LECTURE-

Content : Peripherals, Input-output Interface

Peripherals :

The input-output subsystem of a computer, referred to as I/O, provides an efficient mode of communication between the central system and the outside environment.

Devices that are under the direct control of the computer are said to be connected on-line. These devices are ~~designated~~ designed to read information into or out of the memory unit upon command from the CPU and are considered to be part of the total computer system. Input or output devices attached to the computer are also called peripherals. There are three types of peripherals such as input, output and serial or parallel.

Amongst the most common peripherals are keyboards, display unit, and printers.

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 the communicatio link is to resolve the differences that exist between the central computer and each peripheral. The major differences are:

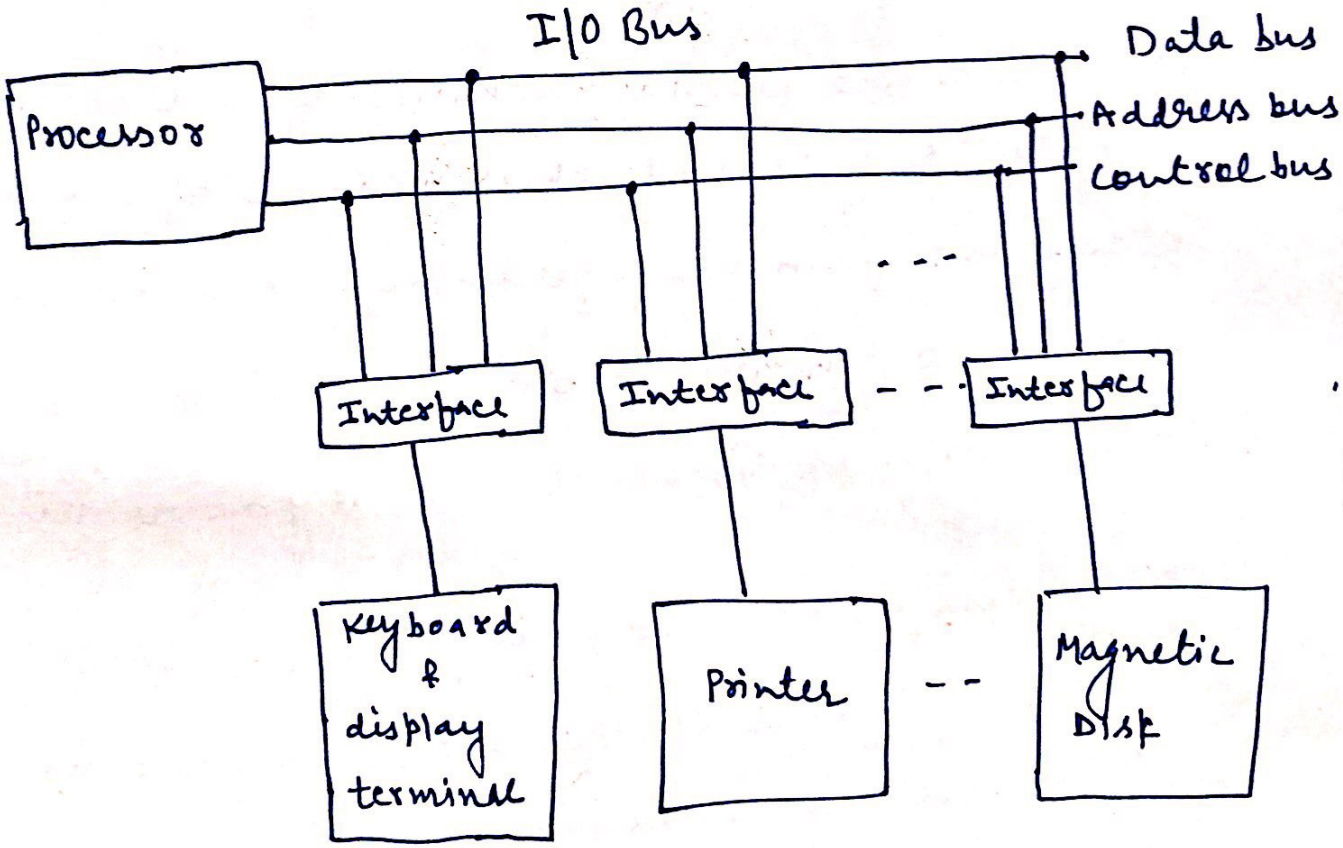
1- Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and Memory, which are electronic devices. Therefore, a conversion of signal values may be required.

2- Data tranfer rate of peripherals is usually slower than that of CPU. Therefore, a synchronization mechanism may be needed.

3- Data codes and formats in peripherals differ from the word format in the 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 peripherals connected to the CPU.

To overcome or resolve the above differences, special hardware components are placed between CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface unit.



Connection of I/O bus to Input-Output devices

Each peripheral device has an interface unit, which decodes the address and control received from the I/O bus interprets them for the peripheral, and provides signal for peripheral controller. It also synchronizes the data from and supervises the transfer between peripherals and processor.

To communicate with a particular device, the processor place a device address on the address lines. Each interface attached to the I/O bus contains an address decodes that monitors the address lines. When the interface detects its own address, it activates the both between the bus lines and the devices that it controls. All peripherals whose address does not correspond to the address in the bus are disabled by their interface.

The time address is made available in the address lines the processor provides a control signal (I/O command) in the control lines. The interface selected responds to the control signals. The various commands that one interface may receive are –

1. Control command : A control command is issued to activate the peripheral and to inform it what to do. Ex:- Read and write operation for a hard disk.

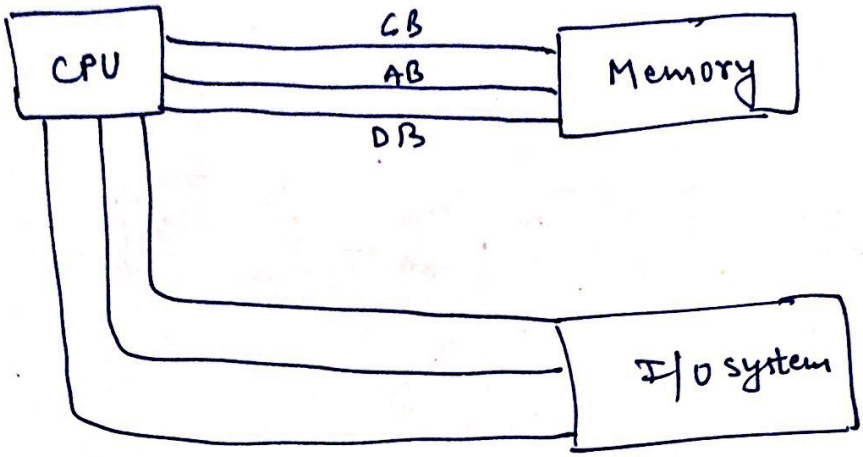
2. Status : It is issued to test various status conditions in the interface and the peripheral.

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

4. Input data : It causes to transfer data from peripheral to interface and then place them on data bus for processor.

The processor like, I/O needs to communicate to the memory. for which it has data, address and read/write control lines. There are three ways that computer buses can be used to communicate with memory and I/O are:

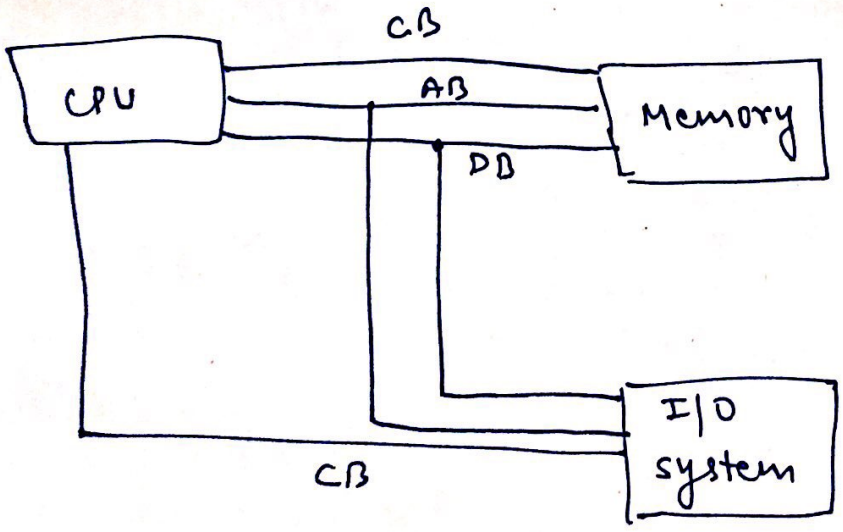
1- Use two separate buses, one for memory and the other for I/O — The computer has independent set of buses for memory and I/O. It also has I/O processor.



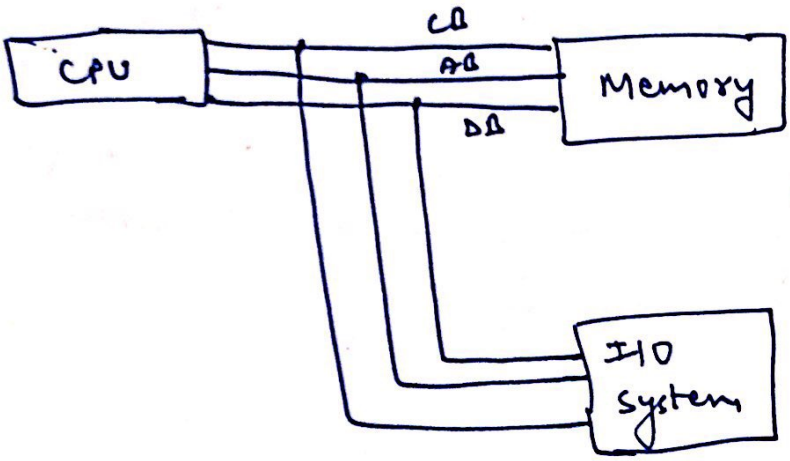
2- We one common bus for both memory and I/O but have separate control lines for each- it uses some address & data bus but separate control lines.

- I/O Read/write
- memory Read/write

It is also called isolated I/O.

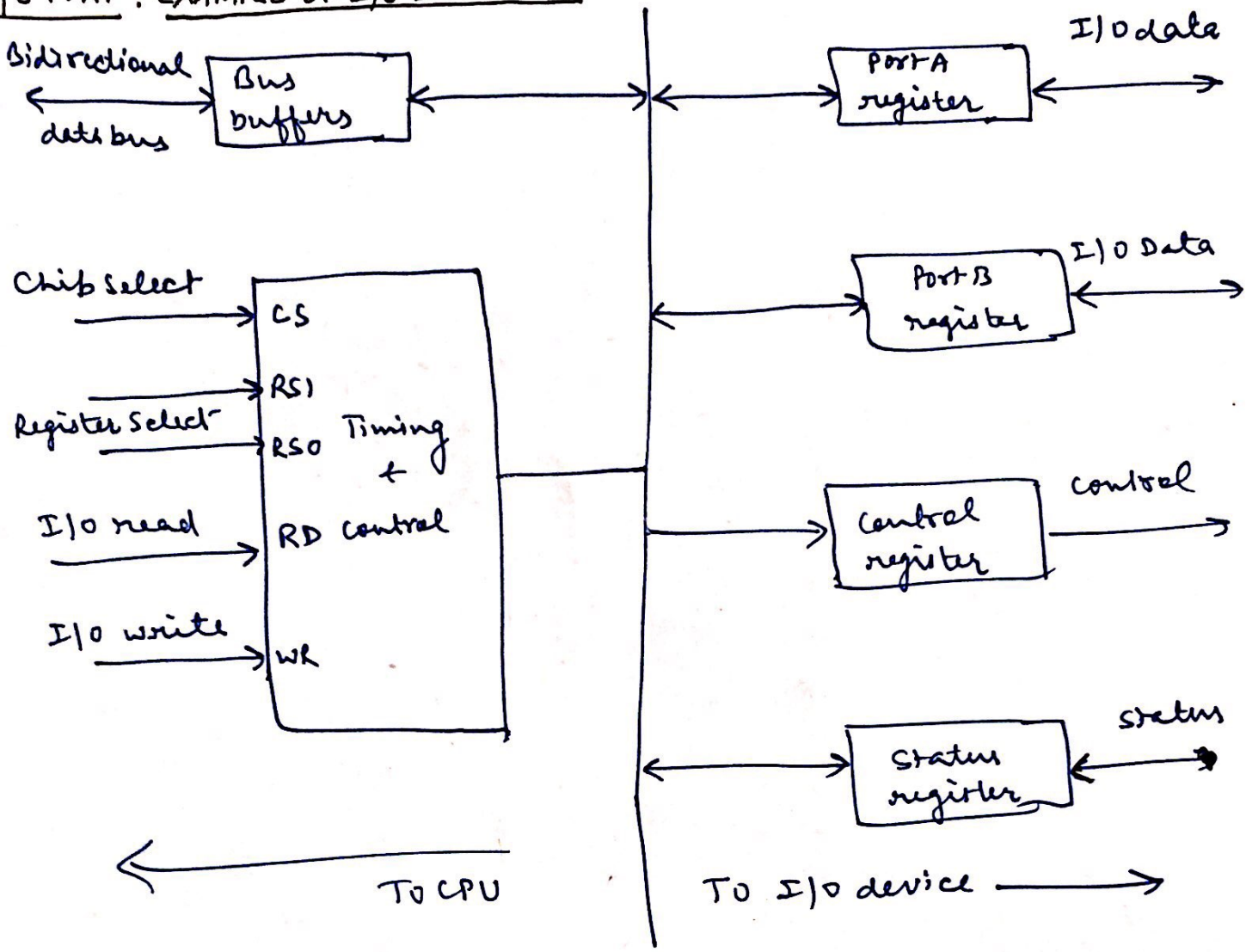


3- Use one common bus for memory and I/O with common control lines - it is also called memory-mapped I/O.



LECTURE

I/O PORT : EXAMPLE OF I/O INTERFACE :



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

Example of I/O Interface unit

LECTURE

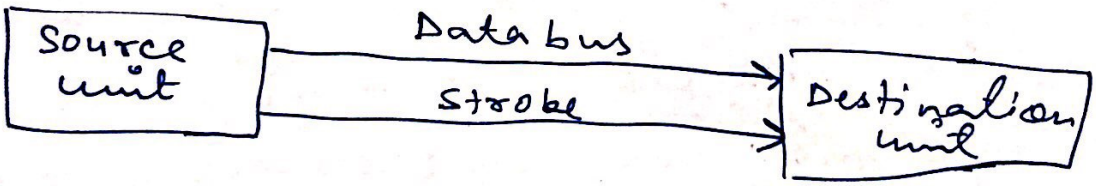
Asynchronous Data Transfer :

As I/O and CPU uses own private clock for internal registers, the timing of each unit is independent from the other, so the data transfer between them is asynchonous data transfer.

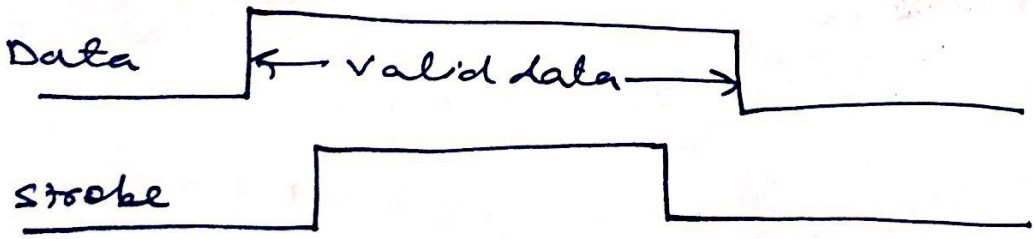
Asynchronous data transfer takes place in two different ways -

① strobe - use single control line to time each transfer. The strobe may be activated by either the source or the destination unit.

ⓐ Source - Initiated Strobe



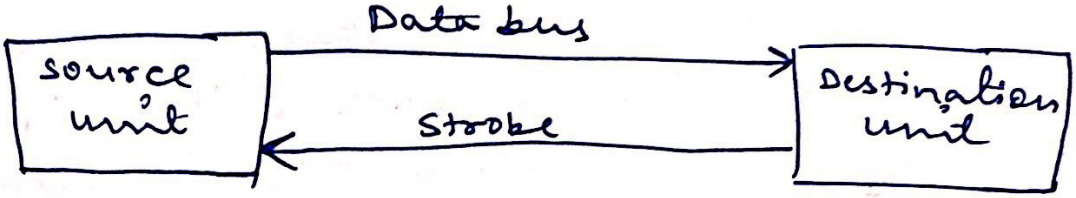
ⓐ Block diagram



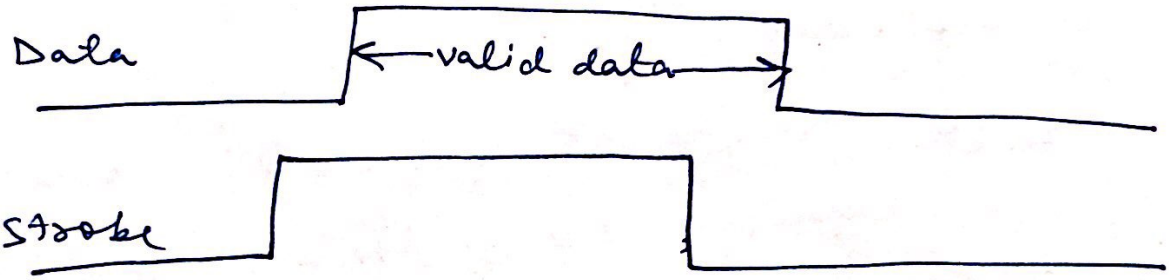
ⓑ Timing diagram

Source - Initiated Strobe for data transfer

(b) Destination-initiated strobe



(a) Block diagram



(b) Timing diagram

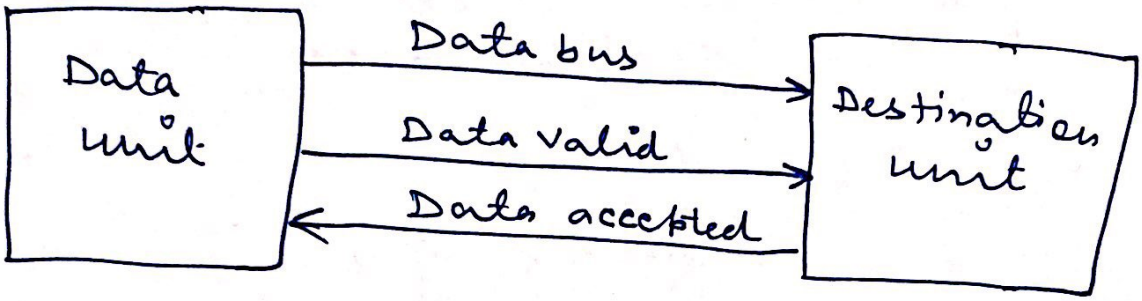
Destination-initiated strobe for data transfer

Disadvantages:

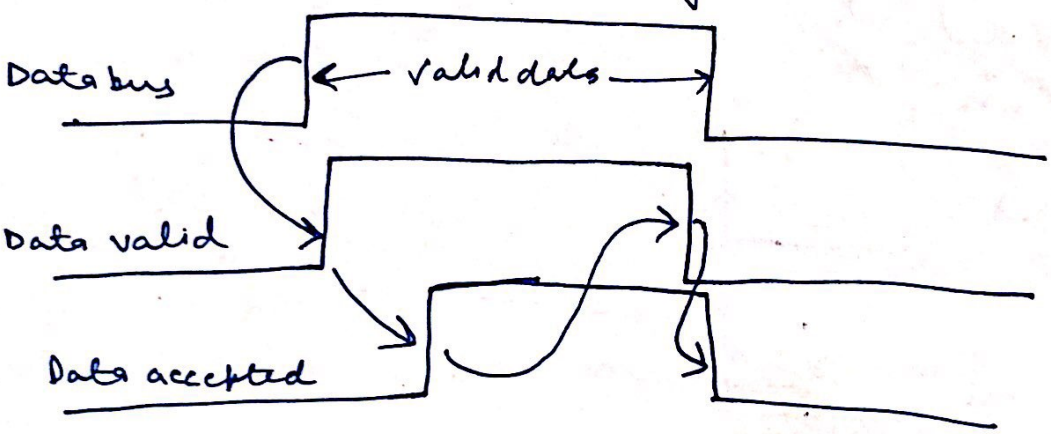
unit initiates the transfer has no way of knowing whether the other unit has actually the data item that was placed in the bus.

Handshaking : use two control signals to time each transfer

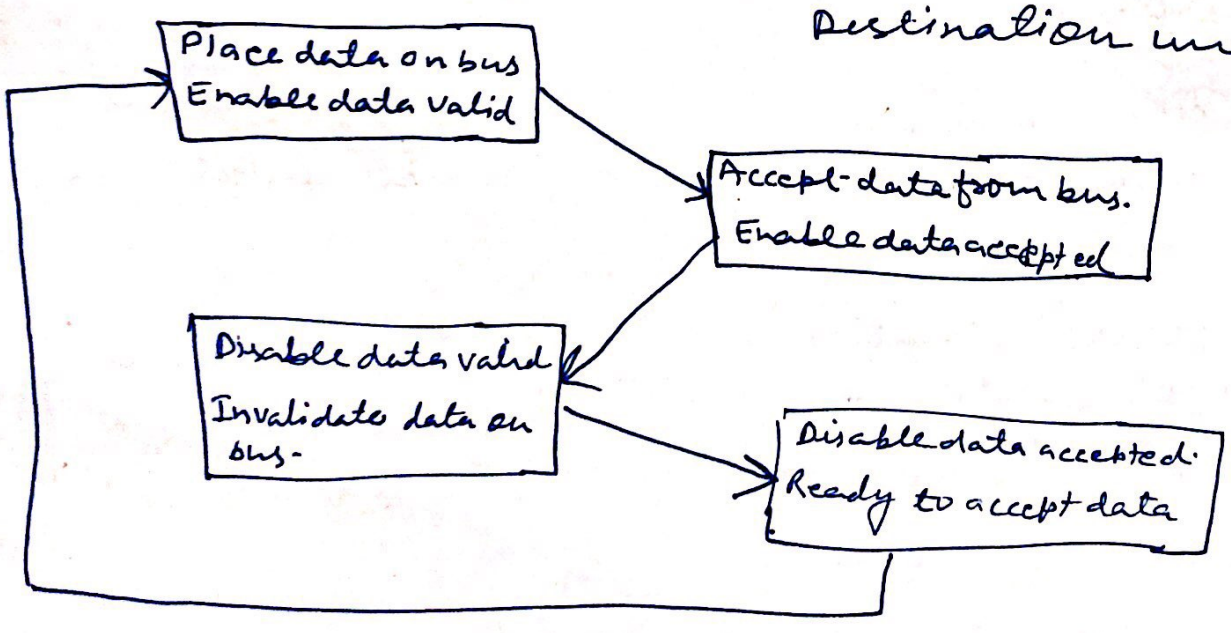
(a) Source-initiated transfer :



(a) Block diagram

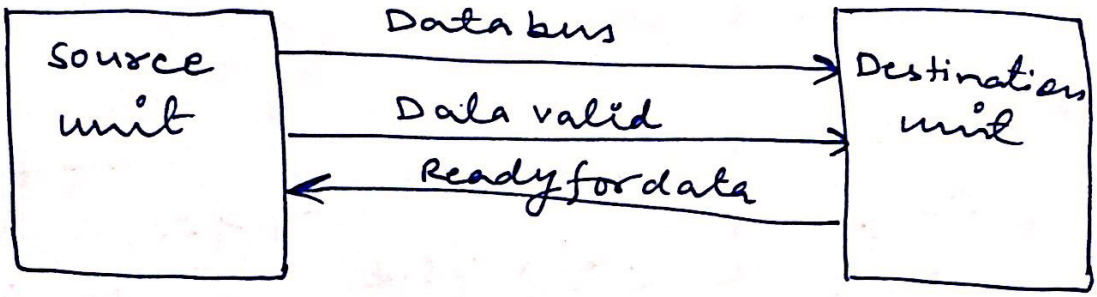


(b) Timing Diagram

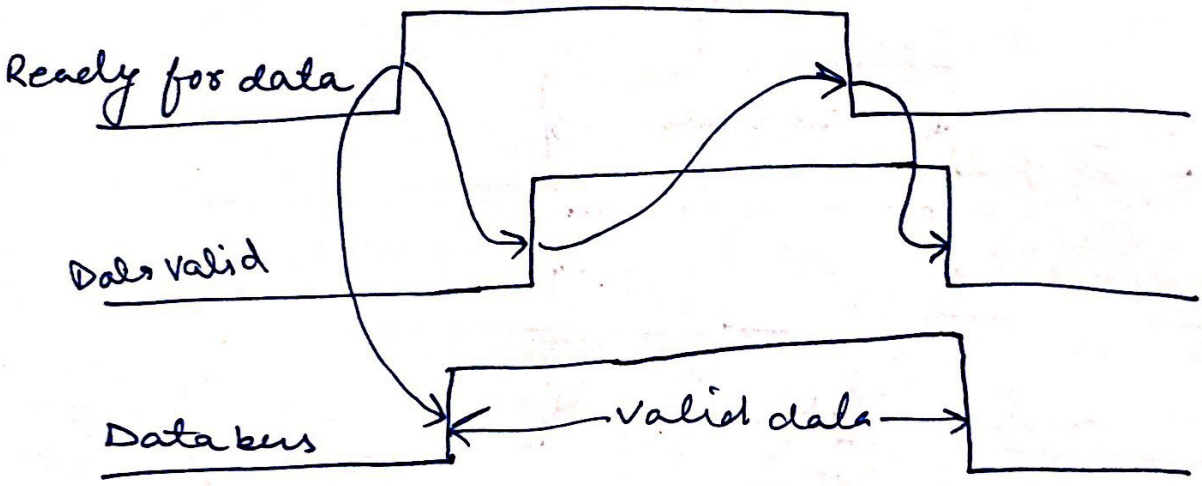


Sequence of Events

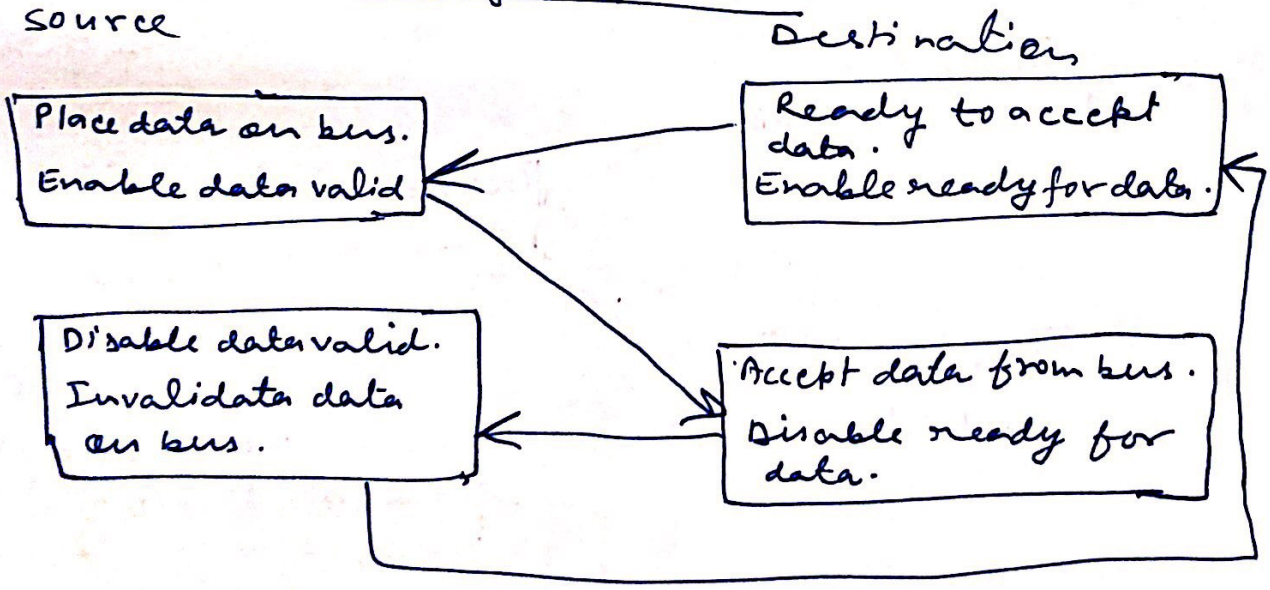
② Destination-initiated transfer



ⓐ Block diagram



ⓑ Timing diagram



ⓒ Sequence of Events

Serial Transfer: In serial transfer each bit in the message is sent in sequence one at a time. It can be -

① Synchronous: Two units share a common clock frequency and bits are transmitted continuously at the rate dictated by the clock pulses.

② Asynchronous: A serial asynchronous technique employs special bits that are inserted at both ends of the character code.

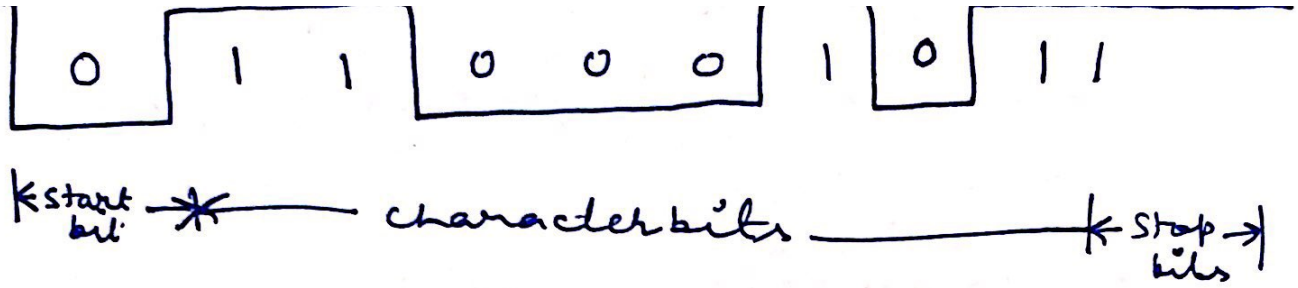
In this each character consists of three parts:

1- A start bit

2. The character bit

3- Stop bits

The convention is that the transmitter rests at the 1-state when no characters are transmitted. The first bit, called the start bit, is always a 0 and is used to indicate the beginning of a character. The last bit called the stop bit is always a 1.



Asynchronous Serial Transmission

LECTURE

Content : Modes of Transfer.

Modes of Transfer :

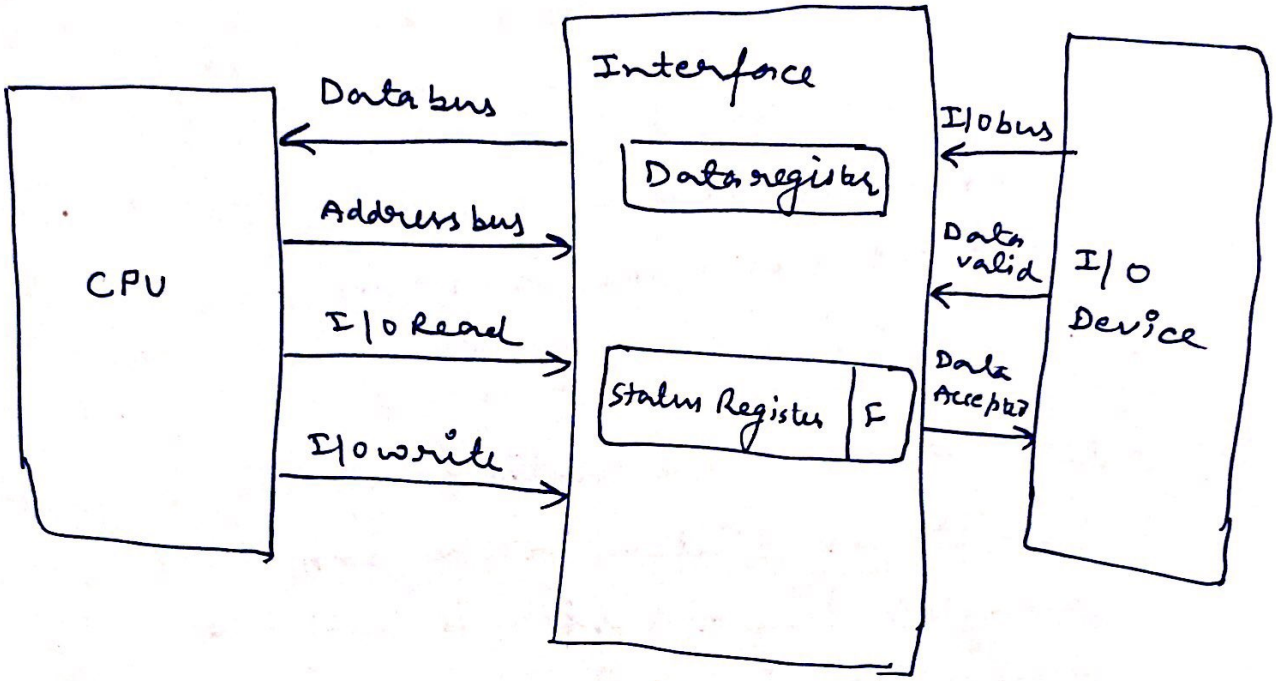
Data transfer to and from peripherals may be handled in one of three possible modes.

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

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

If the speed of an I/O device is in the right range, neither too fast for the processor to read and write the signaling bits nor to slow for the processor to wait for its activity.

An example of data transfer from an I/O device through an interface into the CPU is shown in fig.



F = Flag bit

Data transfer from I/O to CPU

The following operations will occur:

By device: when a byte of data is available, the device places it in the I/O bus and enables its data valid line.

By Interface: Accepts the byte into data register and enables the data accepted line. Sets F.

By device: can now disable the data valid line, but it will not transfer another byte untill the data accepted line is disable by the interface.

By program:

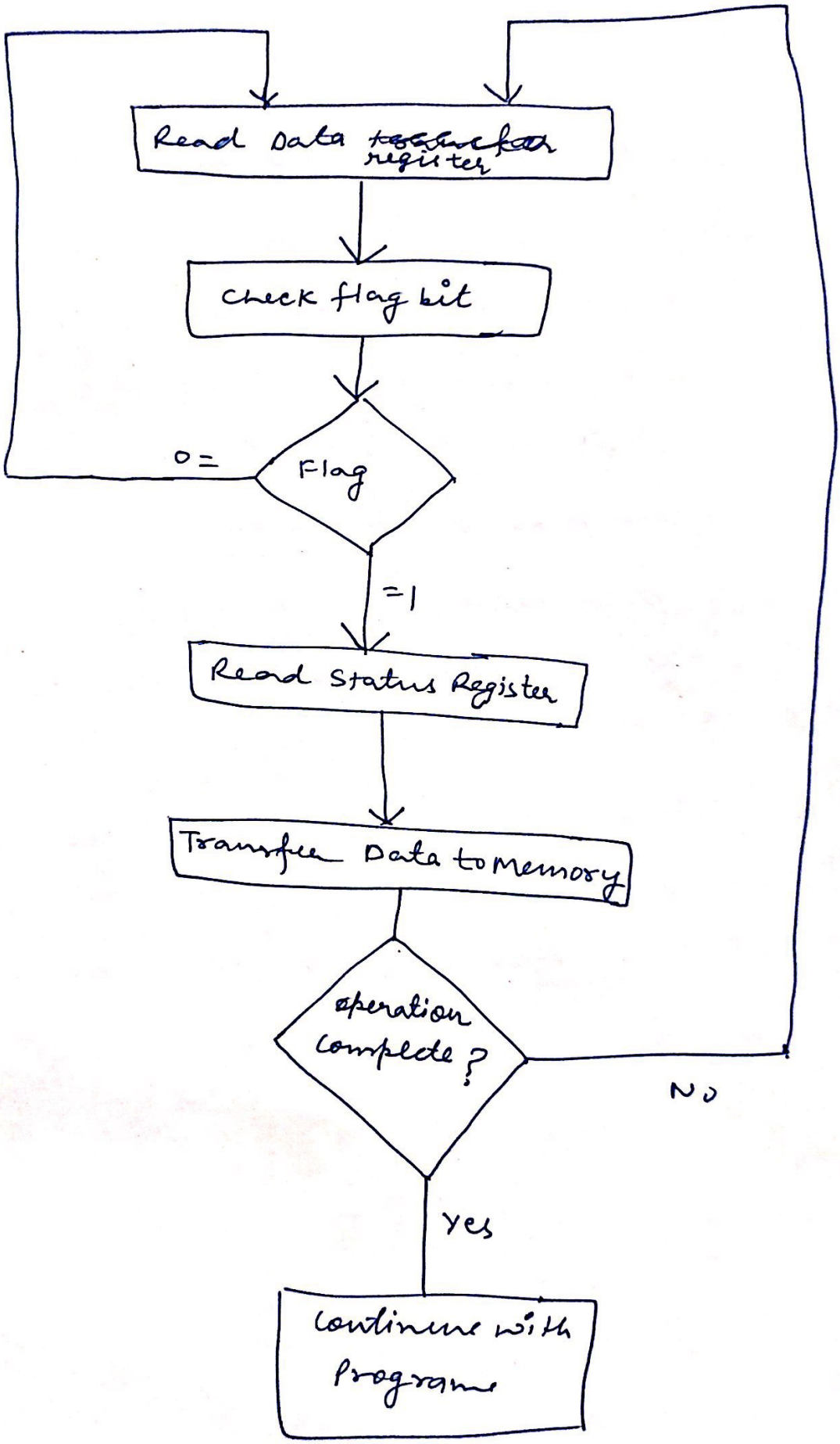
1- Read the status register

2- check the status of the flag bit and branch to step1 if not set or to step 3 if set.

3- Read the data register

The flag bit is then cleared to 0 by either the CPU or the interface, depending on how the interface circuits are designed.

By interface: once the flag is cleared, the interface disables the data accepted line and the device can then transfer the next data byte.



Flow chart for CPU program to input data

(2) - Interrupt - Initiated I/O

An alternative to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data. This mode of transfer uses the interrupt facility. While the CPU is running a program, it does not check the flag. However, when the flag is set, the computer is interrupted from proceeding with the current program and is informed of the fact that the flag has been set. The CPU deviates from what it is doing to take care of the input or output transfer. After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.

For each interrupt there is a service routine, service routine address must be known by CPU to branch to it. This is accomplished by two methods:

Non-vectored Interrupt: The branch address is assigned to a fixed location in memory.

Vectored Interrupt: the source that interrupts supplies the branch information to the computer. This information is called interrupt vector. In some computers the interrupt vector is the first address of the I/O service routine. In other computers the interrupt vector is an address that points to a location in memory where the beginning address of the 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 requests arrive simultaneously. The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced.

Establishing the priority of simultaneous interrupts can be done by software or hardware.

Polling :-

A polling procedure is used to identify the highest-priority source by software means. In this method there is one common branch address for all interrupts. The program that takes care of interrupts begins at the branch address and polls the interrupt-source in sequence. The order in which they are tested determines the priority of each interrupt.

The highest ~~order~~ priority source is tested first, and if its interrupt signal is on, control

branches to a service routine for this source. otherwise, the next-lower-priority source is tested, and so on. Thus the initial service routine for all interrupts consists of a program that tests the interrupt sources in the sequence and branches to one of many possible service routines.

The disadvantage of the software method is that if there are many interrupts, the time required to poll them can exceed the time available to service the I/O device. In this situation a hardware priority-interrupt unit can be used to speed up the operation.

A hardware priority-interrupt unit functions as an overall manager in an interrupt system environment. It accepts interrupt requests from many sources, determines which of the ~~following~~ incoming requests has the highest priority, and issues an interrupt request to the computer based on this determination. To speed up the operation, each interrupt source has its own interrupt vector to access its own service routine directly. Thus no polling is required because all the decisions are established by the hardware priority interrupt unit.

The hardware priority function can be established by either a serial or a parallel connection of interrupt lines. The serial connection is also known as the daisy chaining Method.

Daisy chaining priority:

The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt. The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain. This method of connection between three devices and the CPU is shown in figure.

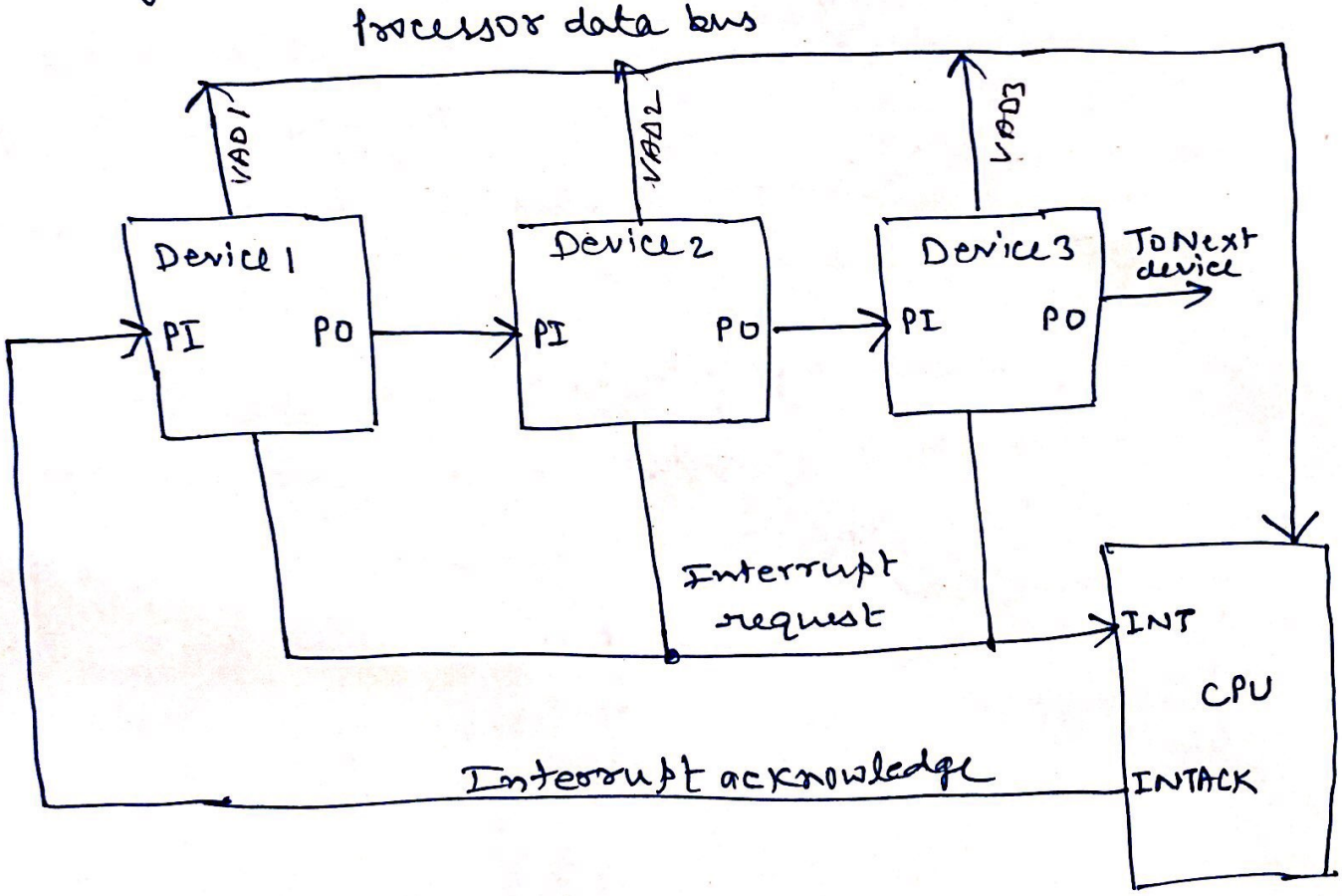


Fig: Daisy-chain priority Interrupt

Parallel Priority Interrupt :

The parallel priority interrupt method uses a register whose bits are set separately by the interrupt signal from each device. Priority is established according to the position of the bits in the register. In addition to the interrupt register, the circuit may include a mask register whose purpose is to control the status of each interrupt request. The mask register can be programmed to disable lower-priority interrupts while a higher-priority device is being serviced. It can also provide a facility that allows a high priority device to interrupt the CPU while a lower-priority device is being serviced.

The priority logic for a system of four interrupt sources is shown in figure. It consists of an interrupt register whose individual bits are set by external conditions and cleared by program instructions.

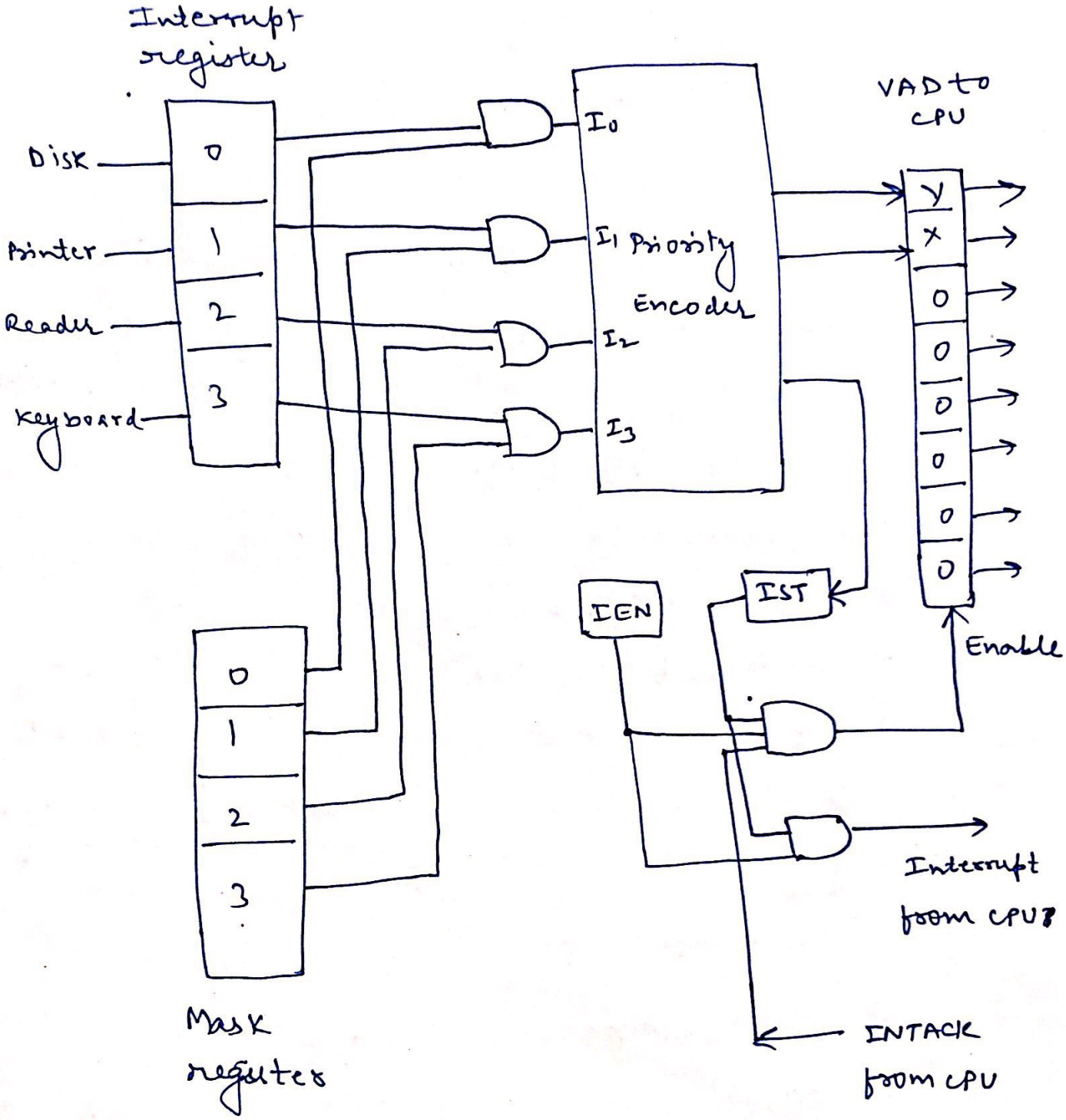


Fig : Priority Interrupt Hardware

Priority Encoder:

The priority encoder is a circuit that implements the priority function. The logic of the priority encoder is such that if two or more inputs arrive at the same time, the input having the highest priority will take precedence. The truth table of a four-input priority encoder is given in table.

Priority Encoder Truth Table

Input				output			Boolean function.
I_0	I_1	I_2	I_3	x	y	IST	
1	x	x	x	0	0	1	$x = I_0' I_1'$
0	1	x	x	0	1	1	$Y = I_0' I_1' + I_0' I_2'$
0	0	1	x	1	0	1	$(IST) = I_0 + I_1 + I_2 + I_3$
0	0	0	1	1	1	1	
0	0	0	0	x	x	0	

Interrupt cycle:

The interrupt enable flip-flop IEN shown in previous figure can be set or cleared by program instructions. When IEN is cleared, the interrupt request coming from IST is neglected by the CPU. The program controlled IEN bit allows the programmer to choose whether to use the interrupt facility.

SP ← SP − 1	Decrement stack pointer
M[SP] ← PC	Push PC into stack
INTACK ← 1	Enable interrupt acknowledge
PC ← RAD	Transfer vector address
IEN ← 0	Disable further interrupt

Go to fetch next instruction.

If an instruction to clear IEN has been inserted in the program, it means that the user does not want this program to be interrupted. An Instruction to set IEN indicates that the Interrupt facility will be used while the current program is running.

At the end of each instruction cycle the CPU checks IEN and the interrupt signal from IST. If either is equal to 0, control continues with the next instruction. If both IEN and IST are equal to 1, the CPU goes to an interrupt cycle.

The CPU pushes the return address from PC into stack. It then acknowledge edges the interrupt by enabling the INTACK line. The priority interrupt unit responds by placing a unique

Direct Memory Access (DMA)

The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripherals 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 the memory buses. A DMA controller takes over the buses to manage the transfer directly between the I/O device and Memory.

The CPU may be placed in an idle state in a variety of ways. one common method extensively used in microprocessors is to disable the buses through special control signals.

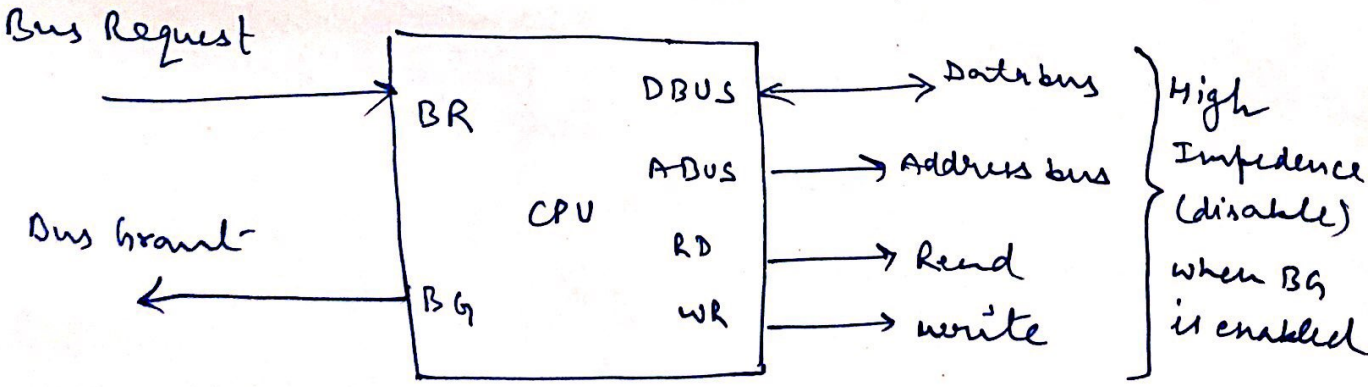


Fig: CPU bus signals for DMA transfer

Figure shows two control signals in the CPU that facilitate the DMA transfer. The bus request (BR) input is used by the DMA controller to request the CPU to relinquish control of the buses. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and write lines into a high-impedence state behave like an open circuit. The CPU activates the bus grant (BG) output to inform the external DMA that the buses are in the high impedence state. The DMA that originated the bus request can now take control of the buses to conduct memory transfer without processor intervention. When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant, takes control of the buses, and returns to its normal operations.

When the DMA takes control of the bus system, it communicates directly with the memory. The transfer can be made in several ways:

* In DMA burst transfer, a block sequence consisting of a number of memory words is transferred in a continuous burst while the DMA controller is master of the memory buses. This mode of transfer is needed for fast devices such as magnetic disks, where the data transmission can not be stopped or slowed down until an entire block is transferred.

* An alternative technique called cycle stealing allows the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow the direct memory I/O transfer to "steal" one memory cycle.

DMA Controller :

The DMA controller needs the usual circuits of an interface to communicate with the 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 the memory. The word count register specifies the number of words that must be transfered. The data transfer may be done directly between the device and memory under control of the DMA.

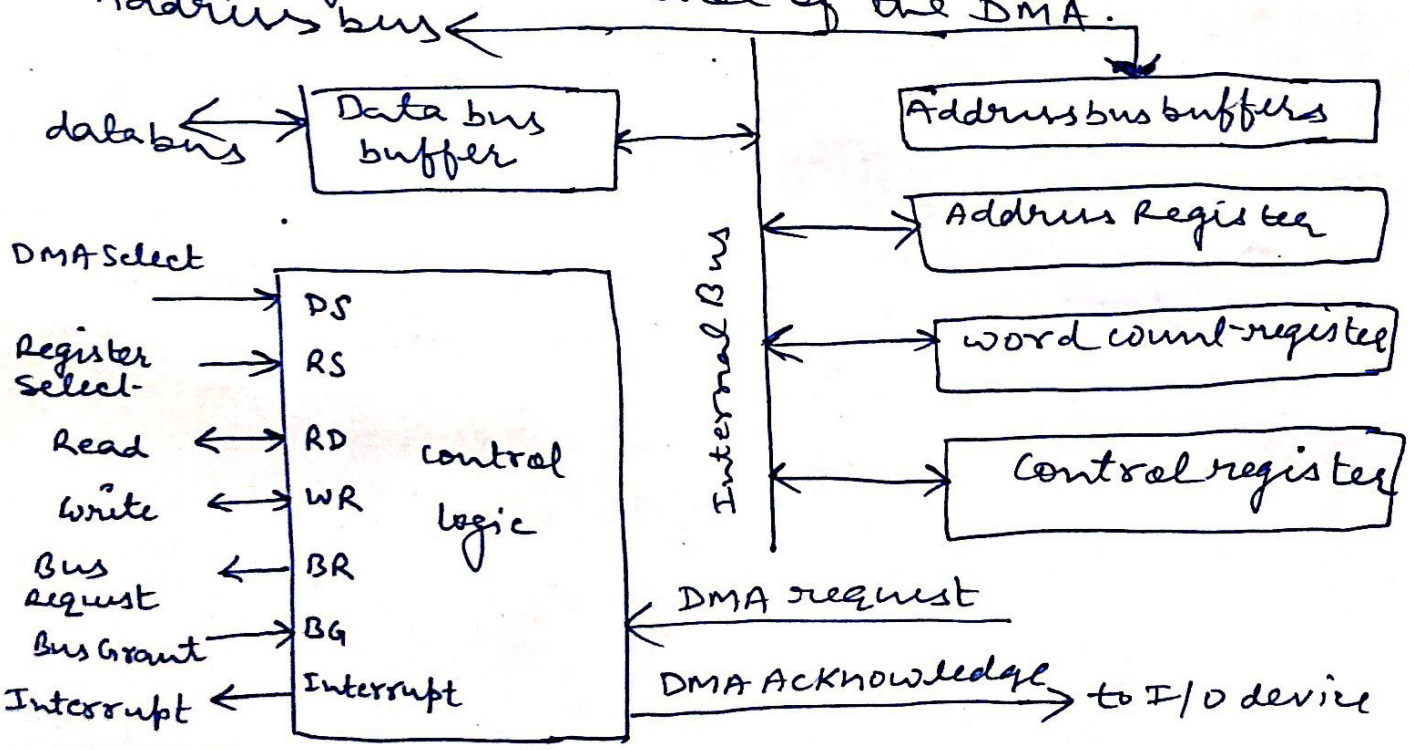


Fig: Block diagram of DMA controller

Figure shows the block diagram of a typical DMA controller. The unit communicates with the CPU via the data bus and control lines. The registers in the DMA are selected by the CPU through the address bus by enabling the DS and RS inputs. The RD and WR inputs are bidirectional. When the BG input is zero, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers. When BG = 1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control; the DMA communicates with the external peripheral through the request and acknowledge lines by using a prescribed handshaking procedure.

The DMA is first initialized by the CPU. After that, the DMA starts and continues to transfer data between memory and peripheral unit until an entire block is transferred. The initialization process is essentially a program

consisting of I/O instructions that includes the address for selecting particular DMA registers. The CPU initializes the DMA by sending the following information through the data bus:

1- The starting address of the memory block where data are available (for read) or where data are to be stored (for write)

2- The word count, which is the number of words in the 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 :

The position of the DMA controller among the other components in a computer system is illustrated

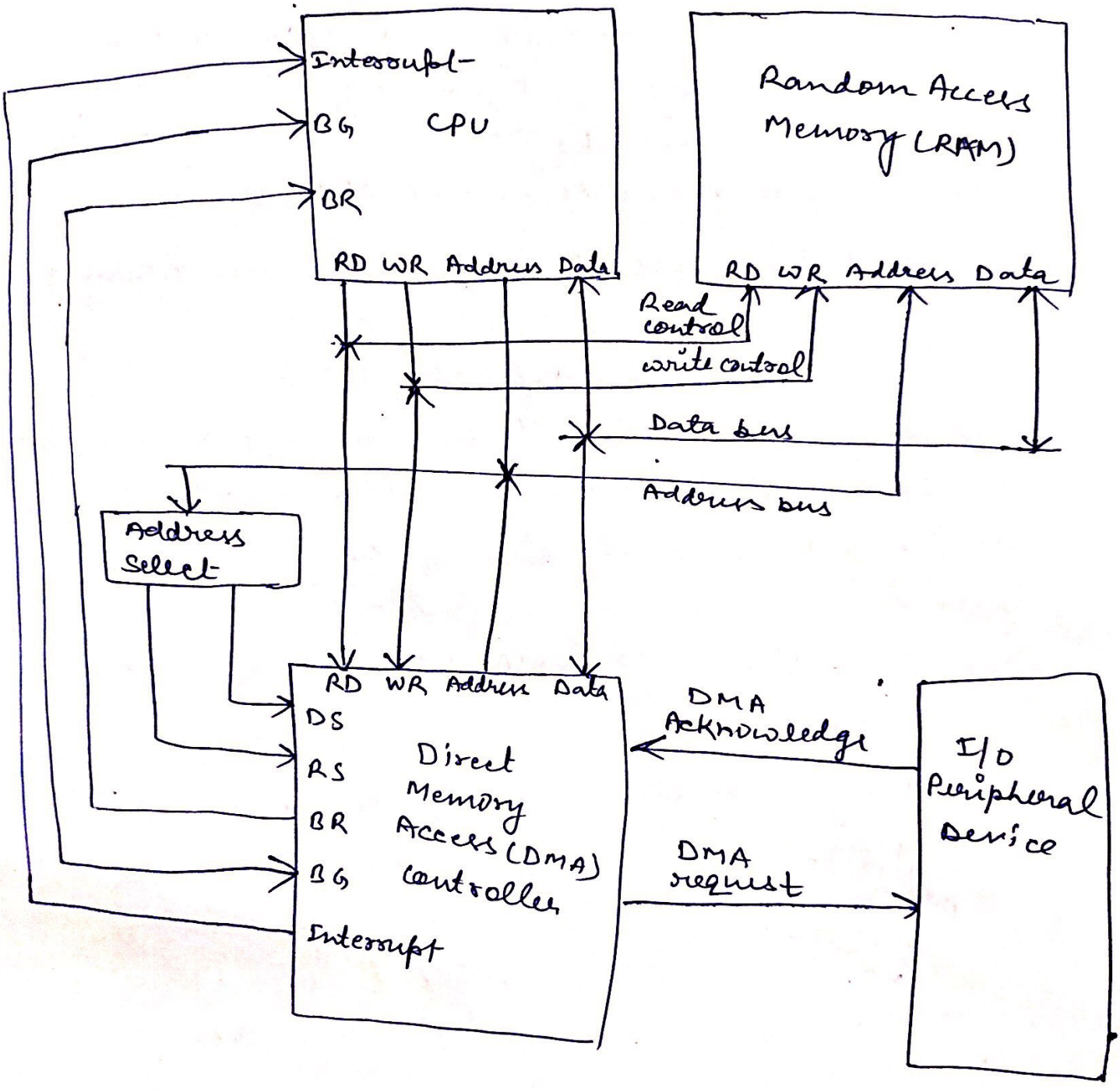


Fig: DMA transfer in a computer system

The CPU communicates with the DMA through the address and data buses as with any interface unit. The DMA has its own address, which activates the DS and RS lines. The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory.

When the 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 the address bus, initiates the RD or WR signal, and sends a DMA acknowledge to the peripheral device.

The RD and WR lines in the DMA controller are bidirectional. The direction of transfer depends on the status of the BG line. When BG=0, the RD and WR are input lines allowing the CPU to communicate with the internal DMA registers. When BG=1, the RD and WR are output lines from the DMA controller to the RAM to specify the Read or Write operation for the data.

Input - Output Processor (IOP):

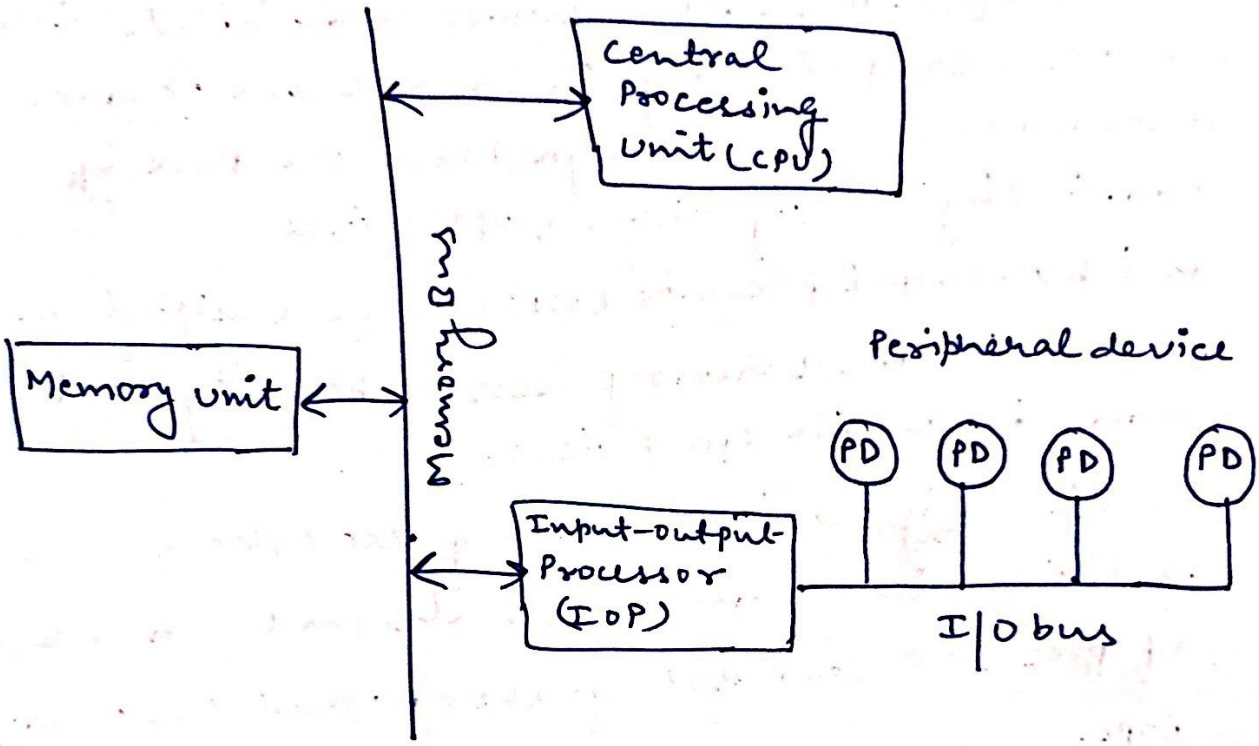
Instead of having each interface communicate with the CPU, a computer may incorporate one or more external processors and assign them the task of communicating directly with all I/O devices.

An input-output processor (IOP) may be classified as processor with direct memory access capability that communicate with I/O devices.

In this configuration, the computer system can be divided into a memory unit, and a number of processor comprised of the CPU and one or more IOPs.

The IOP similar to a CPU except that it is designed to handle the details of I/O Processing, unlike the DMA controller that must be setup entirely by the CPU; the IOP can fetch and execute its own instructions. IOP instructions are specially designed to facilitate I/O transfers. In addition the IOP can perform other processing task, such as arithmetic, logic, branching and code translation.

The block diagram of a computer with two processors is shown in figure



Block diagram of a computer with I/O Processor

The memory unit occupies a central position and can communicate with each processor by means of direct memory access. The CPU is responsible for processing data needed in the solution of computational tasks. The IOP provides a path for transfer of data between various peripheral devices and the memory unit.

CPU-IOP Communication

The communication between CPU and IOP may take different forms, depending on the particular computer considered. In most cases, the memory unit acts as a message center where each processor leaves information for the other. To appreciate the operation of a typical IOP communicate. This is simplified example that omits many operating details in order to provide an overview of basic concepts.

The sequence of operations may be carried out as shown in the flow chart of figure.

Fig! CPU-IOP communication