



COMPUTER ORGANISATION AND ARCHITECTURE **24BTPHY105**

Dr. Chhaya Suryabhan Dule
Associate Professor
Dept of CSE
SOET,SNPSU

MODULE 2

SYLLABUS

- ? **Input-output organization** : Input/output interface ,input bus and interface module ,Isolated versus Memory mapped I/O ,Example of I/O interface – Asynchronous data transfer-strobe control,Handshake –Modes of transfer- Examples of programmed I/O ,Interrupt initiated I/O, Priority Interrupt –Daisy chain ,Direct Memory access
- ? **Memory Organisation** : Memory Hierarchy ,Main Memory RAM and ROM chips,Auxilliary Memory ,Magnetic Disc,Magnetic Tape –Associate memory-Basic Defination-Cache Memory Asssociated Mapping,Direct Mapping,Set associative mapping



MODULE 2

INPUT-OUTPUT ORGANIZATION

INTRODUCTION

- ? **Input-output subsystem in computer architecture facilitates communication between (CPU and Memory) central system and external devices enabling data transfer and interaction with outside world**
- ? **Program and data must be entered into computer memory for processing and results obtained from computation must be recorded or displayed for user**



MODULE 2

INPUT-OUTPUT ORGANIZATION

INTRODUCTION

- ? The I/O subsystem is a crucial part of a computer system that handles all input and output operations
- ? It allows the CPU to interact with peripheral devices like keyboard ,mouse , printers and monitors
- ? It manages the flow of data between the computers internal components and external devices
- ? It is responsible for converting signals between the CPU and peripheral devices to ensure proper communications



PERIPHERAL DEVICES

- ? Input or output devices that are connected to computer are called peripheral devices.**
- ? These devices are designed to read information into or out of the memory unit upon command from the CPU and are considered to be the part of computer system. These devices are also called peripherals.**
- ? For example: Keyboards, display units and printers are common peripheral devices.**



THERE ARE THREE TYPES OF PERIPHERALS

- ? **1.Input peripherals:** Allows user input, from the outside world to the computer. Example: Keyboard, Mouse etc.
- ? **2.Output peripherals:** Allows information output, from the computer to the outside world. Example: Printer, Monitor etc
- ? **3.Input-Output peripherals:** Allows both input (from outside world to computer) as well as, output (from computer to the outside world).
 - ? Example: Touch screen etc.



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



INTERFACES

Interface is a shared boundary between two separate components of the computer system which can be used to attach two or more components to the system for communication purposes.

There are two types of interfaces:

- ? 1. CPU Interface
- ? 2. I/O Interface



INPUT-OUTPUT INTERFACE

- ? **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 central processing unit.**



INPUT-OUTPUT INTERFACE

- ? **USB (Universal Serial Bus):** A common interface for connecting peripherals like keyboards, mouse, printers, and external storage devices.
- ? **HDMI (High-Definition Multimedia Interface):** Used for connecting video and audio devices.
- ? **PCI (Peripheral Component Interconnect):** Used for connecting graphics cards and other high-speed devices.
- ? **Serial Interfaces:** Transmit data as a stream of bits, used for connecting devices like mouse and modems.
- ? **Parallel Interfaces:** Transmit multiple bits of data simultaneously, used for older printers and scanners.
- ? **Ethernet:** Used for connecting computers to a network.

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. Therefore, a conversion of signal values may be required.**
- ? 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**



Input bus & Interface Module:

? A typical communication link between the processor and several peripherals is employed by I/O bus consist of

data lines

address lines

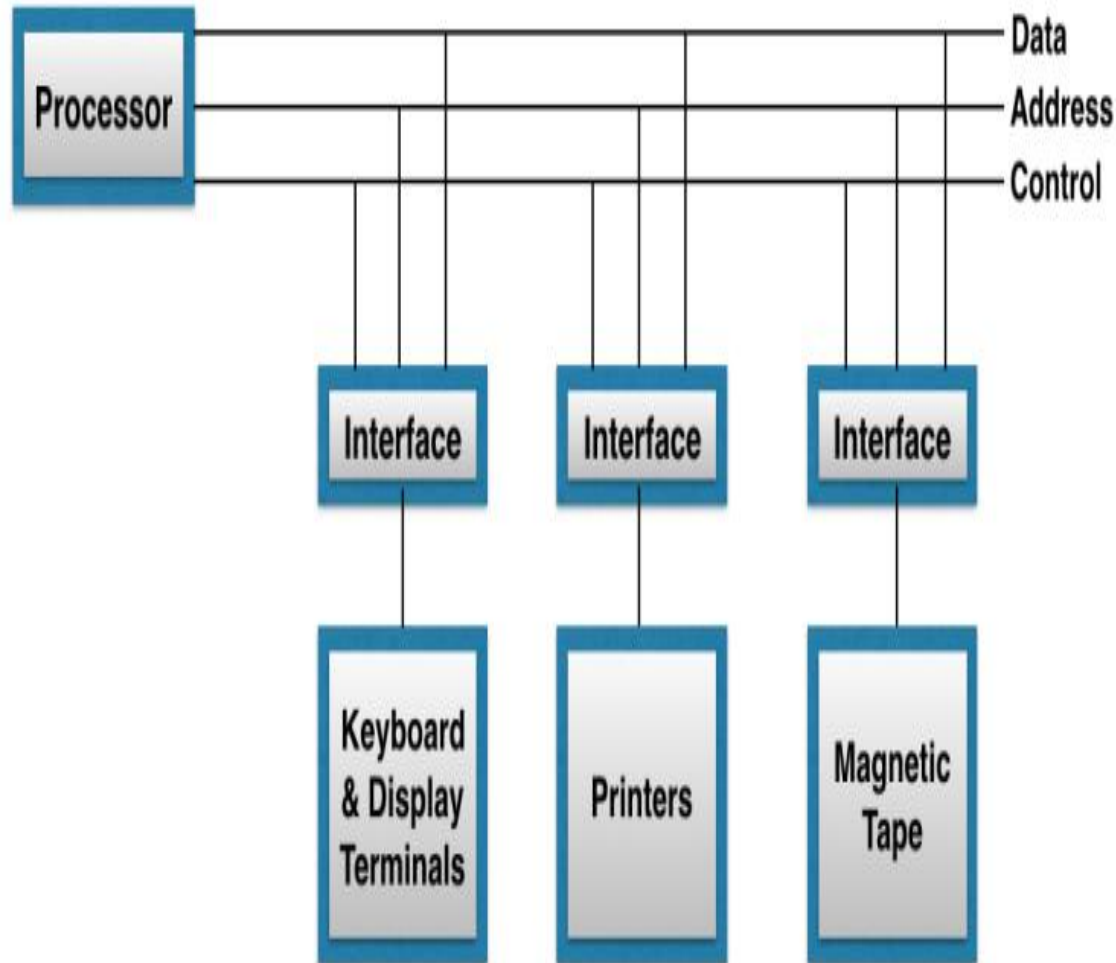
control lines.

The magnetic disk, printer, and terminal are employed in general-purpose computer.

Magnetic Tape and Magnetic Disk are used for Backup storage



FIGURE 2.1 - CONNECTION OF I/O BUS TO INPUT-OUTPUT DEVICES



Input bus & Interface Module:

- ? Each peripheral device has an interface unit
- ? Interface decodes address and control received from I/O bus , interface them for peripherals and provide signal for **peripheral controller**.
- ? It synchronizes data flow and supervise the data transfer between peripherals and processor.
- ? Each peripheral has its own controller that operates particular electromechanical devices
 - ? Ex: printer controller – controls paper motion, print timing ,selection of printing character
 - ? Controller can be separate or integrated with interface



Input bus & Interface Module:

To communicate with a particular device

- ? Processor places a device address on address lines
- ? Each interface attached to I/O bus contains an address decoder that monitors address lines.
- ? When Interface detects its own address ,it activates path between bus lines and device that it controls
- ? Other peripherals whose address does not corresponds to address in bus are disabled by their interface



Input bus & Interface Module:

To communicate with a particular device

- ? Address is made available in address lines and processor provide function code in control lines
- ? The selected interface responds to function code and proceed to execute it
- ? Function code is referred as I/O command executed in the interface attached to peripheral devices



Input bus & Interface Module:

Four types of commands an interface may receive

- ? **Control Command** : control command issued to activate the peripheral and to inform it what to do ?
- ? **Status Command** : Test the various status conditions in the interface and peripherals
- ? **Data Output Command** : causes interface to respond by transferring data from bus into its registers
- ? **Data Input Command** : Interface receives an item from peripheral and places in its buffer register



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.



ISOLATED I/O VERSUS MEMORY-MAPPED I/O

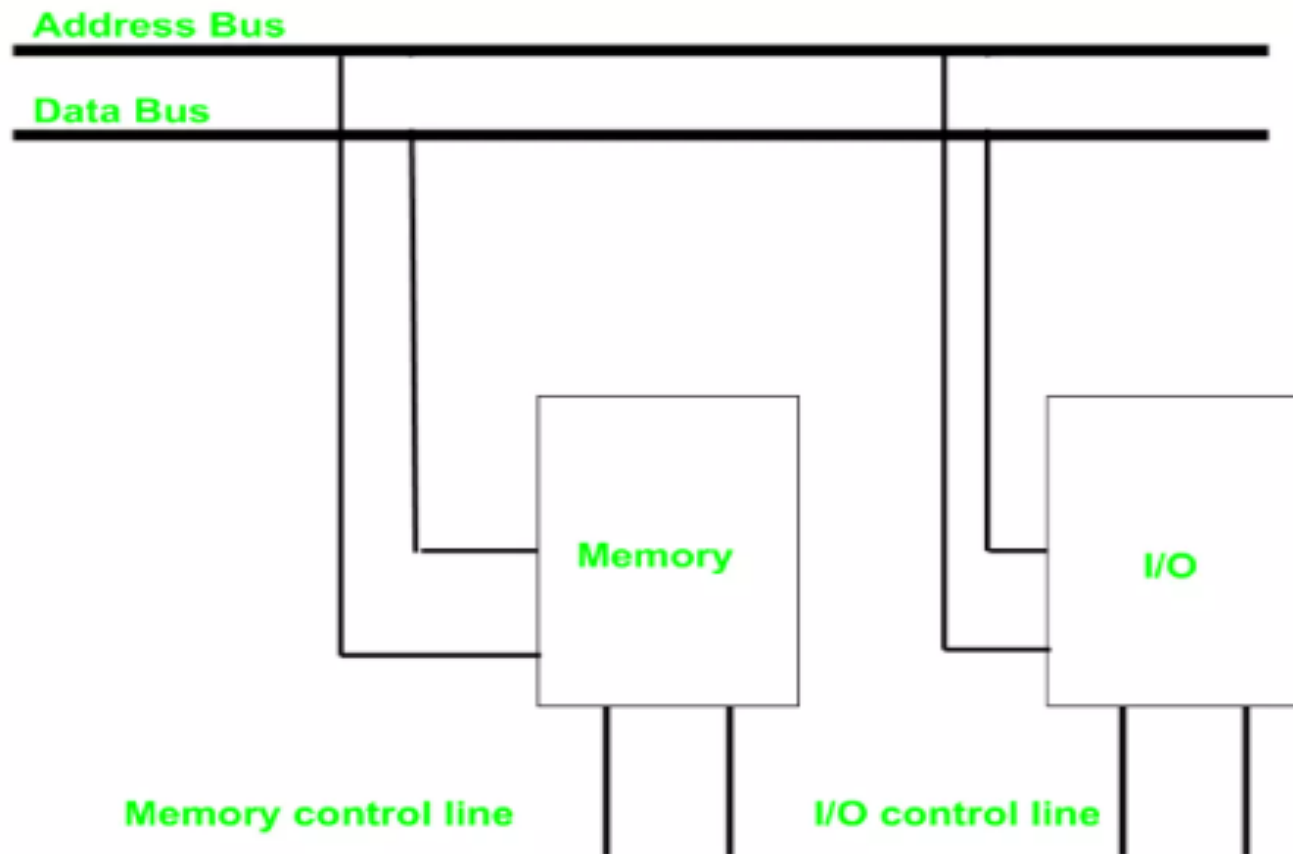
Many computers use one common bus to transfer information between memory or I/O and the CPU.

Isolated Memory Mapped I/O :

- ❖ Memory transfer and I/O transfer is made through separate read and write lines (control lines are separate for memory and I/O device)
- ❖ CPU specifies whether address on address line is for memory word or for interface register by enabling one of the read write control lines
- ❖ I/O read and write enabled during I/O transfer
- ❖ Memory read and write is enabled during memory transfer



ISOLATED I/O VERSUS MEMORY-MAPPED I/O



ISOLATED I/O VERSUS MEMORY-MAPPED I/O

Memory Mapped I/O :

- ❖ Memory transfer and I/O transfer is made through common read and write lines (control lines are same for memory and I/O device)
- ❖ CPU specifies only one set of read and write signals and do not distinguish between I/O and memory address

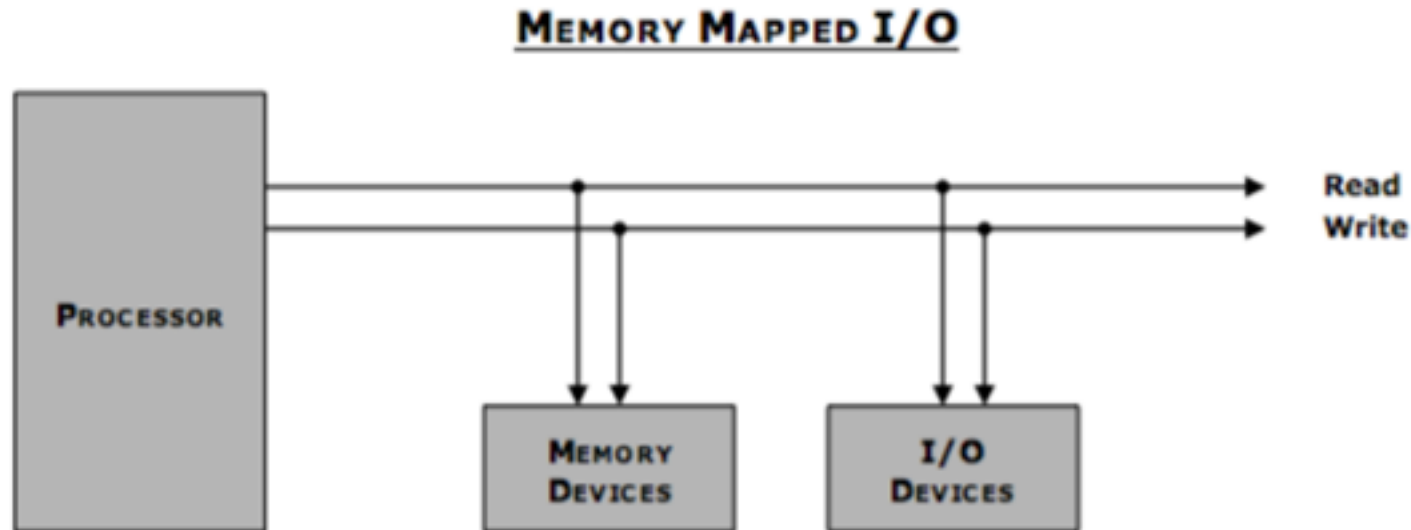
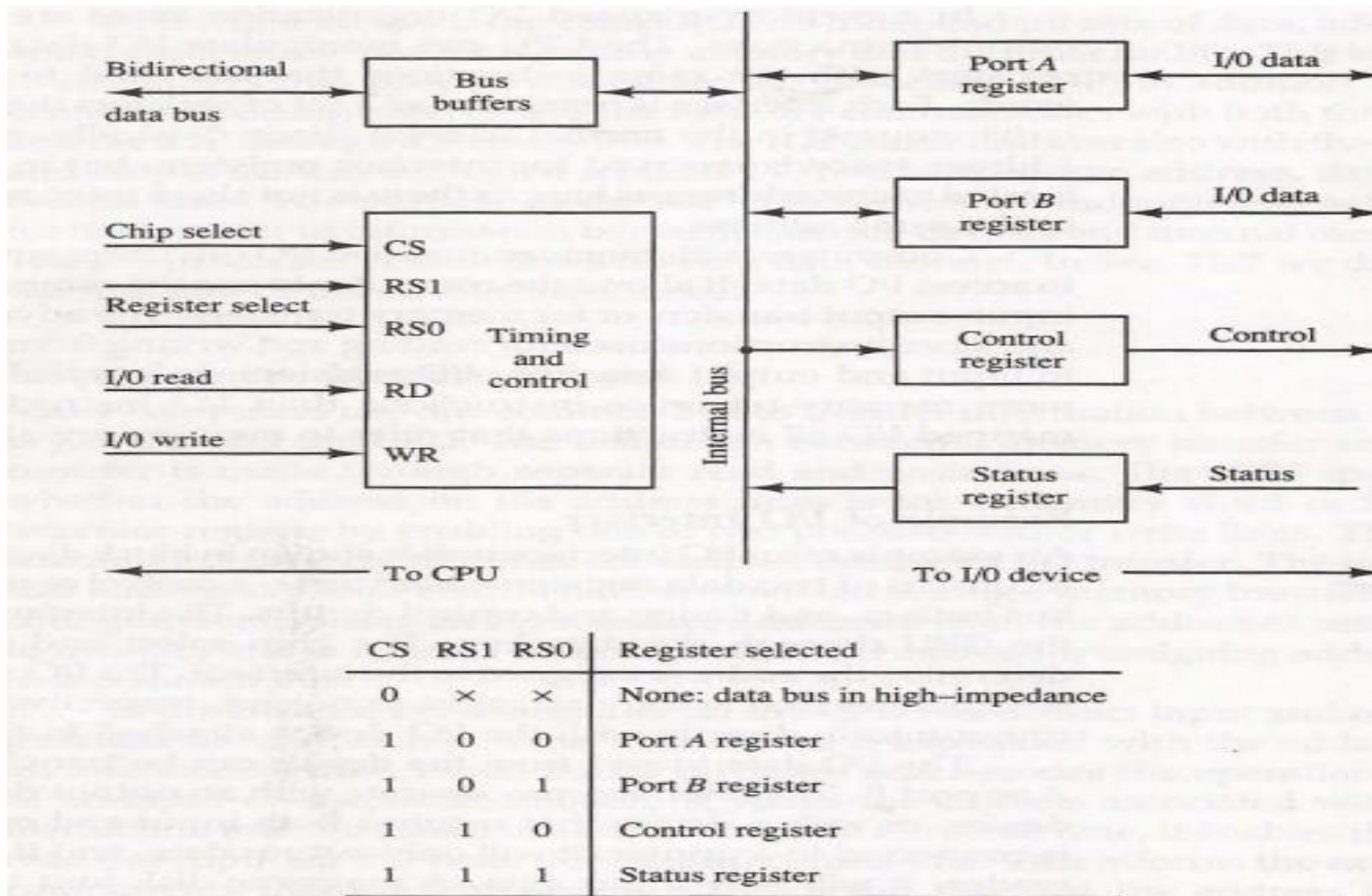


FIGURE 2.2 - EXAMPLE OF I/O INTERFACE UNIT



Memory Mapped I/O Interface :

- ❖ Computers with memory mapped I/O use memory type instruction to access I/O data
- ❖ Interface consist of two Data Registers called **Ports ,control registers , status registers ,bus buffers, timing and control circuit**
- ❖ The interface communicates with CPU through data bus
- ❖ The chip select and register select inputs determine the address assigned to the interface
- ❖ The I/O Read and Write are two control lines that specify an input or output respectively
- ❖ Four registers communicate directly with I/O device attached to the interface



Memory Mapped I/O Interface :

- ❖ The I/O data to and from the device can be transferred into port A and port B
- ❖ The interface may operate with input device ,output device or a device which require both input and output
- ❖ A command is passed to the I/O device by sending appropriate word to interface register (function code is not needed in such system)
- ❖ Control is sent to control register, status information is received from status register and data is transferred from port A or B registers
- ❖ Thus transfer of data ,control and status information is always via common data bus



Data Transfer:


- ❖ The internal operations of a digital system are synchronized by means of a clock pulses supplied by common pulse generator
- ❖ Clock pulses are applied to all registers within a unit and all data transfer among internal registers occurs simultaneously during occurrence of clock pulse



Synchronous Data Transfer:

- ❖ Two units such as CPU and I/O Interface are designed independent of each other and If interface register share common clock with CPU registers ,the data transfer between two unit is called **synchronous data transfer**

Asynchronous Data Transfer:

- ❖ Internal timing in each unit is independent from other in that each uses its own private clock for internal registers then data transfer between two units is called **asynchronous data transfer**
 - ❖ This approach is widely used in most computer systems. 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.
- 

Two way of achieving asynchronous Data Transfer :

Strobe :

strobe pulse supplied by one of the units to indicate to the other unit when data transfer has to occur.

Handshaking

- ? Accompany each data item being transferred with a control signal that indicates presence of data in the bus
- ? The unit receiving the data item responds with another control signal to acknowledge receipt of the data.
- ? The type of agreement between two independent unit is referred as **Handshaking**

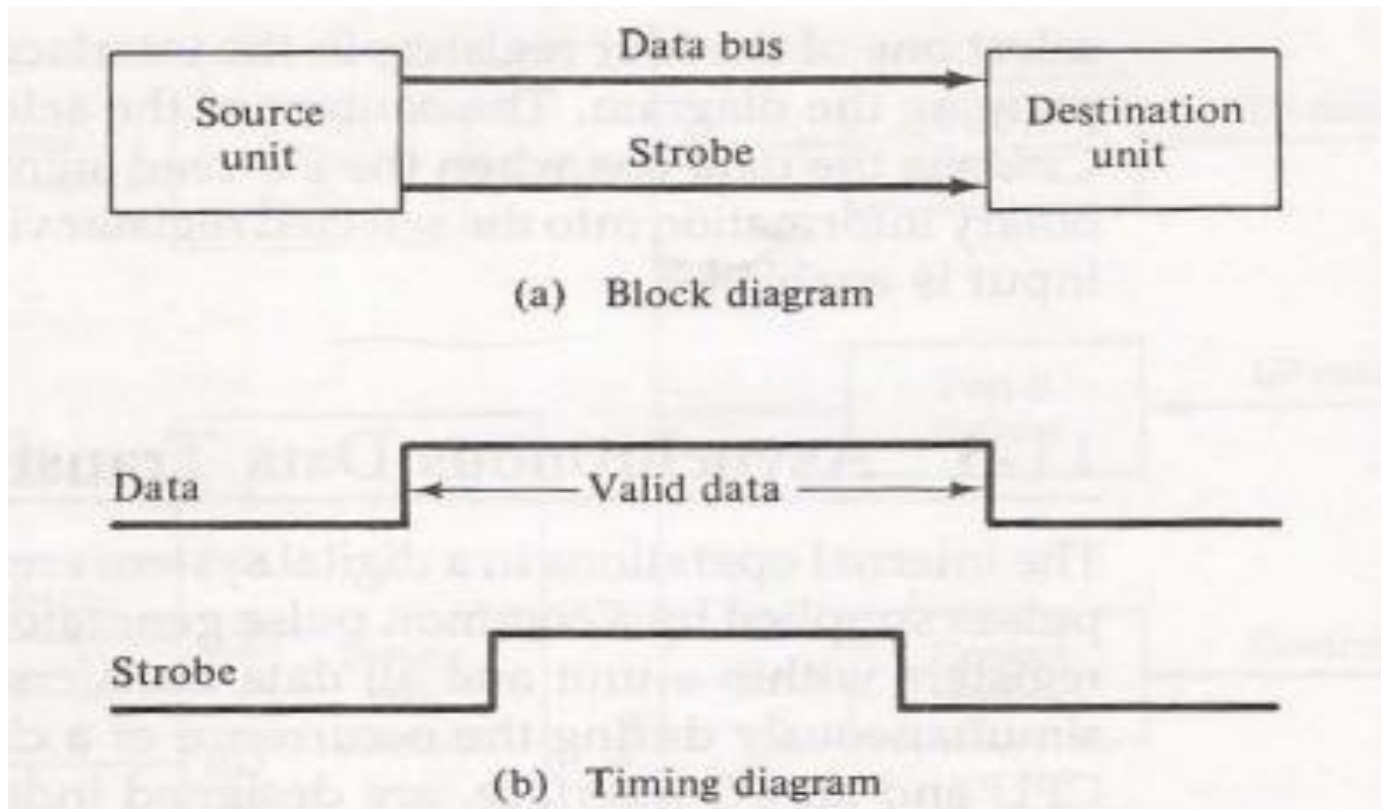
Note: The strobe pulse method and the handshaking method of asynchronous data transfer are not restricted to I/O transfers.



Strobe Control:

- ? Asynchronous data transfer employs only one control line to time each transfer
- ? The strobe may be activated by either the source or the destination unit.

source-initiated transfer and the timing diagram.

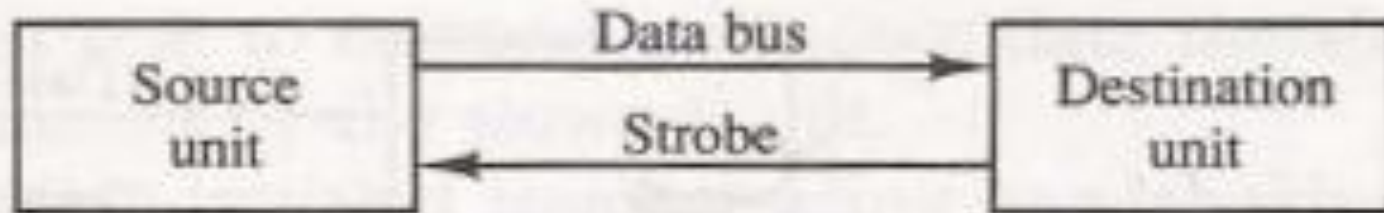


Source Initiated Strobe data transfer

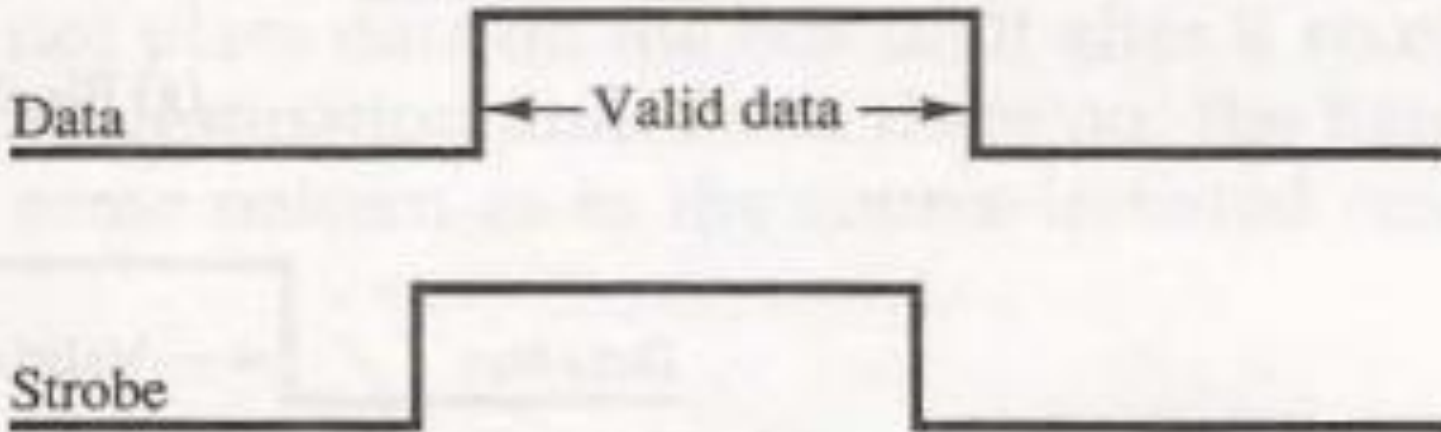
- ? The source unit first places data on data bus
- ? After a brief delay to ensure data settle to steady value source activates strobe pulse
- ? The information on data bus and strobe signal remain active state for a sufficient time period to allow destination unit to receive the data
- ? The destination unit has the falling edge of strobe pulse to transfer the contents of data bus into its one of the internal register
- ? The source removes data from data bus a brief period , after it disables strobe pulse .Actually strobe does not have to change information in data bus
- ? Strobe signal is disabled indicates that data bus does not contain valid data



Destination-initiated transfer and the timing diagram.



(a) Block diagram



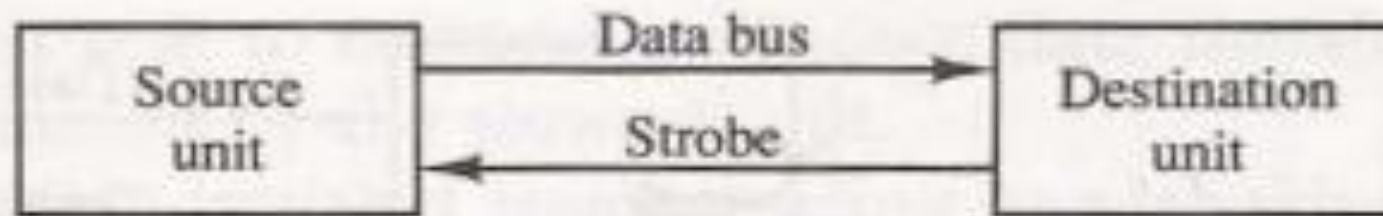
(b) Timing diagram

Destination Initiated Strobe data transfer

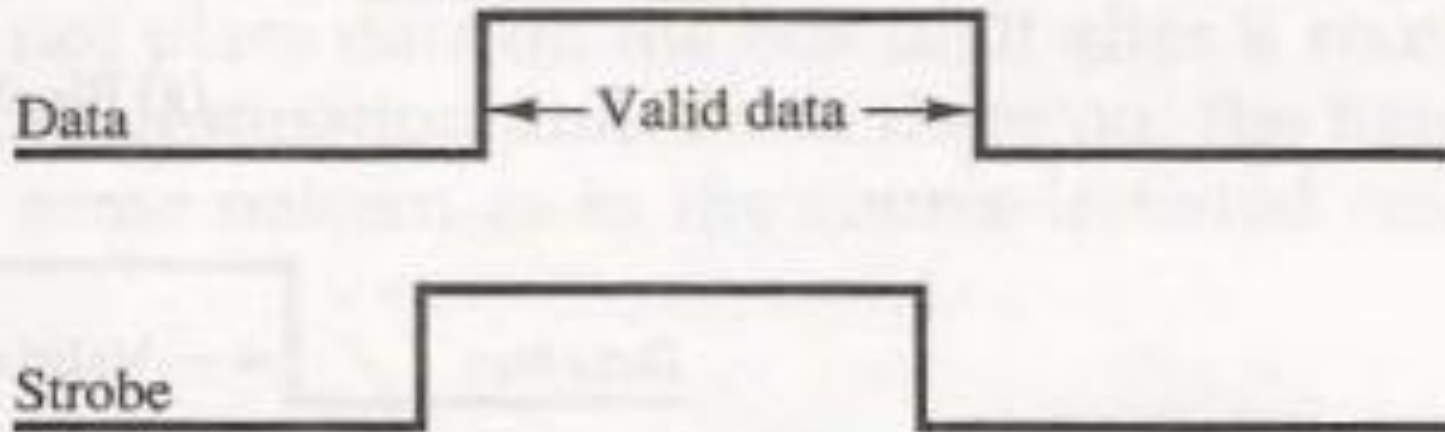
- ? The destination unit activates the strobe pulse informing the source to provide data
- ? Source unit responds by providing requested binary information on data bus
- ? Data may be valid and remain in the bus long enough for destination unit to accept it
- ? falling edge of strobe pulse is used again to trigger destination register
- ? Destination unit then disables strobe
- ? Source removes the data from bus after a predetermined time interval



FIGURE 2.3 SHOWS THE STROBE OF A MEMORY-READ CONTROL SIGNAL FROM THE CPU TO A MEMORY.



(a) Block diagram



(b) Timing diagram

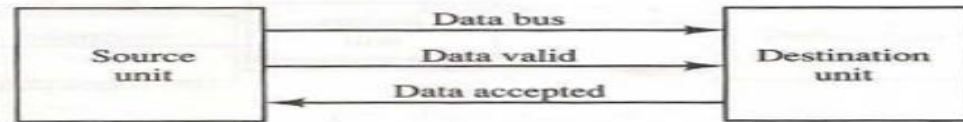
Disadvantage of the Strobe Method

- ? The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus
- ? The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that two-wire control initiates the transfer

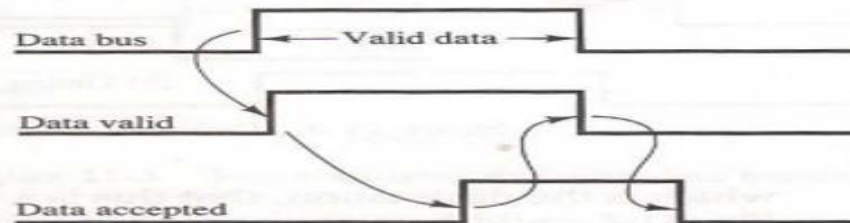


Handshake:

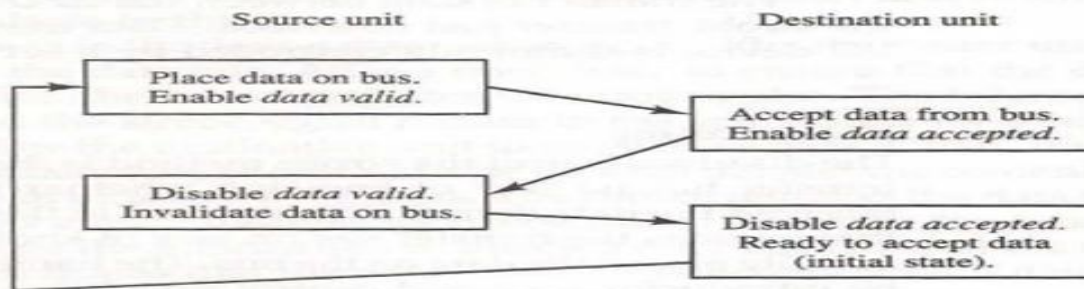
- ? The data transfer procedure when initiated by the source.
- ? The two handshaking lines are data valid, which is generated by the source unit, and data accepted, generated by the destination unit.
- ? The timing diagram shows the exchange of signals between the two units.



(a) Block diagram



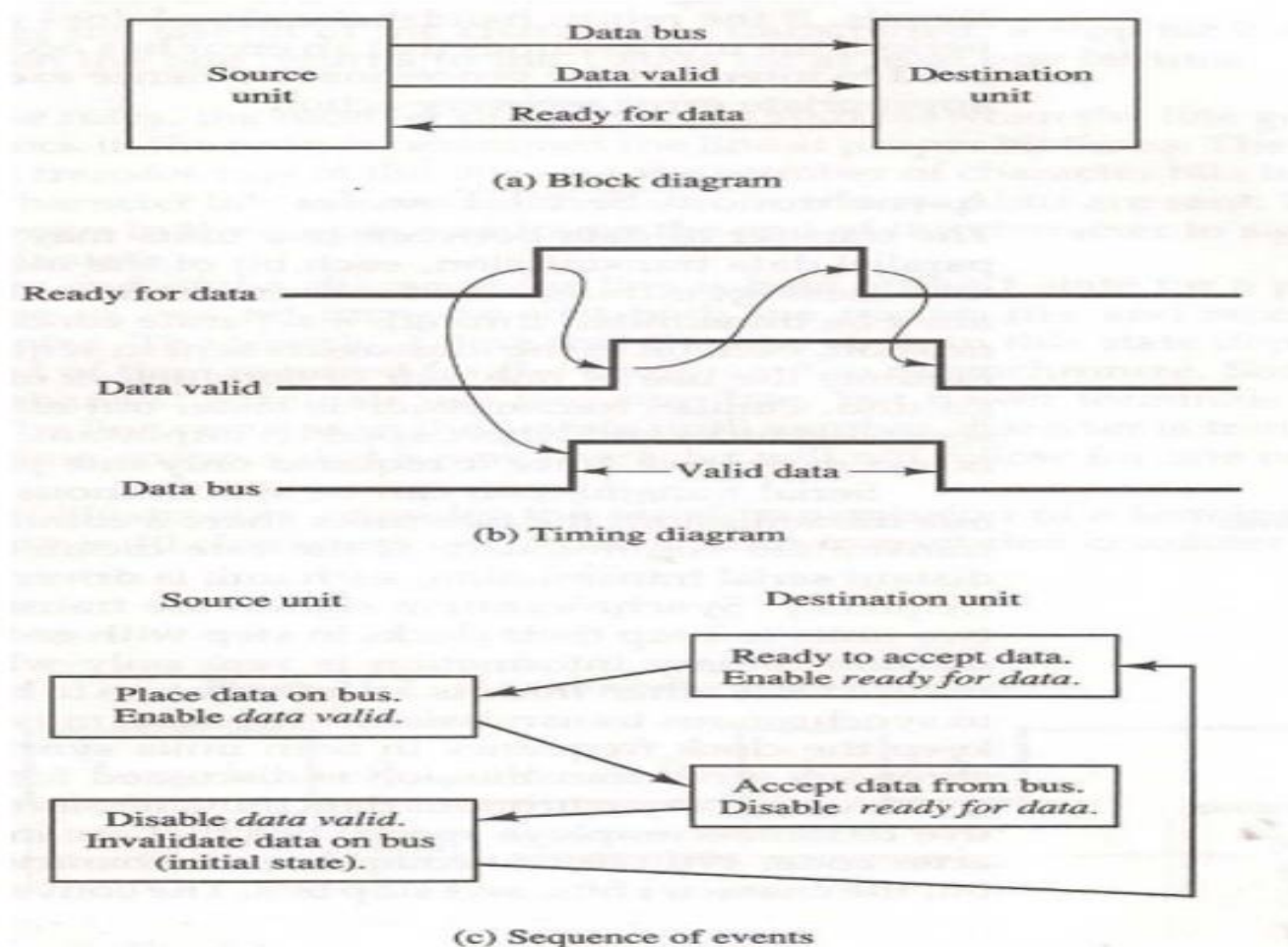
(b) Timing diagram



(c) Sequence of events

Handshake:

- ? The destination-initiated transfer using handshaking lines. Note that the name of the signal generated by the destination unit has been changed to ready for data to reflect its new meaning



Benefits of Handshaking:

- ? Handshaking provides high degree of flexibility and reliability because successful completion of data transfer relies on active participation of both units**
- ? If one unit is faulty ,data transfer will not be completed ,such an error can be detected by time out mechanism which produces an alarm if data transfer is not completed within predetermined time**



MODES OF TRANSFER

- ? Binary information received from an external device is usually stored in memory for later processing.**
- ? Information transferred from the central computer into an external device originates in the memory unit.**
- ? The CPU merely executes the I/O instructions and may accept the data temporarily, but the ultimate source or destination is the memory unit.**
- ? Data transfer between the central computer and I/O devices may be handled in a variety of modes.**
- ? Some modes use the CPU as an intermediate path; other transfer the data directly to and from the memory unit.**



MODES OF TRANSFER

- ? **Data transfer to and from peripherals may be handled in one of three possible 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.



Disadvantage Programmed I/O

- ? In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer.
- ? This is a time-consuming process since it keeps the processor busy needlessly.



Interrupt Initiated I/O

- ? It can be avoided by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device.
- ? In the meantime, the CPU can proceed to execute another program.
- ? The interface keeps monitoring the device. When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer.
- ? Upon detecting the external interrupt signal, the CPU momentarily 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



Direct Memory Access

- ? In direct memory access (DMA), the interface transfers data into and out of the memory unit through the memory bus.
- ? The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred and then proceeds to execute other tasks
- ? When the transfer is made, the DMA requests memory cycles through the memory bus. When the request is granted by the memory controller, the DMA transfers the data directly into memory.
- ? CPU merely delays its memory access operations to allow direct memory I/O transfer

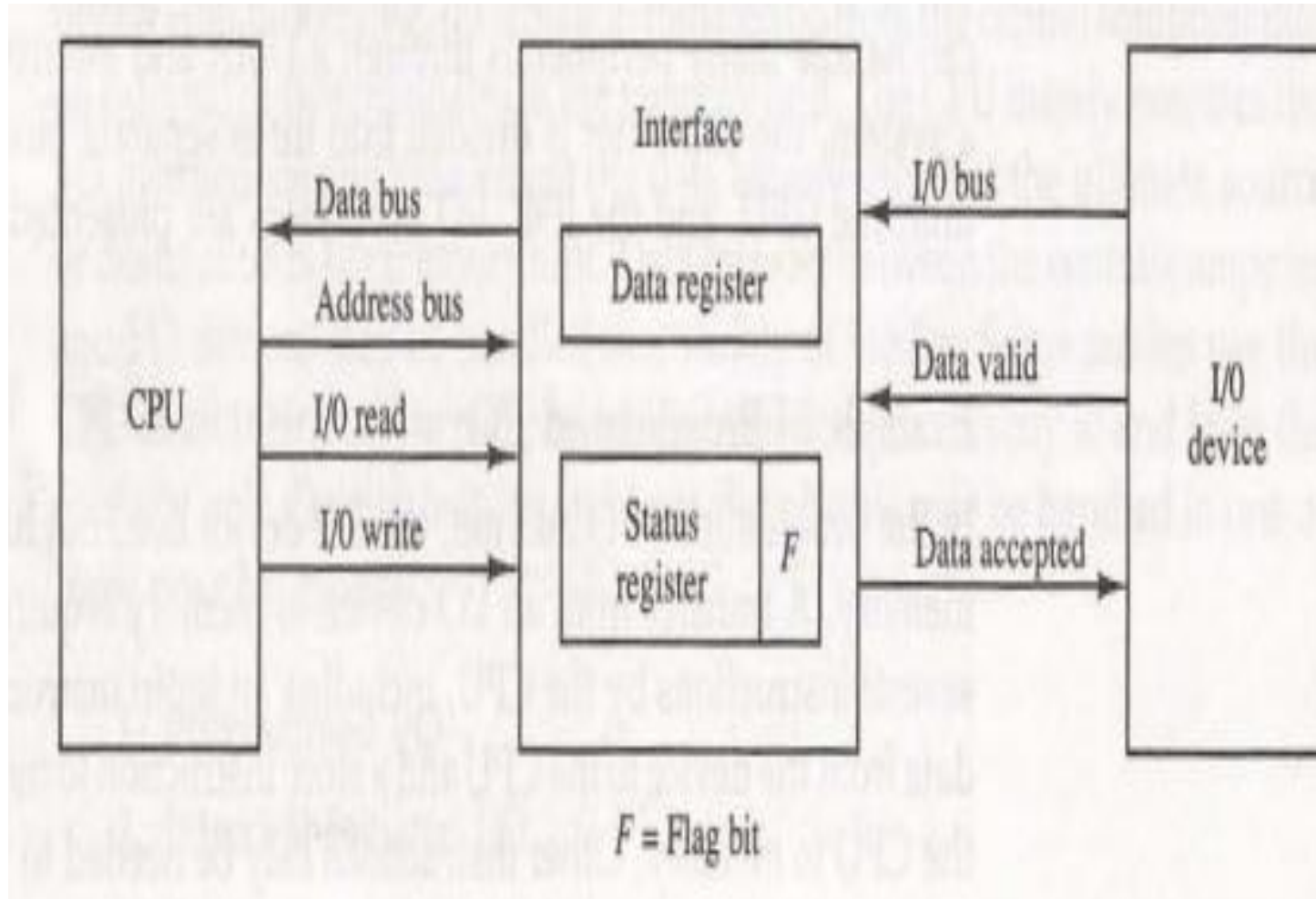


EXAMPLE OF PROGRAMMED I/O

- ? In the programmed I/O method, the I/O device does not have direct access to memory.
- ? A transfer from an I/O device to memory requires the execution of several instructions by the CPU.
- ? Input instruction to transfer the data from the device to the CPU
- ? Store instruction to transfer the data from the CPU to memory.
- ? Other instructions may be needed to verify that the data are available from the device and to count the numbers of words transferred.



DATA TRANSFER FROM I/O DEVICE THROUGH INTERFACE TO CPU



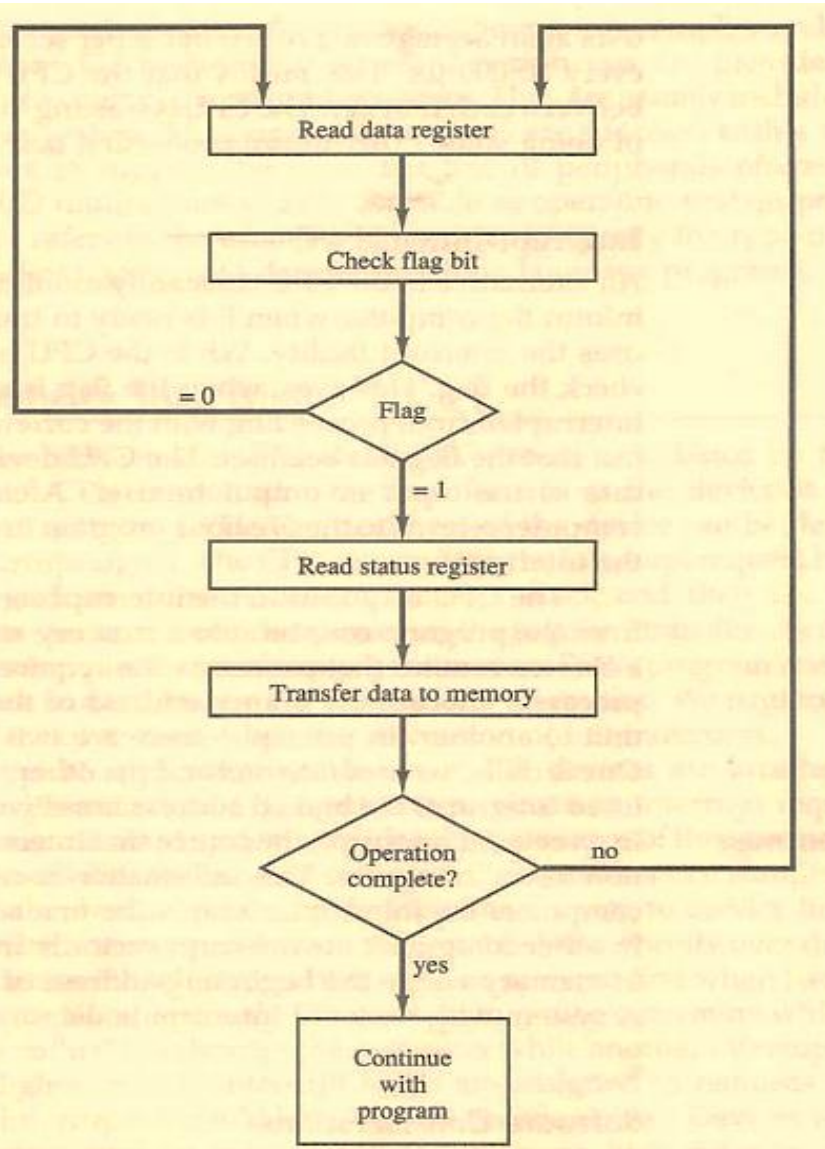
DATA TRANSFER FROM I/O DEVICE THROUGH INTERFACE TO CPU

- ? 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 referred as an F or “flag” bit.
- ? The device can now disable the data valid line, but it will not transfer another byte until the data accepted line is disabled by the interface.

DATA TRANSFER FROM I/O DEVICE THROUGH INTERFACE TO CPU

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

FLOWCHART FOR CPU PROGRAM TO INPUT DATA



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.

DATA TRANSFER FROM I/O DEVICE THROUGH INTERFACE TO CPU

- ? Each byte is read into a CPU register and then transferred to memory with a store instruction.
- ? A common I/O programming task is to transfer a block of words from an I/O device and store them in a memory buffer.
- ? The programmed IO method is particularly useful in small low-speed computers or in systems that are dedicated to monitor a device continuously.
- ? The difference in information transfer rate between the CPU and the I/O device makes this type of transfer inefficient.



INTERRUPT-INITIATE I/O

- ? Rather than CPU constantly monitoring the flag , let interface inform the CPU when it is ready to transfer the data
- ? Interrupt is used in this mode of transfer
- ? While the CPU is running a program, it does not check the flag. However, when the flag is set, the CPU is interrupted to execute the current program and is informed that the flag has been set.
- ? The CPU stops its current program and start input or output transfer.
- ?
- ? Once data transfer is completed, the computer returns to the previous program and continue where it was stopped before interrupt

INTERRUPT-INITIATE I/O

- ? The CPU responds to the interrupt signal by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.
- ? The processor chooses the branch address of the service routine using any one of the method **Vectored interrupt** or **Non-Vectored interrupt**
- ? In a non-vectored interrupt, the branch address is assigned to a fixed location in memory.
- ? In a vectored interrupt, the I/O interface that interrupts supplies the branch information to the CPU. this information is called the interrupt vector.



INTERRUPT-INITIATE I/O

- ? In some computers the interrupt vector is the first address of the I/O service routine.
- ? In other computers the interrupt vector is an address that points to a location in memory where the beginning address of the I/O service routine is stored.



SOFTWARE CONSIDERATIONS

- ? A computer must also have software routines for controlling peripherals and for transfer of data between the processor and peripherals.
- ? I/O routines must issue control commands to activate the peripheral and to check the device status to determine when it is ready for data transfer.
- ? Once ready, information is transferred item by item until all the data are transferred.
- ? A control command is given to execute a device function such as stop tape or print characters.
- ? Error checking and other useful steps often accompany the transfers.



SOFTWARE CONSIDERATIONS

- ? In interrupt-controlled transfers, the I/O software must issue commands to the peripheral to interrupt when ready and to service the interrupt when it occurs.
- ? In DMA transfer, the I/O software must initiate the DMA channel to start its operation
- ? Software control of input-output equipment is a complex undertaking. For this reason I/O routines for standard peripherals are provided by the manufacturer as part of the computer system.
- ? They are usually included within the operating system.
- ? Most operating systems are supplied with a variety of I/O programs to support the particular line of peripherals offered for the computer.
- ? I/O routines are usually available as operating system procedures and the user refers to the established routines to specify the type of transfer required without going into detailed machine language programs.

PRIORITY INTERRUPT

- ? Data transfer between the CPU and an I/O device is initiated by the CPU.
- ? The CPU cannot start the transfer unless the device is ready to communicate with the CPU.
- ? The readiness of the device can be determined from an interrupt signal.
- ? The CPU responds to interrupt request by storing the return address from PC into a memory stack and then program branches to a service routine that processes required transfer



PRIORITY INTERRUPT

- ? In a typical applications ,numbers of I/O devices are attached to the computer; with each device being able to originate an interrupt request.
- ? Several sources will request service simultaneously.
- ? The first task of the interrupt system is to identify the source of the interrupt and decide which device to service first.
- ? A priority interrupts is a system to determine which interrupt is to be served first when two or more requests are made simultaneously based on priority.
- ? 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.



PRIORITY INTERRUPT

- ? Establishing the priority of simultaneous interrupts can be done by software or hardware.

Priority Interrupt by Software (Polling)

- ? - Priority is established by the order of polling the devices (interrupt sources)
- ? - Flexible since it is established by software
- ? - Low cost since it needs a very little hardware - Very slow



PRIORITY INTERRUPT

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.**
- ? **Each interrupt source has its own interrupt vector to access directly to its own service routine**
- ? **The hardware priority function can be established by either a serial or a parallel connection of interrupt lines.**
- ? **The serial connection is also known as the daisy chaining method.**



DAISY CHAINING PRIORITY

- ? Daisy chaining method of establishing priority consist of serial connection of all devices that request an interrupt.
- ? The device with highest priority is placed in first position ,followed by lower priority devices up to device with the lowest priority which is placed last in the chain
- ? The interrupt request line is common to all devices and forms a wired logic connection

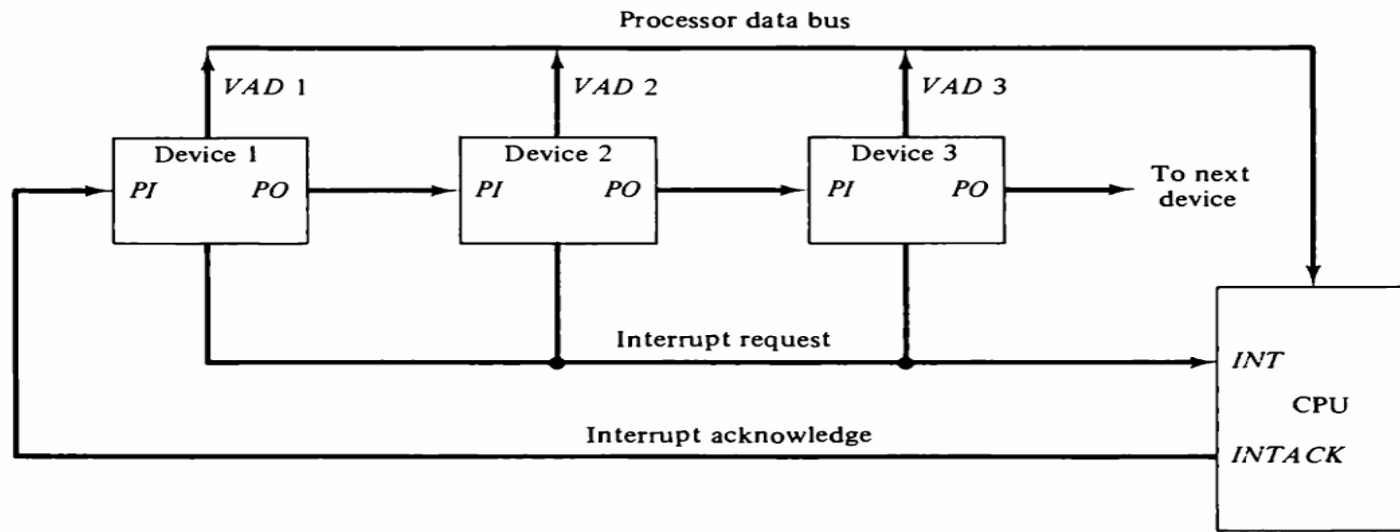


Figure 11-12 Daisy-chain priority interrupt.

DAISY CHAINING PRIORITY

- ? If any device has its interrupt signal in low level state ,the interrupt line goes to low level state and enables interrupt input in the CPU
- ? When no interrupts are pending ,the interrupt line stays in the high-level state and no interrupts are recognised by the CPU, equivalent to a negative logic OR operation
- ? The CPU responds to an interrupt request by enabling the interrupt acknowledge line
- ? This signal is received by device1 at its PI(priority interrupt) input ,the acknowledge signal passes on to the next device the (priority output) output only if device 1 is not requesting an interrupt
- ? If device1 has a pending interrupt ,it blocks the acknowledge signal from next device by placing 0 in PO output , It then proceed to insert its own interrupt vector address (VAD) into data bus for CPU to use during interrupt cycle

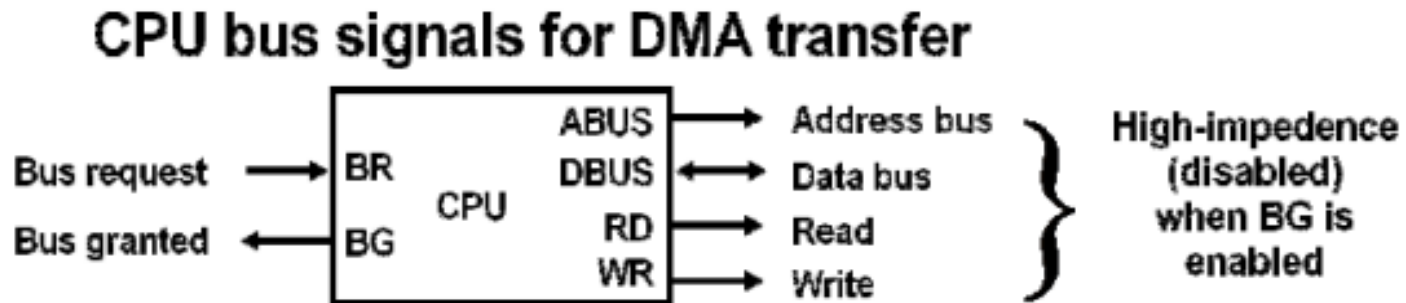
DAISY CHAINING PRIORITY

- ? A device with 0 in its PI input generates 0 in its PO output to inform next lower priority device that acknowledge signal has been blocked
- ? A device that is requesting an interrupt and has 1 in its PI input will interrupt acknowledge signal by placing 0 in its PO output
- ? If device does not have pending interrupts ,it transmits the acknowledge signal to next device by placing 1 in its PO output
- ? Thus device with $PI=1$ and $PO=0$ is one with highest priority that is requesting an interrupt and this device places its vector address (VAD) on data bus
- ? The daisy chain arrangement gives highest priority to device that receives the interrupt acknowledgement signal from CPU
- ? Farther the device from first position ,lower is its priority



DIRECT MEMORY ACCESS (DMA)

- ? The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU.
- ? Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called **Direct Memory Access (DMA)**.



DIRECT MEMORY ACCESS (DMA)

- ? 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 CPU has relinquished(ceased) the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.
- ? The DMA communicates with the external peripheral through the request and acknowledge lines by using a prescribed handshaking procedure.



DIRECT MEMORY ACCESS (DMA)

- ? The DMA controller has three registers: an address register, a word count register, and a control register. The address register contains an address to specify the desired location in memory. The address bits go through bus buffers into the address bus. The address register is incremented after each word that is transferred to memory.
- ? The control register specifies the mode of transfer. All registers in the DMA appear to the CPU as I/O interface registers. Thus, the CPU can read from or write into the DMA registers under program control via the data bus.
- ? The DMA is first initialized by the CPU. After that, the DMA starts and continues to transfer data between memory and peripheral unit until an entire block is transferred. The initialization process is essentially a program consisting of I/O instructions that include the address for selecting particular DMA registers.



DIRECT MEMORY ACCESS (DMA)

- ? The CPU initializes the DMA by sending the following information through the data bus:
 - The starting address of the memory block where data are available (for read) or where data are to be stored (for write)
 - The word count, which is the number of words in the memory block
 - Control to specify the mode of transfer such as read or write
 - 4. A control to start the DMA transfer
- ? The starting address is stored in the address register. The word count is stored in the word count register, and the control information in the control register.
- ? Once the DMA is initialized, the CPU stops communicating with the DMA unless it receives an interrupt signal or if it wants to check how many words have been transferred.



DIRECT MEMORY ACCESS (DMA)

- ? During 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 microprocessors is to disable the buses through special control signals.
- ? The bus request (BR) input is used by the DMA controller to request the CPU to cease control of the buses. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and write lines into a high impedance state behaves like an open circuit, which means that the output is disconnected and does not have a logic significance.



DIRECT MEMORY ACCESS (DMA)

- ? The CPU activates the Bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state. The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor intervention. When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant, takes control of the buses, and returns to its normal operation.
- ? When the DMA takes control of the bus system, it communicates directly with the memory. The transfer can be made in several ways.
 - In **DMA burst transfer**, a block sequence consisting of a number of memory words is transferred in a continuous burst while the DMA controller is master of the memory buses. This mode of transfer is needed for fast devices such as magnetic disks, where data transmission cannot be stopped or slowed down until an entire block is transferred.



DIRECT MEMORY ACCESS (DMA)

- ? An alternative technique called **cycle stealing** allows the 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 delays its operation for one memory cycle to allow the direct memory I/O transfer to “steal” one memory cycle.

