

and the sender operates at different moment than the receiver then such a data transfer is called Asynchronous data transfer.

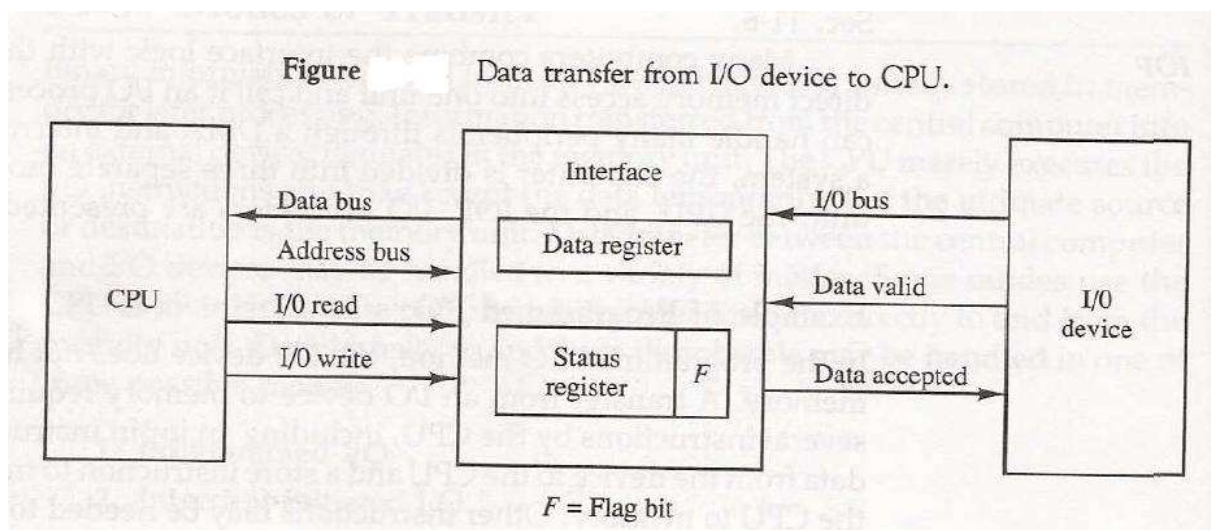
The data transfer can be handled by various modes. some of the modes use CPU as an intermediate path, others transfer the data directly to and from the memory unit and this can be handled by 3 following ways:

- i. Programmed I/O
- ii. Interrupt-Initiated I/O
- iii. Direct Memory Access (DMA)

### **Programmed I/O Mode:**

In this mode of data transfer the operations are the results in I/O instructions which is a part of computer program. Each data transfer is initiated by a instruction in the program. Normally the transfer is from a CPU register to peripheral device or vice-versa.

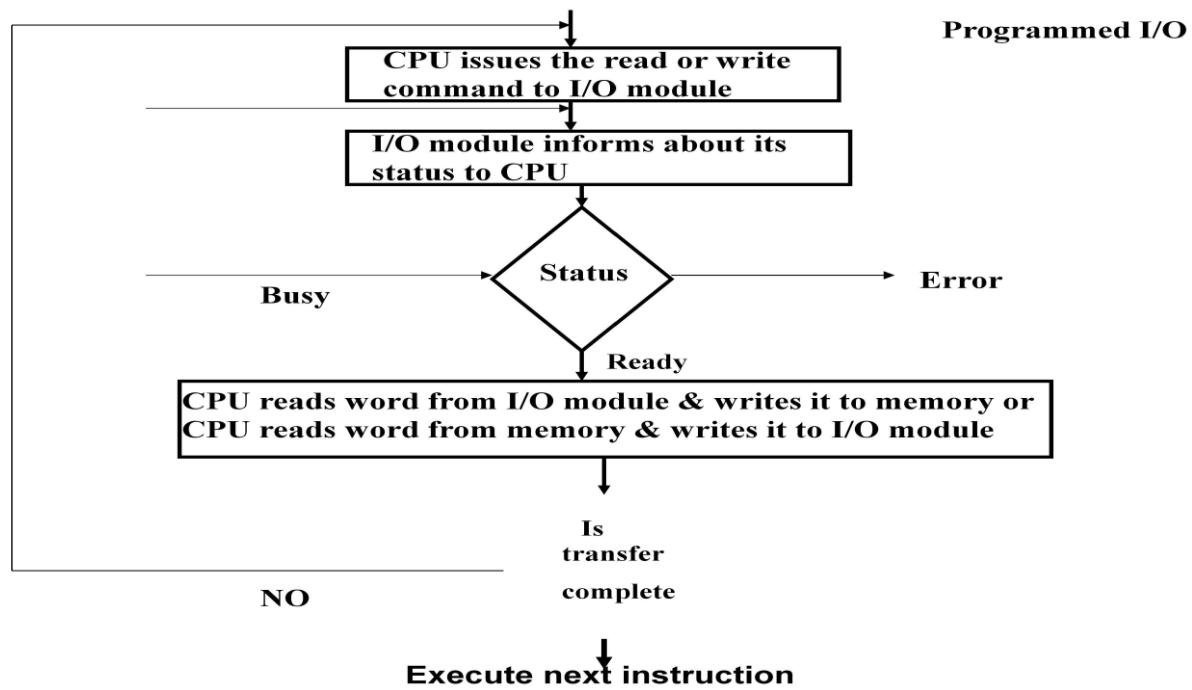
Once the data is initiated the CPU starts monitoring the interface to see when next transfer can made. The instructions of the program keep close tabs on everything that takes place in the interface unit and the I/O devices.



- The transfer of data requires three instructions:

1. Read the status register.
2. Check the status of the flag bit and branch to step 1 if not set or to step 3 if set.
3. Read the data register.

In this technique CPU is responsible for executing data from the memory for output and storing data in memory for executing of Programmed I/O as shown in Flowchart-:



Drawback of the Programmed I/O :

The main drawback of the Program Initiated I/O was that the CPU has to monitor the units all the times when the program is executing. Thus the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process and the CPU time is wasted a lot in keeping an eye to the executing of program.

To remove this problem an Interrupt facility and special commands are used.

### **Interrupt-Initiated I/O :**

In this method an interrupt facility an interrupt command is used to inform the device about the start and end of transfer. In the meantime the CPU executes other program. When the interface determines that the device is ready for data transfer it generates an Interrupt Request and sends it to the computer.

When the CPU receives such a signal, it temporarily stops the execution of the program and branches to a service program to process the I/O transfer and after completing it returns back to task, what it was originally performing.

- In this type of IO, computer does not check the flag. It continue to perform its task.

- Whenever any device wants the attention, it sends the interrupt signal to the CPU.
- CPU then deviates from what it was doing, store the return address from PC and branch to the address of the subroutine.
- There are two ways of choosing the branch address:
  - Vectored Interrupt
  - Non-vectored Interrupt
- In vectored interrupt the source that interrupt the CPU provides the branch information. This information is called interrupt vectored.
- In non-vectored interrupt, the branch address is assigned to the fixed address in the memory.

#### **Priority Interrupt:**

- There are number of IO devices attached to the computer.
- They are all capable of generating the interrupt.
- When the interrupt is generated from more than one device, priority interrupt system is used to determine which device is to be serviced first.
- Devices with high speed transfer are given higher priority and slow devices are given lower priority.
- Establishing the priority can be done in two ways:
  - Using Software
  - Using Hardware
- A pooling procedure is used to identify highest priority in software means.

#### **Polling Procedure :**

- There is one common branch address for all interrupts.
- Branch address contain the code that polls the interrupt sources in sequence. The highest priority is tested first.
- The particular service routine of the highest priority device is served.
- The disadvantage is that time required to poll them can exceed the time to serve them in large number of IO devices.

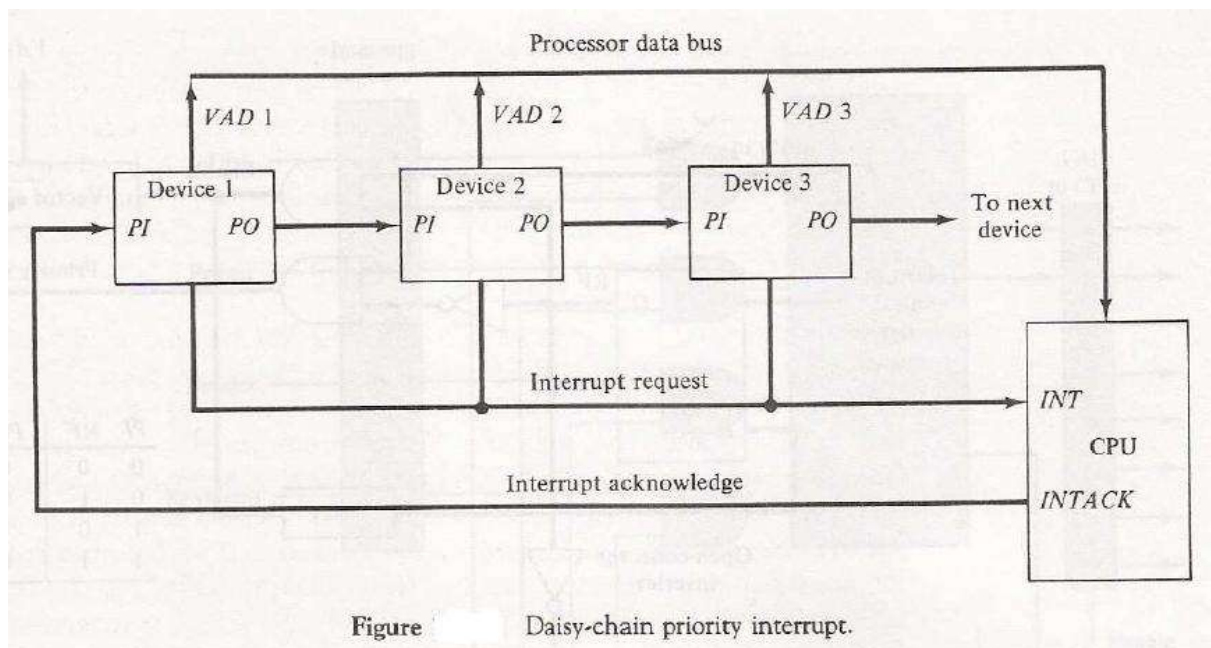
#### **Using Hardware:**

- Hardware priority system function as an overall manager.

- It accepts interrupt request and determine the priorities.
- To speed up the operation each interrupting devices has its own interrupt vector.
- No polling is required, all decision are established by hardware priority interrupt unit.
- It can be established by serial or parallel connection of interrupt lines.

#### **Serial or Daisy Chaining Priority:**

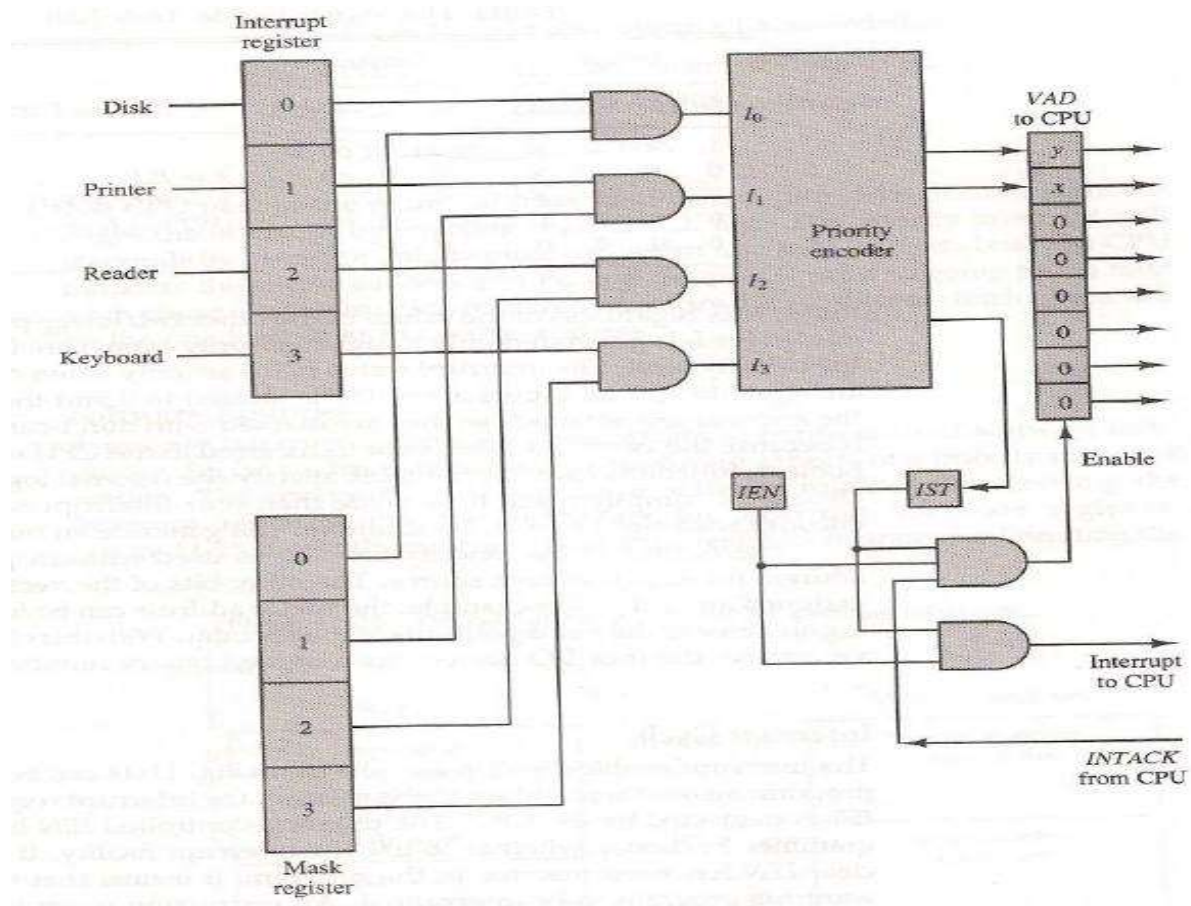
- Device with highest priority is placed first.
- Device that wants the attention send the interrupt request to the CPU.
- CPU then sends the INTACK signal which is applied to PI(priority in) of the first device.
- If it had requested the attention, it place its VAD(vector address) on the bus. And it block the signal by placing 0 in PO(priority out)
- If not it pass the signal to next device through PO(priority out) by placing 1.
- This process is continued until appropriate device is found.
- The device whose PI is 1 and PO is 0 is the device that send the interrupt request.



#### **Parallel Priority Interrupt :**

- It consist of interrupt register whose bits are set separately by the interrupting devices.
- Priority is established according to the position of the bits in the register.

- Mask register is used to provide facility for the higher priority devices to interrupt when lower priority device is being serviced or disable all lower priority devices when higher is being serviced.
- Corresponding interrupt bit and mask bit are ANDed and applied to priority encoder.
- Priority encoder generates two bits of vector address.
- Another output from it sets IST(interrupt status flip flop).



Priority Encoder Truth Table

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

The Execution process of Interrupt-Initiated I/O is represented in the flowchart: