

# Input /Output Organization

Module 4

Jestin James M  
Assistant Professor, Dept of Computer Science  
Little Flower College, Guruvayoor



# Topics To Discuss

- Peripheral Devices
- Input-Output interface◆
- Asynchronous Data Transfer
- Modes of Transfer
- Priority Interrupt
- Direct Memory Access
- input-Output Processor
- Serial Communication

# Peripheral Devices

- Purpose of I/O devices
- Keyboard
- Mouse
- Monitor
- Printer
- Magnetic Tape
- Magnetic Disk



# monitor

- Output device
- CRT most common type monitor
- LCD is another type





# How CRT Monitors works

- The CRT contains an electronic gun that sends an electronic beam to a phosphorescent screen in front of the tube.
- The beam can be deflected horizontally and vertically.
- To produce a pattern on the screen, a grid inside the CRT receives a variable voltage that causes the beam to hit the screen and make it glow at selected spots.
- Horizontal and vertical signals deflect the beam and make it sweep across the tube, causing the visual pattern to appear on the screen.



# Printer

- 3 types printer
- Daisywheel,
- Dot matrix and
- laser printers
- Impact printer & Non Impact Printer

# Daisywheel,

- The daisywheel printer contains a wheel with the characters placed along the circumference.
- To print a character, the wheel rotates to the proper position and an energized magnet then presses the letter against the ribbon.





# Dot matrix printer

- The dot matrix printer contains a set of dots along the printing mechanism.
- For example, a 5 x 7 dot matrix printer that prints 80 characters per line has seven horizontal lines, each consisting of  $5 \times 80 = 400$  dots
- Each dot can be printed or not, depending on the specific characters that are printed on the line.





# Laser printer

- It uses a rotating photographic drum that is used to imprint the character images.
- The pattern is then transferred onto paper in the same manner as a copying machine.
- Alphanumeric Characters - ASCII

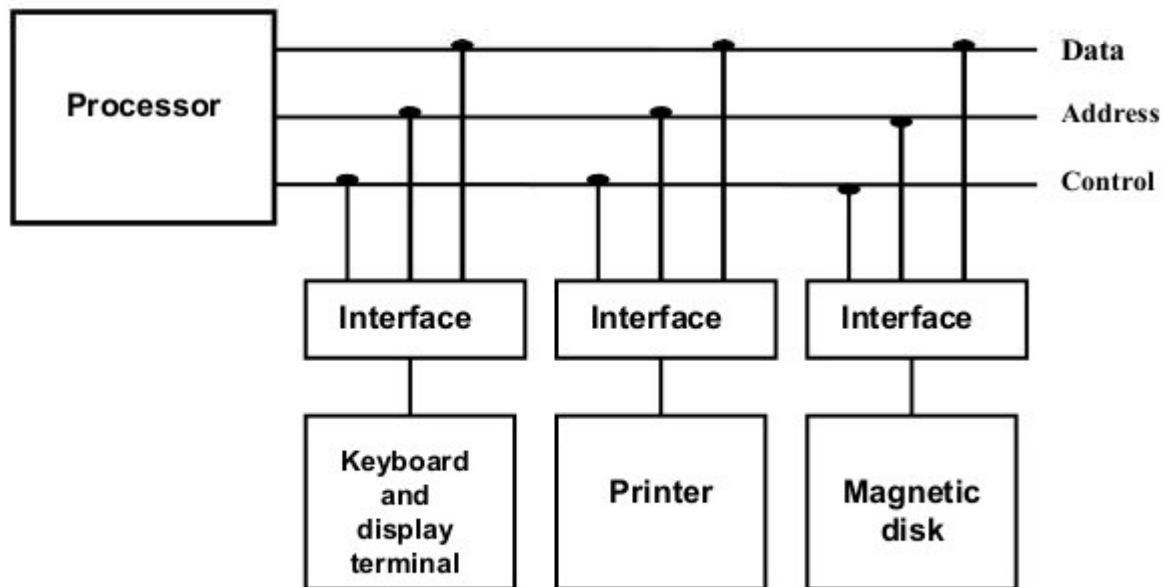


# Input-Output Interface

- Why I/O interface? Hardware
  1. A conversion of signal values may be required.
  2. Speed synchronization mechanism needed
  3. Data codes and formats in peripherals differ from the word format in the CPU and memory.
  4. 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.

# I/O Bus and Interface Modules

## I/O BUS and Interface Module



Connection of I/O bus to input-output devices



# I/O Bus and Interface Modules

- The processor places a device address on the address lines
- Each interface attached to the I/O bus contains an address decoder that monitors the address lines
- When the interface detects its own address, it activates the path between the bus lines and the device that it controls
- the processor provides a function code in the control lines





# I/O Bus and Interface Modules

- The function code is referred to as an I/O command
- 4 types of commands
  1. **Control command** is issued to activate the peripheral and to inform it what to do. Ex. What to do with printer
  2. **A status command** is used to test various status conditions in the interface and the peripheral Ex. Printer connected or not
  3. **A data output command** causes the interface to respond by transferring data from the bus into one of its register
  4. **The data input command** is the opposite of the data output. In this case the interface receives an item of data from the peripheral and places it in its buffer register.



# I/O versus Memory Bus

- 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 VO with common control lines.



# Separate for Memory & I/O

- the computer has independent sets of data, address, and control buses, one for accessing memory and the other for I/O
- This is done in computers that provide a separate I/O processor (IOP) in addition to the CPU.
- The memory communicates with both the CPU and the IOP through a memory bus.
- The IOP communicates also with the I/O devices through a separate I/O bus with its own address, data and control lines





# Separate for Memory & I/O

- The purpose of the IOP is to provide an independent pathway for the transfer of information between external devices and internal memory.
- The I/O processor is sometimes called a data channel





# Isolated v/S Memory-Mapped I/O

- Many computers use one common bus to transfer information between memory or I/O and the CPU
- The CPU specifies whether the address on the address lines is for a memory word or for I/O by enabling one of two possible read or write lines
- The I/O read and I/O write control lines are enabled during an I/O transfer
- The memory read and memory write control lines are enabled during a memory transfer



## Isolated v/S Memory-Mapped I/O

- This configuration isolates all I/O interface addresses from the addresses assigned to memory
- It is referred to as the **isolated I/O method for assigning addresses in a common bus**.
- If CPU want an I/O instruction, it places the address associated with the instruction into the common address lines. At the same time, it enables the I/O read (for input) or I/O write (for output) control line.
- If CPU want memory instruction it places the address associated with the instruction into the common address lines, Memory read or memory write



# Memory mapped I/O

- one set of read and write signals and do not distinguish between memory and I/O addresses. This configuration is referred to as **memory-mapped I/O**
- The computer treats an interface register as being part of the memory system.
- The assigned addresses for interface registers cannot be used for memory words, which reduces the memory address range available.
- In a memory-mapped I/O organization there are no specific input or output instructions.





# Memory mapped I/O

- Computers with memory-mapped I/O can use memory-type instructions to access I/O data
- the load and store instructions used for reading and writing from memory can be used to input and output data from I/O registers.





# Asynchronous Data Transfer

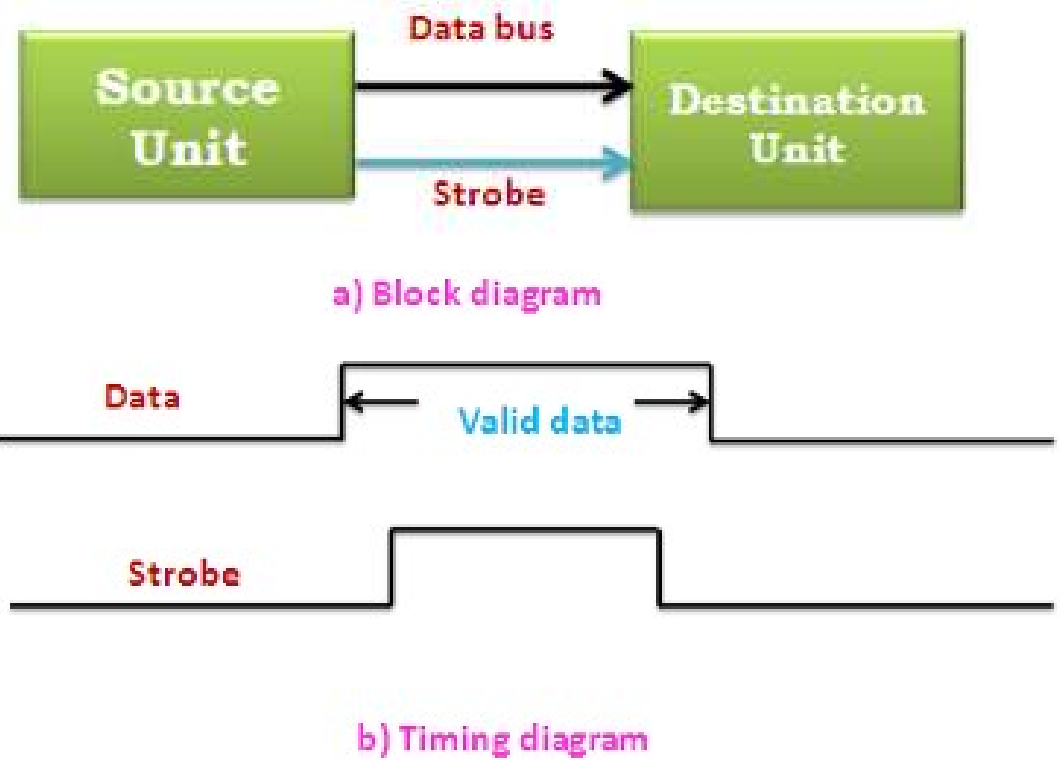
- The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator
- CPU and an I/O interface, are designed independently of each other. If the registers in the interface share a common clock with the CPU registers, the transfer between the two units is said to be *synchronous*.
- internal timing in each unit is independent from the other in that each uses its own private clock for internal registers. In that case, the two units are said to be *asynchronous* to each other



# 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.
  1. **strobe** pulse supplied by one of the units to indicate to the other unit when the transfer has to occur
  2. 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**

# Strobe Control





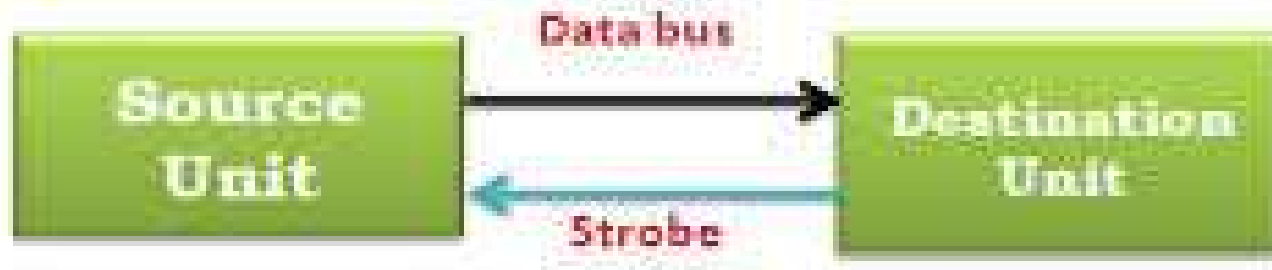


# Strobe Control initiated by Source

- The source unit first places the data on the data bus
- After a brief delay to ensure that the data settle to a steady value, the source activates the strobe pulse.
- The information on the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data.
- The source removes the data from the bus a brief period after it disables its strobe pulse.



# Strobe Control -Destination



a) Block diagram



b) Timing diagram



# Strobe Control -Destination

- In this case the destination unit activates the strobe pulse, informing the source to provide the data.
- The source unit responds by placing the requested binary information on the data bus
- The falling edge of the strobe pulse can be used again to trigger a destination register.
- The destination unit then disables the strobe.
- The source removes the data from the bus after a predetermined time interval.



# Strobe control

- the strobe pulse is actually controlled by the clock pulses in the CPU.
- The CPU is always in control of the buses and informs the external units how to transfer data

## Disadvantage

- 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

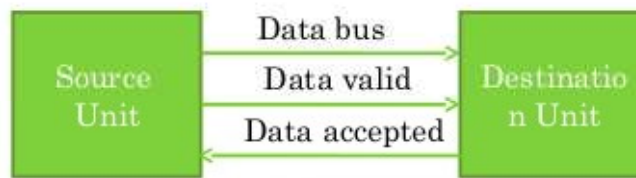


# Handshaking

- One control line is in the same direction as the data flow in the bus from the source to the destination.
- It is used by the source unit to inform the destination unit whether there are valid data in the bus
- The control line from the destination to the source.
- It is used by the destination unit to inform the source whether it can accept data.

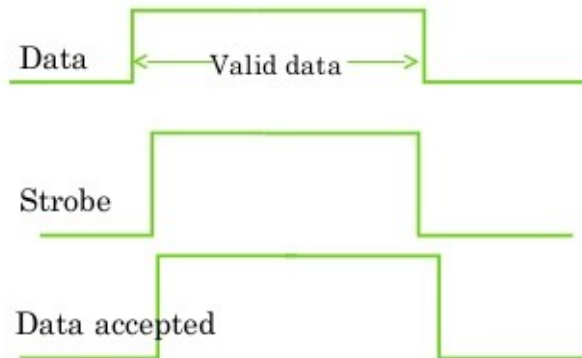


# SOURCE INITIATED TRANSFER

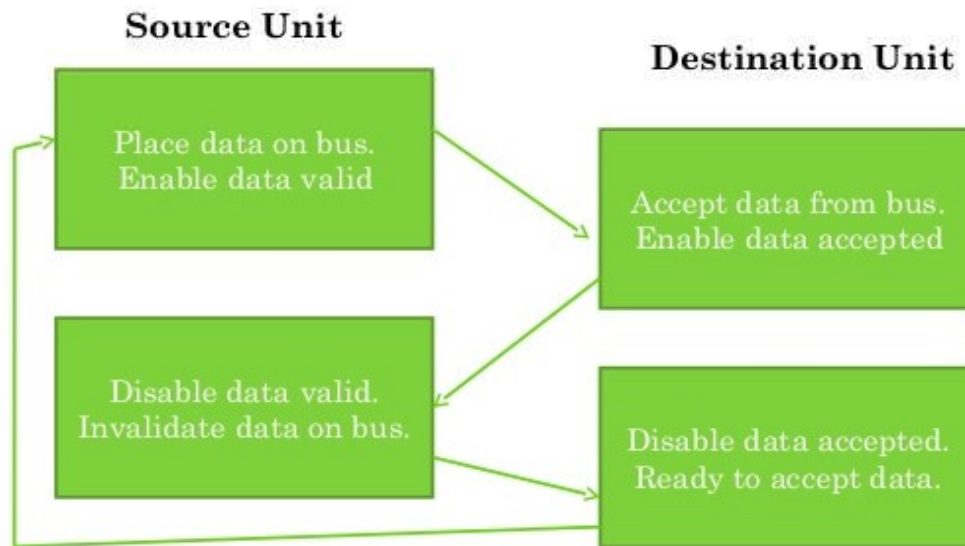


**Block diagram**

**Timing diagram**

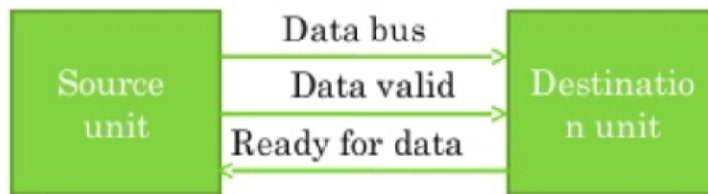


## SOURCE INITIATED TRANSFER USING HANDSHAKING

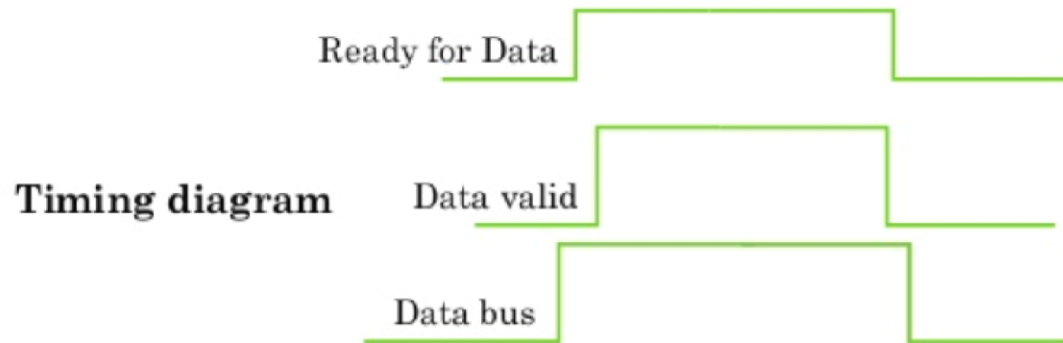


Sequence of events

# DESTINATION INITIATED TRANSFER

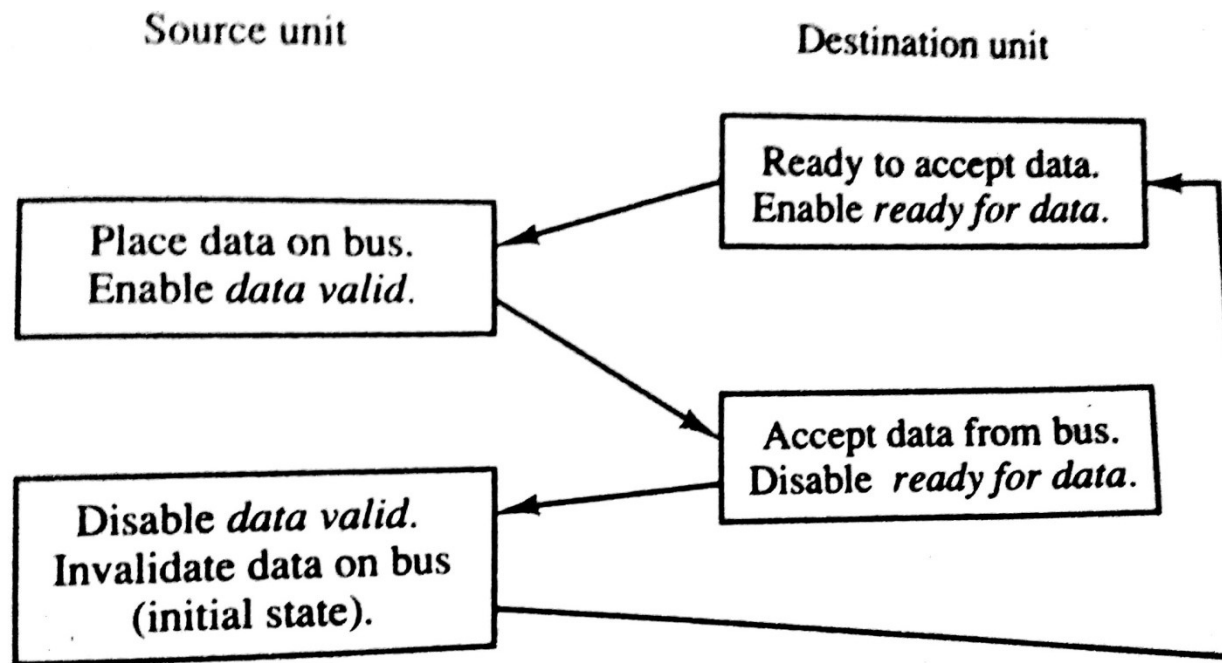


**Block diagram**



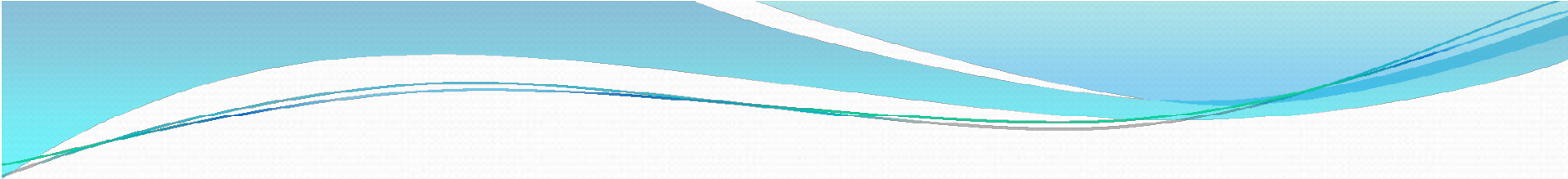
**Timing diagram**

# Handshaking -Destination



(c) Sequence of events



- 
- 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
  - If one unit is faulty, the data transfer will not be completed.
  - timeout



# Asynchronous Serial Transfer

- Transfer may be parallel or serial
- In parallel data transmission n-bit message must be transmitted through n separate conductor paths
- is used for short distances and where speed is important.
- Serial transmission can be synchronous or asynchronous



# Synchronous Serial Transmission

- In synchronous transmission, the two units share a common clock frequency
- bits are transmitted continuously at the rate dictated by the clock pulses
- In long distance serial transmission, each unit is driven by a separate clock of the same frequency
- Synchronization signals are transmitted periodically between the two units to keep their clocks in step with each other





# asynchronous

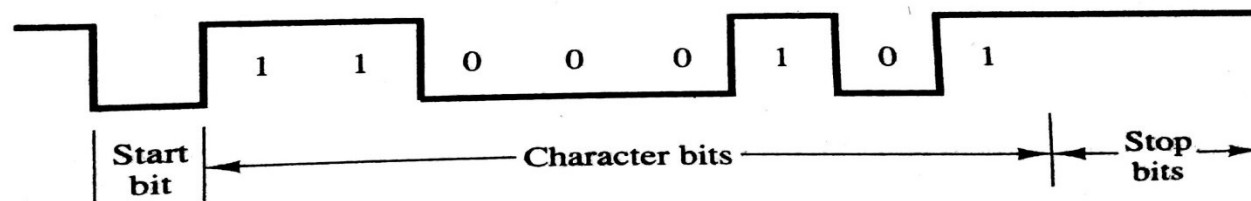
- binary information is sent only when it is available and the line remains idle when there is no information to be transmitted
- each character consists of three parts: a start bit, the character bits, and stop bits
- at the 1-state when no characters are transmitted.
- The first bit, called the start bit, is always a 0 and is used to indicate the beginning of a character.
- The last bit called the stop bit is always a 1.



# Asynchronous serial transmission.

1. When a character is not being sent, the line is kept in the 1-state
2. The initiation of a character transmission is detected from the start bit, which is always 0.

Figure 11-7 Asynchronous serial transmission.



3. The character bits always follow the start bit.

4. a stop bit is detected when the line returns to the 1-state



# Modes of Transfer

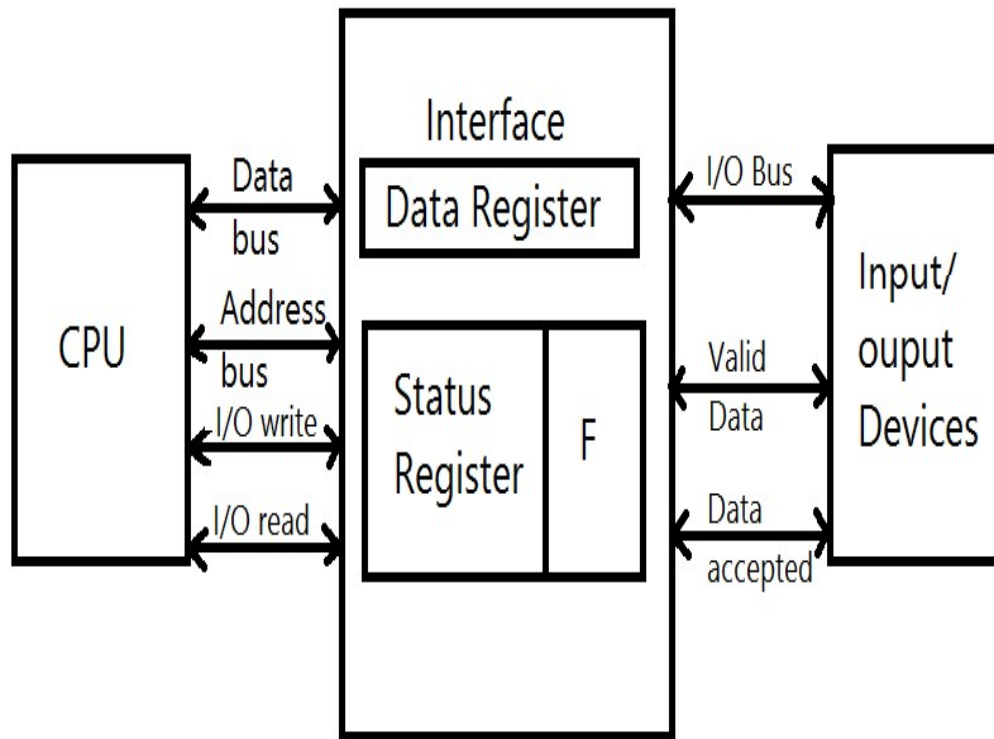
- Binary information received from an external device is usually stored in memory for later processing
- Data transfer between the central computer and I/O devices may be handled in a variety of modes
  1. Programmed I/O
  2. Interrupt-initiated I/O
  3. Direct memory access (DMA)&
  4. IOP (Input/ Output processor)



# 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
- transfer is to and from a CPU register and peripheral
- Or the data to and from CPU and memory.
- Transferring data under program control requires constant monitoring of the peripheral by the CPU

# Programmed I/O







# Programmed I/O

- 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



# Programmed 1/0

- 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
- 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.
- If flag = 1, the CPU reads the data from the data register. flag bit is then cleared to 0 by either the CPU or the interface, depending on how the interface circuits are designed

SECTION 11-4 Modes of Transfer

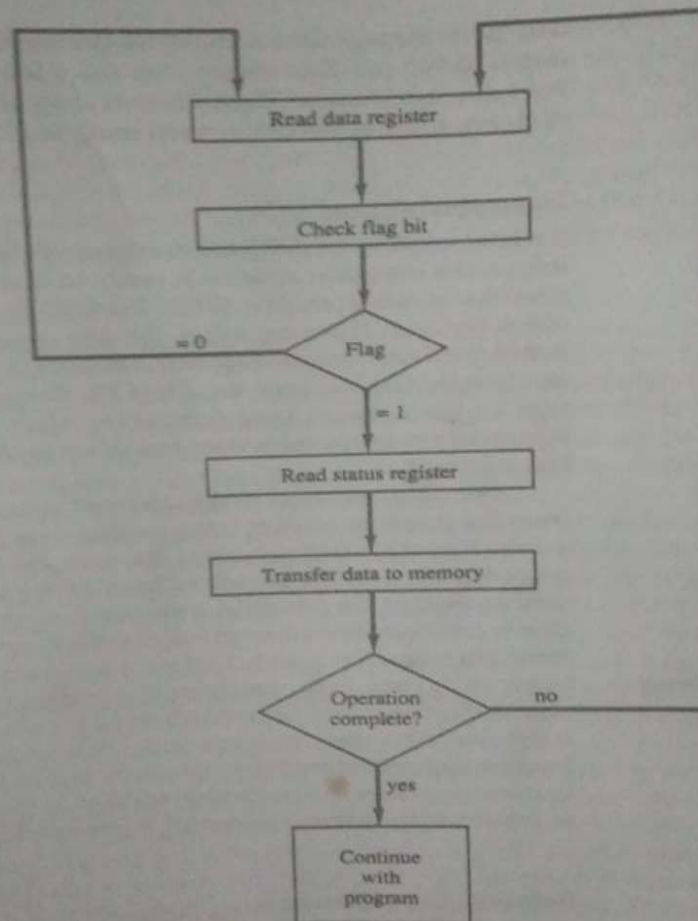


Figure 11-11 Flowchart for CPU program to input data.





# Interrupt-Initiated I/O

- While the CPU is running a program, it does not check the flag.
- when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set.
- The CPU 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





# Interrupt-Initiated I/O

- When Interrupt occur ,store 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.
- processor chooses the branch address of the service routine by two ways
  1. vectored interrupt
- Non vectored interrupt : In a non vectored interrupt, the branch address is assigned to a fixed location in memory.

# vectored interrupt

- The source that interrupts supplies the branch information to the computer. This information is called the **interrupt vector**.
- 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.



# Priority Interrupt

- when two or more Interrupt requests arrive simultaneously which condition is to be serviced first
- Establishing the priority of simultaneous interrupts can be done by software or hardware.
- A polling procedure is used to identify the highest-priority source by software means





# Poll -software

- The order in which they are tested determines the priority of each interrupt.
- The highest-priority source is tested first, and if its interrupt signal is on, control branches to a service routine for this source.
- disadvantage of the software method is that if there are many interrupts, the time required to poll them can exceed the time available to service the I/O device.





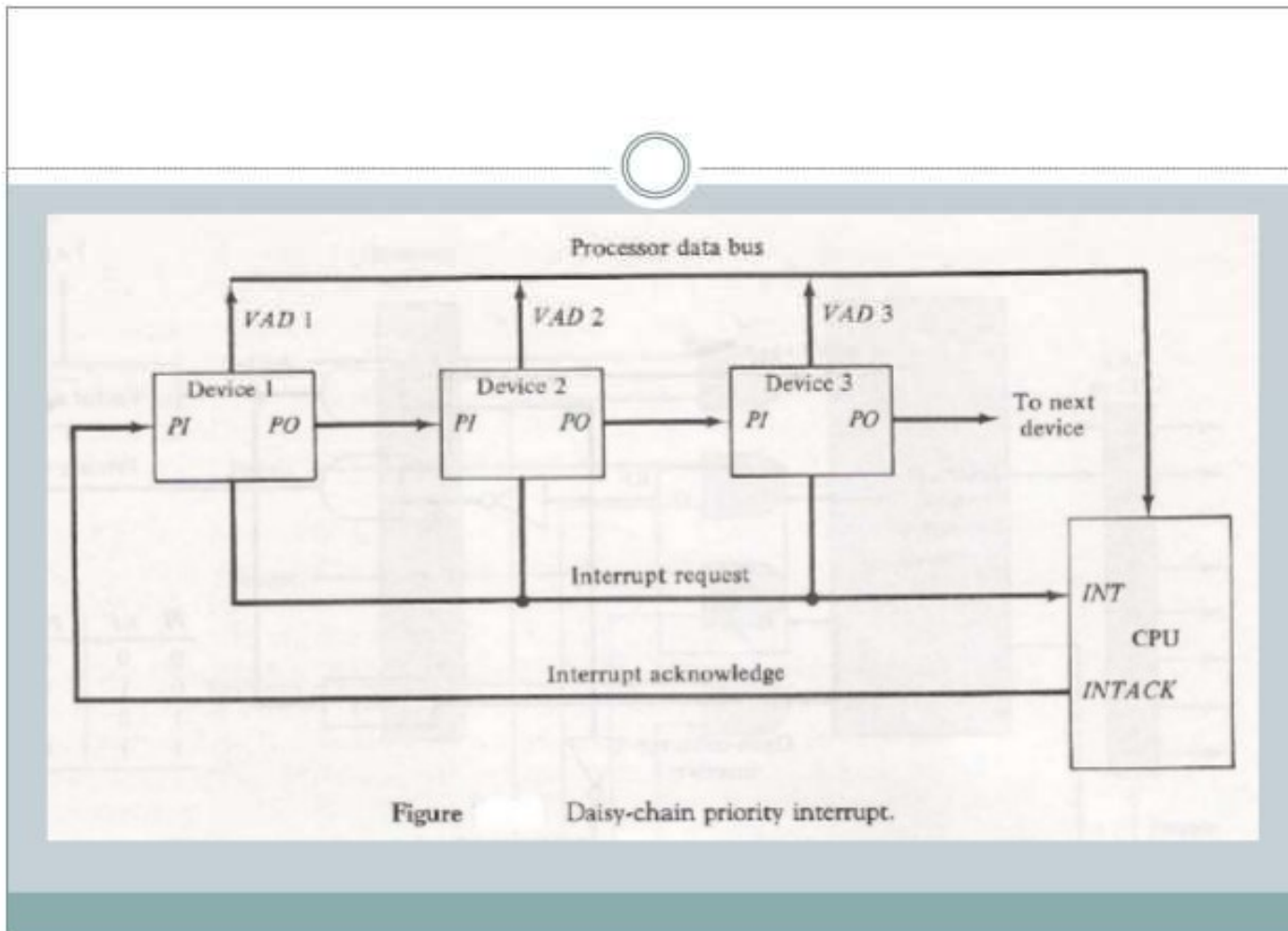
# hardware priority-interrupt

- A hardware priority-interrupt unit functions as an overall manager in an interrupt system environment
- It accepts interrupt requests from many sources, determines which of the incoming requests has the highest priority,
- and issues an interrupt request to the computer based on this determination
- 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

- The device with the highest priority is placed in the first position,
- followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain





# Daisy-Chaining Priority

- The CPU responds to an interrupt request by enabling the interrupt acknowledge line
- This signal is received by device 1 at its PI (priority in) input.
- The acknowledge signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt.
- If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output.





# Daisy-Chaining Priority

- It then proceeds to insert its own interrupt vector address (V AD) into the data bus for the CPU to use during the interrupt cycle.
- A device with a 0 in its PI input generates a 0 in its PO output to inform the next-lower-priority device that the acknowledge signal has been blocked



# Parallel Priority Interrupt

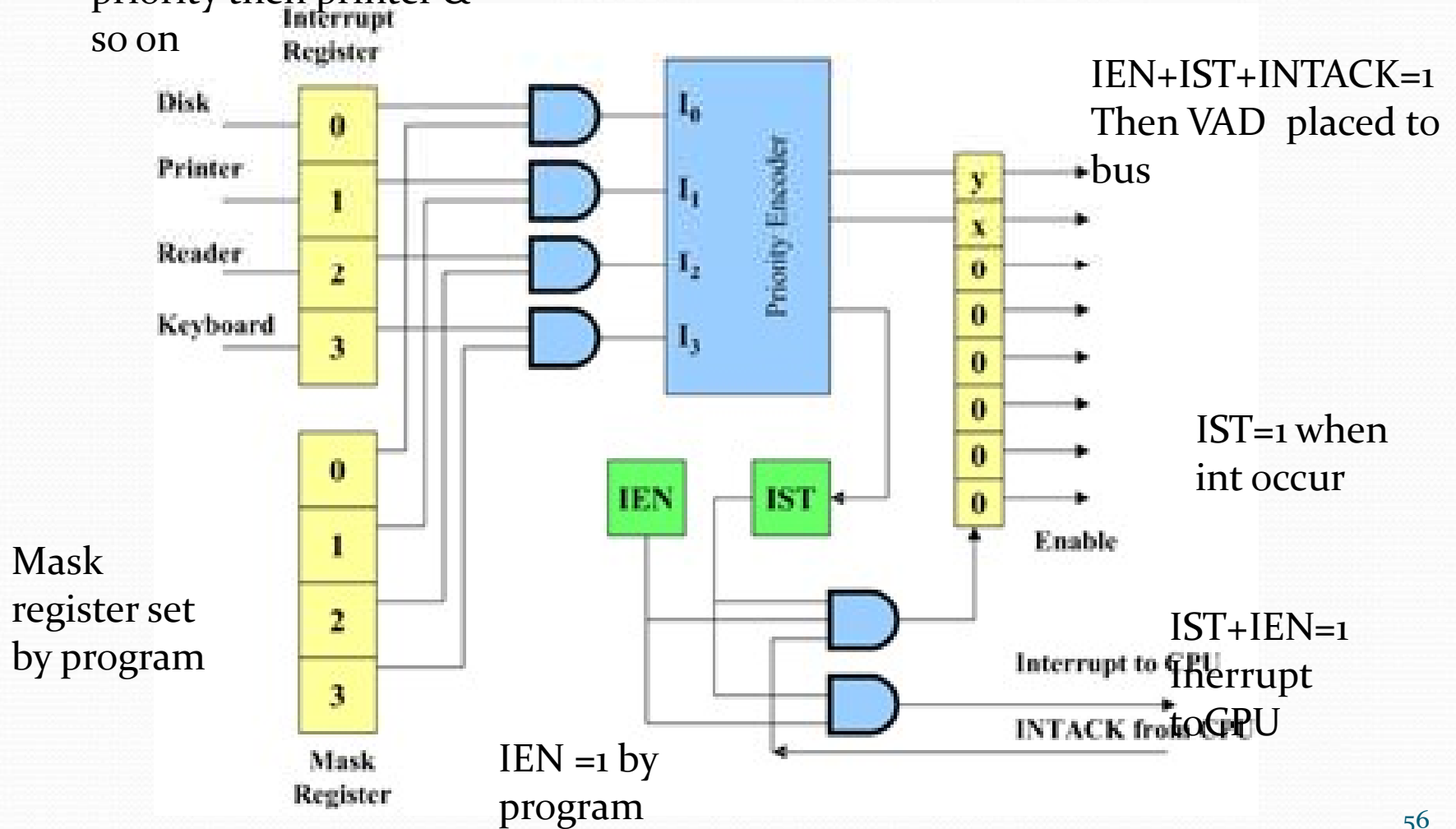
- The parallel priority interrupt method uses a register whose bits are set separately by the interrupt signal from each device.
- Priority is established according to the position of the bits in the register.
- In addition to the interrupt register, the circuit may include a mask register whose purpose is to control the status of each interrupt request.
- The mask register can be programmed to disable lower-priority interrupts while a higher-priority device is being serviced



# Parallel Priority Interrupt

- It can also provide a facility that allows a high-priority device to interrupt the CPU while a lower-priority device is being serviced.

Disk has Higher priority than printer & so on





## INTERRUPT PRIORITY ENCODER

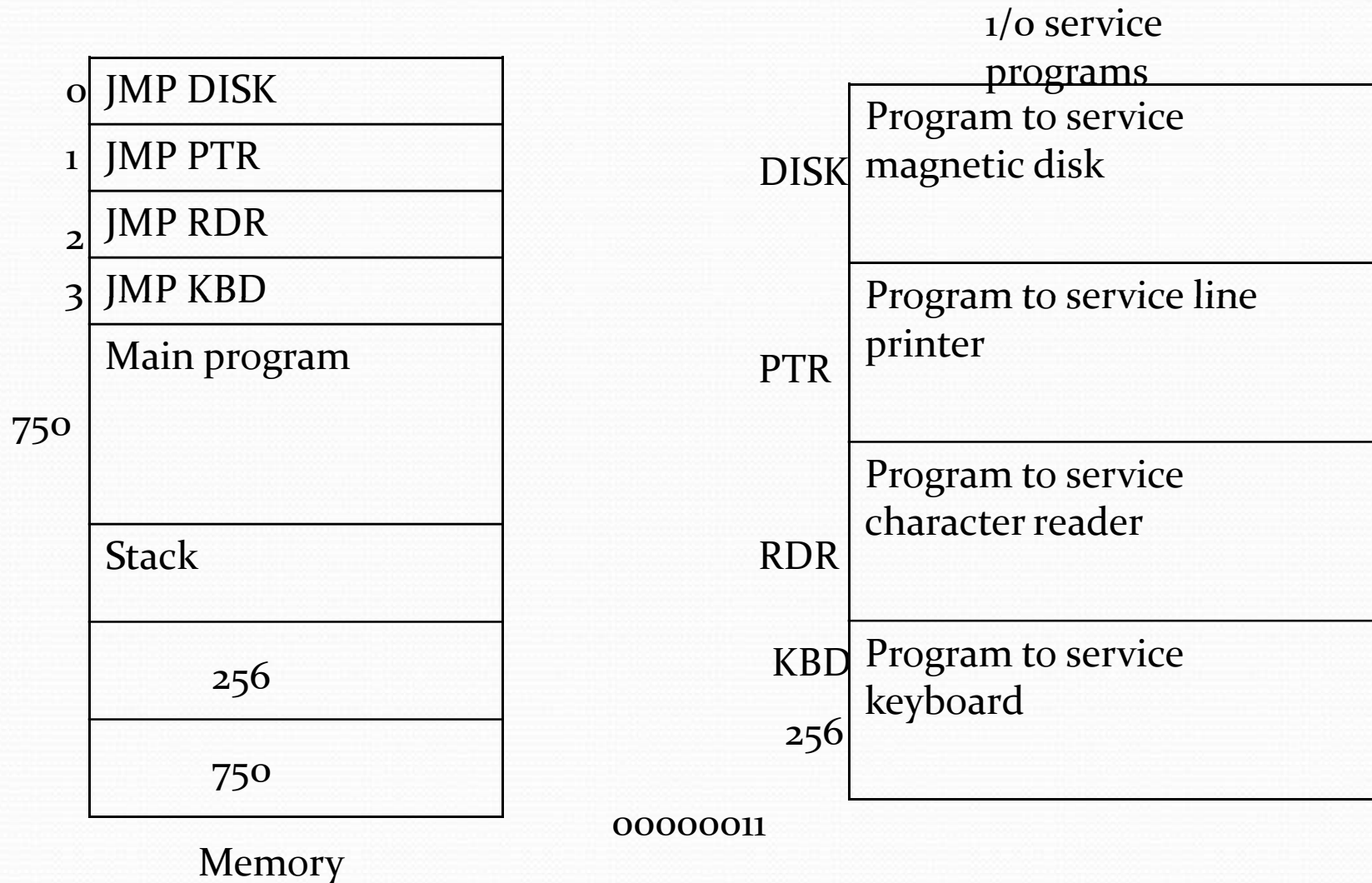
**Determines the highest priority interrupt when more than one interrupts take place**

### Priority Encoder Truth table

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	
0	0	0	1	1	1	1	
0	0	0	0	d	d	0	

$x = I_0' I_1'$   
 $y = I_0' I_1 + I_0' I_2'$   
 $(IST) = I_0 + I_1 + I_2 + I_3$

# Software Routines



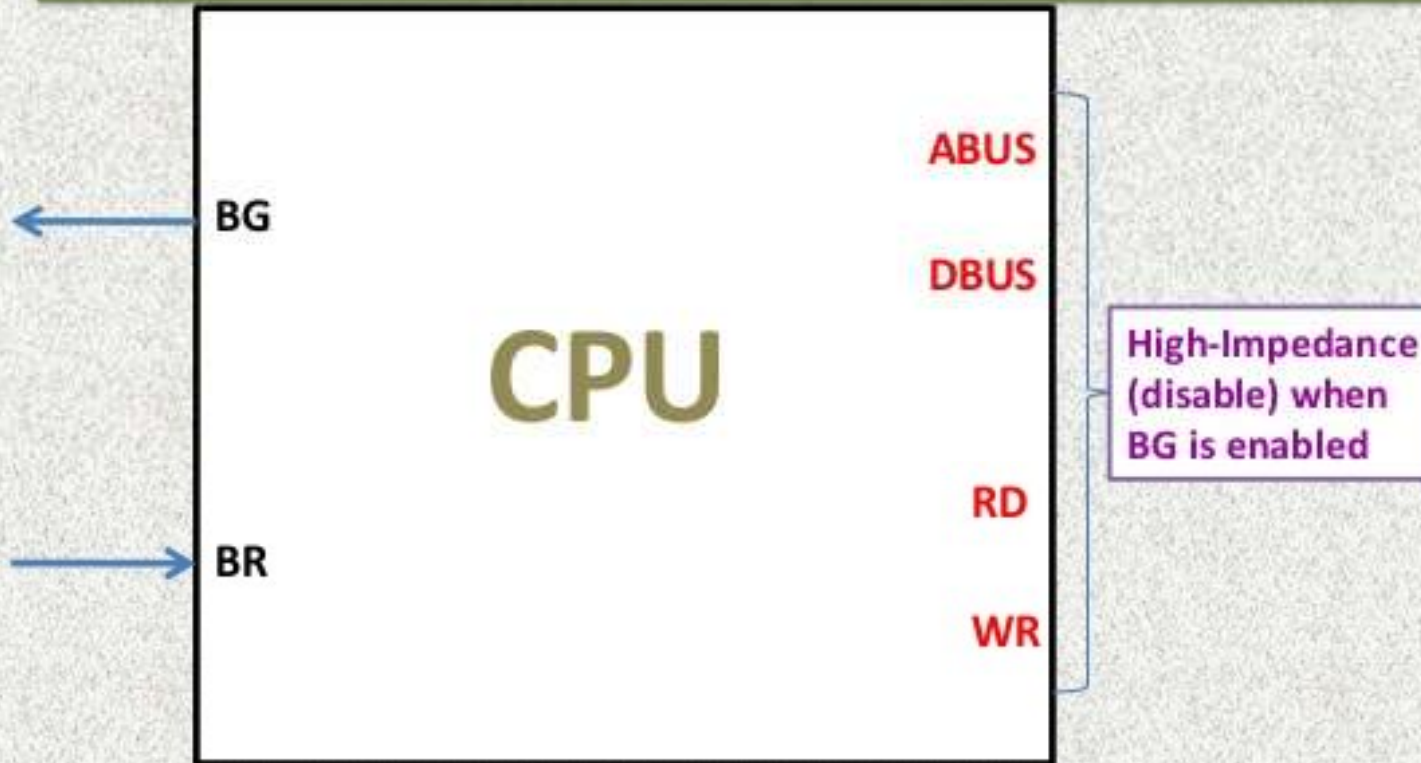


# Direct Memory Access (DMA)

- The data transfer between I/O devices and memory is limited by CPU.
- letting the peripheral device manage the memory buses directly.
- This technique is known as DMA.
- During DMA transfer CPU is idle & no control of memory bus.
- DMA control takes over the buses



# CPU bus signals for DMA transfer



BG → BUS GRANT

ABUS → ADDRESS BUS

RD → READ

BR → BUS REQUEST

DBUS → DATA BUS

WR → WRITE





# DMA

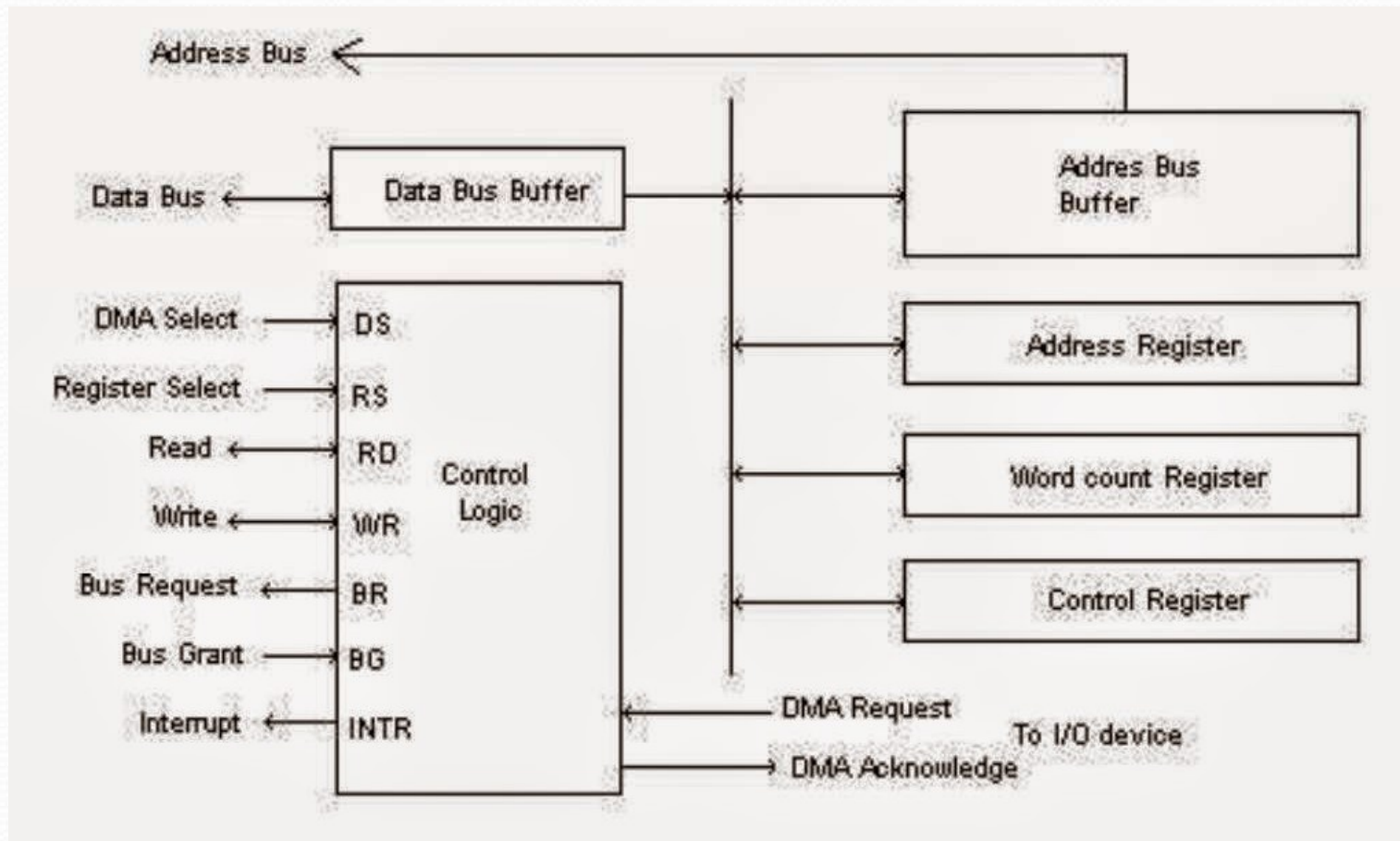
- The transfer can be made several ways.
- DMA burst transfer, : a block sequence consists of a number of memory words is transferred in continuous burst . – DISK
- Cycle stealing : transfer one word at a time .
- During this transfer CPU merely delays its operation for one memory cycle



# DMA Controller

- It needs the usual circuits of an interface to communicate with the CPU and I/O device
- It also needs address register, a word count register and set of address line
- Address register ,address lines are used for direct communication with the memory
- Word count register specifies the number of words that must be transferred

# Block Diagram of DMA controller







# DMA Controller

- DMA communicates with the CPU data bus and control lines
- CPU selects DMA & DMA registers by enabling DS, RS
- RD & WR are bidirectional
- Bus Grant (BG)=0 CPU communicates with DMA registers
- When BG=1 CPU relinquishes the Bus, then DMA communicates directly with memory
- DMA communicates with I/O via DMA request and DMA ACK





# DMA Controller

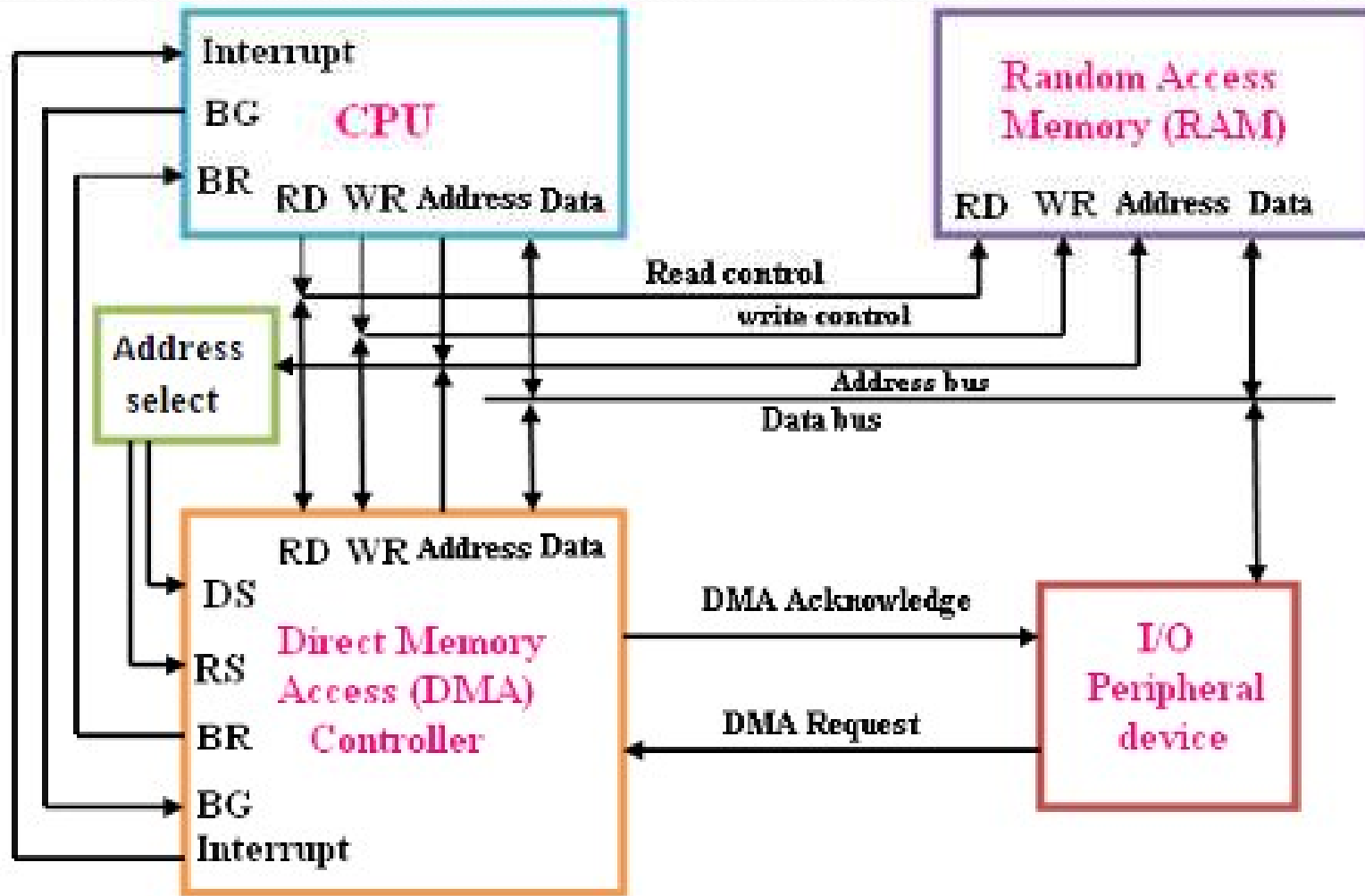
- Address register : desired memory location
- Address bits go via bus buffer to address bus.
- Address register increments after each word transferred to memory
- Word count register : number of words to be transferred
- It is decremented by one after each transfer
- Control register specifies the mode of transfer



# DMA Controller

- CPU initializes the DMA by sending
  1. Starting address of the memory
  2. Word count number of words in the memory bloc
  3. Control to specify mode of transfer(read or write)
  4. A control to start DMA transfer

# DMA transfer in a Computer System







# DMA Controller

- If the word count  $\neq 0$  DMA checks request line coming from the peripheral
- High speed device , line will be active after the previous transfer complete
- If it slower request will come after short time
- In this case DMA disables bus request line
  
- When I/O requests a transfer DMA requests bus again.



# DMA Controller

- If word count =0 ,transfer is complete then DMA enables interrupt signal
- DMA may have more than one channel for multiple I/o devices
- Each channel will have request, acknowledge ,address register ,word count register
- Priority among channel established in this case



# Input –Output processor(IOP)

- An IOP classified as a processor and directly communicates with I/O devices
- A processor that communicates with remote terminals over telephone and other communication media in a serial fashion is called a Data Communication Processor(DCP)
- IOP similar to CPU except it is designed to communicate only with I/O devices
- Unlike DMA IOP can fetch and execute its own instructions
- It can do Arithmetic, logical, branching & code translation

# Block diagram IOP

## SECTION 11-7 Input-Output Processor

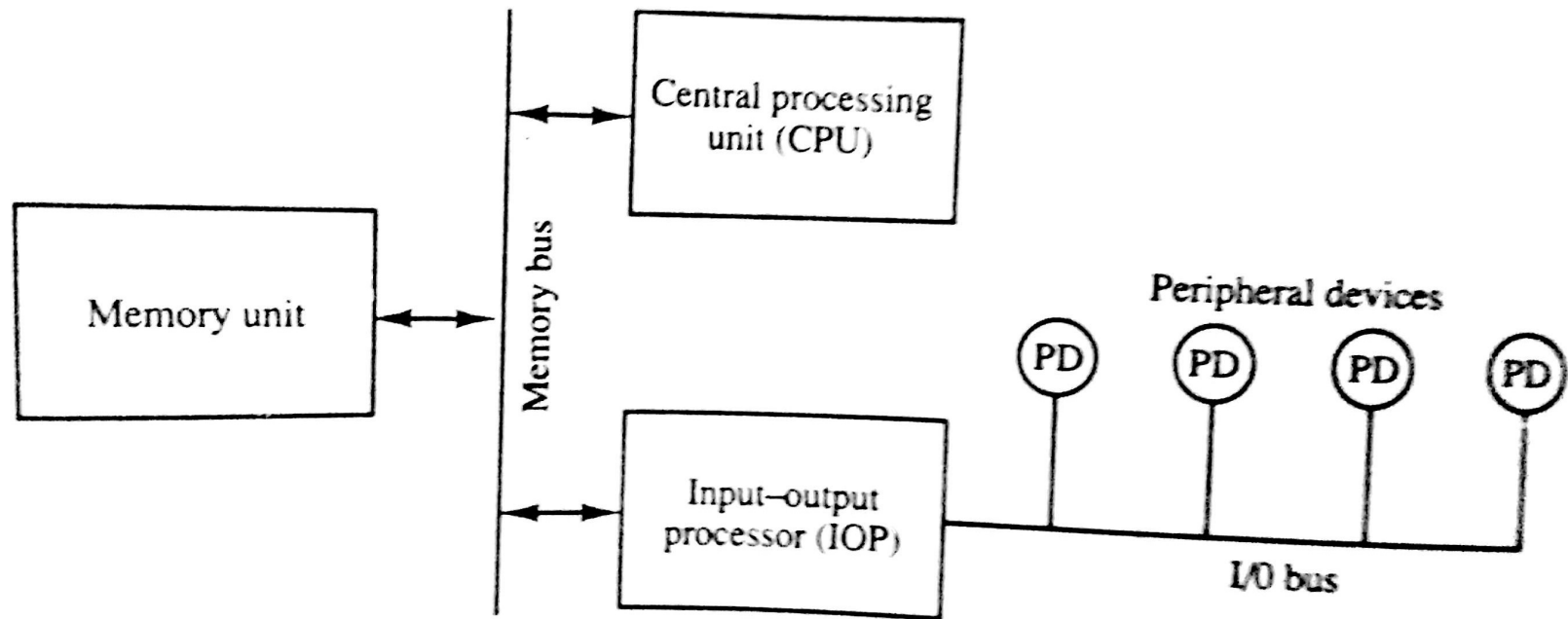


Figure 11-19 Block diagram of a computer with I/O processor.





# IOP

- The data formats of peripheral devices differ from memory and CPU data formats.
- it may be necessary to take four bytes from an input device and pack them into one 32-bit word before the transfer to memory.
- Data are gathered in the IOP at the device rate and bit capacity while the CPU is executing its own program.
- After the input data are assembled into a memory word, they are transferred from IOP directly into memory by "stealing" one memory cycle from the CPU



# IOP

- The communication between the IOP and the devices attached to it is similar to the program control method of transfer
- Communication with the memory is similar to the direct memory access method.
- In most computer systems, the CPU is the master while the IOP is a slave processor
- The CPU is assigned the task of initiating all operations, but I/O instructions are executed in the IOP.

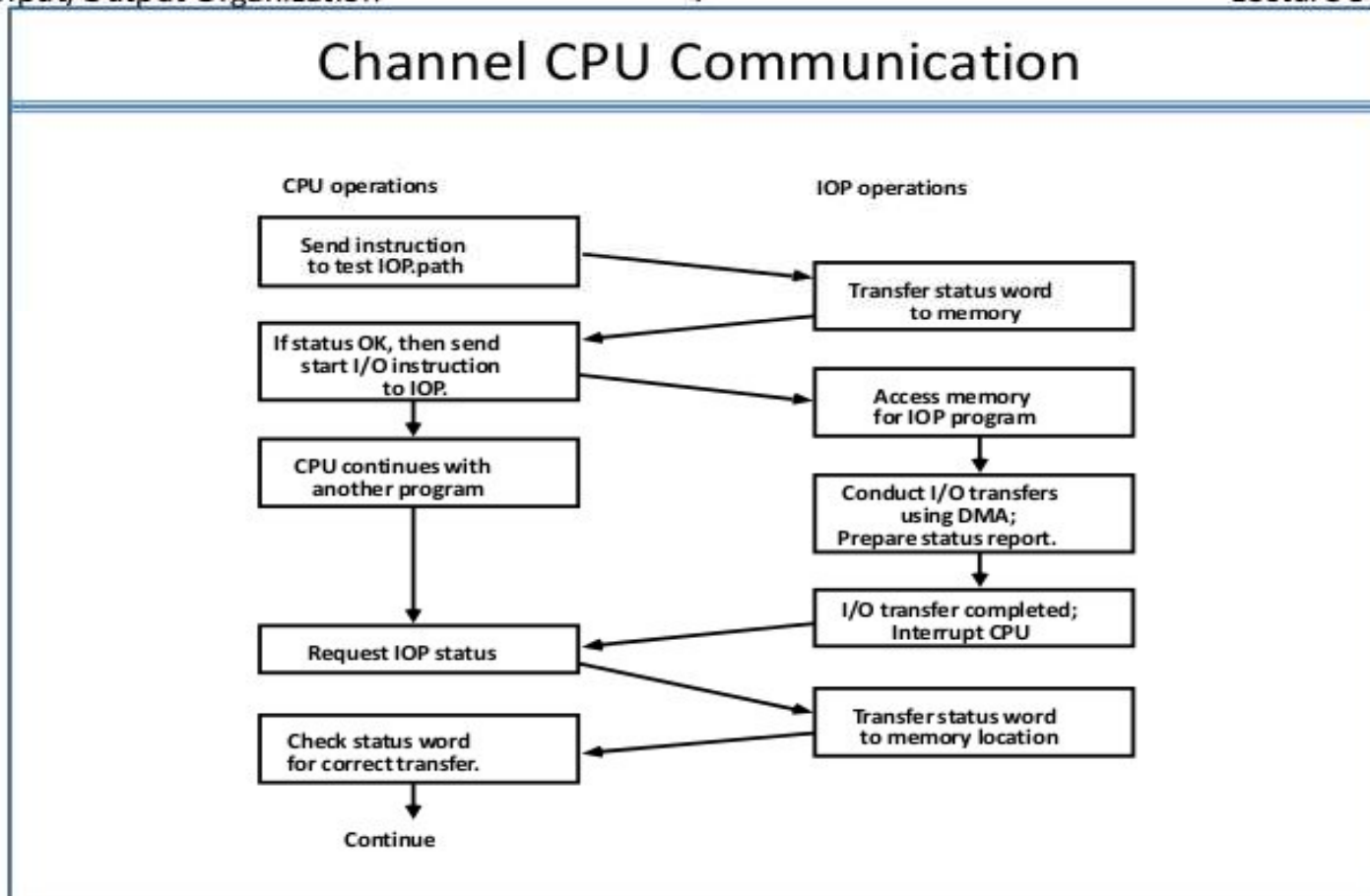


# IOP

- CPU instructions provide operations to start an I/O transfer
- The IOP, in turn, typically asks for CPU attention by means of an interrupt
- It also responds to CPU requests by placing a status word in a prescribed location in memory to be examined later by a CPU program
- Instructions that are read from memory by an IOP are sometimes called commands,



# CPU-IOP Communication







# Serial Communication

- A data communication processor is an I/O processor that distributes and collects data from many remote terminals connected through telephone and other communication lines. DCP
- It is a specialized VO processor designed to communicate directly with data communication networks
- A data communication processor communicates with each terminal through a single pair of wires



# Serial Communication

- The way the remote terminals are connected to a data communication processor is via telephone lines
- Modems needed
- Synchronous transmission does not use start –stop bits
- The modem has internal clock that are set to the frequency that bits are being transmitted
- The message consists of a group of bits transmitted sequentially as a block of data.
- Entire block transmitted with special control characters at beg and end



# Serial Communication

- DCP checks transmission errors
- Parity
- Data can be transmitted between two points in three different modes:
- Simplex : carries information in one direction only
- half-duplex: both directions but data can be transmitted in only one direction at a time
- full-duplex. A full-duplex transmission can send and receive data in both directions simultaneously

