UNIT – IV

INPUT OUTPUT ORGANIZATION

I/O plays a crucial role in any modern computer system. Therefore, a clear understanding and appreciation of the fundamentals of I/O operations, devices, and interfaces are of great importance. A computer serves no useful purpose without the ability to receive information from the outside world and transmit the result in a meaningful form. I/O organization of a computer is a function of size of the computer and the devices connected to it. In other words, amount of hardware computer possesses to communicate with number of peripheral units, differentiate between small and large system.

I/O subsystem

The input-output subsystem of a computer, referred as I/O, provides an efficient mode of communication between the central system and outside environment. Data and programs must be entered into the computer memory for processing and result of computations must be must be recorded or displayed for the user.

Peripheral Devices

Input or output devices attached to the computer are called peripherals. Keyboards ,display units and printers are most common peripheral devices. Magnetic disks, tapes are also peripherals which provide auxiliary storage for the system. Some input devices are keyboard, mouse, touch screen, light pen, card reader, etc. and output devices are CRT, printer(impact, ink jet, dot matrix, laser), digital incremental plotter. I/O devices communicating with people and computer usually transfer alphanumeric information using ASCII binary encoding.

Input – Output Interface

Input-output interface provides a method for transferring information between internal storage and external I/O devices.

It resolves the differences between the computer and peripheral devices.The major differences are:

- Peripherals are electromechanical and electromagnetic devices and manner of operation is different from that of CPU which is electronic component.
- Data transfer rate of peripherals is slower than that of CPU. So some synchronization mechanism may be needed.

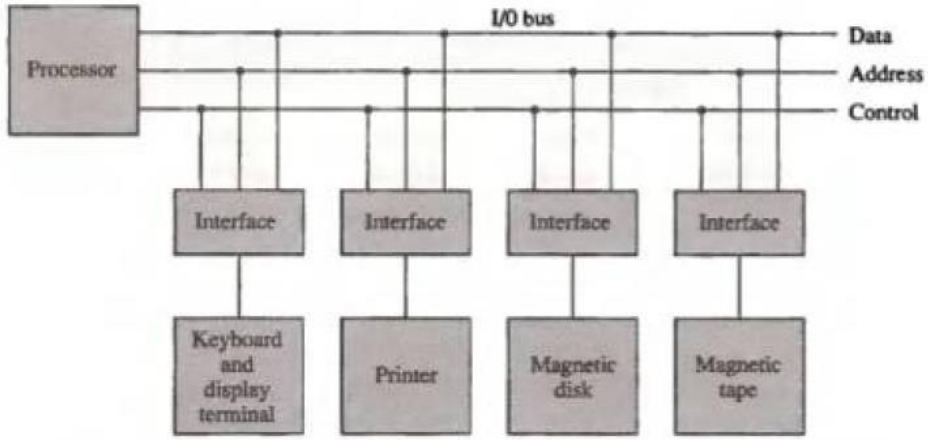
- Data codes and formats in peripherals differ from the word format in CPU and memory.
- Operating modes of peripherals are different from each other and each must be controlled so as not to disturb other.

To resolve these differences, computer system usually includes special hardware unit between CPU and peripherals to supervise and synchronize I/O transfers , which are called Interface units since they interface processor bus and peripherals.

I/O Bus and Interface Modules

Peripherals connected to a computer need special communication link to interface with CPU. This special link is called I/O bus.

Figure 11-1 Connection of I/O bus to input-output devices.



In the above figure, I/O bus from the processor is attached to all peripheral interfaces.

I/O bus consists of data lines , address and control lines.

To communicate with a particular device, the processor places a device address on the address lines. Each peripheral has an interface modules associated with its interface.

The function of an interface are listed below:

- Decodes the device address (device code).
- Decodes the I/O commands (operation or function code) in control lines.
- Provides signals for the peripheral controller.

- Synchronizes the data flow.
- Supervises the transfer rate between peripheral and CPU or Memory.

I/O Command

The function code provided by processor in control line is called I/O command. The interpretation of command depends on the peripheral that the processor is addressing. There are 4 types of commands that an interface may receive:

a) Control command:

This command is issued to activate the peripheral and to inform it what to do? E.g.a magnetic tape unit may be instructed to backspace tape by one record.

b)Status command:

This command is used to check the various status conditions of the interface before a transfer is initiated.

c)Data Input command:

This command causes the interface to read the data from the peripheral and places it into the interface buffer.

d)Data Output command:

This command causes the interface to read the data from the bus and saves it into the interface buffer.

I/O Bus vs Memory Bus

In addition to communicating with I/O, the processor also has to work with memory unit .Like I/O bus, memory bus contains data, address and read/write control lines. 3 physical organizations, the computer buses can be used to communicate with memory and I/O:

- Use two separate buses, one for memory and other for I/O: Computer has independent sets of data, address and control buses, one for accessing memory and other for I/O.usually employed in a computer that has separate IOP (Input Output Processor).
- Use one common bus for both memory and I/O having separate control lines.
- Use one common bus for memory and I/O with common control lines.

Isolated I/O vs Memory-Mapped I/O

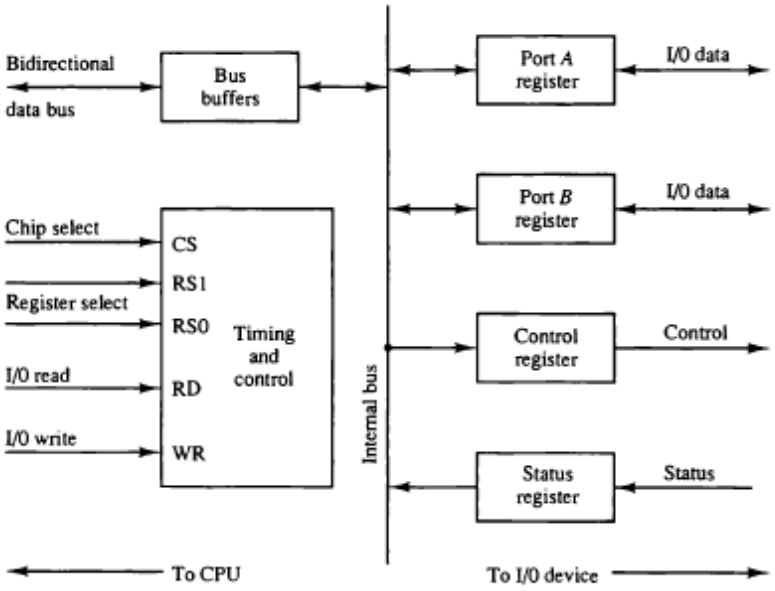
Many computers user one common bus to transfer information between memory or I/O and CPU. The distinction between memory transfer and I/O transfer is made through separate read and write lines. The CPU specifies whether the address on the address line is for a memory word or for a interface register by enabling one of two possible read or write lines. The I/O read and I/O write command lines are enabled during I/O transfer whereas memory read and memory write command lines are enabled during memory transfer.

- **ISOLATED I/O CONFIGURATION**
  - In the isolated configuration, the CPU has separate input and output instruction ,and each of this instruction is associated with the address of interface register.
  - It has two functionally and physically separate bus: memory bus and I/O bus.
  - Each bus consists of same three main groupings of wire : Address, data and control.CPU can execute instructions to manipulate the I/O bus separate from the memory bus.
  - Now, it is only used in very high-performance systems.
  - Each device interface has a port which consists of minimum of control register, status register, data-in register(read buffer), data-out register(write buffer).
- **MEMORY-MAPPED I/O CONFIGURATION**
  - I/O devices and memory location share a single address space but cannot have the same address
  - I/O looks just like memory read/write treat Status and Data registers of I/O module as memory location
  - No special commands for I/O Same machine instructions to access memory and I/O devices
  - However, part of the address space is taken by I/O devices, reducing the amount of main memory that’s accessible.

sn	Isolated I/O	Memory-Mapped I/O
1	**i/o transfer is made through separate read and write line.**	**Memory transfer is made through separate read and write lines.**
2	**The I/O transfer read and write control lines are enabled during the i/o are enabled during a memory transfer.**	**The memory read and memory write control lines are enabled during a memory transfer.**
3	**The isolated i/o configuration the CPU has distinct input and output instructions and each instruction is associated with the address of an interface register.**	**Computer with memory mapped i/o can use memory type instructions to access i/o data.**
4	**The isolated i/o method isolates memory and i/o address values are not affected by interface address assignment.**	**In a memory mapped i/o organization theme are no specific input or output instructions.**

I/O Interface Unit

- I/O interface unit is shown in the block diagram below, it consists:
  - Two data registers called ports.
  - A control register
  - A status register
  - A bus buffer
  - Timing and control circuits
- Interface communicates with CPU through data bus
- Chip select (CS) and Register select (RS) inputs determine the address
- I/O read and write are two control lines that specify input and output.
- 4 registers directly communicate with the I/O device attached to the interface.



CS	RS1	RS0	Register selected
0	×	×	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

Figure 11-2 Example of I/O interface unit.

In this example, address bus selects the interface unit through CS and RS 1 & RS0. and a particular interface is selected by the circuit (decoder) enabling CS. RS1 and RS0 select one of 4 registers.

.SYNCHRONOUS & ASYNCHRONOUS DATA TRANSFER

SYNCHRONOUS DATA TRANSFER

* In a digital system, the internal operations are synchronized by means of clock pulses supplied by a common pulse generator.

* In a computer, CPU and an I/O interface are designed independently of each other

. * If the registers in the interface share a common clock with the CPU registers, the data transfer between two units are said to be synchronous. 2

ASYNCHRONOUS DATA TRANSFER

* In a computer system, CPU and an I/O interface are designed independently of each other.

* When internal timing in each unit is independent from the other and when registers in interface and registers of CPU uses its own private clock.

* In that case the two units are said to be asynchronous to each other. CPU and I/O device must coordinate for data transfers. 3

METHODS USED IN ASYNCHRONOUS DATA TRANSFER

* **Strobe Control**: This is one way of transfer i.e. by means of strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.

* **Handshaking:** This method is used 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.

STROBE CONTROL

* Strobe control method of data transfer uses a single control signal for each transfer. The strobe may be activated by either the source unit or the destination unit.

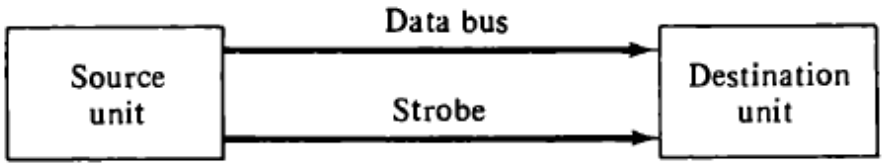
 Source Initiated Strobe

 Destination Initiated Strobe

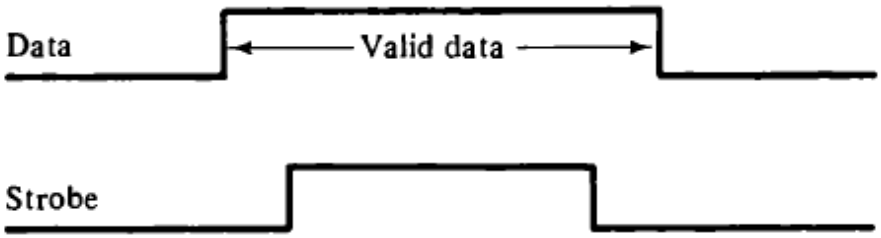
SOURCE INITIATED STROBE

* The data bus carries the binary information from source unit to the destination unit as shown below.

* The strobe is a single line that informs the destination unit when a valid data word is available in the bus.



(a) Block diagram



(b) Timing diagram

Figure 11-3 Source-initiated strobe for data transfer.

* The source unit first places the data on the bus.

* After a brief delay to ensure that the data settle to a steady value, the source activities the strobe pulse.

* The information of the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data.

* The source removes the data from the bus for a 7 brief period of time after it disables its strobe pulse.

DESTINATION INITIATED STROBE

* First, 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 unit to accept it.

* The data must be valid and remain in the bus long enough for the destination unit to accept it.

* The falling edge of the strobe pulse can be used again to trigger a destination register.

* The destination unit then disables the strobe.

* The source removes the data from the bus after a predetermined time interval.

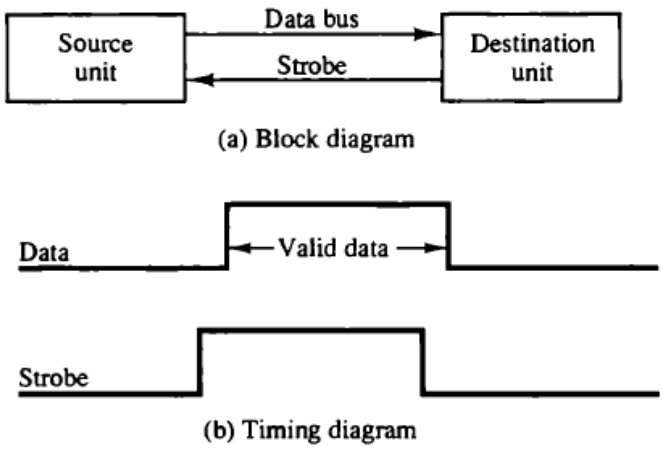


Figure 11-4 Destination-initiated strobe for data transfer.

HANDSHAKING

* In case of source initiated data transfer under strobe control method, the source unit has no way of knowing whether destination unit has received the data or not.

* Similarly, destination initiated transfer has no method of knowing whether the source unit has placed the data on the data bus.

* Handshaking mechanism solves this problem by introducing a second control signal that provides a reply to the unit that initiate the transfer.

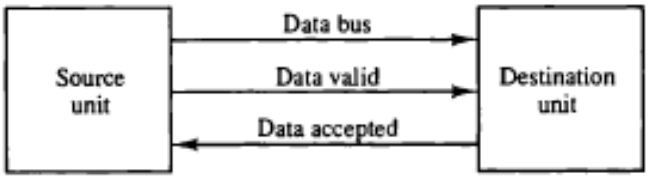
* There are two control lines in handshaking technique:

SOURCE INITIATED TRANSFER USING HANDSHAKING

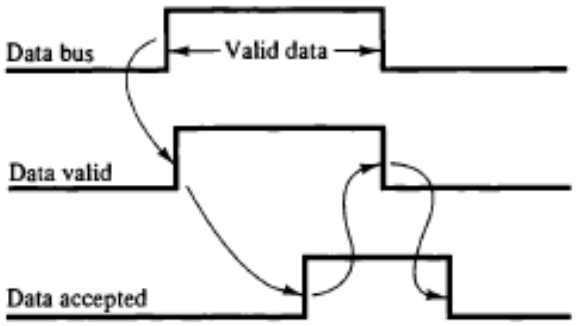
* Handshaking signals are used to synchronize the bus activities.

* The two handshaking lines are data valid, which is generated by the source unit, and data accepted, generated by the destination unit.

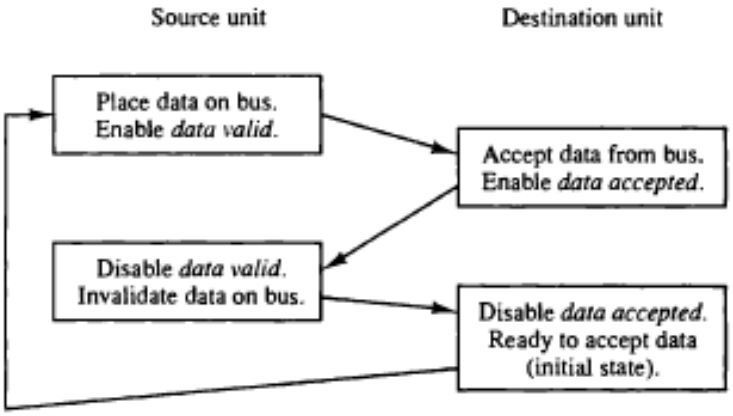
* The timing diagram shows exchange of signals between two units.



(a) Block diagram



(b) Timing diagram



(c) Sequence of events

Figure 11-5 Source-initiated transfer using handshaking.

The sequence of events:

* The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal.

* The data accepted signals is activated by the destination unit after it accepts the data from the bus.

*The source unit then disables its data valid signal, which invalidates the data on the bus.

* The destination unit the disables its data accepted signal and the system goes into its initial state.

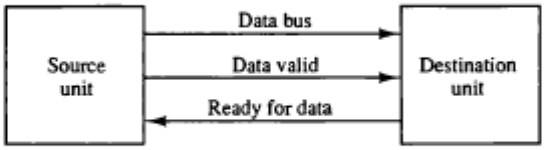
DESTINATION INITIATED TRANSFER USING HANDSHAKING

* In this case the name of the signal generated by the destination unit is ready for data.

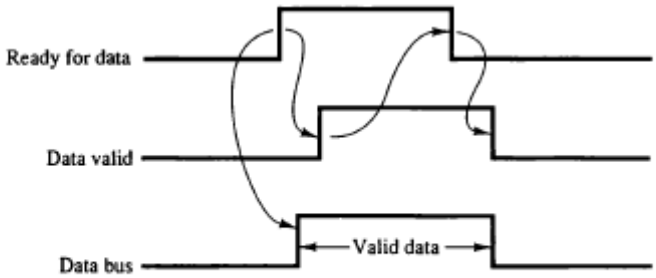
* The source unit does not place the data on the bus until it receives the ready for data signal from the destination unit.

* The handshaking procedure follows the same pattern as in source initiated case. The sequence of events in both the cases is almost same except the ready for signal has been converted from data 15 accepted in case of source initiated.

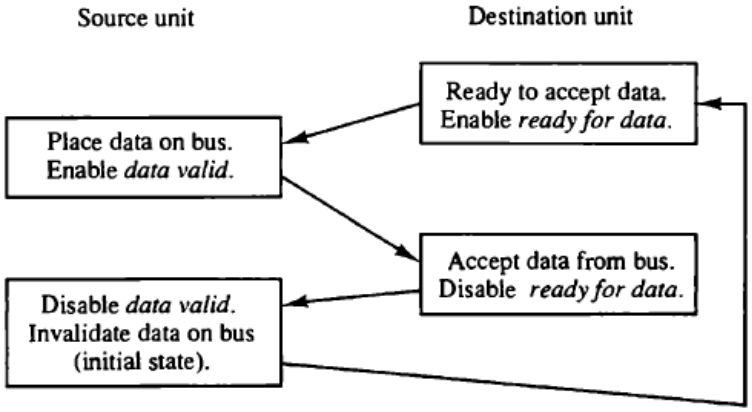
Figure 11-6 Destination-initiated transfer using handshaking.



(a) Block diagram



(b) Timing diagram



(c) Sequence of events

Modes of I/O Transfer(Types of I/O)

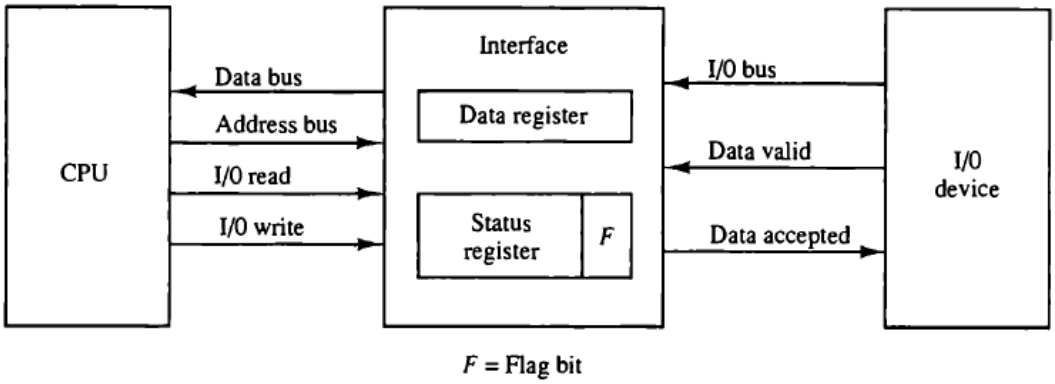
Binary information received from an external device is usually stored in memory for later processing. CPU merely executes I/O instructions and may accept data from memory unit which in fact is ultimate source or destination. Data transfer between the central computer and I/O devices may be handled in one 3 modes:

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

A. 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 and from CPU and memory. Transferring 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. It is upto the programmed instructions executed in the CPU to keep close tabs on everything that is taking place in the interface unit and the I/O device. In programmed I/O method, I/O device does not have direct access to memory. Transfer from peripheral to memory/CPU requires the execution of several I/O instructions by CPU.

Figure 11-10 Data transfer from I/O device to CPU.



F = Flag bit

This diagram shows data transfer from I/O device to CPU. Device transfers bytes of data one at a time as they are available. When a 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 will refer to as an F or 'flag' bit.

Now for programmed I/O, a program is written for the computer to check the flag bit to determine if I/O device has put byte of data in data register of interface.

Flowchart of the program that must be written to the CPU is shown here.The transfer of each byte (assuming device is sending sequence of bytes) requires 3 instructions:

1. Read status register
2. check F bit. if not set branch to a) and if set branch to c).
3. Read data register

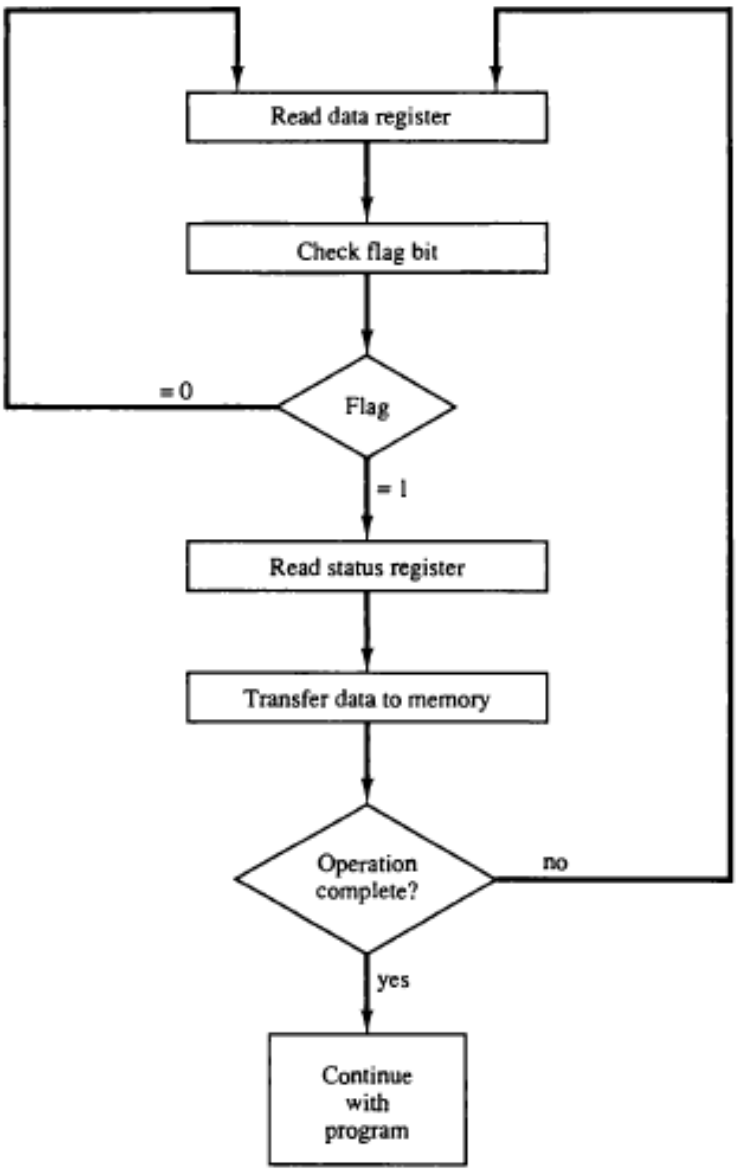


Figure 11-11 Flowchart for CPU program to input data.

B.INTERRUPT-INITIATED I/O

Since polling(constantly monitoring the flag F) takes valuable CPU time, alternative for CPU 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 momentarily 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. The CPU responds to the interrupt signal by storing there turn address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.

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 polling 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.

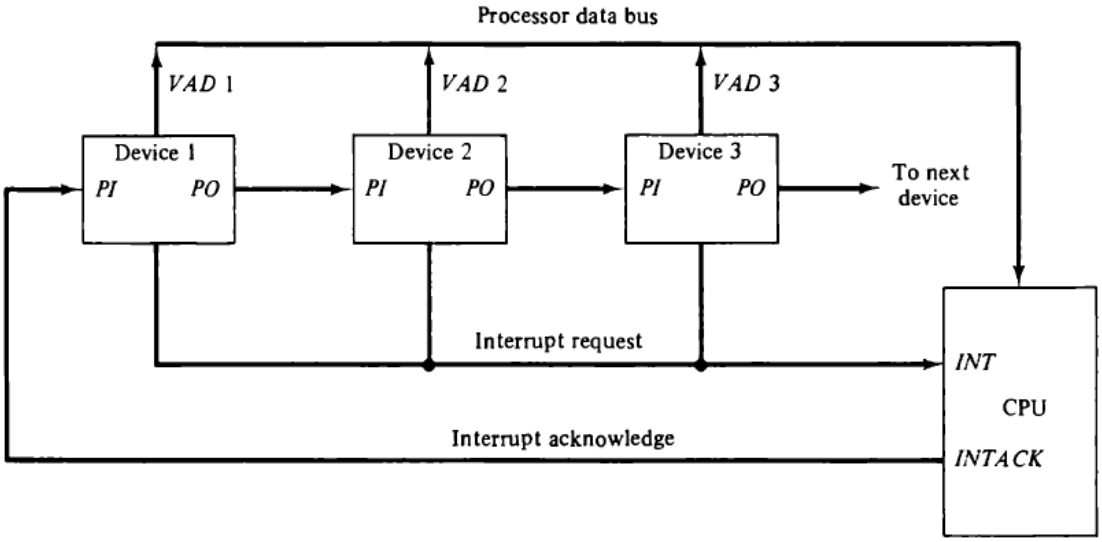


Figure 11-12 Daisy-chain priority interrupt.

 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

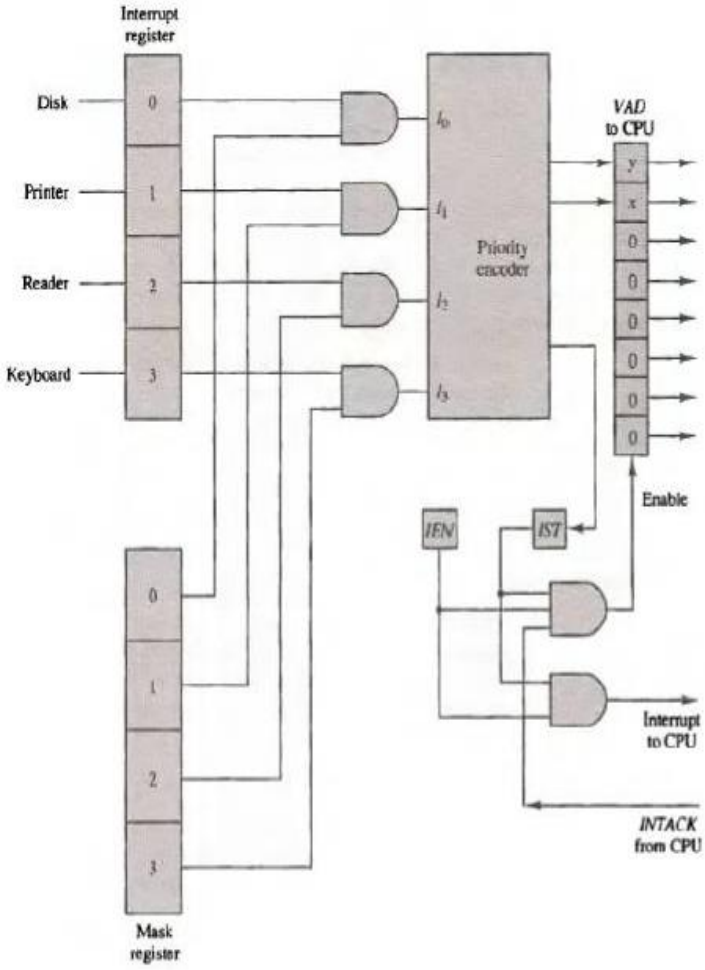


Figure 11-14 Priority interrupt hardware.

- 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)

C. 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 the memory,for fast data transfers.Using DMA ,we get the fastest data transfer rates possible.

Momentum behind DMA: Interrupt driven and programmed I/O require active CPU intervention i.e.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 manages the memory buses directly would improve the speed of transfer.

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

- Bus Request(BR): It is used by DMA controller to request the CPU for buses.When this input is active, CPU terminates the execution the current instruction and places the address bus, data bus and read & write lines into a high impedance state(open circuit).
- 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 the bus system, the transfer with the memory can be made with the following 2 ways:

- Burst Transfer: A block sequence consisting of a number of memory words is transferred in continuous burst . Needed for fast devices as magnetic disks where data transmission can not be stopped or slowed down until whole block is transferred.
- Cycle Stealing: This allows 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 delay sits operation for one memory cycle to allow DMA to “steal”one memory cycle.

DMA Transfer

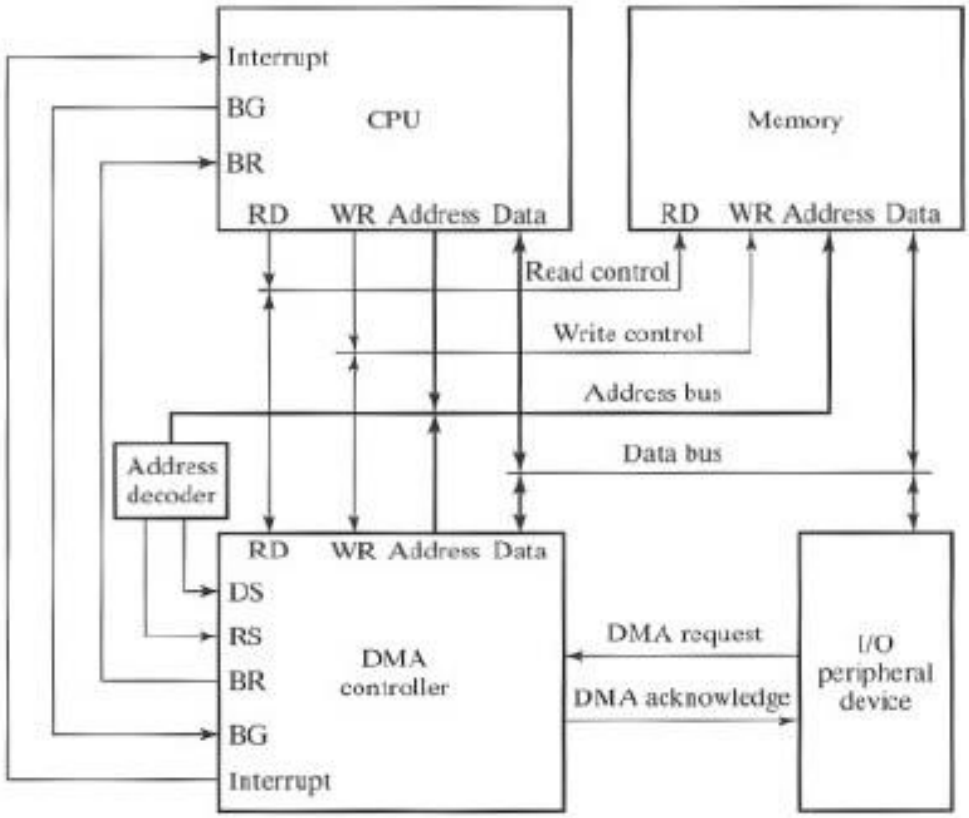


fig. DMA transfer in Computer System

- CPU communicates with the DMA through address and data buses.
- DMA has its own address which activates RS(Register select) and DS (DMA select) lines.
- When a peripheral device sends a DMA request, the DMA controller activates the BR line, informing CPU to leave buses. The CPU responds with its BG line.
- DMA then puts current value of its address register into the address bus,initiates RD or WR signal, and sends a DMA acknowledge to the peripheral devices.
- When BG=0, RD & WR allow CPU to communicate with internal DMA registers and when BG=1, DMA communicates with RAM through RD & WR lines.

Input-Output Processor

- IOP is a processor with direct memory access capability that communicates with I/O devices.
- In this configuration, the computer system can be divided into a memory unit, and a number of processors comprised of CPU and one or more IOPs. IOP is similar to CPU except that it is designed to handle the details of I/O processing.
- Unlike DMA controller which is setup completely by the CPU, IOP can fetch and execute its own instructions.
- IOP instructions are designed specifically to facilitate I/O transfers. Instructions that are read from memory by an IOP are called commands to differ them from instructions read by CPU .
- The command words constitute the program for the IOP. The CPU informs the IOP where to find commands in memory when it is time to execute the I/O program.

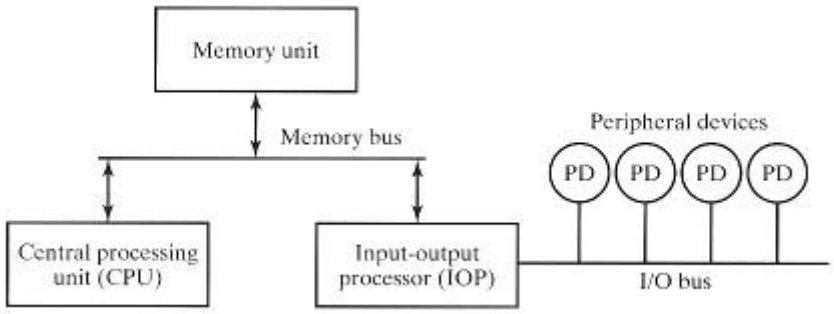


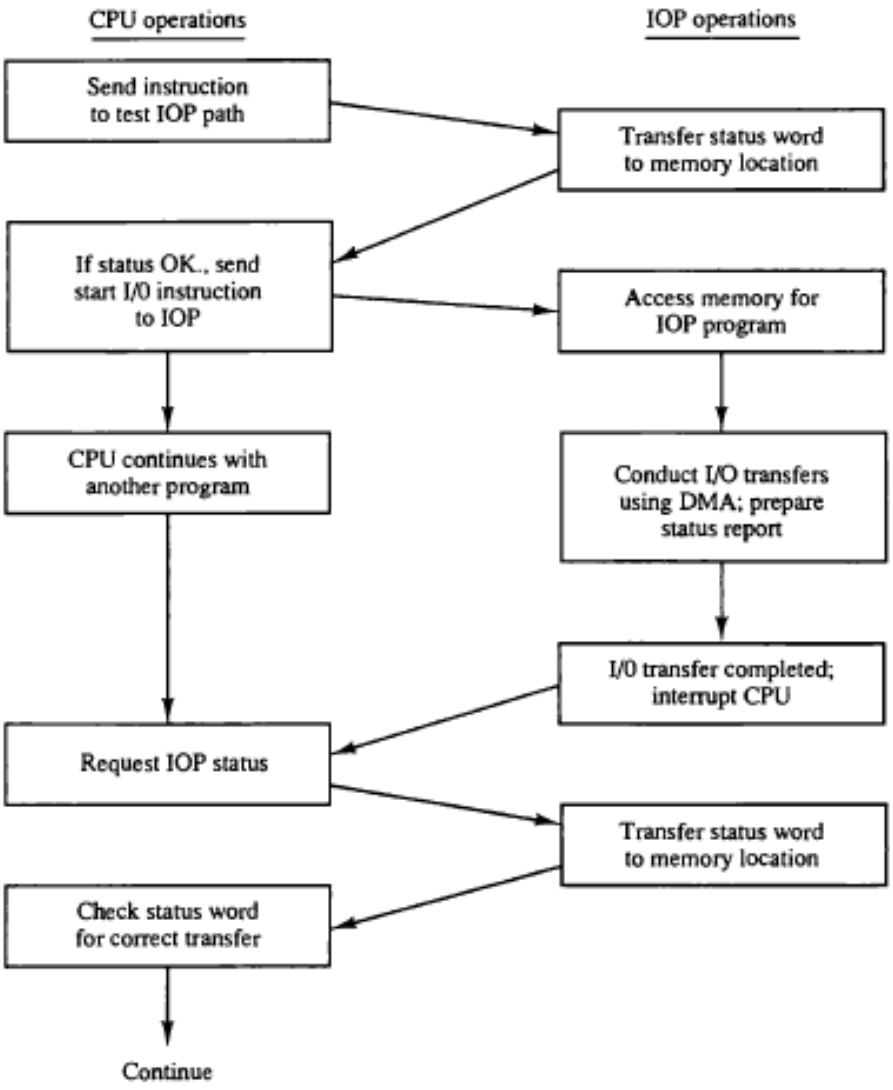
fig. I/O processor

- The memory occupies a central position and can communicate with each processor by means of DMA.
- CPU is usually assigned the task of initiating the I/O program, from the non;IOP operates independent of the CPU and continues to transfer data from external devices and memory.

CPU-IO Communication

- Communication between the CPU and IOP may take different forms depending on the particular computer used. Mostly, memory unit acts as a memory center where each processor leaves information for the other.
- CPU sends an instruction to test the IOP path.
- The IOP responds by inserting a status word in memory for the CPU to check.
- The bits of the status word indicate the condition of IOP and I/O device ("IOP overload condition", "device busy with another transfer" etc).
- CPU then checks status word to decide what to do next .
- If all is in order, CPU sends the instruction to start the I/O transfer.
- The memory address received with this instruction tells the IOP where to find its program.
- CPU may continue with another program while the IOP is busy with the I/O program.
- When IOP terminates the transfer (using DMA), it sends an interrupt request to CPU.
- The CPU responds by issuing an instruction to read the status from the IOP and IOP then answers by placing the status report into specified memory location.
- By inspecting the bits in the status word, CPU determines whether the I/O operation was completed satisfactorily and the process is repeated again.

Figure 11-20 CPU-IOP communication.



****