Chapter 8
Input/Output Architecture

Outline

- Peripheral devices
- Properties of simple I/O devices and their controllers
- Asynchronous data transfer handshaking
- Data transfer modes
- Programmed I/O, Interrupted I/O, DMA
- Transfer of information between I/O devices , CPU and memory.

Peripheral Devices

Input Devices

- Keyboard
- Optical input devices
  - Card Reader
  - Paper Tape Reader
  - Bar code reader
  - Digitizer
  - Optical Mark Reader
- Magnetic Input Devices
  - Magnetic Stripe Reader
- Screen Input Devices
  - Touch Screen
  - Light Pen
  - Mouse
- Analog Input Devices

Output Devices

- Card Puncher, Paper Tape Puncher
- CRT
- Printer (Impact, Ink Jet, Laser, Dot Matrix)
- Plotter
- Analog
- Voice

Input/output Device or External or Peripheral Device

- A typical microcomputer system consist of microprocessor plus memory and I/O interface.
- The various components that form the system are linked through the buses that transfer instructions, data, addresses and control information among the components.
- The block diagram of a microcomputer system is shown in Figure. 1.

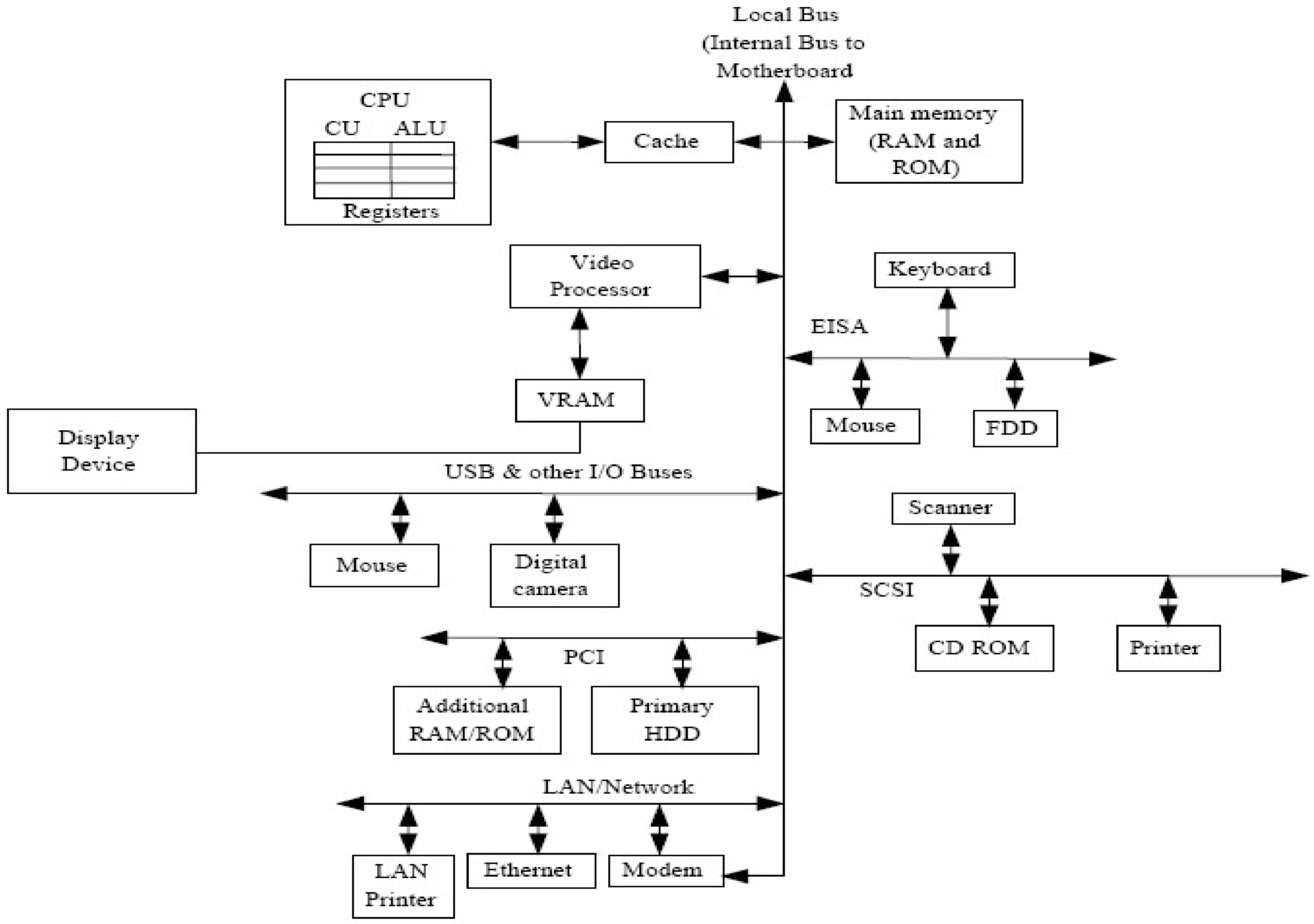


Figure 1: Block Diagram of a Microcomputer System

Input/output Device or External or Peripheral Device

- The microcomputer has a single microprocessor, a number of RAM and ROM chips and an interface units communicates with various external devices through the I/O Bus.
- The Input / Output subsystem of a computer, referred to as I/O, provides an efficient mode of communication between the central system and the output environment.
- External devices that are under the direct control of the computers are said to be connected **on-line.**
- These devices are designed to read information into or out of the memory unit upon command from the CPU and are considered to be part of the computer system.
- Input / Output devices attached to the computer are also called peripherals.

Classification of peripherals or external devices

- Human readable: suitable for communicating with the computer user, e.g, video display terminals (VDTs) & printers.
- Machine-readable: suitable for communicating with equipment, e.g, magnetic disks and tape system.
- Communication: suitable for communicating with remote devices, e.g, terminal, a machine-readable device.

The Input /Output Interface

- The Input /Output interface provides a method for transferring information between internal storage and external I/O devices.
- Peripherals connected to a computer need special communication links for interfacing them with the CPU.
- The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral.

The Input /Output Interface

- The major differences are:
  - Peripherals are electromagnetic and electromechanical devices and their operations are different from the operation of the CPU and the 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

The Input /Output Interface

- To resolve these differences, computer systems include special hardware component between the CPU and peripherals to supervise and synchronize all input and output transfers.
- These components are called ***interface units*** *because they interface between* the processor bus and the peripheral device.
- Each device may have its own controller that supervises the operations of the particular mechanism in the peripherals.

Functions of I/O Interface

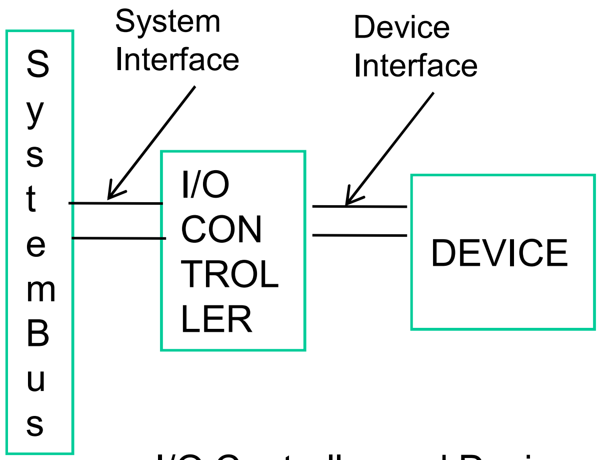
- An I/O interface is bridge between the processor and I/O devices.
- It controls the data exchange of the external devices with the main memory; or external devices and processor registers.
- Therefore, an I/O interface provides an interface internal to
    - The computer which connects it to the processor and main memory and
    - An interface external to the computer connecting it to external device or peripheral.
- The I/O interface should not only communicate the information from processor to main I/O device, but it should also coordinate these two.

Functions of I/O Interface

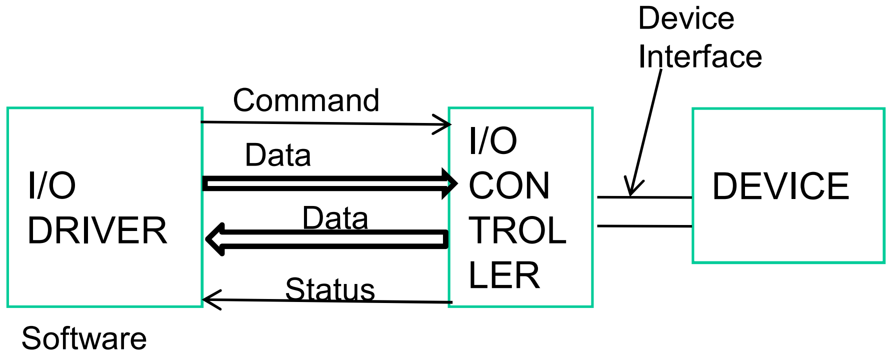
- In addition, since there are speed differences between processor and I/O devices, the I/O interface should have facilities like buffer and error detection mechanism.
- Therefore, the major functions or requirements of an I/O interface are:
  - It should be able to provide control and timing signals
  - It should communicate with the processor
  - It should communicate with the I/O device
  - It should have a provision for data buffering
  - Error detection mechanism should be in-built

I/O Controller and I/O Driver

- For using peripheral device of the computer, two modules are required:
  - A hardware module called ‘I/O controller’ that interfaces the peripheral device to the system (CPU/Memory).
  - A software module called ‘I/O driver’ that issues different commands to the I/O controller, for executing various I/O operations.



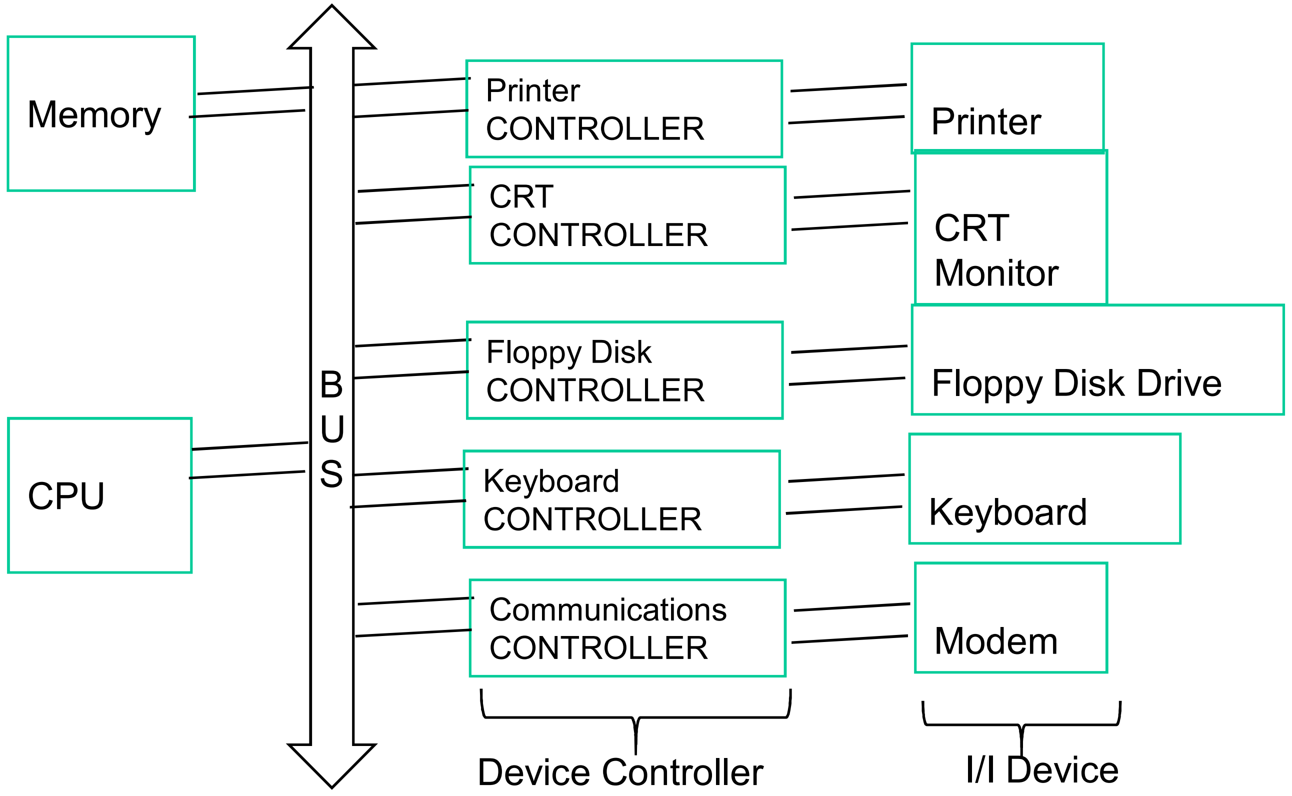
I/O Controller and Device



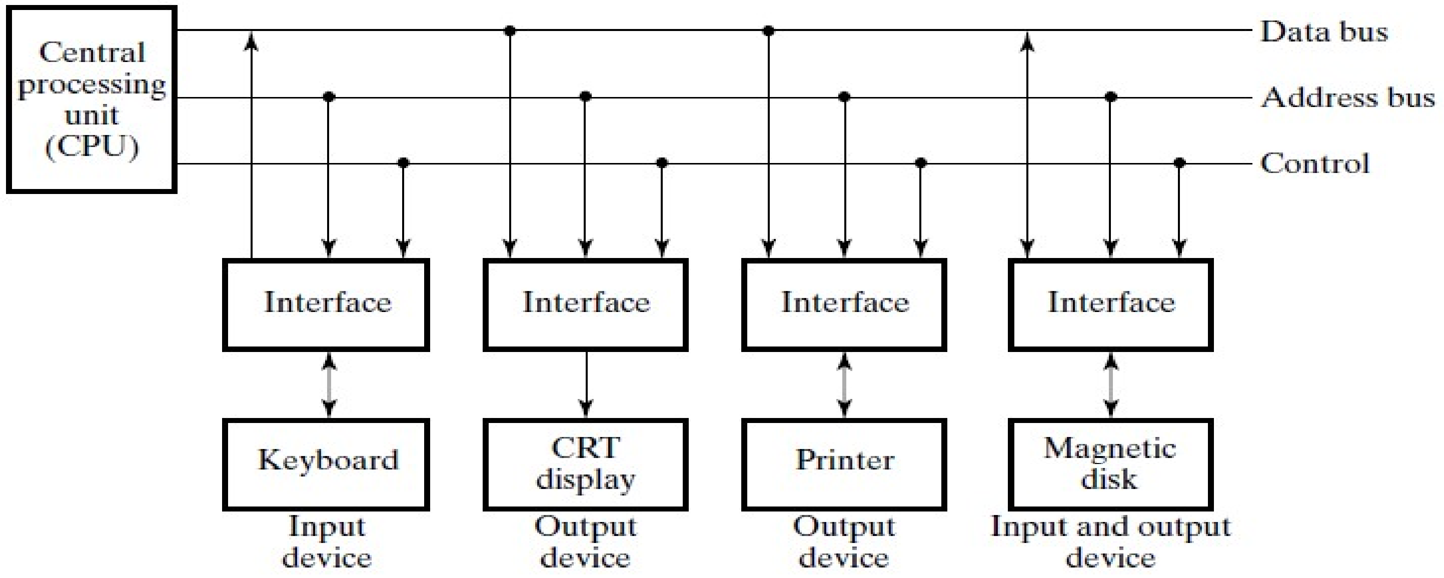
I/O driver and Device

Device Controller

- A peripheral device can be interface to the system using device controller. The linkage of the device controller to the CPU or memory is via bus.



I/O Bus And Interface Modules



Each peripheral has an interface module associated with it.
An Interface:
- 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

Device Controller

- A device controller comes in the form of an electronic circuit board that plugs directly into the system bus, and there is a cable from the controller to each device it controls.
- The cables coming out of the controller are usually terminated at the back panel of the main computer box in the form of connectors known as ports.
- Using device controllers for connecting I/O devices to a computer system instead of connecting them directly to the system bus has the following advantages:
  - A device controller can be shared among multiple I/O devices allowing many I/O devices to be connected to the system.

Device Controller

  - I/O devices can be easily upgraded or changed without any change in the computer system.
  - I/O devices of manufacturers other than the computer manufacturer can be easily plugged in to the computer system. This provides more flexibility to the users in buying I/O devices of their choice.
- The device controller is interfaced to the system bus on one side and the peripheral device on the other side. These two sides are known as system interface and device interface respectively.
- The term ‘interface’ defines the signals between two subsystems and the protocol of communication between the subsystems.
- There are two types of interface : Serial interface and parallel interface.

Device Controller Functions

- Receiving the command from the CPU
- Analyzing the command and executing it.
- Issuing appropriate control signals to the device.
- Receiving the status signals from the device and taking appropriate action.
- Transferring data from CPU/memory to the device.
- Transferring data from the device to the CPU/memory.
- Converting the format of data received from the device e.g. serial to parallel.
- Converting the format of data received from the CPU/memory e.g. parallel to serial
- Generating error checking code (parity bit) during write operation.
- Checking for error in the data received from the device.

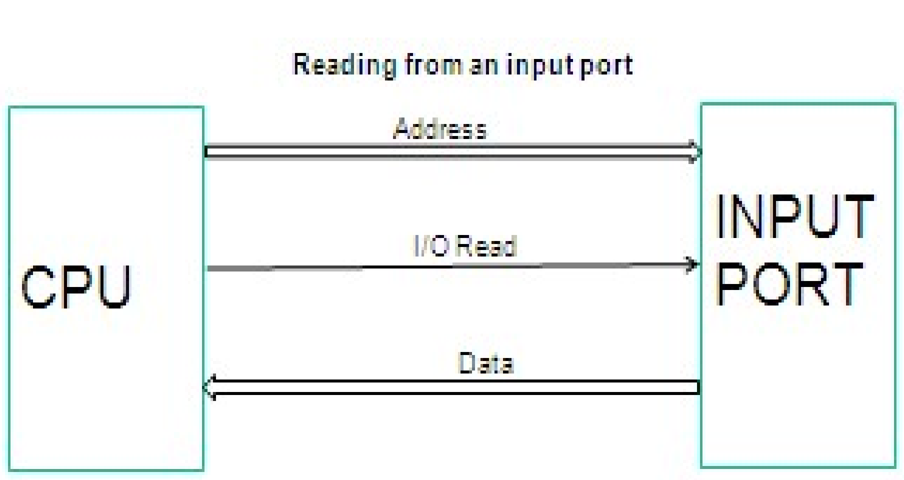
Device Controller Functions

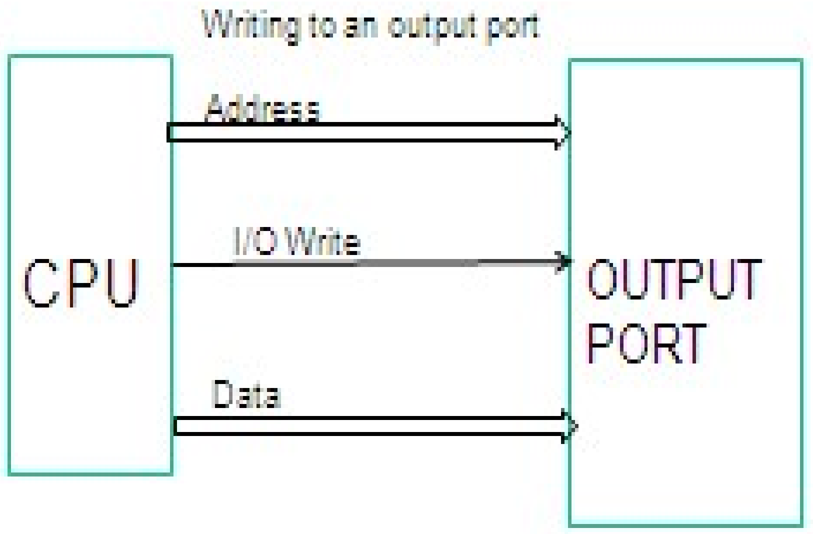
- Aborting the command execution on encountering any error.
- Retrying the command on encountering an error.
- Informing the CPU at the end of command execution.
- A device controller is designed to perform only certain number of these functions depending on the device and system.

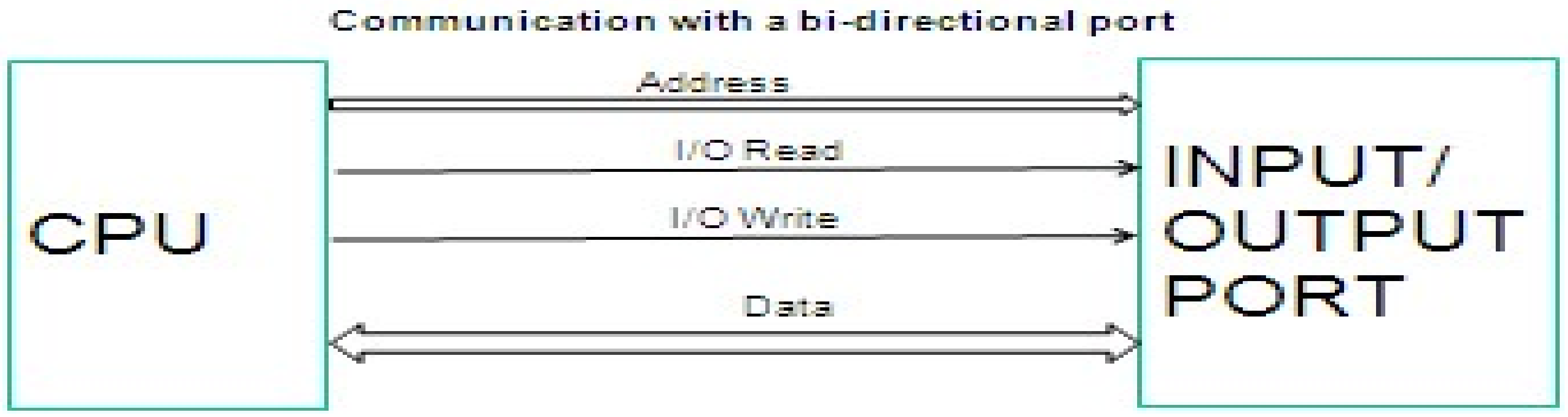
I/O Driver

- The I/O drivers are programs that performs various I/O operations by issuing the appropriate sequence of commands to the peripheral controllers I/O devices.
- Following are certain operations performed by various I/O drivers:
  - Displaying a message on CRT
  - Printing some lines by the printer
  - Reading a file from a floppy diskette
  - Displaying the contents of a memory location
  - Saving (storing) the memory contents on a hard disk.

Communication between I/O port

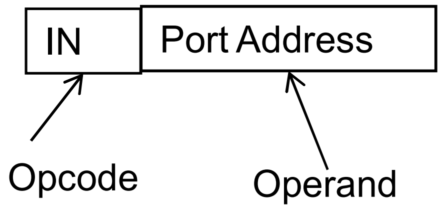






IN Instruction

- A program reads the data from an input port by using an IN instruction whose format is

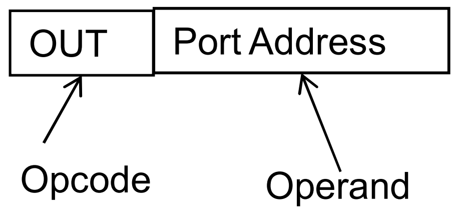


Format of IN Instruction

- The function of the IN instruction is transferring data from the input port to the CPU register. The actions performed by the CPU while executing the IN instruction are listed as follows :
  - CPU sends the port address. Only the port whose address matches, gets selected.
  - CPU transmits IOR signal. The port which is selected in step one responds to IOR signal and presents is data on the bus.
  - CPU reads the data from the bus and loads the data in the processor register.

OUT Instruction

- A program reads the data from a CPU to out port by using an OUT instruction whose format is



Format of OUT Instruction

- The function of OUT instruction is transferring data from the register to the output port. The actions performed by the CPU for the OUT instruction are listed as follows :
  - CPU sends the port address. Only the port whose address matches, gets selected.
  - CPU send data (from the accumulator) on the bus.
  - CPU sends IOW signal. The port which is selected in step one respondents to IOW signal and procures data from the bus.

Memory mapped I/O and I/O mapped I/O

- In a microprocessor system, there are two methods of interfacing input/output (I/O) devices: memory-mapped I/O and I/O mapped I/O.
- In memory-mapped I/O, input/output devices are mapped to the memory address space of the microprocessor. This means that the I/O devices are treated like memory locations and can be accessed using the same read and write instructions as memory.
- In other words, the same bus and control signals used for memory access are used for I/O access as well.

Memory mapped I/O and I/O mapped I/O

- In I/O mapped I/O, input/output devices are mapped to a separate I/O address space that is different from the memory address space.
- The microprocessor uses special instructions to access the I/O devices using specific I/O address signals, which are separate from the memory address signals.
- In the case of the 8085 microprocessor, it uses memory-mapped I/O for accessing input/output devices.
- The input/output devices are mapped to specific memory locations in the address space, and the microprocessor can access these devices using the same instructions it uses for memory access.

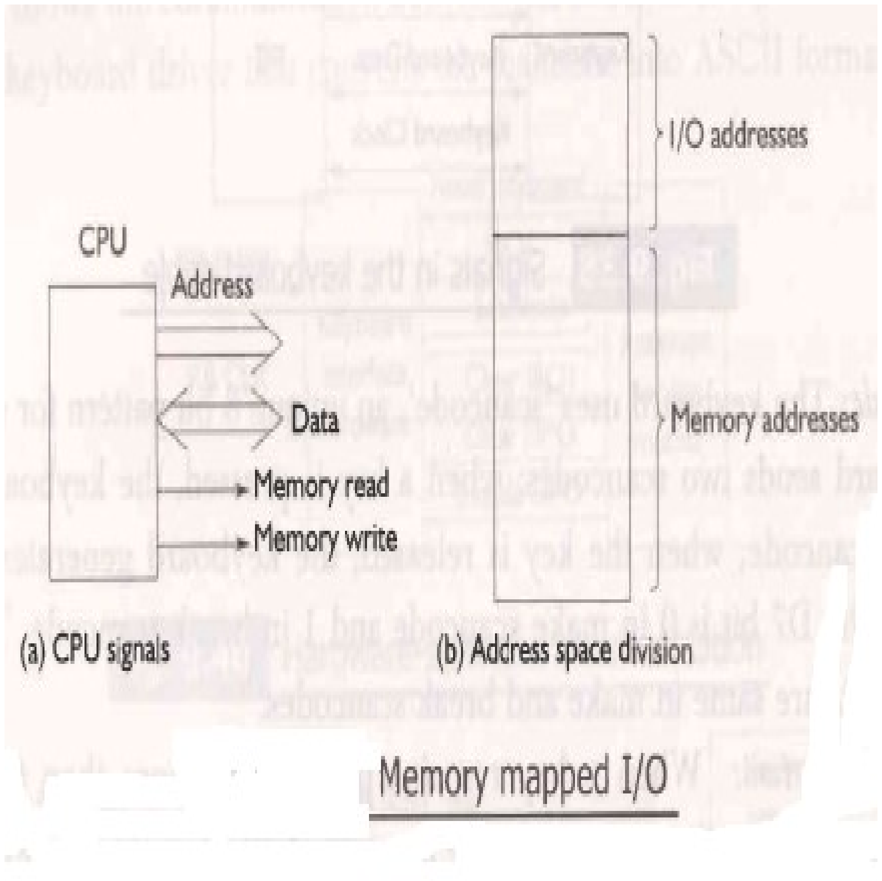
Memory mapped I/O and I/O mapped I/O

- This linking is called Interfacing. The interfacing of the I/O devices in 8085 can be done in two ways :
- **1. Memory-Mapped I/O Interfacing :** In this kind of interfacing, we assign a memory address that can be used in the same manner as we use a normal memory location.
- **2. I/O Mapped I/O Interfacing :** A kind of interfacing in which we assign an 8-bit address value to the input/output devices which can be accessed using IN and OUT instruction is called I/O Mapped I/O Interfacing.

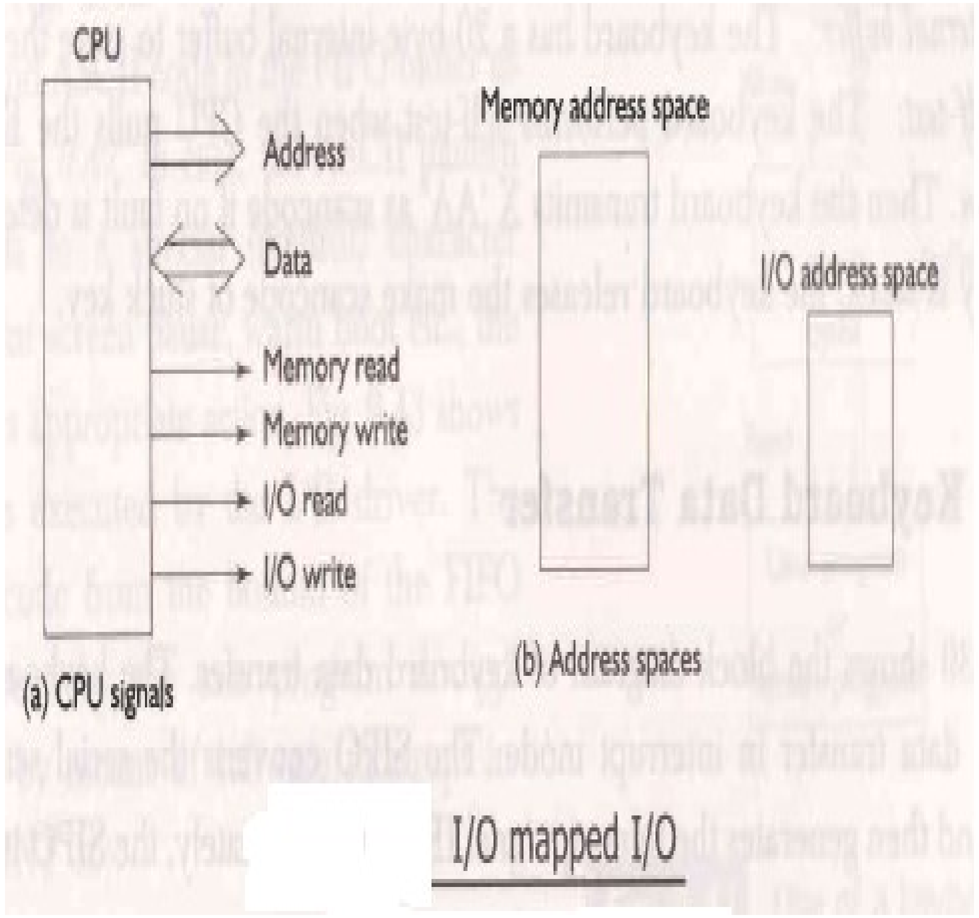
Memory mapped I/O and I/O mapped I/O

Memory mapped I/O	**I/O mapped I/O**
Each port is treated as a memory location	Each port is treated as an independent unit
CPU' s memory address space is divided between memory and I/O port	Separate address spaces for memory and I/O ports
Extra circuit is needed for address space decoding to select memory or I/O ports	Extra signals (IOR and IOW) differentiate between memory and I/O ports.
Single instruction can transfer data between memory and port	Two instructions are necessary to transfer data between memory and port
Data transfer is by means of instructions like MOVE	Each port can be accessed by means of IN or OUT instructions
Microprocessor is not aware whether it is accessing a memory location or a port.	Microprocessor is aware whether it is accessing a memory location or a port.

Memory mapped I/O and I/O mapped I/O



Memory mapped I/O



I/O mapped I/O

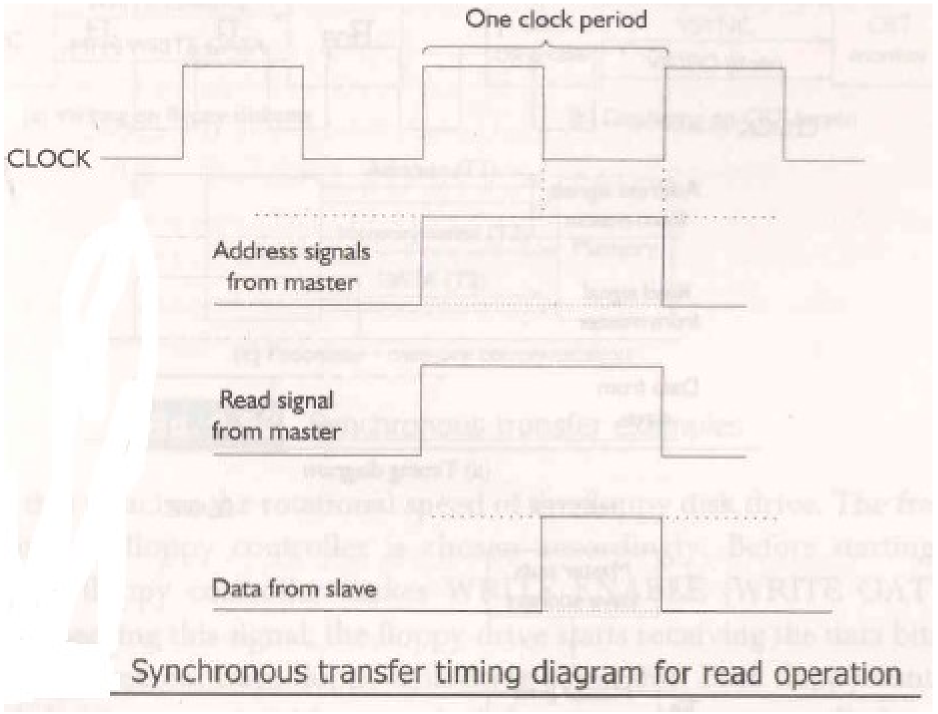
Asynchronous and Synchronous Transfer

- When two units have to communicate with each other for data transfer, usually one of them is the master and the other is the slave.
- One word of data transfer is done in one clock period or sometimes in more than one clock period.
- There are two types of data transfer depending on the mechanism of timing the data : Synchronous and Asynchronous.
  - The synchronous transfer is possible between two units each of which knows the behavior of the other.
  - The asynchronous transfer enables communication between two unknown types of units.

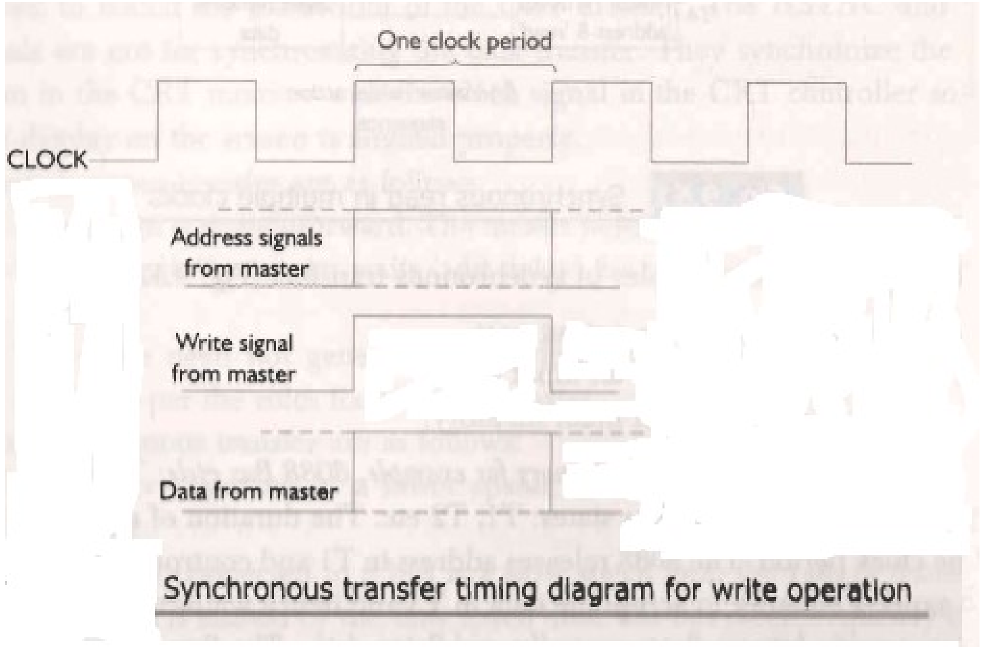
Synchronous Transfer

- In synchronous method of transfer, the sending and receiving unit are supplied with the same clock signal.
- The master performs a sequence of actions for data transfer in a predetermined order; each action is synchronized to either the raising edge or falling edge of the clock.
- The master is designed to supply the data at a time when the slave is definitely ready for it. It also provide sufficient delay for slow slaves.
- The master places following three different items on the raising edge of the clock: Slave address, Data and Write signal.

Synchronous Transfer



Synchronous transfer timing diagram for read operation

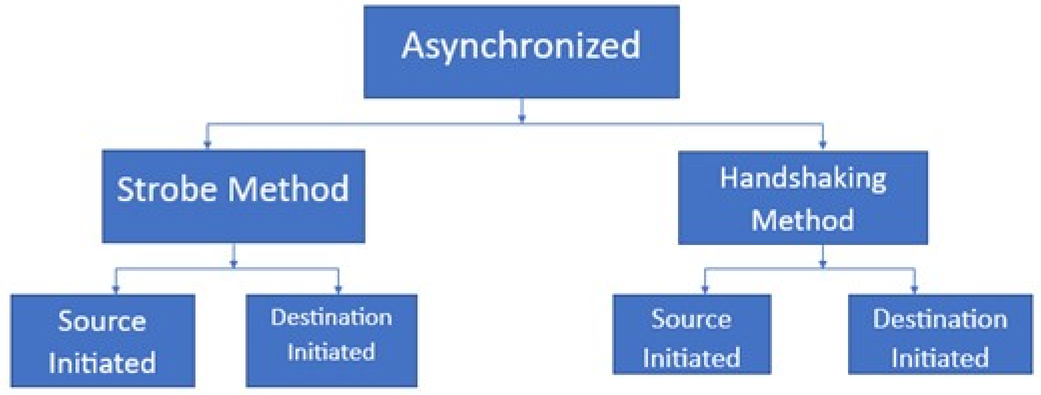


Synchronous transfer timing diagram for write operation

Asynchronous Transfer

- In asynchronous transfer, there is no common clock between the master and slave.
- Asynchronous data transfer enables computers to send and receive data without having to wait for a real-time response.
- They follow some sort of acknowledge protocol during data transfer sequence.
- When master sends data, it informs the slave by means of ACCEPT STROBE signal indicating that it has placed data on the bus.
- In response, the slave receives the data from the bus.

Asynchronous Data Transfer



Asynchronous Transfer

- The source takes care of proper timing delay between the actual data signals and the ACCEPT STROBE. It places the data first, and after some delay, generates the STROBE.
- Similarly, before removing the data it removes the STROBE and after some delay it removes the data.
- A destination unit can also initiate the data transfer by sending REQUEST STROBE to the sending unit
- Handshaking
  - A control signal is accompanied with each data being transmitted to indicate the presence of data
  - The receiving unit responds with another control signal to acknowledge receipt of the data

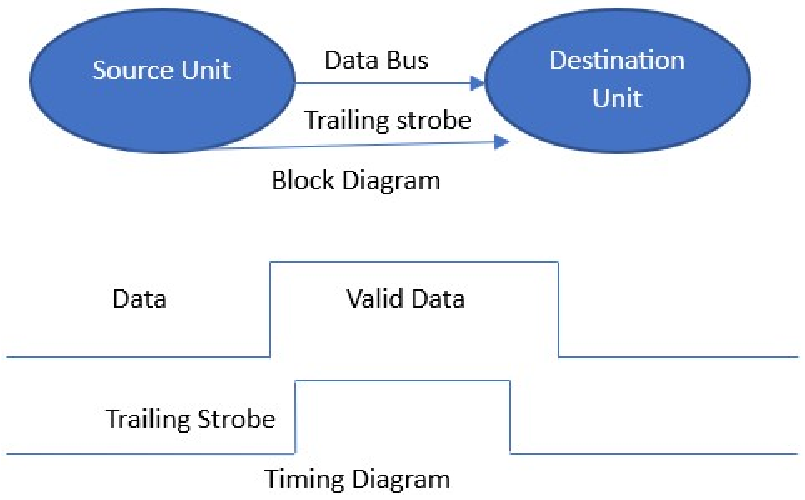
Asynchronous Transfer (Handshaking)

- Strobe Methods
  - Source-Initiated{ The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received data
  - Destination-Initiated: The destination unit that initiates the transfer has no way of knowing whether the source has actually placed the data on the bus
- Handshaking
  - A control signal is accompanied with each data being transmitted to indicate the presence of data
  - The receiving unit responds with another control signal to acknowledge receipt of the data
  - To solve this problem, the *HANDSHAKE* method introduces a second control signal to provide a *Reply* to the unit that initiates the transfer

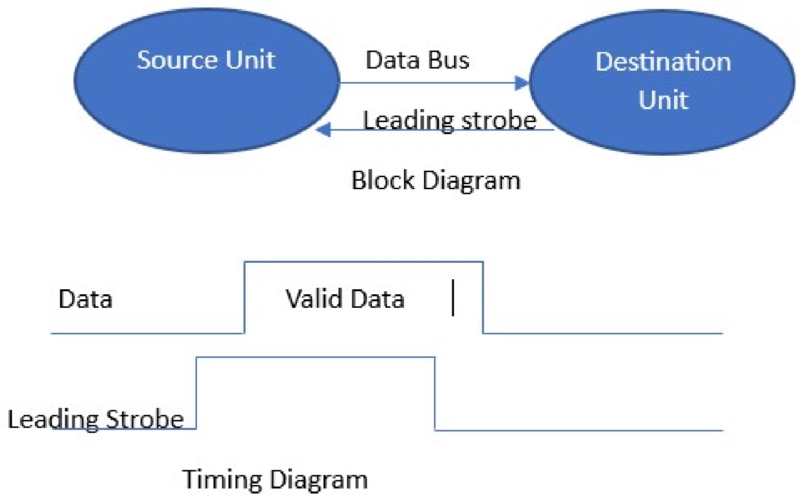
STROBE CONTROL

- Employs a single control line to time each transfer
- The strobe may be activated by either the source or the destination unit

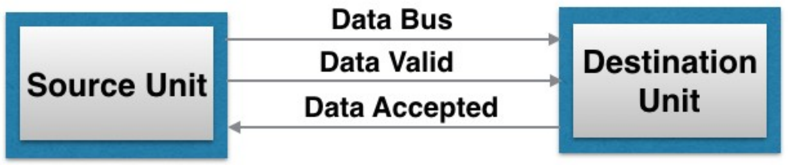
Source Initiated Strobe



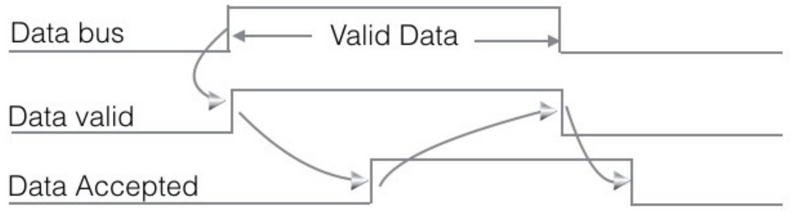
Destination Initiated Strobe



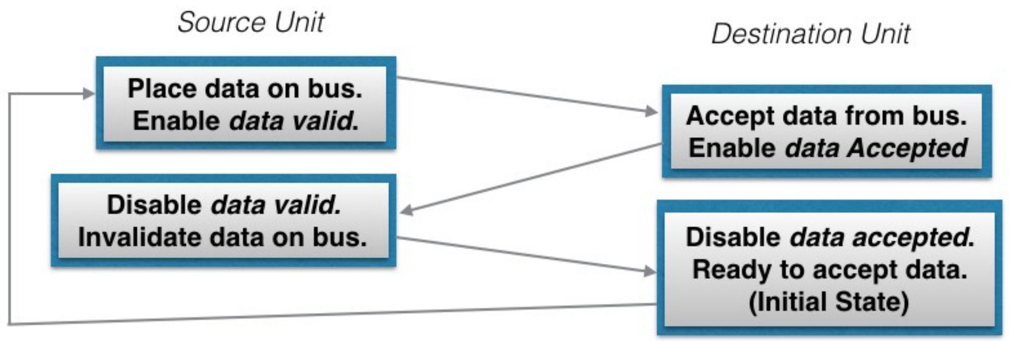
SOURCE-INITIATED TRANSFER USING HANDSHAKE



a) Block Diagram



b) Timing Diagram



Place data on bus. Enable *data valid*.

Accept data from bus. Enable *data Accepted*

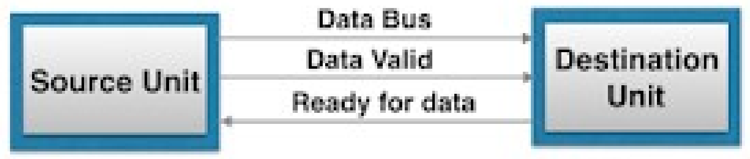
Disable *data valid*. Invalidate data on bus.

Disable *data accepted*. Ready to accept data. (Initial State)

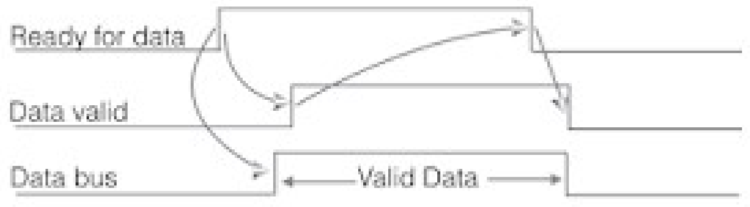
c) Sequence Diagram(Sequence of events)

Allows arbitrary delays from one state to the next, Permits each unit to respond at its own data transfer rate, The rate of transfer is determined by the slower unit

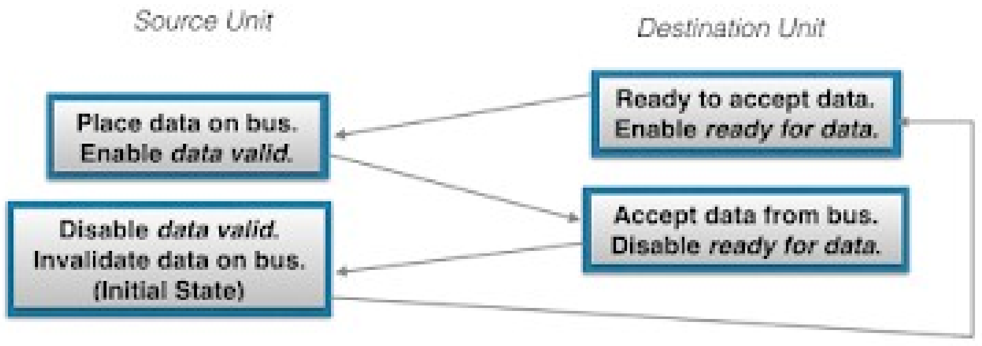
DESTINATION-INITIATED TRANSFER USING HANDSHAKE



a) Block Diagram



b) Timing Diagram


Source Unit
Destination Unit
Ready to accept data. Enable *ready for data.*
Place data on bus. Enable *data valid.*
Accept data from bus. Disable *ready for data.*
Disable *data valid.* Invalidate data on bus. (Initial State)

c) Sequence Diagram(sequence of events)

* Handshaking provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units

* If one unit is faulty, data transfer will not be completed

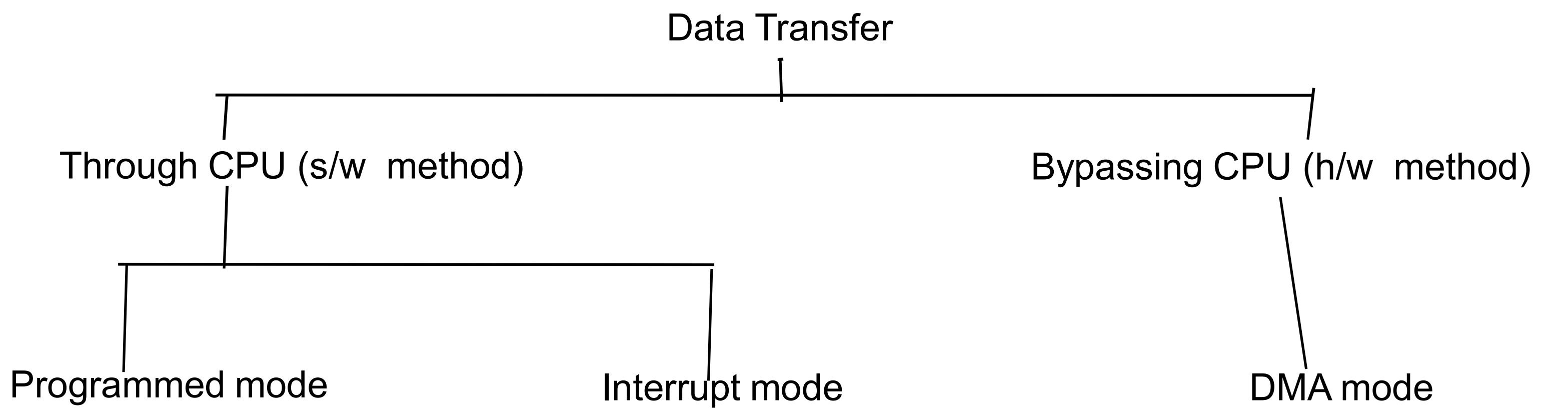
⇒ Can be detected by means of a *timeout* mechanism

I/O Techniques in PC

- Any communication between the user and the computer is either a program or some type of data for a program.
- The term 'Data Transfer' refers to the moving of information between the CPU/memory and the I/O devices/peripherals.
- The data transfer is usually done in steps of one word at a time.
- The following are the two types of data transfer:
  - From an input device to the memory.
  - From the memory to an output device.
- There are two different techniques of performing data transfer :
  - Passing the data through the CPU (s/w method)
  - Bypassing the CPU (h/w method)
- The main difference between the two method is the extent of involvement of CPU in the actual data transfer operation.

I/O Techniques in PC

- In the software method, the tasks related to I/O operations are implemented as a program/routine which is executed by the CPU. Hence , the CPU is the total incharge of the I/O operations.
- In the hardware method, the program delegates the responsibility of performing I/O operations to another hardware unit called DMA (direct memory access) controller.



I/O Techniques in PC

- There are two steps in the software method.
- For example, to move a byte of data from an input device, the following two steps are performed:
  1. Read the data byte from the input device to the CPU
  2. Move the data byte from the CPU to the memory location
- For example, to move a byte of data to output device, the following two steps are performed:
  1. Move the data byte from memory to the CPU
  2. Read the data byte from the CPU to output device
- In the hardware method, the CPU software is not involved in actual transfer of data bytes.
- The software only provides certain parameters initially to the hardware and the hardware performs the actual transfer of data bytes without involving CPU.

I/O Techniques in PC

- In programmed I/O, the I/O operations are completely controlled by the processor.
- The processor executes a program that initiates, directs and terminate an I/O operation.
- It requires a little special I/O hardware, but is quite time consuming for the processor since the processor has to wait for slower I/O operations to complete.
- With interrupt driven I/O, 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 processor stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing which results in the waiting time by the processor being reduced.

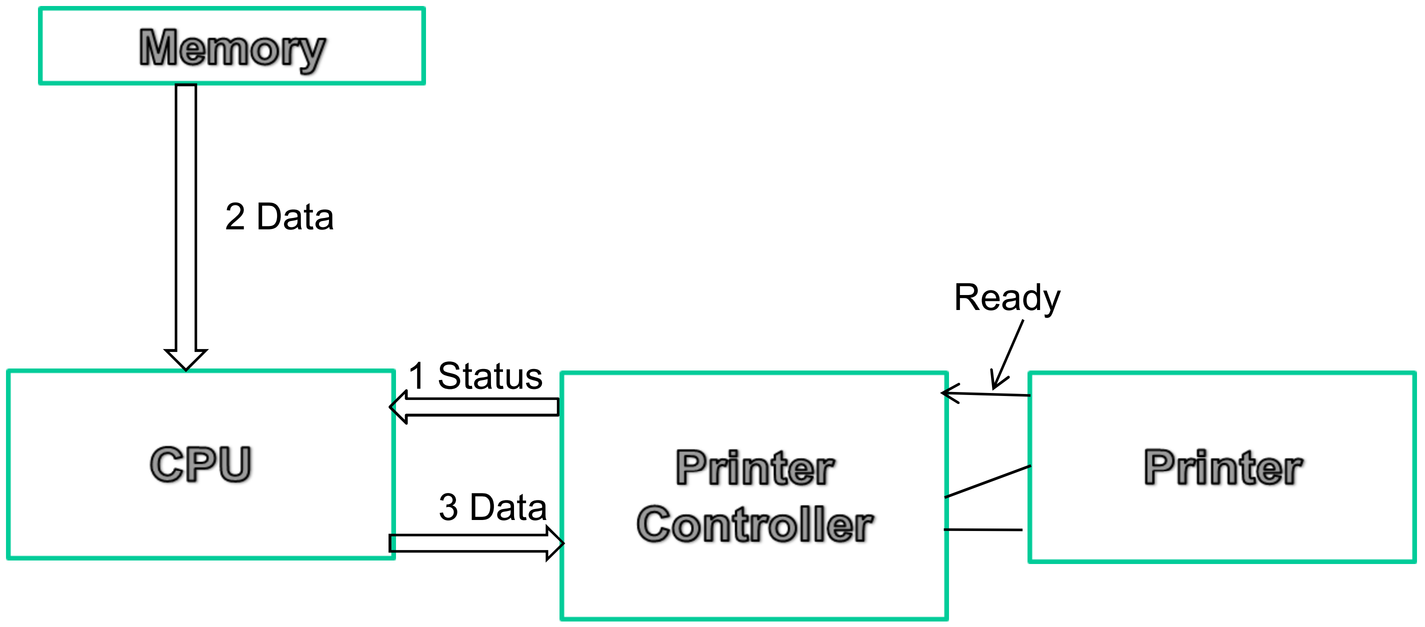
I/O Techniques in PC

- With both programmed and interrupt-driven I/O, the processor is responsible for extracting data from the main memory for output and storing data in the main memory during input.
- What about having an alternative where I/O device may directly store data or retrieve data from memory?
- This alternative is known as direct memory access (DMA).
- In this mode, the I/O interface and main memory exchange data directly, without the involvement of processor.

Programmed Mode

- It performs four distinct activities for each and every data byte transferred:
  1. Reading the status of the peripheral device.
  2. Analyzing whether the device is ready for data transfer or not
  3. If the device is ready, going to step (4) for actual data transfer; if the device is not ready going to step (1) in order to loop on till the device is ready for data transfer.
  4. Performing the data transfer in two steps. For an input operation, the two steps are as follows:
     1. Reading the data from the input device into the CPU
     2. Storing the data in a memory location
  5. For an output operation, the two steps are as follows:
     1. Loading the data from the memory to the CPU
     2. Issuing the data to the output device.

Programmed mode block diagram



Steps : 1. Reading device status 2. Moving data from the memory 3. Sending data to device

All three steps are repeated for each byte. After step 1, step 2 is done if device is ready; otherwise step 1 is repeated.

Programmed Mode

- Programmed input/output is a useful I/O method for computers where hardware costs need to be minimized.
- The input or output operation in such cases may involve:
  - Transfer of data from I/O device to the processor registers.
  - Transfer of data from processor registers to memory.
- With the programmed I/O method, the responsibility of the processor is to constantly check the status of the I/O device to check whether it is free or it has finished inputting the data.
- Thus, this method is very time consuming where the processor wastes a lot of time in checking and verifying the status of an I/O device.

I/O Commands

- There are four types of I/O commands that an I/O interface may receive when it is addressed by a processor:
  - **Control:** These commands are device specific and are used to provide specific instructions to the device, e.g. a magnetic tape requiring rewinding and moving forward by a block.
  - **Test:** This command checks the status such as if a device is ready or not or is in error condition.
  - **Read:** This command is useful for input of data from input device.
  - **Write:** this command is used for output of data to output device.

Interrupt-Driven Input/Output

- The problem with programmed I/O is that the processor has to wait a long time for the I/O interface to see whether a device is free or wait till the completion of I/O.
- The result is that the performance of the processor goes down tremendously. What is the solution?
- What about the processor going back to do other useful work without waiting for the I/O device to complete or get freed up?
- But how will the processor be intimated about the completion of I/O or a device is ready for I/O?
- A well-designed mechanism was conceived for this, which is referred to as interrupt-driven I/O.
- In this mechanism, provision of interruption of processor work, once the device has finished the I/O or when it is ready for the I/O, has been provided.

Interrupt Mode

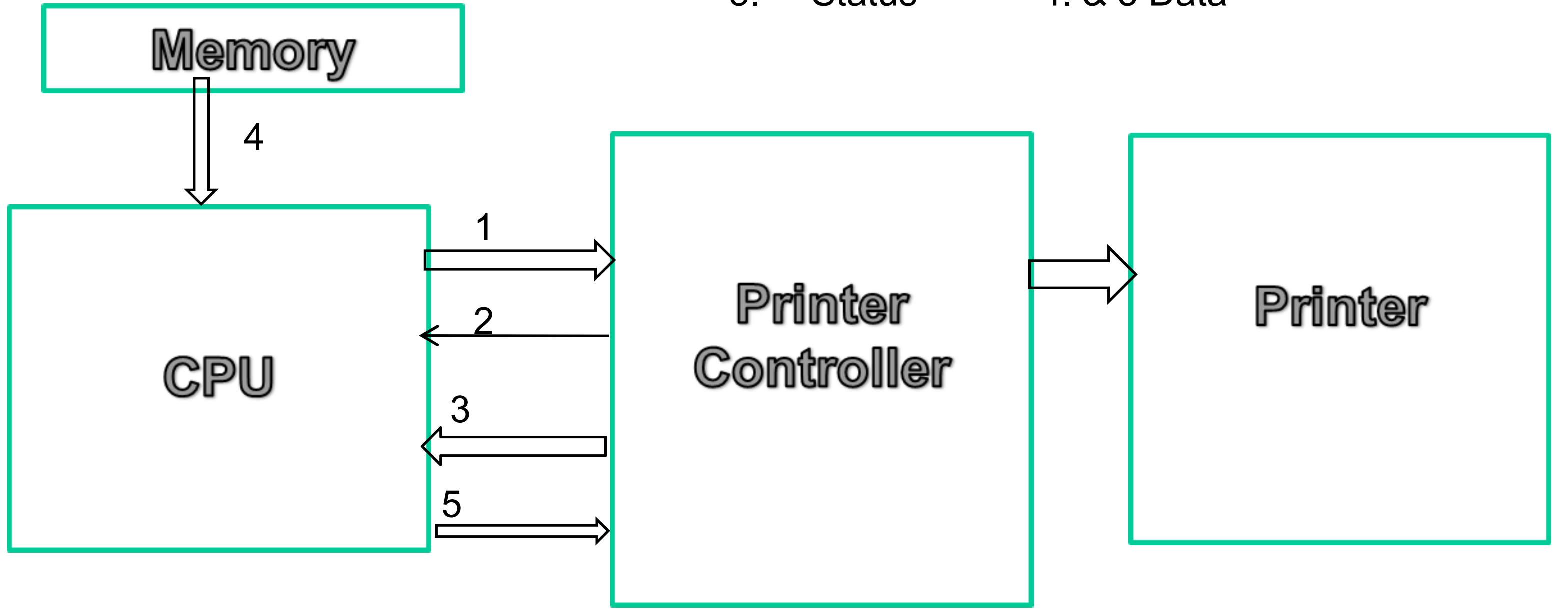
- In programmed mode device status is monitored by s/w (I/O routine).
- In interrupt mode s/w (I/O routine) does not wait until the device is ready.
- Instead the device controller hardware continuously monitors the device status and raise an interrupt to the CPU as soon as the device is ready for data transfer.
- The I/O routine s/w gets hold of the CPU immediately.
- After completing one byte of data transfer, the I/O routine releases the CPU which is free to perform any other program or routine.
- When the next interrupt is generated, again the I/O routine gains control.

Interrupt Mode

- In the interrupt mode, the s/w (CPU) performs data transfer but is not involved in checking whether the device is ready for data transfer or not.
- This leads to better utilization of CPU.
- The CPU can execute some other program until the interrupt is received from the device.
- There device controller should have some additional intelligence for checking device status and raising an interrupt whenever data transfer is required. This result in extra h/w circuitry in the controller.

Interrupt mode block diagram

1. Command 2. Interrupt
3. Status 4. & 5 Data

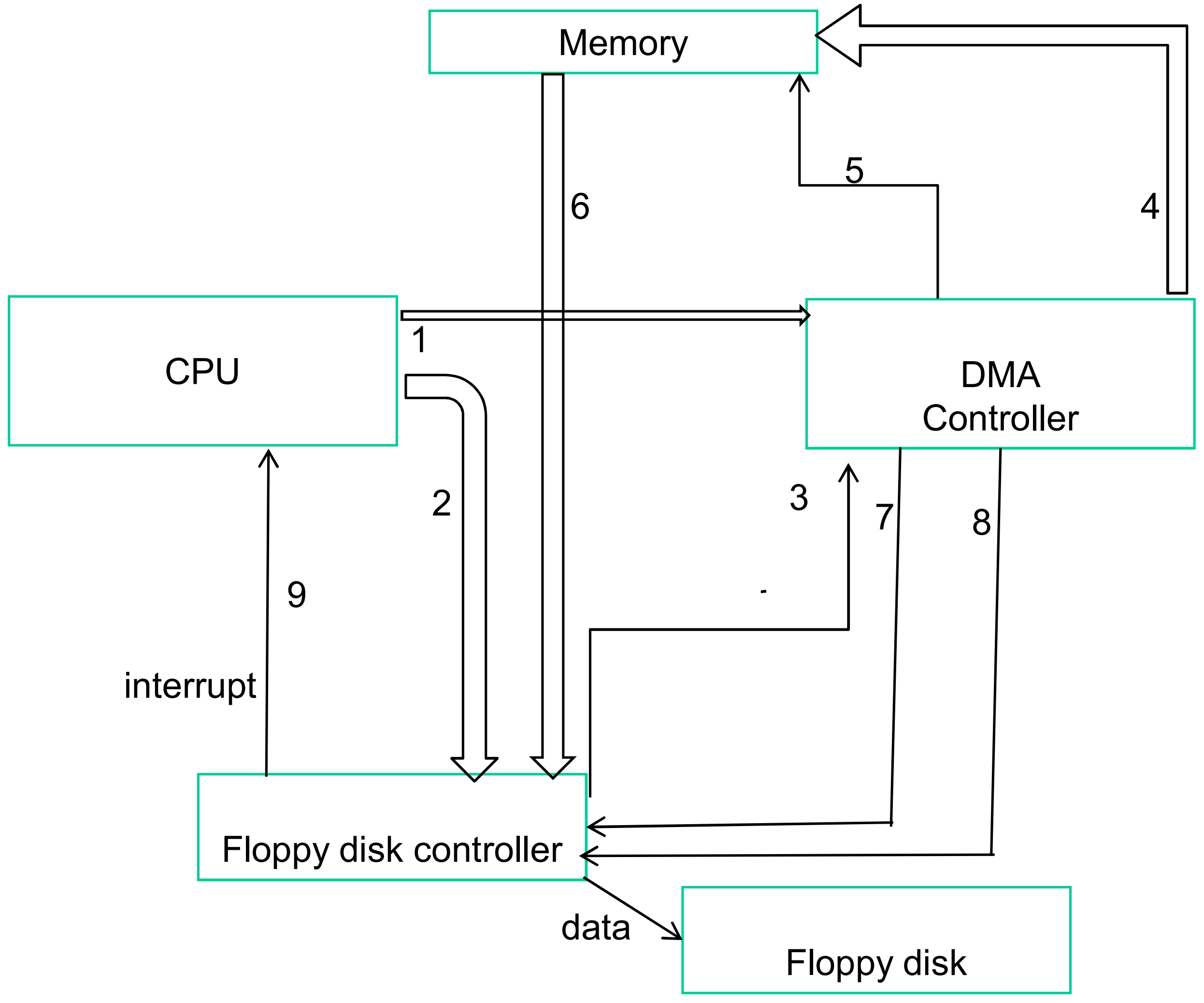


Steps : 1. Giving command
2. Servicing interrupt
3. Reading status
4. Moving data from memory
5. Sending data to device

DMA (Direct Memory Access)

- In both interrupt-driven and programmed I/O, the processor is busy with executing input/output instructions and the I/O transfer rate is limited by the speed with which the processor can test and service a device.
- What about a technique that requires minimal intervention of the CPU for input/output?
- These two types of drawbacks can be overcome with a more efficient technique known as DMA, which acts as if it has taken over control from the processor.
- It is used primarily when a large amount of data is to be transferred from the I/O device to the Memory.

DMA mode block diagram



1. DMA parameters
2. Write command
3. Data request
4. Address
5. Read signal
6. Data
7. Data acknowledgement
8. Last byte signal
9. Interrupt

DMA Function

- Although the CPU intervention in DMA is minimized, yet it must use the path between interfaces that is the system bus.
- The s/w provides the following DMA parameter to the DMA controller (i) Memory start address (ii) byte count (iii) Direction : Input or output
- Thus, DMA involves an additional interface on the system bus.
- A technique called cycle stealing allows the DMA interface to transfer one data word at a time, after which it must return control of the bus to the processor.
- When an I/O is requested, the processor issues a command to the DMA interface by sending to the DMA interface the following information
  - Which operations (read or write) to be performed, using the read or write control lines.

DMA Function

  - The address of I/O devices, which is to be used, communicated on the data lines.
  - The starting location on the memory where the information will be read or written to be communicated on the data lines and is stored by the DMA interface in its address register.
  - The number of words to be read or written is communicated on the data lines and is stored in the data count register.
- The DMA interface transfers the entire block of data, one word at a time, directly to or from memory, without going through the processor.
- When the transfer is complete, the DMA interface sends an interrupt signal to the processor.
- Thus, in DMA the processor involvement can be restricted at the beginning and end of the transfer.