Chapter 9
Input and Output Organization

9.1 Peripheral Devices

→ Those devices which are connected to computer are called peripheral devices. E.g. keyboard, mouse, printer, magnetic tape, magnetic disk etc.

→ Input output operations are accomplished through peripheral device that provides a means of exchanging data between the external environment and the computer.

9.2 Input Output Interface

→ The computer system includes special hardware components between the CPU and peripheral devices to supervise and synchronize all input and output transfer. These components are called interface unit (I/O module).

→ There are several problems while we are trying to connect the peripheral devices directly with the CPU. They are:

- There is wide variety of peripherals with various methods of operation. It would be impractical to incorporate the necessary logic within the processor to control a range of devices.
- The data transfer rate of peripherals is often much slower than that of the memory and processor. Thus, it is impractical to use the high speed system bus to communicate directly with a peripheral.
- Peripheral devices often use different data formats and word lengths than the computer to which they are attached.
- Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.

→ Because of these problems (differences), we cannot connect peripheral directly with CPU. And to resolve these problems, we need I/O interface.

→ The main functions of input-output interface circuit are data conversion, synchronization and device selection. Data conversion refers to conversion between digital and analog signals, and conversion between serial and parallel data formats. Synchronization refers to matching of operating speeds of CPU and other peripherals. Device selection refers to the selection of I/O device by CPU in a queue manner

I/O bus and Interface module

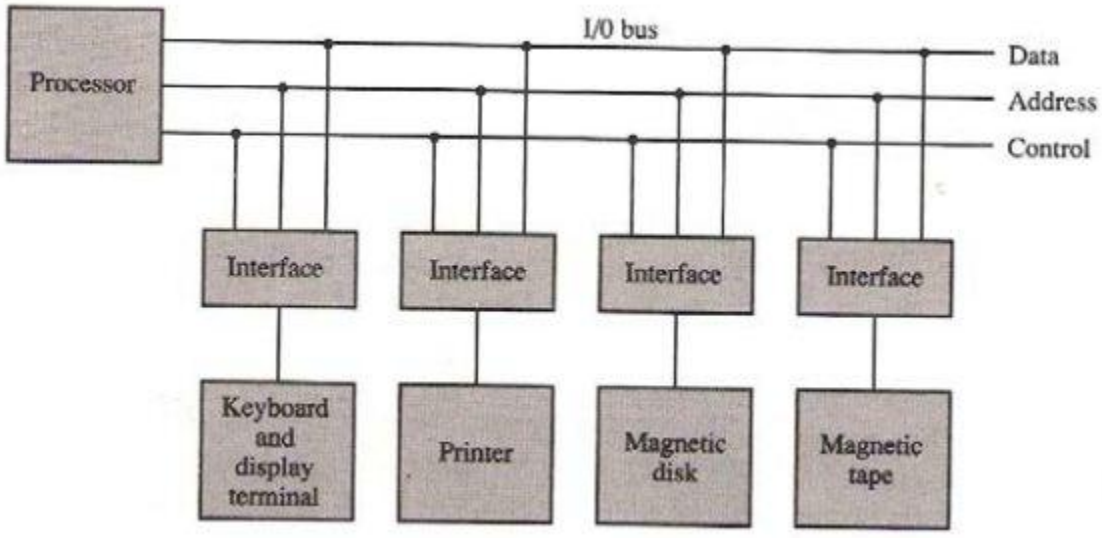


Fig: Connection of I/O bus to I/O devices

→ Above figure shows the I/O bus and I/O device connected with interface unit.

→ Each interface attached to I/O bus contains address decoder that monitors the address line.

→ To communicate with a particular device, the processor places a device address on the address lines. When the interface detects its own address, then it activates the path between the bus lines and the device that it controls. All the peripherals whose addresses don't correspond to the address in the address bus are disabled by their interface.

→ At the same time, the address is made available in the address line; the processor provides a function code in the control line which is also called I/O command. The interface selected responds to the function code and proceeds to execute it.

→ There are four types of I/O command:

i) *Control command*
The control command is issued to activate and inform the peripheral devices what they have to do.

ii) *Status command*
The processor issues status command to test the status condition of interface and peripherals.

iii) *Output data command*
It is issued to transfer data from system bus to one of storage register in I/O module.

iv) *Input data command*
It is issued to transfer data from peripheral to one of its register in I/O module.

I/O and Memory bus

→ The CPU communicates with the memory and I/O. Both I/O and memory have data, address and control buses. There are three ways that computer buses can be used to communicate with memory and I/O. they are:

i) Use two separate buses, one for memory and the other for I/O.
ii) Use one common bus for both memory and I/O but have separate control lines for each.
iii) Use one common bus for both memory and I/O with common control lines.

Isolated I/O

→ The I/O in which one common bus is used for memory and I/O but there are separate read and write controls for I/O and memory transfer is called isolated I/O.

→ This configuration isolates all I/O interface address from the address assigned to memory. Therefore, this method is called isolated I/O.

→ The I/O read and I/O write control lines are enabled during an I/O transfer. The memory read and memory write control lines are enabled during a memory transfer.

→ When the CPU doing I/O read and write operation, the address associated with information (instruction or data) is placed in common address lines. At the same time, the I/O read and I/O write line is enabled. This informs the interface that the address in the address bus is for I/O, not for memory.

→ When the CPU doing memory read and write operation, the address associated with information (instruction or data) is placed in common address lines. At the same time, the memory read and memory write line is enabled. This informs the interface that the address in the address bus is for memory, not for I/O.

Advantage

- Same address can be used for either memory and I/O transfer, only control line identifies whether the transfer is I/O or memory.

Disadvantage

- Needed separate input-output read/write and memory read/write instructions for I/O and memory transfer.

Memory mapped I/O

→ The I/O in which one common bus is used for memory and I/O bus with common control lines is called memory mapped I/O.

→ It uses common read/write instruction for memory and I/O operation.

→ It uses same address space for both memory and I/O and treats interface register as a part of memory.

→ The addresses used by interface register cannot be used for memory space. So, if CPU places the register addresses and data on common bus, the memory system ignores the operation. So, I/O operation is performed.

Advantage

- Same instructions are used for memory and I/O.
- No need of separate control lines for I/O and memory operation.

Disadvantage

- No full memory address can be used.

9.3 Asynchronous Data Transfer

→ Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.

→ One way of achieving this is by means of a ***strobe*** pulse supplied by one of the units to indicate to 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 the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as ***handshaking***.

→ Strobe technique and Handshaking technique are combined and used extensively on numerous occasions requiring the transfer of data between two independent units.

Strobe Control

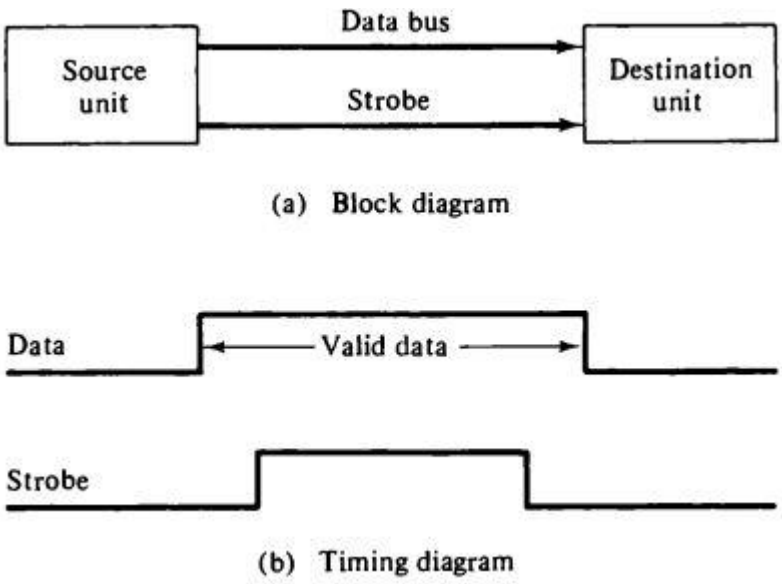


Fig: Source-initiated strobe for data transfer

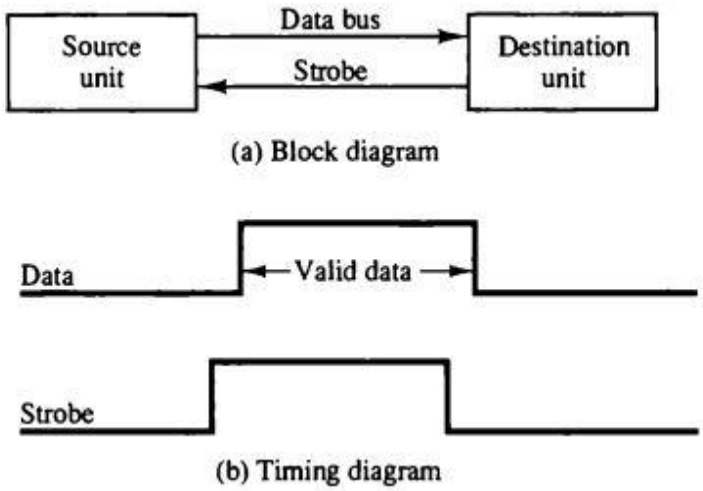
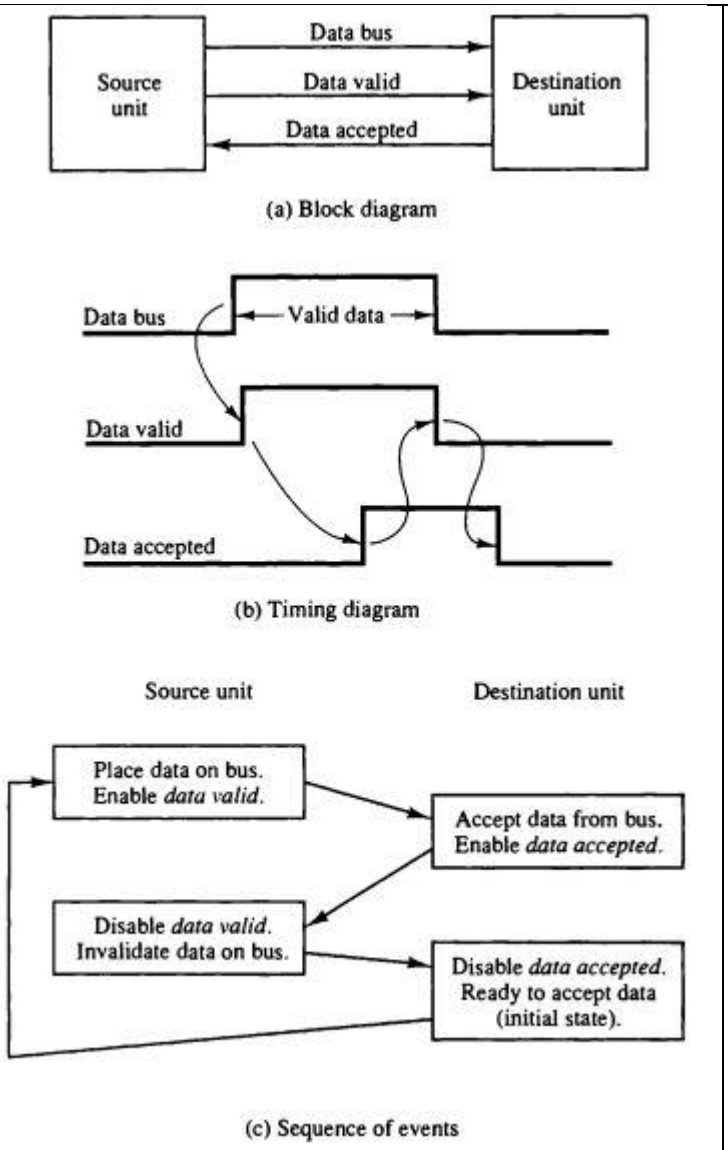
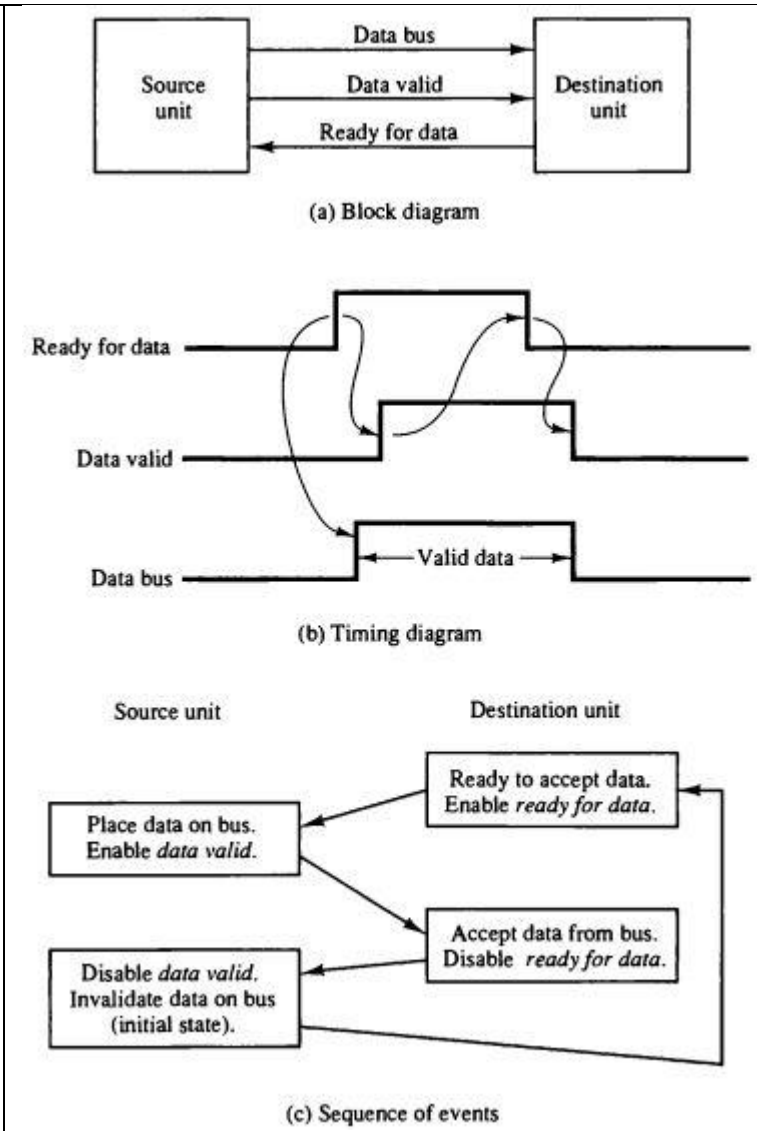


Fig: Destination-initiated strobe for data transfer

Handshaking



Source-initiated transfer using handshaking



Destination-initiated transfer using handshaking

H\W, Describe Asynchronous Serial Transfer.

9.4 Modes of transfer (techniques for I/O operations)

The possible data transfer between to and from peripherals are:

i. Programmed I/O

→ In programmed I/O, the processor executes a program that gives it direct control of the I/O operation, including sensing I/O device status, sending a read or write command, and transferring the data.

→ The execution of I/O related instructions are performed by issuing a command to the appropriate I/O module.

→ I/O module performs the requested action and set the appropriate bits in the I/O status register.

→ The processor periodically checks the status of the I/O module until it finds that the operation is complete.

→ In programmed I/O, CPU stays in programming loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it keeps the processor busy needlessly.

ii. Interrupt-initiated I/O

→ To reduce the time spent on I/O operations for periodically checking the status of I/O device, the CPU can use an interrupt-driven I/O approach.

→ In this method, CPU uses an interrupt and commands to inform the interface to issue an interrupt signal when the data are available from the device and CPU does other work.

→ When I/O module determines that the device is ready for data transfer, it interrupts the CPU. When CPU detects the external interrupt signal, it immediately stops the task it is processing, and jumps to a service routine to process the I/O transfer and then returns to the task it was originally performing.

Example of Interrupt initiated I/O:

1. Vectored interrupt
2. Non-vectored interrupt

Vectored interrupt:

- In vectored interrupt, the source, that interrupts, supplies the branch information to the computer. This information is called the interrupt vector.

Non-vectored interrupt:

- In a non vectored interrupt, the branch address is assigned to a fixed location in memory.

iii. DMA transfer

→ Drawbacks of programmed and interrupt-driven I/O:

- The I/O transfer rate is limited by the speed with which the processor can test and service the device.
- The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.

→ To overcome these drawbacks, the DMA is used to transfer the data between memory and I/O device with less involvement of CPU.

→ Large amount of data can be transferred between memory and peripheral, severely impacting CPU performance.

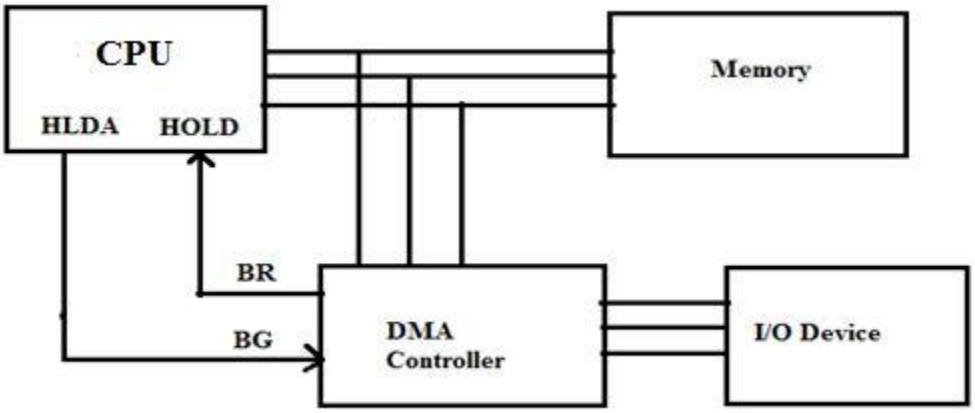


Fig: DMA Transfer

→ In this transfer mode, DMA controller sends Bus Request (BR) to CPU enabling BR line.

→ After receiving BR request, CPU acknowledges the DMA controller by sending Read/Write control signal, Device address, Starting address of memory block for data and amount of data to be transferred and disables its control over system bus by issuing Bus Grant (BG) signal to DMA controller. And CPU does other works.

→ DMA controller then start transfer of data between memory and I/O. After completing all transfer of data, the controller sends interrupt to CPU informing that it has completed the job assigned to it.

→ The CPU finalizes the DMA operation and resumes it operation.

9.5 Interrupt

When a Process is executed by the CPU and when a user Request for another Process, then this will create disturbance for the Running Process. This is also called as the **Interrupt**.

Interrupts can be generated by User, Some Error Conditions and also by Software's and the hardware's. But CPU will handle all the Interrupts very carefully because when Interrupts are generated, then the CPU must

handle all the Interrupts very carefully means the CPU will also provide Response to the various Interrupts those are generated so that when an interrupt has occurred, then the CPU will handle by using the Fetch, Decode and Execute Operations.

Types of Interrupts

Generally there are three types o Interrupts those are occurred. For Example

a) Internal Interrupt
b) Software Interrupt.
c) External Interrupt.

The Internal Interrupts are those which are occurred due to some problem in the execution. For example, when a user performing any operation which contains any type of error so that internal interrupts are those which are occurred by some operations or by some instructions and the operations those are not possible but a user is trying for that operation.

The software interrupts are those which are made some call to the system. For example, while we are processing some instructions and when we want to execute one more application programs.

The External Interrupt occurs when any input and output devices request for any operation and the CPU will execute those instructions first. For example, when a program is executed and when we move the mouse on the screen, then the CPU will handle this external interrupt first and after that he will resume with his operation.

Priority Interrupt

A priority interrupt is a system that determines which condition is to be serviced first when two or more requests arrive simultaneously. Highest priority interrupts are serviced first. Devices with high speed transfers are given high priority and slow devices such as keyboards receive low priority. When two devices interrupt the computer at the same time the computer services the device, with the higher priority first. Establishing the priority of simultaneous interrupts can be done by software or hardware.

The hardware priority function can be established by either a serial or a parallel connection of interrupt lines.

Daisy-Chaining Priority

The serial connection is called daisy chaining method. In daisy chaining method, all the devices are connected in serial. The device with the highest priority is placed in the first position, followed by lower priority devices.

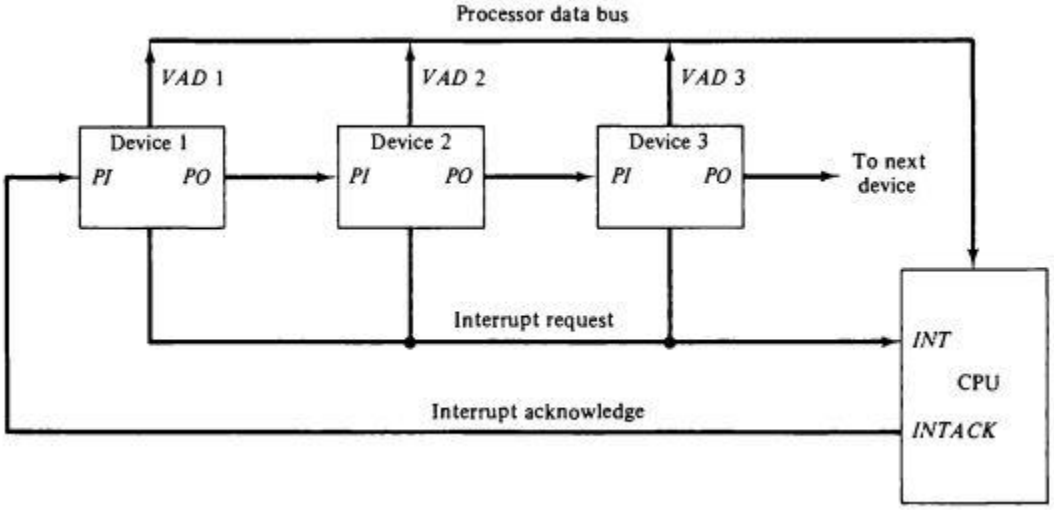


Fig: Daisy-Chain Priority Interrupt

Parallel Priority Interrupt

The parallel priority interrupt hardware is shown in figure. It has an *interrupt register* whose bits are connected to the interrupt request lines of different devices in the system. It also has a *mask register* whose bits can be

used to control the status of each interrupt request. The mask register has the same number of bits as the interrupt register. Each interrupt bit and its corresponding mask bit are applied to an AND gate.

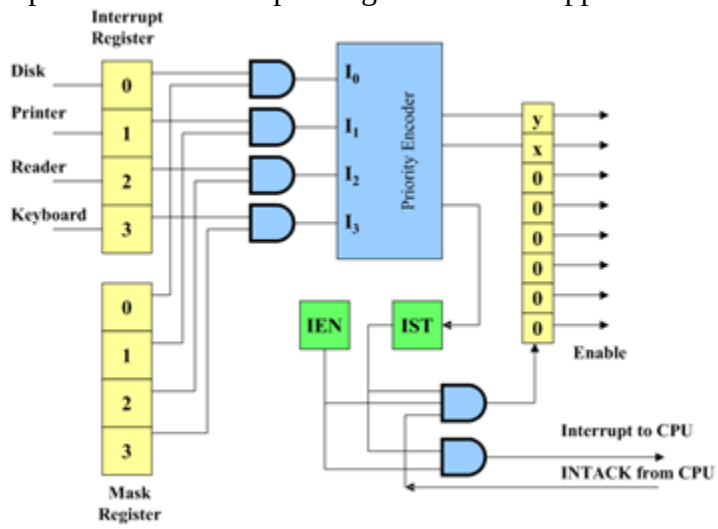


Fig: Parallel Priority Interrupt Hardware

This produces the four inputs to a priority encoder. The priority encoder generates two bits of the vector address. This is transferred to the CPU. Another output from the encoder sets an interrupt status flip flop IST. The outputs of interrupt enable flip-flop IEN and IST are applied to an AND gate. The outputs of this AND gate provide a common interrupt signal for the CPU. The interrupt acknowledge INTACK signal from the CPU enables the bus buffers in the output register and a vector address VAD is placed into the data bus.

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 below.

Inputs				Outputs			
I_0	I_1	I_2	I_3	x	y	IST	Boolean functions
1	×	×	×	0	0	1	
0	1	×	×	0	1	1	$x = I_0'I_1'$
0	0	1	×	1	0	1	$y = I_0'I_1 + I_0'I_2'$
0	0	0	1	1	1	1	$(IST) = I_0 + I_1 + I_2 + I_3$
0	0	0	0	×	×	0	

Fig: Priority Encoder Truth Table

The X's in the table designate don't care conditions. Input I_0 has the highest priority. When I_0 input is 1, the output generates an output xy=00. I_1 has the next priority level. The output is 01 if I_1=1 and I_0=0. The output for I_2 is generated only if higher priority inputs are 0 and so on. The interrupt status IST is set only when one or more inputs are equal to 1. If all inputs are 0, IST is cleared to 0 and the other outputs of the encoder are not used, so they are marked with don't care conditions.

9.6 Direct Memory Access (DMA)

DMA is a sophisticated I/O technique in which a DMA controller replaces the CPU and takes care of the access of both, the I/O device and memory, for fast data transfers. Using DMA you get the fastest data transfer rates possible.

Momentum behind the DMA: interrupt-driven and Programmed I/O require active CPU intervention (All data must pass through CPU).transfer rate is limited by processor's ability to service the device and hence CPU is

tied up managing I/O transfer. Removing CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer.

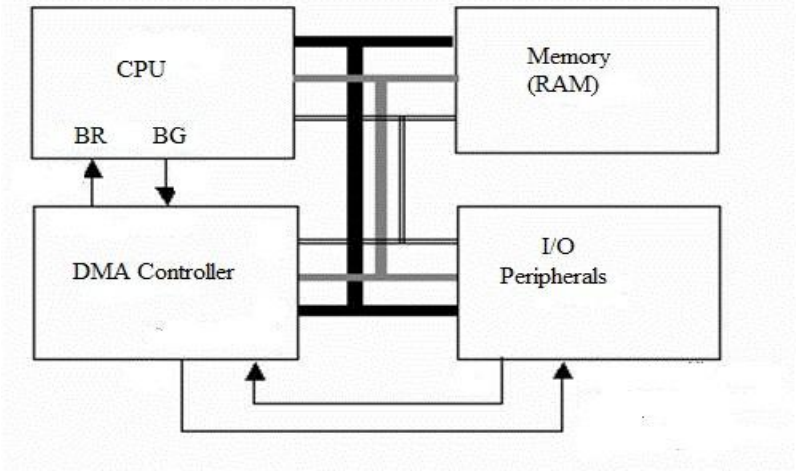


Fig: DMA

Extensively used method to capture buses is through special control signals:

- Bus Request (BR): used by DMA controller to request the CPU for buses. When this input is active, CPU terminates the execution of the current instruction and places the address bus; data bus and read & write lines into high impedance state.
- Bus Grant (BG): CPU activates BG output to inform DMA that buses are available (in high impedance state). DMA now take control over buses to conduct memory transfers without processor intervention. When DMA terminates the transfer, it disables the BR line and CPU disables BG and returns to normal operation.

When DMA takes control of bus system, the transfer with memory can be made in the following ways:

a) Burst transfer mode
   - Entire block of data is transferred to one contiguous sequence.
   - Useful for loading programs or data file into memory, but it makes CPU inactive for long periods of time.

b) Cycle stealing mode
   - DMA controller transfers one byte of data then returns control of system buses to CPU.
   - CPU performs an instruction, and then DMA controller transfers a data value by stealing one machine cycle of CPU.

9.7 Input-Output Processor (IOP)

Instead of having each interface communicate with the CPU, computer may have one or more external processors for the task of communicating directly with all I/O devices. Such processor is called Input-Output processor. I/O processors also have direct memory access capability.

→ IOP is a special processor dedicated to communicate directly with all I/O devices.

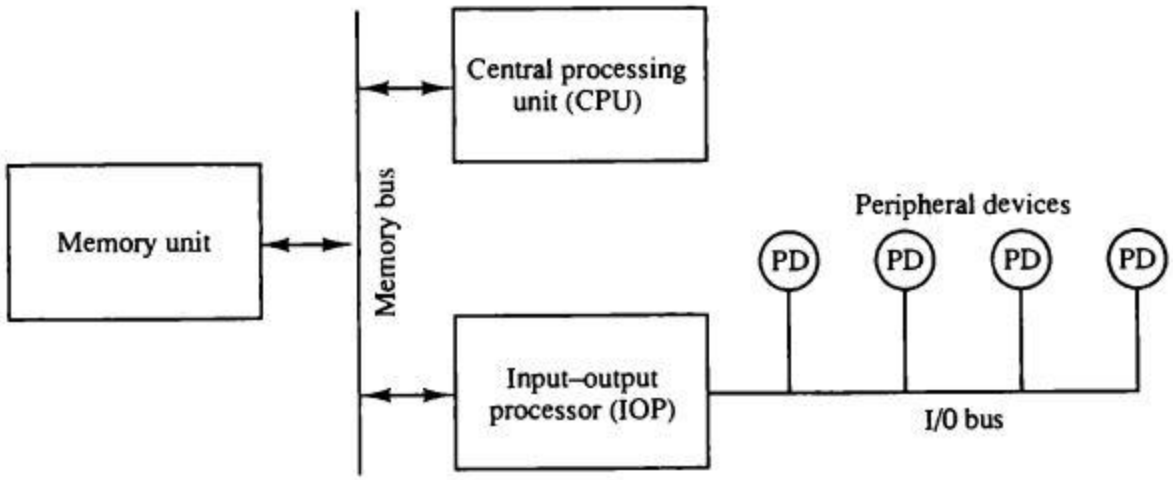


Fig: Block Diagram of computer with I/O Processor

→ Each IOP takes care of I/O task keeping CPU free from involvement of I/O transfer.

→ IOP has ability to execute I/O instruction which gives it complete control over I/O operation. It has its own instruction set with I/O instructions and a local memory in its own right.

→ IOP accesses memory by cycle stealing.

→ CPU directs IOP to execute an I/O programs in memory. The IOP fetches and executes these instructions without CPU intervention. IOP interrupts CPU when entire operation has been performed.

→ The major difference between DMA and IOP is that IOP can fetch and execute I/O instruction from memory but DMA cannot fetch and execute the I/O instruction.