contain valid data. The control register directs the movement of data through the registers. Whenever the fi bit of the control register is set (fi=1) and the f_{i+1} bit is reset ($f_{i+1}'=1$), a clock is generated causing register R(I+1) to accept the data from register RI. Data are inserted into the buffer provided that the input ready signal is enabled. This occurs when the first control flipflop f_1 is reset, indicating that register R1 is empty. Data are loaded from the input lines by enabling the clock in R1 through the insert control line.

The data falling through the register stack up at the o/p end. The output ready control line is enabled when the last control flip flop f_4 is set, indicating that there are valid data in the o/p register R4. The o/p data from R4 are accepted by a destination unit, which then enables the delete control signal. This reset f_4, causing o/p ready to disable, indicating the data on the o/p are no longer valid. Only after the delete signal goes back to 0 can the data from R3 move into R4. If FIFO is empty, there will be no data in R_3 and f_4 will remain in reset state.

MODES OF TRANSFER:

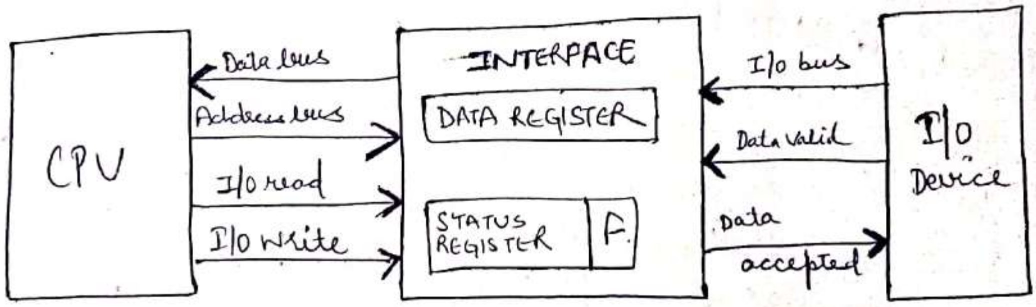
Binary information received from an external device is usually stored in memory. Information transferred from the central computer into an external device originates in the memory unit. The CPU merely executes the I/O instructions, but ultimate source or destination is the memory unit. Data transfer b/w the central computer and I/O device may be handled in variety of modes. Some modes use the CPU as an intermediate path; other transfers the data directly to and from the 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)

1. PROGRAMMED I/O :- 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 & from CPU and memory. Transfering data under program control requires constant monitoring of the peripheral by the CPU. Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.

Example :-

In programmed I/O method, the I/O device doesnot have direct access to memory. A transfere from an I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from the device to the CPU and a store instruction to transfer the data from the CPU to memory. An example of data transfer from an I/O device through an interface into the CPU is shown

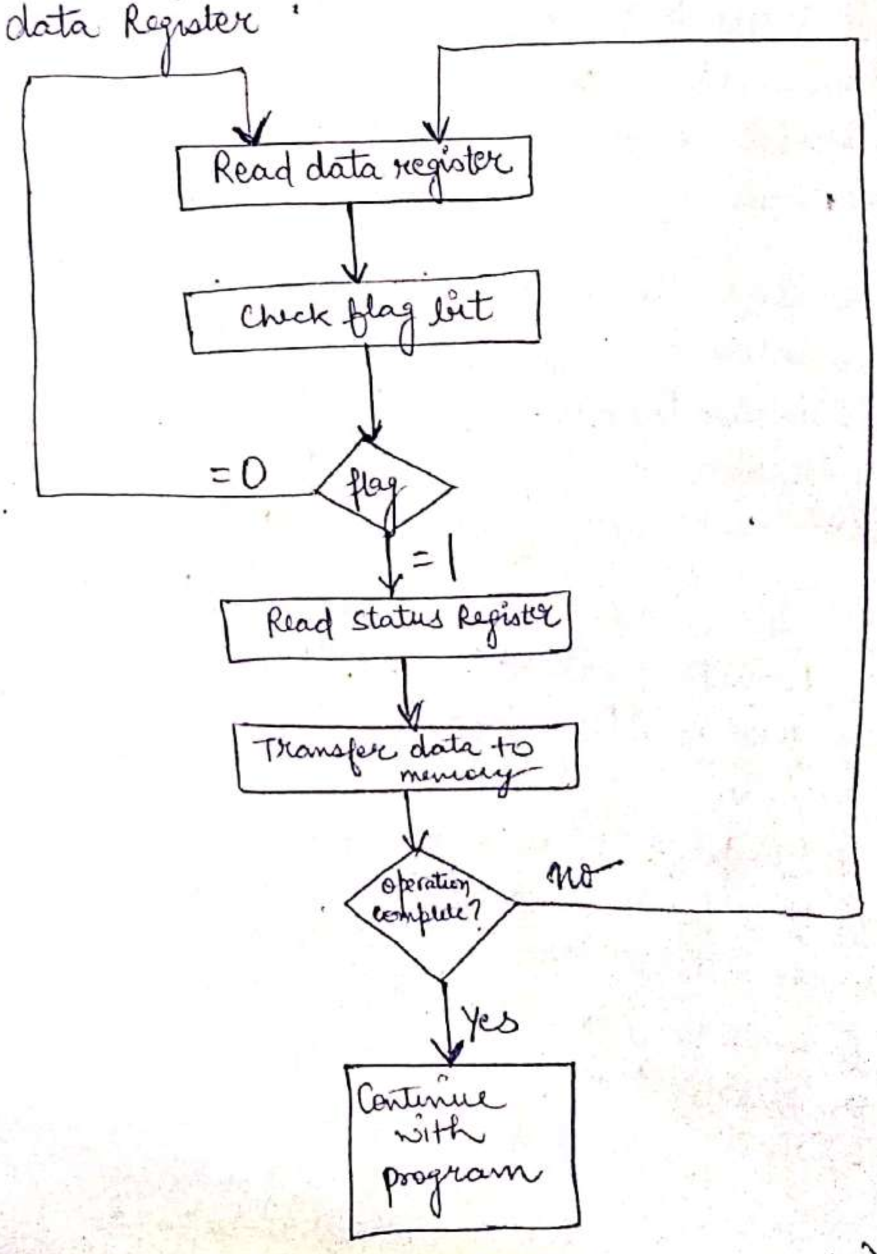


The device transfer bytes of data one at a time as they are available. When byte of data is available, the device places it in the I/O bus and enables its data valid line. The interface accepts the byte into its data register and enables the data accepted line. The interface sets a bit in the status register that we will refer to as an F or "flag" bit. The device can now disable the data valid line but it will not transfere another byte until the data accepted line is disabled by the interface. This is according to the handshaking.

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

Transfer of each byte requires three instructions:

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



(flowchart for CPU program to input data)

The programmed I/O method is useful in small low-speed computers or in systems that are dedicated to monitor a device continuously. The difference in information transfer rate b/w the CPU and I/O device makes this type of transfer inefficient.

2. INTERRUPT INITIATED I/O :-

In programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is time consuming process. It can be avoided by using an interrupt request signal when the data are available from the device. In meantime CPU can proceed to execute another program. The interface meanwhile keeps monitoring the device. When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer upon detecting the external interrupt signal, the CPU momentarily stop the tasks it is processing, branches to a service program to process the I/O transfer and then returns to the task it was originally performing

Eg:- 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 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. The CPU deviates from what it is doing to take care of the I/P or O/P transfer. After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt

The CPU responds to interrupt signal by storing the return address from the program counter into a memory stack & then control branches to a service routine. The way that the processor chooses the branch address of the service routine varies from one unit to another. There are 2 methods for accomplishing this

- Vectored interrupt
- Non vectored interrupt

In nonvectored interrupt, the branch address is assigned to a fixed location in memory.
In vectored interrupt, the source that interrupts supplies the branch information to the computer. This information is called the interrupt vector.

3. DMA (DIRECT MEMORY ACCESS) :— In direct memory access, the interface transfers data into and out of the memory unit through the memory bus. The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred & then proceeds to execute other tasks. When the transfer is made, the DMA requests memory cycles through the memory bus. When the request is granted by the memory controller, the DMA transfers the data directly into memory. The CPU merely delays its memory access operation to allow the direct memory I/O transfer. Since peripheral speed is usually slower than processor speed, I/O - memory transfers compared to processor access to memory.