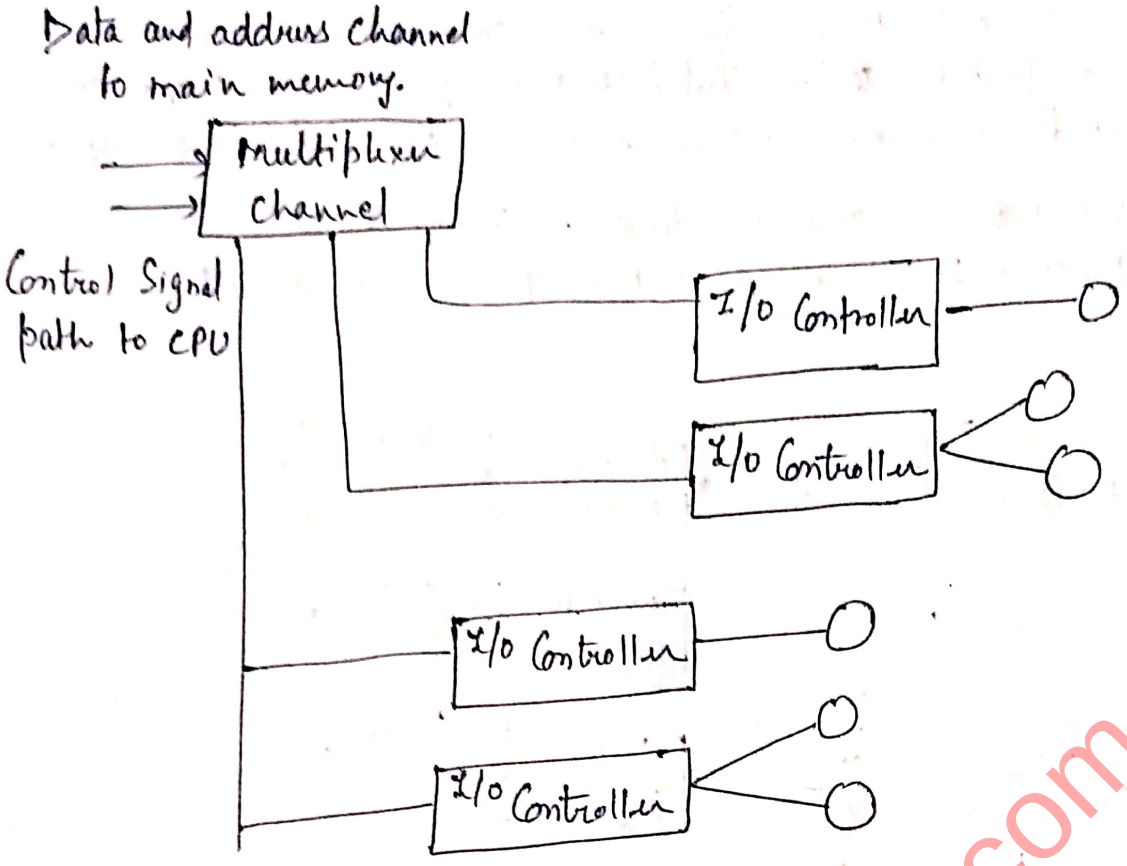
Selector channel:

Selector channel controls multiple high speed devices. It is dedicated to the transfer of data with one of the devices. In Selector channel each device is handled by a controller or I/O module. It controls the I/O controllers shown in figure.



Multiplexer channel:

Multiplexer channel is a DMA controller that can handle multiple devices at the same time. It can do block transfers for several devices at once.



Two Types of multiplexers are used in this channel:

1) Byte Multiplexer:- It is used for low speed devices. It transmits or accepts characters. Interleaves bytes from several devices.

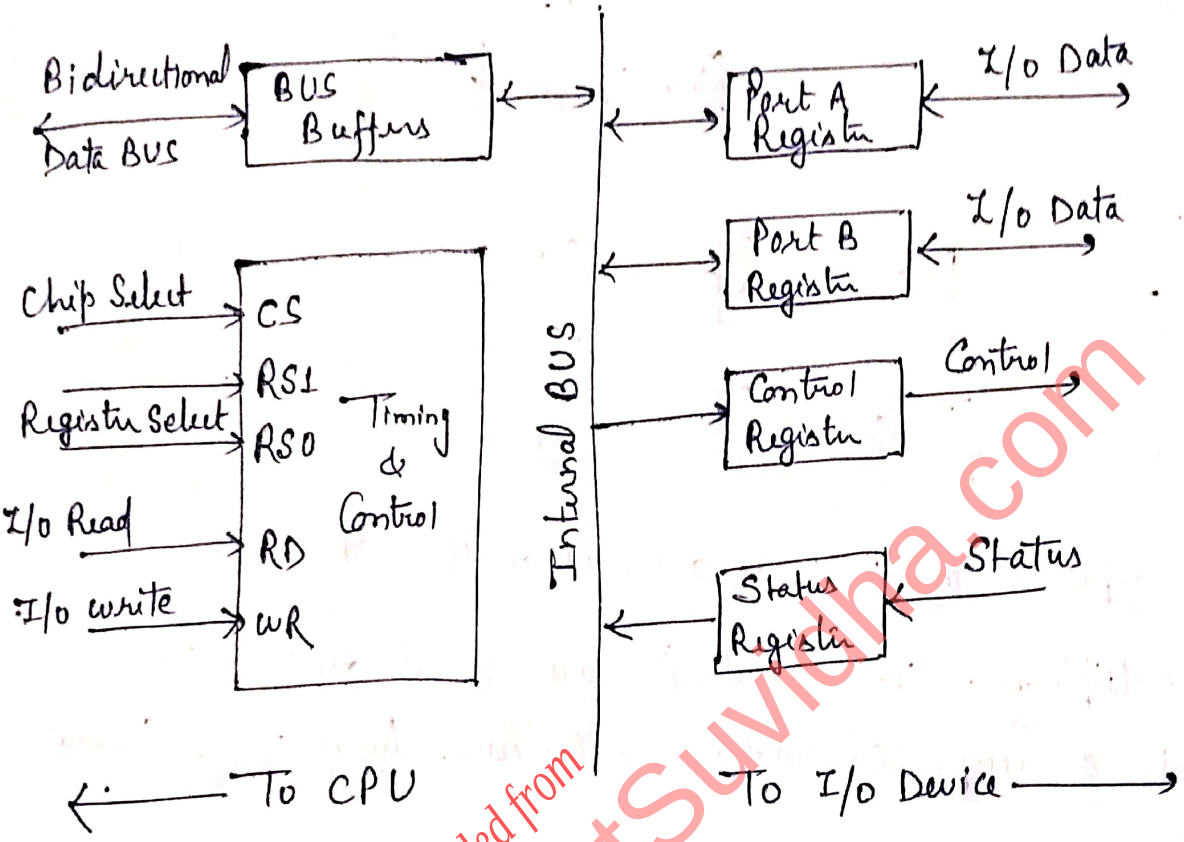
2) Block Multiplexer:- It accepts or transmits blocks of characters. Interleaves blocks of bytes from several devices. Used for High speed devices.

Example of I/O Interface:- (I/O Ports)

It Consists of two data registers called Ports, a Control Register, a Status Register, bus buffers and timing and control circuits. The Interface Communicates with CPU through Data bus. The chip Select and register Select inputs determines the address assigned to the Interface. The I/O Read and write are two Control lines that specify an input or output respectively.

The 4 Registers communicate directly with I/O device attached to the Interface. The I/O Data to and from device can be transferred into either port A or Port B. The Interface may operate with output device or with Input device or device which require both Inputs and output.

Printer → output device, Character Reader → Input device,
Magnetic Disk → Input & output both but not at same time.



CS	RS1	RS0	Register Selected
0	X	X	None: Data BUS in High Impedance
1	0	0	Port A Register
1	0	1	Port B Register
1	1	0	Control Register
1	1	1	Status Register

Fig: Example of I/O Interface Unit

In a System like this, the function code in I/O BUS is not needed because Control is sent to the Control Register, Status ~~Register~~ information is received from Status Register, data are transferred to and from Ports A and B Register.

The Control Register receives control information from CPU. By loading appropriate bits into control Register, interface and I/O device attached to it can be placed in a variety of modes.

Example: Port A may be input Port and Port B may be Output port. ~~the B~~

The Bits in Status Register are used for Status conditions and for recording errors that may occur during Data Transfer.

The Interface Register communicate with CPU through Bi-directional Data BUS. The Address BUS selects interface unit through chip Select and Two register Select inputs. A circuit must be provided externally (decoder) to detect address assigned to interface registers.

This circuit enables the Chip Select (CS) input when interface is selected by address BUS. The Two register Select inputs RS1 and RS0 are usually connected to the 2 least significant lines of address bus. These two inputs select one of 4 registers in interface specified in diagram.

The Content of Selected register is transferred into CPU via Data bus when I/O read signal is enabled. The CPU transfers binary information into selected register via data bus when I/O write input is enabled.

Interrupt and its Types

An interrupt is a signal from a device attached to a computer or from a program within the computer that requires the **operating system** to stop and figure out what to do next.

Interrupt systems are nothing but while the **CPU** can process the programs if the CPU needs any IO operation. Then, it is sent to the queue and it does the CPU process. Later on Input/output (I/O) operation is ready.

The I/O devices interrupt the data which is available and does the remaining process; like that interrupts are useful. If interrupts are not present, the CPU needs to be in idle state for some time, until the IO operation needs to complete. So, to avoid the CPU waiting time interrupts are coming into picture.

Processor handle interrupts

Whenever an interrupt occurs, it causes the CPU to stop executing the current program. Then, comes the control to interrupt handler or interrupt service routine.

These are the steps in which ISR handles interrupts. These are as follows –

Step 1 – When an interrupt occurs let assume processor is executing i'^{th} instruction and program counter will point to the next instruction $(i+1)^{th}$.

Step 2 – When an interrupt occurs the program value is stored on the process stack and the program counter is loaded with the address of interrupt service routine.

Step 3 – Once the interrupt service routine is completed the address on the process stack is popped and placed back in the program counter.

Step 4 – Now it executes the resume for $(i+1)^{th}$ line.

Types of interrupts

There are two types of interrupts which are as follows –

Hardware interrupts

The interrupt signal generated from external devices and i/o devices are made interrupt to CPU when the instructions are ready.

For example – In a keyboard if we press a key to do some action this pressing of the keyboard generates a signal that is given to the processor to do action, such interrupts are called hardware interrupts.

Hardware interrupts are classified into two types which are as follows –

Maskable Interrupt – The hardware interrupts that can be delayed when a highest priority interrupt has occurred to the processor.

Non Maskable Interrupt – The hardware that cannot be delayed and immediately be serviced by the processor.

Software interrupts

The interrupt signal generated from internal devices and software programs need to access any system call then software interrupts are present.

Software interrupt is divided into two types. They are as follows –

Normal Interrupts – The interrupts that are caused by the software instructions are called software instructions.

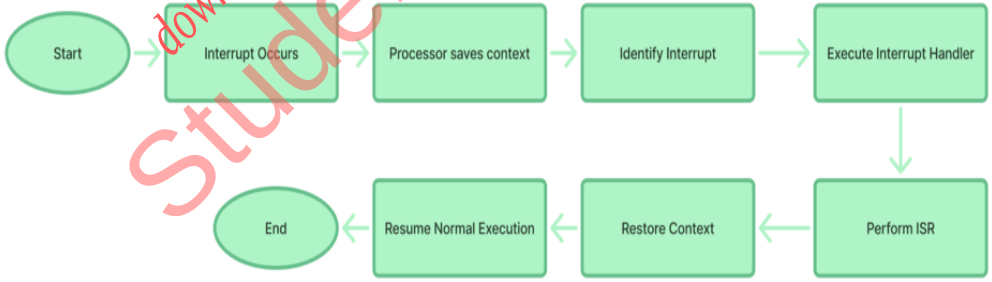
Exception – Exception is nothing but an unplanned interruption while executing a program. For example – while executing a program if we got a value that is divided by zero is called an exception.

Sequences of Events Involved in Handling an IRQ (Interrupt Request)

- Devices raise an IRQ.
- The processor interrupts the program currently being executed.
- The device is informed that its request has been recognized and the device deactivates the request signal.
- The requested action is performed.
- An interrupt is enabled and the interrupted program is resumed.

Flowchart of Interrupt Handling Mechanism

The Image below depicts the flowchart of interrupt handling mechanism



Step 1:- Any time that an interrupt is raised, it may either be an I/O interrupt or a system interrupt.

Step 2:- The current state comprising registers and the program counter is then stored in order to conserve the state of the process.

Step 3:- The current interrupt and its handler is identified through the interrupt vector table in the processor.

Step 4:- This control now shifts to the interrupt handler, which is a function located in the kernel space.

Step 5:- Specific tasks are performed by Interrupt Service Routine (ISR) which are essential to manage interrupt.

Step 6:- The status from the previous session is retrieved so as to build on the process from that point.

Step 7:- The control is then shifted back to the other process that was pending and the normal process continues.

What is Interrupt Latency?

The amount of time between the generation of an interrupt and its handling is known as interrupt latency. The number of created interrupts, the number of enabled interruptions, the number of interrupts that may be handled, and the time required to handle each interrupt all affect interrupt latency.

When the device generating the interrupt needs a specific length of time to generate the interrupt, interrupt latency is required. For instance, if a printer is printing paper, the computer needs to stop the printing program and wait for the document to finish printing. The interrupt latency is the amount of time the computer has to stop the program from operating.

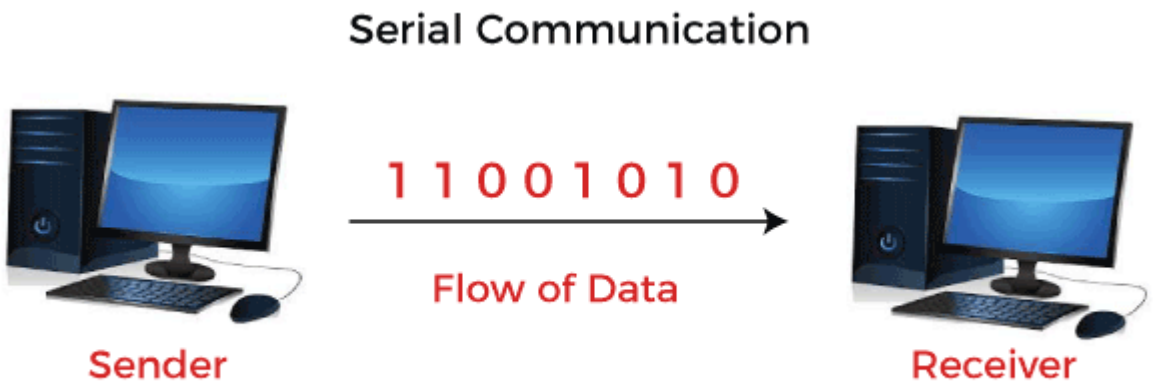
How CPU React when Interrupt occurs?

- **Interrupt Detection:** The CPU continuously video displays unit interrupt lines or alerts from diverse resources, consisting of hardware gadgets or software program commands, to hit upon interrupt requests.
- **Interrupt Acknowledgment:** Upon detecting an interrupt request, the CPU acknowledges the interrupt using sending an acknowledgment sign to the interrupting device or software program.
- **Interrupt Handling:** The CPU identifies the form of interrupt primarily based on its supply, together with a hardware interrupt from a device or a software interrupt from a training. It then seems the cope with the corresponding interrupt handler habitual within the interrupt vector desk.
- **Context Saving:** Before moving manipulate to the interrupt handler ordinary, the CPU saves the present-day execution context, inclusive of the program counter (PC), processor state, and any applicable sign-in contents, onto the stack or in the devoted garage.
- **Transfer Control:** The CPU transfers manipulation to the interrupt handler ordinary with the aid of placing this system counter (PC) to the address of the handler habitual retrieved from the interrupt vector desk.
- **Interrupt Servicing:** The interrupt handler habitual executes to carrier the interrupt. It plays responsibilities to interrupt, such as reading facts from a device, processing enter/output operations, or coping with a software program request.

Serial Communication: Synchronous & Asynchronous Communication

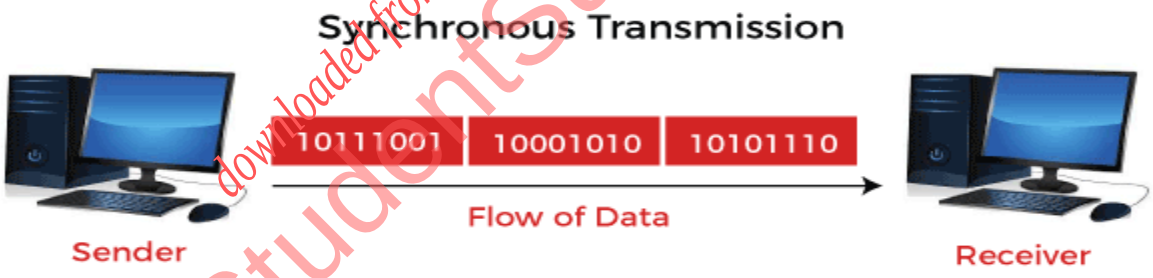
Serial communication is the process of sequentially transferring the information/bits on the same channel. Due to this, the cost of wire will be reduced, but it slows the transmission speed. Generally, communication can be described as the process of interchanging information between individuals in the form of audio, video, verbal words, and written documents. The serial protocol is run on every device that can be our mobile, personal computers, and many more with the help of following some protocols. The protocol is a type of reliable and secure form of communication that contains a set of rules addressed with the help of a source host and a destination host. In serial communication, binary

pulses are used to show the data. 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.



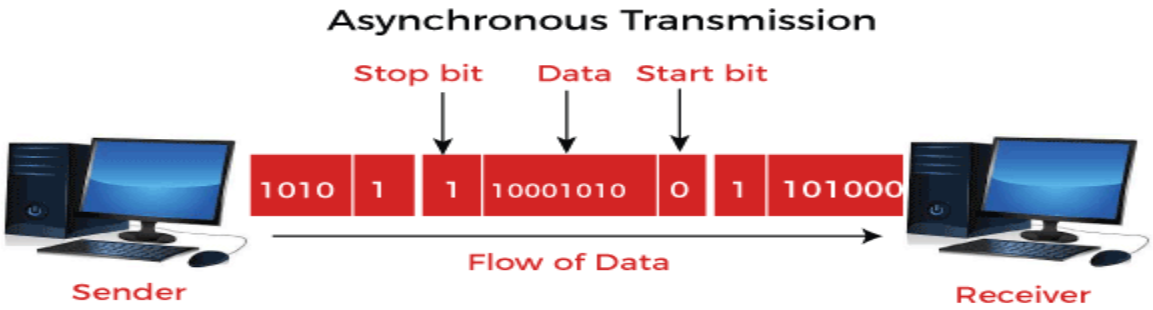
Synchronous Communication

In synchronous communication, the frames or data will be constructed with the help of combining the groups of bits. That frames will be continuously sent in time with a master clock. It uses a synchronized clock frequency to operate the data of 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 that's why the frequency of timing error will be less, and the data will move faster. On the basis of the timing being synced correctly between the sender and receiver devices, the data accuracy is totally 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.



On the basis of the data transfer rate and the type of transmission mode, serial communication will take many forms. The transmission mode can be classified into simplex, half-duplex, and full-duplex. Each transmission mode contains the source, also known as sender or transmitter, and destination, also known as the receiver.

Transmission Mode

The transmission modes are described as follows:

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. In another case, if the receiver sends the data, the sender will only accept it. The sender cannot reply back to the receiver.

There are various examples of the simplex.

Example 1: Keyboard and **CPU** are the best examples of a simplex. The keyboard always transmits characters to the CPU (Central processing unit), but the CPU does not require transmitting characters or data to the keyboard.

Example 2: Printers and **computers** are one more example of the simplex. Computers always send data to the printers, but printers are not able to send the data to the computers. In some cases, printers can also talk back, and this case is an exception. There is only one lane in the simplex.

Half Duplex

In the half-duplex, 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. **For example,** the **internet** is a good example of half-duplex. With the help of internet, if a user sends a web page request to the web server, the server processes the application and sends the requested page to the user.

One lane bridge can also explain the half-duplex. In a one-lane bridge, the two-way vehicles will provide the way so that they can cross. At a time, only one end will send, and the other end will only receive. We can also perform the error correction that means if the information received by the receiver is corrupted, then it can again request the sender to retransmit that information. **Walkie-talkie** is also a classic example of half-duplex. Both ends of walkie talkie contain the speakers. We can use each handset or walkie talkie to either send the message or receive it, but we cannot do both things at the same time.

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. In this mode, signals travelling in one direction are able to share the capacity of links with signals travelling in the opposite directions. There are two ways in which sharing can occur, which is described as follow:

Either capacity of the link is divided into the signals going in both directions.

Or the links have two physically separated transmission parts. Where one part can be used for sending, and another part can be used for receiving.

If we need communication in both directions constantly, in this case, we will use the full-duplex mode. The capacity of the channel will be split into two directions.

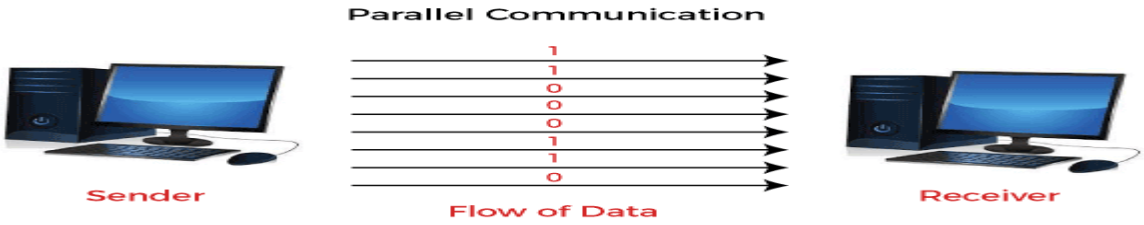
Examples: Telephone Network is a good example of full-duplex mode. While using the telephone or phone, the two persons are able to talk and hear both things at the same time. The **ordinary two-lane highway** is helpful to explain the full-duplex. If traffic is very much, in this case, the railroad is decided to lay a double tack which is used to allow trains to pass in both directions.

Audio calling or Video calling is also an example of full-duplex. In audio or video calling, two persons are able to communicate at the same time. In audio calling, they are able to speak and listen at the same time, and in video calling, they are able to communicate by seeing each other at the same time.

Parallel Communication

There is another type of communication known as parallel communication, which is described as follows:

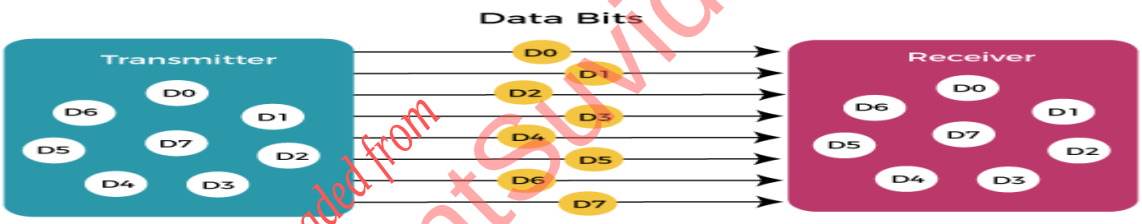
Parallel communication is used to transmit a huge amount of data signals simultaneously on the different channels within the same radio path or cable at a time. It is used to comprise a huge amount of wired channels in parallel. In parallel communication, the data transfer between sender and receiver is done with the help of multiple channels. The data bus in the parallel devices is wider as compared to the serial devices. That's why it can transfer the data from source to destination at a time. The parallel transmission bit rate is higher as compared to the serial transmission bit rate.



The costs of multiple wires are higher as compared to the single wire. The parallel cable gets longer that's why it requires a high cost. If the distance is larger, synchronization timing between more than one channel becomes more sensitive. A constant clocking signal is used to provide the timing in parallel communication. The signal is sent with the help of a separate wire within the parallel cable. So we can say parallel communication is synchronous.

Working of Parallel communication

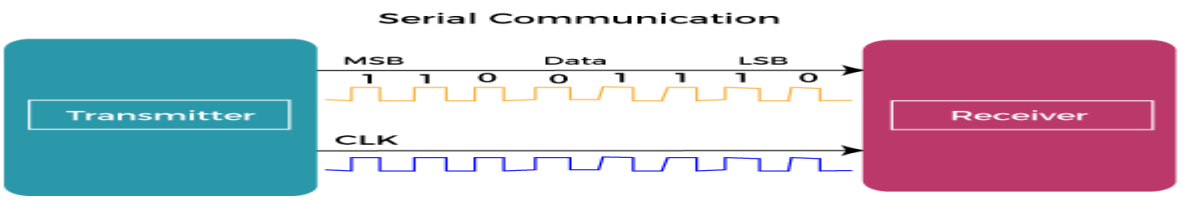
The parallel communication uses the various parallel paths (wires) to transfer many bits once within the same cable in synchronization with the help of a single clock. The clock uses these parallel paths and specifies the timing for transmission in the form of a constant clocking signal.



A huge amount of bits are transferred at the same time with the help of various parallel paths. There are different ways of an order of received bit string, and it depends on various factors like available bandwidth, source distance, and location. The skipping in video calls and internet calls is an example of it.

Difference between Serial communication and Parallel communication

The serial communication is able to send only one bit at a single time. That's why serial communication needs fewer input/output lines. It also occupies more resistance and less space to cross talk. Serial communication has the bigger advantage that the cost to build the whole embedded system becomes very cheap. It can also be able to transmit the data over a long distance with the help of only a single wire or line.



In the DCE devices (Data communication Equipment) such as a modem, serial communication is mostly used. All the major computer networks or communication mostly prefer serial communication. Nowadays, the most common and popular mode for small distance is serial buses because the parallel buses disadvantages prevail over their advantage of simplicity.

Advantages of Serial communication over Parallel communication

Mostly people have a misconception that the parallel ports/buses are faster than the serial ports/buses because, in serial communication, the data transmission is only a bit per unit of time. Even the parallel buses will be clocked considerably at a slower rate as compared to the serial buses. There are various factors that are used to specify that serial communication is better than parallel communication, which is described as follows:

No Clock Required: If there is a case of the asynchronous and unclocked type of serial communication, the problem related to the clock skew between channels will not exist.

Need less Space: At the time of configuration of serial communication, it needs less amount of space because the serial connection needs less amount of cable. Hence because of this feature, we will have an additional space, which can be used to provide better isolation of data lanes/channels from the components of neighboring communication.

No Cross Talks: The serial communication contains fewer amounts of conductors in the nearby space. That's why the possibility of cross-talk is rare.

Low cost: The serial communication contains the serial link. The cost of this link is less than the parallel link.