# Input Output Organization

**Contents**

- **8.1 Input-Output Interface**: *Why IO Interface?*, I/O Bus and Interface Modules, I/O vs Memory Bus, Isolated vs Memory Mapped I/O
- **8.2 Asynchronous Data Transfer**: Strobe, Handshaking
- **8.3 Modes of Transfer**: Programmed I/O, Interrupt-Initiated I/O, Direct memory Access
- **8.4 Priority Interrupt**: Polling, Daisy-Chaining, Parallel Priority Interrupt
- **8.5 DMA and IOP**: Direct Memory Access, Input-Output Processor, DMA vs IOP

**Peripheral Devices**

- The Input / output organization of computer depends upon the size of computer and the peripherals connected to it.
- The I/O Subsystem of the computer, provides an efficient mode of communication between the central system and the outside environment The most common input output devices are:

  i) Monitor

  ii) Keyboard

  iii) Mouse

  iv) Printer

  v) Magnetic tapes
- Input-output interface provides a method for transferring information between internal storage and external I/0 devices.
- The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral.

## Input-Output Interface

- The major differences are:
  - **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.
  - **The data transfer rate** of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be needed.
  - **Data codes and formats** in peripherals differ from the word format in the CPU and memory.
  - **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 resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers, which are called interface units because they interface between the processor bus and the peripheral device.

## I/O Bus and Interface Modules

- Peripherals connected to a computer need special communication links for interfacing them with the central processing unit .this is called I/O bus.

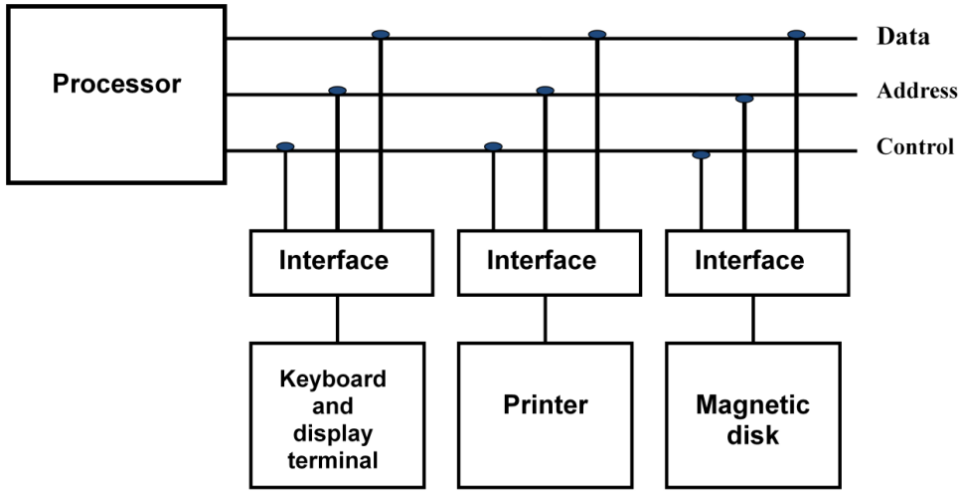



**Figure: Connection of I/O bus to input-output devices**

- Interface performs the following:
  - Decodes the device address (device code)
  - Decodes the commands (operation)
  - Provides signals for the peripheral controller
  - Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory

**I/0 command**

- The function code is referred to as an I/O command and is in essence an instruction that is executed in the interface and its attached peripheral unit.
- There are four types of commands that an interface may receive. They are classified as control, status, data output, and data input.
- **Control command-** A control command is issued to activate the peripheral and to inform it what to do.
- **Status command-** A status command is used to test various status conditions in the interface and the peripheral.
- **Data Output command-** A data output command causes the interface to respond by transferring data from the bus into one of its registers.
- **Data Input command-** The data input command is the opposite of the data output. In this case the interface receives on item of data from the peripheral and places it in its buffer register. I/O Versus Memory Bus .

## I/O versus Memory Bus

- To communicating with I/O, the processor must communicate with the memory unit. Like the I/O bus, the memory bus contains data, address, and read/write control lines.
- There are three ways that computer buses can be used to communicate with memory and I/O:
  1. Use two separate buses, one for memory and the other for I/O.
  2. Use one common bus for both memory and I/O but have separate control lines for each.
  3. Use one common bus for memory and I/O with common control lines.

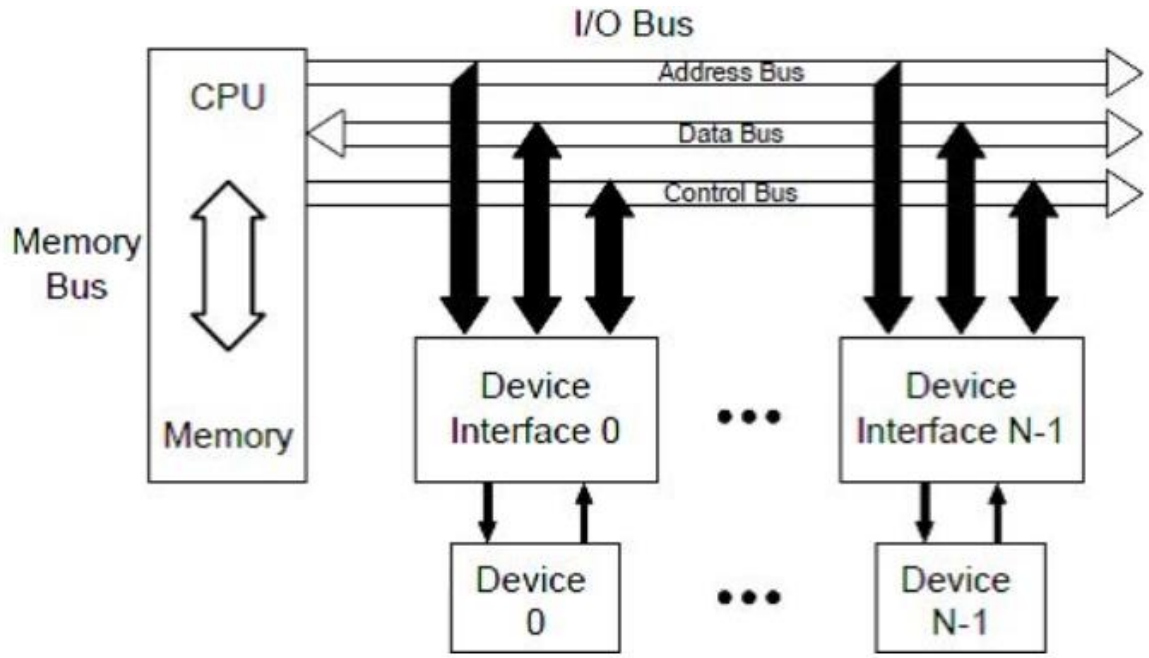
## I/O Processor

- In the first method, the computer has independent sets of data, address and control buses one for accessing memory and other for I/O. This is done in computers that provides a separate I/O processor (IOP).
- The purpose of IOP is to provide an independent pathway for the transfer of information between external device and internal memory.

## Isolated versus Memory-Mapped I/O

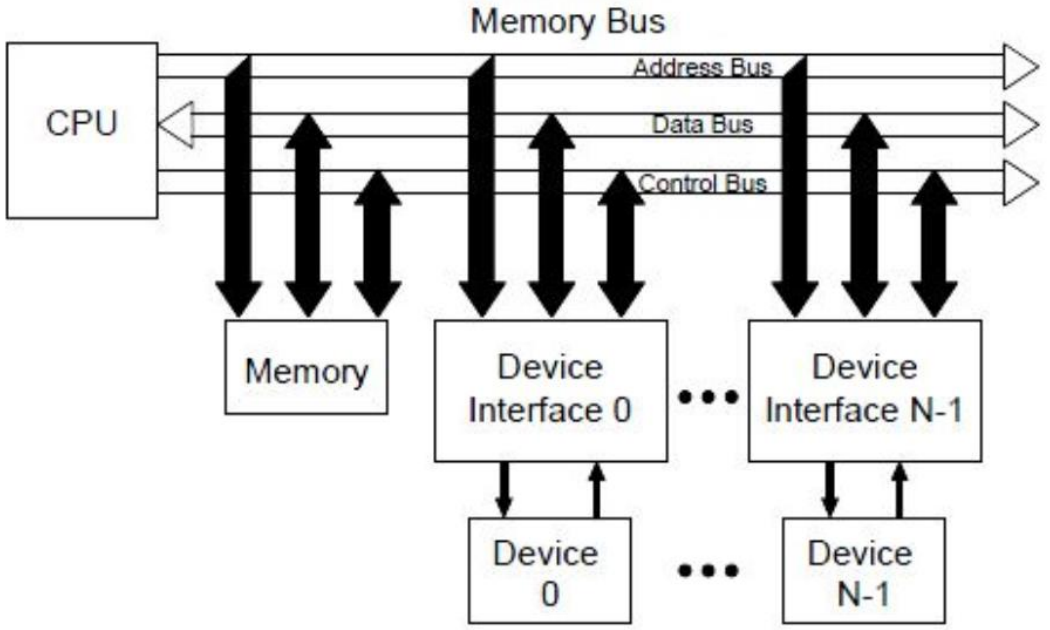
### Isolated I/O

- Common bus(data and address) for I/O and memory but separate read and write control lines for I/O
- Different read-write instruction for both I/O and memory.
- More efficient due to separate buses
- Larger in size due to more buses



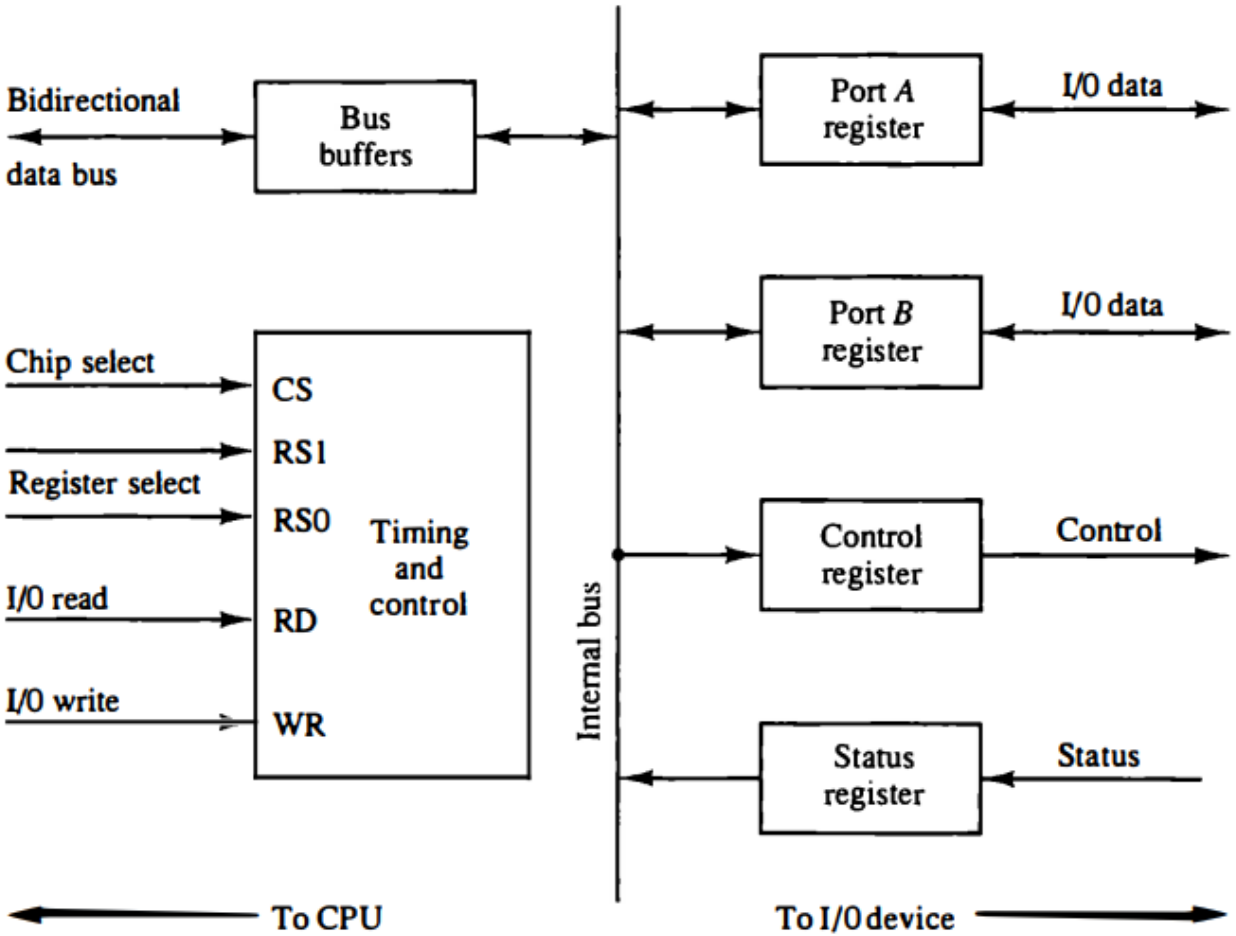
**Memory-mapped I/O**

- A single set of read/write control lines (no distinction between memory and I/O transfer)
- Memory and I/O addresses share the common address space which reduces memory address range available
- No specific input or output instruction so the same memory reference instructions can be used for I/O transfers
- Considerable flexibility in handling I/O operations



**Example of I/O Interface**

- Figure below, It consists of two data registers called ports, a control register, a status register, bus buffers, and timing and control circuits.
- The interface communicates with the CPU through the data bus.
- The chip select and register select inputs determine the address assigned to the interface. The l/0 read and write are two control lines that specify an input or output, respectively.
- The four registers communicate directly with the l/0 device attached to the 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 |

**Figure: Example of I/O interface unit.**

- The interface registers communicate with the CPU through the bidirectional data bus. The address bus selects the interface unit through the chip select and the two register select inputs.
- This circuit enables the chip select (CS) input when the interface is selected by the address bus. The two register select inputs RSl and RS0 are usually connected to the two least significant lines of the address bus. These two inputs select one of the four registers in the interface as specified in the table accompanying the diagram.

## 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.
- Strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.

**Strobe Control :**

- The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either the source or the destination unit.
- The data bus carries the binary information from source unit to the destination unit.

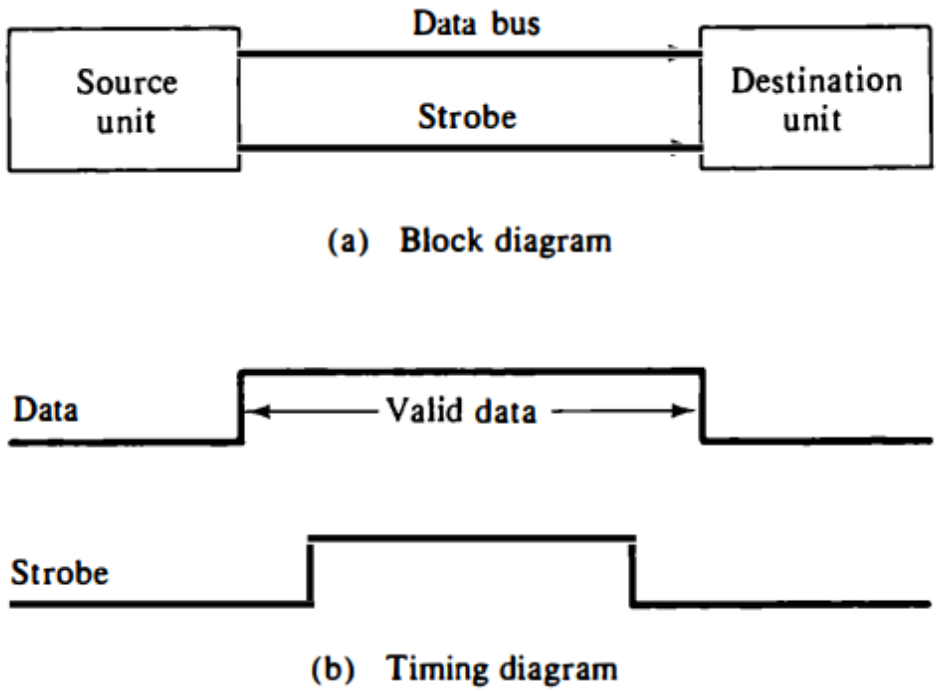



**Figure : Source-initiated strobe for data transfer.**

- **Source-initiated strobe**, the bus has multiple lines to transfer an entire byte or word. The strobe is a single line that informs the destination unit when a valid data word is available in the bus.
- **Destination-initiated strobe**, 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 data bus. The data must be valid and remain in the bus long enough for the destination unit to accept it.

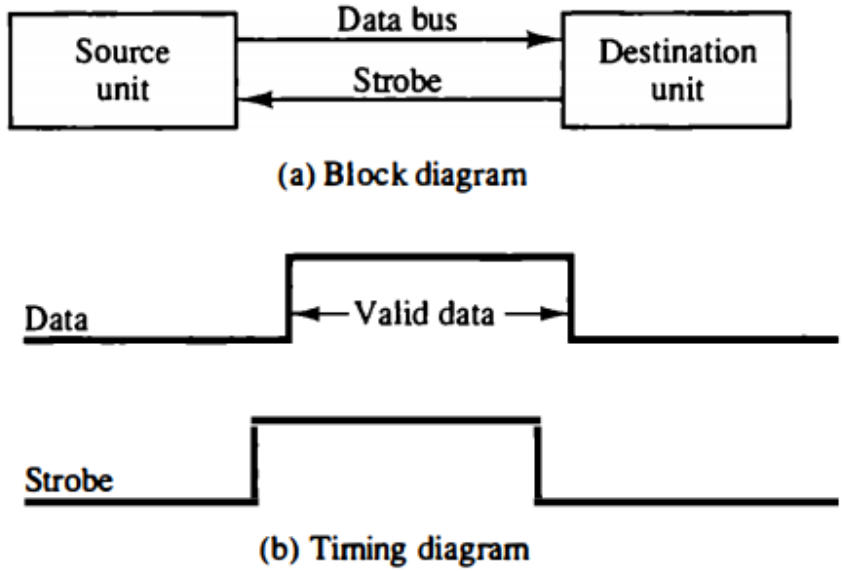



**Figure : Destination-initiated strobe for data transfer.**

**Disadvantage of Strobe Signal**

- The disadvantage of the strobe method is that, the source unit initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was places in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on bus. The Handshaking method solves this problem.

**Handshaking:**

- 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.
- The two handshaking lines are data valid, which is generated by the source unit, and data accepted, generated by the destination unit.
- The **source unit initiates** the transfer by placing the data on the bus and enabling its data valid signal. The data accepted signal 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 then disables its data accepted signal and the system goes into its initial state.
- The source does not send the next data item until after the destination unit shows its readiness to accept new data by disabling its data accepted signal.

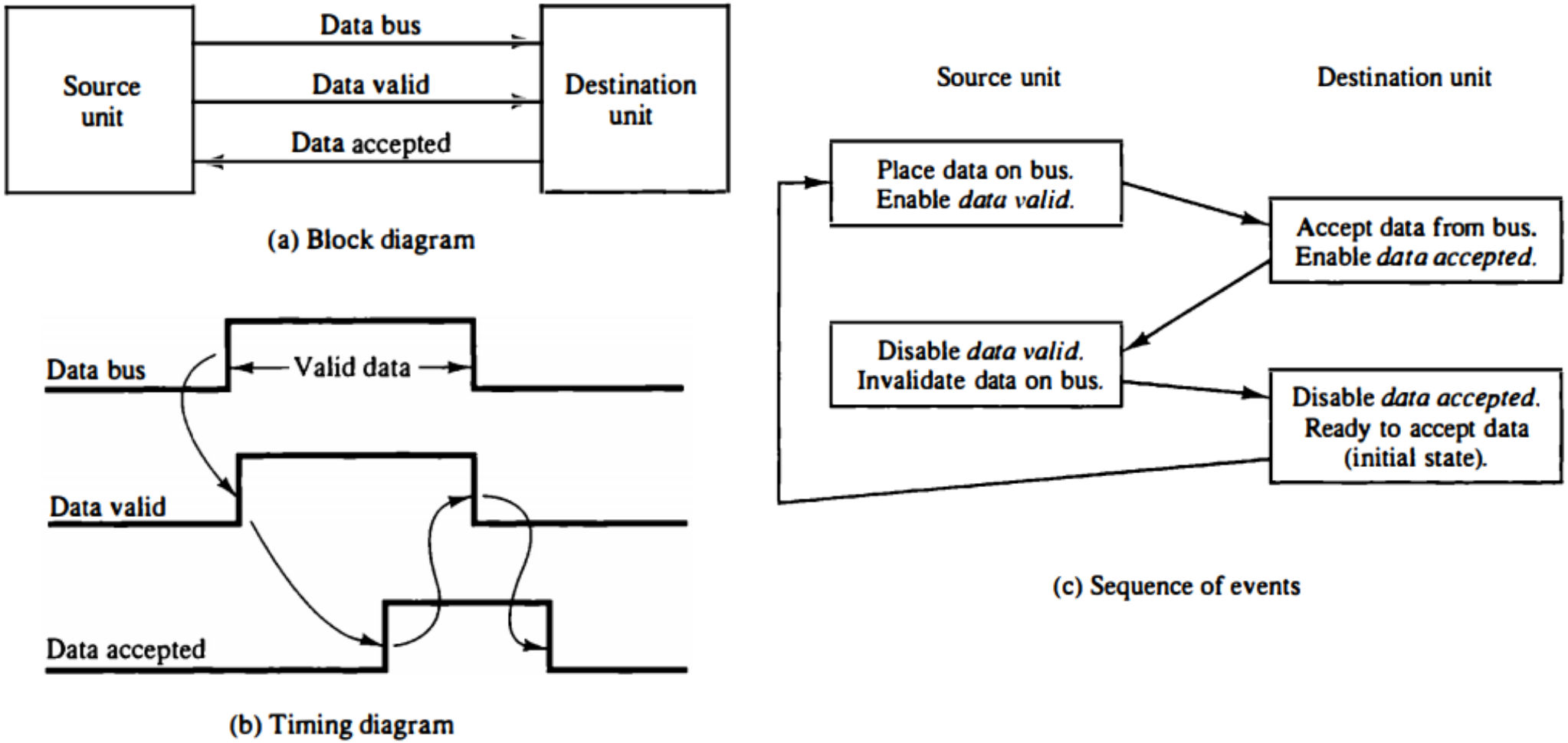



**Figure :Source-initiated transfer using handshaking**

- The **destination-initiated** transfer using handshaking lines is shown in figure.
- The name of the signal generated by the destination unit has been changed to ready for data to reflect its new meaning. The source unit in this case does not place data on the bus until after it receives the ready for data signal from the destination unit. From there on, the handshaking procedure follows the same pattern as in the source-initiated case.
- The handshaking scheme provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units.

Advantage of the Handshaking method:

- The Handshaking scheme provides degree of flexibility and reliability because the successful completion of data transfer relies on active participation by both units.
- If any of one unit is faulty, the data transfer will not be completed. Such an error can be detected by means of a *Timeout mechanism* which provides an alarm if the data is not completed within time.

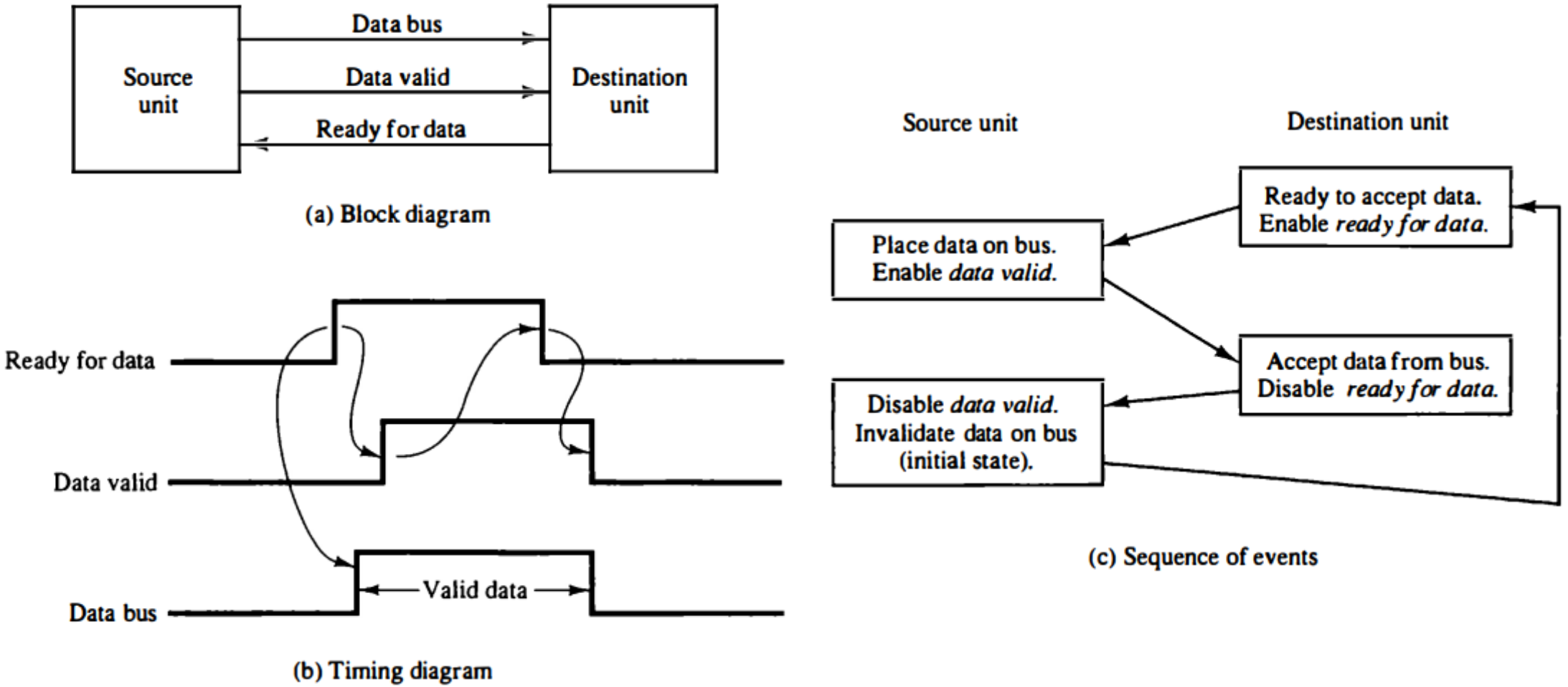



**Figure: Destination-initiated transfer using handshaking.**

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

- Transfer of data is required between CPU and peripherals or memory or sometimes between any two devices or units of your computer system. To transfer a data from one unit to another one should be sure that both units have proper connection and at the time of data transfer the receiving unit is not busy. This data transfer with the computer is Internal Operation.
- All the internal operations in a digital system are synchronized by means of clock pulses supplied by a common clock pulse Generator. The data transfer can be **Synchronous or Asynchronous.**
- When both the transmitting and receiving units use same clock pulse then such a data transfer is called **Synchronous process**.
- If the there is not concept of clock pulses 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:
- Data transfer between the central computer and I/O devices may be handled in a variety of modes.
  - Programmed I/O
  - Interrupt-initiated I/O
  - Direct memory access (DMA)

## 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 up to 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.

- An example of data transfer from an I/O device through an interface into the CPU is shown in Figure.
- The 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 we will refer to as an F or "flag" bit.

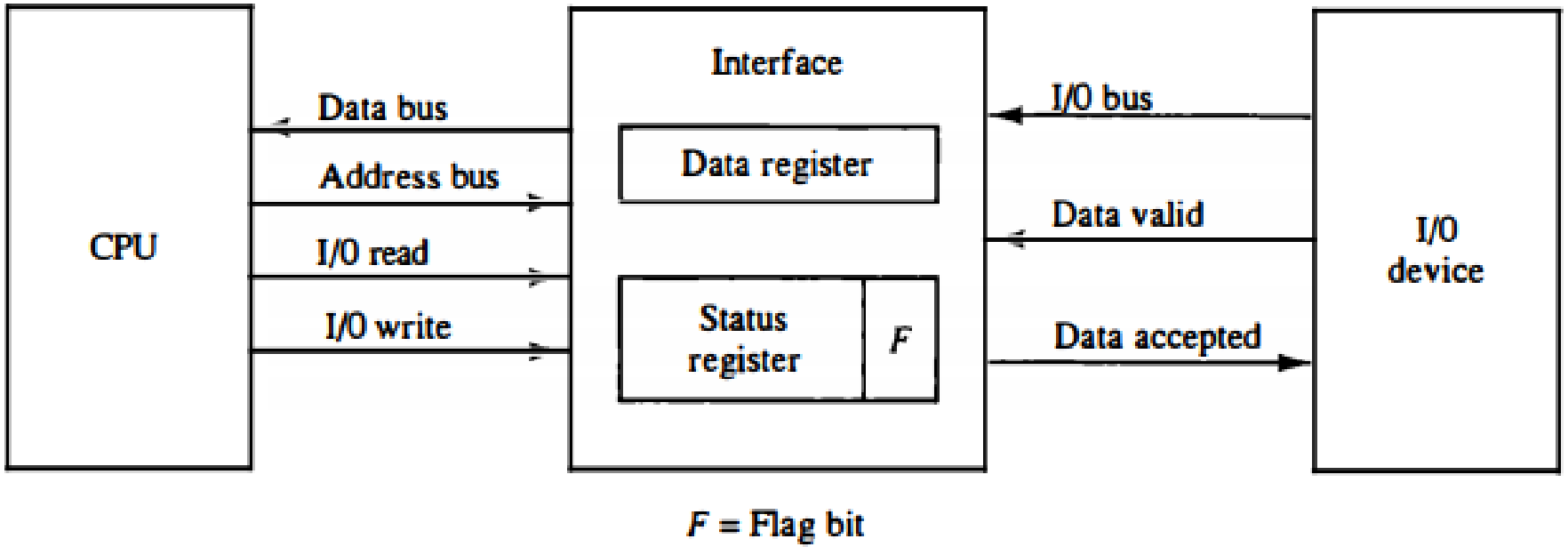



**Figure: Data transfer from I/O device to CPU**

- A flowchart of the program that must be written for the CPU is shown in Figure ,It is assumed that the device is sending a sequence of bytes that must be stored in memory.
- The transfer of each byte 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.

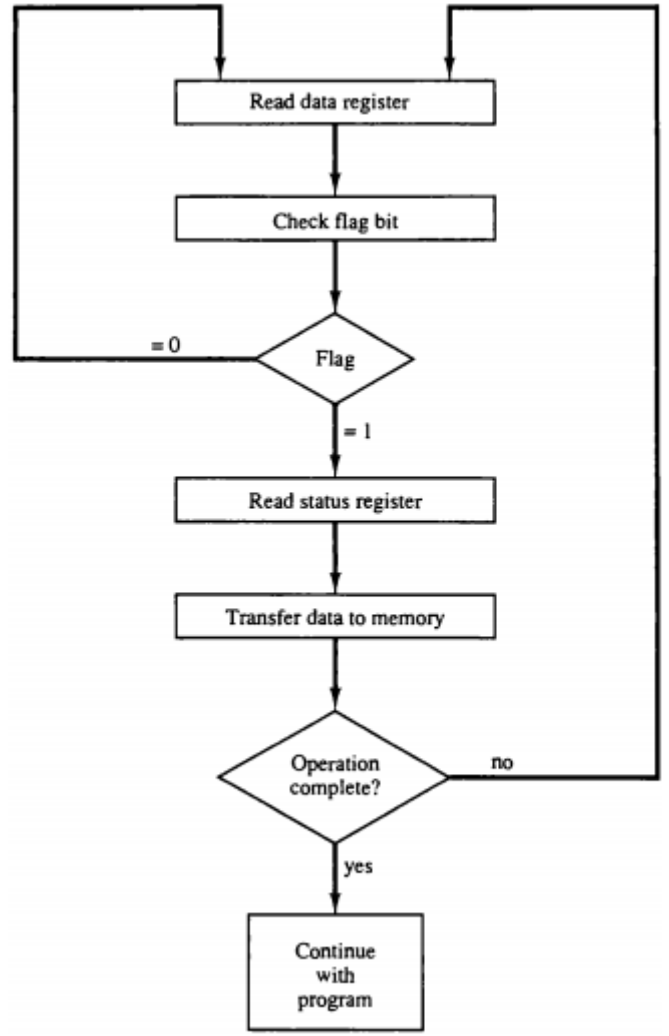


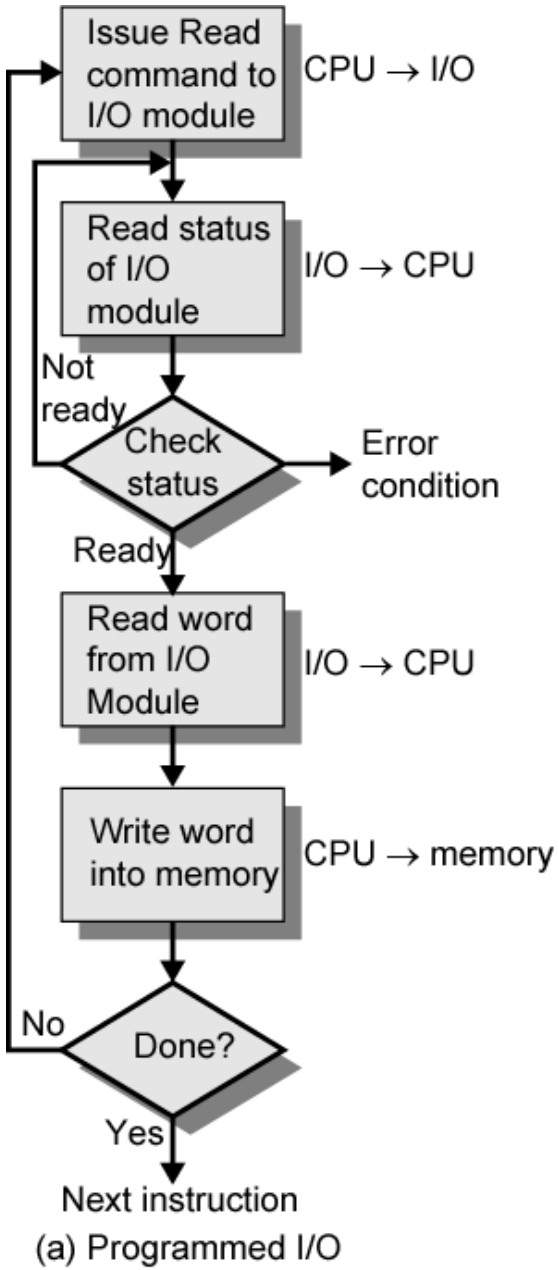

**Figure: Flowchart for CPU program to input data.**

## Interrupt-Initiated I/O

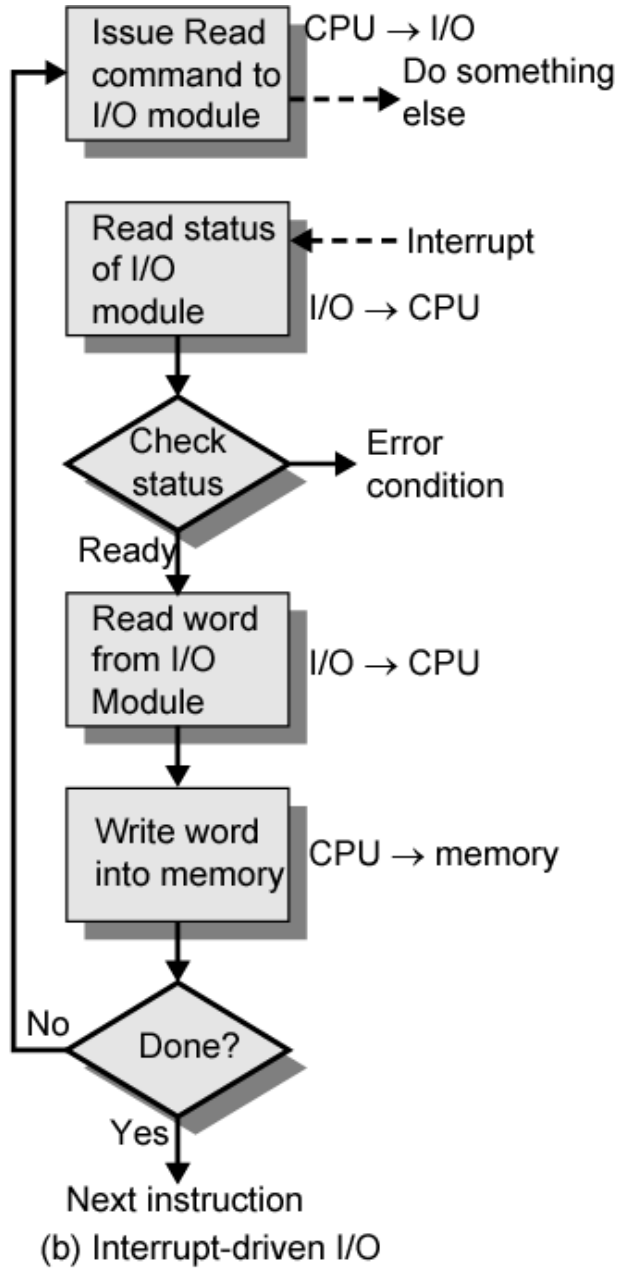
- The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The processor, while waiting, must repeatedly interrogate the status of the I/O module.
- To improve the performance of the system an Interrupt-Initiated I/O can be used where after issuing the I/O command to the I/O module the processor can get itself busy doing some other work, the valuable time of the processor can be utilized.

### Basic Operations of Interrupt

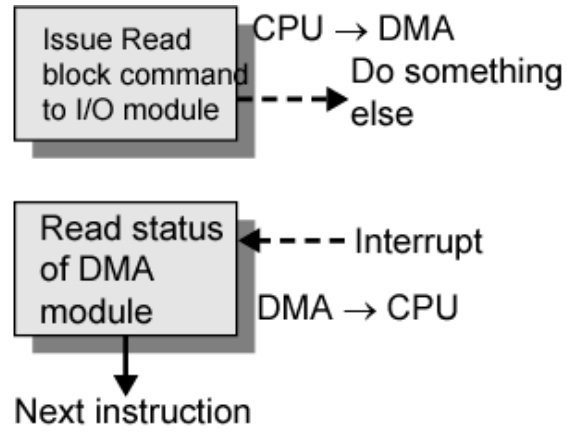
- CPU issues read command.
- I/O module gets data from peripheral whilst CPU does other work.
- I/O module interrupts CPU.
- CPU requests data.
- I/O module transfers data.




(a) Programmed I/O




(b) Interrupt-driven I/O




(c) Direct memory access

## Direct Memory Access(DMA)

- Large blocks of data transferred at a high speed to or from high speed devices, magnetic drums, disks, tapes, etc.
- DMA controller Interface that provides I/O transfer of data directly to and from the memory and the I/O device
- CPU initializes the DMA controller by sending a memory address and the number of words to be transferred
- Actual transfer of data is done directly between the device and memory through DMA controller -> Freeing CPU for other tasks

- The transfer of data between the peripheral and memory without the interaction of CPU and letting the peripheral device manage the memory bus directly is termed as Direct Memory Access (DMA)

- During the 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 microprocessor is to disable the buses through special control signals such as:
  - **Bus Request (BR)**
  - **Bus Grant (BG)**

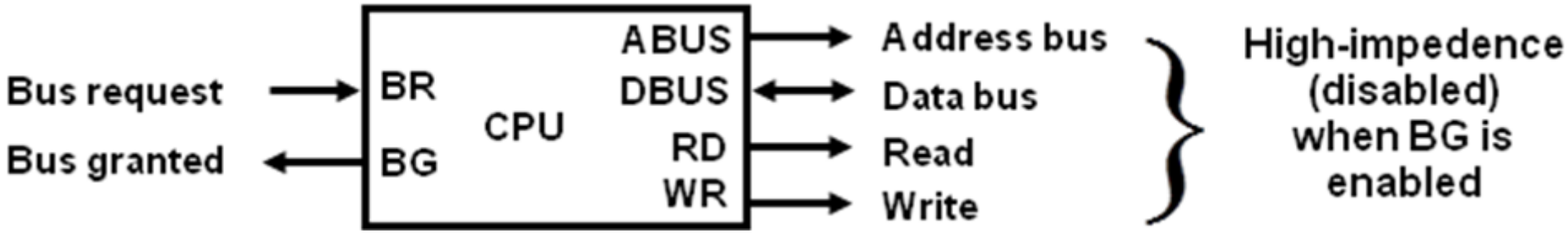



**Figure: CPU bus signal for DMA transfer**

- These two control signals in the CPU that facilitates the DMA transfer. The *Bus Request (BR)* input is used by the *DMA controller* to request the CPU. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, data bus and read write lines into a *high Impedance state*. High Impedance state means that the output is disconnected.
- The CPU activates the *Bus Grant (BG)* output to inform the external DMA that the Bus Request (BR) can now take control of the buses to conduct memory transfer without processor.
- When the DMA terminates the transfer, it disables the *Bus Request (BR)* line. The CPU disables the *Bus Grant (BG)*, takes control of the buses and return to its normal operation.
- The transfer of data between the memory and I/O of course facilitates in two ways which are **DMA Burst and Cycle Stealing**.

- **DMA Burst**: The block of data consisting a number of memory words is transferred at a time.
- **Cycle Stealing**: DMA transfers one data word at a time after which it must return control of the buses to the CPU.
  - CPU is usually much faster than I/O (DMA), thus CPU uses the most of the memory cycles
  - DMA Controller steals the memory cycles from CPU
  - For those stolen cycles, CPU remains idle
  - For those slow CPU, DMA Controller may steal most of the memory cycles which may cause CPU remain idle long time

**DMA Controller:**

- The hardware device used for direct memory access is called the DMA controller. DMA controller is a control unit, part of I/O device's interface circuit, which can transfer blocks of data between I/O devices and main memory with minimal intervention from the processor. or
- The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device.
- The DMA controller has three registers:

  i. **Address Register:** The address register specifies the desired location of the memory which is incremented after each word is transferred to the memory.

  ii**. Word Count Register:** The word count register holds the number of words to be transferred which is decremented after each transfer until it is zero. When it is zero, it indicates the end of transfer. After which the bus grant signal from CPU is made low and CPU returns to its normal operation.

  iii. **Control Register:** The control register specifies the mode of transfer which is Read or Write.

- 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 (DMA select) and RS (Register select) inputs. The RD (read) and WR (write) inputs are bidirectional.
- When the BG (Bus Grant) input is 0, 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 DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.

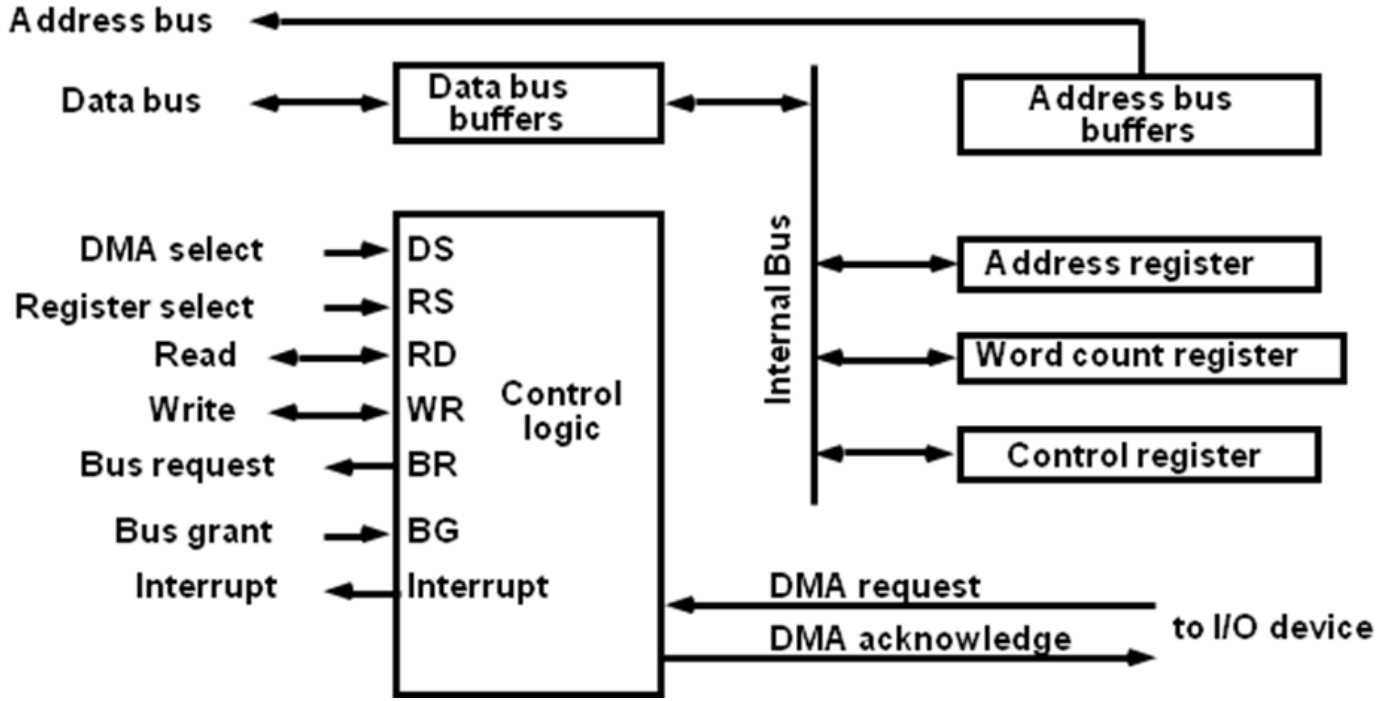



**Figure: Block diagram of DMA controller**

**DMA Transfer:**

- 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 transfer between the peripheral and the memory.
- 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 random access memory to specify the read or write operation of data.

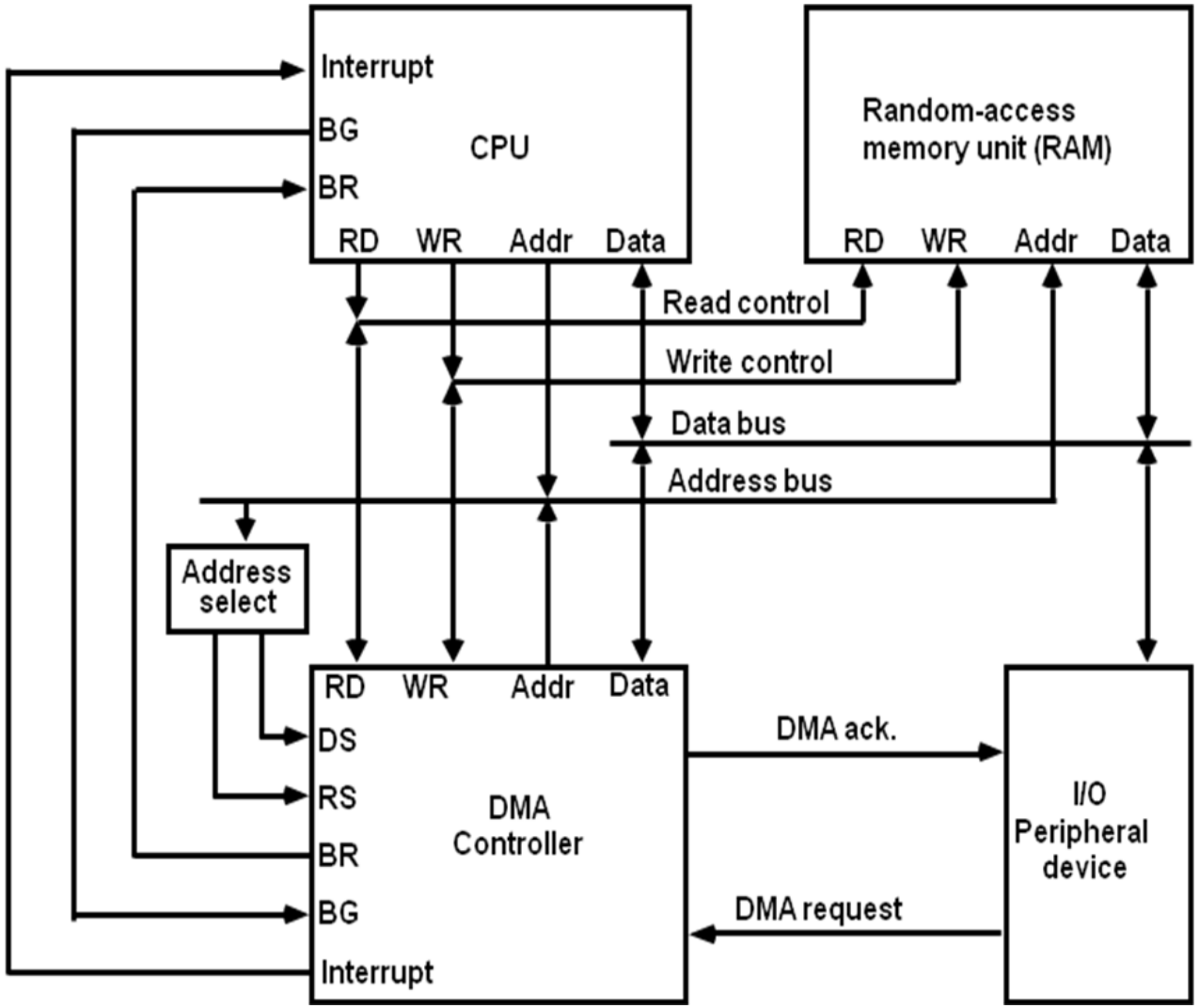



**Figure: DMA transfer in a computer system**

## Priority Interrupt

- Determines which interrupt is to be served first when two or more requests are made simultaneously
- Also determines which interrupts are permitted to interrupt the computer while another is being serviced
- Higher priority interrupts can make requests while servicing a lower priority interrupt
- 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.

**Priority Interrupt by Software (Polling)**

- Priority is established by the order of polling the devices (interrupt sources), that is identify the highest-priority source by software means
- One common branch address is used 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
- Low cost since it needs a very little hardware
- Very slow

**Priority Interrupt by Hardware**

- Require a priority interrupt manager which accepts all the interrupt requests to determine the highest priority request
- Fast since identification of the highest priority interrupt request is identified by the hardware
- Fast since each interrupt source has its own interrupt vector to access directly to its own service routine.

- **Daisy Chain Priority (Serial)**

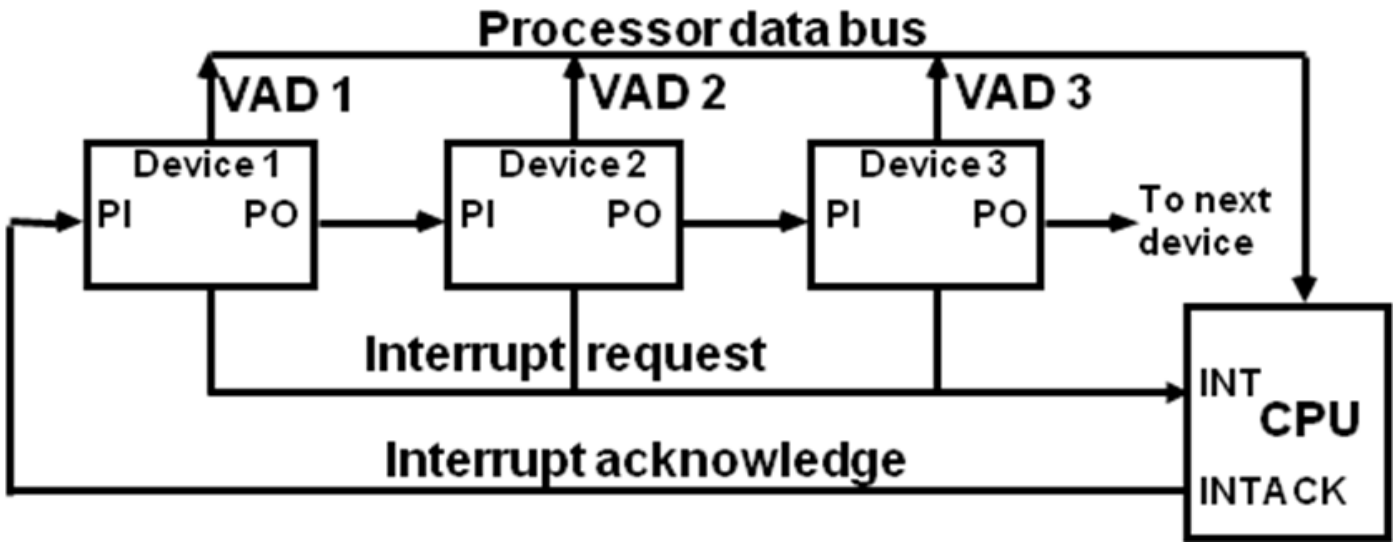
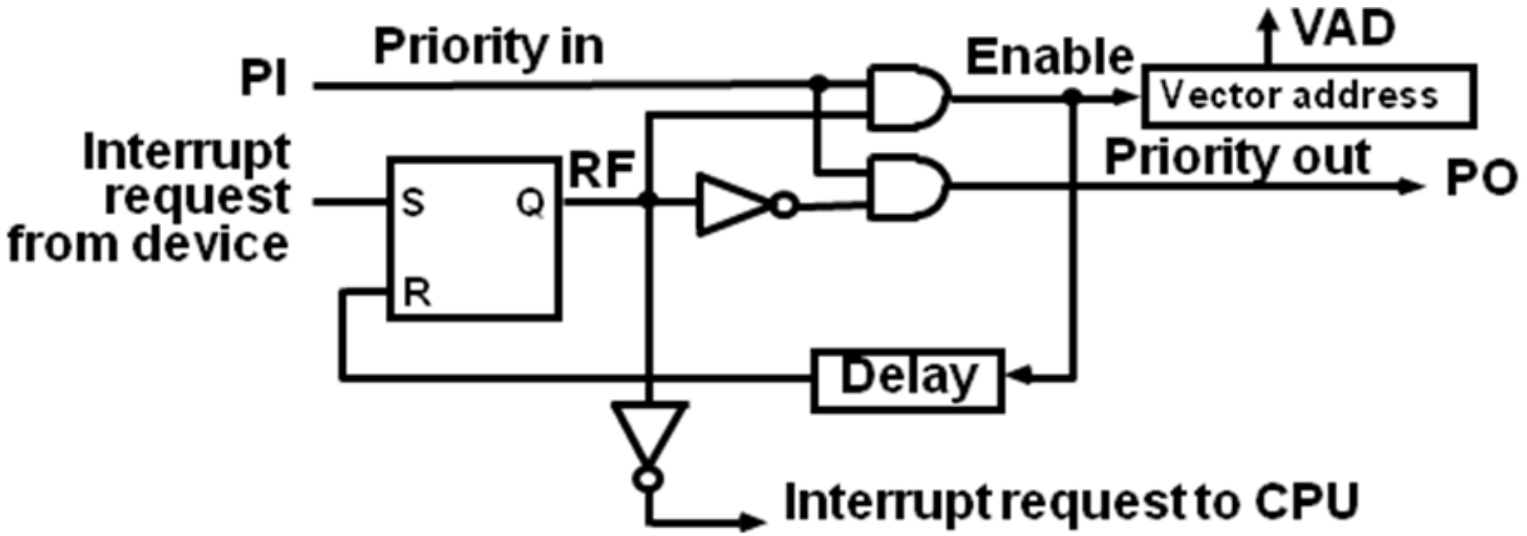



**Figure : Daisy chain priority interrupt**

- Device with highest priority is placed first.
- Interrupt Request from any device
- CPU responds by INTACK
- Any device receives signal(INTACK) at PI puts the VAD on the bus
- Among interrupt requesting devices the only device which is physically closest to CPU gets INTACK and it blocks INTACK to propagate to the next device




| PI | RF | PO | Enable |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |

**Figure: One stage of Daisy chain priority arrangement**

- **Parallel Priority**
- IEN: Set or Clear by instructions ION or IOF
- IST: Represents an unmasked interrupt has occurred. INTACK enables tristate Bus Buffer to load VAD generated by the Priority Logic
- Interrupt Register
  - Each bit is associated with an Interrupt Request from different Interrupt Source - different priority level
  - Each bit can be cleared by a program instruction
- Mask Register:
  - Mask Register is associated with Interrupt Register
  - Each bit can be set or cleared by an Instruction

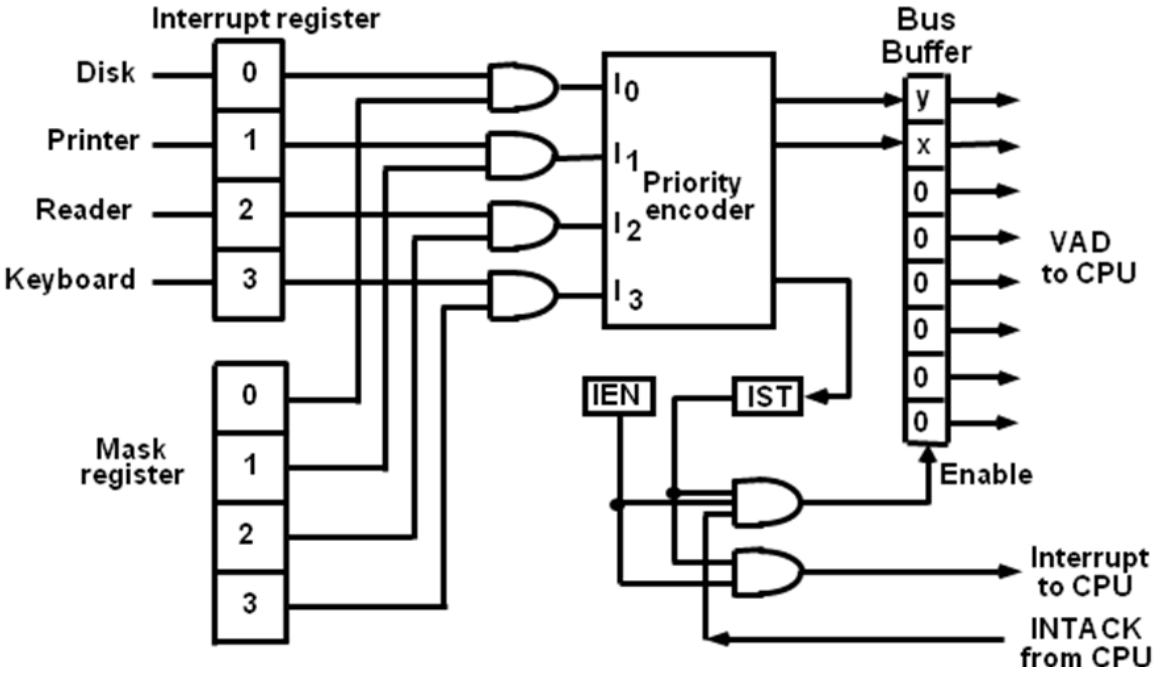



**Figure: Parallel priority interrupts hardware**

## Priority Encoder Truth Table

- Determines the highest priority interrupt when more than one interrupts take place

| Inputs | | | | Outputs | | | Boolean functions |
|---|---|---|---|---|---|---|---|
| $I_0$ | $I_1$ | $I_2$ | $I_3$ | x | y | IST | |
| 1 | d | d | d | 0 | 0 | 1 | |
| 0 | 1 | d | d | 0 | 1 | 1 | |
| 0 | 0 | 1 | d | 1 | 0 | 1 | $x = I_0' \; I_1'$ |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | $y = I_0' I_1 + I_0' I_2'$ |
| 0 | 0 | 0 | 0 | d | d | 0 | $(IST) = I_0 + I_1 + I_2 + I_3$ |

**Figure: Priority Encoder Truth Table**

- 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.
- During the interrupt cycle the CPU performs the following sequence of microoperations:

  SP ← SP – 1; Decrement stack pointer

  M[SP] ← PC; Push PC into stack

  INTACK ← 1; Enable interrupt acknowledge

  PC ← VAD; Transfer vector address to PC

  IEN ← 0; Disable further interrupts

  Go To Fetch to execute the first instruction in the interrupt service routine

## I/O Processors

- It is a processor with direct memory access capability that communicates with IO devices.
- IOP is similar to CPU except that it is designed to handle the details of IO operation.
- Unlike DMA which is initialized by CPU, IOP can fetch and execute its own instructions.
- IOP instruction are specially designed to handle IO operation.
- Memory occupies the central position and can communicate with each processor by DMA.
- CPU is responsible for processing data.
- IOP provides the path for transfer of data between various peripheral devices and memory.
- Data formats of peripherals differ from CPU and memory. IOP maintain such problems.
- Data are transfer from IOP to memory by stealing one memory cycle.
- Instructions that are read from memory by IOP are called commands to distinguish them from instructions that are read by the CPU.

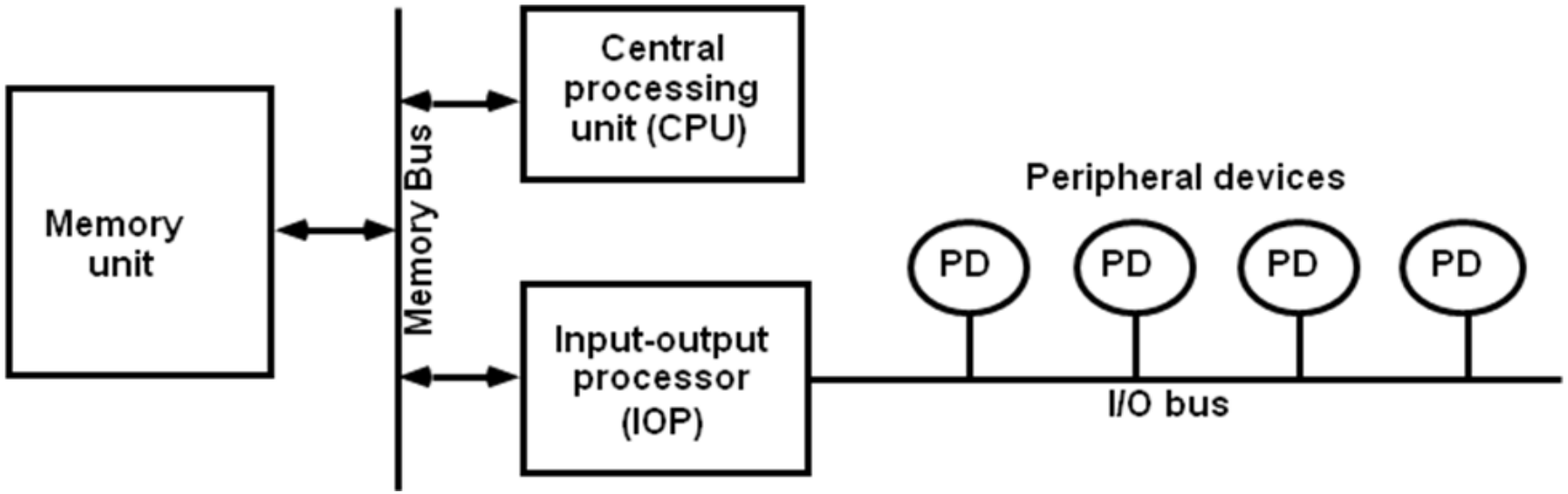



**Figure: Block diagram of a computer with I/O Processor**

## CPU – IOP Communication

The memory unit acts as a message center where each processor leaves information for the other. The operation of typical IOP is appreciated with the example by which the CPU and IOP communication.

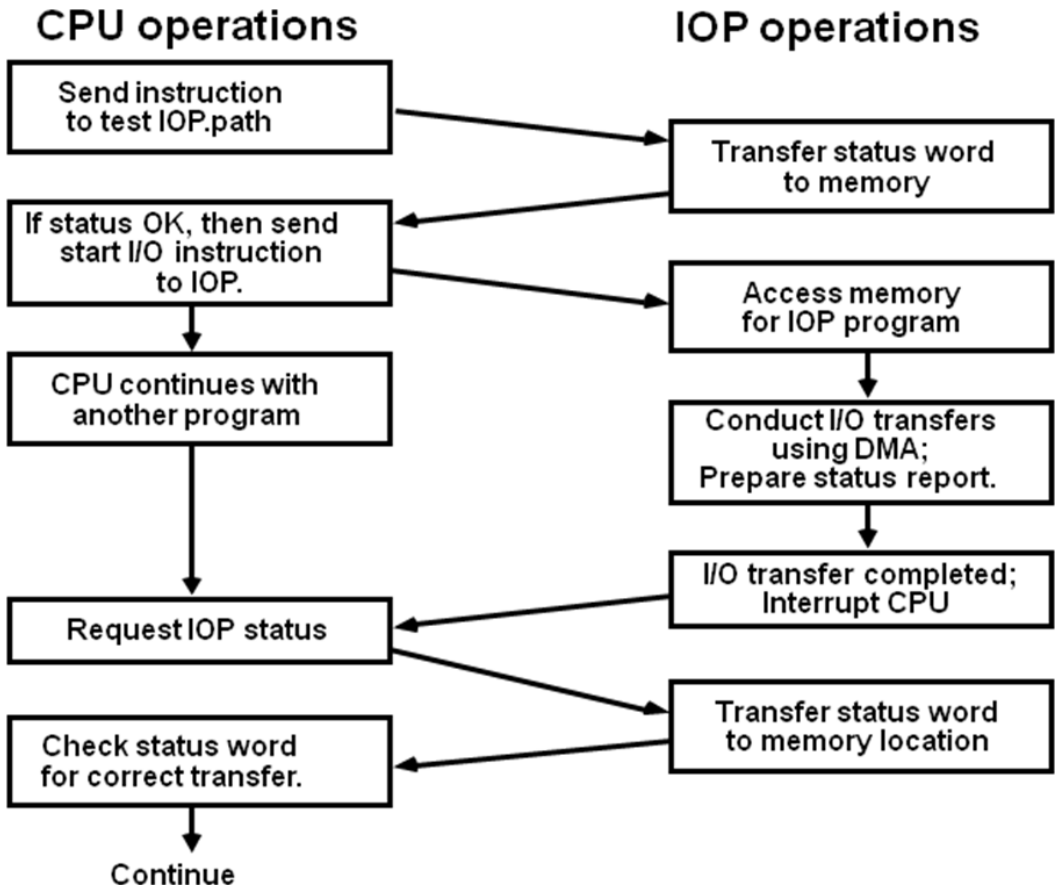



**Figure: CPU – IOP communication**

- The 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 the IOP and I/O device, such as IOP overload condition, device busy with another transfer or device ready for I/O transfer.
- The CPU refers to the status word in in memory to decide what to do next.
- If all right up to this, the CPU sends the instruction to start I/O transfer.
- The CPU now continues with another program while IOP is busy with I/O program.
- When IOP terminates the execution, it sends an interrupt request to CPU.
- CPU responds by issuing an instruction to read the status from the IOP.
- IOP responds by placing the contents to its status report into specified memory location.
- Status word indicates whether the transfer has been completed or with error.